Google Cloud NetApp Volumes를 사용하여 Oracle Database 26ai 고가용성 배포

NetApp database solutions

NetApp
June 12, 2026

목차

Google Cloud NetApp Volumes를 사용하여 Oracle Database 26ai 고가용성 배포

이 가이드에서는 컴퓨팅 인스턴스 및 스토리지를 프로비저닝하고, Oracle Grid Infrastructure 및 Oracle Database를 설치하고, 대기 데이터베이스를 초기화하고, Fast-Start Failover를 사용하여 Oracle Data Guard를 구성하는 방법을 보여줍니다.

시작하기 전에

시작하기 전에 다음 사항을 확인하십시오.

- Compute Engine, VPC 네트워킹, 방화벽 구성, IAM 및 NetApp Volumes에 대한 권한이 있는 Google Cloud 프로젝트

작업	필수 액세스
Compute Engine VM 생성	Compute Instance Admin(또는 이에 상응하는 권한)
방화벽/방화벽 정책	네트워크 관리자 또는 위임된 정책 관리자
GCNV 풀 및 볼륨 생성	NetApp Volumes Admin
PSA 구성	호스트 프로젝트의 네트워크 관리자
IAP를 통한 SSH	IAP로 보호되는 터널 사용자 + OS 로그인(사용하는 경우)

- NetApp Volumes API 활성화됨
- 대상 리전에 맞게 구성된 VPC 및 서브넷
- Google Cloud NetApp Volumes에 대해 구성된 Private Services Access(PSA)
- 필요한 모든 가상 머신용 Oracle Linux 10
- 데이터베이스 호스트와 Observer 호스트에 대해 DNS 및 호스트 이름 확인이 구성되어 있습니다.
- Oracle Database 26ai 및 Grid Infrastructure용 Oracle 설치 미디어 및 패치 파일 사용 가능
- Oracle Data Guard, Oracle Restart 및 iSCSI 스토리지 개념에 대한 이해
- 모든 가상 머신에 대해 시간 동기화가 구성되어 있습니다.

  다음 명령을 사용할 수 있습니다.

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

이 가이드에서 사용된 예시 구성

이 가이드에서는 다음과 같은 배포 가정을 사용합니다.

- Google Compute Engine 가상 머신 3대:
  - `oracdb1` 운영 데이터베이스의 경우
  - `oracdb2` 대기 데이터베이스의 경우
  - `oradg-obs` Fast-Start Failover Observer의 경우
- 데이터베이스 영역당 하나의 GCNV Flex Unified 스토리지 풀
- 데이터베이스 호스트당 5개의 GCNV iSCSI 볼륨
- 자동 페일오버를 위한 Oracle Data Guard Broker 및 Fast-Start Failover
- 데이터베이스 호스트별로 전용 스토리지가 제공되며, 운영 호스트와 대기 호스트는 iSCSI 볼륨을 공유하지 않습니다.

명령에 있는 모든 예시 값을 호스트 이름, IP 주소, 영역, 프로젝트 이름, 포털 IP, 암호 및 Oracle 미디어 파일 이름을 포함하여 사용자 환경의 값으로 바꾸십시오.

목표

이 절차에서는 다음 작업을 완료합니다.

- 기본 데이터베이스, 대기 데이터베이스 및 Observer용 Compute Engine 인스턴스를 프로비저닝합니다.
- Oracle용 GCNV iSCSI 스토리지 및 다중 경로 장치를 구성합니다
- 두 데이터베이스 호스트 모두에 Oracle Grid Infrastructure와 Oracle Database 26ai를 설치합니다
- 기본 Oracle 데이터베이스 생성
- GCNV 복제, 스냅샷 또는 클론을 사용하여 대기 데이터베이스를 초기화합니다
- Oracle Data Guard Broker, Fast-Start Failover 및 Observer 구성

배포 옵션

이 섹션에서는 Google Compute Engine(GCE)의 Oracle 데이터베이스와 Google Cloud NetApp Volumes(GCNV) iSCSI 블록 스토리지를 사용하는 실제 HA/DR 배포 패턴을 비교합니다. 또한 GCNV 복제가 대기 초기화를 가속화하는 방법을 강조합니다. 시작하기 전에 이 매트릭스를 사용하여 계층을 선택하십시오. 이 가이드는 Prod HA(Data Guard + FSFO) — 맨 아래 행을 구현합니다.

환경	요구 사항	권장 아키텍처	**HA**	**DR**	오토메이션	주요 이점
개발/테스트	최저 비용	단일 인스턴스	아니요	예	아니요	스냅샷 클론
Prod Basic(재시작)	충돌로 인한 다운타임 감소	+ Oracle Restart	아니요	예	로컬 전용	자동 재시작
운영 HA (DG 없음)	수동 DR 허용	+ Snapshot / RMAN	부분적	예	부분적	GCNV 클론 복구
Prod HA (DG + FSFO)	진정한 HA(DBA 없음)	Data Guard + FSFO	예	예	전체	진정한 HA + 빠른 페일오버

HA/DR/자동화**:** 예 = 티어 목표 충족, 아니요 = 범위에 포함되지 않음, 부분적 = 스토리지 수준 또는 수동 단계만 해당.

