Google Cloud NetApp Volumes を使用して Oracle Database 26ai の高可用性をデプロイします

NetApp database solutions

NetApp
June 12, 2026

目次

Google Cloud NetApp Volumes を使用して Oracle Database 26ai の高可用性をデプロイします

このガイドでは、コンピューティング インスタンスとストレージのプロビジョニング、Oracle Grid Infrastructure と Oracle Database のインストール、スタンバイ データベースの初期化、および Fast-Start Failover を使用した Oracle Data Guard の設定方法について説明します。

開始する前に

始める前に、以下のものを用意してください：

- Compute Engine、VPC ネットワーク、ファイアウォール構成、IAM、および NetApp Volumes の権限を持つ Google Cloud プロジェクト

Task	必要なアクセス
Compute Engine VM を作成する	Compute Instance Admin（または同等の資格）
ファイアウォール／ファイアウォールポリシー	ネットワーク管理者または委任されたポリシー管理者
GCNV プールとボリュームを作成する	NetApp Volumes管理者
PSAの設定	ホストプロジェクトのネットワーク管理者
IAP経由のSSH	IAPで保護されたトンネルユーザー + OSログイン（使用する場合）

- NetApp Volumes APIが有効になっている
- 対象リージョン用に構成された VPC とサブネット
- Google Cloud NetApp Volumes 用に構成された Private Services Access（PSA）
- 必要なすべての仮想マシンに Oracle Linux 10 を使用
- データベースホストとオブザーバーホストに対して、DNSとホスト名解決が設定されています。
- Oracle Database 26aiおよびGrid Infrastructure用のOracleインストールメディアとパッチファイルが利用可能
- Oracle Data Guard、Oracle Restart、およびiSCSIストレージの概念に関する知識
- すべての仮想マシンで時刻同期が設定されています。

  以下のコマンドを使用できます：

```
gcloud services enable netapp.googleapis.com
chronyc tracking
timedatectl
```

このガイドで使用する構成例

このガイドでは、以下の導入前提条件を使用します：

- 3台のGoogle Compute Engine仮想マシン：
  - `oracdb1`プライマリデータベースの場合
  - `oracdb2`スタンバイデータベース用
  - Fast-Start Failover Observer用の`oradg-obs`
- データベース ゾーンごとに 1 つの GCNV Flex Unified ストレージ プール
- データベース ホストごとに 5 つの GCNV iSCSI ボリューム
- Oracle Data Guard BrokerおよびFast-Start Failoverによる自動フェイルオーバー
- データベース ホストごとに専用ストレージが割り当てられ、プライマリ ホストとスタンバイ ホストは iSCSI ボリュームを共有しません

コマンド内の例の値を、ホスト名、IP アドレス、ゾーン、プロジェクト名、ポータル IP、パスワード、Oracle メディアファイル名など、ご使用の環境の値に置き換えてください。

目的

この手順では、以下のタスクを実行します：

- プライマリデータベース、スタンバイデータベース、およびオブザーバー用の Compute Engine インスタンスをプロビジョニングします
- Oracle 向けに GCNV iSCSI ストレージとマルチパス デバイスを設定する
- 両方のデータベースホストにOracle Grid InfrastructureとOracle Database 26aiをインストールします
- プライマリOracleデータベースを作成する
- Google Cloud NetApp Volumes 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スタンバイデータベースを初期化します。
- Oracle Data Guard Broker、Fast-Start Failover、およびObserverを設定する

導入オプション

このセクションでは、Google Compute Engine（GCE）と Google Cloud NetApp Volumes（GCNV）iSCSI ブロック ストレージを使用した Oracle Database の実用的な HA/DR デプロイメント パターンを比較します。また、GCNV レプリケーションがスタンバイの初期化をいかに高速化するかについても強調しています。始める前に、このマトリックスを使用してティアを選択してください。このガイドでは、Prod HA（Data Guard + FSFO）を実装します（一番下の行）。

環境	要件	推奨されるアーキテクチャ	**HA**	**DR**	**Automation**	主なメリット
開発/テスト	最低コスト	単一インスタンス	×	はい	×	Snapshot クローン

