Table of Contents

On-Premises/Hybrid Cloud ...................................................... 1
TR-4992: Simplified, Automated Oracle Deployment on NetApp C-Series with NFS ......................... 1
TR-4983: Simplified, Automated Oracle Deployment on NetApp ASA with iSCSI ............................... 29
NVA-1155: Oracle 19c RAC databases on FlexPod Datacenter with Cisco UCS and NetApp AFF A800 over FC - Design and deployment guide ...................................................... 46
TR-4250: SAP with Oracle on UNIX and NFS with NetApp Clustered Data ONTAP and SnapManager for SAP 3.4 .................................................................................. 46
Deploying Oracle Database ....................................................... 46
Solution Overview ................................................................. 67
TR-4794: Oracle databases on NetApp EF-Series ................................................................. 90
On-Premises/Hybrid Cloud

TR-4992: Simplified, Automated Oracle Deployment on NetApp C-Series with NFS

Allen Cao, Niyaz Mohamed, NetApp

This solution provides an overview and details for automated Oracle deployment in NetApp AFF C-Series as primary database storage with NFS protocol. The Oracle database deploys as a container database with dNFS enabled.

Purpose

NetApp AFF C-Series is a capacity flash storage that makes all-flash more accessible and affordable for unified storage. It is sufficient performance-wise for many tier 1 or tier 2 Oracle database workloads. Powered by NetApp ONTAP® data management software, AFF C-Series systems deliver industry-leading efficiency, superior flexibility, best-in-class data services, and cloud integration to help you scale your IT infrastructure, simplify your data management, and reduce storage cost and power consumption.

This documentation demonstrates the simplified deployment of Oracle databases in NetApp C-Series via NFS mounts using Ansible automation. The Oracle database deploys in a container database (CDB) and pluggable databases (PDB) configuration with Oracle dNFS protocol enabled to boost performance. Furthermore, the solution provides the best practices in setting up storage networking and storage virtual machine (SVM) with NFS protocol on C-Series storage controllers. The solution also includes information on fast Oracle database backup, restore, and clone with the NetApp SnapCenter UI tool.

This solution addresses the following use cases:

• Automated Oracle container database deployment on NetApp C-Series storage controllers.
• Oracle database protection and clone on C-Series with SnapCenter UI tool.

Audience

This solution is intended for the following people:

• A DBA who would like to deploy Oracle on NetApp C-Series.
• A database solution architect who would like to test Oracle workloads on NetApp C-Series.
• A storage administrator who would like to deploy and manage an Oracle database on NetApp C-Series.
• An application owner who would like to stand up an Oracle database on NetApp C-Series.

Solution test and validation environment

The testing and validation of this solution were performed in a lab setting that might not match the final deployment environment. See the section Key factors for deployment consideration for more information.

Architecture
### Hardware and software components

#### Hardware

<table>
<thead>
<tr>
<th>Component</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetApp C-Series C400</td>
<td>ONTAP Version 9.13.1P3</td>
</tr>
<tr>
<td></td>
<td>Two disk shelves / 24 disks with 278 TiB capacity</td>
</tr>
<tr>
<td>VM for DB server</td>
<td>4 vCPUs, 16GiB RAM</td>
</tr>
<tr>
<td></td>
<td>Two Linux VM instances for concurrent deployment</td>
</tr>
<tr>
<td>VM for SnapCenter</td>
<td>4 vCPUs, 16GiB RAM</td>
</tr>
<tr>
<td></td>
<td>One Windows VM instance</td>
</tr>
</tbody>
</table>

#### Software

<table>
<thead>
<tr>
<th>Component</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedHat Linux</td>
<td>RHEL Linux 8.6 (LVM) - x64 Gen2</td>
</tr>
<tr>
<td></td>
<td>Deployed RedHat subscription for testing</td>
</tr>
<tr>
<td>Windows Server</td>
<td>2022 DataCenter x64 Gen2</td>
</tr>
<tr>
<td></td>
<td>Hosting SnapCenter server</td>
</tr>
<tr>
<td>Oracle Database</td>
<td>Version 19.18</td>
</tr>
<tr>
<td></td>
<td>Applied RU patch</td>
</tr>
<tr>
<td></td>
<td>p34765931_190000_Linux-x86-64.zip</td>
</tr>
<tr>
<td>Oracle OPatch</td>
<td>Version 12.2.0.1.36</td>
</tr>
<tr>
<td></td>
<td>Latest patch</td>
</tr>
<tr>
<td></td>
<td>p6880880_190000_Linux-x86-64.zip</td>
</tr>
<tr>
<td>SnapCenter Server</td>
<td>Version 5.0</td>
</tr>
<tr>
<td></td>
<td>Workgroup deployment</td>
</tr>
<tr>
<td>Open JDK</td>
<td>Version java-11-openjdk</td>
</tr>
<tr>
<td></td>
<td>SnapCenter plugin requirement on DB VMs</td>
</tr>
<tr>
<td>NFS</td>
<td>Version 3.0</td>
</tr>
<tr>
<td></td>
<td>Oracle dNFS enabled</td>
</tr>
<tr>
<td>Ansible</td>
<td>core 2.16.2</td>
</tr>
<tr>
<td></td>
<td>Python 3.6.8</td>
</tr>
</tbody>
</table>
Oracle database configuration in the lab environment

<table>
<thead>
<tr>
<th>Server</th>
<th>Database</th>
<th>DB Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ora_01</td>
<td>NTAP1(NTAP1_PDB1,NTAP1_PD</td>
<td>/u01, /u02, /u03 NFS mounts on C400 volumes</td>
</tr>
<tr>
<td></td>
<td>B2,NTAP1_PDB3)</td>
<td></td>
</tr>
<tr>
<td>ora_02</td>
<td>NTAP2(NTAP2_PDB1,NTAP2_PD</td>
<td>/u01, /u02, /u03 NFS mounts on C400 volumes</td>
</tr>
<tr>
<td></td>
<td>B2,NTAP2_PDB3)</td>
<td></td>
</tr>
</tbody>
</table>

Key factors for deployment consideration

- **Oracle database storage layout.** In this automated Oracle deployment, we provision three database volumes for each database to host Oracle binary, data, and logs by default. The volumes are mounted on Oracle DB server as /u01 - binary, /u02 - data, /u03 - logs via NFS. Dual control files are configured on /u02 and /u03 mount points for redundancy.

- **Multiple DB servers deployment.** The automation solution can deploy an Oracle container database to multiple DB servers in a single Ansible playbook run. Regardless of the number of DB servers, the playbook execution remains the same. You can deploy multiple container databases to a single VM instance by repeating the deployment with different database instance IDs (Oracle SID). But ensure there is sufficient memory on the host to support deployed databases.

- **dNFS configuration.** By using dNFS (available since Oracle 11g), an Oracle database running on a DB VM can drive significantly more I/O than the native NFS client. Automated Oracle deployment configures dNFS on NFSv3 by default.

- **Load balancing on C400 controller pair.** Place Oracle database volumes on C400 controller nodes evenly to balance the workload. DB1 on controller 1, DB2 on controller 2, and so on. Mount the DB volumes to its local lif address.

- **Database backup.** NetApp provides a SnapCenter software suite for database backup, restore, and cloning with a user-friendly UI interface. NetApp recommends implementing such a management tool to achieve fast (under a minute) snapshot backup, quick (minutes) database restore, and database clone.

Solution deployment

The following sections provide step-by-step procedures for automated Oracle 19c deployment and information for Oracle database protection and clone after deployment.

Prerequisites for deployment
Deployment requires the following prerequisites.

1. A NetApp C-Series storage controller pair is racked, stacked, and latest version of ONTAP operating system is installed and configured. Refer to this setup guide as necessary: Detailed guide - AFF C400

2. Provision two Linux VMs as Oracle DB servers. See the architecture diagram in the previous section for details about the environment setup.

3. Provision a Windows server to run the NetApp SnapCenter UI tool with the latest version. Refer to the following link for details: Install the SnapCenter Server

4. Provision a Linux VM as the Ansible controller node with the latest version of Ansible and Git installed. Refer to the following link for details: Getting Started with NetApp solution automation in section -
   Setup the Ansible Control Node for CLI deployments on RHEL / CentOS or
   Setup the Ansible Control Node for CLI deployments on Ubuntu / Debian.

   Enable ssh public/private key authentication between Ansible controller and database VMs.

5. From Ansible controller admin user home directory, clone a copy of the NetApp Oracle deployment automation toolkit for NFS.

   ```bash
   git clone https://bitbucket.ngage.netapp.com/scm/ns-bb/na_oracle_deploy_nfs.git
   ```

6. Stage following Oracle 19c installation files on DB VM /tmp/archive directory with 777 permission.

   ```bash
   installer_archives:
   - "LINUX.X64_193000_db_home.zip"
   - "p34765931_190000_Linux-x86-64.zip"
   - "p6880880_190000_Linux-x86-64.zip"
   ```

Configure Networking and SVM on C-Series for Oracle
This section of deployment guide demonstrates best practices to set up networking and storage virtual machine (SVM) on C-Series controller for Oracle workload with NFS protocol using ONTAP System Manager UI.

1. Login to ONTAP System Manager to review that after initial ONTAP cluster installation, broadcast domains have been configured with ethernet ports properly assigned to each domain. Generally, there should be a broadcast domain for cluster, a broadcast domain for management, and a broadcast domain for workload such as data.

2. From NETWORK - Ethernet Ports, click Link Aggregate Group to create a LACP link aggregate group port a0a, which provides load balance and failover among the member ports in the aggregate group port. There are 4 data ports - e0e, e0f, e0g, e0h available on C400 controllers.

3. Select the ethernet ports in the group, LACP for mode, and Port for load distribution.
4. Validate LACP port a0a created and broadcast domain Data is now operating on LACP port.
5. From Ethernet Ports, click VLAN to add a VLAN on each controller node for Oracle workload on NFS protocol.
6. Login to C-Series controllers from cluster management IP via ssh to validate that network failover groups are configured correctly. ONTAP create and manage failover groups automatically.
7. From STORAGE - Storage VMs, click +Add to create a SVM for Oracle.

8. Name your Oracle SVM, check Enable NFS and Allow NFS client access.
9. Add NFS export policy **Default rules.**
10. In **NETWORK INTERFACE**, fill in IP address on each node for NFS lif addresses.
11. Validate SVM for Oracle is up/running and NFS lifs status is active.
12. From STORAGE-Volumes tab to add NFS volumes for Oracle database.

13. Name your volume, assign capacity, and performance level.
14. In Access Permission, choose the default policy created from previous step. Uncheck Enable Snapshot Copies as we prefer to use SnapCenter to create application consistent snapshots.
15. Create three DB volumes for each DB server: server_name_u01 - binary, server_name_u02 - data, server_name_u03 - logs.

The DB volume naming convention should strictly follow format as stated above to ensure automation to work correctly.

This completes the C-series controller configuration for Oracle.
Automation parameter files

Ansible playbook executes database installation and configuration tasks with predefined parameters. For this Oracle automation solution, there are three user-defined parameter files that need user input before playbook execution.

- hosts - define targets that the automation playbook is running against.
- vars/vars.yml - the global variable file that defines variables that apply to all targets.
- host_vars/host_name.yml - the local variable file that defines variables that apply only to a named target. In our use case, these are the Oracle DB servers.

In addition to these user-defined variable files, there are several default variable files that contain default parameters that do not require change unless necessary. The following sections show how to configure the user-defined variable files.

Parameter files configuration
1. Ansible target hosts file configuration:

```yaml
# Enter Oracle servers names to be deployed one by one, follow by each Oracle server public IP address, and ssh private key of admin user for the server.
[oracle]
ora_01 ansible_host=10.61.180.21 ansible_ssh_private_key_file =ora_01.pem
ora_02 ansible_host=10.61.180.23 ansible_ssh_private_key_file =ora_02.pem
```

2. Global vars/vars.yml file configuration
### Oracle 19c deployment user configuration variables

##### Consolidate all variables from ONTAP, linux and oracle

---

# Prerequisite to create three volumes in NetApp ONTAP storage from System Manager or cloud dashboard with following naming convention:
# db_hostname_u01 - Oracle binary
# db_hostname_u02 - Oracle data
# db_hostname_u03 - Oracle redo
# It is important to strictly follow the name convention or the automation will fail.

---

### ONTAP env specific config variables ###

---

### Linux env specific config variables ###

---

### DB env specific install and config variables ###

---

# Database domain name

db_domain: solutions.netapp.com

# Set initial password for all required Oracle passwords. Change them after installation.

initial_pwd_all: Xxxxxxxxx

---

3. Local DB server host_vars/host_name.yml configuration such as ora_01.yml, ora_02.yml ...
# User configurable Oracle host specific parameters

# Enter container database SID. By default, a container DB is created with 3 PDBs within the CDB
oracle_sid: NTAP1

# Enter database shared memory size or SGA. CDB is created with SGA at 75% of memory_limit, MB. The grand total of SGA should not exceed 75% available RAM on node.
memory_limit: 8192

# Local NFS lif ip address to access database volumes
nfs_lif: 172.30.136.68

Playbook execution
There are a total of five playbooks in the automation toolkit. Each performs different task blocks and serves different purposes.

