Deploy Oracle Database 26ai high availability with Google Cloud NetApp Volumes

NetApp database solutions

NetApp
June 09, 2026

Table of Contents

Deploy Oracle Database 26ai high availability with Google Cloud NetApp Volumes

This guide shows how to provision compute instances and storage, install Oracle Grid Infrastructure and Oracle Database, initialize a standby database, and configure Oracle Data Guard with Fast-Start Failover.

Before you begin

Before you begin, ensure that you have the following:

- A Google Cloud project with permissions for Compute Engine, VPC networking, firewall configuration, IAM, and NetApp Volumes

Task	Required access
Create Compute Engine VMs	Compute Instance Admin (or equivalent)
Firewall / Firewall Policies	Network Admin or delegated policy admin
Create GCNV pools and volumes	NetApp Volumes Admin
Configure PSA	Network admin in host project
SSH via IAP	IAP-secured Tunnel User + OS Login (if used)

- The NetApp Volumes API enabled
- A VPC and subnet configured for the target region
- Private Services Access (PSA) configured for Google Cloud NetApp Volumes
- Oracle Linux 10 for all required virtual machines
- DNS and hostname resolution is configured for the database hosts and Observer host
- Oracle installation media and patch files available for Oracle Database 26ai and Grid Infrastructure
- Familiarity with Oracle Data Guard, Oracle Restart, and iSCSI storage concepts
- Time synchronization is configured for all virtual machines.

  You can use the following commands:

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

Example configuration used in this guide

This guide uses the following deployment assumptions:

- Three Google Compute Engine virtual machines:

    - `oracdb1` for the primary database
    - `oracdb2` for the standby database
    - `oradg-obs` for the Fast-Start Failover Observer
- One GCNV Flex Unified storage pool per database zone
- Five GCNV iSCSI volumes per database host
- Oracle Data Guard Broker and Fast-Start Failover for automatic failover
- Dedicated storage per database host; primary and standby hosts do not share iSCSI volumes

Replace all example values in commands with values from your environment, including host names, IP addresses, zones, project names, portal IPs, passwords, and Oracle media file names.

Objectives

In this procedure, you complete the following tasks:

- Provision Compute Engine instances for the primary database, standby database, and Observer
- Configure GCNV iSCSI storage and multipath devices for Oracle
- Install Oracle Grid Infrastructure and Oracle Database 26ai on both database hosts
- Create the primary Oracle database
- Initialize the standby database by using GCNV replication, snapshots, or clones
- Configure Oracle Data Guard Broker, Fast-Start Failover, and the Observer

Deployment options

This section compares practical HA/DR deployment patterns for Oracle Database on Google Compute Engine (GCE) with Google Cloud NetApp Volumes (GCNV) iSCSI block storage. It also highlights how GCNV replication accelerates standby initialization. Use this matrix to choose a tier before you start. This guide implements Prod HA (Data Guard + FSFO) — the bottom row.

Environment	Requirement	Recommended architecture	HA	DR	Automation	Key benefit
Dev/Test	Lowest cost	Single instance	No	Yes	No	Snapshot clone
Prod Basic (Restart)	Reduce downtime from crashes	+ Oracle Restart	No	Yes	Local only	Auto-restart
Prod HA (no DG)	Manual DR acceptable	+ Snapshots / RMAN	Partial	Yes	Partial	GCNV clone recovery
Prod HA (DG + FSFO)	True HA (no DBA)	Data Guard + FSFO	Yes	Yes	Full	True HA + fast failover

HA / DR / Automation: Yes = meets tier goal; No = not in scope; Partial = storage-level or manual steps only.

Sections to read for your tier

Depending on the level of HA you want to achieve, read the sections in the matrix below. For example, if you want Prod HA with Data Guard and FSFO, read all sections. If you want Dev/Test or Prod Basic with Restart, read only the first column.