環境	要件	推奨されるアーキテクチャ	**HA**	**DR**	**Automation**	主なメリット
Prod Basic（再起動）	クラッシュによるダウンタイムを削減	+ Oracle Restart	×	はい	ローカルのみ	自動再起動
本番環境HA（DGなし）	手動 DR が許容されます	+ Snapshot / RMAN	Partial	はい	Partial	GCNV クローンのリカバリ
本番環境HA（DG + FSFO）	真のHA（DBA不要）	Data Guard + FSFO	はい	はい	フル	真のHAと高速フェイルオーバー

HA / DR / 自動化： はい = 階層の目標を満たしています；いいえ = 対象外です；部分的 = ストレージ レベルまたは手動の手順のみ。

お客様の階層に合わせて読むべきセクション

達成したいHAのレベルに応じて、以下のマトリックスの各セクションをお読みください。例えば、Data GuardとFSFOを使用した本番環境の高可用性（HA）を実現したい場合は、すべてのセクションをお読みください。Dev/TestまたはProd Basic with Restartが必要な場合は、最初の列のみをお読みください。

開発/テスト環境または本番環境の基本設定（再起動）	本番環境**HA**（**Data Guard**なし）	本番環境の高可用性（**Data Guard + FSFO**）
開始する前に このガイドで使用する構成例 目的 導入オプション 概要 アーキテクチャ Compute Engine のプロビジョニング GCNV iSCSI ボリュームをプロビジョニングする Oracleソフトウェアをインストールする プライマリデータベースを作成する	開始する前に このガイドで使用する構成例 目的 導入オプション 概要 アーキテクチャ Compute Engine のプロビジョニング GCNV iSCSI ボリュームをプロビジョニングする Oracleソフトウェアをインストールする プライマリデータベースを作成する スタンバイデータベースを作成する（ステップ3：GCNVスタンバイの初期化経由のみ）	開始する前に このガイドで使用する構成例 目的 導入オプション 概要 アーキテクチャ Compute Engine のプロビジョニング GCNV iSCSI ボリュームをプロビジョニングする Oracleソフトウェアをインストールする プライマリデータベースを作成する スタンバイデータベースを作成する Data Guard Broker、FSFO、およびObserverを構成する

概要

このアーキテクチャは、Google Cloud NetApp Volumes（GCNV）iSCSI ストレージと Oracle Data Guard を使用して、Google Cloud 上に Oracle Database 26ai の高可用性をデプロイします。GCNV は、ハイパフォーマンス ブロック ストレージを提供し、ストレージ レイヤでのスタンバイ初期化をサポートします。Data Guard は、継続的な同期、切り替え、および高速起動フェイルオーバー機能を提供します。

レイヤ	ロール
GCNV	ブロック ストレージ、スタンバイ初期化、ストレージ レプリケーション
Data Guard	継続的な同期、REDO の適用、スイッチオーバー、FSFO

GCNV レプリケーションは、Oracle データベース チャネルを経由するのではなく、ストレージ レイヤでデータを移動することで、RMAN アクティブ複製と比較して、スタンバイ初期化時間を大幅に短縮します。環境が対応している場合は、本番環境のスタンバイ シードには GCNV レプリケーションを優先してください。

コンポーネント	詳細
DBホスト	`oracdb1` / `oracdb2` — 異なるゾーン、それぞれ5つのGCNV iSCSIボリューム
ストレージ	`/u01` + ASM `+DATA`, `+RECO`, `+FRA` on GCNV（EXTERNAL）
スタンバイ初期化	GCNVのレプリケート/スナップショット/クローン → Oracleのファイナライズ → Data Guard
HA	再起動、物理スタンバイ（MOUNTED）、Broker、FSFO、Observer (`oradg-obs`)
アプリ	サービス `orclapp`

ゾーニング: データベースゾーンごとに 1 つの GCNV Flex Unified プール。ホストごとに専用ボリューム。ブートディスクは OS のみです。対象外: TDE、RAC、NVMe/TCP、OEM、Active Data Guard（スタンバイは **MOUNTED** 状態のまま）。

