Last Modified: 21-Feb-2019

DeployCluster tool for Oracle VM Templates for Oracle Database Single Instance and Real Application Clusters Linux X86 64bit (OneCommand/Oracle VM)

Table of Contents:

DeployCluster Overview:

This document describes the **deploycluster** tool that can be used in all Oracle VM 3 environments to quickly and automatically deploy the popular Oracle VM Templates for Oracle Database - Single Instance and Real Application Clusters starting from version 11g Release 2 and up.

The deploycluster tool is a standalone program that can run from any Linux server. It connects to Oracle VM 3 Manager to automatically configure any number of nodes (VMs) to create a fully functional Single Instance, Single Instance/HA (Oracle Restart) or Real Application Cluster environment. The tool assumes all VMs are pre-created from the base Oracle DB/RAC OVM Template and in the case of Oracle RAC, assigned correct disks & network to allow a successful cluster deployment. It then starts the VMs (if not already started), verifies they are configured correctly (disk, network, etc.) then sends the network & build parameters to all VMs. This configures the network in all of the VMs and optionally launches a **buildsingle** to create a Single Instance environment, or **buildcluster** operation on the N-nodes cluster to obtain a fully configured Oracle RAC environment. The tool may also be used to configure a single node (VM), to add to an existing cluster, act as a single RAC node or as a Single Instance or Single Instance/HA (Oracle Restart) environment.

For the latest version of this document and more information on the benefits of Oracle VM and Templates, see OTN at: https://www.oracle.com/database/technologies/rac/vm-db-templates.html and Note# 1185244.1 on My Oracle Support.

Shared Disks Implementation Choices:

The shared disks holding the Oracle RAC database files may be configured as '**Physical'** or '**Virtual'** disks. Recently Oracle RAC Support policies allow Production deployments to use Virtual disks with some cautionary provisions (See Whitepaper below for details).

For more details carefully review the updated Oracle RAC on Oracle VM environments whitepaper at: http://www.oracle.com/technetwork/database/clustering/oracle-rac-in-oracle-vm-environment-131948.pdf

In **Single Instance** deployments Virtual Disks are a lot safer to use from a performance standpoint. See Whitepaper above for further details.

Naming Convention:

The Deploy Cluster tool may be referred to as **deploycluster**, **Deploycluster** or **DeployCluster** the actual command to execute the tool is **deploycluster.py,** all these names are synonymous.

Template file names may look something like:

OVM_OL7U3_**X86_64**_12102RAC_PVM-**1of2** & **2of2** (or **X86** for 32bit)

Screenshots may have slightly different name, these are generic screenshots, substitute your version number and architecture, i.e. 11202, 11203 based on release used.

WHAT'S NEW?

- **Version 3.0 of deploycluster tool supports Oracle VM 3.3.1 or higher**
  - **For Oracle VM 3.2 or lower continue using Deploycluster Version 2.x.**
- Version 2.0 or above of deploycluster tool:

    - Supports **Flex Cluster** and/or **Flex ASM** deployment
        - Allows for optional 3rd NIC, useful in Flex ASM case for ASM traffic
    - Support for **Single Instance** or **Single Instance/HA** (Oracle Restart) deployments
- Version 1.1.5 or above of deploycluster tool supports Oracle VM 3.2.
    - By default connection to current/newer OVM 3.2.1 Manager is assumed; option DEPLOYCLUSTER_DEFAULT_CONNECT_311 allows further control, see FAQ#10.

Minimum Software Requirements

The Linux server that the deploycluster tool is running from should have:

- Python 2.4 (or above)
- Java 1.6 [1](or above) – OpenJDK Java or GNU Compiler for Java (gcj) are not supported
- grep, tail, file, bc & bash (rpms)

The VMs software requirements:

- **VMs:** All VMs must be created from the base Oracle DB/RAC OVM Template and properly recognized by a functioning Oracle VM Manager for the deploycluster tool to work properly.
- **Kernel considerations**: For the **deploycluster** tool to properly send the deployment details to the VMs, the VMs must all run a Linux Kernel that supports the Oracle VM Guest Additions. Currently all versions of Unbreakable Enterprise Kernel (UEK) support that, and that is also the default kernel of the templates. If the VMs use any other kernel, the tool will boot them, however they will not be able to read the sent keys and as a result the network will not be setup nor will the configuration of software take place.
- **Single Instance Support:** Only DB/RAC OVM Templates released in 2013 and later have the Single Instance or Single Instance/HA support (The "**DBRAC**" or "**DBSE"** appears in template's file name). Attempting to deploy Single Instance mode on older templates will not work, and a normal (one or N-nodes) RAC Cluster will be built instead.

Note: The minimum hardware requirements needed to run the Oracle DB/RAC Templates are described in the documentation that comes with the specific template and is based on the release and deployment mode (Single Instance, RAC, clusterware only, etc.).

Main features of DeployCluster tool

- Allows end-to-end fully automated **Single Instance** or **RAC cluster deployment** (assuming VMs are pre-created w/NICs & shared disks) of N-node clusters without ever logging into any of the VMs.
- VMs can be identified by UUID or simple name which may include the "*" and "?" wildcard characters (duplicate or illegal names detected on command line or on Oracle VM Manager)
- Connection to Oracle VM Manager 3.2 and lower can be SSL or non-SSL, with default protocol picked in an automated way based on remote or local connection; connection to Oracle VM 3.3 are SSL only.
- Minimum Memory, Network Adapters (NICs) and shared Disks (in the case of RAC) are checked for all VMs with option to skip any or all checks.
- Allow ANY build configuration (SID name, user name, passwords, ports, etc.) to be modified from outside the guests at deploy time by supplying a custom params.ini (**-P** flag).
- Allow prompting and sending of passwords for all users in the most secure way (SSL end-to-end)

[1] Since Java is not installed and should not be installed on dom0 it is not supported to run the **deplocluster** tool from dom0.

- All exceptions are trapped showing a default of 4 lines from it, configurable with DEPLOYCLUSTER_EXCEPTION_LINES in deploycluster.ini
- Automated logfile written upon each invocation (except -h for usage)
- Ability to simulate a deployment attempt using the **--dryrun** (**-D**) flag
- Ability to list VMs using the **--list_vms_only** (**-L)** flag, without a deployment attempt; honors the **--vms** (**-M**) flag.
- Color coded error (red), warning (yellow), info (green) messages (possible to disable). No color related escape sequences are written to the automated logfile.
- All already released templates are fully compatible (as long as the OVMAPI enabled OS disk is used.)

Reminder regarding Deploycluster Versions:

- **Deploycluster Version 3.* supports Oracle VM Manager 3.3.1 and higher.**
- **Deploycluster Version 2.* supports Oracle VM Manager 3.2 and lower.**

Feedback:

Feel free to post feedback at the Oracle VM or Oracle RAC Forums on the Oracle Technology Network:

Oracle VM:
https://community.oracle.com/community/server_%26_storage_systems/virtualization/oracle_vm_server_for_x86

Oracle RAC:
https://community.oracle.com/community/database/high_availability/rac__asm_%26_clusterware_installation

Or contact Oracle Support.

NOTE: The Deploycluster tool is only supported for deploying the DB/RAC Templates in Oracle VM 3 environments. Do not modify the tool or run in other environments.

Deployment Overview

For a **RAC Deployment**: Each VM requires 2 (or more) network adapters (NICs) as well as 5 common shared disks (by default, or more; using fewer disks is supported) if ASM is configured.
For a **Single Instance/HA Deployment** (Oracle Restart)**:** A single network adapter (NIC) and well as 5 disks (by default, or more; using fewer disks is supported) if ASM is configured.
For a **Single Instance**: A single network adapter (NIC) and no shared disks are needed. The database must reside on a filesystem.

The exact steps to create such VMs are slightly different between the various Oracle VM 3 versions due to User Interface (UI) improvements/changes. The rest of the document will provide the steps for some releases as well as how to use the **deploycluster** tool to create a Single Instance or RAC cluster from these VMs. You may also follow normal Oracle VM documentation to create the VMs for Single Instance or RAC deployment. The high level steps are:

1. Import the desired DB/RAC Template
2. Create or identify the (shared) disks to hold the Database
3. Adjust the template's network bridges
4. Clone VM(s) from the DB/RAC Template
5. Based on deployment mode, attach the (shared) disks to all VMs as needed.
6. Download deploycluster tool
7. Create a netconfig.ini file
8. Run the deploycluster.py tool

Deployment in an Oracle VM 2 environment

Oracle VM 2 users or those who insist on using the 2-node interview, should use the deployment methods described in the PDFs that are included inside the template's zip files. The methods described below using the **deploycluster** tool are only valid in Oracle VM 3 environments with a functioning Oracle VM Manager.

Main steps, using Oracle VM Manager 3 and deploycluster tool

These steps are using Oracle VM Manager 3 Browser User Interface. These steps can easily be automated using Oracle VM Command Line Interface (OVM CLI), see documentation for CLI syntax.

1) Import the desired DB/RAC Template

Each template consists of **two disks**, first (1of2) is the OS disk; the second (2of2) is the Oracle disk. The Oracle disk may be split into **two parts** which **must be combined into a single file** before importing. The template's zip files contain inside them a tar.gz (or .tgz) or .tbz file. Oracle VM 3 Manager allows importing the zip file directly (e.g. p14000016_10_ Linux-x86-64_1of2.zip) or the .tgz (or .tbz) that is inside it (e.g. OVM_OL5U8_X86_64_11202RAC_PVM-1of2.tgz), it **cannot** import partA & partB, those must be **combined into a single file in order to import**. Stage **both** disks (must combine partA and partB of the Oracle disk) of a given template on an ftp/http server, and import them using Oracle VM 3 Manager: by providing **both** URLs at the **same import session.**

Example of template with Oracle disk split into two (3 downloadable zip files):

```
# cd /tmp
# unzip -q /tmp/p24339999_10_Linux-x86-64_1of3.zip &
```

→ Above will produce: OVM_OL7U3_X86_64_12102DBRAC_PVHVM-1of2.tar.gz ready for import.

```
# unzip -q /tmp/p24339999_10_Linux-x86-64_2of3.zip &
# unzip -q /tmp/p24339999_10_Linux-x86-64_3of3.zip &
```

→ Above will create two **tar.gz** files that must be combined before import.

```
# wait
#
# Note below 'cat' command should be all on one line
# cat /tmp/OVM_OL7U3_X86_64_12102DBRAC_PVHVM-2of2-partA.tar.gz
/tmp/OVM_OL7U3_X86_64_12102DBRAC_PVHVM-2of2-partB.tar.gz >
OVM_OL7U3_X86_64_12102DBRAC_PVHVM-2of2.tar.gz
```

Import the **two** template files (tar.gz of first file and the combined .tar.gz of second file) directly in ONE import operation using Oracle VM Manager UI or OVM CLI.

