Oracle HA를 지원하는 Google Cloud NetApp Volumes 배포

NetApp database solutions

NetApp
June 25, 2026

목차

Oracle HA를 지원하는 Google Cloud NetApp Volumes 배포

Google Cloud NetApp Volumes용 Google Compute Engine 인스턴스 프로비저닝

Google Cloud NetApp Volumes iSCSI 스토리지에 Oracle Database 26ai를 호스팅하기 위한 Google Compute Engine 가상 머신을 프로비저닝합니다. 이 절차에서는 기본 및 대기 데이터베이스 호스트와 Fast-Start Failover Observer VM 생성, Oracle Net용 VPC 방화벽 규칙 구성, 호스트 이름 확인 설정, 운영 체제 준비, GCNV 스토리지 프로비저닝을 위한 iSCSI 이니시에이터 이름 캡처에 대해 설명합니다.

1단계: VM 생성

영역별 장애 격리를 위해 동일 리전의 서로 다른 영역에 Google Compute Engine VM 3개를 생성합니다. Cloud Console, `gcloud`, Terraform 또는 표준 프로비저닝 워크플로를 사용하십시오.

1. 아래 표에 제시된 사양으로 세 개의 가상 머신을 생성하십시오.

   지연 시간 및 규정 준수 요구 사항을 충족하면서 총소유비용(TCO) 및 지속 가능성 측면에서 탄소 배출량이 적은 지역을 선호합니다(예: `us-west1` vs `us-central1`):

VM	영역	시스템 유형	부팅 디스크	네트워크	목적
`oracdb1`	`us-west1-a`	`n4-highmem-8` (샘플) 또는 `c4-standard-*`	OL 10, 50 GB Hyperdisk Balanced(OS 전용)	`oracle-vpc` / `oracle-subnet`, gVNIC	기본 DB
`oracdb2`	`us-west1-b`	기본과 동일	OL 10, 50 GB Hyperdisk Balanced(OS 전용)	동일	대기 DB
`oradg-obs`	`us-west1-c`	`e2-medium`	OL 10, 20GB Hyperdisk Balanced	동일	FSFO Observer(Instant Client 전용)

   지연 시간이나 송출량(월 ~200GiB 초과)이 중요한 경우에는 프리미엄 네트워크 등급을 사용하고, 개발/테스트 환경에서 총소유비용(TCO)을 낮추려면 Standard 등급을 사용하십시오.

2. 보호된 VM 기능을 활성화하고 부팅 디스크 구성을 확인합니다.

   세 개의 가상 머신 모두에서 **Secure Boot**, **vTPM** 및 *Integrity Monitoring*을 활성화하십시오.

   부팅 디스크에는 OS만 저장됩니다. `/u01`, Grid/DB 홈, 스테이징 및 모든 ASM 데이터는 GCNV iSCSI 볼륨을 사용합니다(GCNV iSCSI 볼륨 프로비저닝 참조).

을 위해 별도의 GCE 데이터 디스크를 연결하지 마십시오 `/u01`.

2단계: TCP 1521에 대한 VPC 방화벽 구성

Oracle Net 리두 전송 및 옵저버 연결을 위해 세 대의 VM 간에 TCP/1521 포트를 허용하는 VPC 방화벽 규칙을 생성하십시오. 규칙이 누락되면 Data Guard 복제가 실패합니다.

1. VPC 방화벽에 TCP/1521 포트를 허용하는 인바운드 규칙을 생성하여 세 개의 VM 내부 IP 주소 간에 연결을 허용합니다. 동일한 허용 목록을 사용하는 VPC 방화벽 규칙 또는 방화벽 정책을 사용하십시오.

   Cloud Console: VPC 네트워크 → 방화벽 → 규칙 생성 `allow-oracle-net-dbhosts` on `oracle-vpc` — 수신, 허용, 소스 = 세 개의 `/32` IP, TCP 1521. 필요한 경우 송신도 동일하게 설정합니다.

2. 각 VM의 연결을 검증하여 방화벽 규칙이 제대로 적용되었는지 확인합니다.

```
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
```

포트	예상	의미
`22`	연결됨	SSH 경로가 작동합니다
`1521`	연결이 거부되었습니다	방화벽 열림, Grid 리스너가 1단계: 각 DB 호스트에 Oracle Grid Infrastructure(Oracle Restart)를 설치합니다 중에 시작됨
둘 중 하나	시간 초과	방화벽 또는 라우팅 수정

   세 개의 VM 모두에서 각 피어 IP를 향해 실행합니다.

3단계: 호스트 이름, DNS 및 `/etc/hosts` 구성

Oracle Net, Data Guard Broker 및 Observer에 대한 정방향 및 역방향 이름 확인이 작동하도록 세 대의 VM 모두에서 호스트 이름 및 DNS 확인을 구성하십시오.

1. 호스트 이름을 설정하고 `/etc/hosts` 세 호스트 모두에 항목을 추가합니다. GCE 내부 IP 주소(**Compute Engine** → **VM** 인스턴스 목록의 내부 *IP* 열에 표시됨)를 입력합니다.

```
# 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. 각 호스트에서 이름 확인을 검증합니다.

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

4단계: DB 호스트에서만 OS 준비

사전 설치 패키지 설치, 사용자 및 그룹 생성, iSCSI 및 다중 경로 패키지 설치, iSCSI 이니시에이터 구성을 통해 `oracdb1` 및 `oracdb2`의 OS를 Oracle Database 26ai에 맞게 준비합니다. 옵저버 설정은 4단계: Observer 호스트에 Oracle Instant Client 설치에서 다룹니다.

필수 조건: `yum.oracle.com`에 대한 아웃바운드 HTTPS(프라이빗 서브넷의 Cloud NAT 또는 내부 미러).

1. Oracle Database 사전 설치 패키지를 설치하고, `grid` 사용자 및 ASM 그룹을 생성한 다음, `oracle` 사용자를 ASM 그룹에 추가합니다.

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

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

2. iSCSI, 다중 경로 및 JDK 패키지를 설치한 다음 THP 및 시간 동기화를 확인하십시오.

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

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

3. SELinux, 방화벽 및 iSCSI 이니시에이터 설정을 구성한 다음 재부팅하십시오.

보안 상태**(OL 10):** 아래 명령은 SELinux를 허용 모드로 설정하고 `firewalld`를 비활성화합니다. 이는 최소한의 랩 환경을 위한 설정일 뿐입니다. 강화된 SELinux 및 방화벽 구성은 조직의 보안 기준을 참조하십시오.