アーキテクチャ

参考図

このアーキテクチャは、3つの独立したデータパスを提供する。ストレージレプリケーションとData Guardのリドゥ転送は、別々の経路で行われます。

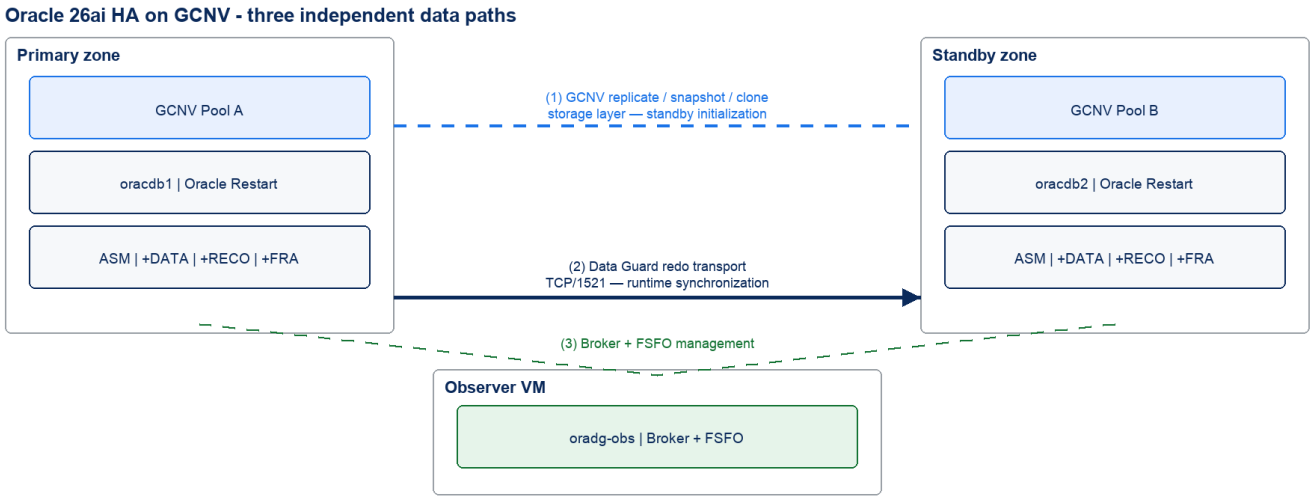

図 **1. GCNV Oracle** デプロイメント - リファレンス アーキテクチャ

*ストレージレプリケーション≠リドゥレプリケーション。*パス1は、スタンバイ初期化のためにGCNVレイヤーでデータファイルを移動します。パス2では、切り替え後もデータベースの同期が維持されます。パス3は、Oracle Data Guard BrokerとObserverを介して、ロールの移行と自動フェイルオーバーをオーケストレー

ションします。

プラットフォームロール

プラットフォーム	提供
GCNV	iSCSI ボリューム、Snapshot、ボリュームレプリケーション（ベースライン+増分）— スタンバイ初期化
Data Guard	Redo トランスポート、MRP、Broker、FSFO — 継続的な同期とフェイルオーバー

GCNV レプリケーションは、まずベースラインコピーを実行し、その後、ポリシーごとにブロックの増分更新を実行します。スタンバイが初期化された後、Data Guard はリドゥ適用と FSFO を所有します。

トポロジとストレージ

ロール	**VM**	ゾーン	**GCNV**プール	ボリューム（ホストあたり）
プライマリ	`oracdb1`	`us-west1-a`	`oracle-pool-a`	`ora_<host>_u01`、`ora_<host>_data_01/02`、`ora_<host>_arch_01`、`ora_<host>_fra_01`
Standby	`oracdb2`	`us-west1-b`	`oracle-pool-b`	同じ命名パターン
オブザーバー	`oradg-obs`	`us-west1-c`	—	ブートディスクのみ

ストレージ	バッキング	用途
OS	GCE ブート ディスク	OSのみ
`/u01`	GCNV iSCSI	グリッド/DBホーム、ステージング（XFS）
`+DATA` / `+RECO` / `+FRA`	GCNV iSCSI	ASM EXTERNAL — データファイル、アーカイブ、FRA

