使用 Google Cloud NetApp Volumes 部署 Oracle Database 26ai 高可用性

NetApp database solutions

NetApp
June 12, 2026

目录

使用 Google Cloud NetApp Volumes 部署 Oracle Database 26ai 高可用性

本指南演示如何配置计算实例和存储、安装 Oracle Grid Infrastructure 和 Oracle Database、初始化备用数据库以及使用快速启动故障转移配置 Oracle Data Guard。

开始之前

开始之前，请确保您已具备以下条件：

- 具有 Compute Engine、VPC 网络、防火墙配置、IAM 和 NetApp Volumes 权限的 Google Cloud 项目

任务	所需访问权限
创建 Compute Engine VM	计算实例管理员（或同等功能）
防火墙 / 防火墙策略	网络管理员或委派策略管理员
创建 GCNV 池和卷	NetApp Volumes 管理员
配置 PSA	主机项目中的网络管理员
通过 IAP 进行 SSH	IAP 保护的隧道用户 + OS Login（如果使用）

- 已启用 NetApp Volumes API
- 为目标区域配置的 VPC 和子网
- 为 Google Cloud NetApp Volumes 配置的 Private Services Access (PSA)
- 适用于所有必需虚拟机的 Oracle Linux 10
- 为数据库主机和 Observer 主机配置 DNS 和主机名解析
- 适用于 Oracle Database 26ai 和 Grid Infrastructure 的 Oracle 安装媒体和修补程序文件
- 熟悉 Oracle Data Guard、Oracle Restart 和 iSCSI 存储概念
- 为所有虚拟机配置时间同步。

  您可以使用以下命令：

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

本指南中使用的配置示例

本指南使用以下部署假设：

- 三台 Google Compute Engine 虚拟机：

- `oracdb1` 用于主数据库
- `oracdb2` 用于备用数据库
- `oradg-obs` 用于快速启动故障转移观察器

- 每个数据库区域一个 GCNV Flex Unified 存储池
- 每个数据库主机五个 GCNV iSCSI 卷
- 用于自动故障转移的 Oracle Data Guard Broker 和 Fast-Start Failover
- 每个数据库主机的专用存储；主主机和备用主机不共享 iSCSI 卷

将命令中的所有示例值替换为环境中的值，包括主机名、IP 地址、区域、项目名称、门户 IP、密码和 Oracle 媒体文件名。

目标

在此过程中，您将完成以下任务：

- 为主数据库、备用数据库和 Observer 配置 Compute Engine 实例
- 为 Oracle 配置 GCNV iSCSI 存储和多路径设备
- 在两个数据库主机上安装 Oracle Grid Infrastructure 和 Oracle 数据库 26ai
- 创建主 Oracle 数据库
- 使用 GCNV 复制、快照或克隆初始化备用数据库
- 配置 Oracle Data Guard Broker、快速启动故障转移和 Observer

部署选项

本节比较了在 Google Compute Engine (GCE) 上使用 Google Cloud NetApp Volumes (GCNV) iSCSI 块存储的 Oracle 数据库的实际 HA/DR 部署模式。它还强调了 GCNV 复制如何加速备用数据库初始化。在开始之前，请使用此矩阵选择层级。本指南实施 Prod HA (Data Guard + FSFO) — 底行。

环境	需求	建议的架构	**HA**	灾难恢复	自动化	主要优势
开发/测试	最低成本	单个实例	否	是	否	Snapshot 克隆
Prod Basic（重新启动）	减少崩溃导致的停机时间	+ Oracle Restart	否	是	仅本地	自动重启
生产 HA（无 DG）	手动 DR 可接受	+ 快照 / RMAN	部分	是	部分	GCNV 克隆恢复
生产 HA (DG + FSFO)	True HA（无 DBA）	Data Guard + FSFO	是	是	完整	真正的 HA + 快速故障转移

HA / DR / Automation: 是 = 满足层级目标；否 = 不在范围内；部分 = 仅存储级或手动步骤。

您所在会员等级需要阅读的部分

根据您想要达到的高可用性水平，请阅读以下矩阵中的部分。例如，如果需要带有 Data Guard 和 FSFO 的 Prod HA，请阅读所有部分。如果需要 Dev/Test 或 Prod Basic with Restart，则仅读取第一列。