```
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
```

5단계: iSCSI 이니시에이터 이름 IQN 캡처

재부팅 후 각 데이터베이스 호스트에서 iSCSI 이니시에이터 이름(IQN)을 캡처하십시오. 이 IQN을 사용하여 2단계: 호스트 그룹 생성에서 GCNV 호스트 그룹을 생성합니다.

1. IQN을 `oracdb1`에서 캡처하여 기록하십시오.

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

2. 이 과정을 반복 `oracdb2`하고 해당 IQN을 기록하십시오. 호스트당 하나의 호스트 그룹을 사용하여 단일 호스트의 재부팅 또는 IQN 재생성이 다른 호스트의 GCNV iSCSI 볼륨 가시성에 영향을 미치지 않도록 하십시오.

복제된 **VM:** 두 호스트가 동일한 IQN을 공유하는 경우 `oracdb2`에서 다시 생성하십시오(중지 `iscsi`, 초기화 `/var/lib/iscsi/nodes/*`, 새로 생성 `InitiatorName` `/etc/iscsi/initiatorname.iscsi`, 재시작 `iscsid`).

다음 단계

Oracle 바이너리 및 ASM 디스크 그룹에 대한 공유 스토리지를 제공하려면 Google Cloud NetApp Volumes iSCSI 풀, 호스트 그룹 및 볼륨 프로비저닝으로 이동하십시오.

Oracle Database 26ai용 Google Cloud NetApp Volumes iSCSI 스토리지 프로비저닝

Google Compute Engine에서 Oracle Database 26ai의 고가용성을 위해 Google Cloud NetApp Volumes iSCSI 블록 스토리지를 프로비저닝합니다. 이 절차에서는 GCNV Flex Unified 스토리지 풀 생성, 호스트 그룹 정의, 각 데이터베이스 호스트에 대한 iSCSI 볼륨 생성, Linux iSCSI 및 다중 경로 구성, ASM 백업 장치 파티셔닝, `/u01`파일 시스템 마운트 등을 다룹니다.

1단계: GCNV iSCSI 풀 생성

기본 호스트와 대기 호스트에 iSCSI 볼륨을 제공하기 위해 각 데이터베이스 영역에 Flex Unified 스토리지 풀을 하나씩, 총 두 개를 생성합니다. 각 데이터베이스 호스트는 해당 로컬 영역의 풀에서 볼륨을 사용합니다.

1. 클라우드 콘솔을 사용하여 두 개의 스토리지 풀을 생성합니다. 아래 표의 사양을 사용하고 각 영역에 대해 생성 프로세스를 반복합니다.

풀 이름	영역	사용 대상
`oracle-pool-a`	`us-west1-a`	`oracdb1` (운영)
`oracle-pool-b`	`us-west1-b`	`oracdb2` (대기)

   NetApp Volumes → 스토리지 풀 → 각 풀에 대해 생성

   - 서비스 수준**:** Flex(Premium 아님)
   - 유형**:** Unified
   - **Zone:** 데이터베이스 VM 영역과 일치(`us-west1-a` / `us-west1-b`)
   - **PSA:** 다음에 연결됨 `oracle-vpc`
   - 용량**:** 워크로드에 맞춰 크기가 조정됩니다. 재실행, 백업 또는 복원 작업이 기본 여유 용량(풀당 최대 5120 MiB/s 또는 160K IOPS, 제품 제한)을 초과하는 경우 사용자 지정으로 프로비저닝된 처리량/IOPS를 사용합니다.

2. 두 풀이 모두 `READY` 상태에 도달할 때까지 기다린 후 진행하십시오. 데이터베이스 사용량에 맞게 풀 크기를 조정하십시오(3단계: GCNV iSCSI 볼륨 생성의 크기는 예시입니다).

기본 모드(이 가이드): Flex Unified 풀은 기본 모드(`--mode=default`를 사용합니다. Cloud Console 또는 `gcloud netapp`를 사용하여 풀 및 iSCSI 볼륨을 생성합니다. 볼륨 복제, 스냅샷 및 클론은 Google Cloud API(3단계: GCNV 대기 초기화)를 사용합니다.

2단계: 호스트 그룹 생성

각 VM이 자체 볼륨만 볼 수 있도록 데이터베이스 호스트별로 호스트 그룹을 하나씩 생성하십시오. 독립적인 스토리지를 유지하려면 기본 호스트와 대기 호스트가 GCNV iSCSI 볼륨을 공유해서는 안 됩니다.

1. Cloud Console을 사용하여 `oracdb1` 호스트 그룹을 생성합니다.

   NetApp 볼륨 → 호스트 그룹 → 생성

   - 이름: `oracdb1-hg`
   - 지역: `us-west1`
   - 유형: iSCSI 이니시에이터
   - **OS** 유형: Linux
   - 호스트: `oracdb1`(값 `/etc/iscsi/initiatorname.iscsi`)에서 IQN을 붙여넣으세요.
   - 설명: "Oracle 기본 호스트 oracdb1"
   - 만들기

2. `oracdb2`에 대해 이름 `oracdb2-hg` 및 `oracdb2`의 IQN으로 프로세스를 반복합니다. Observer 호스트에는 GCNV 리소스가 필요하지 않습니다.

3단계: GCNV iSCSI 볼륨 생성

각 데이터베이스 호스트에 대해 5개의 GCNV iSCSI 볼륨을 생성합니다. 하나는 `/u01`용이고 나머지 4개는 ASM 백업 장치용입니다. 각 호스트의 볼륨은 해당 호스트 그룹과 함께 로컬 영역의 스토리지 풀에 생성되어야 합니다.

1. `oracdb1`의 5개 볼륨을 `oracle-pool-a`에서 호스트 그룹 `oracdb1-hg`으로 생성하십시오. 아래 표의 사양을 사용하십시오.

GCNV iSCSI 볼륨	크기	사용	다중 경로 별칭
`ora_<host>_u01`	100 GiB	`/u01` GCNV iSCSI 볼륨 — Grid/Oracle 홈, 스테이징	`/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`(스트라이프)	`/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`

볼륨 이름: 문자, 숫자, 밑줄만 사용 가능(하이픈 사용 불가).

최소 레이아웃(검증용): 호스트당 LUN 2개(`*_data`, `*_reco`) `arch_01p1`→`+RECO` 및 `arch_01p2`→`+FRA`는 연구실 환경에서 사용 가능하며, 프로덕션 환경에서는3단계: GCNV iSCSI 볼륨 생성당 5개의 볼륨을 사용합니다.

2. `oracdb2`의 5개 볼륨을 `oracle-pool-b`에서 호스트 그룹 `oracdb2-hg`을 사용하여 동일한 사양으로 생성합니다. 각 풀에 대해 **NetApp Volumes** → **Volumes** → **Create** — iSCSI, 올바른 풀 및 호스트 그룹, Linux를 사용합니다. 다음 정보를 기록합니다.
   - iSCSI 포털 IP → `<ISCSI_PORTAL_1>`, `<ISCSI_PORTAL_2>` (기본 풀 포털은 `oracdb1`에 있고, 대기 풀 포털은 `oracdb2`에 있으며 서로 다를 수 있음)
   - 클라우드 콘솔의 볼륨 일련 번호 — 4단계: GCNV iSCSI 볼륨에 대한 Linux iSCSI 및 다중 경로 구성에서 호스트가 검색한 WWID와 함께 사용

4단계: iSCSI 및 다중 경로 구성

각 데이터베이스 호스트에서 iSCSI 및 device-mapper-multipath를 구성하여 두 스토리지 포털 IP를 통해 GCNV 볼륨에 액세스하도록 합니다. `oracdb1`에서 기본 풀의 포털 IP를 사용하여 이 단계를 실행한 다음, `oracdb2`에서 대기 풀의 포털 IP를 사용하여 동일한 단계를 반복합니다. 호스트 송신이 제한된 경우, 각 데이터베이스 VM에서 GCNV iSCSI 포털 IP로 TCP/3260을 허용해야 합니다(2단계: VPC 방화벽 - 세 영역 모두에서 TCP/1521 허용 목록 추가에서의 VM 간 TCP/1521 포함).

1. 대상 검색, 로그인 및 노드 시작 정보 유지:

