Bright Cluster Manager 7.3

OpenStack Deployment Manual

Revision: 9c1a4a5

Date: Fri Sep 6 2024

Trademarks

Linux is a registered trademark of Linus Torvalds. PathScale is a registered trademark of Cray, Inc. Red Hat and all Red Hat-based trademarks are trademarks or registered trademarks of Red Hat, Inc. SUSE is a registered trademark of Novell, Inc. PGI is a registered trademark of NVIDIA Corporation. FLEXlm is a registered trademark of Flexera Software, Inc. ScaleMP is a registered trademark of ScaleMP, Inc. All other trademarks are the property of their respective owners.

Rights and Restrictions

All statements, specifications, recommendations, and technical information contained herein are current or planned as of the date of publication of this document. They are reliable as of the time of this writing and are presented without warranty of any kind, expressed or implied. Bright Computing, Inc. shall not be liable for technical or editorial errors or omissions which may occur in this document. Bright Computing, Inc. shall not be liable for any damages resulting from the use of this document.

Limitation of Liability and Damages Pertaining to Bright Computing, Inc.

The Bright Cluster Manager product principally consists of free software that is licensed by the Linux authors free of charge. Bright Computing, Inc. shall have no liability nor will Bright Computing, Inc. provide any warranty for the Bright Cluster Manager to the extent that is permitted by law. Unless confirmed in writing, the Linux authors and/or third parties provide the program as is without any warranty, either expressed or implied, including, but not limited to, marketability or suitability for a specific purpose. The user of the Bright Cluster Manager product shall accept the full risk for the quality or performance of the product. Should the product malfunction, the costs for repair, service, or correction will be borne by the user of the Bright Cluster Manager product. No copyright owner or third party who has modified or distributed the program as permitted in this license shall be held liable for damages, including general or specific damages, damages caused by side effects or consequential damages, resulting from the use of the program or the un-usability of the program (including, but not limited to, loss of data, incorrect processing of data, losses that must be borne by you or others, or the inability of the program to work together with any other program), even if a copyright owner or third party had been advised about the possibility of such damages unless such copyright owner or third party has signed a writing to the contrary.

Table of Contents

Preface

Welcome to the *OpenStack Deployment Manual* for Bright Cluster Manager 7.3.

0.1 About This Manual

This manual is aimed at helping cluster administrators install, understand, configure, and manage basic OpenStack capabilities easily using Bright Cluster Manager. The administrator is expected to be reasonably familiar with the *Administrator Manual*.

0.2 About The Manuals In General

Regularly updated versions of the Bright Cluster Manager 7.3 manuals are available on updated clusters by default at `/cm/shared/docs/cm`. The latest updates are always online at `http://support.brightcomputing.com/manuals`.

- The *Installation Manual* describes installation procedures for a basic cluster.
- The *Administrator Manual* describes the general management of the cluster.
- The *User Manual* describes the user environment and how to submit jobs for the end user.
- The *Cloudbursting Manual* describes how to deploy the cloud capabilities of the cluster.
- The *Developer Manual* has useful information for developers who would like to program with Bright Cluster Manager.
- The *OpenStack Deployment Manual* describes how to deploy OpenStack with Bright Cluster Manager.
- The *Big Data Deployment Manual* describes how to deploy Big Data with Bright Cluster Manager.
- The *UCS Deployment Manual* describes how to deploy the Cisco UCS server with Bright Cluster Manager.
- The *Machine Learning Manual* describes how to install and configure machine learning capabilities with Bright Cluster Manager.

If the manuals are downloaded and kept in one local directory, then in most pdf viewers, clicking on a cross-reference in one manual that refers to a section in another manual opens and displays that section in the second manual. Navigating back and forth between documents is usually possible with keystrokes or mouse clicks.

For example: `<Alt>-<Backarrow>` in Acrobat Reader, or clicking on the bottom leftmost navigation button of xpdf, both navigate back to the previous document.

The manuals constantly evolve to keep up with the development of the Bright Cluster Manager environment and the addition of new hardware and/or applications. The manuals also regularly incorporate customer feedback. Administrator and user input is greatly valued at Bright Computing. So any comments, suggestions or corrections will be very gratefully accepted at `manuals@brightcomputing.com`.

0.3 Getting Administrator-Level Support

If the reseller from whom Bright Cluster Manager was bought offers direct support, then the reseller should be contacted.

Otherwise the primary means of support is via the website `https://support.brightcomputing.com`. This allows the administrator to submit a support request via a web form, and opens up a trouble ticket. It is a good idea to try to use a clear subject header, since that is used as part of a reference tag as the ticket progresses. Also helpful is a good description of the issue. The followup communication for this ticket typically goes via standard e-mail. Section 11.2 of the *Administrator Manual* has more details on working with support.

0.4 Getting Professional Services

Bright Computing normally differentiates between professional services (customer asks Bright Computing to do something or asks Bright Computing to provide some service) and support (customer has a question or problem that requires an answer or resolution). Professional services can be provided after consulting with the reseller, or the Bright account manager.

1

Quickstart Installation Guide For OpenStack

This quickstart chapter describes, step-by-step, a basic and quick installation of OpenStack for Bright Cluster Manager on a cluster that is already running Bright Cluster Manager. Unlike in the main installation chapter (Chapter 3), the quickstart gives very little explanation of the steps, and is more of a recipe approach. Following these steps should allow a moderately experienced cluster administrator to get an operational OpenStack cluster up and running in a fairly standard configuration as quickly as possible. This would be without even having to read the introductory Chapter 2 of this manual, let alone any of the rest of the manual.

The quickstart chapter ends with section 1.4. This covers tasks to check OpenStack-related functions of the cluster are working as expected.

1.1 Hardware Specifications

The hardware specifications suggested in this quickstart are a minimum configuration. Less powerful hardware is not guaranteed to work with Bright OpenStack.

The minimum number of nodes required to create an OpenStack cluster is 3:

- one head node
- one controller/network node
- and one hypervisor node.

The minimal hardware specifications for these node types are indicated by the following table:

Node Type	CPUs	RAM/GB	Hard Drive/GB	NICs
Head	4	8	40	2 *
Controller	4	8	80	2 *
Hypervisor	4	8	80	1 **

* 2 NICs, one of them connected to the switch where the other compute nodes will be connected and the other is connected to the external world through which it can access the Internet.

** 1 NIC connected to the switch where the other compute nodes will be connected.

Diagram goes here:

1.2 Prerequisites

The starting point of the quickstart installation for Bright OpenStack requires an up and running Bright Cluster Manager. A quickstart on how to set up Bright Cluster Manager is given in Chapter 1 of the *Installation Manual* (http://support.brightcomputing.com/manuals/7.3/installation-manual.pdf)

The head node must have access to the base distribution repositories and to the Bright repositories. This is because cm-openstack-setup—a utility used in section 1.3—must be able to install packages from these repositories. The head node must therefore be connected to the internet, or it must be able to access a local mirror of both repositories.