For example, import these two files in ONE import operation from the Oracle VM UI:

http://server1/ftp/OVM_OL7U3_X86_64_12102DBRAC_PVHVM-1of2.tar.gz
http://server1/ftp/OVM_OL7U3_X86_64_12102DBRAC_PVHVM-2of2.tar.gz

Note: If template has only two .zip files, then simply unzip both, and import, no need to combine anything.

Here is how the Import Template UI looks like in Oracle VM 3:

OVM3.1 & above: Repositories->RepoName->VM Templates->Import Template…
OVM3.0: Home->Server Pools->Templates->Import Template…

Note: Only DB/RAC OVM Templates released in 2013 and later support **Single Instance** deployment. Confirm on the OTN page on release dates.

Sample URLs for import may look like this:

ftp://server1/ovm/OVM_OL6U2_X86_64_11203RAC_PVM-1of2.tgz (or .zip)
ftp://server1/ovm/OVM_OL6U2_X86_64_11203RAC_PVM-2of2.tgz

Clicking OK will create a job that will download and unzip/untar the template files (may take some time) and stage them for further usage. This is a **one time** operation, subsequent VMs can be cloned from this template.

TIP: If you have no proper http/server setup in the environment, the following single python command will start an HTTP server which will serve all files from the current directory and all subdirectories:

$ python -m SimpleHTTPServer 8000[2]

Now browse to this location: **http://hostname:8000/**

Mixing and Matching OS & Oracle disks:

The DB/RAC Templates are designed to support mixing and matching of the OS and Oracle disks, however, that is not possible to do as part of a **template import**. During template import, matching template files should be imported together, and then later individual disks could be cloned and attached to a new VM.

A quicker alternative to obtaining mixed OS & Oracle disks is to directly import individual disks:

OVM3.1 and above: Repositories->RepoName->Virtual Disks->Import Virtual Disk …
OVM3.0: Home->Server Pools->Repositories->RepoName->Virtual Disks->Import Virtual Disk…

Then clone these virtual disks and attach to VMs as needed.

[2] As a security measure run this command from a directory holding only the template files, and terminate it as soon as the import is finished.

2) Create or identify the (shared) disks to hold the Database

Based on the deployment mode (Single Instance, RAC, clusterware only, etc), use Oracle VM manager to create or identify the (shared) disks that will hold the database.
For a **RAC Deployment**: 5 (by default, or more) common **shared** disks are needed if ASM is configured.
For a **Single Instance/HA Deployment** (Oracle Restart)**:** A single network adapter (NIC) and 5 (by default, or more) disks are if ASM is configured.
For a **Single Instance**: A single network adapter (NIC) and no extra disks are needed, hence this step may be skipped. The database must reside on a filesystem.

If using less than or more than 5 disks, a modified params.ini will need to be specified using deploycluster's --**params (-P)** flag (See Appendix C).

First time users deploying **Single Instance/HA** or **RAC** may choose to keep it simple and create 5 virtual disks named: ASM1, ASM2, ASM3, ASM4, ASM5 or similar names.

As stated initially, test deployments can use Virtual Disks, whereas production deployments are required to use physical disks.

To create the (shared) disks follow one of these steps:

- **Virtual disks (Test deployments):**

  OVM3.1 and above: Repositories->RepoName->Virtual Disks->Create Virtual Disk…
  OVM3.0: Home->Server Pools->Repositories->RepoName->Virtual Disks->Create Virtual Disk…

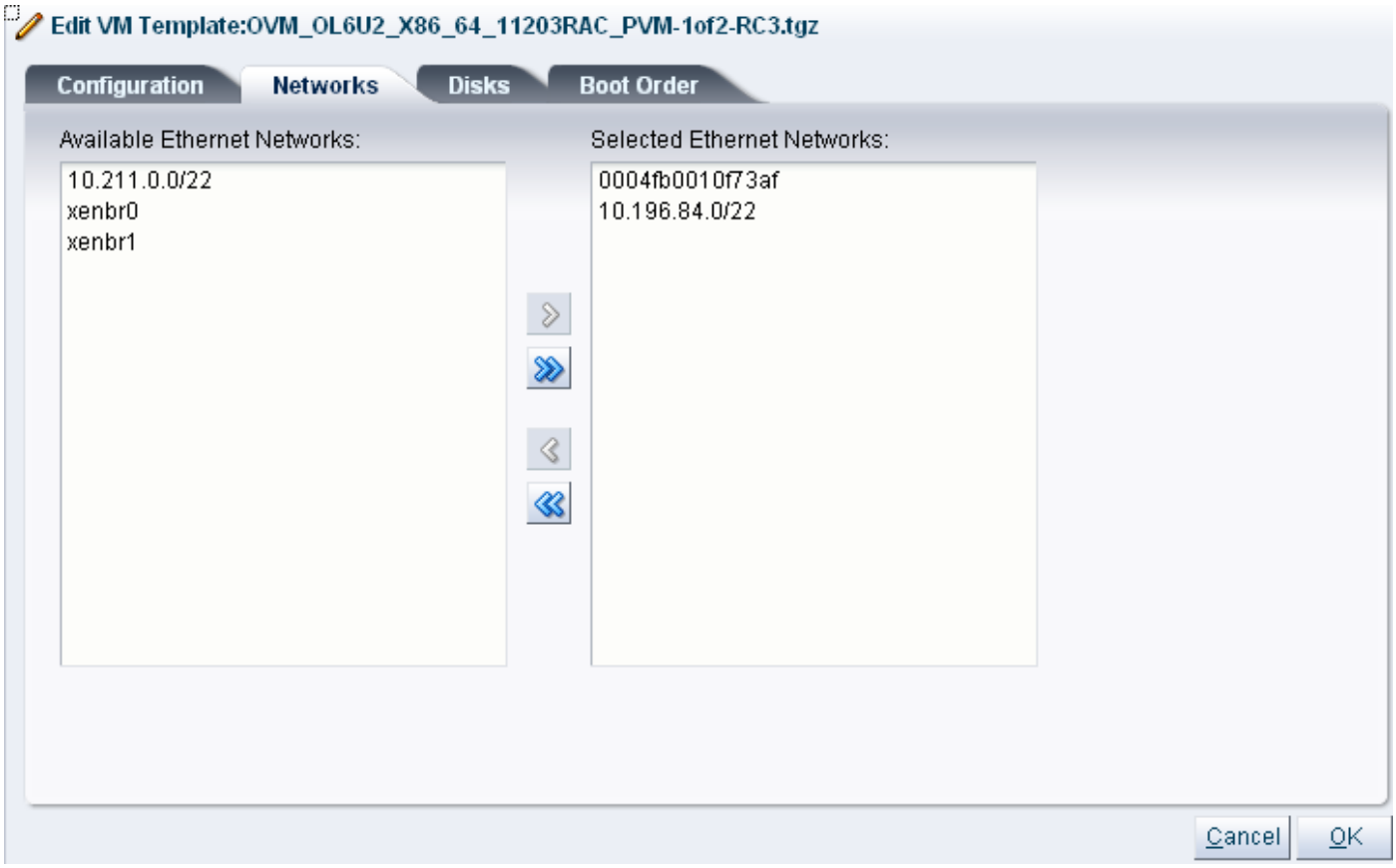
- **Physical disks (Production deployments):**
  In most cases, physical disks/LUNs should automatically be discovered by the server, and selectable to be attached to VMs in step 5 below. Follow Oracle VM documentation for more details about physical disk creation, based on your type of storage & storage plugins.

3) Adjust the template's network bridges

Before cloning VMs from this template it is best to assign it the correct network bridges.

OVM3.1 and above: Repositories->RepoName->VM Templates->TemplateName->Edit…
OVM3.0: Home->Server Pools->Templates->TemplateName->Edit…

Once in Edit mode, click the **Networks** tab and make sure the correct network bridges are shown on the right side (under "Selected Ethernet Networks").By default the templates load with **xenbr0** & **xenbr1**, however, those may not be defined in your environment, hence choose the right bridges from the "Available Ethernet Networks" and remove the xenbr0/1 from the "Selected Ethernet Networks". By default, the public network is on first Virtual NIC (vNIC) and private is on the second vNIC. Below screenshot shows how network bridge mapping may look like after above steps are followed:

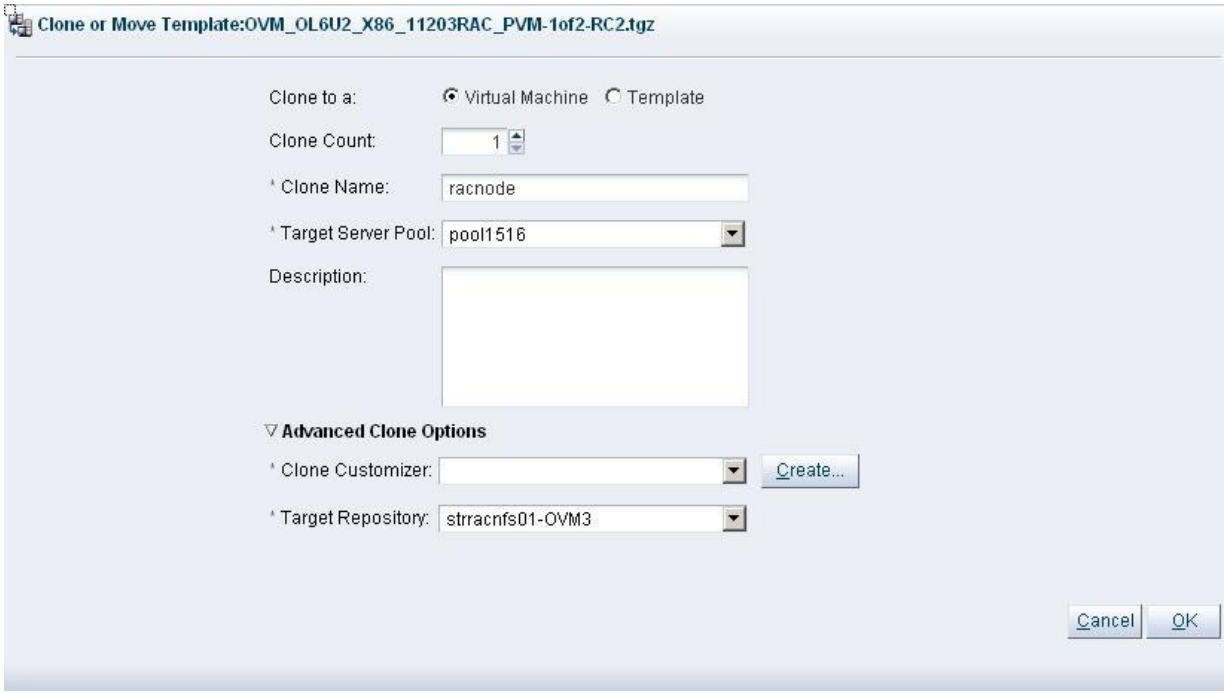


NOTE: Single Instance & Single Instance/HA require only one NIC and supports an optional second NIC. Oracle RAC deployment requires 2 NICs, and Oracle 12c allows for a 3rd NIC for the ASM traffic (See netconfig.txt)

4) Clone VMs from the DB/RAC Template

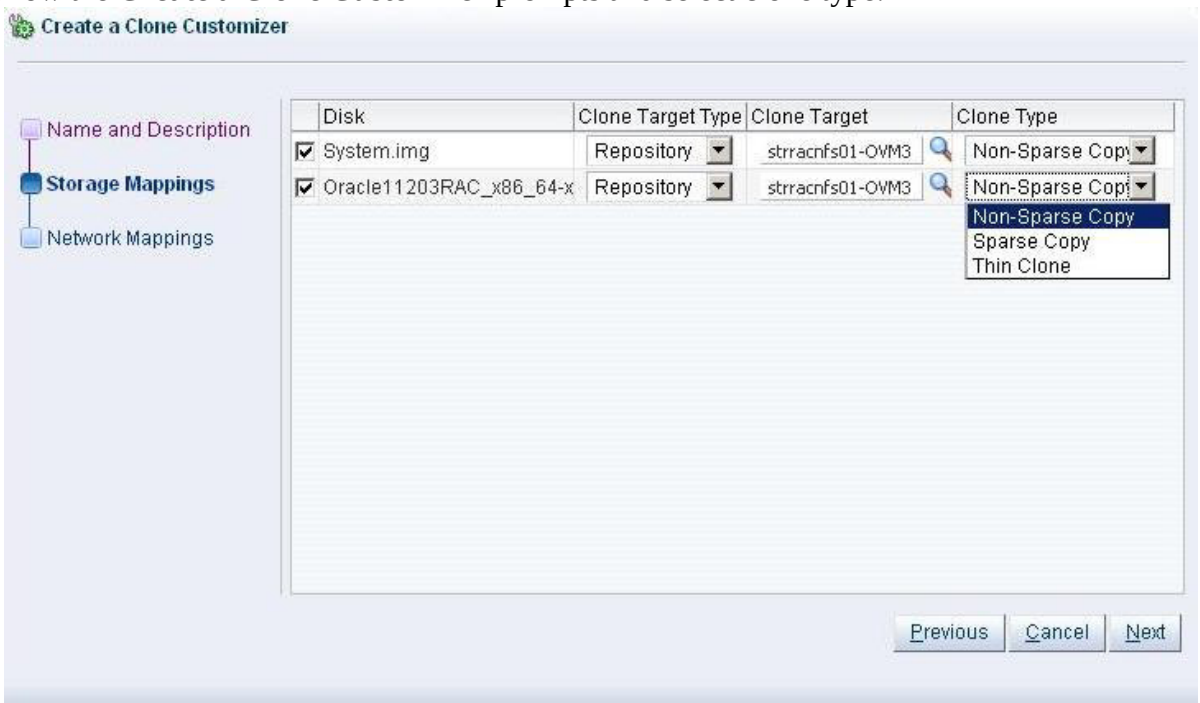
From the DB/RAC Template, clone as many VMs (nodes) as needed for Single Instance or any size cluster.

OVM3.1 and above: Repositories->RepoName->VM Templates->TemplateName->Clone…
OVM3.0: Home->Server Pools->Templates->TemplateName->Clone…



Select "Clone to a Virtual Machine" as well as count & name. In above example, since name is racnode and count is 2, the names will be "racnode.0" & "racnode.1". If thin/sparse allocation is desired, click under **Advanced Clone Options** and click **Create.**

Then follow the **Create a Clone Customizer** prompts and select clone type:



Non-Sparse Copy: Full copy of disk, consuming all space up front.
Sparse Copy: Fully copy, except sections that are all zeros, thus potentially reducing file size.
Thin Clone: Based on OCFS2 v1.8's **reflink** feature, which creates a snapshot of a file, only modified blocks are reproduced, so if file does not get a lot of writes this option can yield huge savings in space. This option should not be used for database files.

5) Attach the (shared) disks to all VMs

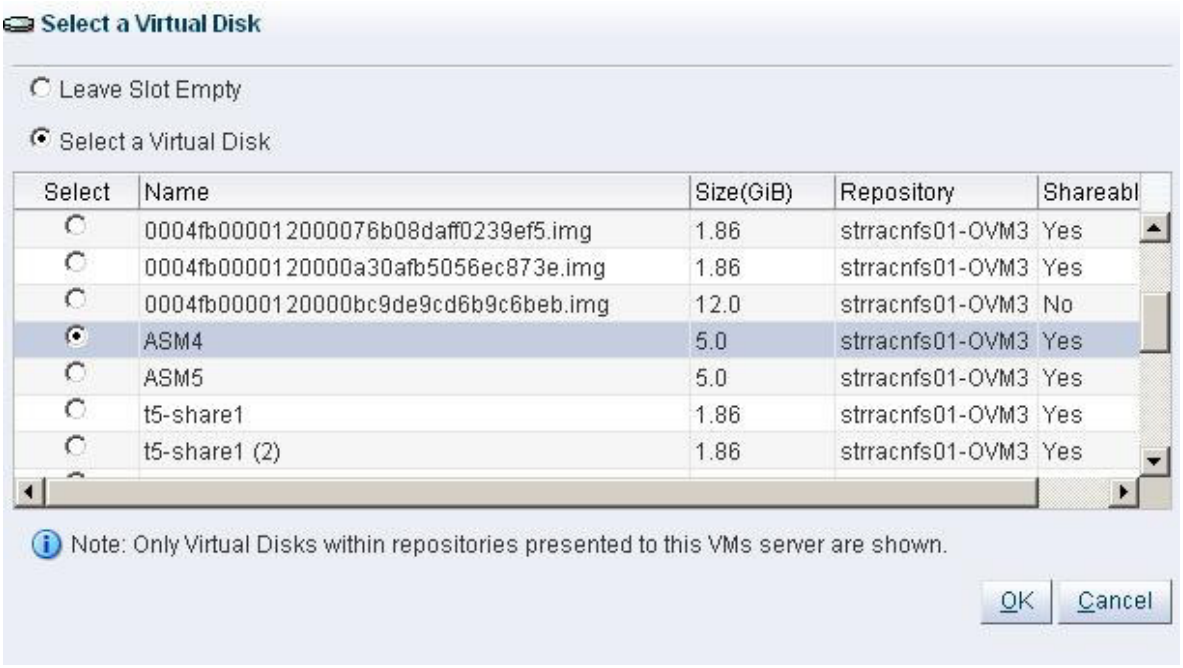
As mentioned in step 2 above; **RAC** & **Single Instance/HA** deployments require (shared) disks if ASM is configured. **Single Instance** users may skip this step.

The OneCommand for Oracle VM build engine, Oracle RAC or ASM do not require that disks have identical names or ordering on all nodes, however, it might be easier for humans to deal with identical names and ordering. Therefore it is recommended to add disks in the same order (which will result them having same device names) to all VMs, e.g.: ASM1, ASM2, ASM3, ASM4, ASM5. Go to:

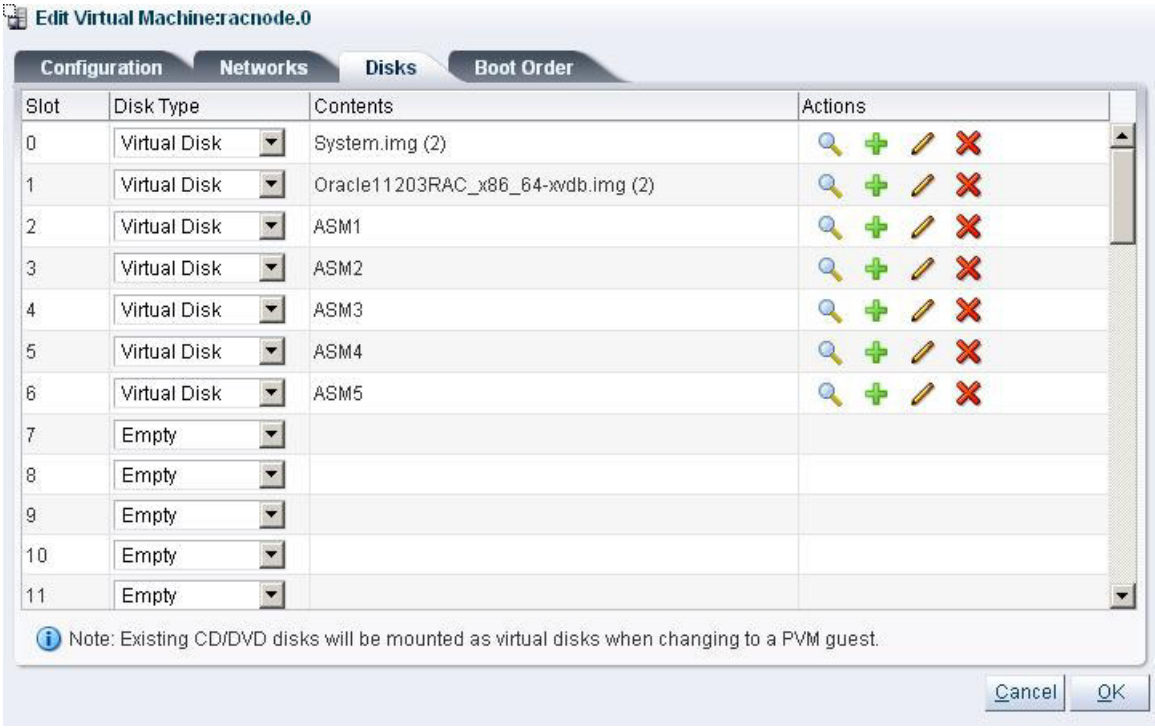
OVM3.1 and above: Servers and VMs->Server Pools->PoolName->ServerName->VMName->Edit->Disks…
OVM3.0: Home->Server Pools->PoolName->ServerName->VMName->Edit->Disk Ordering…

- **Virtual disks (Test deployments):**

  Then choose a disk slot in the VM and select the shared disk that will occupy that slot:



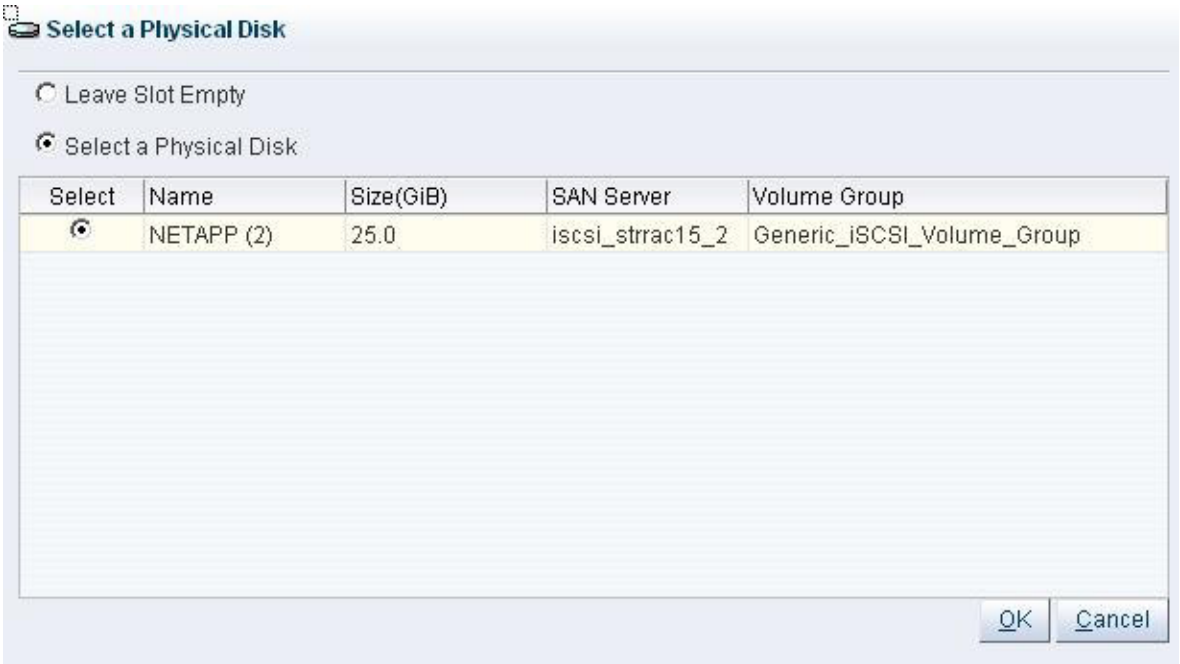
Once completed it will look something like this:



Attach the (shared) disks created or identified in step 2 to all cloned VMs. The order of disks is only important if the traditional 2-node interview is used. Using the 2-node interview should not be needed given the capabilities of the **deploycluster** tool. If the disks are attached in different order such that their names are also different on each of the VMs, for example xvdc is really xvdf on another VM, remember to set PERSISTENT_DISKNAMES=no in params.ini and pass that params.ini file when running deploycluster, since by default, same device names are assumed. In such cases, or if more than 5 disks are used, RACASMDISKSTRING may need to be adjusted to encompass all devices on all nodes, as well as ALLDISKS, see params.ini for details.

- **Physical disks (Production deployments):**

  Navigate to same location, and in pull down select "Physical Disk" instead of "Virtual Disk", and then select the physical disk to be used, for example:



6) Download deploycluster tool

Obtain the latest version of the **deploycluster** tool from OTN (must accept license agreement):

https://www.oracle.com/database/technologies/rac/vm-db-templates.html (may redirect to accept license)
For example: DBRACOVM-Deploycluster3-tool.zip (Oracle VM 3.3 and above)
DBRACOVM-Deploycluster2-tool.zip (Oracle VM 3.2 and below)

Then unzip it anywhere you wish, e.g.

```
$ cd $HOME
$ unzip -q DBRACOVM-Deploycluster3-tool.zip
$ cd deploycluster3
```

Some of the useful files to notice are:

deploycluster.py	**- Deploycluster tool – RUN THIS**
deploycluster.ini	- Options for deploycluster tool itself
README.txt	- Details explanation of all flags/options
utils	- Directory with useful files, e.g:
netconfig*.ini	- Sample netconfig.ini files
netconfig.txt	- Detailed explanation of netconfig.ini syntax and options
netconfig.zip	- Updated netconfig (may be needed when deploying older templates)
params-sample11g.ini	- Sample 11g params.ini
params-sample12c.ini	- Sample 12c params.ini
README.txt	- Backwards compatibility steps for Oracle VM2 users

7) Create a netconfig.ini file for deployment

Using a simple text editor copy the sample netconfig.ini file from the utils directory, then adjust the names and IPs suitable for your environment.

Here is a sample netconfig.ini file for a **Single Instance** deployment (base sample file netconfig-sample-si.ini):

```
# Sample Single Instance or Single Instance/HA
NODE1=test1
NODE1IP=192.168.1.101
#NODE1PRIV=test1-priv       # Optional
#NODE1PRIVIP=10.10.10.101   # Optional

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

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

Here is a sample netconfig.ini file for a **2-node RAC cluster** deployment:

```
# Node specific information
NODE1=test30
NODE1VIP=test30-vip
NODE1PRIV=test30-priv
NODE1IP=192.168.1.30
NODE1VIPIP=192.168.1.32
NODE1PRIVIP=10.10.10.30
NODE2=test31
NODE2VIP=test31-vip
NODE2PRIV=test31-priv
NODE2IP=192.168.1.31
NODE2VIPIP=192.168.1.33
NODE2PRIVIP=10.10.10.31
# Common data
PUBADAP=eth0
PUBMASK=255.255.255.0
PUBGW=192.168.1.1
PRIVADAP=eth1
PRIVMASK=255.255.255.0
RACCLUSTERNAME=twonodes30
DOMAINNAME=localdomain     # May be blank
DNSIP=       # 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=test30-31-scan
SCANIP=192.168.1.34
```

Make sure there are no duplicate values; those will be flagged during deployment in the step below. The setting of NETCONFIG_DEV is irrelevant; it is only used during the traditional 2-node interview.

8) Running deploycluster.py (Sample run logs)

Now run the **deploycluster** tool in the most basic and common use case. Here we only supply user (**-u** flag) & password (**-p** flag) for the Manager, VMs to operate on (**-M** flag), and the netconfig.ini file (**-N** flag) prepared in prior step. This assumes the command is executed on the Oracle VM Manager host, if not, add the **-H** flag and indicate the host that the Manager is running on, see rest of this document for more details. It is also possible to add the **-D** flag, to run in 'dryrun' mode to see a simulation of the operations that will be performed. Running in dryrun mode will make no changes to the environment, and this will give a good indication of weather the actual deployment will succeed. Adding the **-L** flag will only list the details for the specified VMs. In both cases, simply add these flags to the command below.

Sample Single Instance Deployment: The below command will power ON one VM and deploy it in Single Instance mode (partial output shown):

```
$ ./deploycluster.py -u admin -p MyP123 -M single.1 -N netconfig-si.ini
```

```
INFO: Inspecting /home/oracle/deploycluster/netconfig-si.ini for number of nodes
defined...
INFO: Detected 1 nodes in: /home/oracle/deploycluster/netconfig-si.ini

INFO: Located this one VM with a simple name of: ['single.1']

INFO: Detected a Single Instance deployment...

INFO: Starting the single VM...

INFO: VM with a simple name of "single.1" is in a Stopped state, attempting to start
it....OK.

INFO: Verifying that the single VM is in Running State and passes prerequisite
checks....

INFO: The only VM with a simple name of "single.1" specified on command line has (2)
disks

INFO: The (1) VM passed basic sanity checks and in Running state, sending details as
follows:
      netconfig.ini (Network setup): /home/oracle/deploycluster/netconfig-si.ini
      buildsingle: yes

INFO: Starting to send configuration details to the one VM....
INFO: Sending to VM with a simple name of "single.1".....

INFO: Configuration details sent to (1) VM...
      Check log (default location /u01/racovm/buildsingle.log) on build VM
(single.1)...