계층에 대해 읽을 섹션

원하는 HA 수준에 따라 아래 매트릭스의 섹션을 읽어보십시오. 예를 들어, Data Guard 및 FSFO를 사용하는 Prod HA를 원하는 경우 모든 섹션을 읽으십시오. Restart를 사용하는 Dev/Test 또는 Prod Basic을 원하는 경우 첫 번째 열만 읽으십시오.

개발/테스트 또는 운영 기본(재시작)	**Prod HA(Data Guard** 미사용)	운영 **HA(Data Guard + FSFO)**
시작하기 전에	시작하기 전에	시작하기 전에
이 가이드에서 사용된 예시 구성	이 가이드에서 사용된 예시 구성	이 가이드에서 사용된 예시 구성
목표	목표	목표
배포 옵션	배포 옵션	배포 옵션
개요	개요	개요
아키텍처	아키텍처	아키텍처
Compute Engine 프로비저닝	Compute Engine 프로비저닝	Compute Engine 프로비저닝
GCNV iSCSI 볼륨 프로비저닝	GCNV iSCSI 볼륨 프로비저닝	GCNV iSCSI 볼륨 프로비저닝
Oracle 소프트웨어 설치	Oracle 소프트웨어 설치	Oracle 소프트웨어 설치
기본 데이터베이스를 생성합니다	기본 데이터베이스를 생성합니다	기본 데이터베이스를 생성합니다
	대기 데이터베이스를 생성합니다(3단계: GCNV 대기 초기화만 해당)	대기 데이터베이스를 생성합니다
		Data Guard Broker, FSFO 및 Observer 구성

개요

이 아키텍처는 Google Cloud NetApp Volumes(GCNV) iSCSI 스토리지와 Oracle Data Guard를 사용하여 Google Cloud에 Oracle Database 26ai 고가용성을 배포합니다. GCNV는 고성능 블록 스토리지를 제공하고 스토리지 계층에서 대기 초기화를 지원합니다. Data Guard는 지속적인 동기화, 전환 및 Fast-Start 페일오버를 제공합니다.

계층	역할
GCNV	블록 스토리지, 대기 초기화, 스토리지 복제
Data Guard	연속 동기화, 리두 적용, 스위치오버, FSFO

GCNV 복제는 Oracle 데이터베이스 채널을 통하지 않고 스토리지 계층에서 데이터를 이동하므로 RMAN 액티브 복제에 비해 스탠바이 초기화 시간을 크게 단축합니다. 환경에서 지원하는 경우 프로덕션 스탠바이 시드에는 GCNV 복제를 사용하는 것이 좋습니다.

구성 요소	세부 정보
DB 호스트	`oracdb1` / `oracdb2` — 서로 다른 영역, 각 영역당 5개의 GCNV iSCSI 볼륨

구성 요소	세부 정보
스토리지	`/u01` + ASM `+DATA`, `+RECO`, `+FRA` on GCNV(EXTERNAL)
대기 초기화	GCNV 복제/스냅샷/클론 → Oracle 마무리 → Data Guard
HA	재시작, 물리적 대기(MOUNTED), 브로커, FSFO, Observer (`oradg-obs`)
앱	서비스 `orclapp`

영역 설정**:** 데이터베이스 영역당 하나의 GCNV Flex Unified 풀이 사용되며, 호스트별로 전용 볼륨이 할당됩니다. 부팅 디스크는 OS 전용입니다. 범위 외**:** TDE, RAC, NVMe/TCP, OEM, Active Data Guard(대기는 **MOUNTED** 상태 유지).

아키텍처

참조 다이어그램

이 아키텍처는 세 가지 독립적인 데이터 경로를 제공합니다. 스토리지 복제와 Data Guard 리두 전송은 별도의 경로입니다.