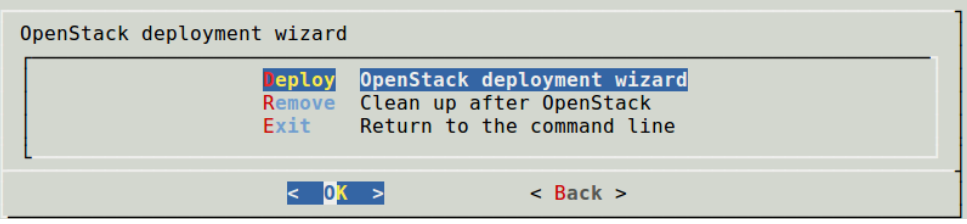
1.3 Installing Bright OpenStack Using cm-openstack-setup

The cm-openstack-setup script is run from the head node and deploys an OpenStack instance. An example session is shown next. This example is based on using node001 as the controller node, and node002 as the hypervisor node:

```
[root@bright73 ~]# cm-openstack-setup
Please wait
Connecting to CMDaemon
```

If all is well, then a deployment screen is seen. The steps are then:

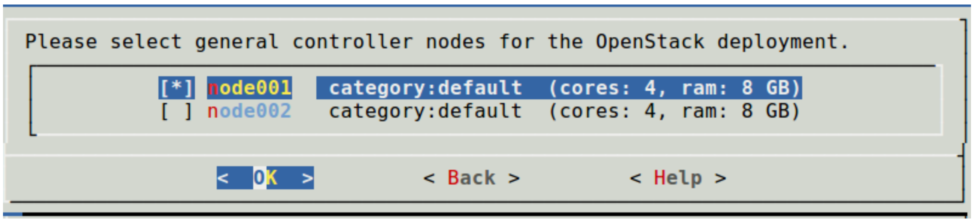
1. Select the Deploy option from the deployment screen (figure 1.1):



```
OpenStack deployment wizard

          Deploy  OpenStack deployment wizard
          Remove  Clean up after OpenStack
          Exit    Return to the command line

          <  OK  >          < Back >
```

Figure 1.1: Deployment Screen

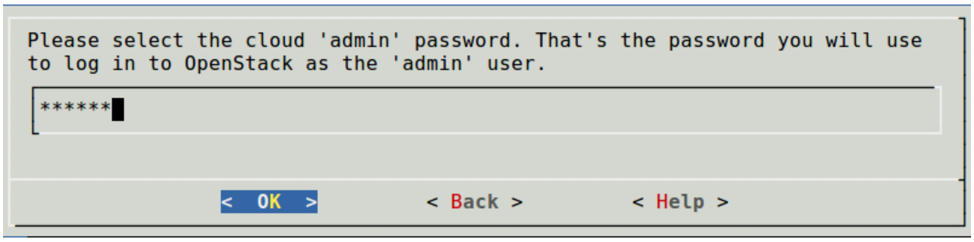
2. Select node001 as the controller node.(figure 1.2):



```
Please select general controller nodes for the OpenStack deployment.

          [*] node001   category:default  (cores: 4, ram: 8 GB)
          [ ] node002   category:default  (cores: 4, ram: 8 GB)

       <  OK  >        < Back >        < Help >
```

Figure 1.2: Setting the controller nodes

3. Set a password for the admin user (figure 1.3):

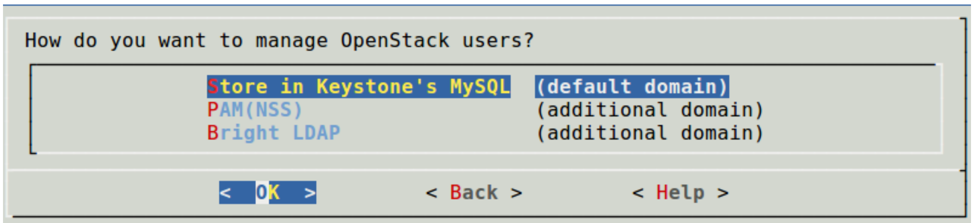


```
Please select the cloud 'admin' password. That's the password you will use
to log in to OpenStack as the 'admin' user.

******

          <  OK  >          < Back >          < Help >
```

Figure 1.3: Setting The admin Password

The admin user is an OpenStack user who is to be created and who is to be given administrator privileges in the OpenStack instance that is being created by the wizard. The admin user can login to the OpenStack Horizon (an administrative dashboard) when OpenStack is running.

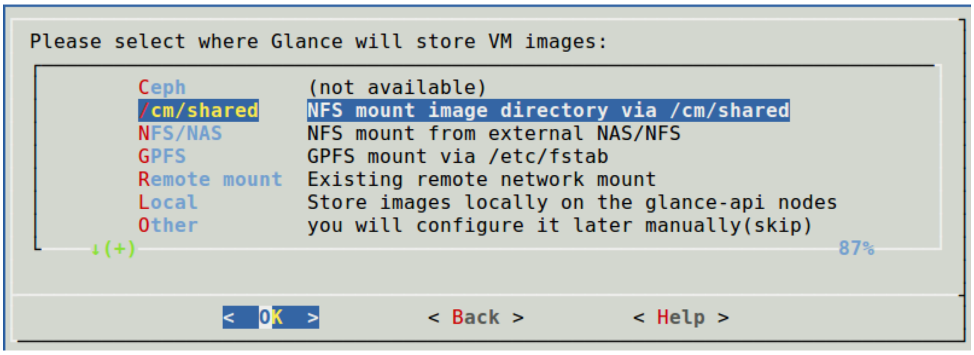
4. Set OpenStack users to be stored in Keystone's MySQL (figure 1.4):



```
How do you want to manage OpenStack users?

                Store in Keystone's MySQL  (default domain)
                PAM(NSS)                   (additional domain)
                Bright LDAP                (additional domain)

          <  OK  >          < Back >          < Help >
```

Figure 1.4: Configuring OpenStack users to be stored within Keystone's MySQL database

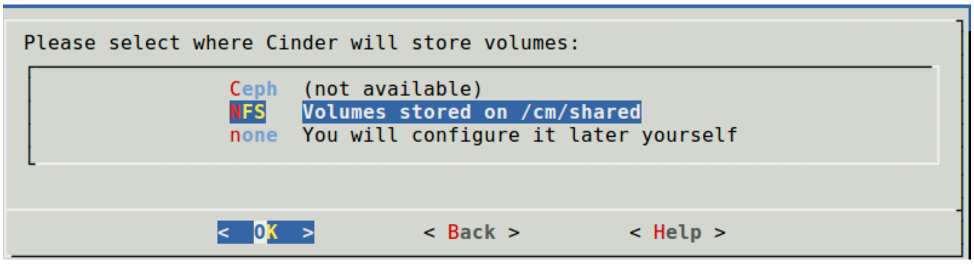
5. Set /cm/shared for Glance (images) storage (figure 1.5):



```
Please select where Glance will store VM images:

         Ceph            (not available)
         /cm/shared      NFS mount image directory via /cm/shared
         NFS/NAS         NFS mount from external NAS/NFS
         GPFS            GPFS mount via /etc/fstab
         Remote mount    Existing remote network mount
         Local           Store images locally on the glance-api nodes
         Other           you will configure it later manually(skip)
    ↓(+)                                                        87%

          <  OK  >          < Back >          < Help >
```

Figure 1.5: Configuring Glance (image) storage

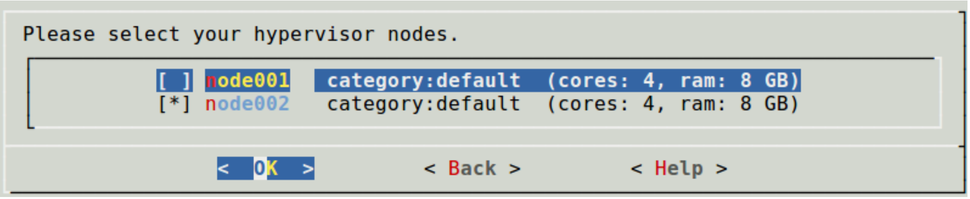
6. Set NFS for Cinder (volume) storage (figure 1.6):



```
Please select where Cinder will store volumes:

                Ceph  (not available)
                NFS   Volumes stored on /cm/shared
                none  You will configure it later yourself

          <  OK  >          < Back >          < Help >
```