Dev/Test or Prod Basic (Restart)	Prod HA (no Data Guard)	Prod HA (Data Guard + FSFO)
Before you begin	Before you begin	Before you begin
Example configuration used in this guide	Example configuration used in this guide	Example configuration used in this guide
Objectives	Objectives	Objectives
Deployment options	Deployment options	Deployment options
Overview	Overview	Overview
Architecture	Architecture	Architecture
Provision Compute Engine	Provision Compute Engine	Provision Compute Engine
Provision GCNV iSCSI volumes	Provision GCNV iSCSI volumes	Provision GCNV iSCSI volumes
Install Oracle software	Install Oracle software	Install Oracle software
Create the primary database	Create the primary database	Create the primary database
	Create the standby database (through Step 3: GCNV standby initialization only)	Create the standby database
		Configure Data Guard Broker, FSFO, and Observer

Overview

This architecture deploys Oracle Database 26ai high availability on Google Cloud using Google Cloud NetApp Volumes (GCNV) iSCSI storage and Oracle Data Guard. GCNV provides high-performance block storage and supports standby initialization at the storage layer. Data Guard provides continuous synchronization, switchover, and Fast-Start Failover.

Layer	Role
GCNV	Block storage, standby initialization, storage replication
Data Guard	Continuous synchronization, redo apply, switchover, FSFO

GCNV replication significantly reduces standby initialization time compared to RMAN active duplicate by moving data at the storage layer instead of through Oracle database channels. Prefer GCNV replication for production standby seeding when your environment supports it.

Component	Detail
DB hosts	`oracdb1` / `oracdb2` — different zones, five GCNV iSCSI volumes each
Storage	`/u01` + ASM `+DATA`, `+RECO`, `+FRA` on GCNV (EXTERNAL)
Standby init	GCNV replicate / snapshot / clone → Oracle finalize → Data Guard
HA	Restart, physical standby (MOUNTED), Broker, FSFO, Observer (`oradg-obs`)
Apps	Service `orclapp`

Zoning: One GCNV Flex Unified pool per database zone; dedicated volumes per host. Boot disks are OS-only.
Out of scope: TDE, RAC, NVMe/TCP, OEM, Active Data Guard (standby stays **MOUNTED**).

Architecture

Reference diagram

The architecture provides three independent data paths. Storage replication and Data Guard redo transport are separate paths.