开发/测试或生产基础（重启）	**Prod HA**（无 **Data Guard**）	生产高可用性（**Data Guard + FSFO**）
开始之前	开始之前	开始之前
本指南中使用的配置示例	本指南中使用的配置示例	本指南中使用的配置示例
目标	目标	目标
部署选项	部署选项	部署选项
概述	概述	概述
架构	架构	架构
配置 Compute Engine	配置 Compute Engine	配置 Compute Engine
配置 GCNV iSCSI 卷	配置 GCNV iSCSI 卷	配置 GCNV iSCSI 卷
安装 Oracle 软件	安装 Oracle 软件	安装 Oracle 软件
创建主数据库	创建主数据库	创建主数据库
	创建备用数据库 （仅通过 步骤 3：GCNV 待机初始化）	创建备用数据库
		配置 Data Guard Broker、FSFO 和 Observer

概述

此架构使用 Google Cloud NetApp Volumes (GCNV) iSCSI 存储和 Oracle Data Guard 在 Google Cloud 上部署 Oracle Database 26ai 高可用性。GCNV 提供高性能块存储，支持存储层的待机初始化。Data Guard 提供连续同步、切换和快速启动故障转移。

层	角色
GCNV	块存储、备用初始化、存储复制
Data Guard	连续同步、重做应用、切换、FSFO

与 RMAN 活动复制相比，GCNV 复制通过在存储层而不是通过 Oracle 数据库通道移动数据，显著减少了备用初始化时间。当您的环境支持时，首选 GCNV 复制用于生产备用播种。

组件	详细信息
数据库主机	`oracdb1` / `oracdb2` — 不同的区域，每个区域五个 GCNV iSCSI 卷

组件	详细信息
存储	`/u01` + ASM `+DATA`, `+RECO`, `+FRA` 在 GCNV（外部）
待机初始化	GCNV 复制/快照/克隆 → Oracle 完成 → Data Guard
HA	重新启动，物理备用（MOUNTED），Broker，FSFO，Observer (`oradg-obs`)
应用	服务 `orclapp`

分区： 每个数据库区域一个 GCNV Flex Unified 池；每个主机专用卷。启动磁盘仅限操作系统。超出范围：TDE、RAC、NVMe/TCP、OEM、Active Data Guard（待机保持 **MOUNTED**）。

架构

参考图

该架构提供了三条独立的数据路径。存储复制和 Data Guard 重做传输是分开的路径。

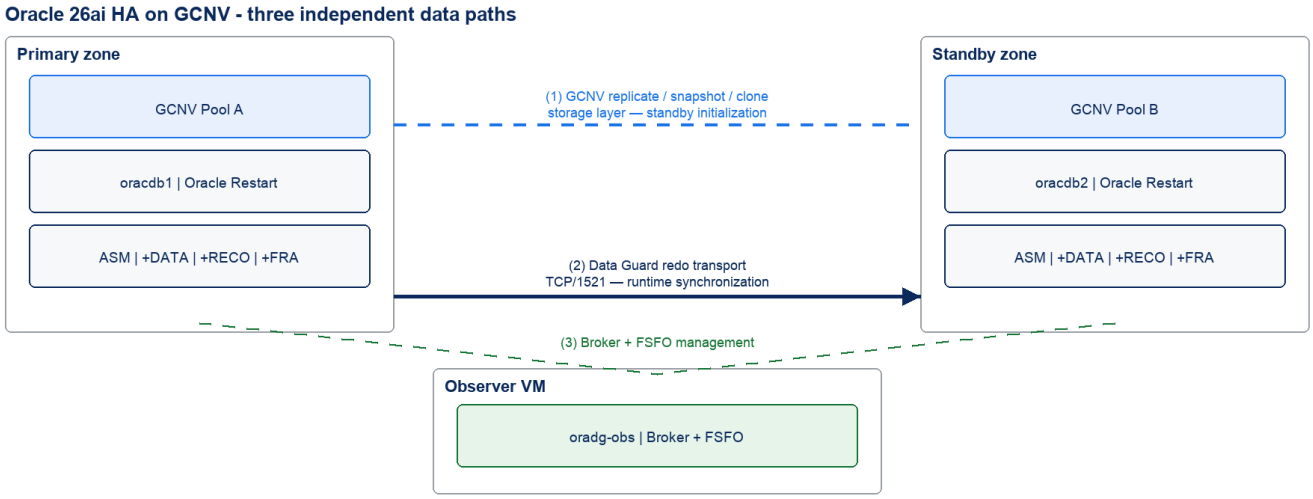

图 **1. GCNV Oracle** 部署 **-** 参考架构

*存储复制 ≠ 重做复制。*路径 1 在 GCNV 层移动数据文件以进行备用初始化。路径 2 在切换后保持数据库同步。路径 3 通过 Oracle Data Guard Broker 和 Observer 协调角色转换和自动故障转移。