INFO: deploycluster.py completed successfully at 08:45:21 in 28.1 seconds (00m:28s)
Logfile at: /home/oracle/deploycluster/deploycluster1.log
```

Sample RAC deployment: The following command will power ON both racnode.0 & racnode.1 VMs; then send them the netconf2nodes.ini file and initiate a buildcluster (full output):

```
$ ./deploycluster.py -u admin -p MyP123 -M racnode.0,racnode.1 -N netconf2nodes.ini
Oracle RAC OneCommand (v1.1.0) for Oracle VM - deploy cluster - (c) 2011-2012
Oracle Corporation
 (com: 26700:v1.1.0, lib: 126075:v1.1.0, var: 1100:v1.1.0) - v2.4.3 -
server01.us.oracle.com (x86_64)
Invoked as root at Sat May 26 12:39:07 2012  (size: 37500, mtime: Wed May 16
00:13:19 2012)
Using: ./deploycluster.py -u admin -p **** -M racnode.0,racnode.1 -N
netconf2nodes.ini

INFO: Attempting to connect to Oracle VM Manager...

INFO: Oracle VM Client  (3.1.1.305) protocol (1.8) CONNECTED (tcp) to
      Oracle VM Manager (3.1.1.212) protocol (1.8) IP (144.25.123.123) UUID
(0004fb0000010000f2bbb9552696c301)

INFO: Inspecting /home/oracle/deploycluster/netconf2nodes.ini for number of nodes
defined...
INFO: Detected 2 nodes in: /home/oracle/deploycluster/netconf2nodes.ini

INFO: Located a total of (2) VMs;
      2 VMs with a simple name of: ['racnode.0', 'racnode.1']

INFO: Verifying all (2) VMs are in Running state

INFO: VM with a simple name of "racnode.0" is in a Stopped state, attempting to
start it...OK.

INFO: VM with a simple name of "racnode.1" is in a Stopped state, attempting to
start it...OK.

INFO: Detected that all (2) VMs specified on command have (5) common shared disks
between them (ASM_MIN_DISKS=5)

INFO: The (2) VMs passed basic sanity checks and in Running state, sending cluster
details as follows:
      netconfig.ini (Network setup): /home/oracle/deploycluster/netconf2nodes.ini
      buildcluster: yes

INFO: Starting to send cluster details to all (2) VM(s).....
INFO: Sending to VM with a simple name of "racnode.0"....
INFO: Sending to VM with a simple name of "racnode.1"......

INFO: Cluster details sent to (2) VMs...
      Check log (default location /u01/racovm/buildcluster.log) on build VM
(racnode.0)...

INFO: deploycluster.py completed successfully at 12:39:33 in 25.7 seconds (00m:25s)
Logfile at: /home/oracle/deploycluster/deploycluster2.log
```

If you were to connect to the console at the same time the VMs are starting, you will see the familiar configuration of network and the starting buildcluster, as follows:

```
INFO (node:test30): Configuring network on local node using node number 1...
    Generating Network Configuration files...
    Modifying /etc/hosts
    Modifying /etc/resolv.conf
    Modifying /etc/sysconfig/network
    Modifying /etc/sysconfig/network-scripts/ifcfg-eth0
    Modifying /etc/sysconfig/network-scripts/ifcfg-eth1
Shutting down loopback interface:  [  OK  ]
Bringing up loopback interface:  [  OK  ]
Bringing up interface eth0:  [  OK  ]
Bringing up interface eth1:  [  OK  ]
Shutting down system logger: [  OK  ]
Starting system logger: [  OK  ]
INFO (node:test30): Network configuration completed...

INFO (node:test30): RAC OVM auto buildcluster detected all (2) nodes are
accepting SSH connections, starting build...

INFO (node:test30): RAC OVM auto buildcluster started (-b flag specified) on
all (2) node(s) at Sat May 26 08:40:07 EDT 2012, pid:1774, check build log in
/u01/racovm/buildcluster.log

INFO:Oracle RAC Template: disabled all RAC Templates first boot services as
well as set ovmd to disable as part of initial boot.

INFO:Oracle RAC Template: Successfully configured the network, as well as
initiated a buildcluster...
Press Return to continue (timeout 30)...

Oracle Linux Server release 6.2
Kernel 2.6.32-300.21.1.el6uek.x86_64 on an x86_64

test30 login:
```

It is possible to monitor the buildcluster progress, by logging to the first VM listed and looking at /u01/racovm/buildcluster.log or /u01/racovm/buildsingle.log (based on deployment mode).
This logfile will have all commands executed in verbose mode, so you can see as the various tools, like clone.pl, netca, dbca, emca are executed along with their output.

Default installation specifications for a 2-node cluster:

SID: ORCL1 & ORCL2
DB name: ORCL
Grid Infrastructure Home: /u01/app/11.2.0/grid **or** /u01/app/12.1.0/grid
Oracle RAC Home: /u01/app/oracle/product/11.2.0/dbhome_1 **or** /u01/app/oracle/product/12.1.0/dbhome_1
ORACLE_BASE: /u01/app/oracle
Central Inventory: /u01/app/oraInventory

Sample RAC deployment with Flex Cluster & Flex ASM: The following command will power ON a 4-node Flex Cluster (with Flex ASM) and a dedicated 3rd Network adapter for ASM traffic. The network details will be sent and a buildcluster will be initiated. Notice the detection of node types (Hub/Leaf) as well as the dedicated ASM network interface:

```
$ ./deploycluster.py -u admin -p MyP123 -M racnode.? -N netconf-4flex.ini
INFO: Inspecting /home/oracle/deploycluster/netconf-4flex.ini for number of nodes
defined....
INFO: Detected 4 nodes in: /home/oracle/deploycluster/netconf-4flex.ini

INFO: Located a total of (4) VMs;
      4 VMs with a simple name of: ['racnode.1', 'racnode.2', 'racnode.3','racnode.4']

INFO: Detected (2) Hub nodes and (2) Leaf nodes in the Flex Cluster

INFO: Detected a RAC deployment...

INFO: Starting all (4) VMs...

INFO: VM with a simple name of "racnode.1" (Leaf node) is in a Stopped state,
attempting to start it....OK.

INFO: VM with a simple name of "racnode.2" (Hub node) is in a Stopped state,
attempting to start it........OK.

INFO: VM with a simple name of "racnode.3" (Hub node) is in a Stopped state,
attempting to start it......OK.

INFO: VM with a simple name of "racnode.4" (Leaf node) is in a Stopped state,
attempting to start it...+.OK.

INFO: Verifying that all (4) VMs are in Running state and pass prerequisite checks...

INFO: Detected Flex ASM enabled with a dedicated network adapter (eth2), all VMs will
require a minimum of (3) Vnics...
....

INFO: Detected that all (2) Hub node VMs specified on command line have (5) common
shared disks between them (ASM_MIN_DISKS=5)

INFO: The (4) VMs passed basic sanity checks and in Running state, sending cluster
details as follows:
      netconfig.ini (Network setup): /home/oracle/deploycluster/netconf-4flex.ini
      buildcluster: yes

INFO: Starting to send configuration details to all (4) VM(s).......
INFO: Sending to VM with a simple name of "racnode.1" (Leaf node).....
INFO: Sending to VM with a simple name of "racnode.2" (Hub node)....
INFO: Sending to VM with a simple name of "racnode.3" (Hub node).....
INFO: Sending to VM with a simple name of "racnode.4" (Leaf node)......

INFO: Configuration details sent to (4) VMs...
      Check log (default location /u01/racovm/buildcluster.log) on build VM
(racnode.2)...

INFO: deploycluster.py completed successfully at 22:35:50 in 133.0 seconds (02m:12s)
Logfile at: /home/oracle/deploycluster/deploycluster3.log
```

DeployCluster Flags

Deploycluster has two basic types of flags "Oracle VM Manager login" and "DB/RAC OVM Templates Options". Flags have short (preceded by one dash) and long names (preceded by two dashes), for example the **-M** flag, is equivalent to the **--vms** flag.

Oracle VM Manager login flags

Usage: deploycluster.py **<Oracle VM Manager login>** <DB/RAC Templates Options>

```
Oracle VM Manager Login:
  Credentials to login to Oracle VM Manager (SSL supported)

  -u <username>, --username=<username>
                        Username to connect to Oracle VM Manager
  -p <password>, --password=<password>
                        Password to connect to Oracle VM Manager

Deploycluster v2.*:
  -H <host>, --host=<host>
                        Manager hostname (use either -H or -U or none)
  -U <url>, --url=<url>
                        Login URL to Manager (default:
                        tcp://localhost:54321
                        or tcps://host:54322 when -H used to remote node)

 Deploycluster v3.*:
  -H <host>, --host=<host>
                        Manager hostname, IP address or omit for local
                        host
  -r <port>, --port=<port>
                        Manager WebServices API port number (default:7002)

 Deploycluster v3.* - CERTIFICATE:

  -I, --insecure      Ignore SSL certificate and host name checks – not
                        for production environments
  --keystore-file=<KEYSTOREFILE>
                        The keystore used to store your login key and
                        certificates (default: ~/.ovmkeystore)
  --keystore-password=<KEYSTOREPASS>
                        The password for the keystore
  --key-password=<KEYPASS>
                        The password for the login key (can be omitted if
                        same as keystore password)
```

-u flag: The username to connect to Oracle VM Manager using traditional authentication.

-p flag: Password to connect to Oracle VM Manager. If password is not supplied it may be defined in the environment variable DEPLOYCLUSTER_MGR_PASSWORD, or prompted from the console.

Deploycluster v2.*:

-H flag: (optional) Identifies the node that runs Oracle VM Manager. If not supplied, the -U flag could be used. If node is remote, by default first SSL port (54322) is used, if connection fails the non-SSL (54321) port is used. Oracle VM typically only allows SSL connections to remote nodes, which requires special configurations of OVM Manager[3].

-U flag: (optional) establishes the protocol type, host and port that is used to connect to Oracle VM Manager. If not supplied, the -H flag could be used. Syntax is: prot://host:port. "prot" is the protocol and may be "tcp" (non-secure) or "tcps" for secure connection.

If neither **-H** or **-U** flags are supplied; it is assumed that the Oracle VM Manager runs on the local node.

Note: The default ports may be overridden using EPLOYCLUSTER_OVMAPI_SSL_PORT and DEPLOYCLUSTER_OVMAPI_NONSSL_PORT in deploycluster.ini.

Deploycluster v3.*:

-H flag: Manager hostname, IP address or omit for local host

-r flag: Manager WebServices API port number (default: 7002)

Deploycluster v3.* - CERTIFICATE:

-I flag**:** Ignore SSL certificate and host name checks - not for production environments

--keystore-file=<KEYSTOREFILE>
The keystore used to store your login key and certificates (default: ~/.ovmkeystore)

--keystore-password=<KEYSTOREPASS>
The password for the keystore. If password is not supplied it may be defined in the environment variable DEPLOYCLUSTER_KEYSTOREPASS, or prompted from the console

--key-password=<KEYPASS>
The password for the login key (can be omitted if same as keystore password). This password may be defined in the environment variable DEPLOYCLUSTER_KEYPASS.

[3] See My Oracle Support Note#1456338.1 on how to setup SSL connections for the Manager (Needed for OVM 3.2 only).

DB/RAC Templates Options

Usage: deploycluster.py <Oracle VM Manager login> **<DB/RAC OVM Templates Options>**

```
Oracle DB/RAC OVM Template Options:
  Identify which VMs to deploy as Single Instance or a cluster - pass any
  special build attributes. Commonly used flags: -M, -N & -P.

