Deploy Google Cloud NetApp Volumes with Oracle HA

NetApp database solutions

NetApp
June 25, 2026

Table of Contents

Deploy Google Cloud NetApp Volumes with Oracle HA

Provision Google Compute Engine instances for Google Cloud NetApp Volumes

Provision Google Compute Engine virtual machines to host Oracle Database 26ai on Google Cloud NetApp Volumes iSCSI storage. This procedure covers creating the primary and standby database hosts and the Fast-Start Failover Observer VM, configuring VPC firewall rules for Oracle Net, setting up hostname resolution, preparing the OS, and capturing iSCSI initiator names for GCNV storage provisioning.

Step 1: Create the VMs

Create three Google Compute Engine VMs in different zones of the same region for zonal failure isolation. Use the Cloud Console, `gcloud`, Terraform, or your standard provisioning workflow.

1. Create the three VMs with the specifications shown in the table below.

   Prefer a lower-carbon region for TCO and sustainability where it meets latency and compliance needs (for example `us-west1` vs `us-central1`):

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
`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)

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

2. Enable Shielded VM features and verify the boot disk configuration:

   Enable **Secure Boot**, **vTPM**, and **Integrity Monitoring** on all three VMs.

   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: Configure VPC firewall for TCP 1521

Create VPC firewall rules to allow TCP/1521 between all three VMs for Oracle Net redo transport and Observer connectivity. Missing rules break Data Guard replication.

1. Create a VPC firewall ingress rule to allow TCP/1521 between all three VM internal IPs. Use VPC firewall rules or Firewall Policies with the same allowlist:

   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.

2. Validate connectivity from each VM to verify the firewall rules are in place:

   ```
   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: Configure hostname, DNS, and `/etc/hosts`

Configure hostname and DNS resolution on all three VMs so forward and reverse name resolution works for Oracle Net, the Data Guard Broker, and the Observer.

1. Set the hostname and add `/etc/hosts` entries on all three hosts. Substitute the GCE internal IP addresses (visible in the **Compute Engine** → **VM instances** list, *Internal IP* column):

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

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

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

2. Validate name resolution from each host:

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

Step 4: Prepare the OS on DB hosts only

Prepare the OS on `oracdb1` and `oracdb2` for Oracle Database 26ai by installing the preinstall package, creating users and groups, installing iSCSI and multipath packages, and configuring the iSCSI initiator. Observer setup is covered in Step 4: Install Oracle Instant Client on the Observer host.

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

1. Install the Oracle Database preinstall package, create the `grid` user and ASM groups, and add the `oracle` user to ASM groups:

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

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

2. Install iSCSI, multipath, and JDK packages, then verify THP and time synchronization:

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

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

3. Configure SELinux, firewall, and iSCSI initiator settings, then reboot:

> **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

Capture the iSCSI initiator name (IQN) from each database host after the reboot. You will use these IQNs to create the GCNV host groups in Step 2: Create the host groups.

1. Capture the IQN from `oracdb1` and record it:

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

2. Repeat on `oracdb2` and record its IQN. Use 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`).

What's next?

To provide shared storage for Oracle binaries and ASM disk groups, go to Provision Google Cloud NetApp Volumes iSCSI pools, host groups, and volumes.

Provision Google Cloud NetApp Volumes iSCSI storage for Oracle Database 26ai

Provision Google Cloud NetApp Volumes iSCSI block storage for Oracle Database 26ai high availability on Google Compute Engine. This procedure covers creating GCNV Flex Unified storage pools, defining host groups, creating iSCSI volumes for each database host, configuring Linux iSCSI and multipath, partitioning ASM backing devices, and mounting the `/u01` filesystem.

Step 1: Create GCNV iSCSI pools

Create two Flex Unified storage pools, one in each database zone, to provide iSCSI volumes for the primary and standby hosts. Each database host uses volumes from its local zone's pool.

1. Create two storage pools using the Cloud Console. Use the specifications in the table below and repeat the creation process 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)

2. Wait for both pools to reach `READY` status before proceeding. 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 host groups

Create one host group per database host so each VM sees only its own volumes. The primary and standby hosts must not share GCNV iSCSI volumes to maintain independent storage.

1. Create the host group for `oracdb1` using the Cloud Console:

   NetApp Volumes → Host groups → Create

   - **Name:** `oracdb1-hg`
   - **Region:** `us-west1`
   - **Type:** iSCSI initiator
   - **OS type:** Linux
   - **Hosts:** paste the IQN from `oracdb1` (the value of `/etc/iscsi/initiatorname.iscsi`)
   - **Description:** "Oracle primary host oracdb1"
   - **Create**

2. Repeat the process for `oracdb2` with the name `oracdb2-hg` and `oracdb2's IQN. The Observer host requires no GCNV resources.

Step 3: Create GCNV iSCSI volumes

Create five GCNV iSCSI volumes for each database host: one for `/u01` and four for ASM backing devices. Each host's volumes must be created in its local zone's storage pool with its corresponding host group.

1. Create the five volumes for `oracdb1` in `oracle-pool-a` with host group `oracdb1-hg`. Use the specifications in the table below:

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.

2. Create the five volumes for `oracdb2` in `oracle-pool-b` with host group `oracdb2-hg` using the same

specifications. For each pool, use **NetApp Volumes** → **Volumes** → **Create** — iSCSI, correct pool and host group, Linux. Record the following information:

- 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 iSCSI and multipath

Configure iSCSI and device-mapper-multipath on each database host to access the GCNV volumes through both storage portal IPs. Run these steps on `oracdb1` using the primary pool's portal IPs, then repeat on `oracdb2` using the standby pool's portal IPs. 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
```