```
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
```

   재부팅 후 Oracle을 시작하기 전에 다시 확인하십시오.

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

2. Configure `device-mapper-multipath` 기본 설정값과 블랙리스트 규칙을 사용하여 구성하십시오.

```
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. 호스트에서 발견한 WWID 별칭을 `/etc/multipath.conf`에 추가합니다(추측하지 마십시오 — `multipath.conf`는 셸 변수를 확장하지 않습니다). WWID 검색:

```
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
```

해당 호스트에 대한 구체적인 별칭을 `/etc/multipath.conf`에 추가한 다음 `sudo systemctl restart multipathd`.

`oracdb1`에서 다음을 추가합니다.

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

`oracdb2`에서 `ora_oracdb2_*` 별칭에 동일한 패턴을 사용한 다음:

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

5단계: ASM 장치 분할

4개의 ASM 백업 장치(u01 제외)를 각각 ASM 사용을 위한 GPT 파티션 하나씩으로 분할한 다음, 그리드 소유권에 대한 udev 규칙을 구성하십시오. 각 데이터베이스 호스트에서 이러한 단계를 실행하십시오.

1. 4개의 ASM 백업 장치를 GPT로 파티션하고 파티션을 확인합니다.

```
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. 그리드 소유권을 할당하고 변경 사항을 트리거하도록 udev 규칙을 구성합니다.

```
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
```

6단계: 포맷 및 마운트 /u01

GCNV 볼륨을 ora_<host>_u01 XFS로 포맷하고 /etc/fstab`에서 UUID를 사용하여 영구적으로 마운트합니다. 해당 `/u01 파일 시스템에는 Grid 홈, Oracle 홈 및 스테이징 파일이 저장됩니다.

1. 다중 경로 장치를 XFS로 포맷하고 해당 장치의 UUID를 캡처합니다.

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

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

2. UUID 기반 마운트 항목을 `/etc/fstab`에 추가하고 파일 시스템을 마운트하십시오.

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

3. Grid 및 Oracle 소프트웨어에 적합한 소유권으로 디렉터리 구조를 생성하십시오.