  -L, --list_vms_only
                        List VMs seen via Oracle VM Manager; Honors -M flag
  -M <List of VMs>, --vms=<List of VMs>
                        List of existing VM names or IDs to deploy cluster on.
                        Supports "*" & "?" wildcard characters
  -P <params.ini>, --params=<params.ini>
                        Location of params.ini file (sent to VMs)
  -N <netconfig.ini>, --netconfig=<netconfig.ini>
                        Location of netconfig.ini file (sent to VMs)
  -B <yes|no>, --buildcluster=<yes|no>
                        Start a buildcluster/buildsingle post-network setup
                        (default: yes.[If netconfig_args passed then default:
                        no])
  -G <args>, --netconfig_args=<args>
                        Advanced: Arguments to netconfig; override defaults
  -K <zip file>, --kitfile=<zip file>
                        Advanced: Unzip new (partial) kitfile inside the VMs
  -X <file>, --extrakeys=<file>
                        Advanced: File containing extra keys to send all VMs
  -D, --dryrun          Show what will be done (do not start VMs or send msgs)
```

-D flag: Indicates "dry run" mode, no messages or start VM commands should be sent, only list what would be done in non-dry run mode.

-L flag: List mode. Will list all VMs that are defined in Oracle VM Manager or only those specified by the **-M** flag. Errors will be generated if duplicate or non-unique VMs are listed, this is by design, to simulate the behavior during a normal cluster deployment.

-M flag: List of VMs to operate on (create cluster, or list details). VMs can be identified by simple names and/or UUIDs. If simple name is used but more than one VM with the same simple name is defined in OVM Manager, an error is generated. In this case a UUID must be used to uniquely identify which VM to operate on. If DEPLOYCLUSTER_VMNAME_HONOR_WILDCARDS is yes (default), then the '*' (matches any number of characters) and "?" (matches any single character) wildcard characters are honored.

-P flag: Path to a params.ini file that will be sent to all VMs, allowing more control of buildcluster options, such as ASM redundancy level, Database name, SID, ports, etc. If this flag is supplied, it overwrites the params.ini inside the guests at path /u01/racovm/params.ini; therefore any settings, such as disk names or discovery string specified should match to the ones that the VMs are configured with.

-N flag: Path to a netconfig.ini file to be sent to all VMs. This file should include the cluster and node details for all cluster members. The VMs will set their network based on the contents of this file.

-B flag: Should a buildcluster or buildsingle be run on the first VM (or first Hub node in the case of Flex Cluster) listed in **-M** flag. By default, it is "yes", unless the **-G** flag is used in which case it defaults to "no". If **-G** flag is used with '-b' and **-B** flag is given to be "no" the one from command line overrides and no buildcluster is performed.

Advanced flags:

-G flag: List of flags to be passed to netconfig first boot. Full details are described in /u01/racovm/netconfig.sh inside the guests. In short they are:
-n<#> : Node number for this node or starting node number
-b[#] : Initiate a buildcluster/buildsingle
-R/W : Read or Write netconfig.ini data from disk (Should only be needed in OVM2)
-c <d> : Disk that holds netconfig.ini details. (Should only be needed in OVM2)

It is also possible to pass the special keyword "yes" or an empty string (""), which indicates the 2-node interview is requested. The "out" keyword can be supplied to skip any network or buildcluster configuration.

-K flag: Path to a kitfile, in zip format, that is sent to all VMs prior to netconfig first-boot configuration or buildcluster operation. If buildcluster is yes, the file is sent only to the build node which will then copy it as part of normal buildcluster operation to all nodes.

-X flag: Path to a file holding key=value pairs that will be sent to all VMs. The keys may be anything, and can be used to configure other subsystems in the VM. These keys are sent after the configuration keys for the RAC templates so may be used to override any auto-calculated key.

DeployCluster Options (deploycluster.ini)

The file deploycluster.ini provides some parameters which can help modify the way **deploycluster** tool operates. For example:

```
# Should each invocation be fully logged. Logs increment each execution
# deploycluster1.log, deploycluster2.log, etc. in current directory.
# Default: YES
DEPLOYCLUSTER_AUTOLOG=yes
#
# Directory location of logfile. Can be relative to current directory
# or full path. Default log location is the current directory.
# Default: Current directory
#DEPLOYCLUSTER_AUTOLOG_DIR=
#
# Allows color coding of log messages, errors (red), warning (yellow),
# info (green). Automated logs will not contain the escape sequences
# used to generate the colors, these are only written to the terminal.
# Default: YES
DEPLOYCLUSTER_LOGWITH_COLORS=yes
#
# Allow '*' & '?' wildcards in VM simple names. This allows
# names like "rac11203-dec23-n*". If set to NO, the '*' and '?' are
```

```
# treated as literals.
# Default: YES
DEPLOYCLUSTER_VMNAME_HONOR_WILDCARDS=yes
```

See the deploycluster.ini file for more details on the available options.

Advanced Deployment Options

Below are some of the more advanced deployment methods.

Sending Extra Keys

First Boot is the time when a newly instantiated Guest VM is started for the first time.
At this point in time the guest machines are identical – they have no "personality" e.g. no hostname, no IP address etc. First boot adds this personality, by configuring the node name, network, etc.

Since the OS images included in the recent DB/RAC OVM Templates are OVMAPI enabled, it is possible to send them basic configurations such as:

'com.oracle.linux.selinux.mode'
'com.oracle.linux.datetime.datetime'
'com.oracle.linux.datetime.timezone'
'com.oracle.linux.datetime.utc'
'com.oracle.linux.datetime.ntp'
'com.oracle.linux.datetime.ntp-servers'
'com.oracle.linux.datetime.ntp-local-time-source'

These keys are discussed fully in the generic OVM and base Linux OS Template documentation. A listing of all possible enabled keys can be obtained using the following command:

ovm-template-config --human-readable --enumerate configure

The **deploycluster** tool is able to easily send any extra keys to the VMs at deploy time, thus allowing adjustments to any other needed OS configuration.

For example, to set SELinux to **enforcing**, create a simple text file with:

com.oracle.linux.selinux.mode=enforcing

When deploying the VMs, simply add the **-X** (eXtra keys) flag and pass this file.

Modifying Build Options

By default, the Single Instance or RAC Cluster build options reside in **/u01/racovm/params.ini** inside the VMs. As shown above, using the **-P** (Capital P) flag it is possible to use a different params.ini than the one supplied with the templates. This allows full control over things like Database name, SID name, port numbers, etc. You do have to ensure that the options and values set in this custom params.ini file match the VMs that will use it. For example, if the params.ini has:

ALLDISKS="/dev/xvdc /dev/xvdd /dev/xvde /dev/xvdf /dev/xvdg /dev/xvdh"

The VMs will have to have these 6 device names, xvdc, xvdd, xvde, xvdf, xvdg & xvdh. If they don't then the buildcluster will fail. To recover from such a failure, either correct params.ini and run

buildsingle.sh/buildcluster.sh manually, or clean all VMs as described in the FAQ, adjust their shared disks as needed then re-deploy using a correct params.ini file.

Changing Oracle Disk location

By default, the Oracle disk is mounted on /u01, and the RAC OVM kit reside in /u01/racovm. If for some reason this path needs to be changed, then edit the file /etc/sysconfig/dbracovm-template.config or /etc/sysconfig/racovm-template.config and modify RACOVM_PATH:

```
# Path to RACOVM OneCommand kit, only change if mounting the kit
# at a different location than the default.
# Default: /u01/racovm
RACOVM_PATH=/u01/racovm
```

Also remember to modify the needed values in params.ini, such as RACROOT. Then clean and save the template for subsequent usage.

APPENDIX A: Build Options

Before invoking **/u01/racovm/buildcluster.sh** (or buildsingle.sh for Single Instance) you may edit **/u01/racovm/params.ini** to modify some build options (bottom part of the file). The top part of params.ini should be modified by advanced users or if instructed to by Oracle Support. If using **deploycluster** tool (OVM3 only), a custom params.ini may be passed using the **-P** flag, it will be sent to all VMs and the buildcluster (or buildsingle) will then use that instead of the shipped params.ini inside the VM. Small samples of the options that may be modified are:

```
# Build Database? The BUILD_RAC_DATABASE will build a RAC database and
# BUILD_SI_DATABASE a single instance database (also in a RAC environment)
# Default: yes
BUILD_RAC_DATABASE=yes
#BUILD_SI_DATABASE=yes

# The Database Name
# Default: ORCL
DBNAME=ORCL

#
# The Instance name, may be different than database name. Limited in length of
# 1 to 8 for a RAC DB & 1 to 12 for Single Instance DB of alphanumeric characters.
# Ignored for Policy Managed DB.
# Default: ORCL
SIDNAME=ORCL