平台角色

平台	提供
GCNV	iSCSI 卷、快照、卷复制（基线 + 增量）— 备用初始化

平台	提供
Data Guard	重做传输、MRP、Broker、FSFO — 连续同步和故障转移

GCNV 复制运行基线副本，然后根据策略进行增量块更新。Data Guard 在待机初始化后拥有 redo apply 和 FSFO。

拓扑和存储

角色	VM	分区	GCNV 池	卷（每台主机）
主云	`oracdb1`	`us-west1-a`	`oracle-pool-a`	`ora_<host>_u01`, `ora_<host>_data_01/02`, `ora_<host>_arch_01`, `ora_<host>_fra_01`
备用	`oracdb2`	`us-west1-b`	`oracle-pool-b`	相同的命名模式
Observer	`oradg-obs`	`us-west1-c`	—	仅启动磁盘

存储	备份	用途
操作系统	GCE 启动磁盘	仅操作系统
`/u01`	GCNV iSCSI	Grid/DB 主目录，暂存 (XFS)
`+DATA` / `+RECO` / `+FRA`	GCNV iSCSI	ASM EXTERNAL — 数据文件、归档、FRA

机器类型

本指南 `n4-highmem-8`中的示例；`OLTP` 工作负载通常使用 `c4-standard-*`。

配置 Compute Engine

步骤 1：创建 VM

在同一区域的不同可用区中创建三个 VM（可用区故障隔离）。如果满足延迟和合规性需求，则优先选择低碳区域以实现 TCO 和可持续性（例如 `us-west1` 与 `us-central1`）。使用 Cloud Console、 `gcloud`、Terraform 或您的标准配置工作流程。

VM	分区	机器类型	启动磁盘	网络	目的
`oracdb1`	`us-west1-a`	`n4-highmem-8` (sample) 或 `c4-standard-*`	OL 10，50 GB Hyperdisk Balanced（仅限操作系统）	`oracle-vpc` / `oracle-subnet`，gVNIC	主 DB
`oracdb2`	`us-west1-b`	与主体相同	OL 10，50 GB Hyperdisk Balanced（仅限操作系统）	相同	备用数据库

VM	分区	机器类型	启动磁盘	网络	目的
oradg-obs	us-west1-c	e2-medium	OL 10，20 GB Hyperdisk Balanced	相同	FSFO Observer（仅限 Instant Client）

网络层： 当延迟或出口流量（>~200 GiB/月）很重要时选择 Premium；在开发/测试中选择 Standard 以降低 TCO。

在所有三者上启用 **Secure Boot**、**vTPM** 和 **Integrity Monitoring**。启动磁盘仅容纳操作系统。 `/u01`、Grid/DB 主页、暂存和所有 ASM 数据都使用 GCNV iSCSI 卷（请参阅 配置 GCNV iSCSI 卷）— 请*勿*为 `/u01` 附加单独的 GCE 数据磁盘。

步骤 2：VPC 防火墙 — 跨所有三个区域的允许列表 TCP/1521

允许所有三个 VM `/32` 内部 IP 在每个区域(`us-west1-a/b/c`此处）之间使用 TCP/1521。使用具有相同允许列表的 VPC 防火墙规则或防火墙策略。缺少规则会破坏重做传输和 Observer 连接。

Cloud Console: VPC 网络 → 防火墙 → 创建规则 `allow-oracle-net-dbhosts` on `oracle-vpc` — 入口，允许，源 = 三个 `/32` IP，TCP 1521。如果需要，镜像出口。从每个 VM 验证：

```
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
```

端口	预期	含义
22	已连接	SSH 路径有效
1521	连接被拒绝	防火墙打开；Grid 侦听器在 步骤 1：在每个 DB 主机上安装 Oracle Grid Infrastructure (Oracle Restart) 期间启动
要么	超时	修复防火墙或路由