マシンタイプ

このガイドのサンプル `n4-highmem-8`。OLTPワークロードは通常 `c4-standard-*`を使用します。

Compute Engine のプロビジョニング

ステップ**1**：**VM**を作成する

同一リージョン内の異なるゾーンに3つの仮想マシンを作成します（ゾーン障害分離）。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`）。Cloud Console、 `gcloud`、Terraform、または標準的なプロビジョニングワークフローを使用します。

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）

ネットワークティア： レイテンシまたは出力（>~200 GiB/月）が重要な場合はプレミアム、開発とテストで TCO を抑える場合はスタンダード。

3台すべてで*セキュアブート*、**vTPM**、および*整合性監視*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）— `/u01`用に別のGCEデータディスクを接続*しないでください*。

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

3つのVM `/32`すべての内部IP間で、各ゾーン(`us-west1-a/b/c`でTCP/1521を許可します）。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。ルールが欠落していると、REDOトランスポートとObserver接続が中断されます。

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。各 VM から検証：

```
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

ポート	想定	説明
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、 /etc/hosts

各VMの起動後、ホスト名を設定し、 `/etc/hosts`エントリを3つのホストすべてに追加して、Oracle Net、ブローカー、およびObserverの正引き/逆引き名前解決が機能するようにします。

```
# 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
```

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）に置き換えてください。

各ホストから検証する：

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

ステップ4：OSベースライン（DBホストのみ）

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）。

`oracdb1`および `oracdb2`で実行（Observerのセットアップについては<<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）。

```
# 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

# iSCSI, multipath, OUI JDK
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
```

*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 DB ホストの 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。

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

`oracdb2`で繰り返します。各ホストのIQNを記録します。ホストごとに1つのホストグループを作成することで、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ることがなくなります。

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）。

GCNV iSCSI ボリュームをプロビジョニングする

ステップ1：データベースゾーンごとに1つのGCNV Flex Unified iSCSIストレージプールを作成します

データベースゾーンごとに 1 つの Flex Unified プール（アーキテクチャを参照）。

このHA設計用に2つのプールを作成します（各ゾーンごとに手順を繰り返します）：

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

`READY`を待ちます。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調整します（ <<create-gcnv-iscsi-volumes,ステップ3：GCNV iSCSIボリュームを作成する >>のサイズは例です）。

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（DBホストごとに1つ）

データベースホストごとにホストグループを1つ作成し、各VMが自身のボリュームのみを参照できるようにします。プライマリとスタンバイはGCNV iSCSIボリュームを共有してはなりません。ObserverにはGCNVリソースがありません。Cloud Consoleで：

1. **NetApp Volumes** → ホストグループ → 作成
2. 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
3. ホスト： `oracdb1`からIQNを貼り付けます（ `/etc/iscsi/initiatorname.iscsi`の値）
4. 概要： "Oracle primary host oracdb1" · **Create**
5. `oracdb2-hg`と `oracdb2`のIQNについて繰り返します

ステップ3：GCNV iSCSIボリュームを作成する（データベースホストごと）

各データベースホストは、ゾーンのプールに 5 つの GCNV iSCSI ボリュームを取得します。1 つは `/u01`用で、4 つは ASM バッキングデバイス用です：

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。

オン `oracdb1`： `oracle-pool-a`に 5 つのボリュームすべてを作成し、ホスト グループ `oracdb1-hg`を作成します。

オン `oracdb2`： `oracle-pool-b`に 5 つのボリュームすべてを作成し、ホスト グループ `oracdb2-hg`を作成します。

NetApp Volumes → **Volumes** → **Create** — iSCSI、正しいプールとホストグループ、Linux。プールごとの記録：

- 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：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

`oracdb1`でそのホストのプール ポータルを使用して実行し、次に `oracdb2`でスタンバイプール ポータルを使用して実行します。

ホストの送信が制限されている場合、各データベース VM からその 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
```

1. 設定する： `device-mapper-multipath`

```
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
```