# Configures a Single Instance environment, including a database as
# specified in BUILD_SI_DATABASE. In this mode, no Clusterware or ASM will be
# configured, hence all related parameters (e.g. ALLDISKS) are not relevant.
# The database must reside on a filesystem.
# This parameter may be placed in netconfig.ini for simpler deployment.
# Default: no
#CLONE_SINGLEINSTANCE=no

# Configures a Single Instance/HA environment, aka Oracle Restart, including
# a database as specified in BUILD_SI_DATABASE. The database may reside in
# ASM (if RACASMGROUPNAME is defined), or on a filesystem.
# This parameter may be placed in netconfig.ini for simpler deployment.
# Default: no
#CLONE_SINGLEINSTANCE_HA=no

# Local Listener port number (default 1521)
# Default: 1521
LISTENERPORT=1521

# Allows color coding of log messages, errors (red), warning (yellow),
# info (green). By default no colors are used.
# Default: NO
CLONE_LOGWITH_COLORS=no
```

If you do not wish to store the passwords for the root or Oracle user in the configuration file, remove or comment them, and they will be prompted for at the start of the build.

APPENDIX B: Adding More Nodes (Oracle VM3)

Oracle VM 2 users should use the method described in the document that ships with the templates in Appendix B.

The steps below only work with the new **deploycluster** tool on OVM3.

It is possible to add any number of nodes to an existing cluster at any time following these steps (In the example we assume 2 nodes already exist in the cluster):

1. Follow step 4 in this document to create a 3rd or 4th VM, using Oracle VM Manager.

2. Follow step 5 to attach the same shared storage as the existing cluster nodes have.

3. Edit the **netconfig.ini** from step 7 (or copy it from any of the existing cluster members from /u01/racovm directory to the node you plan on running deploycluster from) and add the information about the new VMs, e.g:

   ```
   NODE3=racnode3
   NODE3IP=192.168.1.205
   NODE3PRIV=racnode3-priv
   NODE3PRIVIP=10.10.10.205
   NODE3VIP=racnode3-vip
   NODE3VIPIP=192.168.1.206
   ```

   This file is used in next step and may be renamed to any desired name, e.g. netconf3nodes.ini

4. Invoke the **deploycluster** tool to boot and setup the network on the new VM(s), as follows:

   ```
   $ ./deploycluster.py -u admin -p MyP123 -M racnode.2 -N netconf3nodes.ini
      -G '-n3'
   ```

   Note that the **-G** flag is used to indicate that the VM named 'racnode.2' should be configured as the 3rd node from the supplied netconfig.ini file. As mentioned earlier, anytime the **-G** flag is used, the buildcluster option is automatically set to NO, so the new node will configure its network but not attempt to run buildcluster. This is equivalent to somehow copying the netconfig.ini file into the VM, and then running "# /u01/racovm/netconfig.sh -n3" inside it. However, the tricky part is that there is no network setup on a VM during firstboot, and that's where the **deploycluster** tool comes handy.

5. Now that the network is up on the new node, copy the more up-to-date **/u01/racovm/netconfig.ini** from the new node[4] (it should contain all newly added nodes as well as existing nodes) to any existing cluster member (e.g. racnode1) where you plan to run the addnode(s) procedure from.

   # scp netconfig.ini racnode1:/u01/racovm

6. Finally, run the addnode(s) procedure from the cluster member (e.g. racnode1) you copied the updated netconfig.ini to:

[4] It is also possible to copy the netconfig.ini from the node where the deploycluster was run from

```
# cd /u01/racovm
# ./racovm.sh -S addnodes -N racnode3   2>&1 | tee addnode3.log
```

The "2>&1 | tee " means save stdout and stderr to addnode3.log, this is useful since errors are printed to stderr, so using only "tee" or ">" will only capture stdout without any possible errors. If you do not need logging you may omit all of that.

It is possible to add several nodes in one command invocation, just separate them with a comma. By default, new database instance(s) will not be created on the new node(s), if that is required, add the step "addinstances" to the above command, or run it separately at a later time. Here is the sample command to add 2 nodes with their instances:

```
# ./racovm.sh -S addnodes,addinstances -N racnode3,racnode4   2>&1 | tee addnode-inst-node3-4.log
```

Or, to just add instances on a new node (assuming it was already added using addnodes as described above):

```
# ./racovm.sh -S addinstances -N racnode3   2>&1 | tee addinstances-node3.log
```

APPENDIX C: Using More (or less) Disks

Advanced users may want to use differently named, less than or more than 5 disks for database/ASM.

Before the Build:

There is an option to use differently named or more than 5 disks before the build. If you attached more than 5 shared devices to the VMs, before you run **buildcluster.sh** or **buildsingle.sh**, edit **/u01/racovm/params.ini** and modify **ALLDISKS** to contain the list of all devices or partitions that should be used by the database/ASM. The **params.ini** file describes the rules for setting this parameter. Remember that whatever disks/partitions are specified should be discoverable by **RACASMDISKSTRING** in the **params.ini** file who's default is "/dev/xvd[c-g]1". Do not set the discovery string to a too wide value, e.g. /dev/*, the udev rule is written based on this string, and so might affect devices you did not intend on affecting.

As an example – if you add a 6th device: /dev/xvdh.

ALLDISKS="/dev/xvdc /dev/xvdd /dev/xvde /dev/xvdf /dev/xvdg /dev/xvdh"

The **RACASMDISKSTRING** should be:

RACASMDISKSTRING="/dev/xvd[c-h]1" (Discovers xvdc1, xvdd1, through xvdh1)

WRONG: RACASMDISKSTRING="/dev/xvd?1" (Since it also discovers xvda1, xvdd1 which are the /u01 and /boot mount points by default)

The discovery string may have multiple entries separated by space.

By default, the build procedure will automatically partition all the disks with 1 partition each. An already partitioned disk, e.g. /dev/xvdc3 will be used as-is. It will also write a new UDEV rule, default rule file is: /etc/udev/rules.d/99-oracle.rules.

NOTE: If you want to use less than the default and recommended 5 disks (for example 3), set ASM_MIN_DISKS=3 in params.ini.

After making the changes to params.ini, invoke **deploycluster** and supply the **-P** flag along with the modified params.ini. This will be sent to all VMs and they will use that file instead of the default one that is supplied with the templates.

After the build:

After the build, simply add more disks to the VMs, partition them correctly and add them to ASM as you normally would. If you follow this route remember to modify the UDEV rules file on all nodes to give the new disk the correct permissions. Default rules file is: /etc/udev/rules.d/99-oracle.rules.

APPENDIX D: Using Text Mode Console (Oracle VM3)

Dom0 access is not supported in OVM3; hence users should connect to VM's console from the Manager UI.

Using the **deploycluster** tool reduces the need for console access altogether since automated deployment is possible remotely, without ever logging to any VM.

APPENDIX E: Securely deploying the RAC Templates (Oracle VM3)

In order to deploy the DB/RAC Templates using the **deploycluster** tool in the most secure way, the following guidelines should be followed.

1. When running the **deploycluster** tool, do not use the **-p** flag for the Manager password (as the password will be stored in the command history of the shell). Instead let the **deploycluster** tool prompt for the Manager password. Make sure you run it from a terminal so that no echo prompting will function correctly. If run through "ssh" add the -t flag, so that a terminal is allocated to the ssh session and password could be prompted correctly (with echo off). If you need to automate the deployment you may use the environment variable DEPLOYCLUSTER_MGR_PASSWORD. In this case, remember to unset it as soon as the deployment is completed as well as clear the shell's command history, e.g. history -c.

2. Do not keep any password in params.ini, such as ROOTUSERPASSWORD, RACPASSWORD or GRIDPASSWORD. Comment these by placing "#" in front of them in which case[5] the **deploycluster** tool will prompt for and send them to the VMs. If the connection protocol to **Oracle VM Manager 3.2 and below** is non-secure (e.g. tcp), the following warning is printed and users should continue at their own risk:

```
WARNING: Login protocol (tcp) to Oracle VM Manager is non-secure (non-SSL),
password prompting continues at your own risk...

INFO: Found that root's password does not exist in parameter file or
environment (ROOTUSERPASSWORD), prompting...
Password:
```

   It is therefore strongly recommended to follow the procedure to enable secure connections to Oracle VM Manager 3.2 and below. See the Oracle VM documentation as well as My Oracle Support Note#1456338.1 on how to configure SSL connection to the Oracle VM Manager. If a secure connection to Oracle VM Manager is detected the following notice is printed instead:

```
INFO: Login protocol (tcps) to Oracle VM Manager is secure, password
prompting continues...
```

   In this case, the passwords are sent via SSL to Oracle VM Manager. In all cases, the connection from the Manager to the Oracle VM Servers (VMs) is via SSL/secure connection. Oracle VM 3.3 and above uses SSL exclusively without any extra configuration required.

   Inside the VMs, the passwords are read from kernel memory (OVMAPI), not written to any file and immediately discarded. They are used for the initial buildcluster/buildsingle (if chosen to run), however, any subsequent operation, will prompt for them if they are missing from params.ini

   In order to use an older Oracle Template with the **deploycluster** tool, simply combine it with a newer OS disk (one that was released in 2012 or after) which has OVMAPI support. In such a configuration (older Oracle disk and newer OS disk), in order for the password passing to function correctly, when running the **deploycluster** tool, add **'-K utils/netconfig.zip'** to the command line (the netconfig.zip resides in the utils directory). This will send an updated netconfig.sh to the VM and ensure the password passing works correctly.

[5] Password prompting only takes place if the password variables are commented out in params.ini.

APPENDIX F: Troubleshooting / Frequently Asked Questions

For general DB/RAC Template FAQs, please refer to the PDF that is bundled with the templates and also resides on the main DB/RAC OVM Template page on OTN.

Below are FAQs specific to the **deploycluster** tool.

1) Deploycluster declared that a VM failed basic checks, how do I proceed?

Assuming this failure is printed:

```
ERROR: VM with a simple name of "racnode.0" has insufficient shared disks; only
(3) shared disks attached to it. Current setting of ASM_MIN_DISKS is (5),
therefore expecting to detect at least (5) shared disks
```

Simply correct the condition by adding more disks to the VM or change ASM_MIN_DISKS to 3 in params.ini and specify it on the command line using the **-P** flag; then re-run the same command. Similarly, if this is shown:

```
ERROR: VM with a simple name of "racnode.1" has only (1400MB) of memory, a
minimum of 1700MB is required (test configuration) for Oracle RAC
installation... (to disable check set DEPLOYCLUSTER_SKIP_VM_MEMORY_CHECK=yes)
```

Add more memory as per minimum requirements, or set `DEPLOYCLUSTER_SKIP_VM_MEMORY_CHECK=yes` to disable this check (not recommended). Most cases of early errors do not require any sort of cleanup/power off of the VMs, simply fix what is wrong, and re-run the deploycluster command.

2) Is it OK to manually start the VMs before running the deploycluster tool?

Yes, it is OK to do that although not needed since the tool will automatically start all VMs. It may be useful to start manually to see the console messages during boot, or maybe override some of the OVMAPI keys by manually typing them at the console, although this should not be needed, instead use the **-X** flag to pass extra keys to the VMs. This allows for automated ntp, SELinux setup, etc.

3) The Oracle VM Manager I have is version 3.0, during deployment the following incompatibility notice is shown, is this OK?