2. Configure `device-mapper-multipath` with defaults and blacklist rules:

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

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

3. 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 devices

Partition the four ASM backing devices (excluding `u01`) with one GPT partition each for ASM consumption, then configure udev rules for grid ownership. Run these steps on each database host.

1. Partition the four ASM backing devices with GPT and verify the partitions:

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

2. Configure udev rules to assign grid ownership and trigger the changes:

```
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`

Format the `ora_<host>_u01` GCNV volume with XFS and mount it persistently using UUID in `/etc/fstab`. The `/u01` filesystem holds Grid home, Oracle home, and staging files.

1. Format the multipath device with XFS and capture its UUID:

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

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

2. Add the UUID-based mount entry to `/etc/fstab` and mount the filesystem:

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

3. Create the directory structure with proper ownership for Grid and Oracle software:

```
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 installing Oracle software.

What's next?

To install Oracle Grid Infrastructure and Database binaries on prepared hosts, go to Install the Oracle Grid Infrastructure and Oracle Database software on both hosts.

Install Oracle Grid Infrastructure and Oracle Database 26ai on Google Cloud NetApp Volumes

Install Oracle Grid Infrastructure with Oracle Restart and ASM on Google Cloud NetApp Volumes iSCSI storage for each database host, then install Oracle Database 26ai software. This procedure includes staging Oracle GoldImages, running silent installations with response files, creating ASM disk groups on GCNV volumes, and preparing both primary and standby hosts with identical Oracle software before database creation.

Step 1: Install Grid Infrastructure on each DB host

Install the Oracle Grid Infrastructure GoldImage on each database host to enable Oracle Restart and ASM. Both hosts require their own Grid home, ASM instance, and disk groups; Data Guard replicates data over Oracle Net, not over shared storage. Complete all steps on `oracdb1` before repeating on `oracdb2`.

1. Stage the Oracle GoldImages, Release Update, and OPatch binaries in `/u01/stage`:

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

2. Unzip the Grid GoldImage in place at the target Grid home. The 26ai GoldImage installs by unzipping directly into the target directory:

```
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
```

If the Grid GoldImage is older than the target RU, patch the Grid home during setup using the

`gridSetup.sh -applyRU` flow, or use a GoldImage with the RU bundled. Keep Grid and Database homes at the same intended patch level.

3. Build the `gridSetup` response file `/tmp/grid.rsp` on each host. Substitute the hostname and use strong passwords:

```
HOST=$(hostname -s)

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

4. Run `gridSetup.sh` in silent mode to copy the binaries and stage the configuration. Expect `Successfully Setup Software with warning(s).` and exit code 6 (warnings) or 0:

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

5. Run `orainstRoot.sh` and `root.sh` as root. The `root.sh` script creates the `crsctl`, `srvctl`, and `asmcmd` wrappers and brings up OHAS:

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

6. Run `gridSetup.sh -executeConfigTools` to run the configuration assistants (NETCA, ASMCA, CVU) against the response file. This creates the ASM instance and the `+DATA` disk group. Expect `Successfully Configured Software.` after NETCA / ASMCA / CVU:

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

7. Create the `+RECO` and `+FRA` disk groups using `asmca`. The single-shot install only creates `+DATA`:

```
HOST=$(hostname -s)

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

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

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

8. Verify the ASM disk groups and Oracle Restart resource status:

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

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

9. Repeat the steps above on `oracdb2`. The `HOST=$(hostname -s)` pattern in steps 3 and 4 and step 7 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 on each DB host

Install the Oracle Database 26ai software home on each database host using a silent, software-only installation with the latest Release Update applied. Complete all steps on `oracdb1` before repeating on `oracdb2`.