1. 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ます。次に：

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

ステップ5：Google Cloud NetApp Volumes iSCSIボリューム上にASMバッキング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`ではない）をパーティション分割し、それぞれに 1 つの GPT パーティションを作成します。ASM は raw パーティションを使用します。各ホストで実行します。以降のすべてのブロックでは、 `HOST=$(hostname -s)`を使用してローカル ホストのデバイスを自動的に選択します。

```
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

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：ローカルGCNV iSCSIボリューム上で`/u01`をフォーマットしてマウントする

`ora_<host>_u01`GCNV iSCSIボリュームには、Gridホーム、Oracleホーム、およびステージング環境が格納されます。マルチパスデバイス（ASM用にパーティション分割されていないデバイス）にXFSをフォーマットします。 `/etc/fstab`でUUIDを使用します（共有ファイルシステムラベルではありません）：

```
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")

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

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ソフトウェアをインストールする の前に `/u01` がマウントされることを確認します。

Oracleソフトウェアをインストールする

プライマリデータベースを作成するの前に、各DBホストでGridとデータベースホームのインストールを実行します。

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

このセクション全体を `oracdb1`で実行し、 `oracdb2`で繰り返します。両方のホストはそれぞれ独自の Grid ホーム、 ASM インスタンス、およびディスク グループを取得します。 Data Guard はストレージではなく、 Oracle Net を介してレプリケーションを行います。

Oracleバイナリを `**/u01**`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.
```

Grid ホームをその場で解凍します

26ai Grid GoldImageは、ターゲットのGridホームに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 RU レベル*このガイドでは、Grid GoldImage に検証済みの RU がすでに含まれていることを前提としています。Grid GoldImage がターゲット RU よりも古い場合は、Oracle のドキュメントに記載されている `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 がバンドルされた Grid GoldImage を使用してください。Grid ホームと Database ホームは同じパッチレベルに維持してください。

単発 `gridSetup`：完全な**HA_CONFIG**応答ファイル

各ホスト上でビルド `/tmp/grid.rsp`(`responseFileVersion`は必須です。置換 `HOST`して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_respo
nse_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_02
p1
managementOption=NONE
RSP"
sudo -u grid chmod 600 /tmp/grid.rsp
```

`gridSetup`を実行して、バイナリをコピーし、設定をステージングします：

```
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
'
```

`Successfully Setup Software with warning(s).`を想定し、終了コードは6（警告）または0です。

`orainstRoot.sh`と`root.sh`をrootとして実行

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

`root.sh`は `crsctl/ srvctl/ `asmcmd`ラッパーを作成し、OHAS を起動します。

`gridSetup.sh -executeConfigTools` **— ASM** を起動して作成します `+DATA`

の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：

```
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
'
```

NETCA / ASMCA / CVU の後に `Successfully Configured Software.`が表示されます。

`+RECO`および`+FRA`ディスク グループを作成する

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。他の2つは `asmca`から作成します：

```
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
"
```

ASMとOracle Restartを確認する

```
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.
```

`oracdb2`でこのセクションを繰り返します。およびの `HOST=$(hostname -s)`パターンは、そのホストのGCNV iSCSIデバイスを自動的に選択します。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 26aiをインストールする

このセクションを `oracdb1`で実行し、次に `oracdb2`で実行します。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。

DB ホームと **RU** パッチを解凍します

```
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
```

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。

RU を適用したソフトウェアのみのサイレント インストール

```
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
'
```

OL 8/9では、 `runInstaller`行から `-applyOneOffs`を省略します。

`root`として、インストール後のスクリプトを実行します：

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

Oracle環境を設定する

各 DB ホスト（(`orcl` 上 `oracdb1`、 `orcls` 上 `oracdb2`）：

```
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
```

（`ORACLE_SID=orcls`をスタンバイホスト上で使用します。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。）

プライマリデータベースを作成する（(`oracdb1`のみ）

ASM ディスク グループに対してサイレント モードで `dbca`を実行します：

```
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
'
```

アーカイブログをポイントする `+RECO`（スタンバイは、ステップ2：スタンバイ：最小限のinit.ora pfile、パスワードファイル、NOMOUNTで一致する設定を使用します）：

```
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
'
```