- **0-all_playbook.yml** - execute playbooks from 1-4 in one playbook run.
- **1-ansible_requirements.yml** - set up Ansible controller with required libs and collections.
- **2-linux_config.yml** - execute Linux kernel configuration on Oracle DB servers.
- **4-oracle_config.yml** - install and configure Oracle on DB servers and create a container database.
- **5-destroy.yml** - optional to undo the environment to dismantle all.

There are three options to run the playbooks with the following commands.

1. Execute all deployment playbooks in one combined run.

   ```bash
   ansible-playbook -i hosts 0-all_playbook.yml -u admin -e @vars/vars.yml
   ```

2. Execute playbooks one at a time with the number sequence from 1-4.

   ```bash
   ansible-playbook -i hosts 1-ansible_requirements.yml -u admin -e @vars/vars.yml
   ```
   ```bash
   ansible-playbook -i hosts 2-linux_config.yml -u admin -e @vars/vars.yml
   ```
   ```bash
   ansible-playbook -i hosts 4-oracle_config.yml -u admin -e @vars/vars.yml
   ```

3. Execute 0-all_playbook.yml with a tag.

   ```bash
   ansible-playbook -i hosts 0-all_playbook.yml -u admin -e @vars/vars.yml -t ansible_requirements
   ```
   ```bash
   ansible-playbook -i hosts 0-all_playbook.yml -u admin -e @vars/vars.yml -t linux_config
   ```
ansible-playbook -i hosts 0-all_playbook.yml -u admin -e @vars/vars.yml -t oracle_config

4. Undo the environment
ansible-playbook -i hosts 5-destroy.yml -u admin -e @vars/vars.yml

Post execution validation
After the playbook run, login to the Oracle DB server VM to validate that Oracle is installed and configured and a container database is created successfully. Following is an example of Oracle database validation on DB VM ora_01 or ora_02.

1. Validate NFS mounts

[admin@ora_01 ~]$ cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Wed Oct 18 19:43:31 2023
#
# Accessible filesystems, by reference, are maintained under
# '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for
# more info.
#
# After editing this file, run 'systemctl daemon-reload' to update
# systemd
# units generated from this file.
#
/dev/mapper/rhel-root / xfs defaults 0 0
UUID=aff942c4-b224-4b62-807d-6a5c22f7b623 /boot
xfs defaults 0 0
/dev/mapper/rhel-swap none swap defaults 0 0
/root/swapfile swap swap defaults 0 0
172.21.21.100:/ora_01_u01 /u01 nfs
rw,bg,hard,vers=3,proto=tcp,timeo=600,rsize=65536,wsize=65536 0 0
172.21.21.100:/ora_01_u02 /u02 nfs
rw,bg,hard,vers=3,proto=tcp,timeo=600,rsize=65536,wsize=65536 0 0
172.21.21.100:/ora_01_u03 /u03 nfs
rw,bg,hard,vers=3,proto=tcp,timeo=600,rsize=65536,wsize=65536 0 0

[admin@ora_01 tmp]$ df -h
Filesystem Size Used Avail Use% Mounted on
devtmpfs 7.7G 0 7.7G 0% /dev
tmpfs 7.8G 0 7.8G 0% /dev/shm
tmpfs 7.8G 18M 7.8G 1% /run
tmpfs 7.8G 0 7.8G 0% /sys/fs/cgroup
/dev/mapper/rhel-root 44G 28G 17G 62% /
/dev/sda1 1014M 258M 757M 26% /boot
tmpfs 1.6G 12K 1.6G 1% /run/user/42
tmpfs 1.6G 40K 1.6G 1% /run/user/1000
172.21.21.100:/ora_01_u01  50G  8.7G  42G  18% /u01
172.21.21.100:/ora_01_u02  200G 384K 200G  1% /u02
172.21.21.100:/ora_01_u03  100G 320K 100G  1% /u03

[admin@ora_02 ~]$ df -h
Filesystem                Size  Used Avail Use% Mounted on
devtmpfs                   7.7G     0  7.7G   0% /dev
tmpfs                      7.8G     0  7.8G   0% /dev/shm
tmpfs                      7.8G  18M  7.8G   1% /run
tmpfs                      7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/mapper/rhel-root      44G   28G   17G  63% /
/dev/sda1                  1014M  258M  757M  26% /boot
tmpfs                      1.6G  12K  1.6G   1% /run/user/42
tmpfs                      1.6G  4.0K  1.6G   1% /run/user/1000
172.21.21.101:/ora_02_u01  50G  7.8G  43G  16% /u01
172.21.21.101:/ora_02_u02  200G 320K 200G  1% /u02
172.21.21.101:/ora_02_u03  100G 320K 100G  1% /u03

2. Validate Oracle listener

[admin@ora_02 ~]$ sudo su
[root@ora_02 admin]# su - oracle
[oracle@ora_02 ~]$ lsnrctl status listener.ntap2

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 29-MAY-2024 12:13:30
Copyright (c) 1991, 2022, Oracle.  All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=ora_02.cie.netapp.com)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER.NTAP2
Version                   TNSLSNR for Linux: Version 19.0.0.0.0 -
Production
Start Date                23-MAY-2024 16:13:03
Uptime                    5 days 20 hr. 0 min. 26 sec
Trace Level               off
Security                   ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/app/oracle/product/19.0.0/NTAP2/network/admin/listener.ora
Listener Log File
Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=ora_02.cie.netapp.com)(PORT=1521)))

(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=ora_02.cie.netapp.com)(PORT=5500))(Security=(my_wallet_directory=/u01/app/oracle/product/19.0.0/NTAP2/admin/NTAP2/xdb_wallet))(Presentation=HTTP)(Session=RAW))

Services Summary...
Service "192551f1d7e65fc3e06308b43d0a63ae.solutions.netapp.com" has 1 instance(s).
  Instance "NTAP2", status READY, has 1 handler(s) for this service...
Service "1925529a43396002e06308b43d0a2d5a.solutions.netapp.com" has 1 instance(s).
  Instance "NTAP2", status READY, has 1 handler(s) for this service...
Service "1925530776b76049e06308b43d0a49c3.solutions.netapp.com" has 1 instance(s).
  Instance "NTAP2", status READY, has 1 handler(s) for this service...
Service "NTAP2.solutions.netapp.com" has 1 instance(s).
  Instance "NTAP2", status READY, has 1 handler(s) for this service...
Service "NTAP2XDB.solutions.netapp.com" has 1 instance(s).
  Instance "NTAP2", status READY, has 1 handler(s) for this service...
Service "ntap2_pdb1.solutions.netapp.com" has 1 instance(s).
  Instance "NTAP2", status READY, has 1 handler(s) for this service...
Service "ntap2_pdb2.solutions.netapp.com" has 1 instance(s).
  Instance "NTAP2", status READY, has 1 handler(s) for this service...
Service "ntap2_pdb3.solutions.netapp.com" has 1 instance(s).
  Instance "NTAP2", status READY, has 1 handler(s) for this service...

The command completed successfully
[oracle@ora_02 ~]$  

3. Validate Oracle database and dNFS

[oracle@ora-01 ~]$ cat /etc/oratab
#
# This file is used by ORACLE utilities. It is created by root.sh
# and updated by either Database Configuration Assistant while creating
# a database or ASM Configuration Assistant while creating ASM instance.

# A colon, ':', is used as the field terminator. A new line terminates
# the entry. Lines beginning with a pound sign, '#', are comments.
#
# Entries are of the form:
# $ORACLE_SID:$ORACLE_HOME:<N|Y>:
#
# The first and second fields are the system identifier and home directory of the database respectively. The third field indicates to the dbstart utility that the database should, "Y", or should not, "N", be brought up at system boot time.
#
# Multiple entries with the same $ORACLE_SID are not allowed.
#
# NTAP1:/u01/app/oracle/product/19.0.0/NTAP1:Y

[oracle@ora-01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Thu Feb 1 16:37:51 2024
Version 19.18.0.0.0

Copyright (c) 1982, 2022, Oracle. All rights reserved.

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.18.0.0.0

SQL> select name, open_mode, log_mode from v$database;

NAME      OPEN_MODE            LOG_MODE
--------- -------------------- ------------
NTAP1     READ WRITE           ARCHIVELOG

SQL> show pdbs

CON_ID CON_NAME OPEN_MODE RESTRICTED
<table>
<thead>
<tr>
<th>PDB$SEED</th>
<th>READ ONLY</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTAP1_PDB1</td>
<td>READ WRITE</td>
<td>NO</td>
</tr>
<tr>
<td>NTAP1_PDB2</td>
<td>READ WRITE</td>
<td>NO</td>
</tr>
<tr>
<td>NTAP1_PDB3</td>
<td>READ WRITE</td>
<td>NO</td>
</tr>
</tbody>
</table>

SQL> select name from v$datafile;

NAME
---------------------------------------------
/u02/oradata/NTAP1/system01.dbf
/u02/oradata/NTAP1/sysaux01.dbf
/u02/oradata/NTAP1/undotbs01.dbf
/u02/oradata/NTAP1/pdbseed/system01.dbf
/u02/oradata/NTAP1/pdbseed/sysaux01.dbf
/u02/oradata/NTAP1/users01.dbf
/u02/oradata/NTAP1/pdbseed/undotbs01.dbf
/u02/oradata/NTAP1/NTAP1_pdb1/system01.dbf
/u02/oradata/NTAP1/NTAP1_pdb1/sysaux01.dbf
/u02/oradata/NTAP1/NTAP1_pdb1/undotbs01.dbf
/u02/oradata/NTAP1/NTAP1_pdb1/users01.dbf
19 rows selected.

SQL> select name from v$controlfile;

NAME
---------------------------------------------
/u02/oradata/NTAP1/control01.ctl
/u03/orareco/NTAP1/control02.ctl

SQL> select member from v$logfile;

26
/u03/orareco/NTAP1/onlinelog/redo03.log
/u03/orareco/NTAP1/onlinelog/redo02.log
/u03/orareco/NTAP1/onlinelog/redo01.log

SQL> select svrname, dirname from v$dnfs_servers;

SVRNAME
--------------
--------------
--------------
--------------
172.21.21.100
/ora_01_u02
172.21.21.100
/ora_01_u03
172.21.21.100
/ora_01_u01

4. Login to Oracle Enterprise Manager Express to validate database.

---

4. Login to Oracle Enterprise Manager Express to validate database.
Oracle backup, restore, and clone with SnapCenter

NetApp recommends SnapCenter UI tool to manage Oracle database deployed in C-Series. Refer to TR-4979 Simplified, Self-managed Oracle in VMware Cloud on AWS with guest-mounted FSx ONTAP section Oracle backup, restore, and clone with SnapCenter for details on setting up SnapCenter and executing the database backup, restore, and clone workflows.
Where to find additional information

To learn more about the information described in this document, review the following documents and/or websites:

- NetApp AFF C-Series

- NetApp Enterprise Database Solutions

- Deploying Oracle Direct NFS

TR-4983: Simplified, Automated Oracle Deployment on NetApp ASA with iSCSI

Allen Cao, Niyaz Mohamed, NetApp

This solution provides overview and details for automated Oracle deployment and protection in NetApp ASA array as primary database storage with iSCSI protocol and Oracle database configured in standalone ReStart using asm as volume manager.

Purpose

NetApp ASA systems deliver modern solutions to your SAN infrastructure. They simplify at scale and enable you to accelerate your business-critical applications such as databases, make sure that your data is always available (99.9999% uptime), and reduce TCO and carbon footprint. The NetApp ASA systems include A-Series models designed for the most performance-demanding applications and C-Series models optimized for cost-effective, large-capacity deployments. Together, the ASA A-Series and C-Series systems deliver exceptional performance to improve customer experience and reduce time to results, keep business-critical data available, protected, and secure, and provide more effective capacity for any workload, backed by the industry’s most effective guarantee.

This documentation demonstrates the simplified deployment of Oracle databases in a SAN environment built with ASA systems using Ansible automation. The Oracle database is deployed in a standalone ReStart configuration with iSCSI protocol for data access and Oracle ASM for database disks management on the ASA storage array. It also provides information on Oracle database backup, restore, and clone using the NetApp SnapCenter UI tool for storage-efficient database operation in NetApp ASA systems.

This solution addresses the following use cases:

- Automated Oracle database deployment in NetApp ASA systems as primary database storage
- Oracle database backup and restore in NetApp ASA systems using NetApp SnapCenter tool
- Oracle database clone for dev/test or other use cases in NetApp ASA systems using NetApp SnapCenter tool
Audience

This solution is intended for the following people:

- A DBA who would like to deploy Oracle in NetApp ASA systems.
- A database solution architect who would like to test Oracle workloads in NetApp ASA systems.
- A storage administrator who would like to deploy and manage an Oracle database on NetApp ASA systems.
- An application owner who would like to stand up an Oracle database in NetApp ASA systems.