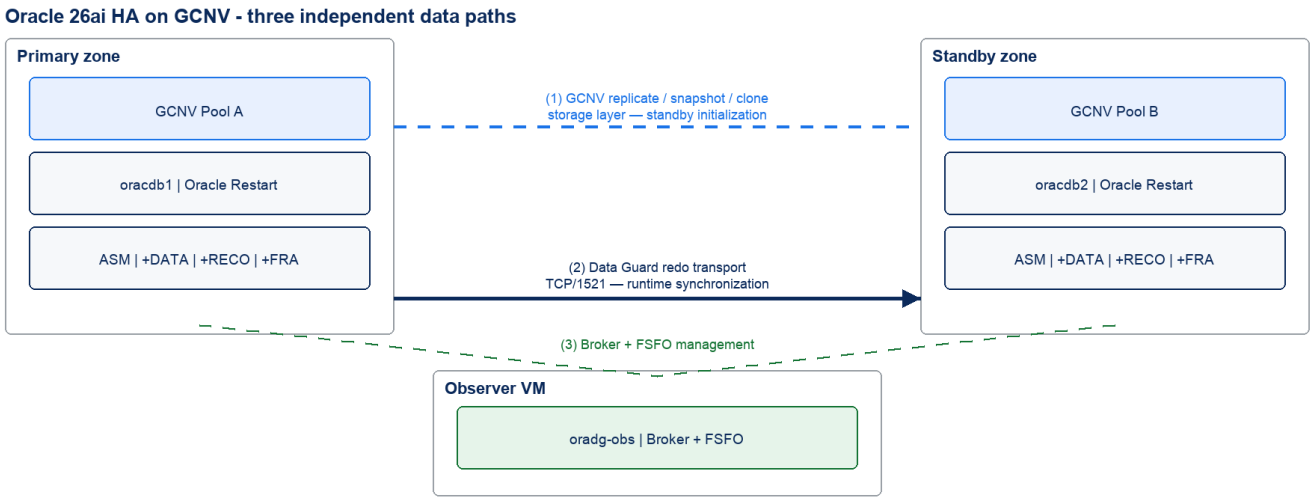

그림 **1. GCNV Oracle** 배포 **-** 참조 아키텍처

스토리지 복제 **≠** 리두 복제**.** 경로 1은 스탠바이 초기화를 위해 GCNV 계층에서 데이터 파일을 이동합니다. 경로 2는 전환 후 데이터베이스를 동기화 상태로 유지합니다. 경로 3은 Oracle Data Guard Broker 및 Observer를 통해 역할 전환 및 자동 페일오버를 오케스트레이션합니다.

플랫폼 역할

플랫폼	제공
GCNV	iSCSI 볼륨, 스냅샷, 볼륨 복제(기준 복제 + 증분 복제) — 대기 상태 초기화
Data Guard	리두 전송, MRP, 브로커, FSFO — 지속적인 동기화 및 페일오버

GCNV 복제는 기준 복사본을 실행한 다음 정책에 따라 증분 블록 업데이트를 실행합니다. Data Guard는 스탠바이 서버가 초기화된 후 리두 적용 및 FSFO를 담당합니다.

토폴로지 및 스토리지

역할	**VM**	영역	**GCNV** 풀	볼륨(호스트당)
주요한	`oracdb1`	`us-west1-a`	`oracle-pool-a`	`ora_<host>_u01`, `ora_<host>_data_01/02`, `ora_<host>_arch_01`, `ora_<host>_fra_01`
대기	`oracdb2`	`us-west1-b`	`oracle-pool-b`	동일한 명명 패턴
Observer	`oradg-obs`	`us-west1-c`	—	부팅 디스크만

스토리지	백업	사용
OS	GCE 부팅 디스크	OS 전용
`/u01`	GCNV iSCSI	그리드/DB 홈, 스테이징(XFS)
`+DATA` / `+RECO` / `+FRA`	GCNV iSCSI	ASM EXTERNAL — 데이터 파일, 아카이브, FRA

시스템 유형

이 가이드의 샘플 `n4-highmem-8`입니다. OLTP 워크로드는 일반적으로 `c4-standard-*`를 사용합니다.

Compute Engine 프로비저닝

1단계: VM 생성

동일 리전 내 서로 다른 영역에 세 개의 VM을 생성합니다(영역별 장애 격리). 총소유비용(TCO) 및 지속가능성을 고려하여 지연 시간 및 규정 준수 요구 사항을 충족하는 저탄소 리전을 선택하는 것이 좋습니다(예: `us-west1` vs `us-central1`). Cloud Console, `gcloud` Terraform 또는 표준 프로비저닝 워크플로를 사용하세요.

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

VM	영역	시스템 유형	부팅 디스크	네트워크	목적
`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)

네트워크 등급: 지연 시간이나 송신(>~200GiB/월)이 중요한 경우 Premium, 개발/테스트에서 TCO를 낮추려면 Standard를 선택하십시오.

세 디스크 모두에서 **Secure Boot**, **vTPM** 및 **Integrity Monitoring***을 활성화하십시오**.** 부팅 디스크에는 **OS**만 저장됩니다**. `/u01` Grid/DB** 홈**,** 스테이징 및 모든 **ASM** 데이터는 **GCNV iSCSI** 볼륨을 사용합니다**(**참조 **GCNV iSCSI** 볼륨 프로비저닝**) —** `**/u01**`에 대해 별도의 **GCE** 데이터 디스크를 연결하지 *마십시오.

2단계: VPC 방화벽 - 세 영역 모두에서 TCP/1521 허용 목록 추가

모든 영역의 세 VM `/32` 내부 IP 간에 TCP/1521을 허용합니다(`us-west1-a/b/c`). 동일한 허용 목록으로 VPC 방화벽 규칙 또는 방화벽 정책을 사용합니다. 규칙이 누락되면 redo 전송 및 Observer 연결이 중단됩니다.