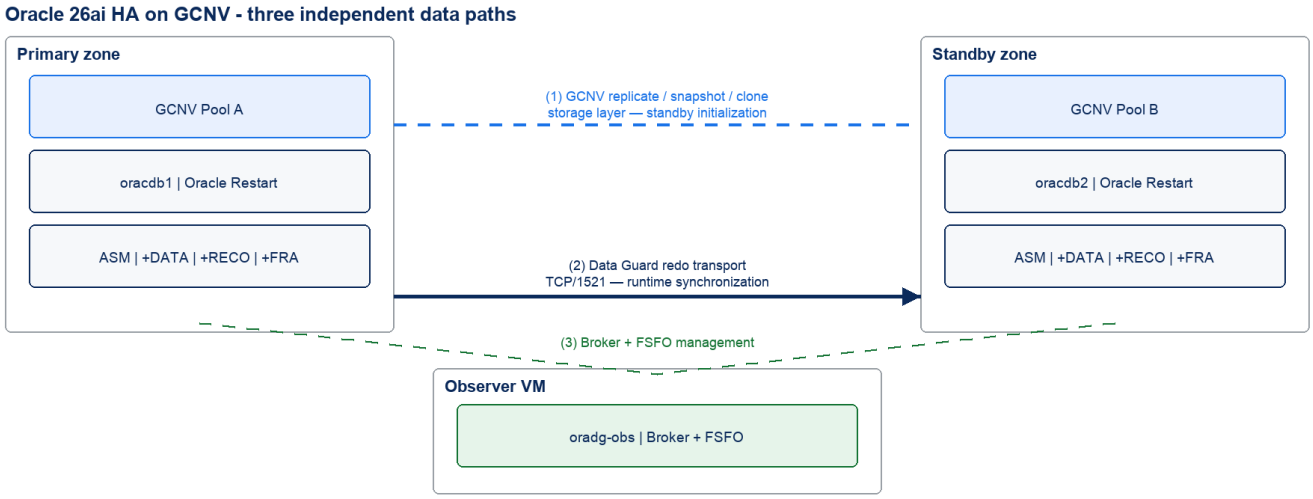

Figure 1. GCNV Oracle Deployment - reference architecture

Storage replication ≠ redo replication. Path 1 moves datafiles at the GCNV layer for standby initialization. Path 2 keeps databases synchronized after cutover. Path 3 orchestrates role transitions and automatic failover through Oracle Data Guard Broker and the Observer.

Platform roles

Platform	Delivers
GCNV	iSCSI volumes, snapshots, volume replication (baseline + incrementals) — standby initialization
Data Guard	Redo transport, MRP, Broker, FSFO — continuous synchronization and failover

GCNV replication runs a baseline copy, then incremental block updates per policy. Data Guard owns redo apply and FSFO after the standby is initialized.

Topology and storage

Role	VM	Zone	GCNV pool	Volumes (per host)
Primary	`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`	Same naming pattern
Observer	`oradg-obs`	`us-west1-c`	—	Boot disk only

Storage	Backing	Use
OS	GCE boot disk	OS only
`/u01`	GCNV iSCSI	Grid/DB homes, staging (XFS)
`+DATA` / `+RECO` / `+FRA`	GCNV iSCSI	ASM EXTERNAL — datafiles, archives, FRA

Machine types

Sample `n4-highmem-8` in this guide; OLTP workloads typically use `c4-standard-*`.

Provision Compute Engine

Step 1: Create the VMs

Create three VMs in different zones of the same region (zonal failure isolation). Prefer a lower-carbon region for TCO and sustainability where it meets latency and compliance needs (for example `us-west1` vs `us-central1`). Use the Cloud Console, `gcloud`, Terraform, or your standard provisioning workflow.

VM	Zone	Machine type	Boot disk	Network	Purpose
`oracdb1`	`us-west1-a`	`n4-highmem-8` (sample) or `c4-standard-*`	OL 10, 50 GB Hyperdisk Balanced (OS only)	`oracle-vpc` / `oracle-subnet`, gVNIC	Primary DB

VM	Zone	Machine type	Boot disk	Network	Purpose
`oracdb2`	`us-west1-b`	Same as primary	OL 10, 50 GB Hyperdisk Balanced (OS only)	Same	Standby DB
`oradg-obs`	`us-west1-c`	`e2-medium`	OL 10, 20 GB Hyperdisk Balanced	Same	FSFO Observer (Instant Client only)

Network tier: Premium when latency or egress (>~200 GiB/month) matters; Standard for lower TCO in dev/test.

Enable **Secure Boot**, **vTPM**, and **Integrity Monitoring** on all three. The boot disk holds the OS only. `/u01`, Grid/DB homes, staging, and all ASM data use GCNV iSCSI volumes (see Provision GCNV iSCSI volumes) — do **not** attach a separate GCE data disk for `/u01`.

Step 2: VPC firewall — allowlist TCP/1521 across all three zones

Allow TCP/1521 between all three VM `/32` internal IPs in every zone (`us-west1-a/b/c` here). Use VPC firewall rules or Firewall Policies with the same allowlist. Missing rules break redo transport and Observer connectivity.

Cloud Console: VPC network → Firewall → Create rule `allow-oracle-net-dbhosts` on `oracle-vpc` — Ingress, Allow, sources = three `/32` IPs, TCP 1521. Mirror egress if required. Validate from each VM:

```
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
```

Port	Expected	Meaning
`22`	Connected	SSH path works
`1521`	Connection refused	Firewall open; Grid listener starts during Step 1: Install Oracle Grid Infrastructure (Oracle Restart) on each DB host
Either	Timeout	Fix firewall or routing

Run from all three VMs toward each peer IP.