Solution test and validation environment

The testing and validation of this solution were performed in a lab setting that might not match the final deployment environment. See the section Key factors for deployment consideration for more information.

Architecture

Simplified, Automated Oracle Database Deployment on NetApp ASA with iSCSI

Hardware and software components

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetApp ASA A400</td>
<td>Version 9.13.1P1</td>
</tr>
<tr>
<td></td>
<td>2 NS224 shelves, 48 NVMe AFF drives with total 69.3 TiB capacity</td>
</tr>
<tr>
<td>UCSB-B200-M4</td>
<td>Intel® Xeon® CPU E5-2690 v4 @ 2.60GHz</td>
</tr>
<tr>
<td></td>
<td>4-node VMware ESXi cluster</td>
</tr>
</tbody>
</table>
### Oracle database configuration in the lab environment

<table>
<thead>
<tr>
<th>Server</th>
<th>Database</th>
<th>DB Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ora_01</td>
<td>NTAP1(NTAP1_PDB1,NTAP1_PD B2,NTAP1_PDB3)</td>
<td>iSCSI luns on ASA A400</td>
</tr>
<tr>
<td>ora_02</td>
<td>NTAP2(NTAP2_PDB1,NTAP2_PD B2,NTAP2_PDB3)</td>
<td>iSCSI luns on ASA A400</td>
</tr>
</tbody>
</table>

**Key factors for deployment consideration**

- **Oracle database storage layout.** In this automated Oracle deployment, we provision four database volumes to host Oracle binary, data, and logs by default. We then create two ASM disk groups from data and logs luns. Within the +DATA asm disk group, we provision two data luns in a volume on each ASA A400 cluster node. Within the +LOGS asm disk group, we create two luns in a log volume on a single ASA A400 node. Multiple luns laid out within an ONTAP volume provides better performance in general.

- **Multiple DB servers deployment.** The automation solution can deploy an Oracle container database to multiple DB servers in a single Ansible playbook run. Regardless of the number of DB servers, the playbook execution remains the same. In the event of multi-DB server deployments, the playbook builds with an algorithm to place database luns on dual controllers of ASA A400 optimally. The binary and logs luns of odd number DB server in server hosts index place on controller 1. The binary and logs luns of even number DB server in the server hosts index place on controller 2. The DB data luns evenly distributed to two controllers. Oracle ASM combines the data luns on two controllers into a single ASM disk group to fully utilize the processing power of both controllers.

- **iSCSI configuration.** The database VMs connect to ASA storage with the iSCSI protocol for storage access. You should configure dual paths on each controller node for redundancy and set up iSCSI multi-path on the DB server for multi-path storage access. Enable jumbo frame on storage network to maximize performance and throughput.
• **Oracle ASM redundancy level to use for each Oracle ASM disk group that you create.** Because the ASA A400 configures storage in RAID DP for data protection at the cluster disk level, you should use **External Redundancy**, which means that the option does not allow Oracle ASM to mirror the contents of the disk group.

• **Database backup.** NetApp provides a SnapCenter software suite for database backup, restore, and cloning with a user-friendly UI interface. NetApp recommends implementing such a management tool to achieve fast (under a minute) SnapShot backup, quick (minutes) database restore, and database clone.

**Solution deployment**

The following sections provide step-by-step procedures for automated Oracle 19c deployment and protection in NetApp ASA A400 with directly mounted database luns via iSCSI to DB VM in a single node Restart configuration with Oracle ASM as database volume manager.

**Prerequisites for deployment**
Deployment requires the following prerequisites.

1. It is assumed that the NetApp ASA storage array has been installed and configured. This includes iSCSI broadcast domain, LACP interface groups a0a on both controller nodes, iSCSI VLAN ports (a0a-<iscsi-a-vlan-id>, a0a-<iscsi-b-vlan-id>) on both controller nodes. The following link provides detailed step-by-step instructions if help is needed. Detailed guide - ASA A400

2. Provision a Linux VM as an Ansible controller node with the latest version of Ansible and Git installed. Refer to the following link for details: Getting Started with NetApp solution automation in section - Setup the Ansible Control Node for CLI deployments on RHEL / CentOS or Setup the Ansible Control Node for CLI deployments on Ubuntu / Debian.

3. Clone a copy of the NetApp Oracle deployment automation toolkit for iSCSI.

   ```bash
   git clone https://bitbucket.ngage.netapp.com/scm/ns-bb/na_oracle_deploy_iscsi.git
   ```

4. Provision a Windows server to run the NetApp SnapCenter UI tool with the latest version. Refer to the following link for details: Install the SnapCenter Server

5. Build two RHEL Oracle DB servers either bare metal or virtualized VM. Create an admin user on DB servers with sudo without password privilege and enable SSH private/public key authentication between Ansible host and Oracle DB server hosts. Stage following Oracle 19c installation files on DB servers /tmp/archive directory.

   ```
   installer_archives:
   - "LINUX.X64_193000_grid_home.zip"
   - "p34762026_190000_Linux-x86-64.zip"
   - "LINUX.X64_193000_db_home.zip"
   - "p34765931_190000_Linux-x86-64.zip"
   - "p6880880_190000_Linux-x86-64.zip"
   ```

   Ensure that you have allocated at least 50G in Oracle VM root volume to have sufficient space to stage Oracle installation files.

6. Watch the following video:

   Simplified and automated Oracle deployment on NetApp ASA with iSCSI

Automation parameter files
Ansible playbook executes database installation and configuration tasks with predefined parameters. For this Oracle automation solution, there are three user-defined parameter files that need user input before playbook execution.

- hosts - define targets that the automation playbook is running against.
- vars/vars.yml - the global variable file that defines variables that apply to all targets.
- host_vars/host_name.yml - the local variable file that defines variables that apply only to a local target. In our use case, these are the Oracle DB servers.

In addition to these user-defined variable files, there are several default variable files that contain default parameters that do not require change unless necessary. The following sections show how the user-defined variable files are configured.

**Parameter files configuration**
1. Ansible target hosts file configuration:

```yaml
# Enter NetApp ASA controller management IP address
[ontap]
172.16.9.32

# Enter Oracle servers names to be deployed one by one, follow by each Oracle server public IP address, and ssh private key of admin user for the server.
[oracle]
ora_01 ansible_host=10.61.180.21 ansible_ssh_private_key_file =ora_01.pem
ora_02 ansible_host=10.61.180.23 ansible_ssh_private_key_file =ora_02.pem
```

2. Global vars/vars.yml file configuration

```yaml
# Enter the supported ONTAP platform: on-prem, aws-fsx.
ontap_platform: on-prem

# Enter ONTAP cluster management user credentials
username: "xxxxxxxx"
password: "xxxxxxxx"

# Enter Oracle SVM iSCSI lif addresses. Each controller configures
```
with dual paths iscsi_a, iscsi_b for redundancy
ora_iscsi_lif_mgmt:
  - {name: '{{ svm_name }}_mgmt', address: 172.21.253.220, netmask: 255.255.255.0, vlan_name: ora_mgmt, vlan_id: 3509}

ora_iscsi_lifs_node1:
  - {name: '{{ svm_name }}_lif_1a', address: 172.21.234.221, netmask: 255.255.255.0, vlan_name: ora_iscsi_a, vlan_id: 3490}
  - {name: '{{ svm_name }}_lif_1b', address: 172.21.235.221, netmask: 255.255.255.0, vlan_name: ora_iscsi_b, vlan_id: 3491}
ora_iscsi_lifs_node2:
  - {name: '{{ svm_name }}_lif_2a', address: 172.21.234.223, netmask: 255.255.255.0, vlan_name: ora_iscsi_a, vlan_id: 3490}
  - {name: '{{ svm_name }}_lif_2b', address: 172.21.235.223, netmask: 255.255.255.0, vlan_name: ora_iscsi_b, vlan_id: 3491}

########################################################################
#########################################
###                   Linux env specific config variables
###
#########################################
# Enter RHEL subscription to enable repo
redhat_sub_username: xxxxxxxxx
redhat_sub_password: "xxxxxxxx"

########################################################################
#########################################
###                   Oracle DB env specific config variables
###
#########################################
# Enter Database domain name
db_domain: solutions.netapp.com

# Enter initial password for all required Oracle passwords. Change them after installation.
initial_pwd_all: xxxxxxxxx

3. Local DB server host_vars/host_name.yml configuration
# User configurable Oracle host specific parameters

# Enter container database SID. By default, a container DB is created with 3 PDBs within the CDB

oracle_sid: NTAP1

# Enter database shared memory size or SGA. CDB is created with SGA at 75% of memory_limit, MB. The grand total of SGA should not exceed 75% available RAM on node.

memory_limit: 8192

---

Playbook execution
There are a total of six playbooks in the automation toolkit. Each performs different task blocks and serves different purposes.

0-all_playbook.yml - execute playbooks from 1-4 in one playbook run.
1-ansible_requirements.yml - set up Ansible controller with required libs and collections.
2-linux_config.yml - execute Linux kernel configuration on Oracle DB servers.
3-ontap_config.yml - configure ONTAP svm/volumes/luns for Oracle database and grant DB server access to luns.
4-oracle_config.yml - install and configure Oracle on DB servers for grid infrastructure and create a container database.
5-destroy.yml - optional to undo the environment to dismantle all.

There are three options to run the playbooks with the following commands.

1. Execute all deployment playbooks in one combined run.

   ```bash
   ansible-playbook -i hosts 0-all_playbook.yml -u admin -e @vars/vars.yml
   ```

2. Execute playbooks one at a time with the number sequence from 1-4.

   ```bash
   ansible-playbook -i hosts 1-ansible_requirements.yml -u admin -e @vars/vars.yml
   ```

   ```bash
   ansible-playbook -i hosts 2-linux_config.yml -u admin -e @vars/vars.yml
   ```

   ```bash
   ansible-playbook -i hosts 3-ontap_config.yml -u admin -e @vars/vars.yml
   ```

   ```bash
   ansible-playbook -i hosts 4-oracle_config.yml -u admin -e @vars/vars.yml
   ```

3. Execute 0-all_playbook.yml with a tag.

   ```bash
   ansible-playbook -i hosts 0-all_playbook.yml -u admin -e -t tag
   ```
ansible-playbook -i hosts 0-all_playbook.yml -u admin -e @vars/vars.yml -t ansible_requirements

ansible-playbook -i hosts 0-all_playbook.yml -u admin -e @vars/vars.yml -t linux_config

ansible-playbook -i hosts 0-all_playbook.yml -u admin -e @vars/vars.yml -t ontap_config

ansible-playbook -i hosts 0-all_playbook.yml -u admin -e @vars/vars.yml -t oracle_config

4. Undo the environment

ansible-playbook -i hosts 5-destroy.yml -u admin -e @vars/vars.yml

Post execution validation
After the playbook run, login to the Oracle DB server as oracle user to validate that Oracle grid infrastructure and database are created successfully. Following is an example of Oracle database validation on host ora_01.