Oracle Restartでデータベースが起動していることを確認してください：

```
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
```

*ロールベースのアプリケーション サービスを作成します（Oracle Restart）。*アプリケーションは、DB名ではなく `orclapp`経由で接続する必要があります。これにより、フェイルオーバーが透過的になります。

```
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
'
```

Broker有効化後、 `orclapp` はPRIMARYでのみ実行されます。ASMディスク グループ全体で制御ファイルを多重化し、ワークロードに合わせてメモリのサイズを設定します（このガイドでは例として4 GB / 3 GB SGAを使用しています）。

スタンバイデータベースを作成する

`orcls`上に構築 `oracdb2`（ `oracle-pool-b`上の専用ボリューム）。プライマリとスタンバイの初期セットアップを完了してから、<<gcnv-standby-initialization,ステップ3：GCNVスタンバイの初期化>>、スタンバイ完了タスク、および<<broker-fsfo-observer,Data Guard Broker、FSFO、およびObserverを構成する>>を実行します。ターゲット：プライマリ READ WRITE；スタンバイ PHYSICAL STANDBY、MOUNTED、MRP 適用中。

ステップ1：プライマリ：SYSパスワード、パスワード ファイル、およびDGパラメータ

`oracdb1`の `oracle`：

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

オン oracdb2：

```
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
```

各ホストで Oracle Restart を使用して再起動します：

```
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：最小限のinit.ora pfile、パスワードファイル、NOMOUNT

プライマリパスワードファイルをスタンバイ（IAP `gcloud compute scp`）にコピーします：

```
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
```

`oracdb2`で、所有権を設定し、スタンバイpfileを作成します。*プライマリ*で、最初に `*.compatible`をコピーします：

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

`oracdb2`で、以下のブロック内の `<COPY_FROM_PRIMARY>`をその値に置き換えます：

```
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
'
```

ステップ3：GCNVスタンバイの初期化までデータファイルなしでNOMOUNT。

ステップ3：GCNVスタンバイの初期化

`**oracle-pool-b**`のスタンバイ**ASM**ボリュームに、***Google Cloud NetApp Volumes**レプリケーション（SnapMirrorベース、デフォルトモード）、ボリューム**Snapshot**、または*クローン*を使用してデータを取り込み、Oracle の最終処理を実行します。iSCSIとASMはGCNV iSCSI ボリュームをプロビジョニングすると同じです。

API	用途
`gcloud netapp storage-pools`	**Default-mode** Flex Unified プールを作成します(`--mode=default`)
`gcloud netapp volumes`	iSCSI ボリューム、ホスト グループ、Snapshot
`gcloud netapp volumes replications`	クロスロケーション*ボリューム レプリケーション*

手順	アクション
1	プライマリアーカイブログ + 強制ログ記録（ステップ1：プライマリ：SYSパスワード、パスワード ファイル、およびDGパラメータ）
2	静止： `BEGIN BACKUP`SCN を記録、スタンバイ制御ファイル
3	ボリュームレプリケーション（ボリュームレプリケーションを作成するおよびカットオーバー — 休止、レプリケーションの停止、スタンバイ LUN の接続）またはスナップショット（代替案：スナップショットシード）
4	`oracdb2`：iSCSI、マルチパス、ASMマウント（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5：Google Cloud NetApp Volumes iSCSIボリューム上にASMバッキングデバイスをパーティション分割する、）
5	Oracle finalize — SCN にリカバリ、**MOUNTED** (Oracle の最終処理)
6	SRL 再構築（ステップ4：スタンバイ REDO ログファイル）、MRP、ブローカー（Data Guard Broker、FSFO、およびObserverを構成する）

RMANのアクティブ重複機能は、小規模なラボでは引き続き有効です。本番環境のスタンバイシードには、*GCNVレプリケーション*が推奨されます。

前提条件

- `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。

```
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>
```

ボリュームレプリケーションを作成する