1. Unzip the Database home, the latest OPatch, and the RU patch into their respective directories. See the Oracle documentation for RU directory layout and the `-applyRU` path:

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

2. Write the install response file and run the silent software-only installation with the RU applied. On OL 8/9, omit `-applyOneOffs` from the `runInstaller` line:

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

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

3. Run the post-install root script:

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

4. Set the Oracle environment on each DB host. Use `ORACLE_SID=orcl` on `oracdb1` and `ORACLE_SID=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
```

The standby database is created in Create the standby database.

What's next?

To create the production primary instance for your HA deployment, go to Create the Oracle primary database on `oracdb1`.

Create the Oracle primary database on Google Cloud NetApp Volumes

Create the Oracle primary database on Google Cloud NetApp Volumes iSCSI storage using Oracle Database Configuration Assistant in silent mode. This procedure covers running `dbca` to create the container database and pluggable database on GCNV-backed ASM disk groups, configuring archive log destinations, and adding a role-based application service for transparent failover after Data Guard is enabled.

Steps

Create the Oracle container database and pluggable database on `oracdb1` using `dbca` in silent mode, configure archive log destinations, verify Oracle Restart registration, and add a role-based application service for transparent client failover.

1. Run `dbca` in silent mode to create the CDB and PDB on 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
'
```

2. Point archivelogs at `+RECO` and open and save the pluggable database state. The standby uses matching archivelog settings in Step 2: Standby init.ora, pfile, and 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
'
```

3. Verify the database is running 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
```

4. Create a role-based application service so applications connect via `orclapp` and failover is transparent when Data Guard is enabled:

```
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 the Data Guard Broker is enabled, `orclapp` runs only on the PRIMARY. Multiplex control files across ASM disk groups and size memory to workload.

What's next?

To establish standby protection and readiness for failover, go to Create the Oracle standby database on `oracdb2`.

Create the Oracle standby database with Google Cloud NetApp Volumes storage-layer seeding

Create the Oracle physical standby database using Google Cloud NetApp Volumes storage-layer replication, snapshots, or clones to accelerate standby initialization compared to traditional RMAN methods. This procedure covers configuring the listener, creating the standby pfile, seeding standby volumes with GCNV replication, finalizing the Oracle instance, and registering the standby with Oracle Restart. All HA tiers complete these steps. For **Prod HA (Data Guard + FSFO)** tier, continue with Data Guard finalization before configuring Data Guard Broker, Fast-Start Failover, and the Observer.

Step 1: Configure listener and Data Guard parameters

Configure the listener on both database hosts to support Data Guard connections, including the `_DGMGRL` service required for the broker. Set up the password file and configure archive log parameters on the primary database.

1. Configure the primary listener and verify the environment on `oracdb1`:

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

2. Configure the standby listener on `oracdb2` to include the `orcls` and `orcls_DGMGRL` services:

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

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

3. Restart the listener through Oracle Restart on both hosts and verify the `_DGMGRL` service is registered:

```
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: Prepare standby pfile and NOMOUNT

Prepare the standby database instance by copying the password file from the primary, creating a minimal init.ora pfile with Data Guard parameters, and starting the instance in NOMOUNT mode.

1. Copy the primary password file to the standby host using IAP and `gcloud compute scp`:

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

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

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

2. Query the `compatible` parameter value from the primary database:

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

3. Create the standby pfile on `oracdb2`, set ownership on the password file, and start the instance in NOMOUNT mode. Substitute the `compatible` value from the previous step for `<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
'
```

The standby instance is now in NOMOUNT mode with no datafiles until Step 3: Initialize standby storage with GCNV.

Step 3: Initialize standby storage with GCNV

Seed the standby volumes in `oracle-pool-b`, attach them to `oracdb2`, mount ASM disk groups, and finalize the standby instance at MOUNT state.

Use GCNV replication for production seeding, and use snapshot seeding for one-time lab workflows.

Choose the seeding path

Choose the standby seeding method based on your environment and recovery requirements.

- **Recommended for production:** Use the replication path in Replication path: create and sync replications and Replication path: cut over and attach standby volumes.
- **Alternative for labs:** Use Alternate path: seed from snapshot.

All paths rejoin at Mount standby ASM disk groups and Finalize the standby instance.

Check prerequisites

Confirm the following prerequisites before you seed standby volumes.

- `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.
- On `oracdb2`, complete iSCSI and ASM host setup from Step 4, Step 5, and Step 6.