1. Validate the grid infrastructure and resources created.

```
[oracle@ora_01 ~]$ df -h
Filesystem  Size  Used  Avail  Use% Mounted on
devtmpfs  7.7G   40K   7.7G    1% /dev
tmpfs     7.8G  1.1G   6.7G  15% /dev/shm
tmpfs     7.8G  312M   7.5G   4% /run
tmpfs     7.8G     0   7.8G    0% /sys/fs/cgroup
/dev/mapper/rhel-root  44G  38G   6.8G  85% /
/dev/sda1  1014M  258M   757M  26% /boot
tmpfs     1.6G   12K   1.6G    1% /run/user/42
tmpfs     1.6G  4.0K   1.6G    1% /run/user/1000
/dev/mapper/ora_01_biny_01p1  40G  21G   20G  52% /u01
[oracle@ora_01 ~]$ asm
[oracle@ora_01 ~]$ crsctl stat res -t
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Target</th>
<th>State</th>
<th>Server</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ora.DATA.dg</td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>ora_01</td>
<td>STABLE</td>
</tr>
<tr>
<td>ora.LISTENER.lsnr</td>
<td>ONLINE</td>
<td>INTERMEDIATE</td>
<td>ora_01</td>
<td>Not All</td>
</tr>
<tr>
<td>gistered,STABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ora.LOGS.dg</td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>ora_01</td>
<td>STABLE</td>
</tr>
<tr>
<td>ora.asm</td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>ora_01</td>
<td></td>
</tr>
<tr>
<td>Started,STABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ora.ons</td>
<td>OFFLINE</td>
<td>OFFLINE</td>
<td>ora_01</td>
<td>STABLE</td>
</tr>
</tbody>
</table>

Cluster Resources
ora.cssd
  1        ONLINE  ONLINE       ora_01                   STABLE
ora.diskmon
  1        OFFLINE  OFFLINE                               STABLE
ora.driver.afd
  1        ONLINE  ONLINE       ora_01                   STABLE
ora.evmd
  1        ONLINE  ONLINE       ora_01                   STABLE
ora.ntap1.db
  1        ONLINE  ONLINE       ora_01
  Open,HOME=/u01/app/oracle/product/19.0.0/NTAP1,STABLE

[oracle@ora_01 ~]$ Ignore the Not All Endpoints Registered in State details. This results from a conflict of manual and dynamic database registration with the listener and can be safely ignored.

2. Validate ASM filter driver is working as expected.
3. Login to Oracle Enterprise Manager Express to validate database.
Enable additional port from sqlplus for login to individual container database or PDBs.

SQL> show pdbs

<table>
<thead>
<tr>
<th>CON_ID</th>
<th>CON_NAME</th>
<th>OPEN MODE</th>
<th>RESTRICTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>PDB$SEED</td>
<td>READ ONLY</td>
<td>NO</td>
</tr>
<tr>
<td>3</td>
<td>NTAP1_PDB1</td>
<td>READ WRITE</td>
<td>NO</td>
</tr>
<tr>
<td>4</td>
<td>NTAP1_PDB2</td>
<td>READ WRITE</td>
<td>NO</td>
</tr>
<tr>
<td>5</td>
<td>NTAP1_PDB3</td>
<td>READ WRITE</td>
<td>NO</td>
</tr>
</tbody>
</table>

SQL> alter session set container=NTAP1_PDB1;

Session altered.

SQL> select dbms_xdb_config.gethttpsport() from dual;

DBMS_XDB_CONFIG.GETHTTPSPORT()
-------------------------------
0

SQL> exec DBMS_XDB_CONFIG.SETHTTPSPORT(5501);

PL/SQL procedure successfully completed.

SQL> select dbms_xdb_config.gethttpsport() from dual;

DBMS_XDB_CONFIG.GETHTTPSPORT()
-------------------------------
5501

login to NTAP1_PDB1 from port 5501.
Oracle backup, restore, and clone with SnapCenter

Refer to TR-4979 Simplified, self-managed Oracle in VMware Cloud on AWS with guest-mounted FSx ONTAP section Oracle backup, restore, and clone with SnapCenter for details on setting up SnapCenter and executing the database backup, restore, and clone workflows.

Where to find additional information

To learn more about the information described in this document, review the following documents and/or websites:

- NETAPP ASA: ALL-FLASH SAN ARRAY
  https://www.netapp.com/data-storage/all-flash-san-storage-array/

- Installing Oracle Grid Infrastructure for a Standalone Server with a New Database Installation

- Installing and Configuring Oracle Database Using Response Files

- Use Red Hat Enterprise Linux 8.2 with ONTAP
  https://docs.netapp.com/us-en/ontap-sanhost/hu_rhel_82.html#all-san-array-configurations
Deploying Oracle Database

**Solution Overview**

This page describes the Automated method for deploying Oracle19c on NetApp ONTAP storage.

**Automated Deployment of Oracle19c for ONTAP on NFS**

Organizations are automating their environments to gain efficiencies, accelerate deployments, and reduce manual effort. Configuration management tools like Ansible are being used to streamline enterprise database operations. In this solution, we demonstrate how you can use Ansible to automate the provisioning and configuration of Oracle 19c with NetApp ONTAP. By enabling storage administrators, systems administrators, and DBAs to consistently and rapidly deploy new storage, configure database servers, and install Oracle 19c software, you achieve the following benefits:
• Eliminate design complexities and human errors, and implement a repeatable consistent deployment and best practices
• Decrease time for provisioning of storage, configuration of DB hosts, and Oracle installation
• Increase database administrators, systems and storage administrators productivity
• Enable scaling of storage and databases with ease

NetApp provides customers with validated Ansible modules and roles to accelerate deployment, configuration, and lifecycle management of your Oracle database environment. This solution provides instruction and Ansible playbook code, to help you:

• Create and configure ONTAP NFS storage for Oracle Database
• Install Oracle 19c on RedHat Enterprise Linux 7/8 or Oracle Linux 7/8
• Configure Oracle 19c on ONTAP NFS storage

For more details or to begin, please see the overview videos below.

**AWX/Tower Deployments**

**Part 1: Getting Started, Requirements, Automation Details and Initial AWX/Tower Configuration**

**AWX Deployment**

**Part 2: Variables and Running the Playbook**

**AWX Playbook Run**

**CLI Deployment**

**Part 1: Getting Started, Requirements, Automation Details and Ansible Control Host Setup**

**CLI Deployment**

**Part 2: Variables and Running the Playbook**

**CLI Playbook Run**

**Getting started**

This solution has been designed to be run in an AWX/Tower environment or by CLI on an Ansible control host.

**AWX/Tower**

For AWX/Tower environments, you are guided through creating an inventory of your ONTAP cluster management and Oracle server (IPs and hostnames), creating credentials, configuring a project that pulls the Ansible code from NetApp Automation Github, and the Job Template that launches the automation.

1. Fill out the variables specific to your environment, and copy and paste them into the Extra Vars fields in your job template.
2. After the extra vars have been added to your job template, you can launch the automation.
3. The job template is run in three phases by specifying tags for ontap_config, linux_config, and oracle_config.
CLI via the Ansible control host

1. To configure the Linux host so that it can be used as an Ansible control host
   [click here for detailed instructions]

2. After the Ansible control host is configured, you can git clone the Ansible Automation repository.

3. Edit the hosts file with the IPs and/or hostnames of your ONTAP cluster management and Oracle server’s management IPs.

4. Fill out the variables specific to your environment, and copy and paste them into the `vars.yml` file.

5. Each Oracle host has a variable file identified by its hostname that contains host-specific variables.

6. After all variable files have been completed, you can run the playbook in three phases by specifying tags for `ontap_config`, `linux_config`, and `oracle_config`.

Requirements

<table>
<thead>
<tr>
<th>Environment</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ansible environment</td>
<td>AWX/Tower or Linux host to be the Ansible control host</td>
</tr>
<tr>
<td></td>
<td>Ansible v.2.10 and higher</td>
</tr>
<tr>
<td></td>
<td>Python 3</td>
</tr>
<tr>
<td></td>
<td>Python libraries</td>
</tr>
<tr>
<td></td>
<td>- netapp-lib</td>
</tr>
<tr>
<td></td>
<td>- xmltodict</td>
</tr>
<tr>
<td></td>
<td>- jmespath</td>
</tr>
<tr>
<td>ONTAP</td>
<td>ONTAP version 9.3 - 9.7</td>
</tr>
<tr>
<td></td>
<td>Two data aggregates</td>
</tr>
<tr>
<td></td>
<td>NFS vlan and ifgrp created</td>
</tr>
<tr>
<td>Oracle server(s)</td>
<td>RHEL 7/8</td>
</tr>
<tr>
<td></td>
<td>Oracle Linux 7/8</td>
</tr>
<tr>
<td></td>
<td>Network interfaces for NFS, public, and optional mgmt</td>
</tr>
<tr>
<td></td>
<td>Oracle installation files on Oracle servers</td>
</tr>
</tbody>
</table>

Automation Details

This automated deployment is designed with a single Ansible playbook that consists of three separate roles.

The roles are for ONTAP, Linux, and Oracle configurations.

The following table describes which tasks are being automated.

<table>
<thead>
<tr>
<th>Role</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ontap_config</td>
<td>Pre-check of the ONTAP environment</td>
</tr>
<tr>
<td></td>
<td>Creation of NFS based SVM for Oracle</td>
</tr>
<tr>
<td></td>
<td>Creation of export policy</td>
</tr>
<tr>
<td></td>
<td>Creation of volumes for Oracle</td>
</tr>
<tr>
<td></td>
<td>Creation of NFS LIFs</td>
</tr>
<tr>
<td>Role</td>
<td>Tasks</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>linux_config</td>
<td>Create mount points and mount NFS volumes</td>
</tr>
<tr>
<td></td>
<td>Verify NFS mounts</td>
</tr>
<tr>
<td></td>
<td>OS specific configuration</td>
</tr>
<tr>
<td></td>
<td>Create Oracle directories</td>
</tr>
<tr>
<td></td>
<td>Configure hugepages</td>
</tr>
<tr>
<td></td>
<td>Disable SELinux and firewall daemon</td>
</tr>
<tr>
<td></td>
<td>Enable and start chronyd service</td>
</tr>
<tr>
<td></td>
<td>increase file descriptor hard limit</td>
</tr>
<tr>
<td></td>
<td>Create pam.d session file</td>
</tr>
<tr>
<td>oracle_config</td>
<td>Oracle software installation</td>
</tr>
<tr>
<td></td>
<td>Create Oracle listener</td>
</tr>
<tr>
<td></td>
<td>Create Oracle databases</td>
</tr>
<tr>
<td></td>
<td>Oracle environment configuration</td>
</tr>
<tr>
<td></td>
<td>Save PDB state</td>
</tr>
<tr>
<td></td>
<td>Enable instance archive mode</td>
</tr>
<tr>
<td></td>
<td>Enable DNFS client</td>
</tr>
<tr>
<td></td>
<td>Enable database auto startup and shutdown between OS reboots</td>
</tr>
</tbody>
</table>

**Default parameters**

To simplify automation, we have preset many required Oracle deployment parameters with default values. It is generally not necessary to change the default parameters for most deployments. A more advanced user can make changes to the default parameters with caution. The default parameters are located in each role folder under defaults directory.

**Deployment instructions**

Before starting, download the following Oracle installation and patch files and place them in the /tmp/archive directory with read, write, and execute access for all users on each DB server to be deployed. The automation tasks look for the named installation files in that particular directory for Oracle installation and configuration.

```
LINUX.X64_193000_db_home.zip -- 19.3 base installer
p31281355_190000_Linux-x86-64.zip -- 19.8 RU patch
p6880880_190000_Linux-x86-64.zip -- opatch version 12.2.0.1.23
```

**License**

You should read license information as stated in the Github repository. By accessing, downloading, installing, or using the content in this repository, you agree the terms of the license laid out here.
Step-by-step deployment procedure

This page describes the Automated method for deploying Oracle19c on NetApp ONTAP storage.

AWX/Tower deployment Oracle 19c Database

1. Create the inventory, group, hosts, and credentials for your environment

This section describes the setup of inventory, groups, hosts, and access credentials in AWX/Ansible Tower that prepare the environment for consuming NetApp automated solutions.

1. Configure the inventory.
   a. Navigate to Resources → Inventories → Add, and click Add Inventory.
   b. Provide the name and organization details, and click Save.
   c. On the Inventories page, click the inventory created.
   d. If there are any inventory variables, paste them in the variables field.
   e. Navigate to the Groups sub-menu and click Add.
   f. Provide the name of the group for ONTAP, paste the group variables (if any) and click Save.
   g. Repeat the process for another group for Oracle.
   h. Select the ONTAP group created, go to the Hosts sub-menu and click Add New Host.
   i. Provide the IP address of the ONTAP cluster management IP, paste the host variables (if any), and click Save.
   j. This process must be repeated for the Oracle group and Oracle host(s) management IP/hostname.

2. Create credential types. For solutions involving ONTAP, you must configure the credential type to match username and password entries.
   a. Navigate to Administration → Credential Types, and click Add.
   b. Provide the name and description.
   c. Paste the following content in Input Configuration:
fields:
- id: username
  type: string
  label: Username
- id: password
  type: string
  label: Password
  secret: true
- id: vsadmin_password
  type: string
  label: vsadmin_password
  secret: true

a. Paste the following content into Injector Configuration:

```yaml
extra_vars:
  password: '{{ password }}'
  username: '{{ username }}'
  vsadmin_password: '{{ vsadmin_password }}'
```

1. Configure the credentials.
   a. Navigate to Resources → Credentials, and click Add.
   b. Enter the name and organization details for ONTAP.
   c. Select the custom Credential Type you created for ONTAP.
   d. Under Type Details, enter the username, password, and vsadmin_password.
   e. Click Back to Credential and click Add.
   f. Enter the name and organization details for Oracle.
   g. Select the Machine credential type.
   h. Under Type Details, enter the Username and Password for the Oracle hosts.
   i. Select the correct Privilege Escalation Method, and enter the username and password.

2. Create a project

   1. Go to Resources → Projects, and click Add.
      a. Enter the name and organization details.
      b. Select Git in the Source Control Credential Type field.
      c. enter `https://github.com/NetApp-Automation/na_oracle19c_deploy.git` as the source control URL.
      d. Click Save.
      e. The project might need to sync occasionally when the source code changes.
3. Configure Oracle host_vars

The variables defined in this section are applied to each individual Oracle server and database.

1. Input your environment-specific parameters in the following embedded Oracle hosts variables or host_vars form.

The items in blue must be changed to match your environment.

Host VARS Config

```
# Add your Oracle Host
ansible_host: "10.61.180.15"

# Oracle db log archive mode: true - ARCHIVELOG or false - NOARCHIVELOG
log_archive_mode: "true"

# Number of pluggable databases per container instance identified by sid.
Pdb_name specifies the prefix for container database naming in this case
cdb2_pdb1, cdb2_pdb2, cdb2_pdb3
oracle_sid: "cdb2"
pdb_num: "3"
pdb_name: "{{ oracle_sid }}_pdb"

# CDB listener port, use different listener port for additional CDB on same host
listener_port: "1523"

# CDB is created with SGA at 75% of memory_limit, MB. Consider how many databases to be hosted on the node and how much ram to be allocated to each DB. The grand total SGA should not exceed 75% available RAM on node.
memory_limit: "5464"

# Set "em_configuration: DBEXPRESS" to install enterprise manager express and choose a unique port from 5500 to 5599 for each sid on the host.
# Leave them black if em express is not installed.
em_configuration: "DBEXPRESS"
em_express_port: "5501"
```

# {{groups.oracle[0]}} represents first Oracle DB server as defined in Oracle hosts group [oracle]. For concurrent multiple Oracle DB servers deployment, [0] will be incremented for each additional DB server. For example, {{groups.oracle[1]}} represents DB server 2,
"{{groups.oracle[2]}}" represents DB server 3 ... As a good practice and
the default, minimum three volumes is allocated to a DB server with
 corresponding /u01, /u02, /u03 mount points, which store oracle binary,
 oracle data, and oracle recovery files respectively. Additional volumes
can be added by click on "More NFS volumes" but the number of volumes
allocated to a DB server must match with what is defined in global vars
file by volumes_nfs parameter, which dictates how many volumes are to be
created for each DB server.