```
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
```

한 번 재부팅하고 `/u01` 마운트가 Oracle 소프트웨어 설치 전에 완료되는지 확인합니다.

다음 단계

준비된 호스트에 Oracle Grid Infrastructure 및 Database 바이너리를 설치하려면 두 호스트 모두에서 Oracle Grid Infrastructure 및 Oracle Database 소프트웨어를 설치합니다.로 이동하십시오.

Google Cloud NetApp Volumes에 Oracle Grid Infrastructure 및 Oracle Database 26ai 설치

각 데이터베이스 호스트용 Google Cloud NetApp Volumes iSCSI 스토리지에 Oracle Restart 및 ASM을 함께 구성한 Oracle Grid Infrastructure를 설치한 다음, Oracle Database 26ai 소프트웨어를 설치합니다. 이 절차에는 Oracle GoldImages 스테이징, 응답 파일을 사용한 자동 설치 실행, GCNV 볼륨에 ASM 디스크 그룹 생성, 그리고 데이터베이스 생성 전에 주 호스트와 대기 호스트 모두에 동일한 Oracle 소프트웨어를 준비하는 과정이 포함됩니다.

1단계: 각 DB 호스트에 Grid Infrastructure 설치

Oracle Restart 및 ASM을 활성화하려면 각 데이터베이스 호스트에 Oracle Grid Infrastructure GoldImage를 설치하십시오. 두 호스트 모두 각각 고유한 Grid 홈, ASM 인스턴스 및 디스크 그룹이 필요합니다. Data Guard는 공유 스토리지가 아닌 Oracle Net을 통해 데이터를 복제합니다. `oracdb1`에서 모든 단계를 완료한 후 `oracdb2`에서 반복하십시오.

1. Oracle GoldImages, 릴리스 업데이트 및 OPatch 바이너리를 `/u01/stage`에 스테이징합니다:

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

2. Grid GoldImage 파일을 대상 Grid 홈 디렉터리에 압축을 풀어 설치합니다. 26ai GoldImage는 대상 디렉터리에 직접 압축을 풀어 설치합니다:

```
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 GoldImage의 버전이 대상 RU보다 오래된 경우, `gridSetup.sh -applyRU` 워크플로를 사용하여 설치 과정에서 Grid 홈에 패치를 적용하거나, RU가 번들로 포함된 GoldImage를 사용하십시오. Grid와 데이터베이스 홈은 의도한 패치 수준을 동일하게 유지하십시오.

3. 각 호스트에서 `gridSetup` 응답 파일 `/tmp/grid.rsp`을 생성하십시오. 호스트명을 대체하고 강력한 비밀번호를 사용하십시오:

```
HOST=$(hostname -s)

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

4. `gridSetup.sh`를 무음 모드로 실행하여 바이너리 파일을 복사하고 구성을 준비합니다. `Successfully Setup Software with warning(s).` 및 종료 코드 6(경고) 또는 0을 예상합니다:

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

5. `orainstRoot.sh` 와 `root.sh` 를 루트 권한으로 실행하십시오. `root.sh` 스크립트는 `crsctl`, `srvctl`, 및 `asmcmd` 래퍼를 생성하고 OHAS를 시작합니다:

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

6. `gridSetup.sh -executeConfigTools` 를 실행하여 구성 어시스턴트(NETCA, ASMCA, CVU)를 응답 파일 에 대해 실행합니다. 이렇게 하면 ASM 인스턴스와 `+DATA` 디스크 그룹이 생성됩니다. NETCA / ASMCA / CVU 실행 후에는 `Successfully Configured Software.` 가 표시될 것으로 예상됩니다:

```
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. `asmca`을 사용하여 `+RECO 및 +FRA 디스크 그룹을 생성합니다. single-shot 설치는 `+DATA`만 생성합니다:

```
HOST=$(hostname -s)

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

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

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

8. ASM 디스크 그룹 및 Oracle Restart 리소스의 상태를 확인하십시오:

```
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. `oracdb2`에서 위의 단계를 반복하십시오. `HOST=$(hostname -s)` 패턴은 3단계와 4단계 및 7단계에서 해당 호스트의 GCNV iSCSI 장치를 자동으로 선택합니다.

   동일한 ASM 디스크 그룹 이름을 사용하십시오. Data Guard는 스토리지가 아닌 Oracle Net을 통해 복제됩니다.

2단계: 각 DB 호스트에 Oracle 데이터베이스 설치

최신 릴리스 업데이트가 적용된 상태에서, 무인 방식의 소프트웨어 전용 설치를 통해 각 데이터베이스 호스트에 Oracle Database 26ai 소프트웨어 홈을 설치하십시오. `oracdb1`에 있는 모든 단계를 완료한 후, `oracdb2`에서 동일한 절차를 반복하십시오.

1. 데이터베이스 홈, 최신 OPatch 및 RU 패치를 각각의 디렉터리에 압축을 풀어주세요. RU 디렉터리 구조 및 `-applyRU` 경로에 대해서는 Oracle 설명서를 참조하십시오:

```
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. 설치 응답 파일을 작성하고, RU가 적용된 상태에서 자동 소프트웨어 전용 설치를 실행합니다. OL 8/9의 경우, `-applyOneOffs` 를 `runInstaller` 줄에서 생략하십시오:

```
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. 설치 후 루트 스크립트를 실행하십시오:

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

4. 각 DB 호스트에서 Oracle 환경을 설정합니다. `oracdb1`에서 `ORACLE_SID=orcl`을 사용하고 `oracdb2`에서 `ORACLE_SID=orcls`을 사용하십시오:

```
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
```

대기 데이터베이스가 대기 데이터베이스를 생성합니다에 생성됩니다.

다음 단계

HA 배포를 위한 프로덕션 기본 인스턴스를 생성하려면 `oracdb1`에서 Oracle 주 데이터베이스 생성(으)로 이동합니다.

Google Cloud NetApp Volumes에 Oracle 기본 데이터베이스를 생성합니다.