Step 3: Hostname, DNS, and `/etc/hosts`

After each VM boots, set the hostname and add `/etc/hosts` entries on all three hosts so forward/reverse name resolution works for Oracle Net, the broker, and the 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
```

Substitute the GCE internal IP addresses (visible in the **Compute Engine** → **VM instances** list, *Internal IP* column).

Validate from each host:

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

Step 4: OS baseline (DB hosts only)

Prerequisite: Outbound HTTPS to `yum.oracle.com` (Cloud NAT or internal mirror on private subnets).

Run on `oracdb1` and `oracdb2` (Observer setup is covered in Step 4: Install Oracle Instant Client on the Observer host).

```
# 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
```

Security posture (OL 10): The commands below set SELinux to permissive and disable `firewalld`. This is a minimal lab posture only. For hardened SELinux and firewall configuration, consult your organization's security baseline.

```
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
```

Step 5: Capture the iSCSI initiator name (IQN)

After the reboot, capture the IQN of each DB host. You will use these IQNs to create the GCNV host groups in Step 2: Create the host groups.

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

Repeat on `oracdb2`. Record each host's IQN — one host group per host so a single host's reboot or IQN regeneration cannot affect another host's GCNV iSCSI volume visibility.

Cloned VMs: If both hosts share the same IQN, regenerate on `oracdb2` (stop `iscsi`, clear `/var/lib/iscsi/nodes/*`, new `InitiatorName` in `/etc/iscsi/initiatorname.iscsi`, restart `iscsid`).

Provision GCNV iSCSI volumes

Step 1: Create one GCNV Flex Unified iSCSI storage pool per database zone

One Flex Unified pool per database zone (see Architecture).

Create two pools for this HA design (repeat the steps for each zone):

Pool name	Zone	Used by
`oracle-pool-a`	`us-west1-a`	`oracdb1` (primary)
`oracle-pool-b`	`us-west1-b`	`oracdb2` (standby)

NetApp Volumes → **Storage pools** → **Create** for each pool:

- **Service level:** Flex (not Premium)
- **Type:** Unified
- **Zone:** match the database VM zone (`us-west1-a` / `us-west1-b`)
- **PSA:** connected to `oracle-vpc`
- **Capacity:** sized for workload; use custom provisioned throughput/IOPS when redo, backup, or restore exceeds default headroom (up to 5120 MiB/s or 160K IOPS per pool, per product limits)

Wait for `READY`. Scale pool sizes to your database footprint (the sizes in Step 3: Create the GCNV iSCSI volumes are examples).

Default-mode (this guide): Flex Unified pools use Default-mode (`--mode=default`). Create pools and iSCSI volumes with Cloud Console or `gcloud netapp`. Volume replication, snapshots, and clones use Google Cloud APIs (Step 3: GCNV standby initialization).

Step 2: Create the host groups (one per DB host)

Create one host group per database host so each VM only sees its own volumes — primary and standby must not share GCNV iSCSI volumes. The Observer has no GCNV resources. In the Cloud Console:

1. **NetApp Volumes** → **Host groups** → **Create**
2. **Name:** `oracdb1-hg` · **Region:** `us-west1` · **Type:** iSCSI initiator · **OS type:** Linux

3. **Hosts:** paste the IQN from `oracdb1` (the value of `/etc/iscsi/initiatorname.iscsi`)
4. **Description:** “Oracle primary host oracdb1” · **Create**
5. Repeat for `oracdb2-hg` with `oracdb2`’s IQN

Step 3: Create the GCNV iSCSI volumes (per database host)

Each database host gets five GCNV iSCSI volumes in its zone’s pool — one for `/u01` and four ASM backing devices:

GCNV iSCSI volume	Size	Use	Multipath alias
`ora_<host>_u01`	100 GiB	`/u01` GCNV iSCSI volume — Grid/Oracle homes, staging	`/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` (striped)	`/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`

Volume names: letters, numbers, underscores only (no hyphens).

Minimal layout (validation only): Two LUNs per host (`*_data`, `*_reco`) with `arch_01p1` →`+RECO` and `arch_01p2`→`+FRA` is acceptable for lab; production uses five volumes per Step 3: Create the GCNV iSCSI volumes.

On `oracdb1`: create all five volumes in `oracle-pool-a`, host group `oracdb1-hg`.

On `oracdb2`: create all five volumes in `oracle-pool-b`, host group `oracdb2-hg`.

NetApp Volumes → **Volumes** → **Create** — iSCSI, correct pool and host group, Linux. Record per pool:

- iSCSI portal IPs → `<ISCSI_PORTAL_1>`, `<ISCSI_PORTAL_2>` (primary pool portals on `oracdb1`; standby pool portals on `oracdb2` — they may differ)
- Volume serial from the Cloud Console — use with host-discovered WWID in Step 4: Configure Linux iSCSI and multipath for GCNV iSCSI volumes

Step 4: Configure Linux iSCSI and multipath for GCNV iSCSI volumes

Run on `oracdb1` using that host’s pool portals, then on `oracdb2` using the standby pool portals.

If host egress is restricted, allow TCP/3260 from each database VM to its GCNV iSCSI portal IPs (in addition to inter-VM TCP/1521 from Step 2: VPC firewall — allowlist TCP/1521 across all three zones).

1. Discover targets, log in, and persist node startup:

```
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
```

After reboot, re-check before starting Oracle:

+

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

1. Configure `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. Add host-discovered WWID aliases to `/etc/multipath.conf` (do not guess — `multipath.conf` does **not** expand shell variables). Discover WWIDs:

```
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
```

Append concrete aliases for that host to `/etc/multipath.conf`, then `sudo systemctl restart multipathd`.

On `oracdb1`, append:

```
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  }
}
```

+ On `oracdb2`, use the same pattern with `ora_oracdb2_*` aliases, then:

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

Step 5: Partition ASM backing devices on GCNV iSCSI volumes

Partition the four ASM backing devices (not `u01`) with one GPT partition each. ASM consumes the raw partition. Run on each host. All subsequent blocks use `HOST=$(hostname -s)` to select the local host's devices automatically.

```
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
```

Step 6: Format and mount `/u01` on the local GCNV iSCSI volume

The `ora_<host>_u01` GCNV iSCSI volume holds Grid home, Oracle home, and staging. Format XFS on the multipath device (not partitioned for ASM). Use UUID in `/etc/fstab` (not a shared filesystem label):

```
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
```

Reboot once and confirm `/u01` mounts before Install Oracle software.

Install Oracle software

Run Grid and then database home installation on each DB host before Create the primary database.

Step 1: Install Oracle Grid Infrastructure (Oracle Restart) on each DB host

Run all of this section on `oracdb1`, then repeat on `oracdb2`. Both hosts get their own Grid home, ASM instance, and disk groups — Data Guard replicates over Oracle Net, not over storage.

Stage the Oracle binaries on `/u01`

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

Unzip the Grid home in place

The 26ai Grid GoldImage installs by unzipping in place at the target Grid home:

```
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 level. This guide assumes the Grid GoldImage already includes your validated RU. If the Grid GoldImage is older than the target RU, patch the Grid home during setup using the Oracle-documented `gridSetup.sh -applyRU` flow, or use a Grid GoldImage with the RU bundled. Keep Grid and Database homes at the same intended patch level.

Single-shot `gridSetup` — full HA_CONFIG response file

Build `/tmp/grid.rsp` on each host (`responseFileVersion` is mandatory; substitute `HOST` and use strong passwords):

```
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
```

Run `gridSetup` to copy the binaries and stage the configuration:

```
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
'
```

Expect `Successfully Setup Software with warning(s).` and exit code 6 (warnings) or 0.

