Linux
KVM

Using Deploycluster Tool with Oracle Linux Virtualization Manager – OLVM

Create Oracle Database Single Instance or Oracle Real Application Clusters using the Deploycluster tool and Database Templates for Oracle Linux KVM managed by Oracle Linux Virtualization Manager

June 2021, Version 1.1

Purpose statement

This document provides an overview of features and sample use cases for the Deploycluster tool. It is intended solely as a reference guide on using the tool along with official Oracle Database templates to rapidly create Oracle databases on the Oracle Linux Virtualization Manager (OLVM) infrastructure.

Disclaimer

Table of contents

Purpose

This document provides an overview of the features and enhancements included in Deploycluster release 4.0. It is intended solely to help assess the business benefits of using and implementing the Deploycluster tool for rapid Single Instance or n-node Oracle RAC database deployments on enterprise level infrastructure as well as test environments. Several use cases are demonstrated in this document.

History

The original Deploycluster tool was written in 2011 for Oracle VM and has served customers by rapidly provisioning repeatable and certified environments of Oracle database single instance and n-node Oracle RAC clusters on Oracle VM. Customers are able to use the Oracle VM database templates to spin up a database from the command line using a single command. Throughout the years, it has proven to be a robust tool and has been enhanced to add additional features requested from Oracle VM customers.

With Linux KVM becoming the de-facto virtualization standard used in enterprises and large cloud providers, Oracle is delighted to provide the enhanced Deploycluster V4 tool for use with database templates on Oracle Linux KVM with Oracle Linux Virtualization Manager.

Note that the Deploycluster V3 tool is used with Oracle VM version 3.4, whereas the new Deploycluster V4 tool is used exclusively in OLVM environments.

Prerequisites

It is assumed that the reader has an OLVM infrastructure set up and has imported the Oracle Database template into their environment. At least two physical compute hosts with an external OLVM manager or an embedded self-hosted engine are in place. The Deploycluster tool connects to the API server running on the OLVM host or VM. Ensure an appropriate SSL certificate is in place, and the host running the Deploycluster tool can access the OLVM server with no firewalls blocking the SSL port for a seamless connection. You should also have appropriate storage space available and two or more networks in place when building an Oracle RAC environment. An Oracle RAC build will also require LUNS (iSCSI or FC) or disk images from one or more storage domains.

Installing and configuring Deploycluster

The Deploycluster V4 tool is distributed as a compressed zip archive and can be downloaded from:

https://www.oracle.com/database/technologies/rac/vm-db-templates.html [2]

It can be installed on a VM running inside the OLVM cluster or in an external environment. Minimal resources are needed to run the **deploycluster** tool. Note that the package version number may differ from the following example, to install the tool, simply:

```
$ unzip DBRACOVM-Deploycluster4-tool.zip
```

A sample `deploycluster.ini` configuration file is provided in the zip archive. Since more than one user may use a custom `deploycluster.ini` file, this configuration is not global. It is recommended to have this configuration file in the same directory that the **deploycluster** command will be run from or in the user's home directory (e.g. `~/.deploycluster.ini`). Among other configuration options, the OLVM user password may be stored in this file, and best practices dictate to securely protect it using no more than `0600` permission. Also, it is recommended to never use the `-p <password>` option on the **deploycluster** command line. This option has only been retained for legacy purposes, instead let the password be prompted.

Please read the `deploycluster.ini` sample file as it has several options which will become more evident in the rest of this document, but at the very least, one should adjust the `DEPLOYCLUSTER_HOSTNAME`, `DEPLOYCLUSTER_USERNAME`, and `DEPLOYCLUSTER_PASSWORD` that allows for a connection to the OLVM portal. While the default username is **admin@internal**, in some customer environments, this may be a different user that has been configured to have rights to create VMs for each line of business. The username may either be an internal or an external provided username, e.g., from an Active Directory or LDAP service, with these rights. Contact your OLVM administrator for the correct values here.

Template Security

All templates use the **cloud-init** first boot service to configure the network, ssh access, and many other options. Also, **cloud-init** defines a standard security level where no user can access the instance locally or remotely except for using a ssh public key. For example, the **root** user has no password and its account is locked.

When creating a VM from the template using the OLVM portal, it is necessary to set a password or provisioning an SSH key for the **cloud-user** user via the **Cloud-Init/Sysprep** option at **Initial Run** left tab. For the Deploycluster tool, it is mandatory to provide one or more SSH public key to each cloned VM for the user named **cloud-user**. This option in the `deploycluster.ini` is called `DEPLOYCLUSTER_PUBLICKEYS` and should point to one or more public key files separated by commas. If this option is not provided in the `ini` file, the **deploycluster** command will search for a public key in encryption order of **RSA**, **DSA**, **ECDSA,** and **ED25519** in the `~/.ssh` folder. If the **deploycluster** command cannot find a ssh public key file or is unreadable, the **deploycluster** command will fail with the following error message and refuse to provision the instance:

```
ERROR: No public key to add; Terminating! as it will not be possible to access the VM.
Specify a public key using --pubkeys or set DEPLOYCLUSTER_PUBLICKEYS in deploycluster.ini
```

In general, to be compliant with Security Best Practices, companies demand users to not share generic accounts and passwords to access servers. To avoid having a shared account using password authentication, **cloud-user** is only allowed to access the server using ssh public keys to be able to unique identify the user accessing the server. As mentioned earlier, the **deploycluster** tool allows to configure as many ssh public keys as needed to be published to the provisioning instance.

As the deploycluster CLI tool was designed to run on a Linux machine, the following example shows creating a keypair on a Linux shell. Also, you can create an ssh keypair on a Mac or Windows system too and copy it to the machine running the Deploycluster tool. Press enter when prompted for a password-less key if required:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:le2I6asmfn0wDuazogIUDUa8qhTmo/qPejJUcSZ8dyM user@odb-deploycluster-console
The key's randomart image is:
+---[RSA 3072]----+
|o++              |
|.o = + E o o     |
|  o * . o + .    |
| = .     + o     |
|= o     S . .    |
|o=    o.o        |
|* .  o +.o       |
|* ..o = o..      |
|+B++o=o+..       |
+----[SHA256]-----+
```

To configure deploycluster to publish the ssh public key saved to `~/.ssh/id_rsa.pub` using the `DEPLOYCLUSTER_PUBLICKEYS` option in `deploycluster.ini` file or passing it as an argument using `--pubkeys` option to the **deploycluster** command. The private key at `~/.ssh/id_rsa` should be safely guarded as it will be used later to login to the VM as the **cloud-user** user.

On the other hand, some companies demand servers to be provisioned with service or admin accounts. In this case, the following example can be used to create the ssh keypair:

```
$ ssh-keygen -f admin
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in admin.
Your public key has been saved in admin.pub.
The key fingerprint is:
SHA256:o1Zul2O8NW+JSlCabJF24yQ8zVhzI2e1+E+Mo7OMx0s user@odb-deploycluster-console
The key's randomart image is:
+---[RSA 3072]----+
|          + =..  |
|       . * * o . |
|        O B . .  |
|       o @ . . o |
|        S .   + o|
|       = + . . + |
|      o o B.E. ..|
|     . . + B+=o  |
|          +o+o.  |
+----[SHA256]-----+
```

Both public keys can be added to the `deploycluster.ini` file as follow:

```
...
# Public key for cloud-user login
DEPLOYCLUSTER_PUBLICKEYS=~/.ssh/id_rsa.pub,~/admin.pub
```

Using Secure connections to OLVM

OLVM exposes its API using SSL by default. To successfully connect to the OLVM API, the CA certificate must be downloaded and passed to the **deploycluster** command using the `--ca <filename>` option or configure it in the `deploycluster.ini` file using the `DEPLOYCLUSER_OLVM_CA` option. Alternatively, the `--insecure` (or `-I`) option can be used to ignore the certificate verification when connecting to the OLVM API.

Using the `--ca <filename>` will enable encrypted TLS connections to the OLVM and KVM nodes. Configure the `deploycluster.ini` file to use `DEPLOYCLUSER_OLVM_INSECURE` or `DEPLOYCLUSER_OLVM_CA` options or use the command line options.

OLVM nomenclature

Basic Oracle Linux Virtualization Manager (OLVM) terms:

- Data Center: high-level logical entity for all physical and logical resources in an OLVM environment. A single OLVM host can manage multiple Data Centers. Each Data Center must have at least one Data Storage domain provisioned, where the VM's disks are created, on either an **NFS**, **iSCSI**, **FC**, **POSIX Complaint Filesystem** or **Local** storage provider. This storage is shared between multiple hosts in a container called Data Center.
  External storage can provide direct iSCSI, or FC LUNs to be attached to an instance. A direct LUN doesn't belong to any Storage Domain.
- Cluster: an association of physical hosts comprising of a migration domain and having compatible processors. Every cluster belongs to a Data Center and every host belongs to a Cluster. A Cluster must have a minimum of one host, and at least one active host is required to connect the system to a storage domain.

As a single OLVM instance can manage multiple Data Centers and Clusters, the **deploycluster** command needs to know where it will run cloning and where to deploy the Single Instance or RAC database. The two relevant variables are `DEPLOYCLUSTER_DATACENTER` and `DEPLOYCLUSTER_CLUSTER` that can be set in the `deploycluster.ini`. The default for both options is called **Default** and does not need to be set or changed in most circumstances. Also, it can be specified at the command line while running the **deploycluster** command by adding the options `--datacenter <name>` and/or `--cluster <name>`.

Listing VM's in the environment

Once the `deploycluster.ini` file is configured, VMs in an OLVM Cluster may be listed using the `-L` option as noted below. The list of machines will be separated with a horizontal dashed line making it simple to parse via scripts and narrow in on a specific VM. We will show later how to restrict this list to a specific set of virtual machines matched by wildcards. For now, let us just run without any additional options:

```
$ deploycluster -L
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 13:07:28 -03 2021 by user user
Using: deploycluster -L
[LIST MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
INFO: Listing completed in 628ms
INFO: Logfile at: 'deploycluster.log'
```

This is a fresh environment with no VMs provisioned. The Data Center and Cluster we are operating on are shown along with the total runtime of the command. All options used on the previous command output came from the `~/.deploycluster.ini` file, such as OLVM FQDN, Data Center and Cluster to connect to, reducing chance for typos or errors. It is also possible to pass all options on the command line, and it would look as follows:

```
$ deploycluster -H olvm-manager.example.com -u admin@internal -p <userpassword> --ca ~/ca-
olvm-manager.example.com.pem -L
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 13:10:38 -03 2021 by user user
Using: deploycluster -H olvm-manager.example.com -u admin@internal -p ****** --ca
/home/user/ca-olvm-manager.example.com.pem -L
```

```
[LIST MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
INFO: Listing completed in 579ms
INFO: Logfile at: 'deploycluster1.log'
```

If any errors appear at this stage, check/verify the options passed on command line or configured in the `deploycluster.ini` file before trying again.

Logging

All invocations of the **deploycluster** command are logged by default in the same directory where the **deploycluster** command runs from. This can be changed with the `DEPLOYCLUSTER_AUTOLOG` option to enable or disable logging and the `DEPLOYCLUSTER_AUTOLOG_DIR` command to specify where to write out the log files. These options are useful for auditing as the tool will print out the full command line options used for each invocation, date, user that ran the command (note elevated privileges are not required), the deploycluster version and the full command output. An example is shown below, and the output file will be called `deployclusterNN.log` with numerically increasing `NN` value.

```
$ deploycluster -L
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 13:07:28 -03 2021 by user user
Using: deploycluster -L
[LIST MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
INFO: Listing completed in 628ms
INFO: Logfile at: 'deploycluster.log'
```

Listing Templates

Templates are essentially golden images that are signed and optionally sealed that have all the necessary information from which a VM instances can be created easily and repeatably. The Oracle Database templates have been created for a variety of database versions and include, appropriate database version and Patch Set Updates (PSU), with a tested OS patch set and packages, and the certified Oracle Linux Unbreakable Enterprise Kernel (UEK). Please refer to the virtualization matrix

[4] for the latest list of certified database versions supported with Oracle Linux KVM and Oracle Linux UEK.

As described in the previous prerequisite section, the appropriate Oracle Linux KVM database template must be imported in advance via the OLVM portal. To list the already imported and available templates, use the `-LT` option with the **deploycluster** command. For example:

```
$ deploycluster -LT
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 13:13:08 -03 2021 by user user
Using: deploycluster -LT
[LIST MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...

 ------------------------------------------------------------------------------------------
  Template: Blank Ver: 0                                   [00000000-0000-0000-0000-0000...
  Status: ok   Memory: 1.00G [1073741824]   Vcpus: 1
  Description: Blank template
  Comment:
  Nics:
  Disks:
-------------------------------------------------------------------------------------------
  Template: OLVM-OL8U4-19110DBRAC-KVM Ver: 1               [0014cda4-0de3-49b7-8bd2-689f...
  Status: ok   Memory: 4.00G [4294967296]   Vcpus: 2
  Description:
  Comment:
  Nics:
   - nic1 [mac not assigned] (ovirtmgmt)
   - nic2 [mac not assigned] (ovirtmgmt)
  Disks:
   - Disk_OL8U4_x86_64-olvm-b85 (3.50G/37.00G, image)      [981b0da8-bfa4-429e-9df7-1494...
   - OL8U4Oracle19110DBRAC_x86_64 (26.38G/60.00G, image)   [f01b2f54-f245-47e5-8603-bde8...
 ------------------------------------------------------------------------------------------

INFO: Listing completed in 462ms
INFO: Logfile at: 'deploycluster2.log'
```

Ignoring the Blank template, the previous output shows the **OLVM-OL8U4-19110DBRAC-KVM** template with 2 disk attachments, 2 NICs, 4GB of memory, and 2 vCPUs. This template has no description or comments. The UUID may be used to uniquely select the template to clone from. While there could be multiple templates with an identical name that point to multiple different template versions, referring to a template's UUID ensures that the instance is created using the appropriate template version.

Please note that the templates ship with both NICs on the same network – named ovirtmgmt. Therefore, while is possible to successfully create a RAC or a Single Instance (SI) having both NICs on the same network, it is recommended to configure the second NIC on a private network.

Listing Available Storage

For a Single Instance database, typically no additional storage is required other than the disks shipped with the template and listed in the previous command output. When creating an n-node RAC it is required to create/attach ASM shared disks to host the databases. Listing disks can be performed using the `-LD` option, as follow:

```
$ deploycluster -LD
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 13:18:18 -03 2021 by user user
Using: deploycluster -LD
[LIST MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...