Google Cloud NetApp Volumes iSCSI 스토리지에서 Oracle Database Configuration Assistant를 자동 모드로 실행하여 Oracle 기본 데이터베이스를 생성합니다. 이 절차에서는 `dbca`를 실행하여 GCNV 기반 ASM 디스크 그룹에 컨테이너 데이터베이스 및 플러그형 데이터베이스를 생성하고, 아카이브 로그 대상을 구성하며, Data Guard 활성화 후 투명한 장애 조치를 위해 역할 기반 애플리케이션 서비스를 추가하는 방법을 다룹니다.

단계

에서 `oracdb1` Oracle 컨테이너 데이터베이스 및 플러그인 가능 데이터베이스를 `dbca` 를 사용하여 자동 모드로 생성하고, 아카이브 로그 대상을 구성하며, Oracle Restart 등록을 확인하고, 투명한 클라이언트 장애 조치를 위한 역할 기반 애플리케이션 서비스를 추가합니다.

1. ASM 디스크 그룹에 CDB 및 PDB를 생성하려면 자동 모드로 실행하십시오. `dbca`

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

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

2. 아카이브 로그를 다음 위치로 지정 `+RECO`하고 플러그형 데이터베이스 상태를 열어 저장합니다. 스탠바이 서버는 다음 위치에 있는 일치하는 아카이브 로그 설정을 사용합니다.2단계: 대기 init.ora, pfile 및 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. Oracle Restart를 통해 데이터베이스가 실행 중인지 확인하십시오.

```
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. 애플리케이션이 `orclapp`를 통해 연결되고 Data Guard가 활성화된 경우 장애 조치가 투명하게 이루어지도록 역할 기반 애플리케이션 서비스를 생성합니다.

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

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

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

Data Guard Broker가 활성화되면 `orclapp`는 PRIMARY에서만 실행됩니다. ASM 디스크 그룹 전체에 제어 파일을 멀티플렉싱하고 워크로드에 맞게 메모리 크기를 조정합니다.

다음 단계

대기 보호 및 장애 조치 준비 상태를 설정하려면 Oracle 스탠바이 데이터베이스 생성에서 `oracdb2`로 이동하십시오.

Google Cloud NetApp Volumes 스토리지 계층 시딩을 사용하여 Oracle 스탠바이 데이터베이스 생성

Google Cloud NetApp Volumes 스토리지 계층 복제, 스냅샷 또는 클론을 사용하여 Oracle 물리적 스탠바이 데이터베이스를 생성하면 기존 RMAN 방식보다 스탠바이 초기화 속도를 높일 수 있습니다. 이 절차에서는 리스너 구성, 스탠바이 pfile 생성, GCNV 복제를 사용한 스탠바이 볼륨 시드, Oracle 인스턴스 최종화, Oracle Restart를 사용한 스탠바이 등록을 다룹니다. 모든 HA 계층에서 이러한 단계를 완료해야 합니다. 프로덕션 **HA(Data Guard + FSFO)** 계층의 경우, Data Guard 최종화 구성 전에 계속 진행하십시오. Data Guard Broker, Fast-Start Failover 및 Observer

1단계: 리스너 및 Data Guard 매개변수 구성

두 데이터베이스 호스트 모두에서 Data Guard 연결을 지원하도록 리스너를 구성하고, 브로커에 필요한 서비스도 설정하십시오. `_DGMGRL` 기본 데이터베이스에서 암호 파일을 설정하고 아카이브 로그 매개변수를 구성하십시오.

1. 기본 리스너를 구성하고 다음 환경에서 환경을 확인하십시오 `oracdb1`:

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

2. `oracdb2`의 대기 수신기를 구성하여 `orcls` 및 `orcls_DGMGRL` 서비스를 포함하도록 합니다.

```
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. 두 호스트 모두에서 Oracle Restart를 통해 리스너를 다시 시작하고 `_DGMGRL` 서비스가 등록되었는지

확인하십시오.

```
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`은(는) `<SID>` 및 `<SID>_DGMGRL`을(를) 나열해야 합니다.

2단계: 대기 pfile 및 NOMOUNT 준비

기본 데이터베이스에서 암호 파일을 복사하고, Data Guard 매개변수가 포함된 최소한의 init.ora pfile을 생성하고, 인스턴스를 NOMOUNT 모드로 시작하여 대기 데이터베이스 인스턴스를 준비합니다.

1. IAP 및 `gcloud compute scp`를 사용하여 기본 암호 파일을 대기 호스트로 복사합니다.

```
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. 기본 데이터베이스에서 `compatible` 매개변수 값을 조회합니다.

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

3. 대기 pfile을 `oracdb2`에 생성하고, 암호 파일의 소유권을 설정한 다음, 인스턴스를 `NOMOUNT` 모드로 시작하십시오. `compatible` 값을 이전 단계에서 `<COPY_FROM_PRIMARY>`으로 대체하십시오.

```
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
'
```

대기 인스턴스는 이제 3단계: GCNV를 사용하여 대기 스토리지 초기화까지 데이터 파일이 없는 NOMOUNT 모드입니다.

3단계: GCNV를 사용하여 대기 스토리지 초기화

대기 볼륨을 `oracle-pool-b`에 시드하고, `oracdb2`에 연결하고, ASM 디스크 그룹을 마운트한 다음, 대기 인스턴스를 MOUNT 상태로 최종화합니다.

프로덕션 시딩에는 GCNV 복제를 사용하고, 일회성 랩 워크플로우에는 스냅샷 시딩을 사용하십시오.

시딩 경로를 선택합니다

사용 환경 및 복구 요구 사항에 따라 대기 시드 방법을 선택하십시오.