Run `orainstRoot.sh` and `root.sh` as root

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

`root.sh` creates the `crsctl` / `srvctl` / `asmcmd` wrappers and brings up OHAS.

`gridSetup.sh -executeConfigTools` — bring up ASM and create `+DATA`

Run the configuration assistants (NETCA, ASMCA, CVU) against the response file from Single-shot `gridSetup` — full HA_CONFIG response file — this creates the ASM instance and the `+DATA` disk group:

```
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
'
```

Expect `Successfully Configured Software.` after NETCA / ASMCA / CVU.

Create `+RECO` and `+FRA` disk groups

The single-shot install only creates `+DATA`. Create the other two via `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
"
```

Verify ASM and 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.
```

Repeat this section on `oracdb2`. The `HOST=$(hostname -s)` pattern in Single-shot `gridSetup` — full HA_CONFIG response file and Create `+RECO` and `+FRA` disk groups selects that host's GCNV iSCSI devices automatically. Use the same ASM disk group names — Data Guard replicates over Oracle Net, not storage.

Step 2: Install Oracle Database 26ai on each DB host

Run this section on `oracdb1`, then on `oracdb2`. The standby database is created in Create the standby database.

Unzip the DB home and the RU patch

```
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
```

See the Oracle documentation for RU directory layout and `-applyRU` path.

Silent software-only install with the RU applied

```
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
'
```

On OL 8/9, omit `-applyOneOffs` from the `runInstaller` line.

As `root`, run the post-install script:

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

Set the Oracle environment

On each DB host (`orcl` on `oracdb1`, `orcls` on `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
```

(Use `ORACLE_SID=orcls` on the standby host. The standby database is created in Create the standby database.)

Create the primary database (`oracdb1` only)

Run `dbca` in silent mode against the ASM disk groups:

```
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
'
```

Point archivelogs at `+RECO` (the standby uses matching settings in Step 2: Standby: minimal init.ora pfile, password file, NOMOUNT):

```
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
'
```

Verify the database is up under Oracle Restart:

```
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
```

Create a role-based application service (Oracle Restart). Applications should connect via `orclapp`, not the DB name, so failover is transparent.

```
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
'
```

After Broker enable, `orclapp` runs only on PRIMARY. Multiplex control files across ASM disk groups; size memory to workload (this guide uses 4 GB / 3 GB SGA as examples).

Create the standby database

Build `orcls` on `oracdb2` (dedicated volumes on `oracle-pool-b`). Complete the initial primary and standby setup, then Step 3: GCNV standby initialization, the standby completion tasks, and Configure Data Guard Broker, FSFO, and Observer. Target: primary READ WRITE; standby PHYSICAL STANDBY, MOUNTED, MRP applying.

Step 1: Primary: SYS password, password file, and DG parameters

On `oracdb1` as `oracle`:

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

On `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
```

Restart through Oracle Restart on each host:

```
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` must list `<SID>` and `<SID>_DGMGRL`.

Step 2: Standby: minimal init.ora pfile, password file, NOMOUNT

Copy the primary password file to the standby (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
```

On `oracdb2`, set ownership and create the standby pfile. On the **primary**, copy `*.compatible` first:

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

On `oracdb2`, substitute that value for `<COPY_FROM_PRIMARY>` in the block below:

```
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 with no datafiles until Step 3: GCNV standby initialization.

Step 3: GCNV standby initialization

Populate standby ASM volumes in **`oracle-pool-b`** using **Google Cloud NetApp Volumes replication** (SnapMirror-based, **Default-mode**), **volume snapshots**, or **clone**, then run Oracle finalize. iSCSI and ASM are the same as Provision GCNV iSCSI volumes.

API	Use
`gcloud netapp storage-pools`	Create **Default-mode** Flex Unified pools (`--mode=default`)
`gcloud netapp volumes`	iSCSI volumes, host groups, snapshots
`gcloud netapp volumes replications`	Cross-location **volume replication**

Step	Action
1	Primary archivelog + force logging (Step 1: Primary: SYS password, password file, and DG parameters)
2	Quiesce: `BEGIN BACKUP`, record SCN, standby controlfile
3	Volume replication (Create volume replications and Cutover — quiesce, stop replication, attach standby LUNs) or snapshot (Alternative — snapshot seed)
4	On `oracdb2`: iSCSI, multipath, ASM mount (Step 4: Configure Linux iSCSI and multipath for GCNV iSCSI volumes, Step 5: Partition ASM backing devices on GCNV iSCSI volumes, and Step 6: Format and mount `/u01` on the local GCNV iSCSI volume)
5	Oracle finalize — recover to SCN, **MOUNTED** (Oracle finalize)
6	SRL rebuild (Step 4: Standby redo log files), MRP, broker (Configure Data Guard Broker, FSFO, and Observer)

RMAN active duplicate remains valid for small labs. **GCNV replication is preferred** for production standby seeding.

Prerequisites

- `gcloud netapp` with volume replication support.
- Two **Default-mode** pools in different locations (`oracle-pool-a`, `oracle-pool-b`).
- Source volumes on primary pool attached to `oracdb1-hg`; destination volumes created by replication.
- Run replication from Cloud Shell or workstation — not from DB VMs.

```
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"
```

Create standby pool if needed:

```
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>
```

Create volume replications

```
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"
```

Wait until `mirrorState` is **MIRRORED** / initial sync complete.

Cutover — quiesce, stop replication, attach standby LUNs

Primary:

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

Allow final replication cycle, then:

```
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
```

Attach destination volumes to `oracdb2-hg` (replicated LUN may keep source name — use `name=oracdb1_data_lun` on update):

```
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"
```

Copy controlfile to `oracdb2`, then on primary:

```
ALTER DATABASE END BACKUP;
```

Alternative — snapshot seed

One-time seed: snapshot on source volume → create volume from snapshot in standby pool (Cloud Console or API). Proceed to Oracle finalize after attach to `oracdb2-hg`.

Standby iSCSI and ASM (before RMAN)

On `oracdb2`, log in to **standby pool** iSCSI portals. If ASM disk headers match primary naming, use **primary-style multipath aliases** (lab: `ora_oracdb1_data_01`, `ora_oracdb1_arch_01`), set `asm_diskstring='/dev/mapper/ora_oracdb1_*p*'`, `chown grid:asmadmin` on partitions, then:

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

Oracle finalize

```
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;
```

After convert, **rebuild standby redo logs** (replicated controlfile still has `+DATA/ORCL/…` SRL paths — causes `ORA-19527` / `ORA-16086` on primary). See Step 4: Standby redo log files.

Step 4: Standby redo log files

Required on **both** hosts for FSFO. Size ≥ largest primary online redo; count = (online groups per thread) + 1.

After GCNV seed: Drop all standby logfile groups on the standby and recreate on `+DATA` only (`db_create_file_dest='+DATA'`). Replicated paths under `+DATA/ORCL/…` cause `ORA-19527` / `ORA-16086` until rebuilt.

Primary (`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
```

Standby (`orcls`) after GCNV seed:

```
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;
```

Step 5: Enable flashback on the standby and start managed recovery

Enable flashback **before** starting managed recovery (flashback cannot be enabled while MRP is active).

```
# 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` enables real-time apply (redo applied as it lands in SRLs).

Step 6: Enable redo shipping on the primary

> `LOG_ARCHIVE_DEST_2` was deliberately set to `DEFER` in Step 2: Standby: minimal init.ora pfile, password file, NOMOUNT to suppress continuous `ORA-12154` errors during standby creation. Enable it now that the standby is ready.

Switch `LOG_ARCHIVE_DEST_STATE_2` to `ENABLE` and force a log switch so the first round of redo ships immediately:

```
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.
```

If `dest_2` shows `ORA-12154`, bounce the primary. After Step 1: Enable the broker on both databases, manage transport via DGMGRL.

Step 7: Verify the target Data Guard state

On the **primary** (`oracdb1`):

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

On the **standby** (`oracdb2` — Cloud Console **SSH**, or IAP from your workstation):

```
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
```

Expected on the standby: `PHYSICAL STANDBY | MOUNTED`; `MRP0` with `APPLYING_LOG`.

If the standby reports `MOUNTED` but apply is not running, restart MRP on `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'
```

Step 8: Register the standby database with Oracle Restart

Register the standby with Restart after GCNV seeding so reboots recover ASM, MOUNT, and apply.

On `oracdb2`, capture the spfile location and register with Oracle Restart (substitute `<STANDBY_SPFILE_PATH>` from the query, often under `+DATA`):

```
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
'
```

On `oracdb1`, verify the primary Oracle Restart database resource lists ASM disk group dependencies. Add `RECO` if DBCA registered only `DATA` and `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
'
```

Add the same application service on the standby database resource (`orcls` on `oracdb2`). Use `role PRIMARY` on both sides so `orclapp` is available after switchover:

```
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
'
```

On `oracdb2`, confirm the standby database resource:

```
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
'
```

Configure Data Guard Broker, FSFO, and Observer

After `ENABLE CONFIGURATION`, manage transport and roles through **DGMGRL** (not ad-hoc `LOG_ARCHIVE_DEST_*` SQL).

Step 1: Enable the broker on both databases

On the **primary** and **standby** database hosts:

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

On the **primary database host**, connect with OS authentication (the Observer wallet from Step 5: Start the Observer as a systemd unit is not required on DB hosts):

```
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 /
'
```

On the Observer host only, use `dgmgrl /@orcl` after the auto-login wallet exists. Do not put passwords on the `dgmgrl` command line.

```
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.
```

Run `VALIDATE DATABASE` immediately — it surfaces orphaned datafiles, missing SRLs, redo apply not running, and other common gotchas before you attempt a switchover. Any `WARNING` or non-NULL `ERROR` must be fixed before Step 3: Configure FSFO properties and enable.

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

Step 2: Confirm flashback (required for FSFO auto-reinstate)

Confirm `flashback_on` on **both** hosts before enabling FSFO:

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

On the **primary** only, if retention is not already set:

```
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'
```

Step 3: Configure FSFO properties and enable

```
-- 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.
```

Step 4: Install Oracle Instant Client on the Observer host

```
# 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
```

Step 5: Start the Observer as a systemd unit

Credentials live in an auto-login wallet — never on a `dgmgrl` command line (visible to `ps` / `journalctl`). Connect with `/@<tns_alias>` on the Observer only.

- **Use a dedicated account:** Store credentials for a dedicated Data Guard administrative account (for example, `SYSDG`) in the wallet rather than `SYS`.
- **Auto-login wallet required:** The Observer systemd service requires an auto-login wallet (`cwallet.sso`). If `cwallet.sso` is missing after running `mkstore`, use `orapki` from the Instant Client **tools** package or a database home to create the auto-login wallet, then re-add the stored credentials.

1. Create the wallet on the Observer with credentials for both members:

```
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 unit + log rotation:

```
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
```

Verify from the primary — Observer must read `CONNECTED` (a `DISCONNECTED` Observer silently suspends FSFO):

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

Step 6: Test FSFO (switchover and failover)

Planned switchover:

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

Unplanned failover: Use a VM **Reset** (crash-style test) in a test window; a normal **Stop** may not trigger FSFO. Tail `/var/log/dgmgrl-observer.log` on `oradg-obs`; restore topology when done.

Copyright information

Trademark information