从所有三个虚拟机向每个对等 IP 运行。

步骤 3：主机名、DNS 和 `/etc/hosts`

每次虚拟机启动后，在所有三个主机上设置主机名并添加 `/etc/hosts`条目，以便正向/反向名称解析适用于 Oracle Net、代理和 Observer。

```
# 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
```

替换 GCE 内部 IP 地址（在 **Compute Engine → VM instances** 列表中可见，*Internal IP* 列）。

从每个主机验证：

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

步骤 4：OS 基线（仅限 DB 主机）

先决条件： 出站 HTTPS 到 `yum.oracle.com`（私有子网上的 Cloud NAT 或内部镜像）。

在 `oracdb1` 和 `oracdb2` 上运行（观察者设置详见步骤 4：在 Observer 主机上安装 Oracle Instant Client）。

```
# 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
```

安全态势 **(OL 10):** 以下命令将 SELinux 设置为允许模式并禁用 `firewalld`。这仅适用于最小化实验室环境。有关强化的 SELinux 和防火墙配置，请参阅您组织的安全基准。

```
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
```

第 5 步：捕获 iSCSI 启动程序名称 (IQN)

重新启动后，捕获每个数据库主机的 IQN。您将使用这些 IQN 在 步骤 2：创建主机组 中创建 GCNV 主机组。

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

在 `oracdb2`上重复。记录每个主机的 IQN — 每个主机一个主机组，以便单个主机的重新启动或 IQN 重新生成不会影响另一个主机的 GCNV iSCSI 卷可见性。

克隆的 **VM**： 如果两台主机共享相同的 IQN，请重新生成 `oracdb2`（停止 `iscsi`，清除 `/var/lib/iscsi/nodes/*`，新建 `InitiatorName /etc/iscsi/initiatorname.iscsi`，重新启动 `iscsid`）。

配置 GCNV iSCSI 卷

步骤 1：为每个数据库区域创建一个 GCNV Flex Unified iSCSI 存储池

每个数据库区域一个 Flex Unified 池（请参见架构）。

为此 HA 设计创建两个池（重复每个区域的步骤）：

池名称	分区	使用者
`oracle-pool-a`	`us-west1-a`	`oracdb1` (主要)
`oracle-pool-b`	`us-west1-b`	`oracdb2`（待机）

NetApp Volumes → **Storage pools** → **Create** 为每个池：

- 服务级别： Flex（非 Premium）
- 类型： 统一
- 区域： 匹配数据库 VM 区域 (`us-west1-a` / `us-west1-b`)
- **PSA**： 连接到 `oracle-vpc`
- *容量：*针对工作负载进行大小调整；当重做、备份或还原超过默认余量时，使用自定义配置的吞吐量/IOPS（每个池高达 5120 MiB/s 或 160K IOPS，每个产品限制）

等待 `READY`。将池大小缩放到数据库封装（步骤 3：创建 GCNV iSCSI 卷中的大小是示例）。

默认模式（本指南）： Flex 统一池使用默认模式 (`--mode=default`）。使用 Cloud Console 或 `gcloud netapp` 创建池和 iSCSI 卷。卷复制、快照和克隆使用 Google Cloud API （步骤 3：GCNV 待机初始化）。

步骤 2：创建主机组（每个 DB 主机一个）

为每个数据库主机创建一个主机组，以便每个虚拟机只能看到自己的卷 — 主卷和备用卷不得共享 GCNV iSCSI 卷。Observer 没有 GCNV 资源。在 Cloud Console 中：

1. **NetApp** 卷 → 主机组 → 创建

2. 名称： `oracdb1-hg` · 区域： `us-west1` · 类型： iSCSI 启动程序 · 操作系统类型： Linux
3. 主机： 粘贴来自 `oracdb1` 的 IQN（ `/etc/iscsi/initiatorname.iscsi` 的值）
4. 描述： "Oracle 主主机 oracdb1" · 创建
5. 对 `oracdb2-hg` 使用 `oracdb2`的 IQN 重复这些步骤

步骤 3：创建 GCNV iSCSI 卷（每个数据库主机）

每个数据库主机在其区域的池中获得五个 GCNV iSCSI 卷—一个用于 `/u01`，四个用于 ASM 后备设备：

GCNV iSCSI 卷	大小	用途	多路径别名
`ora_<host>_u01`	100 GiB	`/u01` GCNV iSCSI 卷 — 网格/Oracle 主目录，暂存	`/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` (条纹)	`/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`