```
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 a 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 and sync replications

Create replication relationships from primary volumes to standby volumes, then wait for initial synchronization to finish.

```
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` and initial sync is complete for each replication.

Cut over and attach standby volumes

Quiesce the primary, stop replication after the final sync, and attach destination volumes to the standby host group.

On the primary, quiesce writes and capture recovery metadata:

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

Allow one final replication cycle, then stop replications:

```
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 LUNs may keep source names):

```
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 the standby controlfile to `oracdb2`, then end backup mode on the primary:

```
ALTER DATABASE END BACKUP;
```

Seed from snapshot

Use this path for one-time lab seeding when continuous replication is not required.

For a one-time lab seed, create a source snapshot and create standby volumes from that snapshot in `oracle-pool-b` (Cloud Console or API). Attach created volumes to `oracdb2-hg`, then continue with Mount standby ASM disk groups.

Mount standby ASM disk groups

On the standby host, discover the attached storage paths and mount the ASM disk groups before database recovery.

On `oracdb2`, log in to standby-pool iSCSI portals and rescan multipath devices. If ASM disk headers match primary naming in a lab workflow, use primary-style aliases (for example `ora_oracdb1_data_01`, `ora_oracdb1_arch_01`), set `asm_diskstring='/dev/mapper/ora_oracdb1_*p*'`, and confirm partition ownership is `grid:asmadmin`, then mount disk groups:

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

Finalize the standby instance

Restore the standby controlfile, recover to the captured SCN, convert to physical standby, and start managed recovery.

```
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;
```

At this point, the standby should be `PHYSICAL STANDBY` and `MOUNTED` with managed recovery started.

Tier-specific next steps:

- **Prod HA (no Data Guard):** Continue directly to Step 4: Register standby with Oracle Restart.
- **Prod HA (Data Guard + FSFO):** Continue to Step 4: Register standby with Oracle Restart, then proceed to Data Guard finalization steps.

Step 4: Register standby with Oracle Restart

Register the standby database with Oracle Restart so that reboots automatically recover ASM disk groups, mount the standby database, and restart managed recovery. Also add the application service to both database resources.

1. Capture the spfile location from the standby database and register it with Oracle Restart on `oracdb2`. 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
'
```

2. Verify and update the primary database resource on `oracdb1` to include all ASM disk group dependencies:

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

3. Add the application service to 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
'
```

4. Verify the standby database resource on `oracdb2`:

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

What's next?

Tier-specific:

- **Prod HA (no Data Guard):** To maintain a storage-replication-based recovery target, standby initialization is complete and the standby database is registered with Oracle Restart as a backup instance.
- **Prod HA (Data Guard + FSFO):** To enable broker-managed switchover and fast-start failover, continue with Finalize the standby database for Data Guard.

Finalize the standby database for Data Guard on Google Cloud NetApp Volumes

Finalize the standby database for Oracle Data Guard on Google Cloud NetApp Volumes by creating standby redo log files, enabling flashback database, activating redo shipping, and verifying Data Guard state.

Tier-specific: This procedure is required only for **Prod HA (Data Guard + FSFO)** tier.

Step 1: Create standby redo log files

Create standby redo log files on both database hosts to support Fast-Start Failover. Size must be greater than or equal to the largest primary online redo log, and count should equal (online groups per thread) + 1. After GCNV seeding, drop and recreate standby redo logs on the standby to fix replicated paths.

1. Create standby redo log files on the primary database (`orcl`):

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

2. Drop and recreate standby redo log files on the standby database (`orcls`) after GCNV seeding. Replicated paths under `+DATA/ORCL/…` cause `ORA-19527` / `ORA-16086` until rebuilt:

```
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 2: Enable flashback and start recovery

Enable flashback database on the standby to support automatic reinstatement after failover, then start managed recovery with real-time apply. Flashback must be enabled before starting managed recovery because it cannot be enabled while MRP is active.

1. Shutdown the standby database, restart in MOUNT mode, and enable flashback database on `oracdb2`:

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

2. Start managed recovery with real-time apply:

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

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

Step 3: Enable redo shipping

Enable redo transport from the primary to the standby by activating `LOG_ARCHIVE_DEST_STATE_2`, which was deliberately set to `DEFER` in Step 2 of the standby initialization procedure to suppress `ORA-12154` errors during standby creation.