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. 각 VM에서 유효성을 검사합니다.

```
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`

각 VM이 부팅된 후 호스트 이름을 설정하고 `/etc/hosts` 세 호스트 모두에 항목을 추가하여 Oracle Net, 브로커 및 Observer에 대한 정방향/역방향 이름 확인이 작동하도록 하십시오.

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

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

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

GCE 내부 IP 주소(**Compute Engine** → **VM instances** 목록, *Internal IP* 열에 표시됨)를 대체하십시오.

각 호스트에서 유효성을 검사합니다.

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

4단계: OS 기준선(DB 호스트만 해당)

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

`oracdb1` 및 `oracdb2`에서 실행(Observer 설정은 <<install-instant-client-on-observer,4단계: Observer 호스트에 Oracle Instant Client 설치>>에서 다룹니다).

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

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

# iSCSI, multipath, OUI JDK
sudo dnf install -y iscsi-initiator-utils device-mapper-multipath
sg3_utils \
  java-21-openjdk-headless libxcrypt-compat

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

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 DB 호스트의 IQN을 캡처하십시오. 이 IQN을 사용하여 2단계: 호스트 그룹 생성에서 GCNV 호스트 그룹을 생성합니다.

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

`oracdb2`에서 반복합니다. 각 호스트의 IQN을 기록합니다. 호스트당 하나의 호스트 그룹을 사용하여 단일 호스트의 재부팅 또는 IQN 재생성이 다른 호스트의 GCNV iSCSI 볼륨 가시성에 영향을 미치지 않도록 합니다.

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`).

GCNV iSCSI 볼륨 프로비저닝

1단계: 데이터베이스 영역별로 GCNV Flex Unified iSCSI 스토리지 풀을 하나씩 생성합니다

데이터베이스 영역당 하나의 Flex Unified 풀(아키텍처 참조).

이 HA 설계를 위해 두 개의 스토리지 풀을 생성합니다(각 영역에 대해 위 단계를 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.

`READY`을(를) 기다립니다. 데이터베이스 사용 공간에 맞게 풀 크기를 조정합니다(<<create-gcnv-iscsi-volumes,3단계: GCNV iSCSI 볼륨 생성>>의 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 그룹을 생성합니다(DB 호스트당 하나씩)

각 VM이 자체 볼륨만 볼 수 있도록 데이터베이스 호스트별로 호스트 그룹을 하나씩 생성합니다. 기본 VM과 대기 VM은 GCNV iSCSI 볼륨을 공유해서는 안 됩니다. Observer에는 GCNV 리소스가 없습니다. Cloud Console에서 다음을 수행합니다.

1. **NetApp** 볼륨 → 호스트 그룹 → 생성
2. 이름: `oracdb1-hg` · 지역: `us-west1` · 유형: iSCSI 이니시에이터 · 운영체제 유형: Linux
3. 호스트: `oracdb1`(값 `/etc/iscsi/initiatorname.iscsi`)에서 IQN을 붙여넣으세요.
4. 설명: "Oracle 운영 호스트 oracdb1" · 생성
5. `oracdb2-hg` `oracdb2`의 IQN을 사용하여 반복합니다.

3단계: GCNV iSCSI 볼륨 생성(데이터베이스 호스트당)

각 데이터베이스 호스트는 해당 영역의 풀에 5개의 GCNV iSCSI 볼륨을 할당받습니다. 이 중 하나는 `/u01`용이고 나머지 4개는 ASM 백업 장치용입니다.

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.

켜짐 `oracdb1`: `oracle-pool-a`의 호스트 그룹 `oracdb1-hg`에 5개의 볼륨을 모두 생성합니다.

켜짐 `oracdb2`: `oracle-pool-b`의 호스트 그룹 `oracdb2-hg`에 5개의 볼륨을 모두 생성합니다.

NetApp Volumes → **Volumes** → **Create** — iSCSI, 올바른 풀 및 호스트 그룹, Linux. 풀별 기록:

- 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: GCNV iSCSI 볼륨에 대한 Linux iSCSI 및 다중 경로 구성

해당 호스트의 풀 포털을 사용하여 `oracdb1`에서 실행한 다음 대기 풀 포털을 사용하여 `oracdb2`에서 실행합니다.

호스트 송신이 제한된 경우 각 데이터베이스 VM에서 해당 GCNV iSCSI 포털 IP로 TCP/3260을 허용하십시오(2단계: VPC 방화벽 - 세 영역 모두에서 TCP/1521 허용 목록 추가의 VM 간 TCP/1521 외에도).

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
```

1. 구성 `device-mapper-multipath`:

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

1. 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_*` 별칭과 동일한 패턴을 사용한 다음:

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

5단계: GCNV iSCSI 볼륨에서 ASM 백업 디바이스 파티션 생성

4개의 ASM 백업 장치(`u01`아님)를 각각 하나의 GPT 파티션으로 분할합니다. ASM은 원시 파티션을 사용합니다. 각 호스트에서 실행합니다. 이후의 모든 블록은 `HOST=$(hostname -s)`를 사용하여 로컬 호스트의 장치를 자동으로 선택합니다.

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

HOST=$(hostname -s)
sudo tee /etc/udev/rules.d/99-oracle-asm.rules >/dev/null <<'EOF'
KERNEL=="dm-*", ENV{DM_UUID}=="part?-mpath-*",
ENV{DM_NAME}=="ora_oracdb*_*p?", \
    OWNER="grid", GROUP="asmadmin", MODE="0660"
EOF

sudo udevadm control --reload-rules
for part in /dev/mapper/ora_${HOST}_*p1; do
  dm=$(readlink -f "$part" | xargs basename)
  sudo udevadm trigger --action=change --name-match="/dev/${dm}"
done
sudo udevadm settle
ls -lL /dev/mapper/ora_${HOST}_*p1    # grid:asmadmin 0660
```

6단계: 로컬 GCNV iSCSI 볼륨을 포맷하고 마운트합니다 /u01

`ora_<host>_u01`GCNV iSCSI 볼륨에는 Grid 홈, Oracle 홈 및 스테이징이 저장됩니다. 다중 경로 디바이스에서 XFS 포맷(ASM용으로 파티션되지 않음). `/etc/fstab`에서 UUID 사용(공유 파일 시스템 레이블 아님):

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

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

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

sudo mkdir -p /u01/app/oraInventory /u01/app/26ai/grid /u01/app/grid \
  /u01/app/oracle/product/26ai/db_1 /u01/stage
sudo chown -R grid:oinstall /u01/app/oraInventory /u01/app/26ai
/u01/app/grid
sudo chown -R oracle:oinstall /u01/app/oracle /u01/stage
sudo chmod -R 775 /u01/app /u01/stage
```