 ---------------------------------------------------------------------------------------------
  Storage Domain: iSCSI
 ---------------------------------------------------------------------------------------------
   - Disk_OL7U9_x86_64-olvm-b77              3.88G/37.00G       image              [a46f...
   - OL7U9Oracle19100DBRAC_x86_64-xvdb-new8  26.12G/60.00G      image              [f3e6...
   - OVF_STORE                               128.00M/128.00M    image, shareable   [4a9a...
   - OVF_STORE                               128.00M/128.00M    image, shareable   [1c76...
   - asm0                                    5.00G/5.00G        image, shareable   [89e4...
   - asm1                                    5.00G/5.00G        image, shareable   [4bbd...
   - asm2                                    5.00G/5.00G        image, shareable   [7c35...
---------------------------------------------------------------------------------------------
  Storage Domain: NFS
 ---------------------------------------------------------------------------------------------
   - OVF_STORE                               128.00M/128.00M    image, shareable   [83e8...
   - OVF_STORE                               128.00M/128.00M    image, shareable   [c3f3...
   - asm3                                    1.20G/5.00G        image, shareable   [ef57...
   - asm4                                    1.22G/5.00G        image, shareable   [a503...
 ---------------------------------------------------------------------------------------------
  Direct LUN
 ---------------------------------------------------------------------------------------------
```

```
   - asm-lun3                                   5.00G/5.00G       lun, shareable     [1f4b...
   - asm-lun4                                   5.00G/5.00G       lun, shareable     [49c9...
 ------------------------------------------------------------------------------------------
INFO: Listing completed in 469ms
INFO: Logfile at: 'deploycluster3.log'
```

The previous output show two storage domains aptly named **iSCSI** and **NFS** (the OVF_STORE disks show that these are shared between cluster hosts and are used for the VM disk images). Also, there are 7 pre-created shared disks named **asm**. Note that 5 are disk images and 2 are iSCSI Direct LUNs. Direct LUNs do not belong to any Storage Domain.

Finally, notice the template disks match with the above output from earlier `-LT` run.

When VMs are cloned from a template they will be created in the same storage domain as that of the template itself. The disk images "Disk_OL7U9_x86_64-olvm-b77" and "OL7U9Oracle19100DBRAC_x86_64-xvdb-new8" listed in the previous output were provisioned on the iSCSI storage domain. The VM disks may be live migrated to another storage domain if required at a later stage.

Before you start

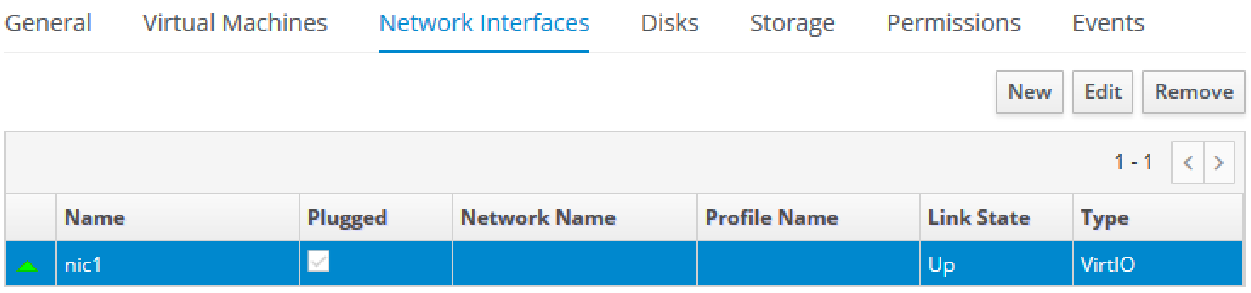
Before provisioning any instance, cloning any template to a new VM, verify which networks the template network interfaces are assigned to. The following example list the details of a recent imported template for an Oracle Linux 7.9 minimum installation. Take note of the **Nics** section:

```
...
  Template: OL7U9_x86_64-olvm-b86 Ver: 1                  [e06e6831-2174-481f-bd6c-9a13...
  Status: ok   Memory: 8.00G [8589934592]   Vcpus: 4
  Description: Generated by oracle-linux-image-tools
  Comment:
  Nics:
   - nic1 [mac not assigned] (empty)
  Disks:
   - Disk_OL7U9_x86_64-olvm-b86 (1.50G/37.00G, image)      [70f18d05-578b-45c5-9c66-fbe3...
...
```

The previous output shows a base template with an improper network assignment (mapped to "empty" network), as a result any VM cloned from it will not have any functioning networking!

It is recommended to edit the template in the OLVM portal and assign NICs to the correct network for that environment/cluster/datacenter for successful future deployments.

To do this, access the OLVM portal, **Compute** tab, **Templates** option, and click the name of the template to change. Click on the **Network Interfaces** tab, select the NIC to change, then click the **Edit** button.



	Name	Plugged	Network Name	Profile Name	Link State	Type
▲	nic1	✓			Up	VirtIO

On the next dialog box, at Profile drop-down list, select the appropriate default network to use. Remember to always set the correct NIC/assignment after a template is imported and before cloning to a new instance or use the `--clonenet` flag to instruct the Deploycluster tool to adjust this mapping during deployment. Listing of templates to confirm the Deploycluster tool shows the change:

```
...
  Template: OL7U9_x86_64-olvm-b86 Ver: 1               [e06e6831-2174-481f-bd6c-9a13...
  Status: ok   Memory: 8.00G [8589934592]   Vcpus: 4
  Description: Generated by oracle-linux-image-tools
  Comment:
  Nics:
   - nic1 [mac not assigned] (ovirtmgmt)
  Disks:
   - Disk_OL7U9_x86_64-olvm-b86 (1.50G/37.00G, image)      [70f18d05-578b-45c5-9c66-fbe3...
...
```

Cloning from the Oracle Database template

Provisioning a new instance is a process of cloning the template to a new VM, a full clone may be a heavyweight process and as such the default clone type performed by the Deploycluster tool is a thin clone. Full cloning means that all the disks from the template are copied to the storage domain in full for the new VM. Once the clone is completed there is no association with the template and the template can be deleted if required. For testing purposes, it is acceptable to use the `--clonetype thin` option (or omit it as it is the default) which will create a copy-on-write thin clone. Cloning will appear exceptionally fast; however, the resulting clone is dependent on the source disks and template and may suffer performance issues especially under heavy write activity. For production usage, always use full (not thin) clone type: –clonetype full.

The following example is the simplest form of a **deploycluster** run to clone the database template to an instance called **odb-si**:

```
$ deploycluster -C OLVM-OL8U4-19110DBRAC* -M odb-si
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 13:38:07 -03 2021 by user user
Using: deploycluster -C OLVM-OL8U4-19110DBRAC* -M odb-si
[CLONE MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
INFO: Setting clone storage domain to 'iSCSI'
INFO: Thin cloning 1 VM's from template 'OLVM-OL8U4-19110DBRAC-KVM' [0014cda4-0de3-49b7-
8bd2-689f5a206057]
INFO: Cloning VM 'odb-si' from template
INFO: Waiting 20m0s for VM 'odb-si' [82c51bbd-5847-4b7b-97c4-2ce47f454efc] to complete
INFO: Clone 'odb-si' complete
INFO: Cloning completed in 11.901s
INFO: Logfile at: 'deploycluster7.log'
```

NOTE

- This new VM instance will clone from the base template and remain in down state until a subsequent deploying operation. Do not try to start it using the OLVM portal. Alternatively, the `--clonedeploy` flag may be added to above command to also POWER on and initiate the database build.
- If any error appears while running the previous command and you like to reprovision the instance using the same name, it is safe to remove the target VM using the OLVM portal while the instance is down or it is possible to run the following command to remove it:

  ```
  $ deploycluster --remove -M odb-si
  ```

  The previous command will appear to "stall" (sleep) due to a safeguard to prevent accidental removal of VMs, add the –wait 0 to bypass the wait/delay. Note this is a synchronous command and may take longer for the OLVM cluster to remove the instance.

To provision a new instance, the template NAME or UUID may be used. It is possible to use a wildcard match on the template NAME or UUID for easy writing commands. As long as the NAME or UUID is unique, the command should complete successfully. Avoid using too short wildcards, such as '0014*', as these short wildcards have a higher chance of matching multiple templates which would cause a runtime failure as the tool will not know which template to clone from.

Since the previous clone (`-C` OLVM-OL8U4-19110DBRAC`*`) command did not change anything from the original template, all configuration and description details were copied from the template, except for the newly added **Comment** field inserted for convenience to identify clones created by the **deploycluster** command and by which version. The comment can be overwritten at clone time by adding the `--comment` option. The comment field may be used, for example, to identify which ssh key the VM has by using `--comment "<ssh-key-name>"` or for any other purpose.

List the VM provisioned using the following command:

```
$ deploycluster -L
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 13:42:05 -03 2021 by user user
Using: deploycluster -L
[LIST MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...

 ------------------------------------------------------------------------------------------
  Vm name: odb-si                                         [82c51bbd-5847-4b7b-97c4-2ce47...
  Status: down   Memory: 4.00G [4294967296]   Vcpus: 2
  Description
  Comment: created by deploycluster v4.0.0
  Nics:
   - nic1 [56:6f:f8:bf:00:00] (ovirtmgmt)
   - nic2 [56:6f:f8:bf:00:01] (ovirtmgmt)
  Disks:
   - Disk_OL8U4_x86_64-olvm-b85 (1.00G/37.00G, image)     [4c232c0c-b6ad-4c74-b63e-c7585...
   - OL8U4Oracle19110DBRAC_x86_64 (1.00G/60.00G, image)   [d8198b3b-6c8c-483d-8e10-7ecf1...
 ------------------------------------------------------------------------------------------

INFO: Listing completed in 518ms
INFO: Logfile at: 'deploycluster8.log'
```

Other valuable options to modify the clone would be `--clonecpu`, `--clonemem`, and `--clonenet`, which change the target clone's CPU, memory, and NICs appropriately. Adjust these options per database requirements. The descriptions of all the options can be found using the `deploycluster --help` command.

These settings may also be changed using the OLVM portal after provisioning.

Deploying the database

Unless the `--clonedeploy` flag is used to fully power on the newly cloned VM, all new instances cloned by **deploycluster** remain offline until a subsequent deploy operation done by **deploycluster** command which may be used to further configure the network interfaces, SID name, character set, etc. These options are partially specified on the command line as well as `params.ini` and `netconfig.ini` files. See the following example for a Single Instance deployment:

```
$ cat ~/deploycluster/utils/netconfig-sample-si.ini
# Sample Single Instance or Single Instance/HA (Oracle Restart)
NODE1=test1
NODE1IP=192.168.1.101
#NODE1PRIV=test1-priv
#NODE1PRIVIP=10.10.10.101

# Common data
PUBADAP=eth0
PUBMASK=255.255.255.0
PUBGW=192.168.1.1
#PRIVADAP=eth1
#PRIVMASK=255.255.255.0
DOMAINNAME=localdomain  # May be blank
DNSIP=""  # Starting from 2013 Templates allows multi value

# Single Instance (description in params.ini)
CLONE_SINGLEINSTANCE=yes  # Setup Single Instance
# CLONE_SINGLEINSTANCE_HA=yes  # Setup Single Instance/HA (Oracle Restart)
```

The default network setup for a **cloud-init** based VM is DHCP configured network, however, in real-life environments the database may require a static IP. In this case, it is recommended to provide a `netconfig.ini` to the **deploycluster** command with basic name/IP information. Sample for the `netconfig.ini` and `params.ini` files can be found in the utils subdirectory of the deploycluster zip archive. Please read the commented sample files for more detailed information.

This network configuration file can be passed to a single instance clone during the deployment phase and properly set up the network during power on.

It is recommended to use the dry-run mode by adding the `-D` option to ensure the validity of the configuration and the options passed prior to real deployment. The next example shows how to manually pass the previously created ssh public key **admin.pub** into the **odb-si** VM. Also, note the `NODE1` name provided in the `netconfig-si.ini` file – **oracle-db-si** – is the node name inside the guest OS and does not need to match the VM name in the OLVM portal – **odb-si**. Typically, the guest hostname and the VM name in the portal are the same, but it is kept distinct in this example to demonstrate they need not be identical. The `params.ini` file is also optional and is used to change any database defaults (charset, SID , etc.). To specify the `params.ini` file use the `-P` option. Please refer to the sample `params.ini` file included with the zip archive for more information.

```
$ deploycluster -M odb-si -N netconfig-si.ini --pubkeys admin.pub -D
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 13:49:35 -03 2021 by user user
Using: deploycluster -M odb-si -N netconfig-si.ini --pubkeys admin.pub -D
[DRY RUN MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
INFO: Validating netconfig-si.ini
INFO: Adding admin.pub as public key for cloud-user
INFO: 1 machine(s) to be started
INFO: Generating 2048 bit rsa ssh key pairs for root user on odb-si
INFO: Passing 'netconfig-si.ini' to buildcluster.sh
INFO: Deployment completed in 618ms
INFO: Logfile at: 'deploycluster9.log'
```

The previous output shows the dry run was completed successfully. The database build can be started running the same command but removing the `-D` option. The full output for the real provisioning command is a little different from the previous dry run mode:

```
$ deploycluster -M odb-si -N netconfig-si.ini --pubkeys admin.pub
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 13:50:20 -03 2021 by user user
Using: deploycluster -M odb-si -N netconfig-si.ini --pubkeys admin.pub
[DEPLOY MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
INFO: Validating netconfig-si.ini
INFO: Adding admin.pub as public key for cloud-user
INFO: 1 machine(s) to be started
INFO: Generating 2048 bit rsa ssh key pairs for root user on odb-si
INFO: Passing 'netconfig-si.ini' to buildcluster.sh
INFO: Starting 'odb-si' [82c51bbd-5847-4b7b-97c4-2ce47f454efc]
INFO: Buildcluster will run on node 'odb-si'; to access 'ssh -i admin cloud-
user@192.168.0.21'
INFO: Deployment completed in 1.436s
INFO: Logfile at: 'deploycluster10.log'
```

By default, all commands are passed asynchronously to the OLVM API, so the **deploycluster** command does not wait for the VM to boot up. However, in synchronous automated pipelines, it is essential to wait for commands to finish before commencing the next step. For this purpose, add the `--wait 50m` option to wait up to 50 for the VM to get to started state as reported by the API. Adjust the time as need; default wait time is 20 minutes, which is typically more than enough for most deployments.