Figure 1.6: Configuring Cinder (volume) storage

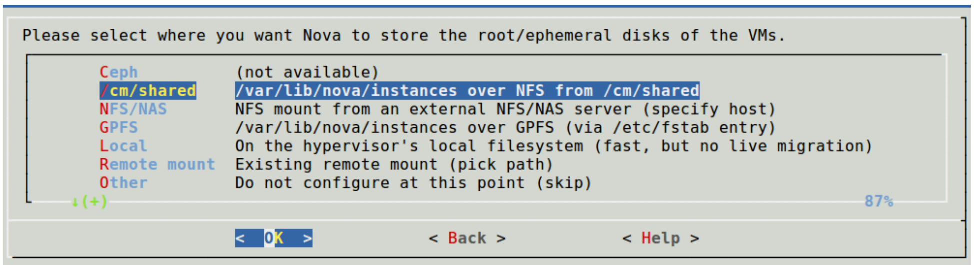
7. Select node002 as the hypervisor node (figure 1.7):



```
Please select your hypervisor nodes.

          [ ] node001   category:default  (cores: 4, ram: 8 GB)
          [*] node002   category:default  (cores: 4, ram: 8 GB)
```

Figure 1.7: Configuring the hypervisor nodes

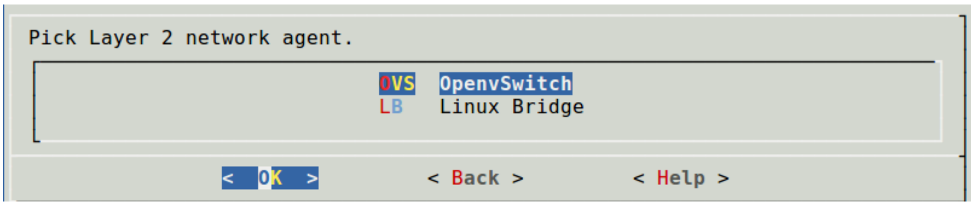
8. Set `/cm/shared` for Nova (virtual machines) storage (figure 1.8):



```
Please select where you want Nova to store the root/ephemeral disks of the VMs.

     Ceph          (not available)
     /cm/shared    /var/lib/nova/instances over NFS from /cm/shared
     NFS/NAS       NFS mount from an external NFS/NAS server (specify host)
     GPFS          /var/lib/nova/instances over GPFS (via /etc/fstab entry)
     Local         On the hypervisor's local filesystem (fast, but no live migration)
     Remote mount  Existing remote mount (pick path)
     Other         Do not configure at this point (skip)
  ↓(+)                                                                       87%
```

Figure 1.8: Configuring the Nova virtual machine disk storage

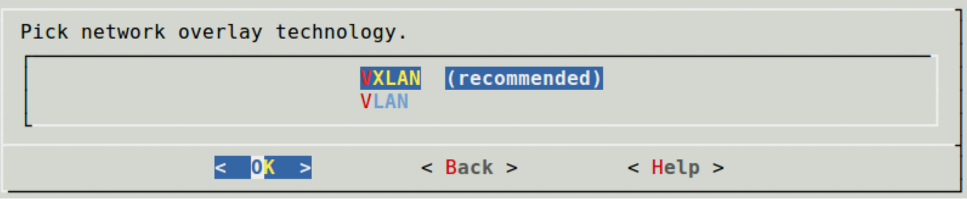
9. Set OpenvSwitch as the layer 2 network agent (figure 1.9):



```
Pick Layer 2 network agent.

                     OVS  OpenvSwitch
                     LB   Linux Bridge
```

Figure 1.9: Setting OpenvSwitch as the layer 2 network agent

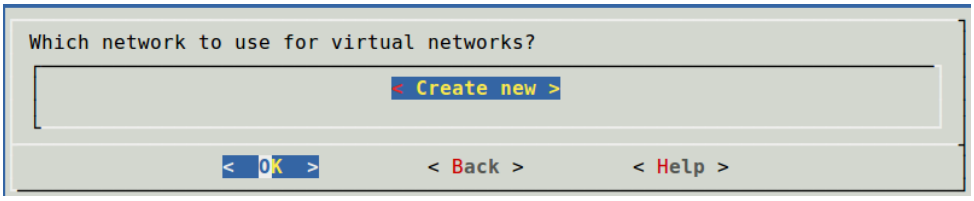
10. Set VXLAN as the network overlay technology (figure 1.10):



```
Pick network overlay technology.

                    VXLAN  (recommended)
                    VLAN
```

Figure 1.10: Setting VXLAN as the network overlay technology

11. Select the `<Create new>` option to create a new network for virtual networks in the OpenStack cluster (figure 1.11):

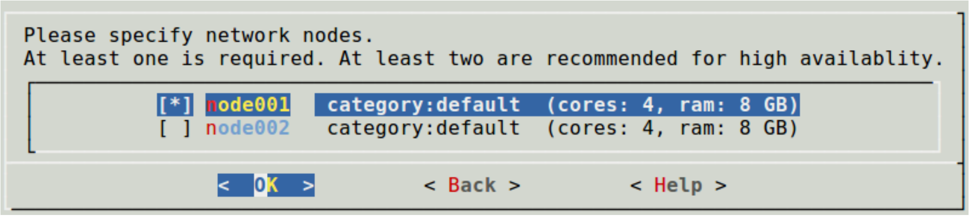


```
Which network to use for virtual networks?

                      < Create new >
```

Figure 1.11: Configuring the creation of a new network for virtual networks

The default values for the new network can be accepted.

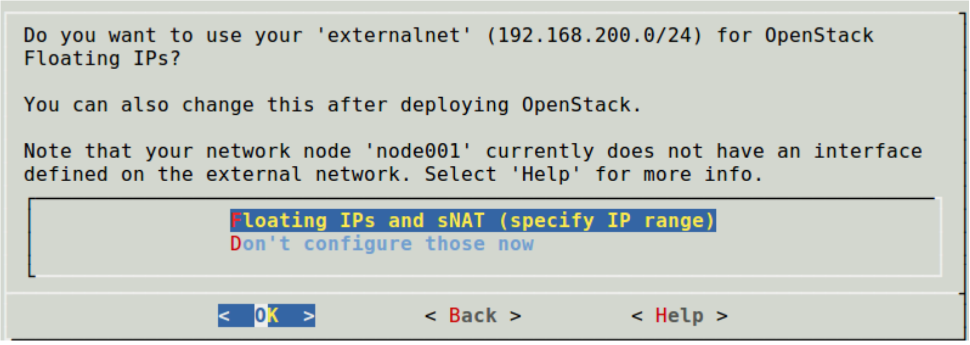
12. The OpenStack controller node can also be a network node. The controller node `node001` is selected to be a network node as well for this example (figure 1.12):



```
Please specify network nodes.
At least one is required. At least two are recommended for high availablity.

        [*] node001   category:default  (cores: 4, ram: 8 GB)
        [ ] node002   category:default  (cores: 4, ram: 8 GB)
```

Figure 1.12: Setting the network nodes

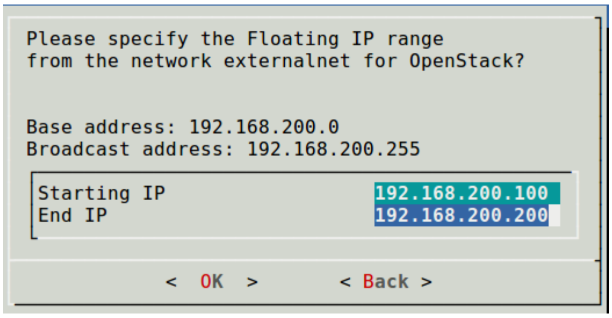
13. Floating IP addresses and sNAT should be selected for the external network (figure 1.13):



```
Do you want to use your 'externalnet' (192.168.200.0/24) for OpenStack
Floating IPs?

You can also change this after deploying OpenStack.

Note that your network node 'node001' currently does not have an interface
defined on the external network. Select 'Help' for more info.

          Floating IPs and sNAT (specify IP range)
          Don't configure those now
```

Figure 1.13: Configuring floating IP addresses to be used on the external network