卷名：仅限字母、数字、下划线（无连字符）。

> *最小布局（仅验证）：*每个主机两个 LUN (`*_data`, `*_reco`) with `arch_01p1`→`+RECO` and `arch_01p2`→`+FRA` 对于实验室是可接受的；生产环境每个 步骤 3：创建 GCNV iSCSI 卷 使用五个卷。

开 `oracdb1`：在 `oracle-pool-a`中创建所有五个卷，主机组 `oracdb1-hg`。

开 `oracdb2`：在 `oracle-pool-b`中创建所有五个卷，主机组 `oracdb2-hg`。

NetApp Volumes → Volumes → Create — iSCSI，正确的池和主机组，Linux。每个池的记录数：

- iSCSI 门户 IP → `<ISCSI_PORTAL_1>`， `<ISCSI_PORTAL_2>`（主池门户位于 `oracdb1`；备用池门户位于 `oracdb2`——它们可能不同）
- 来自 Cloud Console 的卷序列 — 与主机发现的 WWID 一起使用 步骤 4：为 GCNV iSCSI 卷配置 Linux iSCSI 和多路径

步骤 4：为 GCNV iSCSI 卷配置 Linux iSCSI 和多路径

在 `oracdb1`上使用该主机的池门户运行，然后在 `oracdb2`上使用备用池门户运行。

如果主机出口受到限制，则允许从每个数据库 VM 到其 GCNV iSCSI 门户 IP 的 TCP/3260（除了来自 步骤 2：VPC 防火墙 — 跨所有三个区域的允许列表 TCP/1521 的 VM 间 TCP/1521）。

1. 发现目标、登录并持久化节点启动：

```
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
```

重新启动后，在启动 Oracle 之前重新检查：

+

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

1. 配置 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. 将主机发现的 WWID 别名添加到 /etc/multipath.conf（不要猜测— multipath.conf *不*展开 shell 变量）。发现 WWID：

```
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
```

将该主机的具体别名附加到 `/etc/multipath.conf`，然后 `sudo systemctl restart multipathd`。

在 `oracdb1`上，追加：

```
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  }
}
```

+ 在 `oracdb2`上，对 `ora_oracdb2_*`别名使用相同的模式，然后：

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

第 5 步：对 GCNV iSCSI 卷上的 ASM 后备设备进行分区

用一个 GPT 分区对四个 ASM 后备设备（不是 `u01`）进行分区。ASM 消耗原始分区。在每台主机上运行。所有后续块使用 `HOST=$(hostname -s)`自动选择本地主机的设备。

```
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
```

第 6 步：格式化并挂载 `/u01`本地 GCNV iSCSI 卷

GCNV `ora_<host>_u01` iSCSI 卷包含 Grid home、Oracle home 和暂存。在多路径设备上格式化 XFS（未针对 ASM 进行分区）。在 `/etc/fstab`中使用 UUID（非共享文件系统标签）：

```
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
```

重新启动一次并确认 `/u01` 挂载在 安装 Oracle 软件 之前。

安装 Oracle 软件

在 创建主数据库 之前，在每个数据库主机上运行 Grid，然后进行数据库主页安装。

步骤 1：在每个 DB 主机上安装 Oracle Grid Infrastructure (Oracle Restart)

在 `oracdb1` 上运行此部分的所有内容，然后在 `oracdb2` 上重复。两台主机都有自己的 Grid 主目录、ASM 实例和磁盘组 — Data Guard 通过 Oracle Net 进行复制，而不是通过存储进行复制。

将 **Oracle** 二进制文件暂存到 `/u01`

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

就地解压缩 **Grid** 主目录

26ai Grid GoldImage 通过在目标 Grid 主目录就地解压缩来安装：

```
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 级别。*本指南假设 Grid GoldImage 已包括经过验证的 RU。如果 Grid GoldImage 比目标 RU 旧，请在安装过程中使用 Oracle 文档记录的 `gridSetup.sh -applyRU`流程修补 Grid 主页，或使用捆绑 RU 的 Grid GoldImage。将 Grid 和数据库主页保持在相同的预期补丁级别。

单次 `gridSetup` — 完整 **HA_CONFIG** 响应文件

必须在每台主机上构建 `/tmp/grid.rsp`(`responseFileVersion`；替换 `HOST`并使用强密码）：