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

Oracle 소프트웨어 설치

기본 데이터베이스를 생성합니다 전에 각 DB 호스트에서 Grid를 실행한 다음 데이터베이스 홈 설치를 진행하십시오.

1단계: 각 DB 호스트에 Oracle Grid Infrastructure(Oracle Restart)를 설치합니다

`oracdb1`에서 이 섹션의 모든 작업을 실행한 다음 `oracdb2`에서 반복합니다. 두 호스트 모두 자체 `Grid` 홈, `ASM` 인스턴스 및 디스크 그룹을 갖게 됩니다. `Data Guard`는 스토리지가 아닌 `Oracle Net`을 통해 복제됩니다.

Oracle 바이너리를 스테이징합니다 `/u01`

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

Grid 홈을 제자리에 압축 해제합니다

26ai Grid GoldImage는 대상 Grid 홈 디렉터리에 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
```

그리드 **RU** 레벨. 이 가이드는 그리드 GoldImage에 이미 검증된 RU가 포함되어 있다고 가정합니다. 그리드 GoldImage가 대상 RU보다 이전 버전인 경우, Oracle에서 문서화한 `gridSetup.sh -applyRU` 절차에 따라 설치 중에 그리드 홈에 패치를 적용하거나, RU가 포함된 그리드 GoldImage를 사용하십시오. 그리드 홈과 데이터베이스 홈을 동일한 의도된 패치 레벨로 유지하십시오.

단일 샷 `gridSetup` — 전체 **HA_CONFIG** 응답 파일

각 호스트에 `/tmp/grid.rsp`빌드(`responseFileVersion`는 필수입니다(`HOST`를 대체하고 강력한 암호 사용):

```
HOST=$(hostname -s)

sudo -u grid bash -c "cat > /tmp/grid.rsp <<RSP
oracle.install.responseFileVersion=/oracle/install/rspfmt_crsinstall_respo
nse_schema_v23.0.0
INVENTORY_LOCATION=/u01/app/oraInventory
installOption=HA_CONFIG
ORACLE_BASE=/u01/app/grid
clusterUsage=GENERAL_PURPOSE
OSDBA=asmdba
OSOPER=asmoper
OSASM=asmadmin
storageOption=FLEX_ASM_STORAGE
sysasmPassword=WelcomeOracle1!
asmsnmpPassword=WelcomeOracle1!
diskGroupName=DATA
redundancy=EXTERNAL
auSize=4
diskString=/dev/mapper/ora_${HOST}_*p*
diskList=/dev/mapper/ora_${HOST}_data_01p1,/dev/mapper/ora_${HOST}_data_02
p1
managementOption=NONE
RSP"
sudo -u grid chmod 600 /tmp/grid.rsp
```

`gridSetup`를 실행하여 바이너리를 복사하고 구성을 준비합니다.

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

`Successfully Setup Software with warning(s).` 및 종료 코드 6(경고) 또는 0이 예상됩니다.

orainstRoot.sh 및 **`root.sh`**을(를) 루트로 실행합니다

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

root.sh crsctl/ srvctl/ asmcmd 래퍼를 생성하고 OHAS를 실행합니다.

gridSetup.sh -executeConfigTools **— ASM**을 실행하고 생성합니다 +DATA

의 응답 파일에 대해 구성 도우미(NETCA, ASMCA, CVU)를 실행합니다. 그러면 ASM 인스턴스와 +DATA 디스크 그룹이 생성됩니다.

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

NETCA/ASMCA/CVU 이후에 예상됩니다 Successfully Configured Software..

+RECO 및 +FRA 디스크 그룹을 생성합니다

일회성 설치는 `+DATA`만 생성합니다. `asmca`를 통해 나머지 두 개를 생성하십시오:

```
HOST=$(hostname -s)

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

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

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

ASM 및 **Oracle Restart** 확인

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

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

`oracdb2`에 대해서도 이 섹션을 반복하십시오. `HOST=$(hostname -s)`의 및 패턴은 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 Database 26ai를 설치합니다

`oracdb1`에서 이 섹션을 실행한 다음 `oracdb2`에서 실행하세요. 대기 데이터베이스가 <<create-standby-database,대기 데이터베이스를 생성합니다>>에 생성됩니다.

DB 홈과 **RU** 패치의 압축을 풉니다

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

RU 디렉터리 레이아웃 및 `-applyRU` 경로는 Oracle 설명서를 참조하십시오.

RU가 적용된 자동 소프트웨어 전용 설치

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

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

OL 8/9에서는 `-applyOneOffs`를 `runInstaller` 줄에서 생략하십시오.