```
$ deploycluster -M odb-si -N netconfig-si.ini --pubkeys myvmkey.pub --wait 50m
```

The SI database build can be monitored by accessing the VM using the **admin** ssh key and inspecting the `buildsingle.log` file. Here is an example and the output will vary depending on the build stage:

```
$ ssh -i admin cloud-user@10.169.239.169 "tail -f /u01/racovm/buildsingle.log"
tmpfs           1.8G     0  1.8G   0% /dev/shm
2021-06-27 16:51:34:[adjustmemlocal:Done :oracle-db-si] Adjusting memory settings
2021-06-27 16:51:34:[adjustmemlocal:Time :oracle-db-si] Completed successfully in 0 seconds
(0h:00m:00s)

INFO (node:oracle-db-si): Creating database (ORCL) (Single Instance)

INFO (node:oracle-db-si): Running on: oracle-db-si as oracle: export
ORACLE_HOME=/u01/app/oracle/product/19c/dbhome_1; export ORACLE_BASE=/u01/app/oracle;
/u01/app/oracle/product/19c/dbhome_1/bin/dbca -silent -createDatabase  -emConfiguration NONE
-templateName 'General_Purpose.dbc' -storageType FS -datafileDestination
'/u01/app/oracle/oradata' -datafileJarLocation
'/u01/app/oracle/product/19c/dbhome_1/assistants/dbca/templates' -sampleSchema false -
oratabLocation /etc/oratab  -runCVUChecks false -continueOnNonFatalErrors true -
createAsContainerDatabase true -numberOfPDBs 1 -pdbName orclpdb -gdbName 'ORCL' -sid 'ORCL'
-initParams filesystemio_options=setall -ignorePrereqs
Prepare for db operation
8% complete
Copying database files
^C
```

This session can be kept open to track the database build or it can be rechecked later to see its status and confirm that the Single Instance deployment has been completed successfully. Of course, the time to complete this deployment is entirely dependent on the resources (CPU, storage, NICs, etc.) performance.

NOTE

- If the previous output presents any error, connect to the instance and check `/u01/racovm/buildsingle.log` for Oracle Database instance errors or `/var/log/cloud-init-output.log` for any other configuration issue.

Multi-node RAC build

For those new to building a RAC instance, this deployment mode requires shared volumes/disks used for ASM.

The next example creates a 2 node RAC, and can be expanded to an n-node configuration by specifying as many additional VM's as the infrastructure can handle. Splitting the command into lines for easy reading:

```
$ deploycluster -C OLVM-OL8U4-19110DBRAC* \
  -M rac0,rac1 \
  --clonetype thin \
  --cloneattach asm[0-4] \
  --clonemem 6G \
  --comment "First RAC deployment"
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 14:27:30 -03 2021 by user user
Using: deploycluster -C OLVM-OL8U4-19110DBRAC* -M rac0,rac1 --clonetype thin --cloneattach
asm[0-4] --clonemem 6G –comment First RAC deployment
[CLONE MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
INFO: Setting clone storage domain to 'iSCSI'
INFO: Attachment found 'asm3' [8640af2a-ed25-44f7-8533-80a0ef58be67]
INFO: Attachment found 'asm1' [d44f92ce-6f73-4c4a-b8e1-d35d600dcf23]
INFO: Attachment found 'asm4' [a0535c17-d8b0-40cb-8c30-e96c0c6d7739]
INFO: Attachment found 'asm2' [646b3eb6-bc1f-4986-a92c-d442f6ee08c7]
INFO: Attachment found 'asm0' [aa4aa126-7d33-4342-b2c9-743933759e00]
INFO: Thin cloning 2 VM's from template 'OLVM-OL8U4-19110DBRAC-KVM' [0014cda4-0de3-49b7-
8bd2-689f5a206057]
INFO: Setting memory to 6.00G
INFO: Cloning VM 'rac0' from template
INFO: Waiting 20m0s for VM 'rac0' [1f4a1471-160f-4858-acf9-c6760b830ae5] to complete
INFO: Cloning VM 'rac1' from template
INFO: Waiting 20m0s for VM 'rac1' [40c5ccb0-65df-4caa-94e7-82db6447eaeb] to complete
INFO: Clone 'rac0' complete
INFO: Attaching disk 'asm3' [8640af2a-ed25-44f7-8533-80a0ef58be67] to VM 'rac0'
INFO: Attaching disk 'asm1' [d44f92ce-6f73-4c4a-b8e1-d35d600dcf23] to VM 'rac0'
INFO: Attaching disk 'asm4' [a0535c17-d8b0-40cb-8c30-e96c0c6d7739] to VM 'rac0'
```

```
INFO: Attaching disk 'asm2' [646b3eb6-bc1f-4986-a92c-d442f6ee08c7] to VM 'rac0'
INFO: Attaching disk 'asm0' [aa4aa126-7d33-4342-b2c9-743933759e00] to VM 'rac0'
INFO: Clone 'rac1' complete
INFO: Attaching disk 'asm3' [8640af2a-ed25-44f7-8533-80a0ef58be67] to VM 'rac1'
INFO: Attaching disk 'asm1' [d44f92ce-6f73-4c4a-b8e1-d35d600dcf23] to VM 'rac1'
INFO: Attaching disk 'asm4' [a0535c17-d8b0-40cb-8c30-e96c0c6d7739] to VM 'rac1'
INFO: Attaching disk 'asm2' [646b3eb6-bc1f-4986-a92c-d442f6ee08c7] to VM 'rac1'
INFO: Attaching disk 'asm0' [aa4aa126-7d33-4342-b2c9-743933759e00] to VM 'rac1'
INFO: Cloning completed in 22.83s
INFO: Logfile at: 'deploycluster18.log'
```

The options from the previous command are:

- `-C OLVM-OL8U4-19110DBRAC*`: specifying the base template used to provision the instances, avoid short wildcards as "OLVM*KVM" as they may match more than one template and hence halt future deployments.
- `-M rac0,rac1`: list of VMs instance names to be provisioned. Such expansions as rac[0-3] to create 4 VMs is not allowed, instead use the `namefmt` option (see help for details/examples).
- `--clonetype thin`: using thin provisioning is feasible for test purposes but not recommended for production servers.
- `--cloneattach asm[0-4]`: Attach pre-existing listed disks to the cloning instances. For a RAC deployment, ASM shared disks are required. This option accepts a list like `--cloneattach asm0,asm1,asm2,asm3,asm4` or basic regular expressions as in the command.
- `--clonemem 6G`: define the new VM's memory to 6GB each.
- `--comment "…"`: desired comment to add to instances.

Unless the `--clonedeploy` option is passed, the newly cloned instances are kept in the down state and may be inspected using the `deploycluster -L` command. To filter the listing, add the `-M rac?` option which will match any VM name starting with **rac,** followed by one character. In this example **rac0** and **rac1** will be listed.

Starting the RAC cluster

To successfully build a RAC cluster, the `netconfig.ini` file and optionally `params.ini` need to be prepared. While the params and netconfig files can be merged into a single file, from a separation of duties perspective the `netconfig.ini` file contains the network configuration for one or more nodes and should be unique as it will vary with each build, whereas the optional `params.ini` file should only have database specific key/value pairs.

Please read the options for a RAC configuration in the sample `netconfig-sample64.ini` file included in the deploycluster zip archive. The following `netconfig.ini` example show the node names as well as IPs and other relevant networking details:

```
$ cat netconfig-rac.ini
# Node specific information
NODE1=odb-rac0
NODE1IP=192.168.0.31
NODE1PRIV=odb-rac0-priv
NODE1PRIVIP=192.168.0.33
NODE1VIP=odb-rac0-vip
NODE1VIPIP=192.168.0.35

NODE2=odb-rac1
NODE2IP=192.168.0.32
NODE2PRIV=odb-rac1-priv
NODE2PRIVIP=192.168.0.34
NODE2VIP=odb-rac1-vip
NODE2VIPIP=192.168.0.36