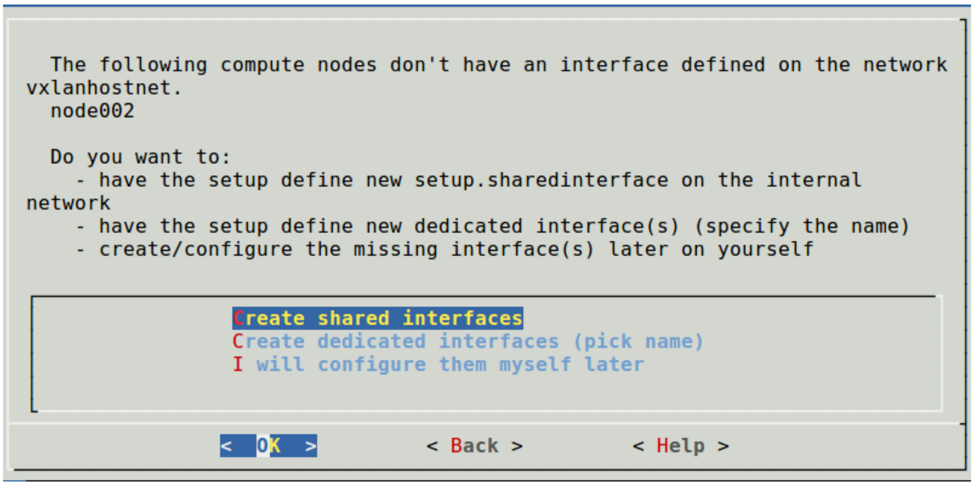
14. The IP address range can then be set up. Many ranges are possible. However, for this example, the range 192.168.200.100-192.168.200.200 is chosen (figure 1.14):



```
Please specify the Floating IP range
from the network externalnet for OpenStack?


Base address: 192.168.200.0
Broadcast address: 192.168.200.255

Starting IP                         192.168.200.100
End IP                              192.168.200.200
```

Figure 1.14: Configuring floating IP address range to be used on the external network

15. For the network for virtual networks, `vxlanhostnet`, that was set up in figure 1.11, the hypervisor node should have an interface that connects to it. A shared interface can be set up, and will be an alias for the bridged interface (figure 1.15):

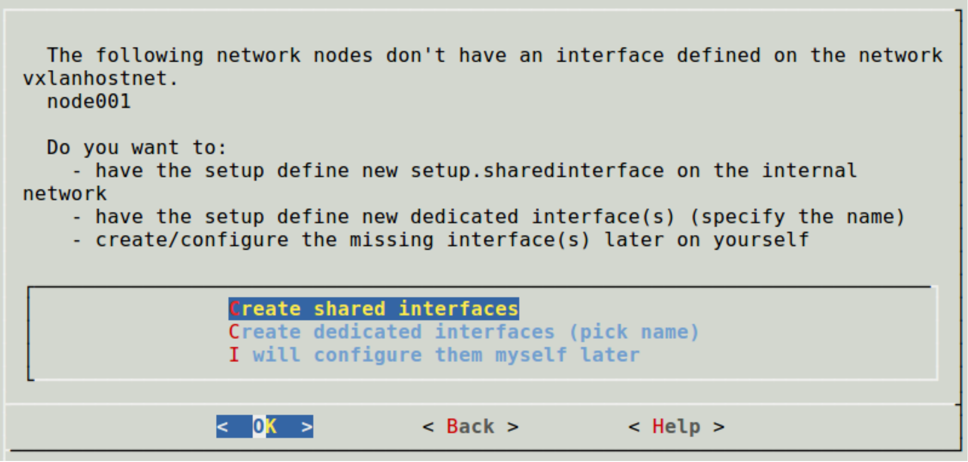


```
  The following compute nodes don't have an interface defined on the network
vxlanhostnet.
  node002

  Do you want to:
    - have the setup define new setup.sharedinterface on the internal
network
    - have the setup define new dedicated interface(s) (specify the name)
    - create/configure the missing interface(s) later on yourself
```

Create shared interfaces
Create dedicated interfaces (pick name)
I will configure them myself later
< OK >
< Back >
< Help >

Figure 1.15: Configuring the shared interface on the hypervisor (compute) node

16. Similarly, for `vxlanhostnet`, the network node should also have an interface that connects to it. A shared interface can be set up, and as before will be an alias for the bridged interface (figure 1.16):

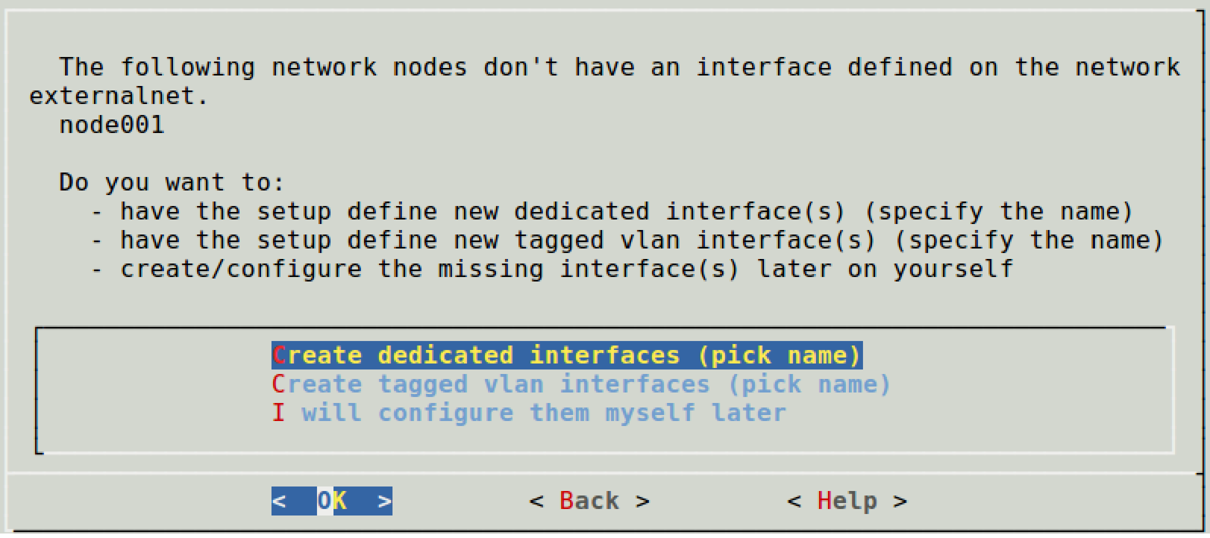


```
  The following network nodes don't have an interface defined on the network
vxlanhostnet.
  node001

  Do you want to:
    - have the setup define new setup.sharedinterface on the internal
network
    - have the setup define new dedicated interface(s) (specify the name)
    - create/configure the missing interface(s) later on yourself
```

Create shared interfaces
Create dedicated interfaces (pick name)
I will configure them myself later
< OK >
< Back >
< Help >

Figure 1.16: Configuring the shared interface on the network node to the virtual networks

17. The head node and the network/controller node are both connected to the external network of the cluster. For the external network that the network/controller node is attached to, a dedicated interface is created (figure 1.17). A name is set for the new interface, for example `eth1`.