`root`로 설치 후 스크립트를 실행하십시오.

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

Oracle 환경을 설정합니다

각 DB 호스트에서 (`orcl` on `oracdb1`, `orcls` on `oracdb2`):

```
sudo -u oracle tee -a /home/oracle/.bash_profile >/dev/null <<'EOF'
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export ORACLE_SID=orcl                          # use 'orcls' on oracdb2
export GRID_HOME=/u01/app/26ai/grid
export PATH=$ORACLE_HOME/bin:$GRID_HOME/bin:$PATH
export TNS_ADMIN=$ORACLE_HOME/network/admin
EOF
```

(대기 호스트에서 `ORACLE_SID=orcls`을(를) 사용하십시오. 대기 데이터베이스는 대기 데이터베이스를 생성합니다에 생성됩니다.)

기본 데이터베이스 생성((`oracdb1` 전용)

ASM 디스크 그룹에 대해 `dbca` 자동 모드로 실행합니다.

```
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
'
```

아카이브 로그를 다음 위치에 지정하십시오 `+RECO`(대기 서버는 다음 위치의 일치하는 설정을 사용합니다2단계: 대기: 최소 init.ora pfile, 암호 파일, 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
'
```

Oracle Restart에서 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
```

역할 기반 애플리케이션 서비스**(Oracle Restart)**를 생성합니다**.** 애플리케이션은 DB 이름이 아닌 `orclapp`를 통해 연결해야 하므로 페일오버가 투명하게 이루어집니다.

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

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

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

Broker 활성화 후 `orclapp`는 PRIMARY에서만 실행됩니다. ASM 디스크 그룹 전체에 제어 파일을 다중화하고 워크로드에 맞게 메모리 크기를 조정합니다(이 가이드에서는 4GB/3GB SGA를 예시로 사용합니다).

대기 데이터베이스를 생성합니다

`orcls` `oracdb2`(전용 볼륨 `oracle-pool-b`)에 구축합니다. 초기 운영 및 대기 설정을 완료한 다음 <<gcnv-standby-initialization,3단계: GCNV 대기 초기화>>, 대기 완료 작업 및 <<broker-fsfo-observer,Data Guard Broker, FSFO 및 Observer 구성>>을(를) 수행합니다. 대상: 운영 READ WRITE, 대기 PHYSICAL STANDBY, MOUNTED, MRP 적용.

1단계: 운영: SYS 암호, 암호 파일 및 DG 매개 변수

켜짐 oracdb1 상태 oracle:

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

설정: oracdb2

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

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

각 호스트에서 Oracle Restart를 통해 재시작하십시오.

```
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: 최소 init.ora pfile, 암호 파일, NOMOUNT

기본 암호 파일을 대기(IAP `gcloud compute scp`)로 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
```

`oracdb2`에서 소유권을 설정하고 대기 `pfile`을 생성합니다. *primary*에서 `*.compatible`를 먼저 복사합니다.

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

`oracdb2`에서 아래 블록의 `<COPY_FROM_PRIMARY>`에 해당 값을 대입하세요.

```
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
'
```

3단계: GCNV 대기 초기화까지 데이터 파일이 없는 NOMOUNT입니다.

3단계: GCNV 대기 초기화

`oracle-pool-b`*에서 ***Google Cloud NetApp Volumes** 복제(SnapMirror 기반, 기본 모드), 볼륨 스냅샷 또는 *클론*을 사용하여 대기 ASM 볼륨을 채운 다음 Oracle 최종 확정를 실행합니다. iSCSI 및 ASM은 GCNV iSCSI 볼륨 프로비저닝와(과) 동일합니다.

API	사용
`gcloud netapp storage-pools`	기본 모드 Flex Unified 풀 생성(`--mode=default`)
`gcloud netapp volumes`	iSCSI 볼륨, 호스트 그룹, 스냅샷
`gcloud netapp volumes replications`	위치 간 볼륨 복제

단계	작업
1	기본 아카이브 로그 + 강제 로깅(1단계: 운영: SYS 암호, 암호 파일 및 DG 매개 변수)
2	정지: `BEGIN BACKUP` SCN 기록, 대기 제어 파일
3	볼륨 복제(볼륨 복제를 생성합니다 및 컷오버 — 정지, 복제 중지, 대기 LUN 연결) 또는 스냅샷(대안 — 스냅샷 시드)
4	활성화 `oracdb2`: iSCSI, 다중 경로, ASM 마운트(4단계: GCNV iSCSI 볼륨에 대한 Linux iSCSI 및 다중 경로 구성, 5단계: GCNV iSCSI 볼륨에서 ASM 백업 디바이스 파티션 생성,)
5	Oracle 최종화 — SCN으로 복구, 마운트됨 (Oracle 최종 확정)
6	SRL 재구축(4단계: 대기 리두 로그 파일), MRP, 브로커(Data Guard Broker, FSFO 및 Observer 구성)

소규모 연구실에서는 RMAN 활성 복제 방식이 여전히 유효합니다. 프로덕션 대기 시드 구축에는 **GCNV** 복제 방식이 권장됩니다.