```
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
```

运行 `gridSetup` 以复制二进制文件并暂存配置：

```
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
'
```

期望 `Successfully Setup Software with warning(s).`和退出代码 6（警告）或 0。

以根用户身份运行 `orainstRoot.sh`和 `root.sh`

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

`root.sh` 创建 `crsctl` / `srvctl` / `asmcmd` 包装器并启动 OHAS。

`gridSetup.sh -executeConfigTools` — 启动 **ASM** 并创建 `+DATA`

根据 的响应文件运行配置助理（NETCA、ASMCA、CVU）—这将创建 ASM 实例和 `+DATA` 磁盘组：

```
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
'
```

预计 `Successfully Configured Software.`在 NETCA / ASMCA / CVU 之后。

创建 `+RECO` 和 `+FRA` 磁盘组

单次安装只会创建 `+DATA`。通过以下方式创建另外两个 `asmca`：

```
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
"
```

验证 **ASM** 和 **Oracle Restart**

```
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.
```

在 `oracdb2`上重复此部分。 `HOST=$(hostname -s)`模式在和中自动选择该主机的 GCNV iSCSI 设备。使用相同的 ASM 磁盘组名称 — Data Guard 通过 Oracle Net 而不是存储进行复制。

步骤 2：在每个 DB 主机上安装 Oracle Database 26ai

在 `oracdb1` 上运行此部分，然后在 `oracdb2` 上运行。备用数据库在 创建备用数据库 中创建。

解压缩 **DB home** 和 **RU** 补丁

```
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
```

有关 RU 目录布局和 `-applyRU`路径，请参见 Oracle 文档。

应用 **RU** 的无提示纯软件安装

```
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
'
```

在 OL 8/9 上，从 `-applyOneOffs` 行中省略 `runInstaller`。

作为 `root`，运行安装后脚本：

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

设置 **Oracle** 环境

在每个 DB 主机(`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
```

（在备用主机上使用 `ORACLE_SID=orcls`。备用数据库在 创建备用数据库 中创建。）

创建主数据库(`oracdb1`仅)

对 ASM 磁盘组以静默模式运行 `dbca`：

```
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
'
```

将存档日志指向 `+RECO`（备用使用步骤 2：待机：最小 init.ora pfile，密码文件，NOMOUNT中的匹配设置）：

```
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
'
```

验证数据库在 Oracle Restart 下是否启动：

```
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
```

*创建基于角色的应用程序服务（Oracle Restart）。*应用程序应通过 `orclapp` 连接，而不是 DB 名称，因此故障转移是透明的。

```
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
'
```

启用 Broker 后， `orclapp` 仅在 PRIMARY 上运行。跨 ASM 磁盘组多路复用控制文件；将内存调整为工作负载大小（本指南使用 4 GB / 3 GB SGA 作为示例）。

创建备用数据库

在 orcls 上 oracdb2 构建（在 oracle-pool-b 上的专用卷）。完成初始主库和备库的设置，然后步骤 3：GCNV 待机初始化，执行备库完成任务，以及配置 Data Guard Broker、FSFO 和 Observer。目标：主库 READ WRITE；备库 PHYSICAL STANDBY，已挂载，正在应用 MRP。

步骤 1：Primary：SYS 密码、密码文件和 DG 参数

在 oracdb1`上作为 `oracle：

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

开启 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
```

在每个主机上通过 Oracle Restart 重新启动：

```
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 必须列出 <SID> 和 <SID>_DGMGRL。

步骤 2：待机：最小 init.ora pfile，密码文件，NOMOUNT

将主密码文件复制到备用 (IAP `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
```

在 `oracdb2`上，设置所有权并创建备用 `pfile`。在 **primary** 上，首先复制 `*.compatible`：

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

在 `oracdb2`上，在下面的块中将该值替换为 `<COPY_FROM_PRIMARY>`：