```
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* / 初期同期が完了するまで待ちます。

カットオーバー — 休止、レプリケーションの停止、スタンバイ **LUN** の接続

プライマリ：

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

最終レプリケーション サイクルを許可してから、次の手順を実行します。

```
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可能性があります — 更新時に `name=oracdb1_data_lun`を使用）：

```
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"
```

コントロールファイルを `oracdb2`にコピーし、プライマリで次を実行します：

```
ALTER DATABASE END BACKUP;
```

代替案：スナップショットシード

ワンタイム シード：ソース ボリューム上のスナップショット → スタンバイ プールのスナップショットからボリュームを作成（Cloud ConsoleまたはAPI）。 `oracdb2-hg`にアタッチした後、Oracle の最終処理に進みます。

スタンバイ **iSCSI** および **ASM** （ **RMAN** 導入前）

`oracdb2`で、*スタンバイプール* iSCSIポータルにログインします。
ASMディスクヘッダーが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マルチパスエイリアス*を使用します（lab: `ora_oracdb1_data_01`、
`ora_oracdb1_arch_01`）、パーティションに
`asm_diskstring='/dev/mapper/ora_oracdb1_*p*'`、 `chown
grid:asmadmin`を設定してから、次の操作を行います：

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

Oracle の最終処理

```
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;
```

変換後、スタンバイ **REDO** ログを再構築（複製された制御ファイルにはまだ `+DATA/ORCL/…` SRL パスが含まれています — プライマリで `ORA-19527/ `ORA-16086`が発生します）。ステップ4：スタンバイ REDO ログファイルを参照してください。

ステップ**4**：スタンバイ **REDO** ログファイル

FSFO の*両方の*ホストで必要です。サイズ ≥ 最大のプライマリ オンライン REDO（カウント = （スレッドあたりのオンライン グループ） + 1）。

GCNV シード後： スタンバイ上のすべてのスタンバイログファイルグループを削除し、 `+DATA`のみで再作成（`db_create_file_dest='+DATA'`）。 `+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`の原因となります。

プライマリ(**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
```

スタンバイ(**orcls**）**GCNV** シード後：

```
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5：スタンバイでフラッシュバックを有効にし、マネージドリカバリを開始します

管理リカバリを開始する*前に*フラッシュバックを有効にしてください（MRPがアクティブな間はフラッシュバックを有効にできません）。

```
# 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;
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6：プライマリで**REDO**転送を有効にする

スタンバイの作成中に継続的な `ORA-12154`エラーを抑制するために、ステップ2：スタンバイ：最小限のinit.ora pfile、パスワードファイル、NOMOUNTで`LOG_ARCHIVE_DEST_2`は意図的に `DEFER`に設定されていました。スタンバイの準備ができたので、これを有効にします。

`LOG_ARCHIVE_DEST_STATE_2`に切り替え
`ENABLE`、ログスイッチを強制して、最初のラウンドのREDOがすぐに送信されるようにします：

```
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;
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。<<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7：ターゲットのData Guard状態を確認する

プライマリ (oracdb1）：

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

スタンバイ(oracdb2（Cloud Console **SSH**、またはワークステーションからのIAP）：

```
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）。

スタンバイがレポート `MOUNTED`しているが、適用が実行されていない場合は、`MRP` を再起動してください `oracdb2`：

```
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'
```

ステップ8：スタンバイデータベースをOracle Restartに登録する

GCNV シーディング後にスタンバイを Restart に登録すると、再起動時に ASM、MOUNT、および適用が回復されます。

`oracdb2`で、`spfile` の場所を取得し、`Oracle Restart` に登録します（クエリから `<STANDBY_SPFILE_PATH>`を代入します。多くの場合 `+DATA`にあります）：

```
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
'
```

`oracdb1`で、プライマリOracle Restartデータベース リソースにASMディスクグループの依存関係がリストされていることを確認します。DBCAが `DATA`と`FRA`のみを登録した場合は、 `RECO`を追加します：

```
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 config database -db orcl
srvctl modify database -db orcl -diskgroup DATA,RECO,FRA
srvctl config database -db orcl
'
```