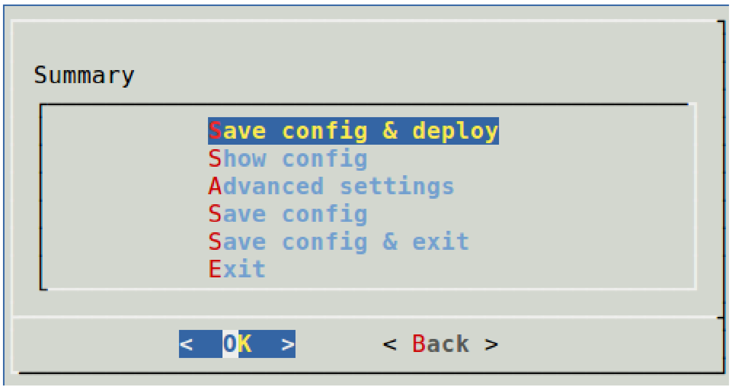


```
  The following network nodes don't have an interface defined on the network
externalnet.
  node001

  Do you want to:
    - have the setup define new dedicated interface(s) (specify the name)
    - have the setup define new tagged vlan interface(s) (specify the name)
    - create/configure the missing interface(s) later on yourself
```

Create dedicated interfaces (pick name)
Create tagged vlan interfaces (pick name)
I will configure them myself later
< OK >
< Back >
< Help >

Figure 1.17: Configuring the dedicated interface on the network node to the external network

18. The `Save config & deploy` option (figure 1.18) saves a YAML configuration file of the settings:



Summary

Figure 1.18: Saving and deploying the YAML configuration file

Deployment can then begin.

A deployment can take some time. Progress is displayed throughout the deployment procedure, and the session should end with something like:

```
Took:     35:48 min.
Progress: 100/100
################### Finished execution for 'Bright OpenStack', status: completed

Bright OpenStack finished!
```

1.4 Testing OpenStack Deployment

The example tasks that follow can be used to check if OpenStack has been successfully deployed and is behaving as expected. All the commands are run from Bright head node, and are a handy set of relatively common OpenStack-related actions. The commands in this testing section mostly avoid using the Bright Cluster Manager interface so that the direct OpenStack behavior is visible rather than Bright Cluster Manager behavior. If a command does not work in a similar way to what is shown, then the behavior should be investigated further.

Download a CirrOS image:

```
[root@bright73 ~]# wget -P /tmp/images http://download.cirros-cloud.net/0.3.3/cirros-0.3.3-\
x86_64-disk.img
```

Create an OpenStack test project:

```
[root@bright73 ~]# openstack project create brighttest
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | 0e6cd466a0f849ff8743654940b5f8b8 |
| enabled     | True                             |
| id          | 4c522f2ce1ad4cd18d67de341d1481ff |
| is_domain   | False                            |
| name        | brighttest                       |
| parent_id   | 0e6cd466a0f849ff8743654940b5f8b8 |
+-------------+----------------------------------+
```

Create an OpenStack test user

```
[root@bright73 ~]# openstack user create --project brighttest --password Ch@ngeMe --enable\
 brightuser
+--------------------+----------------------------------+
| Field              | Value                            |
+--------------------+----------------------------------+
| default_project_id | 4c522f2ce1ad4cd18d67de341d1481ff |
| domain_id          | 0e6cd466a0f849ff8743654940b5f8b8 |
| enabled            | True                             |
| id                 | df27f5f7b7da457984616651c2aaed71 |
| name               | brightuser                       |
+--------------------+----------------------------------+
```

Create a role for the test project and test user:

```
[root@bright73 ~]# openstack role add --project brighttest --user brightuser member
```

Create the test user in Bright:

```
[root@bright73 ~]# cmsh
[bright73]% user add brightuser
[bright73->user*[brightuser*]]% set password Ch@ngeMe
[bright73->user*[brightuser*]]% commit
```

Create a `.openstackrc` file for the test user:

```
[root@bright73 ~]# su - brightuser
Last login: Wed Feb 22 15:23:11 CET 2017 on pts/0
Creating ECDSA key for ssh
[brightuser@bright73 ~]$
[brightuser@bright73 ~]$ tail .openstackrc
export OS_PROJECT_DOMAIN_ID="0e6cd466a0f849ff8743654940b5f8b8"
export OS_USER_DOMAIN_ID="0e6cd466a0f849ff8743654940b5f8b8"
# Public Auth URL (used by users)
export OS_AUTH_URL="http://10.2.62.216:5000/v3"
```

```
# For keystone v3
export OS_IDENTITY_API_VERSION=3 # for the ’openstack’ utility to work
export OS_CACERT="/etc/keystone/ssl/certs/ca.pem"
# END AUTOGENERATED SECTION   -- DO NOT REMOVE
export OS_PASSWORD="Ch@ngeMe"
[brightuser@bright73 ~]$ source .openstackrc
[brightuser@bright73 ~]$
```

Create a key pair to be used by the test user:

```
[brightuser@bright73 ~]$ openstack keypair create --public-key
~/.ssh/id_ecdsa.pub brightuser-key
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| fingerprint | c6:50:f6:9b:c8:ac:7f:5c:e7:ff:54:b7:f7:8e:ec:fd |
| name        | brightuser-key                                  |
| user_id     | df27f5f7b7da457984616651c2aaed71                |
+-------------+-------------------------------------------------+
```

Create an OpenStack network:

```
[brightuser@bright73 ~]$ openstack network create brightnet
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | UP                                   |
| availability_zone_hints |                                      |
| availability_zones      |                                      |
| created_at              | 2017-02-22T14:34:04                  |
| description             |                                      |
| headers                 |                                      |
| id                      | 6abc5e3e-f6d0-4226-97f6-5dcfced70ed1 |
| ipv4_address_scope      | None                                 |
| ipv6_address_scope      | None                                 |
| mtu                     | 1450                                 |
| name                    | brightnet                            |
| project_id              | 4c522f2ce1ad4cd18d67de341d1481ff     |
| router_external         | Internal                             |
| shared                  | False                                |
| status                  | ACTIVE                               |
| subnets                 |                                      |
| tags                    | []                                   |
| updated_at              | 2017-02-22T14:34:05                  |
+-------------------------+--------------------------------------+
```

Create a subnet for the network:

```
[brightuser@bright73 ~]$ neutron subnet-create brightnet 192.168.100.0/24
Created a new subnet:
+-------------------+------------------------------------------------------+
| Field             | Value                                                |
+-------------------+------------------------------------------------------+
| allocation_pools  | {"start": "192.168.100.2", "end": "192.168.100.254"} |
| cidr              | 192.168.100.0/24                                     |
| dns_nameservers   |                                                      |
| enable_dhcp       | True                                                 |
| gateway_ip        | 192.168.100.1                                        |
| host_routes       |                                                      |
| id                | 600a31e9-74b7-424c-93de-db067d04c880                 |
| ip_version        | 4                                                    |
| ipv6_address_mode |                                                      |
| ipv6_ra_mode      |                                                      |
| name              |                                                      |
| network_id        | 023350ae-4979-4b89-b584-3e148526df73                 |
| subnetpool_id     |                                                      |
| tenant_id         | 599dc6e8bfe54c34a9cc41b2d3b8b497                     |
+-------------------+------------------------------------------------------+
```

Create a router:

```
[brightuser@bright73 ~]$ neutron router-create brightrouter
Created a new router:
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | True                                 |
| availability_zone_hints |                                      |
| availability_zones      |                                      |
| description             |                                      |
| external_gateway_info   |                                      |
| id                      | 41cd7089-becd-458d-9c55-cc4ee10c1f3a |
| name                    | brightrouter                         |
| routes                  |                                      |
| status                  | ACTIVE                               |
| tenant_id               | 4c522f2ce1ad4cd18d67de341d1481ff     |
+-------------------------+--------------------------------------+
```

Attach the router to the bright-external-flat-externalnet: (this is the flat network which is bridged with the interface to the outside)

```
[brightuser@bright73 ~]$ neutron router-gateway-set brightrouter bright-external-flat-externalnet
Set gateway for router brightrouter
```

Attach the router to the network created earlier:

```
[brightuser@bright73 ~]$ neutron subnet-list
+--------------+------------------+-------------------------------------------------------+
| id           | cidr             | allocation_pools                                      |
+--------------+------------------+-------------------------------------------------------+
| 6ec4bf65-... | 192.168.100.0/24 | {"start": "192.168.100.2", "end": "192.168.100.254"} |
+--------------+------------------+-------------------------------------------------------+
[brightuser@bright73 ~]$ neutron router-interface-add brightrouter 6ec4bf65-...
Added interface 511d1ce9-1a12-4454-afa1-1f50f4fb6b5b to router brightrouter.
```