```
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
'
```

NOMOUNT 无数据文件，直到 步骤 3：GCNV 待机初始化。

步骤 3：GCNV 待机初始化

使用 **Google Cloud NetApp Volumes** 复制（基于 SnapMirror，**Default-mode**）、*卷快照*或*克隆*在 *`oracle-pool-b` 中填充备用 ASM 卷，然后运行 Oracle 最终确定。iSCSI 和 ASM 与 配置 GCNV iSCSI 卷 相同。

API	用途
`gcloud netapp storage-pools`	创建 **Default-mode** Flex Unified 池 (`--mode=default`)
`gcloud netapp volumes`	iSCSI 卷、主机组、快照
`gcloud netapp volumes replications`	跨位置*卷复制*

步骤	操作
1	主存档日志 + 强制日志记录（步骤 1：Primary：SYS 密码、密码文件和 DG 参数）
2	静止： `BEGIN BACKUP`，记录 SCN，待机控制文件
3	卷复制（创建卷复制 和 切换—静止、停止复制、连接备用 LUN）或快照（替代方案 — snapshot 种子）
4	开 `oracdb2`：iSCSI、多路径、ASM 挂载（步骤 4：为 GCNV iSCSI 卷配置 Linux iSCSI 和多路径、第 5 步：对 GCNV iSCSI 卷上的 ASM 后备设备进行分区 和 ）
5	Oracle finalize — 恢复到 SCN，**MOUNTED** (Oracle 最终确定)
6	SRL rebuild (步骤 4：待机重做日志文件)、MRP、broker (配置 Data Guard Broker 、FSFO 和 Observer)

RMAN 活动副本对于小型实验室仍然有效。*首选 GCNV 复制*用于生产备用播种。

前提条件

- `gcloud netapp` 支持卷复制。
- 两个 **Default-mode** 池位于不同位置 (`oracle-pool-a`, `oracle-pool-b`)。
- 连接到 `oracdb1-hg`的主池上的源卷；通过复制创建的目标卷。
- 从 Cloud Shell 或工作站运行复制，而不是从 DB VM。

```
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"
```

根据需要创建备用池：

```
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>
```

创建卷复制

```
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"
```

等到 `mirrorState`为 **MIRRORED** / 初始同步完成。

切换—静止、停止复制、连接备用 **LUN**

主要：

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

允许最终复制周期，然后：

```
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
```

将目标卷附加到 `oracdb2-hg`（复制的 LUN 可能保留源名称—在更新时使用 `name=oracdb1_data_lun`）：

```
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"
```

将控制文件复制到 `oracdb2`，然后在主节点上：

```
ALTER DATABASE END BACKUP;
```

替代方案 — snapshot 种子

一次性种子：在源卷上创建快照 → 在备用池中从快照创建卷（Cloud Console 或 API）。在附加到 `oracdb2-hg`后，继续Oracle 最终确定。

备用 iSCSI 和 ASM（在 RMAN 之前）

在 `oracdb2`上，登录到*备用池* `iSCSI` 门户。如果 `ASM` 磁盘标头与主命名匹配，请使用*主样式多路径别名* (lab: `ora_oracdb1_data_01`, `ora_oracdb1_arch_01`)，在分区上设置 `asm_diskstring='/dev/mapper/ora_oracdb1_*p*'`, `chown grid:asmadmin`，然后：

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

Oracle 最终确定

```
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;
```

转换后，重建备用重做日志（复制的控制文件仍然具有 `+DATA/ORCL/...` SRL 路径——导致 `ORA-19527`/ `ORA-16086`在主节点上）。请参阅步骤 4：待机重做日志文件。

步骤 4：待机重做日志文件

FSFO 的*两个*主机都需要。大小 ≥ 最大主在线重做；计数 = (每个线程的在线组) + 1。

GCNV 种子后： 丢弃待机上的所有待机日志文件组，仅在 `+DATA`上重新创建（`db_create_file_dest='+DATA'`）。 `+DATA/ORCL/…`下的复制路径导致 `ORA-19527/`ORA-16086`直到重建。

主要(**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
```

待机 **(orcls)** 在 **GCNV** 种子后：

```
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;
```

步骤 5：在待机状态下启用闪回并启动托管恢复

*在*启动托管恢复之前启用闪回（在 MRP 处于活动状态时无法启用闪回）。

```
# 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 启用实时应用（重做在到达 SRL 时应用）。

步骤 6：在主服务器上启用 redo shipping

> LOG_ARCHIVE_DEST_2 被故意设置为 `DEFER`在步骤 2：待机：最小 init.ora pfile，密码文件，NOMOUNT中抑制连续 `ORA-12154`错误在待机创建过程中。现在待机就绪后启用它。

切换 LOG_ARCHIVE_DEST_STATE_2 到 ENABLE 并强制进行日志切换，以便第一轮重做立即发送：