필수 조건

- `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.

```
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>
```

볼륨 복제를 생성합니다

```
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`가 *MIRRORED*/초기 동기화가 완료될 때까지 기다립니다.

컷오버 — 정지, 복제 중지, 대기 **LUN** 연결

기본:

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

최종 복제 주기를 허용한 다음:

```
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 `oracdb2-hg`에 연결합니다(복제된 LUN은 소스 이름을 유지할 수 있습니다. 업데이트 시 `name=oracdb1_data_lun` 사용):

```
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"
```

제어 파일을 `oracdb2`에 복사한 다음 운영 서버에서 다음을 수행합니다.

```
ALTER DATABASE END BACKUP;
```

대안 — 스냅샷 시드

일회성 시드: 소스 볼륨에서 스냅샷 → 대기 풀에서 스냅샷으로 볼륨 생성(Cloud Console 또는 API). attach to `oracdb2-hg` 후 Oracle 최종 확정로 진행합니다.

대기 **iSCSI** 및 **ASM(RMAN** 이전)

`oracdb2`에서 *대기 풀* iSCSI 포털에 로그인합니다. ASM 디스크 헤더가 기본 명명 규칙과 일치하면 *기본 스타일 다중 경로 별칭*(lab: `ora_oracdb1_data_01`, `ora_oracdb1_arch_01`)을 사용하고, 파티션에 `asm_diskstring='/dev/mapper/ora_oracdb1_*p*'`, `chown grid:asmadmin`를 설정한 다음:

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

Oracle 최종 확정

```
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;
```

변환 후 *스탠바이 리두 로그를 재구축*하십시오(복제된 제어 파일에 여전히 +DATA/ORCL/... SRL 경로가 남아 있어

ORA-19527 / ORA-16086 기본 시스템에서 오류가 발생합니다). 4단계: 대기 리두 로그 파일을 참조하십시오.

4단계: 대기 리두 로그 파일

FSFO를 사용하려면 두 호스트 모두에 필요합니다. 크기는 가장 큰 운영 온라인 redo 로그의 크기 이상이어야 하며, 개수는 (스레드당 온라인 그룹 수) + 1입니다.

GCNV 시드 후: 대기 서버의 모든 대기 로그 파일 그룹을 삭제하고 +DATA 전용 (db_create_file_dest='+DATA'`에서만 다시 생성합니다). `+DATA/ORCL/… 아래의 복제된 경로는 재구축될 때까지 ORA-19527 / `ORA-16086`를 발생시킵니다.

기본 **(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
```

GCNV 시드 이후 대기 모드**(orcls**:

```
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;
```

5단계: 대기 모드에서 플래시백을 활성화하고 관리형 복구를 시작합니다

관리형 복구를 시작하기 전에 플래시백을 활성화하십시오(MRP가 활성 상태인 동안에는 플래시백을 활성화할 수 없습니다).

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

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

6단계: 운영 시스템에서 **redo shipping**을 활성화합니다

> `LOG_ARCHIVE_DEST_2`는 2단계: 대기: 최소 `init.ora` pfile, 암호 파일, `NOMOUNT`에서 지속적인 `ORA-12154` 오류를 억제하기 위해 의도적으로 `DEFER`(으)로 설정되었습니다. 대기 생성 중입니다. 이제 대기가 준비되었으니 활성화하십시오.

`LOG_ARCHIVE_DEST_STATE_2`로 전환하고 `ENABLE` 로그 전환을 강제로 수행하여 첫 번째 재실행 라운드가 즉시 전송되도록 합니다.

```
sudo -u oracle bash -c '
. ~/.bash_profile
sqlplus / as sysdba <<SQL
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE SCOPE=BOTH;
ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM ARCHIVE LOG CURRENT;
SELECT dest_id, status, error FROM v$archive_dest_status WHERE dest_id IN
(1,2);
EXIT
SQL
'
# Expected: dest_id=2, STATUS=VALID, ERROR null.
```

`dest_2`에 `ORA-12154`가 표시되면 기본 서버를 재시작하십시오. <<enable-broker-on-both-databases,1단계: 두 데이터베이스 모두에서 브로커를 활성화합니다>> 후 `DGMGRL`을 통해 전송을 관리하십시오.

7단계: 대상 **Data Guard** 상태를 확인합니다

운영 (`oracdb1`):

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

대기 (`oracdb2` — Cloud Console **SSH** 또는 워크스테이션의 IAP):

```
gcloud compute ssh oracdb2 --tunnel-through-iap --zone=us-west1-b

sudo -u oracle bash <<'BASH'
. ~/.bash_profile
export ORACLE_SID=orcls

sqlplus -s / as sysdba <<'SQL'
SELECT database_role || ' | ' || open_mode
FROM v$database;

SELECT process, status, sequence#
FROM v$managed_standby
WHERE process IN ('MRP0','RFS');

EXIT
SQL
BASH
```

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

대기 보고서가 `MOUNTED`로 표시되지만 적용이 실행되지 않는 경우 `oracdb2`에서 MRP를 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'
```

8단계: Oracle Restart에 대기 데이터베이스 등록

GCNV 시딩 후 재시작을 사용하여 대기 상태를 등록하면 재부팅 시 ASM, MOUNT 및 적용이 복구됩니다.