- 프로덕션 환경에 권장: 복제 경로: 복제본 생성 및 동기화 및 복제 경로: 컷오버 및 대기 볼륨 연결에 있는 복제 경로를 사용하십시오.
- 실험실 대안: 대체 경로: 스냅샷에서 시드 생성을(를) 사용합니다.

모든 경로는 대기 ASM 디스크 그룹 마운트와 대기 인스턴스를 완료합니다.에서 다시 합쳐집니다.

필수 조건을 확인합니다

대기 볼륨을 시드하기 전에 다음 필수 조건을 확인하십시오.

- `gcloud netapp` 볼륨 복제 지원.
- 서로 다른 위치에 있는 두 개의 기본 모드 풀 (`oracle-pool-a`, `oracle-pool-b`).
- 기본 풀에 연결된 소스 볼륨 `oracdb1-hg`, 복제를 통해 생성된 대상 볼륨.
- DB VM이 아닌 Cloud Shell 또는 워크스테이션에서 복제를 실행하십시오.
- `oracdb2`에서 4단계, 5단계, 6단계의 iSCSI 및 ASM 호스트 설정을 완료하십시오.

```
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"
```

- 필요한 경우 대기 풀을 생성합니다.

```
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>
```

복제본 생성 및 동기화

기본 볼륨에서 대기 볼륨으로 복제 관계를 생성한 다음 초기 동기화가 완료될 때까지 기다립니다.

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

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

+

`mirrorState`이(가) `MIRRORED`이 될 때까지 기다리고 각 복제에 대한 초기 동기화가 완료될 때까지 기다리십시오.

전환 후 대기 볼륨 연결

기본 서버를 정지시키고, 최종 동기화 후 복제를 중지한 다음, 대상 볼륨을 대기 호스트 그룹에 연결합니다.

기본 서버에서 쓰기 작업을 중지하고 복구 메타데이터를 캡처합니다.

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

마지막 복제 주기를 한 번 더 허용한 다음 복제를 중지합니다.

```
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
```

대상 볼륨을 다음 위치에 연결합니다 `oracdb2-hg`(복제된 LUN은 원본 이름을 유지할 수 있습니다):

```
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"
```

대기 제어 파일을 `oracdb2`에 복사한 다음 기본 시스템에서 백업 모드를 종료합니다.

```
ALTER DATABASE END BACKUP;
```

스냅샷에서 시드

지속적인 복제가 필요하지 않은 경우 이 경로를 사용하여 일회성 랩 시딩을 수행하십시오.

일회성 랩 시드를 생성하려면 소스 스냅샷을 생성하고 `oracle-pool-b`(클라우드 콘솔 또는 API)에서 해당 스냅샷으로부터 대기 볼륨을 생성합니다. 생성된 볼륨을 `oracdb2-hg`에 연결한 다음 대기 ASM 디스크 그룹 마운트을 계속 진행합니다.

대기 **ASM** 디스크 그룹 마운트

대기 호스트에서 연결된 스토리지 경로를 검색하고 데이터베이스 복구 전에 ASM 디스크 그룹을 마운트합니다.

켜짐 상태에서 `oracdb2`, 대기 풀 iSCSI 포털에 로그인하고 다중 경로 장치를 다시 검색합니다. 랩 워크플로에서 ASM 디스크 헤더가 기본 명명 규칙과 일치하는 경우 기본 스타일 별칭(예: `ora_oracdb1_data_01`, `ora_oracdb1_arch_01`)을 사용하고, `asm_diskstring='/dev/mapper/ora_oracdb1_*p*'`를 설정하고, 파티션 소유권이 `grid:asmadmin`인지 확인한 다음 디스크 그룹을 마운트합니다.

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

대기 인스턴스를 완료합니다.

대기 제어 파일을 복원하고, 캡처된 SCN으로 복구하고, 물리적 대기 상태로 전환한 다음 관리형 복구를 시작합니다.

```
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;
```

이 시점에서 대기 모드는 `PHYSICAL STANDBY` 및 `MOUNTED` 상태여야 하며 관리형 복구가 시작되어야 합니다.

등급별 다음 단계:

- 프로덕션 **HA(**데이터 가드 미사용**):** 바로 4단계: Oracle Restart에 대기 상태 등록으로 진행하세요.
- 프로덕션 **HA(Data Guard + FSFO):** 4단계: Oracle Restart에 대기 상태 등록로 계속 진행한 다음 Data Guard 최종화 단계로 진행합니다.

4단계: Oracle Restart에 대기 상태 등록

대기 데이터베이스를 Oracle Restart에 등록하여 재부팅 시 ASM 디스크 그룹이 자동으로 복구되고, 대기 데이터베이스가 마운트되고, 관리형 복구가 다시 시작되도록 합니다. 또한 애플리케이션 서비스를 두 데이터베이스 리소스 모두에 추가합니다.

1. 대기 데이터베이스에서 spfile 위치를 캡처하고 Oracle Restart에 등록합니다 `oracdb2`. 쿼리에서 `<STANDBY_SPFILE_PATH>`를 대체합니다 (대개 `+DATA` 아래에 있음).

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

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

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

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

2. `oracdb1`에서 모든 ASM 디스크 그룹 종속성을 포함하도록 기본 데이터베이스 리소스를 확인하고 업데이트하십시오.

```
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. 대기 데이터베이스 리소스에 애플리케이션 서비스를 추가합니다 (`orcls` (on `oracdb2`). `role PRIMARY`을 양쪽에서 모두 사용하므로 전환 후에도 `orclapp`을 사용할 수 있습니다.