Import the CirrOS image from the downloaded CirrOS cloud image:

```
[brightuser@bright73 ~]$ glance image-create --name "cirros-0.3.3-x86_64" --file /tmp/image\
s/cirros-0.3.3-x86_64-disk.img --disk-format qcow2 --container-format bare --progress
[============================>] 100%
+------------------+--------------------------------------------------------------------+
| Property         | Value                                                               |
+------------------+---------------------------------------------------------------------+
| checksum         | 133eae9fb1c98f45894a4e60d8736619                                    |
| container_format | bare                                                                |
| created_at       | 2017-02-22T14:50:34Z                                                |
| direct_url       | file:///var/lib/glance/images/6adef364-3d89-4cfc-b780-c1fc701668ae |
| disk_format      | qcow2                                                               |
| id               | 6adef364-3d89-4cfc-b780-c1fc701668ae                                |
| min_disk         | 0                                                                   |
| min_ram          | 0                                                                   |
| name             | cirros-0.3.3-x86_64                                                 |
| owner            | 4c522f2ce1ad4cd18d67de341d1481ff                                    |
| protected        | False                                                               |
| size             | 13200896                                                            |
| status           | active                                                              |
| tags             | []                                                                  |
| updated_at       | 2017-02-22T14:50:34Z                                                |
| virtual_size     | None                                                                |
| visibility       | private                                                             |
+------------------+---------------------------------------------------------------------+
```

Create a CirrOS VM:

```
[brightuser@bright73 ~]$ neutron net-list
+-----------+-------------------------------------------------+---------------------------+
| id        | name                                            | subnets                   |
+-----------+-------------------------------------------------+---------------------------+
| 93022c8...| bright-external-flat-externalnet                | d00b8...                  |
| 6abc5e3...| brightnet                                       | 6ec42... 192.168.100.0/24 |
+-----------+-------------------------------------------------+---------------------------+
[brightuser@bright73 ~]$ openstack server create --image cirros-0.3.3-x86_64 --flavor m1.xs\
mall --key-name brightuser-key --nic net-id=6abc5e3... cirrosvm
+--------------------------------------+--------------------------------------------------+
| Field                                | Value                                            |
+--------------------------------------+--------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                           |
| OS-EXT-AZ:availability_zone          |                                                  |
| OS-EXT-STS:power_state               | 0                                                |
| OS-EXT-STS:task_state                | scheduling                                       |
| OS-EXT-STS:vm_state                  | building                                         |
| OS-SRV-USG:launched_at               | None                                             |
| OS-SRV-USG:terminated_at             | None                                             |
| accessIPv4                           |                                                  |
| accessIPv6                           |                                                  |
| addresses                            |                                                  |
| adminPass                            | sm6kxWwgPDnK                                     |
| config_drive                         |                                                  |
| created                              | 2017-02-22T16:43:31Z                             |
| flavor                               | m1.xsmall (235bd82a-5cdc-438e-9526-261848da5714)|
| hostId                               |                                                  |
| id                                   | 484e7243-9ad5-42b4-9886-7d177a99696a             |
| image                                | cirros-0.3.3-x86_64 (6adef364-3d89-4cfc-...)     |
| key_name                             | brightuser-key                                   |
| name                                 | cirrosvm                                         |
| os-extended-volumes:volumes_attached | []                                               |
| progress                             | 0                                                |
| project_id                          | 4c522f2ce1ad4cd18d67de341d1481ff                |
| properties                          |                                                 |
| security_groups                     | [{u’name’: u’default’}]                         |
| status                              | BUILD                                           |
| updated                             | 2017-02-22T16:43:31Z                            |
| user_id                             | df27f5f7b7da457984616651c2aaed71                |
+-------------------------------------+-------------------------------------------------+
```

Create a floating IP:

```
[brightuser@bright73 ~]$ openstack ip floating create bright-external-flat-externalnet
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| fixed_ip    | None                                 |
| id          | bee67a97-1855-4b1e-a53f-317e66c898c4 |
| instance_id | None                                 |
| ip          | 192.168.200.101                      |
| pool        | bright-external-flat-externalnet     |
+-------------+--------------------------------------+
```

Attach the floating IP to the CirrOS VM:

```
[brightuser@bright73 ~]$ openstack ip floating add 192.168.200.101 cirrosvm
```

Enable ssh port 22 in the default security group:

```
[brightuser@bright73 ~]$ openstack security group rule create --dst-port 22 default
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| id                    | 9f10223b-4cdf-4e7b-aa72-879c85710bb8 |
| ip_protocol           | tcp                                  |
| ip_range              | 0.0.0.0/0                            |
| parent_group_id       | 5affac60-34b8-4217-8670-c82a8c8e2d88 |
| port_range            | 22:22                                |
| remote_security_group |                                      |
+-----------------------+--------------------------------------+
```

Test ssh access to the CirrOS VM:

```
[brightuser@bright73 ~]$ ssh 192.168.200.101 -l cirros
Warning: Permanently added ’192.168.200.101’ (RSA) to the list of known hosts.
cirros@192.168.200.101’s password:
$ hostname
cirrosvm
```

2

Introduction

OpenStack is an open source implementation of cloud services. It is currently (2017) undergoing rapid development, and its roadmap is promising.

An implementation of OpenStack, based on the OpenStack Mitaka release (`https://www.openstack.org/software/mitaka/`) is integrated into the Bright Cluster Manager 7.3 for OpenStack edition. It is supported for RHEL7 onwards.

The implementation of OpenStack is usable and stable for regular use in common configurations. In a complex and rapidly-evolving product such as OpenStack, the number of possible unusual configuration changes is vast. As a result, the experience of Bright Computing is that Bright Cluster Manager can sometimes run into OpenStack issues while implementing the less common OpenStack configurations.

As one of the supporting organizations of OpenStack, Bright Computing is committed towards working together with OpenStack developers to help Bright customers resolve any such issue. The end result after resolving the issue means that there is a selection pressure that helps evolve that aspect of OpenStack, so that it becomes convenient and stable for regular use. This process benefits all participants in the OpenStack software ecosystem.

OpenStack consists of subsystems, developed as upstream software projects[1]. A software project provides capabilities to OpenStack via the implementation of a backend service, and thereby provides an OpenStack service. The OpenStack service can thus be implemented by interchangeable backends, which projects can provide.

For example, the OpenStack Cinder project provides block storage capabilities to OpenStack via the implementation of, for example, NFS or Ceph block storage. The OpenStack's block storage service can therefore be implemented by the interchangable backends of the NFS or Ceph projects. Indeed, the entire Cinder project itself can be replaced by a Cinder rewrite from scratch. As far as the user is concerned the end result is the same.

An analogy to OpenStack is operating system packaging, as provided by distributions:

An operating system distribution consists of subsystems, maintained as packages and their dependencies. Some subsystems provide capabilities to the operating system via the implementation of a backend service. The service can often be implemented by interchangeable backends for the subsystem.

A specific example for an operating system distribution would be the mailserver subsystem that provides mail delivery capabilities to the operating system via the implementation of, for example, Postfix or Sendmail. The mailserver package and dependencies can therefore be implemented by the interchangeable backends of the Postfix or Sendmail software. As far as the e-mail user is concerned, the end result is the same.

The project that implements the backend can also change, if the external functionality of the project remains the same.

Some of the more common OpenStack projects are listed in the following table:

[1] The term projects must not be confused with the term used in OpenStack elsewhere, where projects, or sometimes tenants, are used to refer to a group of users

Service	OpenStack Project	Managed By Bright
Compute	Nova	✓
Object Storage	Swift	depends*
Block Storage	Cinder	✓
Networking	Neutron	✓
Dashboard	Horizon	✓
Identity Service	Keystone	✓
Orchestration	Heat	✓
Telemetry	Ceilometer	×
Database Service	Trove	×
Image Service	Glance	✓

* Bright Cluster Manager does not manage the OpenStack reference implementation for Swift object storage, but does manage a replacement, the API-compatible Ceph RADOS Gateway implementation.