```yaml
copy
host_datastores_nfs:
  - {vol_name: "{{groups.oracle[0]}}_u01", aggr_name: "aggr01_node01",
    lif: "172.21.94.200", size: "25"}
  - {vol_name: "{{groups.oracle[0]}}_u02", aggr_name: "aggr01_node01",
    lif: "172.21.94.200", size: "25"}
  - {vol_name: "{{groups.oracle[0]}}_u03", aggr_name: "aggr01_node01",
    lif: "172.21.94.200", size: "25"}
```

a. Fill in all variables in the blue fields.
b. After completing variables input, click the Copy button on the form to copy all variables to be transferred to
   AWX or Tower.
c. Navigate back to AWX or Tower and go to Resources → Hosts, and select and open the Oracle server
   configuration page.
d. Under the Details tab, click edit and paste the copied variables from step 1 to the Variables field under the
   YAML tab.
e. Click Save.
f. Repeat this process for any additional Oracle servers in the system.

4. Configure global variables

Variables defined in this section apply to all Oracle hosts, databases, and the ONTAP cluster.

1. Input your environment-specific parameters in following embedded global variables or vars form.

   The items in blue must be changed to match your environment.
# Change only if you are changing the group name either in inventory/hosts file or in inventory groups in case of AWX/Tower
hosts_group: "ontap"

# CA signed certificates (ONLY CHANGE to 'true' IF YOU ARE USING CA SIGNED CERTIFICATES)
ca_signed_certs: "false"

# Names of the Nodes in the ONTAP Cluster
nodes:
  - "AFF-01"
  - "AFF-02"

# Storage VLANs
# Add additional rows for vlans as necessary
storage_vlans:
  - {vlan_id: "203", name: "infra_NFS", protocol: "NFS"}

More Storage VLANs Enter Storage VLANs details

# Details of the Data Aggregates that need to be created
# If Aggregate creation takes longer, subsequent tasks of creating volumes may fail.
# There should be enough disks already zeroed in the cluster, otherwise aggregate create will zero the disks and will take long time
data_aggregates:
  - {aggr_name: "aggr01_node01"}
  - {aggr_name: "aggr01_node02"}

# SVM name
svm_name: "ora_svm"

# SVM Management LIF Details
svm_mgmt_details:
  - {address: "172.21.91.100", netmask: "255.255.255.0", home_port: "e0M"}

# NFS storage parameters when data_protocol set to NFS. Volume named after Oracle hosts name identified by mount point as follow for oracle DB server 1. Each mount point dedicates to a particular Oracle files: u01 - Oracle binary, u02 - Oracle data, u03 - Oracle redo. Add additional volumes by click on "More NFS volumes" and also add the volumes list to corresponding host vars as host_datastores_nfs variable. For multiple DB server deployment, additional volumes sets needs to be added for additional DB server. Input variable "{{groups.oracle[1]}}_u01", "{{groups.oracle[1]}}_u02", and "{{groups.oracle[1]}}_u03" as vol_name for second DB server. Place volumes for multiple DB servers alternatingly between controllers for balanced IO performance, e.g. DB server 1 on
controller node1, DB server 2 on controller node2 etc. Make sure match lif address with controller node.

volumes_nfs:
  - {vol_name: "{{groups.oracle[0]}}_u01", aggr_name: "aggr01_node01", lif: "172.21.94.200", size: "25"}
  - {vol_name: "{{groups.oracle[0]}}_u02", aggr_name: "aggr01_node01", lif: "172.21.94.200", size: "25"}
  - {vol_name: "{{groups.oracle[0]}}_u03", aggr_name: "aggr01_node01", lif: "172.21.94.200", size: "25"}

#NFS LIFs IP address and netmask

nfs_lifs_details:
  - address: "172.21.94.200" # for node-1
    netmask: "255.255.255.0"
  - address: "172.21.94.201" # for node-2
    netmask: "255.255.255.0"

#NFS client match

client_match: "172.21.94.0/24"

#############################################################
### Linux env specific config variables ###
#############################################################

#NFS Mount points for Oracle DB volumes

mount_points:
  - "/u01"
  - "/u02"
  - "/u03"

# Up to 75% of node memory size divided by 2mb. Consider how many databases to be hosted on the node and how much ram to be allocated to each DB.
# Leave it blank if hugepage is not configured on the host.

hugepages_nr: "1234"

# RedHat subscription username and password

redhat_sub_username: "xxx"
redhat_sub_password: "xxx"

#############################################################
### DB env specific install and config variables ###

```plaintext
# Set initial password for all required Oracle passwords. Change them after installation.

initial_pwd_all: "netapp123"
```

1. Fill in all variables in blue fields.

2. After completing variables input, click the Copy button on the form to copy all variables to be transferred to AWX or Tower into the following job template.

5. **Configure and launch the job template.**

1. **Create the job template.**
   a. Navigate to Resources → Templates → Add and click Add Job Template.
   b. Enter the name and description
   c. Select the Job type; Run configures the system based on a playbook, and Check performs a dry run of a playbook without actually configuring the system.
   d. Select the corresponding inventory, project, playbook, and credentials for the playbook.
   e. Select the all_playbook.yml as the default playbook to be executed.
   f. Paste global variables copied from step 4 into the Template Variables field under the YAML tab.
   g. Check the box Prompt on Launch in the Job Tags field.
   h. Click Save.

2. **Launch the job template.**
   a. Navigate to Resources → Templates.
   b. Click the desired template and then click Launch.
   c. When prompted on launch for Job Tags, type in requirements_config. You might need to click the Create Job Tag line below requirements_config to enter the job tag.

    **requirements_config** ensures that you have the correct libraries to run the other roles.

   a. Click Next and then Launch to start the job.
   b. Click View → Jobs to monitor the job output and progress.
   c. When prompted on launch for Job Tags, type in ontap_config. You might need to click the Create "Job Tag" line right below ontap_config to enter the job tag.
   d. Click Next and then Launch to start the job.
   e. Click View → Jobs to monitor the job output and progress
   f. After the ontap_config role has completed, run the process again for linux_config.
   g. Navigate to Resources → Templates.
h. Select the desired template and then click Launch.

i. When prompted on launch for the Job Tags type in linux_config, you might need to select the Create "job tag" line right below linux_config to enter the job tag.

j. Click Next and then Launch to start the job.

k. Select View → Jobs to monitor the job output and progress.

l. After the linux_config role has completed, run the process again for oracle_config.

m. Go to Resources → Templates.

n. Select the desired template and then click Launch.

o. When prompted on launch for Job Tags, type oracle_config. You might need to select the Create "Job Tag" line right below oracle_config to enter the job tag.

p. Click Next and then Launch to start the job.

q. Select View → Jobs to monitor the job output and progress.

6. Deploy additional database on same Oracle host

The Oracle portion of the playbook creates a single Oracle container database on an Oracle server per execution. To create additional container databases on the same server, complete the following steps.

1. Revise host_vars variables.
   a. Go back to step 2 - Configure Oracle host_vars.
   b. Change the Oracle SID to a different naming string.
   c. Change the listener port to different number.
   d. Change the EM Express port to a different number if you are installing EM Express.
   e. Copy and paste the revised host variables to the Oracle Host Variables field in the Host Configuration Detail tab.

2. Launch the deployment job template with only the oracle_config tag.

3. Log in to Oracle server as oracle user and execute the following commands:

   ```bash
   ps -ef | grep ora
   ```

   This will list oracle processes if installation completed as expected and oracle DB started.

4. Log in to the database to check the db configuration settings and the PDBs created with the following command sets.
[oracle@localhost ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Thu May 6 12:52:51 2021
Version 19.8.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.8.0.0.0

SQL>

SQL> select name, log_mode from v$database;
NAME      LOG_MODE
--------- ------------
CDB2      ARCHIVELOG

SQL> show p dbs

CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
2 PDB$SEED                       READ ONLY  NO
3 CDB2_PDB1                      READ WRITE NO
4 CDB2_PDB2                      READ WRITE NO
5 CDB2_PDB3                      READ WRITE NO

col svrname form a30
col dirname form a30
select svrname, dirname, nfsversion from v$dnfs_servers;

SQL> col svrname form a30
SQL> col dirname form a30
SQL> select svrname, dirname, nfsversion from v$dnfs_servers;

SVRNAME                        DIRNAME                        NFSVERSION
------------------------------ ------------------------------ ----------------
172.21.126.200                 /rhelora03_u02                 NFSv3.0
172.21.126.200                 /rhelora03_u03                 NFSv3.0
172.21.126.200                 /rhelora03_u01                 NFSv3.0

This confirms that dNFS is working properly.

5. Connect to database via listener to check the Oracle listener configuration with the following command.
Change to the appropriate listener port and database service name.
[oracle@localhost ~]$ sqlplus
system@//localhost:1523/cdb2_pdb1.cie.netapp.com

SQL*Plus: Release 19.0.0.0.0 - Production on Thu May 6 13:19:57 2021
Version 19.8.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Enter password:
Last Successful login time: Wed May 05 2021 17:11:11 -04:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.8.0.0.0

SQL> show user
USER is "SYSTEM"
SQL> show con_name
CON_NAME
CDB2_PDB1

This confirms that Oracle listener is working properly.

Where to go for help?
If you need help with the toolkit, please join the NetApp Solution Automation community support slack channel and look for the solution-automation channel to post your questions or inquiries.

Step-by-step deployment procedure
This document details the deployment of Oracle 19c using the automation command line interface (cli).

CLI deployment Oracle 19c Database
This section covers the steps required to prepare and deploy Oracle19c Database with the CLI. Make sure that you have reviewed the Getting Started and Requirements section and prepared your environment accordingly.

Download Oracle19c repo
1. From your ansible controller, run the following command:

```bash
git clone https://github.com/NetApp-Automation/na_oracle19c_deploy.git
```

2. After downloading the repository, change directories to na_oracle19c_deploy <cd na_oracle19c_deploy>.
**Edit the hosts file**

Complete the following before deployment:

1. Edit your hosts file na_oracle19c_deploy directory.
2. Under [ontap], change the IP address to your cluster management IP.
3. Under the [oracle] group, add the oracle hosts names. The host name must be resolved to its IP address either through DNS or the hosts file, or it must be specified in the host.
4. After you have completed these steps, save any changes.

The following example depicts a host file:

```
#ONTAP Host
[ontap]
"10.61.184.183"

#Oracle hosts
[oracle]
"rtpora01"
"rtpora02"
```

This example executes the playbook and deploys oracle 19c on two oracle DB servers concurrently. You can also test with just one DB server. In that case, you only need to configure one host variable file.

![Info icon] The playbook executes the same way regardless of how many Oracle hosts and databases you deploy.

**Edit the host_name.yml file under host_vars**

Each Oracle host has its host variable file identified by its host name that contains host-specific variables. You can specify any name for your host. Edit and copy the `host_vars` from the Host VARS Config section and paste it into your desired `host_name.yml` file.

![Info icon] The items in blue must be changed to match your environment.

**Host VARS Config**