1. Switch `LOG_ARCHIVE_DEST_STATE_2` to `ENABLE` and force a log switch to initiate redo shipping:

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

2. Verify redo shipping is working correctly:

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

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

Step 4: Verify Data Guard state

Verify that the primary database is in READ WRITE mode and the standby database is mounted with managed recovery applying redo logs.

1. Verify the primary database role and open mode on `oracdb1`:

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

2. Verify the standby database role, open mode, and managed recovery status on `oracdb2`:

```
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`.

3. If the standby reports `MOUNTED` but apply is not running, restart managed recovery 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'
```

What's next?

To activate automated role management and failover protection, continue with Configure Oracle Data Guard Broker, Fast-Start Failover, and the Observer.

Configure Data Guard Broker and Fast-Start Failover for Oracle Database 26ai on Google Cloud NetApp Volumes

Configure Oracle Data Guard Broker and Fast-Start Failover with a dedicated Observer to enable automatic role transitions for Oracle Database 26ai on Google Cloud NetApp Volumes.

Tier-specific: This procedure applies only to the **Prod HA (Data Guard + FSFO)** tier.

This procedure covers enabling the broker on both databases, creating the Data Guard configuration, enabling FSFO with MaxAvailability protection mode, installing Oracle Instant Client on the Observer host, starting the Observer as a systemd service with wallet-based credentials, and testing switchover and failover. After `ENABLE CONFIGURATION`, manage transport and roles through **DGMGRL** (not ad-hoc `LOG_ARCHIVE_DEST_*` SQL).

Step 1: Enable Data Guard Broker

Enable the Data Guard Broker on both database hosts and create the broker configuration that links the primary and standby databases under unified management.

1. Set `dg_broker_start=TRUE` 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'
```

2. On the primary, connect to DGMGRL with OS authentication and create the broker configuration:

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

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

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

3. Validate the configuration — fix any `WARNING` or non-NULL `ERROR` before Step 3: Configure FSFO properties and enable:

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

Step 2: Confirm flashback for FSFO

Confirm that flashback database is enabled on both hosts. Flashback is required for FSFO auto-reinstate, which allows the former primary to automatically rejoin the configuration as a standby after a failover.

1. Confirm `flashback_on` is `YES` on both database hosts:

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

2. On the primary only, if flashback 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 and enable FSFO

Set SYNC redo transport, configure MaxAvailability protection mode, define FSFO targets on each database, and enable Fast-Start Failover.

1. Set the redo transport mode to `SYNC` on both databases and raise the protection mode to `MaxAvailability`:

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

2. Set the FSFO targets so each database names the other as its failover target, then configure the threshold and auto-reinstate behavior:

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

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

3. Enable Fast-Start Failover and confirm the configuration:

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

Step 4: Install Instant Client on Observer

Install Oracle Instant Client on the dedicated Observer VM (`oradg-obs`), create a dedicated `oracle` OS user, and configure the Oracle Net environment so the Observer can connect to both database members on TCP/1521.

1. Install Oracle Instant Client packages on the Observer host (`oradg-obs`):

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

2. Create a dedicated `oracle` OS user that will own the wallet and the systemd unit:

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

3. Configure the Oracle Net environment and create `tnsnames.ora` with entries for both database hosts:

```
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: Run Observer as a systemd service

Create an auto-login wallet with credentials for both database members, then configure and start the Observer as a systemd service so it survives reboots and automatically reconnects to the configuration.

Store credentials for a dedicated Data Guard administrative account (for example, `SYSDG`) in the wallet rather than `SYS`. Credentials must never appear on a `dgmgrl` command line, where they are visible to `ps` and `journalctl`; always connect using `/@<tns_alias>` on the Observer.

1. Create the encrypted wallet and populate credentials for both database 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;'
```

2. Generate the auto-login wallet (`cwallet.sso`) so the Observer systemd service can start without a password prompt. If `cwallet.sso` is missing after running `mkstore`, use `orapki` from the Instant Client tools package or a database home to create it, then re-add the stored credentials:

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

3. Create the systemd unit, enable the service, and verify the Observer is connected:

```
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
```

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

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

Step 6: Test FSFO

Validate the Data Guard configuration with `VALIDATE DATABASE`, then perform a planned switchover and, in a test window, an unplanned VM-reset failover to confirm FSFO is working end to end.

1. Test a planned switchover and restore the original topology:

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

2. Test an unplanned failover using a VM reset in a controlled test window:

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

What’s next?

Oracle Data Guard Broker, Fast-Start Failover, and Observer configuration is now in place for this deployment.

Copyright information

Trademark information