# Common data
PUBADAP=eth0
PUBMASK=255.255.255.0
PUBGW=192.168.0.1
PRIVADAP=eth1
PRIVMASK=255.255.255.0

RACCLUSTERNAME=crs64bitR2
DOMAINNAME=localdomain  # May be blank
DNSIP="192.168.0.110"  # Starting from 2013 Templates allows multi value
# Device used to transfer network information to second node
# in interview mode
NETCONFIG_DEV=/dev/xvdc
# RAC specific data
SCANNAME=odb-rac01-scan
SCANIP=192.168.0.30
# 12c Flex parameters (uncomment to take effect)
#FLEX_CLUSTER=yes  # If 'yes' implies Flex ASM as well
#FLEX_ASM=yes
#ASMADAP=eth2  # Must be different than private/public
#ASMMASK=255.255.0.0
#NODE1ASMIP=10.11.0.231
#NODE2ASMIP=10.11.0.232
```

```
# Single Instance (description in params.ini)
# CLONE_SINGLEINSTANCE=yes  # Setup Single Instance
# CLONE_SINGLEINSTANCE_HA=yes  # Setup Single Instance/HA (Oracle Restart)
```

NOTE

- To successfully deploy a RAC environment, DNS records for **hostnames**, **scanname**, and **IPs** described in the `netconfig.ini` file are required. In test environments to avoid the dependency on DNS servers to add those records, the option `CLONE_ALLOW_SCAN_NOT_IN_DNS` can be added in the `netconfig.ini` file to dynamically include those records in the `/etc/hosts` file. Example adding this option to the end of the `netconfig-rac.ini` file:

  ```
  ... (previous output omitted)

  # Add SCANNAME to /etc/hosts file
  CLONE_ALLOW_SCAN_NOT_IN_DNS=yes
  ```

- The resulting instance will have the following records in the`/etc/hosts` file (should only be used in testing, not production environments):

  ```
  ... (previous output omitted)

  10.169.238.201 odb-rac0.localdomain odb-rac0
  10.169.238.207 odb-rac0-priv.localdomain odb-rac0-priv
  10.169.238.203 odb-rac0-vip.localdomain odb-rac0-vip

  10.169.238.202 odb-rac1.localdomain odb-rac1
  10.169.238.208 odb-rac1-priv.localdomain odb-rac1-priv
  10.169.238.204 odb-rac1-vip.localdomain odb-rac1-vip

  10.169.238.205 odb-rac01-scan.localdomain odb-rac01-scan
  ```

- It is recommended to have the private IP address in a different subnet.

As in previous single instance example, the command line is expanded to include the additional VM names using wildcard matching. Running this will send the command to the OLVM and exit in a very short time when the `--waitforip` option is not in place.

```
$ deploycluster -M rac? -N netconfig-rac.ini --pubkeys admin.pub
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 14:33:14 -03 2021 by user user
Using: deploycluster -M rac? -N netconfig-rac.ini --pubkeys admin.pub
[DEPLOY MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
INFO: Validating netconfig-rac.ini
WARNING: Multiple nics on VM 'rac0' connect to the same network 'ovirtmgmt'
WARNING: Multiple nics on VM 'rac1' connect to the same network 'ovirtmgmt'
INFO: Shared disk 'asm4' [a0535c17-d8b0-40cb-8c30-e96c0c6d7739] found
INFO: Shared disk 'asm0' [aa4aa126-7d33-4342-b2c9-743933759e00] found
INFO: Shared disk 'asm1' [d44f92ce-6f73-4c4a-b8e1-d35d600dcf23] found
INFO: Shared disk 'asm2' [646b3eb6-bc1f-4986-a92c-d442f6ee08c7] found
INFO: Shared disk 'asm3' [8640af2a-ed25-44f7-8533-80a0ef58be67] found
INFO: Total 5 shared disks between VM's
INFO: Adding admin.pub as public key for cloud-user
INFO: 2 machine(s) to be started
INFO: Generating 2048 bit rsa ssh key pairs for root user on rac0
INFO: Generating 2048 bit rsa ssh key pairs for root user on rac1
INFO: Passing 'netconfig-rac.ini' to buildcluster.sh
INFO: Starting 'rac1' [40c5ccb0-65df-4caa-94e7-82db6447eaeb]
INFO: Starting 'rac0' [1f4a1471-160f-4858-acf9-c6760b830ae5]
INFO: Buildcluster will run on node 'rac0'; to access 'ssh -i admin cloud-user@192.168.0.31'
INFO: Deployment completed in 1.292s
INFO: Logfile at: 'deploycluster19.log'
```

Notice the third line from the end specifies in which node the **buildcluster.sh** script will run on. This will typically be the first node in the `netconfig.ini`. The `--wait` option can be used to increase wait time on API operations if facing errors during deployments. To monitor the RAC build, ssh to **rac0** and tail the mentioned log file. In a RAC build, the log file is called `buildcluster.log` instead of `buildsingle.log`, as shown previously for the single instance deployment. The following output is a sample progress from the start of a buildcluster.log:

```
$ ssh -i admin cloud-user@10.169.238.201 "tail -f /u01/racovm/buildcluster.log"
The authenticity of host '192.168.0.31 (192.168.0.31)' can't be established.
```

```
ECDSA key fingerprint is SHA256:/1ZNDTrCEOp1CqmeGTtStP4NHHm+0XNOOqE8jDdM4Vs.
ECDSA key fingerprint is MD5:54:ec:96:f8:6f:3e:d6:3d:98:1e:b2:c8:be:d1:1c:19.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.31' (ECDSA) to the list of known hosts.
.
2021-06-27 17:35:23:[diskconfig:Disks:odb-rac0] Verifying disks exist, are free and with no
overlapping partitions (localhost)...
2021-06-27 17:35:23:[diskconfig:DisksExist:odb-rac0] Verify all disks exist (localhost)...
/dev/sda./dev/sdb./dev/sdc./dev/sdd./dev/sde...................OK
2021-06-27 17:35:24:[diskconfig:Disks:odb-rac0] Checking contents of disks (localhost)...
/dev/sda1/dev/sdb1/dev/sdc1/dev/sdd1/dev/sde1.
2021-06-27 17:35:24:[diskconfig:Remote:odb-rac0] Discovering disk names on remote nodes with
stamping (existence check)...
/dev/sda./dev/sdb./dev/sdc./dev/sdd./dev/sde......OK
2021-06-27 17:35:28:[diskconfig:Remote:odb-rac0] Verify disks are free on remote nodes...
odb-rac1................OK
2021-06-27 17:35:40:[diskconfig:Disks:odb-rac0] Checking contents of disks (remote nodes)...
odb-rac1.....OK
2021-06-27 17:35:41:[diskconfig:Disks:odb-rac0] Setting disk permissions for next startup
(all nodes)........OK
2021-06-27 17:35:42:[diskconfig:ClearPartTables:odb-rac0] Clearing partition tables...
./dev/sda./dev/sdb./dev/sdc./dev/sdd./dev/sde...................OK
^C
```

Putting it together

The options seen so far may be merged into a single command. For example:

```
$ deploycluster \
   -C OLVM-OL8U4-19110DBRAC* \
   -M rac0,rac1 \
   --cloneattach asm[0-2],asm-lun[34] \
   --clonemem 8G \
   --pubkeys admin \
   --comment "demo rac cluster provisioned with ssh key admin ssh pubkey" \
   --clonedeploy \
   --wait 25m \
   --waitforip \
   -N netconfig-rac.ini
```

This command will provision 2 VM RAC, named **rac0** and **rac1** from the base template that matches the name OLVM-OL8U4-19110DBRAC*, attach 5 pre-existing shared disks named `asm0,asm1,asm2,asm-lun3,asm-lun4,` increase the memory to 8G, a comment to easily identify which ssh key were provisioned to instances, wait 25 minutes until one or more IP addresses are reported to be assigned to the VM's. Network setup will be taken from the provided `netconfig-rac.ini` file.

The example is a one command to clone, attach disks, and deploy a 2 node Oracle Database RAC.

Tips and tricks

Some additional tips are shown below to make deployments easier.

1. The `netconfig.ini` file can take arrays for the `NODE#` variables meaning for multi-node clusters. It may be easier to read instead using multiple blocks of NODES like this:

```
$ head -n 15 netconfig-rac.ini
# Node specific information
NODE1=odb-rac0
NODE1IP=10.169.238.201
NODE1PRIV=odb-rac0-priv
NODE1PRIVIP=10.169.238.207
NODE1VIP=odb-rac0-vip
NODE1VIPIP=10.169.238.203

NODE2=odb-rac1
NODE2IP=10.169.238.202
NODE2PRIV=odb-rac1-priv
NODE2PRIVIP=10.169.238.208
NODE2VIP=odb-rac1-vip
NODE2VIPIP=10.169.238.204
```

   It may be rewritten as this more readable and straightforward file:

```
$ head netconfig-rac.ini
NODES=(odb-rac0 odb-rac1)
NODEIPS=(10.169.238.201 10.169.238.202)
NODEPRIVS=(odb-rac0-priv odb-rac1-priv)
NODEPRIVIPS=(10.169.238.207 10.169.238.208)
NODEVIPS=(odb-rac0-vip odb-rac1-vip)
NODEVIPIPS=(10.169.238.203 10.169.238.204)
```

2. By default, the database templates ship with both network interfaces attached to the **ovirtmgmt** network. Only one connection is required for a SI database, so the second VM NIC may be removed before or after deployments if not needed using the `--clonenet` flag it is possible to adjust the private network interface to a different vNIC profile

3. To list the available networks:

```
$ deploycluster -LN
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 15:07:43 -03 2021 by user user
Using: deploycluster -LN
[LIST MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...

 -----------------------------------------------------------
  VNIC NAME   HOST NET    DESCRIPTION          ROLES
 -----------------------------------------------------------
  hostnet     hostnet     host only network    vm
  ovirtmgmt   ovirtmgmt   Management Network   vm
                                               display
                                               migration
                                               management
                                               default_route
-----------------------------------------------------------

INFO: Listing completed in 491ms
INFO: Logfile at: 'deploycluster20.log'
```

Use the `--clonenet` option during cloning to specify one or more networks in a list for the new VM instance. For example, specifying `--clonenet nic1`, will only result in one NIC provisioned with the name **nic1** attached to the default **ovirtmgmt** network. If the new VM instance needs to have two NICs, one network interface connected to the **ovirtmgmt** network and the other network interface connected to the private host-only network named **hostnet** listed in the previous output, use `--clonenet nic1,nic2=hostnet`. Finally, if three vNICs are required on the VM, with two vNICs connected to the **ovirtmgmt** network and one provisioned but not connected to any network, specify **no string** for the vNIC profile using `--clonenet nic1,nic2,nic3=`.

4. The **deploycluster** command may be used to clone from other non-database templates or boot VMs while not running any YAML **cloud-init** commands by adding the `--noexec` option and only provide the appropriate ssh public keys to the **cloud-user** user. For example, to use the **deploycluster** command to launch a generic Oracle Linux 8 VM instance, import the Oracle Linux 8 template from https://yum.oracle.com/oracle-linux-templates.html, and run the following command to only provision a VM named **ol8**, with 2GB memory, boot the instance and deploy the ssh public keys to it:

```
$ deploycluster \
  -C OL8U3_x86_64-olvm-b85 \
  -M ol8 \
  --clonedeploy \
  --clonemem 2g \
  --waitforip \
  --noexec
```

Passing `-N netconfig.ini` option to deploy instances using non Oracle DB KVM templates will not work as expected since there will not be any functions waiting to configure the network on any other template.

5. The **deploycluster** command can create disk images on a storage domain specified on the command line using the `--createdisk`. While the primary purpose of these disk images is for ASM disks, the image may be attached to VMs for any other purpose. Since a cloned VM and the ASM disks need not be on the same storage domain, the disk image creation command can be run separately before provisioning a new VM instance. Otherwise, combining the VM provisioning and disk image creation in one command will create all on a single storage domain. Creating the disk images before provisioning the instance may fit in well with a pipeline workflow. By default, new disk images are shared, 5GB in size, pre-allocated. The minimum information needed is the disk image name and the storage domain to create it on. Here is an example of creating three pre-allocated disk images on a storage domain called iSCSI:

```
$ deploycluster --createdisk disk01,disk02,disk03,storagedomain=iSCSI
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 15:14:05 -03 2021 by user user
Using: deploycluster --createdisk disk01,disk02,disk03,storagedomain=iSCSI
[CREATE DISK MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
INFO: Create raw disk 'disk01' of size '5g' in storagedomain 'iSCSI'
INFO: Create raw disk 'disk02' of size '5g' in storagedomain 'iSCSI'
INFO: Create raw disk 'disk03' of size '5g' in storagedomain 'iSCSI'
INFO: Waiting 20m0s for newly created disks to settle before proceeding
```

```
INFO: Disk 'disk01' creation complete
INFO: Disk 'disk02' creation complete
INFO: Disk 'disk03' creation complete
INFO: Create disks completed in 21.887s
INFO: Logfile at: 'deploycluster21.log'
```

Another example, creating one thin provisioned non-shared disk image, 10GB in size, on a storage domain called NFS:

```
$ deploycluster \
  --createdisk disk04,storagedomain=NFS,size=10G,shared=false,thin=true,sparse=true
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 15:17:32 -03 2021 by user user
Using: deploycluster --createdisk
disk04,storagedomain=NFS,size=10G,shared=false,thin=true,sparse=true
[CREATE DISK MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
INFO: Create cow disk 'disk04' of size '10G' in storagedomain 'NFS'
INFO: Waiting 20m0s for newly created disks to settle before proceeding
INFO: Disk 'disk04' creation complete
INFO: Create disks completed in 21.044s
INFO: Logfile at: 'deploycluster22.log'
```

Note that thin provisioned disk images cannot be shared. Also, you need to specify `thin=true,sparse=true`.

Listing available disks in OLVM (for Default cluster and DataCenter):

```
$ deploycluster -LD
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Sun Jun 27 15:18:26 -03 2021 by user user
Using: deploycluster -LD
[LIST MODE]
INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host olvm-manager.example.com (Cluster: Default DataCenter:
Default) processing request...
...
  Storage Domain: NFS
...
```

```
   - disk04                                    448.00k/10.00G    image
[509a1556-b74e-45de-b35b-0c665003b8ff]
...
INFO: Listing completed in 545ms
INFO: Logfile at: 'deploycluster23.log'
```

The below command will clone an instance from the OL79 template called **ol79-vm1**, create and attach a thin provisioned disk image 10GB in size on a storage domain called **FC**, set 2 vCPUs (using 2 sockets, with 1 core and 1 thread per socket) and attach `nic1` to the **ovirtmgmt** network:

```
$ deploycluster \
  -C OL7U9_x86_64-olvm-b86 \
  -M ol79-vm1 \
  --createdisk disk01,storagedomain=FC,size=10G,shared=false,thin=true,sparse=true \
  --clonenet nic1=ovirtmgmt \
  --clonemem 2G \
  --clonecpu 2,1,1
```

When creating a disk image while cloning the instance, there is no need to use `--cloneattach` to attach it to the instance, as any `--createdisk` when cloning automatically assumes `--cloneattach`.

Since the `--clonedeploy` was not passed, the cloned VM will remain in shutdown state, this is normal and expected. Do not boot that instance using the OLVM portal, the ssh public key still need to be deployed to it. Boot it using the following **deploycluster** command:

```
$ deploycluster -M ol79-vm1
```

As this is an asynchronous operation, and the `--waitforip` option was not used, it is necessary to check the instance IP at the OLVM portal before attempting to access it.

It is possible to mix creating disk images and attaching pre-existing images to a clone using the following example:

```
$ deploycluster \
  -C OLVM-OL8U4-19110DBRAC* \
  -M si-19c1-1 \
  --createdisk datadisk01,datadisk02 \
  --cloneattach asm5
```

The following example clones a template to create a 2-node RAC, creating shared disks for ASM using a mask for its names with `namefmt` and attaching it to the instances, configuring instance names and network using the `netconfig.ini` file, setting memory to 16GB, vCPUs to 8, starting the deployment, attaching nic1 and nic2 to ovirtmgmt network, publishing the user's ssh public key and the admin public key, and changing wait time to one hour. See following snippet of the `netconfig.ini` file:

```
$ cat netconfig-rac.ini
NODE1=odb-rac0
...
NODE2=odb-rac1
...
```

The command to clone and deploy this RAC instance:

```
$ deploycluster \
  -C OLVM-OL8U4-19110DBRAC-KVM \
  --createdisk namefmt=odb-asm%1d,count=5 \
  -N netconfig.ini \
  --clonedeploy \
  --clonemem 16G \
  --clonecpu 8,1,1 \
  --clonenet nic1,nic2=ovirtmgmt \
  --pubkeys ~/.ssh/id_rsa.pub,~/admin.pub \
  --waitforip \
  --wait 1h
```

Note, when not passing `-M <instance-name>` but providing the `netconfig.ini` file as argument, the instances names will be set from `NODE#` option included in `netconfig.ini`.

6. An optional workbook can be created detailing what disks and VMs were created by the Deploycluster tool, which can help clean up an environment or logging what was created and when. Set the `deploycluster.ini` option `DEPLOYCLUSTER_WORKBOOK` to a filename or specify the `--workbook <filename>` option on the command line. Every creation will be recorded in that filename. The workbook option may then be used with the `--remove` command to destroy all that was previously created.

Configuring a persistent workbook in the `deploycluster.ini` file:

```
...
# Should each invocation create a workbook. The workbook file lists the
# created vm's and disk uuid's for easier perusal or removal. This file
# is appended to on each run with a timestamp.
# Default: No workbook is created
DEPLOYCLUSTER_WORKBOOK=~/workbook.txt
...
```

Cloning a new instance and creating an additional disk for it:

```
$ deploycluster -C OL7U9_x86_64-olvm-b86 -M ol79-vm2 --createdisk ol79-vm2-disk01
...
```

Checking the workbook.txt file:

```
$ cat ~/workbook.txt
#
# deploycluster v4.0.0 workbook
# created on Mon Jun 28 14:46:26 UTC 2021
# command: deploycluster -C OL7U9_x86_64-olvm-b86 -M ol79-vm2 --createdisk ol79-vm2-
disk01
#
id=abd062d9-a8a5-4a43-bfe8-019e875b9fdd
disk=549f196b-05dd-4b2e-9143-ab61089af9a4
vm=2f7aedf4-5496-4b58-abf7-0724f0cdbb7e
```

Removing objects using the workbook (to avoid the 20min safeguard/wait, add –wait 0):

```
$ deploycluster --remove --workbook ~/workbook.txt
Oracle DB/RAC Deploycluster (v4.0.0) for Oracle Linux Virtualization Manager - (c) 2021
Oracle Corporation
Started on Mon Jun 28 14:48:08 UTC 2021 by user user
Using: deploycluster --remove --workbook /home/user/workbook.txt
[REMOVE MODE]

INFO: Reading defaults from ~/.deploycluster.ini
INFO: Connecting to OLVM host ovirt.it.oracle.com (Cluster: cluster03 DataCenter: kvm03)
processing request...
INFO: Sleeping for 20m0s to give a chance to terminate this command. To avoid wait,
specify --wait time as 0 on the command line
INFO: Removing disk 'ol79-vm2-disk01' [549f196b-05dd-4b2e-9143-ab61089af9a4]
INFO: Removing VM 'ol79-vm2' [2f7aedf4-5496-4b58-abf7-0724f0cdbb7e]
```

```
INFO: Removal completed in 20m10.533s
INFO: Logfile at: 'deploycluster105.log'
```

7. For easier machine parsing, the `--json` option may be used to list disks, VMs, storage domains, Data Center, templates in JSON format. Then, any compliant json parser can consume the output.

Basic Troubleshooting

1. Passing an option more than once does work as expected. In this example customer intend to create three instances:

   ```
   $ deploycluster \
     -C OL7U9_x86_64-olvm-b86 \
     -M ol79-vm1 \
     -M ol79-vm2 \
     -M ol79-vm3
   ```

   The previous command will only create **ol79-vm3**. The correct command to create more than one clone from that template is by separating them with a “,” (comma):

   ```
   $ deploycluster \
     -C OL7U9_x86_64-olvm-b86 \
     -M ol79-vm1,ol79-vm2,ol79-vm3
   ```

2. Cloning an instance and the resultant VM does not get an IP address, even when IP address is configured in `netconfig.ini` file, inspect the instance and check if its network attachments are correct:

   ```
   $ deploycluster -L -M odb-si-ol79
   ...
     Vm name: odb-si-ol79                                  [1ffe2b77-818c-4114-9b3e-...
     Status: up   Memory: 8.00G [8589934592]   Vcpus: 4
     Description: Generated by oracle-linux-image-tools
     Comment: created by deploycluster v4.0.0
     Nics:
      - nic1 [56:6f:a7:e0:00:35] (empty)
   ```

```
  Disks:
   - Disk_OL7U9_x86_64-olvm-b86 (1.00G/37.00G, image)   [e58fe97e-11f6-42bd-a332-...
...
```

In example above, the cloned instance did not configure network properly as the interfaces from the template with which it was cloned from has improper network attachment, during cloning it is possible to pass `--clonenet` option to the **deploycluster** command as follow:

```
$ deploycluster \
  -C OLVM-OL7U9-19110DBRAC-KVM \
  -M odb-si-19col79 \
  --clonenet nic1=ovirtmgmt
```

Choose the appropriate network to match the environment. List networks is available using the `–LN` option in the **deploycluster** command.

References

[1] https://golang.org

[2] Deploycluster https://www.oracle.com/database/technologies/rac/vm-db-templates.html

[3] https://www.oracle.com/database/technologies/virtualization-matrix.html

[4] https://www.oracle.com/a/tech/docs/oracle-rac-in-oracle-linux-kvm.pdf