```
# Add your Oracle Host
```
ansible_host: "10.61.180.15"

# Oracle db log archive mode: true - ARCHIVELOG or false - NOARCHIVELOG
log_archive_mode: "true"

# Number of pluggable databases per container instance identified by sid. Pdb_name specifies the prefix for container database naming in this case cdb2_pdb1, cdb2_pdb2, cdb2_pdb3
oracle_sid: "cdb2"
pdb_num: "3"
pdb_name: "{{ oracle_sid }}_pdb"

# CDB listener port, use different listener port for additional CDB on same host
listener_port: "1523"

# CDB is created with SGA at 75% of memory_limit, MB. Consider how many databases to be hosted on the node and how much ram to be allocated to each DB. The grand total SGA should not exceed 75% available RAM on node.
memory_limit: "5464"

# Set "em_configuration: DBEXPRESS" to install enterprise manager express and choose a unique port from 5500 to 5599 for each sid on the host. Leave them black if em express is not installed.
em_configuration: "DBEXPRESS"
em_express_port: "5501"

# {{groups.oracle[0]}} represents first Oracle DB server as defined in Oracle hosts group [oracle]. For concurrent multiple Oracle DB servers deployment, [0] will be incremented for each additional DB server. For example, {{groups.oracle[1]}}" represents DB server 2, "{{groups.oracle[2]}}" represents DB server 3 ... As a good practice and the default, minimum three volumes is allocated to a DB server with corresponding /u01, /u02, /u03 mount points, which store oracle binary, oracle data, and oracle recovery files respectively. Additional volumes can be added by click on "More NFS volumes" but the number of volumes allocated to a DB server must match with what is defined in global vars file by volumes_nfs parameter, which dictates how many volumes are to be created for each DB server.
host_datastores_nfs:
  - {vol_name: "{{groups.oracle[0]}}_u01", aggr_name: "aggr01_node01", lif: "172.21.94.200", size: "25"}
  - {vol_name: "{{groups.oracle[0]}}_u02", aggr_name: "aggr01_node01", lif: "172.21.94.200", size: "25"}
  - {vol_name: "{{groups.oracle[0]}}_u03", aggr_name: "aggr01_node01", lif: "172.21.94.200", size: "25"}
Edit the vars.yml file

The vars.yml file consolidates all environment-specific variables (ONTAP, Linux, or Oracle) for Oracle deployment.

1. Edit and copy the variables from the VARS section and paste these variables into your vars.yml file.

```yaml
#Inventory group name
#Default inventory group name - 'ontap'
hosts_group: "ontap"

#CA_signed_certificates (ONLY CHANGE to 'true' IF YOU ARE USING CA SIGNED CERTIFICATES)
ca_signed_certs: "false"

#Names of the Nodes in the ONTAP Cluster
nodes:
  - "AFF-01"
  - "AFF-02"

#Storage VLANs
#Add additional rows for vlans as necessary
storage_vlans:
  - {vlan_id: "203", name: "infra_NFS", protocol: "NFS"}

#Details of the Data Aggregates that need to be created
#If Aggregate creation takes longer, subsequent tasks of creating volumes may fail.
#There should be enough disks already zeroed in the cluster, otherwise aggregate create will zero the disks and will take long time
data_aggregates:
  - {aggr_name: "aggr01_node01"}
  - {aggr_name: "aggr01_node02"}

#SVM name
# SVM Management LIF Details

svm_mgmt_details:

- {address: "172.21.91.100", netmask: "255.255.255.0", home_port: "e0M"}

# NFS storage parameters when data protocol set to NFS. Volume named after Oracle hosts name identified by mount point as follow for oracle DB server 1. Each mount point dedicates to a particular Oracle files: u01 - Oracle binary, u02 - Oracle data, u03 - Oracle redo. Add additional volumes by click on "More NFS volumes" and also add the volumes list to corresponding host_vars as host_datastores_nfs variable. For multiple DB server deployment, additional volumes sets needs to be added for additional DB server. Input variable "{{groups.oracle[1]}}_u01", "{{groups.oracle[1]}}_u02", and "{{groups.oracle[1]}}_u03" as vol_name for second DB server. Place volumes for multiple DB servers alternatingly between controllers for balanced IO performance, e.g. DB server 1 on controller node1, DB server 2 on controller node2 etc. Make sure match lif address with controller node.

volumes_nfs:

- {vol_name: "{{groups.oracle[0]}}_u01", aggr_name: "aggr01_node01", lif: "172.21.94.200", size: "25"}
- {vol_name: "{{groups.oracle[0]}}_u02", aggr_name: "aggr01_node01", lif: "172.21.94.200", size: "25"}
- {vol_name: "{{groups.oracle[0]}}_u03", aggr_name: "aggr01_node01", lif: "172.21.94.200", size: "25"}

#NFS LIFs IP address and netmask

nfs_lifs_details:

- address: "172.21.94.200" #for node-1 netmask: "255.255.255.0"
- address: "172.21.94.201" #for node-2 netmask: "255.255.255.0"

#NFS client match

client_match: "172.21.94.0/24"

#NFS Mount points for Oracle DB volumes

mount_points:
Run the playbook

After completing the required environment prerequisites and copying the variables into vars.yml and your_host.yml, you are now ready to deploy the playbooks.

<username> must be changed to match your environment.

1. Run the ONTAP playbook by passing the correct tags and ONTAP cluster username. Fill the password for ONTAP cluster, and vsadmin when prompted.

   ansible-playbook -i hosts all_playbook.yml -u username -k -K -t ontap_config -e @vars/vars.yml

2. Run the Linux playbook to execute Linux portion of deployment. Input for admin ssh password as well as sudo password.

   ansible-playbook -i hosts all_playbook.yml -u username -k -K -t linux_config -e @vars/vars.yml
3. Run the Oracle playbook to execute Oracle portion of deployment. Input for admin ssh password as well as sudo password.

```bash
ansible-playbook -i hosts all_playbook.yml -u username -k -K -t oracle_config -e @vars/vars.yml
```

### Deploy Additional Database on Same Oracle Host

The Oracle portion of the playbook creates a single Oracle container database on an Oracle server per execution. To create additional container database on the same server, complete the following steps:

1. Revise the host_vars variables.
   a. Go back to step 3 - Edit the `host_name.yml` file under `host_vars`.
   b. Change the Oracle SID to a different naming string.
   c. Change the listener port to different number.
   d. Change the EM Express port to a different number if you have installed EM Express.
   e. Copy and paste the revised host variables to the Oracle host variable file under `host_vars`.

2. Execute the playbook with the `oracle_config` tag as shown above in Run the playbook.

### Validate Oracle installation

1. Log in to Oracle server as oracle user and execute the following commands:

```bash
ps -ef | grep ora
```

   This will list oracle processes if installation completed as expected and oracle DB started

2. Log in to the database to check the db configuration settings and the PDBs created with the following command sets.
[oracle@localhost ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Thu May 6 12:52:51 2021
Version 19.8.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.8.0.0.0

SQL>

SQL> select name, log_mode from v$database;
NAME      LOG_MODE
--------- ------------
CDB2      ARCHIVELOG

SQL> show pdbs

CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
2 PDB$SEED                       READ ONLY  NO
3 CDB2_PDB1                      READ WRITE NO
4 CDB2_PDB2                      READ WRITE NO
5 CDB2_PDB3                      READ WRITE NO

col svrname form a30
col dirname form a30
select svrname, dirname, nfsversion from v$dnfs_servers;

SVRNAME                        DIRNAME                        NFSVERSION
------------------------------ ------------------------------ ----------------
172.21.126.200                 /rhelora03_u02                 NFSv3.0
172.21.126.200                 /rhelora03_u03                 NFSv3.0
172.21.126.200                 /rhelora03_u01                 NFSv3.0

This confirms that dNFS is working properly.

3. Connect to database via listener to check the Oracle listener configuration with the following command. Change to the appropriate listener port and database service name.
This confirms that Oracle listener is working properly.

**Where to go for help?**

If you need help with the toolkit, please join the NetApp Solution Automation community support slack channel and look for the solution-automation channel to post your questions or inquiries.

**Solution Overview**

This page describes the Automated method for deploying Oracle19c on NetApp ONTAP storage.

**Automated Data Protection for Oracle Databases**

Organizations are automating their environments to gain efficiencies, accelerate deployments, and reduce manual effort. Configuration management tools like Ansible are being used to streamline enterprise database operations. In this solution, we demonstrate how you can use Ansible to automate the data protection of Oracle with NetApp ONTAP. By enabling storage administrators, systems administrators, and DBAs to consistently and rapidly setup data replication to an offsite data center or to public cloud, you achieve the following benefits:

- Eliminate design complexities and human errors, and implement a repeatable consistent deployment and best practices
- Decrease time for configuration of Intercluster replication, CVO instantiation, and recovery of Oracle databases
- Increase database administrators, systems and storage administrators productivity
NetApp provides customers with validated Ansible modules and roles to accelerate deployment, configuration, and lifecycle management of your Oracle database environment. This solution provides instruction and Ansible playbook code, to help you:

**On Prem to on prem replication**
- Create intercluster lifs on source and destination
- Establish cluster and vserver peering
- Create and initialize SnapMirror of Oracle volumes
- Create a replication schedule through AWX/Tower for Oracle binaries, databases, and logs
- Restore Oracle DB on the destination, and bring database online

**On Prem to CVO in AWS**
- Create AWS connector
- Create CVO instance in AWS
- Add On-Prem cluster to Cloud Manager
- Create intercluster lifs on source
- Establish cluster and vserver peering
- Create and initialize SnapMirror of Oracle volumes
- Create a replication schedule through AWX/Tower for Oracle binaries, databases, and logs
- Restore Oracle DB on the destination, and bring database online

After you are ready, click here for getting started with the solution.

**Getting started**

This solution has been designed to be run in an AWX/Tower environment.

**AWX/Tower**

For AWX/Tower environments, you are guided through creating an inventory of your ONTAP cluster management and Oracle server (IPs and hostnames), creating credentials, configuring a project that pulls the Ansible code from NetApp Automation Github, and the Job Template that launches the automation.

1. The solution has been designed to run in a private cloud scenario (on-premise to on-premise), and hybrid cloud (on-premise to public cloud Cloud Volumes ONTAP [CVO])
2. Fill out the variables specific to your environment, and copy and paste them into the Extra Vars fields in your job template.
3. After the extra vars have been added to your job template, you can launch the automation.
4. The automation is set to be ran three phases (Setup, Replication Schedule for Oracle Binaries, Database, Logs, and Replication Schedule just for Logs), and a forth phase to recovering the database at a DR site.
5. For detailed instructions for obtaining the keys and tokens necessary for the CVO Data Protection visit Gather Pre-requisites For CVO and Connector Deployments
Requirements
### On-Prem

<table>
<thead>
<tr>
<th>Environment</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ansible environment</td>
<td>AWX/Tower</td>
</tr>
<tr>
<td></td>
<td>Ansible v.2.10 and higher</td>
</tr>
<tr>
<td></td>
<td>Python 3</td>
</tr>
<tr>
<td></td>
<td>Python libraries</td>
</tr>
<tr>
<td></td>
<td>- netapp-lib</td>
</tr>
<tr>
<td></td>
<td>- xmltodict</td>
</tr>
<tr>
<td></td>
<td>- jmespath</td>
</tr>
<tr>
<td>ONTAP</td>
<td>ONTAP version 9.8 +</td>
</tr>
<tr>
<td></td>
<td>Two data aggregates</td>
</tr>
<tr>
<td></td>
<td>NFS vlan and ifgrp created</td>
</tr>
<tr>
<td>Oracle server(s)</td>
<td>RHEL 7/8</td>
</tr>
<tr>
<td></td>
<td>Oracle Linux 7/8</td>
</tr>
<tr>
<td></td>
<td>Network interfaces for NFS, public, and optional mgmt</td>
</tr>
<tr>
<td></td>
<td>Existing Oracle environment on source, and the equivalent Linux operating system at the destination (DR Site or Public Cloud)</td>
</tr>
</tbody>
</table>

### CVO

<table>
<thead>
<tr>
<th>Environment</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ansible environment</td>
<td>AWX/Tower</td>
</tr>
<tr>
<td></td>
<td>Ansible v.2.10 and higher</td>
</tr>
<tr>
<td></td>
<td>Python 3</td>
</tr>
<tr>
<td></td>
<td>Python libraries</td>
</tr>
<tr>
<td></td>
<td>- netapp-lib</td>
</tr>
<tr>
<td></td>
<td>- xmltodict</td>
</tr>
<tr>
<td></td>
<td>- jmespath</td>
</tr>
<tr>
<td>ONTAP</td>
<td>ONTAP version 9.8 +</td>
</tr>
<tr>
<td></td>
<td>Two data aggregates</td>
</tr>
<tr>
<td></td>
<td>NFS vlan and ifgrp created</td>
</tr>
<tr>
<td>Oracle server(s)</td>
<td>RHEL 7/8</td>
</tr>
<tr>
<td></td>
<td>Oracle Linux 7/8</td>
</tr>
<tr>
<td></td>
<td>Network interfaces for NFS, public, and optional mgmt</td>
</tr>
<tr>
<td></td>
<td>Existing Oracle environment on source, and the equivalent Linux operating system at the destination (DR Site or Public Cloud)</td>
</tr>
<tr>
<td></td>
<td>Set appropriate swap space on the Oracle EC2 instance, by default some EC2 instances are deployed with 0 swap</td>
</tr>
<tr>
<td>Environment</td>
<td>Requirements</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Cloud Manager/AWS</td>
<td>AWS Access/Secret Key</td>
</tr>
<tr>
<td></td>
<td>NetApp Cloud Manager Account</td>
</tr>
<tr>
<td></td>
<td>NetApp Cloud Manager Refresh Token</td>
</tr>
</tbody>
</table>

Automation Details
### On-Prem

This automated deployment is designed with a single Ansible playbook that consists of three separate roles. The roles are for ONTAP, Linux, and Oracle configurations. The following table describes which tasks are being automated.