```
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.
```

如果 dest_2`显示 `ORA-12154，则弹跳主节点。在步骤 1：在两个数据库上启用 broker之后，通过 DGMGRL 管理传输。

步骤 7：验证目标 Data Guard 状态

在*主* (oracdb1)：

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

在*待机* (oracdb2 — Cloud Console **SSH**，或从您的工作站使用 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
```

待机时应为： `PHYSICAL STANDBY | MOUNTED`; `MRP0` with `APPLYING_LOG`。

如果待机报告 `MOUNTED`但应用未运行，请在 `oracdb2`上重新启动 MRP：

```
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'
```

第 8 步：在 Oracle Restart 中注册备用数据库

在 GCNV 播种后通过重新启动注册待机，以便重新启动恢复 ASM、MOUNT 和应用。

在 `oracdb2` 上，捕获 spfile 位置并向 Oracle Restart 注册（从查询中替换 `<STANDBY_SPFILE_PATH>`，通常在 `+DATA` 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
'
```

在 `oracdb1`，验证主 Oracle Restart 数据库资源是否列出了 ASM 磁盘组依赖关系。如果仅通过 DBCA 注册，则添加 `RECO DATA`和 `FRA`：

```
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
'
```

在备用数据库资源上添加相同的应用程序服务 (`orcls` on `oracdb2`)。在两侧使用 `role PRIMARY`，以便 `orclapp`在切换后可用：

```
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
'
```

在 `oracdb2`上，确认备用数据库资源：

```
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
'
```

配置 Data Guard Broker、FSFO 和 Observer

之后 `ENABLE CONFIGURATION`，通过 **DGMGRL**（非临时 `LOG_ARCHIVE_DEST_*` SQL）管理传输和角色。

步骤 1：在两个数据库上启用 broker

在*主*和*备用*数据库主机上：

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

在*主数据库主机*上，使用 OS 身份验证进行连接（DB 主机上不需要来自第 5 步：将 Observer 作为 systemd 单元启动的 Observer 钱包）：

```
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 /
'
```

仅在 Observer 主机上，在自动登录钱包存在后使用 `dgmgrl /@orcl`。请勿将密码放在 `dgmgrl`命令行上。

```
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.
```

立即运行 `VALIDATE DATABASE`—在尝试切换之前，它会显示孤立的数据文件、丢失的 SRL、重做应用未运行以及其他常见问题。任何 `WARNING`或非 NULL `ERROR`必须在步骤 3：配置 FSFO 属性并启用之前修复。

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

步骤 2：确认闪回（FSFO 自动恢复所需）

在启用 FSFO 之前，在*两台*主机上确认 `flashback_on`：

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

仅在 主 上，如果尚未设置保留：

```
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'
```

步骤 3：配置 FSFO 属性并启用

```
-- 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.
```

步骤 4：在 Observer 主机上安装 Oracle Instant Client

```
# 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
```

第 5 步：将 Observer 作为 systemd 单元启动

凭据位于自动登录钱包中—从不在 `dgmgrl`命令行上（对 `ps/`journalctl`可见）。仅在 Observer 上与 `/@<tns_alias>`连接。

- 使用专用帐户： 将专用 Data Guard 管理帐户（例如 `SYSDG`）的凭据存储在钱包中，而不是 `SYS`。
- 需要自动登录钱包： Observer systemd 服务需要自动登录钱包 `cwallet.sso`）。如果 `cwallet.sso`在运行 `mkstore`后丢失，请使用 `orapki`从 Instant Client **tools** 包或数据库主页创建自动登录钱包，然后重新添加存储的凭据。

1. 使用两个成员的凭据在 Observer 上创建钱包：

```
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 单元 + 日志轮换：

```
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
```

从主数据库验证 — Observer 必须读取 `CONNECTED`（ `DISCONNECTED` Observer 静默暂停 FSFO）：

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

第 6 步：测试 FSFO（切换和故障转移）

计划切换：

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

计划外故障转移： 在测试窗口中使用 VM 重置（崩溃式测试）；正常的 停止 可能不会触发 FSFO。跟踪 `/var/log/dgmgrl-observer.log`在 `oradg-obs；完成后恢复拓扑。