Not all of these projects are integrated, or needed by Bright Cluster Manager for a working OpenStack system. For example, Bright Cluster Manager already has an extensive monitoring system and therefore does not for now implement `Ceilometer`, while `Trove` is ignored for now until it becomes more popular.

Projects that are not yet integrated can in principle be added by administrators on top of what is deployed by Bright Cluster Manager, even though this is not currently supported or tested by Bright Computing. Integration of the more popular of such projects, and greater integration in general, is planned in future versions of Bright Cluster Manager.

This manual explains the installation, configuration, and some basic use examples of the OpenStack projects that have so far been integrated with Bright Cluster Manager.

3

OpenStack Installation

To Use Ceph, It Must Be Installed Before Deploying OpenStack

If OpenStack is to access Ceph for storage purposes, for any combination of block storage (Cinder), image storage (Glance), ephemeral storage (Nova), or object storage (RADOS Gateway), then the Ceph components must first be installed with `cm-ceph-setup` (Chapter 4) before starting the OpenStack installation procedure covered here.

Hardware Requirement For Running OpenStack

The optimum hardware requirements for OpenStack depend on the intended use. A rule of thumb is that the number of cores on the compute nodes determines the number of virtual machines.

OpenStack itself can run entirely on one physical machine for limited demonstration purposes.

However, if running OpenStack with Bright Cluster Manager, then a standard reference architecture used by Bright Computing consists of the following three types of nodes:

- A head node.
- Several regular nodes that can be used as hypervisor hosts. Regular nodes (Bright Cluster Manager terminology) are also commonly called compute nodes, and are typically multicore. Running guest VMs is therefore a suitable use for regular nodes.
- 3 nodes that combine OpenStack controller and OpenStack network node functionality.

For a standard reference configuration, recommended hardware specifications for useful demonstration purposes are:

- **Head node**: 8GB RAM, 4 cores and two network interfaces. In a standard configuration the head node does not run OpenStack services, other than the OpenStack-associated Haproxy service.
- **Regular nodes:** 2GB RAM per core. Each regular node has a network interface.
  - In larger clusters, it may be a good idea to separate the OpenStack controller functionality from networking functionality. If a regular node is configured as a controller, then it is best to have at least 8GB RAM.
- **3 OpenStack controller/network nodes**: 8GB RAM and two network interfaces. 3 nodes is the minimum needed to provide OpenStack high availability via Galera cluster for OpenStack databases.

  Networking nodes prior to Liberty could run as standalone nodes. In Liberty this is still possible, but not officially supported by OpenStack. Bright Cluster Manager OpenStack edition therefore uses combined controller/network nodes.

  The database for the controller nodes cannot run with two OpenStack controllers. If the administrator would like use something other than the standard reference controller configuration of

3 controllers, then it is possible to run with just one OpenStack controller, without OpenStack database high availability. More than three controllers are also allowed, in a high-availability configuration.

The OpenStack controller/network nodes provide:

- OpenStack API endpoint services for Nova, Cinder, Keystone, Neutron, Glance, and Heat.
- Horizon Dashboard. This is a Django-based web service.
- RabbitMQ nodes, deployed as a RabbitMQ cluster. This is used in the OpenStack backend for internal communication within an OpenStack service. For example, such as between `nova-api`, `nova-conductor`, `nova-scheduler`, `nova-compute`, or such as between `neutron-server` and the Neutron L2 agents.
- If Ceph is used, then the controller nodes can also be used as Ceph monitor nodes, in order to provide high availability for the Ceph monitor node data. In this case, more than 8GB of memory is needed for the controller nodes.

An *ethernet fabric* is used as a terminology to talk about treating the network architecture as being based on a giant flat logical OSI Layer 2-style network connected to a single switch, with point-to-point routing, rather than the traditional OSI 2/3 mixture with a hierarchy of access, distribution, and core routers.

The reference architecture networking runs on an ethernet fabric for the:

- internal network of the cluster, which is also the OpenStack management network.
- V(X)LAN network of the cluster, which is used by OpenStack virtual networks.

If Ceph is also deployed, then an ethernet fabric is assumed for:

- The public Ceph network.
- The Ceph replication network.
- An optional external network that is used to access virtual machines in OpenStack via Floating IPs.

Hard drive requirements for minimal systems can remain as for those required for a regular Bright Cluster Manager cluster. For production systems, these minimal requirements are however unlikely to work for very long. Storage requirements should therefore be considered with care according to the use case. If necessary, Bright Computing can provide advice on this.

Running OpenStack under Bright Cluster Manager with fewer resources than suggested in the preceding is possible but may cause issues. While such issues can be resolved, they are usually not worth the time spent analyzing them, due to the great number of possible configurations. It is better to run with ample resources, and then analyze the resource consumption in the configuration that is used, to see what issues to be aware of when scaling up to a production system.

Running a Bright Cluster Manager OpenStack cluster that varies greatly from the reference cluster is also possible. If necessary, Bright Computing can provide advice on this.

Ways Of Installing OpenStack

The version of OpenStack that is integrated with Bright Cluster Manager can be installed in the following two ways:

- Using the GUI-based `Setup Wizard` button from within `cmgui` (section 3.1). This is the recommended installation method.
- Using the text-based `cm-openstack-setup` utility (section 3.2). The utility is a part of the standard `cluster-tools` package.

The priorities that the package manager uses are expected to be at their default settings, in order for the installation to work.

By default, deploying OpenStack installs the following projects: Keystone, Nova, Cinder, Glance, Neutron, Heat and Horizon (the dashboard).

If Ceph is used, then Bright also deploys RADOS Gateway as a Swift-API-compatible object storage system. Using RADOS Gateway instead of the reference Swift object storage is regarded in the OpenStack community as good practice, and is indeed the only object storage system that Bright Cluster Manager manages for OpenStack. Alternative backend storage is possible at the same time as object storage, which means, for example, that block and image storage are options that can be used in a cluster at the same time as object storage.

3.1 Installation Of OpenStack From `cmgui`

The `cmgui` OpenStack `Setup Wizard` is the preferred way to install OpenStack. A prerequisite for running it is that the head node should be able to connect to the distribution repositories, or alternatively the head node should have OpenStack RPMs preinstalled on it. Preinstalled OpenStack RPMs can be configured as part of the head node installation from the ISO, if the ISO that is used the Bright Cluster Manager OpenStack edition.

Some suggestions and background notes These are given here to help the administrator understand what the setup configuration does, and to help simplify deployment. Looking at these notes after a dry-run with the wizard will probably be helpful.

- A VXLAN (Virtual Extensible LAN) network is similar to a VLAN network in function, but has features that make it more suited to cloud computing.
  - If VXLANs are to be used, then the wizard is able to help create a VXLAN *overlay network* for OpenStack *tenant networks*.

    An OpenStack tenant network is a network used by a group of users allocated to a particular virtual cluster.

    A VXLAN overlay network is a Layer 2 network "overlaid" on top of a Layer 3 network. The VXLAN overlay network is a virtual LAN that runs its frames encapsulated within UDP packets over the regular TCP/IP network infrastructure. It is very similar to VLAN technology, but with some design features that make it more useful for cloud computing needs. One major improvement is that around 16 million VXLANs can be made to run over the underlying Layer 3 network. This is in contrast to the 4,000 or so VLANs that can be made to run over their underlying Layer 2 network, if the switch port supports that level of simultaneous capability.