<table>
<thead>
<tr>
<th>Playbook</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ontap_setup</strong></td>
<td>Pre-check of the ONTAP environment</td>
</tr>
<tr>
<td></td>
<td>Creation of Intercluster LIFs on source cluster (OPTIONAL)</td>
</tr>
<tr>
<td></td>
<td>Creation of Intercluster LIFs on destination cluster (OPTIONAL)</td>
</tr>
<tr>
<td></td>
<td>Creation of Cluster and SVM Peering</td>
</tr>
<tr>
<td></td>
<td>Creation of destination SnapMirror and Initialization of designated Oracle volumes</td>
</tr>
<tr>
<td><strong>ora_replication_cg</strong></td>
<td>Enable backup mode for each database in /etc/oratab</td>
</tr>
<tr>
<td></td>
<td>Snapshot taken of Oracle Binary and Database volumes</td>
</tr>
<tr>
<td></td>
<td>Snapmirror Updated</td>
</tr>
<tr>
<td></td>
<td>Turn off backup mode for each database in /etc/oratab</td>
</tr>
<tr>
<td><strong>ora_replication_log</strong></td>
<td>Switch current log for each database in /etc/oratab</td>
</tr>
<tr>
<td></td>
<td>Snapshot taken of Oracle Log volume</td>
</tr>
<tr>
<td></td>
<td>Snapmirror Updated</td>
</tr>
<tr>
<td><strong>ora_recovery</strong></td>
<td>Break SnapMirror</td>
</tr>
<tr>
<td></td>
<td>Enable NFS and create junction path for Oracle volumes on the destination</td>
</tr>
<tr>
<td></td>
<td>Configure DR Oracle Host</td>
</tr>
<tr>
<td></td>
<td>Mount and verify Oracle volumes</td>
</tr>
<tr>
<td></td>
<td>Recover and start Oracle database</td>
</tr>
</tbody>
</table>

### CVO

This automated deployment is designed with a single Ansible playbook that consists of three separate roles. The roles are for ONTAP, Linux, and Oracle configurations. The following table describes which tasks are being automated.
<table>
<thead>
<tr>
<th>Playbook</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>cvo_setup</td>
<td>Pre-check of the environment</td>
</tr>
<tr>
<td></td>
<td>AWS Configure/AWS Access Key ID/Secret Key/Default Region</td>
</tr>
<tr>
<td></td>
<td>Creation of AWS Role</td>
</tr>
<tr>
<td></td>
<td>Creation of NetApp Cloud Manager Connector instance in AWS</td>
</tr>
<tr>
<td></td>
<td>Creation of Cloud Volumes ONTAP (CVO) instance in AWS</td>
</tr>
<tr>
<td></td>
<td>Add On-Prem Source ONTAP Cluster to NetApp Cloud Manager</td>
</tr>
<tr>
<td></td>
<td>Creation of destination SnapMirror and Initialization of designated Oracle volumes</td>
</tr>
<tr>
<td>ora_replication_cg</td>
<td>Enable backup mode for each database in /etc/oratab</td>
</tr>
<tr>
<td></td>
<td>Snapshot taken of Oracle Binary and Database volumes</td>
</tr>
<tr>
<td></td>
<td>Snapmirror Updated</td>
</tr>
<tr>
<td></td>
<td>Turn off backup mode for each database in /etc/oratab</td>
</tr>
<tr>
<td>ora_replication_log</td>
<td>Switch current log for each database in /etc/oratab</td>
</tr>
<tr>
<td></td>
<td>Snapshot taken of Oracle Log volume</td>
</tr>
<tr>
<td></td>
<td>Snapmirror Updated</td>
</tr>
<tr>
<td>ora_recovery</td>
<td>Break SnapMirror</td>
</tr>
<tr>
<td></td>
<td>Enable NFS and create junction path for Oracle volumes on the destination CVO</td>
</tr>
<tr>
<td></td>
<td>Configure DR Oracle Host</td>
</tr>
<tr>
<td></td>
<td>Mount and verify Oracle volumes</td>
</tr>
<tr>
<td></td>
<td>Recover and start Oracle database</td>
</tr>
</tbody>
</table>

**Default parameters**

To simplify automation, we have preset many required Oracle parameters with default values. It is generally not necessary to change the default parameters for most deployments. A more advanced user can make changes to the default parameters with caution. The default parameters are located in each role folder under defaults directory.

**License**

You should read license information as stated in the Github repository. By accessing, downloading, installing, or using the content in this repository, you agree the terms of the license laid out here.

Note that there are certain restrictions around producing and/or sharing any derivative works with the content in this repository. Please make sure you read the terms of the License before using the content. If you do not agree to all of the terms, do not access, download, or use the content in this repository.

After you are ready, click here for detailed AWX/Tower procedures.
**Step-by-step deployment procedure**

This page describes the Automated Data Protection of Oracle 19c on NetApp ONTAP storage.

**AWX/Tower Oracle Data Protection**

Create the inventory, group, hosts, and credentials for your environment

This section describes the setup of inventory, groups, hosts, and access credentials in AWX/Ansible Tower that prepare the environment for consuming NetApp automated solutions.

1. Configure the inventory.
   a. Navigate to Resources → Inventories → Add, and click Add Inventory.
   b. Provide the name and organization details, and click Save.
   c. On the Inventories page, click the inventory created.
   d. Navigate to the Groups sub-menu and click Add.
   e. Provide the name oracle for your first group and click Save.
   f. Repeat the process for a second group called dr_oracle.
   g. Select the oracle group created, go to the Hosts sub-menu and click Add New Host.
   h. Provide the IP address of the Source Oracle host’s management IP, and click Save.
   i. This process must be repeated for the dr_oracle group and add the DR/Destination Oracle host’s management IP/hostname.

Below are instructions for creating the credential types and credentials for either On-Prem with ONTAP, or CVO on AWS.
On-Prem

1. Configure the credentials.

2. Create Credential Types. For solutions involving ONTAP, you must configure the credential type to match username and password entries.
   a. Navigate to Administration → Credential Types, and click Add.
   b. Provide the name and description.
   c. Paste the following content in Input Configuration:

   ```json
   fields:
   - id: dst_cluster_username
     type: string
     label: Destination Cluster Username
   - id: dst_cluster_password
     type: string
     label: Destination Cluster Password
     secret: true
   - id: src_cluster_username
     type: string
     label: Source Cluster Username
   - id: src_cluster_password
     type: string
     label: Source Cluster Password
     secret: true
   
   d. Paste the following content into Injector Configuration and then click Save:

   ```yaml
   extra_vars:
   dst_cluster_username: '{{ dst_cluster_username }}'
   dst_cluster_password: '{{ dst_cluster_password }}'
   src_cluster_username: '{{ src_cluster_username }}'
   src_cluster_password: '{{ src_cluster_password }}'
   
3. Create Credential for ONTAP
   a. Navigate to Resources → Credentials, and click Add.
   b. Enter the name and organization details for the ONTAP Credentials
   c. Select the credential type that was created in the previous step.
   d. Under Type Details, enter the Username and Password for your Source and Destination Clusters.
   e. Click Save

4. Create Credential for Oracle
   a. Navigate to Resources → Credentials, and click Add.
   b. Enter the name and organization details for Oracle
c. Select the Machine credential type.
d. Under Type Details, enter the Username and Password for the Oracle hosts.
e. Select the correct Privilege Escalation Method, and enter the username and password.
f. Click Save
g. Repeat process if needed for a different credential for the dr_oracle host.

**CVO**

1. Configure the credentials.

2. Create credential types. For solutions involving ONTAP, you must configure the credential type to match username and password entries, we will also add entries for Cloud Central and AWS.
   a. Navigate to Administration → Credential Types, and click Add.
   b. Provide the name and description.
   c. Paste the following content in Input Configuration:
fields:
- id: dst_cluster_username
  type: string
  label: CVO Username
- id: dst_cluster_password
  type: string
  label: CVO Password
  secret: true
- id: cvo_svm_password
  type: string
  label: CVO SVM Password
  secret: true
- id: src_cluster_username
  type: string
  label: Source Cluster Username
- id: src_cluster_password
  type: string
  label: Source Cluster Password
  secret: true
- id: regular_id
  type: string
  label: Cloud Central ID
  secret: true
- id: email_id
  type: string
  label: Cloud Manager Email
  secret: true
- id: cm_password
  type: string
  label: Cloud Manager Password
  secret: true
- id: access_key
  type: string
  label: AWS Access Key
  secret: true
- id: secret_key
  type: string
  label: AWS Secret Key
  secret: true
- id: token
  type: string
  label: Cloud Central Refresh Token
  secret: true

d. Paste the following content into Injector Configuration and click Save:
extra_vars:
  dst_cluster_username: '{{ dst_cluster_username }}'
  dst_cluster_password: '{{ dst_cluster_password }}'
  cvo_svm_password: '{{ cvo_svm_password }}'
  src_cluster_username: '{{ src_cluster_username }}'
  src_cluster_password: '{{ src_cluster_password }}'
  regular_id: '{{ regular_id }}'
  email_id: '{{ email_id }}'
  cm_password: '{{ cm_password }}'
  access_key: '{{ access_key }}'
  secret_key: '{{ secret_key }}'
  token: '{{ token }}'

3. Create Credential for ONTAP/CVO/AWS
   a. Navigate to Resources → Credentials, and click Add.
   b. Enter the name and organization details for the ONTAP Credentials
   c. Select the credential type that was created in the previous step.
   d. Under Type Details, enter the Username and Password for your Source and CVO Clusters, Cloud Central/Manager, AWS Access/Secret Key and Cloud Central Refresh Token.
   e. Click Save

4. Create Credential for Oracle (Source)
   a. Navigate to Resources → Credentials, and click Add.
   b. Enter the name and organization details for Oracle host
   c. Select the Machine credential type.
   d. Under Type Details, enter the Username and Password for the Oracle hosts.
   e. Select the correct Privilege Escalation Method, and enter the username and password.
   f. Click Save

5. Create Credential for Oracle Destination
   a. Navigate to Resources → Credentials, and click Add.
   b. Enter the name and organization details for the DR Oracle host
   c. Select the Machine credential type.
   d. Under Type Details, enter the Username (ec2-user or if you have changed it from default enter that), and the SSH Private Key
   e. Select the correct Privilege Escalation Method (sudo), and enter the username and password if needed.
   f. Click Save

Create a project

1. Go to Resources → Projects, and click Add.
a. Enter the name and organization details.
b. Select Git in the Source Control Credential Type field.
c. Enter https://github.com/NetApp-Automation/na_oracle19c_data_protection.git as the source control URL.
d. Click Save.
e. The project might need to sync occasionally when the source code changes.

Configure global variables

Variables defined in this section apply to all Oracle hosts, databases, and the ONTAP cluster.