```
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. 대기 데이터베이스 리소스를 다음 위치에서 확인하십시오. `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
'
```

다음 단계

등급별 특징:

- 프로덕션 **HA(Data Guard** 미사용**):** 스토리지 복제 기반 복구 대상을 유지하기 위해 대기 데이터베이스 초기화가 완료되었으며, 대기 데이터베이스가 Oracle Restart에 백업 인스턴스로 등록되었습니다.
- **Prod HA (Data Guard + FSFO):** 브로커 관리형 스위치오버 및 빠른 시작 페일오버를 활성화하려면 Data Guard용 대기 데이터베이스를 최종 설정합니다.을(를) 계속 진행하십시오.

Google Cloud NetApp Volumes에서 Data Guard용 대기 데이터베이스를 최종 구성합니다.

Google Cloud NetApp Volumes에서 Oracle Data Guard의 스탠바이 데이터베이스를 최종 구성하려면 스탠바이 리두 로그 파일을 생성하고, 플래시백 데이터베이스를 활성화하고, 리두 전송을 활성화하고, Data Guard 상태를 확인해야 합니다.

티어별 적용: 이 절차는 **Prod HA (Data Guard + FSFO)** 티어에만 필요합니다.

1단계: 대기 리두 로그 파일 생성

Fast-Start Failover를 지원하기 위해 두 데이터베이스 호스트 모두에 대기 리두 로그 파일을 생성합니다. 크기는 가장 큰 기본 온라인 리두 로그 파일보다 크거나 같아야 하며, 개수는 (스레드당 온라인 그룹 수) + 1이어야 합니다. GCNV 시딩 후, 복제된 경로를 수정하기 위해 대기 데이터베이스에서 대기 리두 로그 파일을 삭제하고 다시 생성합니다.

1. 기본 데이터베이스에 대기 리두 로그 파일을 생성합니다(`orcl`):

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

2. GCNV 시딩 후 대기 데이터베이스에서 대기 리두 로그 파일을 삭제하고 다시 생성합니다((`orcls`). `+DATA/ORCL/...` 아래의 복제된 경로는 재구축될 때까지 `ORA-19527/ `ORA-16086`을(를) 유발합니다.

```
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;
```

2단계: 플래시백 활성화 및 복구 시작

장애 조치 후 자동 복구를 지원하려면 대기 시스템에서 플래시백 데이터베이스를 활성화한 다음 실시간 적용을 사용하여 관리형 복구를 시작하십시오. MRP가 활성화된 동안에는 플래시백을 활성화할 수 없으므로 관리형 복구를 시작하기 전에 플래시백을 활성화해야 합니다.

1. 대기 데이터베이스를 종료하고, MOUNT 모드로 다시 시작한 다음, 플래시백 데이터베이스를 활성화하십시오 `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. 실시간 적용을 통해 관리형 복구를 시작합니다.

```
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` 실시간 적용이 가능합니다(SRL에 도착하는 즉시 redo가 적용됨).

3단계: 재실행 로그 전달 활성화

스탠바이 생성 중 `ORA-12154` 오류를 억제하기 위해 스탠바이 초기화 절차의 2단계에서 의도적으로 `DEFER`(으)로 설정했던 `LOG_ARCHIVE_DEST_STATE_2`을(를) 활성화하여 프라이머리에서 스탠바이로의 redo 전송을 활성화합니다.

1. Switch `LOG_ARCHIVE_DEST_STATE_2` to `ENABLE`로 전환하고 강제로 로그 스위치를 실행하여 리두 전송을 시작하십시오.

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

2. redo 전송이 올바르게 작동하는지 확인하십시오.

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

`dest_2`에 `ORA-12154`가 표시되면 기본 서버를 재시작하십시오. xref:{relative_path}gcnv-oracle-data-guard.adoc#enable-broker-on-both-databases[1단계: 두 데이터베이스 모두에서 브로커를 활성화합니다] 후 DGMGRL을 통해 전송을 관리하십시오.

4단계: Data Guard 상태 확인

기본 데이터베이스가 READ WRITE 모드인지, 그리고 대기 데이터베이스가 리두 로그를 적용하는 관리형 복구로 마운트되었는지 확인하십시오.

1. 기본 데이터베이스 역할 및 열기 모드를 확인하십시오 `oracdb1`:

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

2. 대기 데이터베이스 역할, 열림 모드 및 관리형 복구 상태를 다음 위치에서 확인하십시오. `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
```

대기 상태에서 예상되는 값: `PHYSICAL STANDBY | MOUNTED`; `MRP0` 포함 `APPLYING_LOG`.

3. 대기 상태가 `MOUNTED`를 보고하지만 apply가 실행되지 않는 경우, `oracdb2`에서 관리형 복구를 다시 시작하십시오.

```
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'
```

다음 단계

자동화된 역할 관리 및 장애 조치 보호를 활성화하려면 Oracle Data Guard Broker, Fast-Start Failover 및 Observer 구성을(를) 계속 진행하십시오.

Google Cloud NetApp Volumes에서 Oracle Database 26ai용 Data Guard Broker 및 Fast-Start Failover 구성

Google Cloud NetApp Volumes에서 Oracle Database 26ai에 대한 자동 역할 전환을 활성화하려면 전용 옵저버를 사용하여 Oracle Data Guard Broker 및 Fast-Start Failover를

구성하십시오.

Tier-specific: 이 절차는 **Prod HA (Data Guard + FSFO)** 티어에만 적용됩니다.

이 절차에서는 두 데이터베이스 모두에서 브로커를 활성화하고, Data Guard 구성을 생성하고, MaxAvailability 보호 모드로 FSFO를 활성화하고, 옵저버 호스트에 Oracle Instant Client를 설치하고, 지갑 기반 자격 증명을 사용하여 systemd 서비스로 옵저버를 시작하고, 스위치오버 및 페일오버를 테스트합니다. 이후에는 `ENABLE CONFIGURATION`, 임시 `LOG_ARCHIVE_DEST_*` SQL이 아닌 *DGMGRL*을 통해 전송 및 역할을 관리하십시오.

1단계: Data Guard Broker 활성화

두 데이터베이스 호스트 모두에서 Data Guard Broker를 활성화하고 기본 데이터베이스와 대기 데이터베이스를 통합 관리하에 연결하는 브로커 구성을 생성합니다.

1. 기본 및 대기 데이터베이스 호스트에 `dg_broker_start=TRUE`을(를) 설정하십시오.

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

2. 기본 서버에서 OS 인증을 사용하여 DGMGRL에 연결하고 브로커 구성을 생성합니다.

> Observer 호스트에서만 자동 로그인 지갑이 존재한 후 `dgmgrl /@orcl`을(를) 사용하십시오. `dgmgrl` 명령줄에 암호를 입력하지 마십시오.