スタンバイ データベース リソースに同じアプリケーション サービスを追加する (orcls on oracdb2）。`role PRIMARY`を両側で使用して、 `orclapp`がスイッチオーバー後に使用できるようにします：

```
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 orcls \
  -service orclapp \
  -pdb orclpdb \
  -role PRIMARY \
  -policy AUTOMATIC

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

`oracdb2`で、スタンバイデータベースリソースを確認します：

```
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 status database -db orcls
'
```

Data Guard Broker、FSFO、およびObserverを構成する

`ENABLE CONFIGURATION`後、*DGMGRL*
を使用してトランスポートとロールを管理します（アドホック `LOG_ARCHIVE_DEST_*` SQL ではありません）。

手順1：両方のデータベースでブローカーを有効にする

*プライマリ*および*スタンバイ*のデータベースホスト上：

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

*プライマリデータベースホスト*で、OS認証を使用して接続します（ステップ5：ObserverをsystemdユニットとしてのObserverウォレットはDBホストでは必要ありません）：

```
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 /
'
```

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。

```
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.
```

実行 `VALIDATE DATABASE`即座に — スイッチオーバーを試みる前に、孤立したデータファイル、欠落したSRL、実行されていないredo apply、その他の一般的な問題を明らかにします。任意の `WARNING`またはNULL以外の `ERROR`は、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する必要があります。

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

ステップ2：フラッシュバックの確認（FSFO自動復旧に必要）

FSFO を有効にする前に、*両方の*ホストで `flashback_on`確認してください：

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

*プライマリ*のみで、保持期間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プロパティを設定して有効にする

```
-- Transport mode and protection mode
DGMGRL> EDIT DATABASE 'orcl'  SET PROPERTY LogXptMode='SYNC';
DGMGRL> EDIT DATABASE 'orcls' SET PROPERTY LogXptMode='SYNC';
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;

-- FSFO targets (each side names the other)
DGMGRL> EDIT DATABASE 'orcl'  SET PROPERTY FastStartFailoverTarget =
'orcls';
DGMGRL> EDIT DATABASE 'orcls' SET PROPERTY FastStartFailoverTarget =
'orcl';

-- 30 s = 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;

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：オブザーバーホストにOracle Instant Clientをインストールする

```
# On oradg-obs, as root (use -el8 / -el9 if the Observer is on an older
OL/RHEL)
sudo dnf install -y oracle-instantclient-release-el10
sudo dnf install -y oracle-instantclient-basic \
                    oracle-instantclient-sqlplus \
                    oracle-instantclient-tools

# Dedicated 'oracle' OS user (owns the wallet and the systemd unit)
sudo useradd -u 54321 -m oracle
sudo passwd -l oracle

# Oracle environment for the user
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ユニットとしてStartする

認証情報は自動ログインウォレットに保存されます。 `dgmgrl`コマンドライン（ `ps` / `journalctl`から見える）には保存されません。Observerでのみ `/@<tns_alias>`と接続します。

- 専用アカウントを使用する： `SYS`ではなく、専用のData Guard管理アカウント（たとえば、 `SYSDG`）のクレデンシャルをウォレットに保存します。
- 自動ログインウォレットが必要： Observer systemdサービスには自動ログインウォレット（`cwallet.sso`）が必要です。 `cwallet.sso`の実行後に `mkstore`が欠落している場合は、Instant Client **tools** パッケージまたはデータベース ホームから `orapki`を使用して自動ログイン ウォレットを作成し、保存済みの認証情報を再度追加します。

1. オブザーバー上に、両方のメンバーの認証情報を使用してウォレットを作成します：

```
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;'
```

```
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
```

1. Systemd ユニット + ログ ローテーション：

```
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
```

プライマリから確認 — Observerは `CONNECTED`を読む必要があります（ `DISCONNECTED` Observer はFSFOをサイレントに中断します）：

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

ステップ6：FSFO（スイッチオーバーとフェイルオーバー）のテスト

計画的スイッチオーバー：

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

計画外フェイルオーバー： テストウィンドウでVM リセット（クラッシュスタイルのテスト）を使用します。 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`を `oradg-obs`で追跡し、完了したらトポロジを復元します。

著作権に関する情報

商標に関する情報