



Automated deployment of Oracle 19c for ONTAP on NFS

NetApp database solutions

NetApp
August 18, 2025

This PDF was generated from <https://docs.netapp.com/us-en/netapp-solutions-databases/automation/marketing-overview.html> on August 18, 2025. Always check docs.netapp.com for the latest.

Table of Contents

- Automated deployment of Oracle 19c for ONTAP on NFS 1
 - Solution Overview 1
 - Automated Deployment of Oracle19c for ONTAP on NFS. 1
 - Getting started 2
 - AWX/Tower. 2
 - CLI via the Ansible control host 2
 - Requirements. 2
 - Automation Details 3
 - Default parameters. 4
 - Deployment instructions. 4
 - License. 4
 - Step-by-step deployment procedure 4
 - AWX/Tower deployment Oracle 19c Database 4
 - Step-by-step deployment procedure 13
 - CLI deployment Oracle 19c Database 13

Automated deployment of Oracle 19c for ONTAP on NFS

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

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

Environment	Requirements
Ansible environment	AWX/Tower or Linux host to be the Ansible control host
	Ansible v.2.10 and higher
	Python 3
	Python libraries - netapp-lib - xmltodict - jmespath
ONTAP	ONTAP version 9.3 - 9.7
	Two data aggregates
	NFS vlan and ifgrp created

Environment	Requirements
Oracle server(s)	RHEL 7/8
	Oracle Linux 7/8
	Network interfaces for NFS, public, and optional mgmt
	Oracle installation files on Oracle servers

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.

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

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](#).

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 deployment procedures](#) or [here for CLI deployment](#).

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:

```
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

```
#####
##### Host Variables Configuration #####
#####

# 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 blank 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"}
```

- Fill in all variables in the blue fields.
- After completing variables input, click the Copy button on the form to copy all variables to be transferred to AWX or Tower.
- Navigate back to AWX or Tower and go to Resources → Hosts, and select and open the Oracle server configuration page.
- Under the Details tab, click edit and paste the copied variables from step 1 to the Variables field under the YAML tab.
- Click Save.
- 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.

- 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 19c deployment global user configuration variables #####
##### Consolidate all variables from ontap, linux and oracle #####
#####

#####
### Ontap env specific config variables ###
#####

#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 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 VLANsEnter 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 ###
#####

db_domain: "your.domain.com"

# 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:

```
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 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;

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:

```
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.



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.



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

Host VARS Config

```
#####
##### Host Variables Configuration #####
#####
```



```

# 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 blank 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.

```
#####  
##### Oracle 19c deployment global user configuration variables #####  
##### Consolidate all variables from ontap, linux and oracle #####  
#####  
  
#####  
### Ontap env specific config variables ###  
#####  
  
#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 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 VLANsEnter 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 alternately
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 ###
#####

db_domain: "your.domain.com"

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

initial_pwd_all: "netapp123"
```

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.

```
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:

```
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;

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.

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.

```
[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.

Copyright information

Copyright © 2025 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.