```
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. 구성 유효성 검사 — `WARNING` 또는 NULL이 아닌 `ERROR` 값을 3단계: FSFO 속성을 구성하고 활성화합니다 이전에 수정하십시오:

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

2단계: FSFO에 대한 플래시백 확인

두 호스트 모두에서 플래시백 데이터베이스가 활성화되어 있는지 확인하십시오. 플래시백은 FSFO 자동 복구에 필요하며, 이를 통해 장애 조치 후 이전 기본 호스트가 자동으로 대기 호스트로 구성에 다시 참여할 수 있습니다.

1. 두 데이터베이스 호스트 모두에서 `flashback_on`이(가) `YES` 켜져 있는지 확인합니다.

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

2. 플래시백 보존이 이미 설정되어 있지 않은 경우 기본 장치에만 해당:

```
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'
```

3단계: FSFO 구성 및 활성화

SYNC 리두 전송을 설정하고, MaxAvailability 보호 모드를 구성하고, 각 데이터베이스에 FSFO 대상을 정의하고, Fast-Start Failover를 활성화합니다.

1. 두 데이터베이스 모두에서 리두 전송 모드를 `SYNC`로 설정하고 보호 모드를 `MaxAvailability`로 높이십시오:

```
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. 각 데이터베이스가 서로를 장애 조치 대상으로 지정하도록 FSFO 대상을 설정한 다음 임계값 및 자동 복구 동작을 구성합니다.

```
-- 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. 빠른 시작 페일오버를 활성화하고 구성을 확인합니다:

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

4단계: Observer에 Instant Client 설치

전용 Observer VM에 Oracle Instant Client를 설치하고(`oradg-obs`, 전용 `oracle` OS 사용자를 생성하고, Observer가 TCP/1521을 통해 두 데이터베이스 멤버 모두에 연결할 수 있도록 Oracle Net 환경을 구성합니다.

1. 옵저버 호스트에 Oracle Instant Client 패키지를 설치합니다. (`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. 지갑과 systemd 유닛을 소유할 전용 `oracle` OS 사용자를 생성합니다.

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

3. Oracle Net 환경을 구성하고 두 데이터베이스 호스트 모두에 대한 항목을 사용하여 `tnsnames.ora`을(를) 생성하십시오.

```
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
```

5단계: Observer를 systemd 서비스로 실행

데이터베이스 멤버 두 곳 모두에 대한 자격 증명을 사용하여 자동 로그인 지갑을 생성한 다음, 재부팅 후에도 유지되고 구성에 자동으로 재연결되도록 Observer를 systemd 서비스로 구성하고 시작합니다.

전용 Data Guard 관리자 계정(예: `SYSDG`)의 자격 증명은 `SYS` 대신 지갑에 저장하십시오. 자격 증명은 `dgmgrl` 명령줄에 표시되어서는 안 됩니다. 명령줄에서는 `ps` 및 `journalctl`에서 볼 수 있으므로, 항상 Observer에서 `/@<tns_alias>`을 사용하여 연결하십시오.

1. 암호화된 지갑을 생성하고 두 데이터베이스 멤버에 대한 자격 증명을 입력하십시오.

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

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

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

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

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

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

2. Observer systemd 서비스가 암호 입력 없이 시작될 수 있도록 자동 로그인 지갑(`cwallet.sso`)을 생성합니다. `cwallet.sso`가 `mkstore` 실행 후 누락된 경우 Instant Client 도구 패키지 또는 데이터베이스 홈에서 `orapki`를 사용하여 생성한 다음 저장된 자격 증명을 다시 추가하십시오.

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

3. systemd 유닛을 생성하고, 서비스를 활성화하고, Observer가 연결되었는지 확인합니다.

```
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
```

Observer는 기본에서 읽어야 합니다 `CONNECTED`(a `DISCONNECTED` Observer는 FSFO를 자동으로 중단합니다):

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

6단계: FSFO 테스트

Data Guard 구성을 `VALIDATE DATABASE`으로 검증한 다음 계획된 스위치오버를 수행하고, 테스트 기간 내에 계획되지 않은 VM 재설정 페일오버를 수행하여 FSFO가 엔드 투 엔드로 제대로 작동하는지 확인합니다.

1. 계획된 전환을 테스트하고 원래 토폴로지를 복원합니다.

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

2. 제어된 테스트 환경에서 VM 재설정을 사용하여 예기치 않은 장애 조치를 테스트합니다.

   VM 재설정(크래시 테스트 방식)을 사용하십시오. 일반적인 중지 작업으로는 FSFO가 발생하지 않을 수 있습니다. Tail `/var/log/dgmgrl-observer.log` on `oradg-obs`을 사용하여 장애 조치 진행 상황을 모니터링하고, 완료되면 토폴로지를 복원하십시오.

다음 단계

이 배포 환경에 Oracle Data Guard Broker, Fast-Start Failover 및 Observer 구성이 완료되었습니다.