```
INFO: Attempting to connect to Oracle VM Manager...
Warning: Minor Version Incompatibility
INFO: Incompatibility (MINOR_INCOMPATIBILITY) detected during connection (safe
to ignore)...
INFO: Oracle VM Client  (3.1.1.305) protocol (1.8) CONNECTED (tcp) to
      Oracle VM Manager (3.0.3.150) protocol (1.7) IP (10.165.244.227) UUID
(0004fb00000100006abbccf3e6445425)
```

Yes, this is normal when connecting to an older Manager, as the notice says, it is safe to ignore.

4) I'm running deploycluster from a server that is not the Manager server, what should be taken into account?

This answer applies to Oracle VM 3.2 and lower:
When connecting to the Manager from a remote server, typically the Manager will require that the secure (tcps) port (54322) be used rather than the default insecure (tcp) port (54321). This requires special configuration on the Manager node, and is described in OVM Documentation as well as My Oracle Support **Note#1456338.1**. If such configuration is not performed to enable the secure port on the Manager, then only local connections, directly on the Manager server, are possible. In that case, the **deploycluster** tool should be run from the Manager node.

If the **-H** flag is used, and the node is a remote node, then **deploycluster** will first attempt to connect on port 54322 (secure), if the connection fails, an attempt on the non-secure 54321 port will be made before an error is reported to the user.

To get full control over the protocol/port, one should use the **-U** flag, e.g. tcps://mgrhost:54333, this will use the secure protocol (tcps), on Manager host "mgrhost" and port 54333.

The default secure (54322) & non-secure (54321) port numbers may be overridden in deploycluster.ini.

5) If the VMs pass basic checks and were started successfully and still the deployment is unsuccessful for any reason, where should I look?

Start looking in the VMs at the following log files:

/u01/racovm/buildcluster.log or **buildsingle.log**	(only on the build node, first (hub node) VM listed)
/var/log/messages	(Look for "Oracle DB/RAC Template" messages)
/var/log/ovm-template-config-*rac-configure.log	(DB/RAC Template Config/first-boot logfile)
/var/log/ovm-template-config.log	(Generic OVM Template Config/first-boot logfile)

For example, a typical failure could be non-empty disks used for ASM. As a safeguard buildcluster will fail if existing ASM data is seen on the disks with messages similar to:

```
INFO (node:test3): Specified disk (/dev/xvdc) in ALLDISKS that will
automatically be partitioned and renamed to (/dev/xvdc1) appears to be an
active ASM disk: DATA_0000 Failgroup: DATA_0000 in Diskgroup: DATA
```

In this specific case, the corrective action would be to make sure the disks are the correct ones, and then clear them using any of the following methods:

# /u01/racovm/racovm.sh -S cleanlocal	(run from first node)
OR:	
# /u01/racovm/cleanlocal.sh -X	(run from first node)
OR:	
# /u01/racovm/diskconfig.sh -X	(run from first node)
OR:	
# /u01/racovm/racovm.sh -S clean	(run from any node; first node must be up)

WARNING: The above commands are **destructive** in that they will wipe any data written to the (shared) disks, as well as the 'clean' ones will also remove any installed RAC & Grid Infrastructure software from the local or all nodes -- use with caution!

Other common failures include specifying disks in params.ini that do not exist in the VM, or the RACASMDISKSTRING is set to a value which does not discover all disks on all nodes.

Additionally, specifying IP addresses that are already taken or wrong subnet mask for the network may result in failures. Follow the cleanup procedure described in the following question and re-attempt the deployment.

6) Deploy attempt failed, how do I cleanup and start again?

Depending on the stage at which the **deploycluster** failed, if the network was set up right, it is not needed to re-run deploycluster. It is typically enough to correct the error (clear data disk from former ASM headers, or change conflicting IP), then simply re-run **buildcluster.sh** or **buildsingle.sh** (see FAQ#2 in the template's documentation).

If however a complete clean of all the VMs is required to re-run deploycluster, follow the steps below:
To clear shared disks (might be needed; use with caution; FAQ#5 has other disk cleanup options) run:

```
# /u01/racovm/racovm.sh -S setsshroot,clean
```

Delete all System, network and DB/RAC software configuration (might be needed):

```
# ovmd -s cleanup
```

Enables all DB/RAC Template services for next boot (enables first-boot, deploycluster can then be used):

```
OL5/OL6: # service dbracovm-template.init[6] enable-all
    OL7: # /usr/bin/dbracovm-template.init enable-all
```

Shuts down the VM, it will be restarted by next deployment attempt:

init 0 (Or shutdown from Oracle VM Manager)

NOTE: Although the "ovmd -s cleanup" internally calls /u01/racovm/cleanlocal.sh, it does not clear the shared disks, this is a designed safeguard. If shared disk cleanup is required it must be done **before** calling "ovmd -s cleanup" as shown above, because after 'ovmd -s cleanup' the VM's host name will be set to 'localhost' and the cleanlocal.sh procedure will correctly refuse to run on an unknown host (one that is not defined in netconfig.ini).
To run this operation on an N-node cluster, the **doall.sh** utility might be run as follows:

 # /u01/racovm/doall.sh -sp -L last "ovmd -s cleanup; service dbracovm-template.init[6] enable-all; init 0"

This command will run on all nodes (and the local node LAST), it will clear the template configuration by running ovmd -s cleanup, then enable all DB/RAC first boot service, and shutdown the node. As a

[6] In the original "RAC OVM Templates" (without Single Instance Support, released prior to 2013), the service is called **racovm-template.init**

result, the entire cluster will be cleaned and shutdown (ready for the next deploycluster attempt), hence **extreme caution should be used.**

7) A deployment attempt failed and seems related to an OVMAPI exception, but only the first 4 lines of the exception are printed, is this normal?

Yes, this is by design. For example, attempting to start a VM on a server with insufficient memory will fail with a similar error:

```
ERROR: Failed to start VM with a simple name of (racnode.1) (first 4 lines
of exception; see output below):
com.oracle.ovm.mgr.api.exception.RuleException: OVMRU_005007E Virtual
Machine: racnode.1, cannot be started/resumed on Server: server15. Reasons:
Memory required: 1.758 GB; Memory available: 1.040 GB, on Server: server15

Sat Mar 24 08:07:51 PDT 2012
```

Typically the first few lines are enough to understand the reason for the failure, however, if more lines are needed, simply edit deploycluster.ini and change DEPLOYCLUSTER_EXCEPTION_LINES.

8) During first invocation of the deploycluster tool these files may appear in the current directory:

racovmvar.pyc racovmcom.pyc racovmvar$py.class racovmcom$py.class

This is normal and how j/python work, there is no need to remove them, although no harm is done if they are removed, they will again be created on the next invocation.

9) I tried to use the deploycluster tool on a VM that was created from a template released prior to 2012, however always getting the 2-node interview, why?

The **deploycluster** tool can automate DB/RAC deployments only with an OS disk that has the OVMAPI support, as such, the OS disk must be from a template that is released in 2012 or afterwards. Failing to use a newer OS disk will present the 2-node interview as a fallback boot option. In order to use an older Oracle template, simply combine it with a newer OS disk (one that was released in 2012 or after). If using an older Oracle disk, in order for the password passing to function correctly, when running the **deploycluster** tool, add **'-K utils/netconfig.zip'** to the command line (the netconfig.zip resides in the utils directory). This will send an updated netconfig.sh to the VM and ensure the password passing works correctly.

10) What are the considerations for connecting to older or newer Oracle VM Managers?

There are **two** deploycluster tools released as follows, deploycluster version 3.* that supports any Oracle VM Manager 3.3 **or above**, and deploycluster version 2.* that supports any Oracle VM Manager 3.2 **or lower**. Attempting to use the wrong version will print a message to that effect.

Further, when using older versions of the Manager (along with deploycluster version 2.*), consider that by default Deploycluster uses the latest version client to connect, however, if an incompatibility (older Manager) is detected an attempt to automatically re-connect using an older client is made and usually works fine. This re-connection takes a bit more time, so if you know all the environments use an older

Manager; that is lower than 3.0.3.486 or 3.1.1.521 then you may set DEPLOYCLUSTER_DEFAULT_CONNECT_311 to yes and have the older client be used right away.

11) Single Instance deployment fails with older templates, why?

Older templates, named OVM_OL5U8_X86_64_11202**RAC**_PVM do not have the Single Instance logic built into them. Only templates with the name "DBRAC" support both Single Instance and RAC deployments.

12) Can Deploycluster be used to deploy Oracle 12c Release 2 templates? 18c?

YES! Deploycluster can be used to deploy all versions of the Oracle VM Database Templates **released** since Oracle 11gR2 and higher, including Oracle 12c (R1 & R2) as well as Oracle 18c; on any OL5, OL6, OL7 on all supported Oracle VM 3 versions subject to support Matrix (e.g. 18c is not supported on OL5). If sending a params.ini (using the -P flag) be sure to send the correct one based on the actual release used (11gR2, 12cR1, 12cR2, etc.).