1. Input your environment-specific parameters in following embedded global variables or vars form.

   The items in blue must be changed to match your environment.
# Oracle Data Protection global user configuration variables
# Ontap env specific config variables
hosts_group: "ontap"
cal_signed_certs: "false"

# Inter-cluster LIF details
src_nodes:
  - "AFF-01"
  - "AFF-02"
dst_nodes:
  - "DR-AFF-01"
  - "DR-AFF-02"

create_source_intercluster_lifs: "yes"

source_intercluster_network_port_details:
  using_dedicated_ports: "yes"
  using_ifgrp: "yes"
  using_vlans: "yes"
  failover_for_shared_individual_ports: "yes"
  ifgrp_name: "a0a"
  vlan_id: "10"
  ports:
    - "e0b"
    - "e0g"
  broadcast_domain: "NFS"
  ipspace: "Default"
  failover_group_name: "iclifs"

source_intercluster_lif_details:
  - name: "icl_1"
    address: "10.0.0.1"
    netmask: "255.255.255.0"
    home_port: "a0a-10"
    node: "AFF-01"
  - name: "icl_2"
    address: "10.0.0.2"
    netmask: "255.255.255.0"
    home_port: "a0a-10"
    node: "AFF-02"

create_destination_intercluster_lifs: "yes"
destination_intercluster_network_port_details:
  using_dedicated_ports: "yes"
  using_ifgrp: "yes"
  using_vlans: "yes"
  failover_for_shared_individual_ports: "yes"
  ifgrp_name: "a0a"
  vlan_id: "10"
  ports:
    - "e0b"
    - "e0g"
  broadcast_domain: "NFS"
  ipspace: "Default"
  failover_group_name: "iclifs"

destination_intercluster_lif_details:
  - name: "icl_1"
    address: "10.0.0.3"
    netmask: "255.255.255.0"
    home_port: "a0a-10"
    node: "DR-AFF-01"
  - name: "icl_2"
    address: "10.0.0.4"
    netmask: "255.255.255.0"
    home_port: "a0a-10"
    node: "DR-AFF-02"

# Variables for SnapMirror Peering
passphrase: "your-passphrase"

# Source & Destination List
dst_cluster_name: "dst-cluster-name"
dst_cluster_ip: "dst-cluster-ip"
dst_vserver: "dst-vserver"
dst_nfs_lif: "dst-nfs-lif"
src_cluster_name: "src-cluster-name"
src_cluster_ip: "src-cluster-ip"
src_vserver: "src-vserver"

# Variable for Oracle Volumes and SnapMirror Details
cg_snapshot_name_prefix: "oracle"
src_orabinary_vols:
  - "binary_vol"
src_db_vols:
  - "db_vol"
src_archivelog_vols:
  - "log_vol"
snapmirror_policy: "async_policy_oracle"

# Export Policy Details
export_policy_details:
    name: "nfs_export_policy"
    client_match: "0.0.0.0/0"
    ro_rule: "sys"
    rw_rule: "sys"

# Linux env specific config variables
mount_points:
    - "/u01"
    - "/u02"
    - "/u03"
hugepages_nr: "1234"
redhat_sub_username: "xxx"
redhat_sub_password: "xxx"

# DB env specific install and config variables
recovery_type: "scn"
control_files:
    - "/u02/oradata/CDB2/control01.ctl"
    - "/u03/orareco/CDB2/control02.ctl"

CVO

# Inventory group name
# Default inventory group name - "ontap"
# Change only if you are changing the group name either in inventory/hosts file or in inventory groups in case of AWX/Tower
hosts_group: "ontap"

# CA_signed_certificates (ONLY CHANGE to "true" IF YOU ARE USING CA SIGNED CERTIFICATES)
ca_signed_certs: "false"

# Names of the Nodes in the Source ONTAP Cluster
src_nodes:
    - "AFF-01"
    - "AFF-02"

# Names of the Nodes in the Destination CVO Cluster
dst_nodes:
- "DR-AFF-01"
- "DR-AFF-02"

#Define whether or not to create intercluster lifs on source cluster
(ONLY CHANGE to "No" IF YOU HAVE ALREADY CREATED THE INTERCLUSTER LIFS)
create_source_intercluster_lifs: "yes"

source_intercluster_network_port_details:
  using_dedicated_ports: "yes"
  using_ifgrp: "yes"
  using_vlans: "yes"
  failover_for_shared_individual_ports: "yes"
  ifgrp_name: "a0a"
  vlan_id: "10"
  ports:
    - "e0b"
    - "e0g"
  broadcast_domain: "NFS"
  ipspace: "Default"
  failover_group_name: "iclifs"

source_intercluster_lif_details:
- name: "icl_1"
  address: "10.0.0.1"
  netmask: "255.255.255.0"
  home_port: "a0a-10"
  node: "AFF-01"
- name: "icl_2"
  address: "10.0.0.2"
  netmask: "255.255.255.0"
  home_port: "a0a-10"
  node: "AFF-02"

#########################################################################
### CVO Deployment Variables ###
#########################################################################

########### Access Keys Variables ########

# Region where your CVO will be deployed.
region_deploy: "us-east-1"

########### CVO and Connector Vars ########

# AWS Managed Policy required to give permission for IAM role creation.
aws_policy: "arn:aws:iam::1234567:policy/OCCM"

# Specify your aws role name, a new role is created if one already does not exist.
aws_role_name: "arn:aws:iam::1234567:policy/OCCM"

# Name your connector.
connector_name: "awx_connector"

# Name of the key pair generated in AWS.
key_pair: "key_pair"

# Name of the Subnet that has the range of IP addresses in your VPC.
subnet: "subnet-12345"

# ID of your AWS security group that allows access to on-prem resources.
security_group: "sg-123123123"

# You Cloud Manager Account ID.
account: "account-A23123A"

# Name of your CVO instance

cvo_name: "test_cvo"

# ID of the VPC in AWS.
vpc: "vpc-123123123"

# For Federated users, Client ID from API Authentication Section of Cloud Central to generate access token.
sso_id: "123123123123123123123"

# For regular access with username and password, please specify "pass" as the connector_access. For SSO users, use "refresh_token" as the variable.
connector_access: "pass"
passphrase: "your-passphrase"

# Source & Destination List

# Please Enter Destination Cluster Name
dst_cluster_name: "dst-cluster-name"

# Please Enter Destination Cluster (Once CVO is Created Add this Variable to all templates)
dst_cluster_ip: "dst-cluster-ip"

# Please Enter Destination SVM to create mirror relationship
dst_vserver: "dst-vserver"

# Please Enter NFS Lif for dst vserver (Once CVO is Created Add this Variable to all templates)
dst_nfs_lif: "dst-nfs-lif"

# Please Enter Source Cluster Name
src_cluster_name: "src-cluster-name"

# Please Enter Source Cluster
src_cluster_ip: "src-cluster-ip"

# Please Enter Source SVM
src_vserver: "src-vserver"

# Variable for Oracle Volumes and SnapMirror Details

cg_snapshot_name_prefix: "oracle"

# Please Enter Source Oracle Binary Volume(s)
src_orabinary_vols:
  - "binary_vol"

# Please Enter Source Database Volume(s)
src_db_vols:
  - "db_vol"
src_archivelog_vols:
  - "log_vol"

# Please Enter Destination Snapmirror Policy
snapmirror_policy: "async_policy_oracle"

# Enter the destination export policy details (Once CVO is Created Add this Variable to all templates)
export_policy_details:
  name: "nfs_export_policy"
  client_match: "0.0.0.0/0"
  ro_rule: "sys"
  rw_rule: "sys"

# NFS Mount points for Oracle DB volumes
mount_points:
  - "/u01"
  - "/u02"
  - "/u03"

# Up to 75% of node memory size divided by 2mb. Consider how many databases to be hosted on the node and how much ram to be allocated to each DB.
# Leave it blank if hugepage is not configured on the host.
hugepages_nr: "1234"

# RedHat subscription username and password
redhat_sub_username: "xxx"
redhat_sub_password: "xxx"

# Recovery Type (leave as scn)
recovery_type: "scn"
# Oracle Control Files

control_files:
- "/u02/oradata/CDB2/control01.ctl"
- "/u03/orareco/CDB2/control02.ctl"

## Automation Playbooks

There are four separate playbooks that need to be ran.

1. Playbook for Setting up your environment, On-Prem or CVO.
2. Playbook for replicating Oracle Binaries and Databases on a schedule
3. Playbook for replicating Oracle Logs on a schedule
4. Playbook for Recovering your database on a destination host
ONTAP/CVO Setup
ONTAP and CVO Setup

Configure and launch the job template.

1. Create the job template.
   a. Navigate to Resources → Templates → Add and click Add Job Template.
   b. Enter the name ONTAP/CVO Setup
   c. Select the Job type; Run configures the system based on a playbook.
   d. Select the corresponding inventory, project, playbook, and credentials for the playbook.
   e. Select the ontap_setup.yml playbook for an On-Prem environment or select the cvo_setup.yml for replicating to a CVO instance.
   f. Paste global variables copied from step 4 into the Template Variables field under the YAML tab.
   g. Click Save.

2. Launch the job template.
   a. Navigate to Resources → Templates.
   b. Click the desired template and then click Launch.

   We will use this template and copy it out for the other playbooks.

Replication For Binary and Database Volumes
Scheduling the Binary and Database Replication Playbook

Configure and launch the job template.

1. Copy the previously created job template.
   a. Navigate to Resources → Templates.
   b. Find the ONTAP/CVO Setup Template, and on the far right click on Copy Template
   c. Click Edit Template on the copied template, and change the name to Binary and Database Replication Playbook.
   d. Keep the same inventory, project, credentials for the template.
   e. Select the ora_replication_cg.yml as the playbook to be executed.
   f. The variables will remain the same, but the CVO cluster IP will need to be set in the variable dst_cluster_ip.
   g. Click Save.

2. Schedule the job template.
   a. Navigate to Resources → Templates.
   b. Click the Binary and Database Replication Playbook template and then click Schedules at the top set of options.
   c. Click Add, add Name Schedule for Binary and Database Replication, choose the Start date/time at the beginning of the hour, choose your Local time zone, and Run frequency. Run frequency will be often the SnapMirror replication will be updated.
A separate schedule will be created for the Log volume replication, so that it can be replicated on a more frequent cadence.

**Replication for Log Volumes**

**Scheduling the Log Replication Playbook**

**Configure and launch the job template.**

1. Copy the previously created job template.
   a. Navigate to Resources → Templates.
   b. Find the ONTAP/CVO Setup Template, and on the far right click on Copy Template
   c. Click Edit Template on the copied template, and change the name to Log Replication Playbook.
   d. Keep the same inventory, project, credentials for the template.
   e. Select the ora_replication_logs.yml as the playbook to be executed.
   f. The variables will remain the same, but the CVO cluster IP will need to be set in the variable dst_cluster_ip.
   g. Click Save.

2. Schedule the job template.
   a. Navigate to Resources → Templates.
   b. Click the Log Replication Playbook template and then click Schedules at the top set of options.
   c. Click Add, add Name Schedule for Log Replication, choose the Start date/time at the beginning of the hour, choose your Local time zone, and Run frequency. Run frequency will be often the SnapMirror replication will be updated.

   It is recommended to set the log schedule to update every hour to ensure the recovery to the last hourly update.

**Restore and Recover Database**

**Scheduling the Log Replication Playbook**

**Configure and launch the job template.**

1. Copy the previously created job template.
   a. Navigate to Resources → Templates.
   b. Find the ONTAP/CVO Setup Template, and on the far right click on Copy Template
   c. Click Edit Template on the copied template, and change the name to Restore and Recovery Playbook.
   d. Keep the same inventory, project, credentials for the template.
   e. Select the ora_recovery.yml as the playbook to be executed.
   f. The variables will remain the same, but the CVO cluster IP will need to be set in the variable dst_cluster_ip.
   g. Click Save.
This playbook will not be ran until you are ready to restore your database at the remote site.

Recovering Oracle Database

1. On-premises production Oracle databases data volumes are protected via NetApp SnapMirror replication to either a redundant ONTAP cluster in secondary data center or Cloud Volume ONTAP in public cloud. In a fully configured disaster recovery environment, recovery compute instances in secondary data center or public cloud are standby and ready to recover the production database in the case of a disaster. The standby compute instances are kept in sync with on-prem instances by running parallel updates on OS kernel patch or upgrade in a lockstep.

2. In this solution demonstrated, Oracle binary volume is replicated to target and mounted at target instance to bring up Oracle software stack. This approach to recover Oracle has advantage over a fresh installation of Oracle at last minute when a disaster occurred. It guarantees Oracle installation is fully in sync with current on-prem production software installation and patch levels etc. However, this may or may not have additional software licensing implication for the replicated Oracle binary volume at recovery site depending on how the software licensing is structured with Oracle. User is recommended to check with its software licensing personnel to assess the potential Oracle licensing requirement before deciding to use the same approach.

3. The standby Oracle host at the destination is configured with the Oracle prerequisite configurations.

4. The SnapMirrors are broken and the volumes are made writable and mounted to the standby Oracle host.

5. The Oracle recovery module performs following tasks to recovery and startup Oracle at recovery site after all DB volumes are mounted at standby compute instance.
   a. Sync the control file: We deployed duplicate Oracle control files on different database volume to protect critical database control file. One is on the data volume and another is on log volume. Since data and log volumes are replicated at different frequency, they will be out of sync at the time of recovery.
   b. Relink Oracle binary: Since the Oracle binary is relocated to a new host, it needs a relink.
   c. Recover Oracle database: The recovery mechanism retrieves last System Change Number in last available archived log in Oracle log volume from control file and recovers Oracle database to recoup all business transactions that was able to be replicated to DR site at the time of failure. The database is then started up in a new incarnation to carry on user connections and business transaction at recovery site.

Before running the Recovering playbook make sure you have the following:
Make sure it copy over the /etc/oratab and /etc/oraInst.loc from the source Oracle host to the destination host

TR-4794: Oracle databases on NetApp EF-Series

Mitch Blackburn, Ebin Kadavy, NetApp

TR-4794 is intended to help storage administrators and database administrators successfully deploy Oracle on NetApp EF-Series storage.

TR-4794: Oracle databases on NetApp EF-Series