`oracdb2`에서 spfile 위치를 캡처하고 Oracle Restart에 등록합니다(쿼리에서 `<STANDBY_SPFILE_PATH>`를 대체하며, 일반적으로 `+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
'
```

`oracdb1`에서, 기본 Oracle Restart 데이터베이스 리소스가 ASM 디스크 그룹 종속성을 나열하는지 확인합니다. DBCA가 등록만 한 경우 `RECO`를 추가하고 `DATA` 및 `FRA`를 추가합니다.

```
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
'
```

대기 데이터베이스 리소스에 동일한 애플리케이션 서비스 추가 (`orcls oracdb2`). 양쪽에서 모두 `role PRIMARY`을(를) 사용하면 전환 후에도 `orclapp`을(를) 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
'
```

`oracdb2`에서 대기 데이터베이스 리소스를 확인하십시오.

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

Data Guard Broker, FSFO 및 Observer 구성

`ENABLE CONFIGURATION` 이후에는 임시 `LOG_ARCHIVE_DEST_*` SQL이 아닌 *DGMGRL*을 통해 전송 및 역할을 관리하십시오.

1단계: 두 데이터베이스 모두에서 브로커를 활성화합니다

기본 및 대기 데이터베이스 호스트에서:

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

*기본 데이터베이스 호스트*에서 OS 인증을 사용하여 연결하십시오(5단계: Observer를 systemd 유닛으로 시작합니다의 Observer 지갑은 DB 호스트에서 필요하지 않습니다):

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

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

```
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.
```

`VALIDATE DATABASE`즉시 실행하세요. 이 명령은 전환을 시도하기 전에 고아 데이터 파일, 누락된 SRL, 리두 적용이 실행되지 않는 문제 등 일반적인 문제점을 찾아냅니다. `WARNING` 또는 NULL이 아닌 `ERROR` 값은 <<configure-fsfo-properties-and-enable,3단계: FSFO 속성을 구성하고 활성화합니다>> 전에 모두 수정해야 합니다.

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

2단계: 플래시백 확인(FSFO 자동 복구에 필요)

두 호스트 모두에서 `flashback_on` FSFO를 활성화하기 전에 확인하십시오.

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

운영 시스템에만 보존이 아직 설정되지 않은 경우:

```
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 속성을 구성하고 활성화합니다

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

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

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

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

4단계: Observer 호스트에 Oracle Instant Client 설치

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

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

# Oracle environment for the user
sudo mkdir -p /etc/oracle/network/admin
sudo chown -R oracle:oracle /etc/oracle
sudo -u oracle tee /home/oracle/.bash_profile >/dev/null <<'EOF'
export ORACLE_HOME=/usr/lib/oracle/26/client64
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
export PATH=$ORACLE_HOME/bin:$PATH
export TNS_ADMIN=/etc/oracle/network/admin
EOF

# tnsnames.ora — must reach both DB hosts on TCP/1521
sudo tee /etc/oracle/network/admin/tnsnames.ora >/dev/null <<'EOF'
orcl =
  (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = oracdb1)(PORT = 1521))
               (CONNECT_DATA = (SERVER = DEDICATED)(SERVICE_NAME =
orcl)))
orcls =
  (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = oracdb2)(PORT = 1521))
               (CONNECT_DATA = (SERVER = DEDICATED)(SERVICE_NAME =
orcls)))
EOF
sudo chown oracle:oracle /etc/oracle/network/admin/tnsnames.ora
```

5단계: Observer를 systemd 유닛으로 시작합니다

자격 증명은 자동 로그인 지갑에 저장되며 `dgmgrl` 명령줄(`ps`/ `journalctl`에 표시됨)에는 절대 노출되지 않습니다. Observer에서만 `/@<tns_alias>`을(를) 사용하여 연결하십시오.

- **전용 계정 사용:** 전용 Data Guard 관리 계정(예: `SYSDG`)의 자격 증명을 `SYS`이 아닌 지갑에 저장합니다.
- **자동 로그인 지갑 필요:** Observer systemd 서비스에는 자동 로그인 지갑 `cwallet.sso`이 필요합니다). `cwallet.sso`를 실행한 후 `mkstore`이(가) 없는 경우 Instant Client **tools** 패키지 또는 데이터베이스 홈의 `orapki`를 사용하여 자동 로그인 지갑을 생성한 다음 저장된 자격 증명을 다시 추가하십시오.

1. 두 멤버 모두의 자격 증명을 사용하여 Observer에서 지갑을 생성합니다.

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

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

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

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

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

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

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

1. Systemd 유닛 + 로그 로테이션:

```
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
```

기본 서버에서 확인하십시오. Observer는 다음 내용을 읽어야 합니다 `CONNECTED`(a `DISCONNECTED` Observer는 FSFO를 자동으로 일시 중단합니다):

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

6단계: FSFO(스위치오버 및 페일오버) 테스트

계획된 전환:

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

계획되지 않은 페일오버: 테스트 기간 내에 VM 재설정(크래시 테스트 방식)을 사용하십시오. 일반적인 중지 작업으로는 FSFO가 발생하지 않을 수 있습니다. `/var/log/dgmgrl-observer.log`에서 Tail `oradg-obs`을 수행하고 완료되면 토폴로지를 복원하십시오.