    By default, if the VXLAN network and VXLAN network object do not exist, then the wizard helps the administrator create a `vxlanhostnet` network and network object (section 3.1.11). The network is attached to, and the object is associated with, all non-head nodes taking part in the OpenStack deployment. If a `vxlanhostnet` network is pre-created beforehand, then the wizard can guide the administrator to associate a network object with it, and ensure that all the non-head nodes participating in the OpenStack deployment are attached and associated accordingly.
  - The VXLAN network runs over an IP network. It should therefore have its own IP range, and each node on that network should have an IP address. By default, a network range of 10.161.0.0/16 is suggested in the VXLAN configuration screen (section 3.1.11, figure 3.12).
  - The VXLAN network can run over a dedicated physical network, but it can also run over an alias interface on top of an existing internal network interface. The choice is up to the administrator.

  - It is possible to deploy OpenStack without VXLAN overlay networks if user instances are given access to the internal network. Care must then be taken to avoid IP addressing conflicts.

- When allowing for Floating IPs and/or enabling outbound connectivity from the virtual machines (VMs) to the external network via the network node, the network node can be pre-configured manually according to how it is connected to the internal and external networks. Otherwise, if the node is not pre-configured manually, the wizard then carries out a basic configuration on the network node that

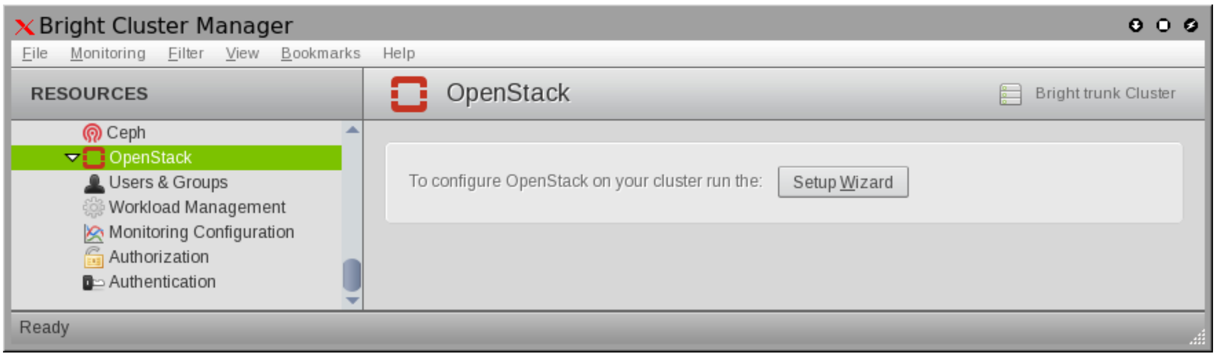
  - configures one physical interface of the network node to be connected to the internal network, so that the network node can route packets for nodes on the internal network.

  - configures the other physical interface of the network node to be connected to the external network so that the network node can route packets from external nodes.

The wizard asks the user several questions on the details of how OpenStack is to be deployed. From the answers, it generates an YAML document with the intended configuration. Then, in the back-end, largely hidden from the user, it runs the text-based `cm-openstack-setup` script (section 3.2) with this configuration on the active head node. In other words, the wizard can be regarded as a GUI front end to the `cm-openstack-setup` utility.

The practicalities of executing the wizard: The explanations given by the wizard during its execution steps are intended to be verbose enough so that the administrator can follow what is happening.

The wizard is accessed via the OpenStack resource in the left pane of `cmgui` (figure 3.1). Launching the wizard is only allowed if the Bright Cluster Manager license (Chapter 4 of the *Installation Manual*) entitles the license holder to use OpenStack.

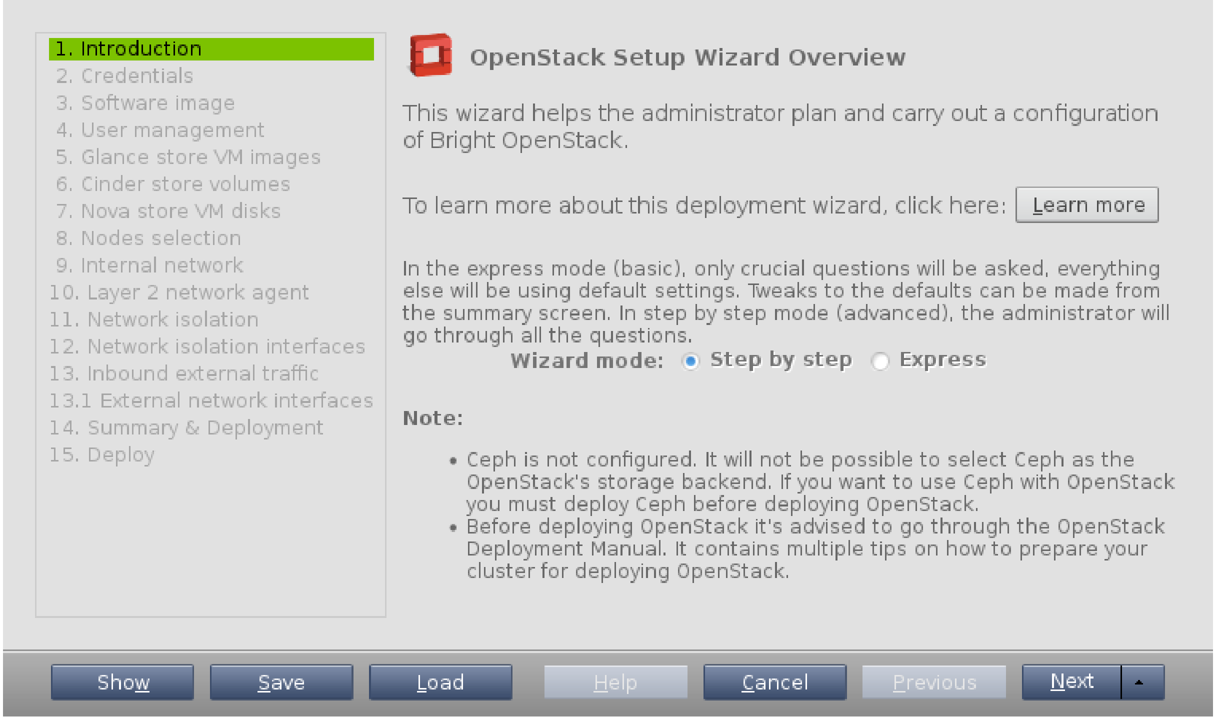


To configure OpenStack on your cluster run the:

Figure 3.1: The `Setup Wizard` Button In `cmgui`'s OpenStack Resource

The wizard runs through the screens in sections 3.1.1-3.1.15, described next.

3.1.1 OpenStack Setup Wizard Overview



OpenStack Setup Wizard Overview

This wizard helps the administrator plan and carry out a configuration of Bright OpenStack.

To learn more about this deployment wizard, click here: Learn more

In the express mode (basic), only crucial questions will be asked, everything else will be using default settings. Tweaks to the defaults can be made from the summary screen. In step by step mode (advanced), the administrator will go through all the questions.

Wizard mode: ● Step by step ○ Express

Note:

- Ceph is not configured. It will not be possible to select Ceph as the OpenStack's storage backend. If you want to use Ceph with OpenStack you must deploy Ceph before deploying OpenStack.
- Before deploying OpenStack it's advised to go through the OpenStack Deployment Manual. It contains multiple tips on how to prepare your cluster for deploying OpenStack.

Figure 3.2: OpenStack Setup Wizard Overview Screen

The main overview screen (figure 3.2) gives an overview of how the wizard runs. The Learn more button displays a pop up screen to further explain what information is gathered, and what the wizard intends to do with the information.

The main overview screen also asks for input on if the wizard should run in step-by-step mode, or in express mode.

- Step-by-step mode asks for many explicit configuration options, and can be used by the administrator to become familiar with the configuration options.
- Express mode asks for very few configuration options, and uses mostly default settings. It can be used by an administrator that would like to try out a relatively standard configuration.

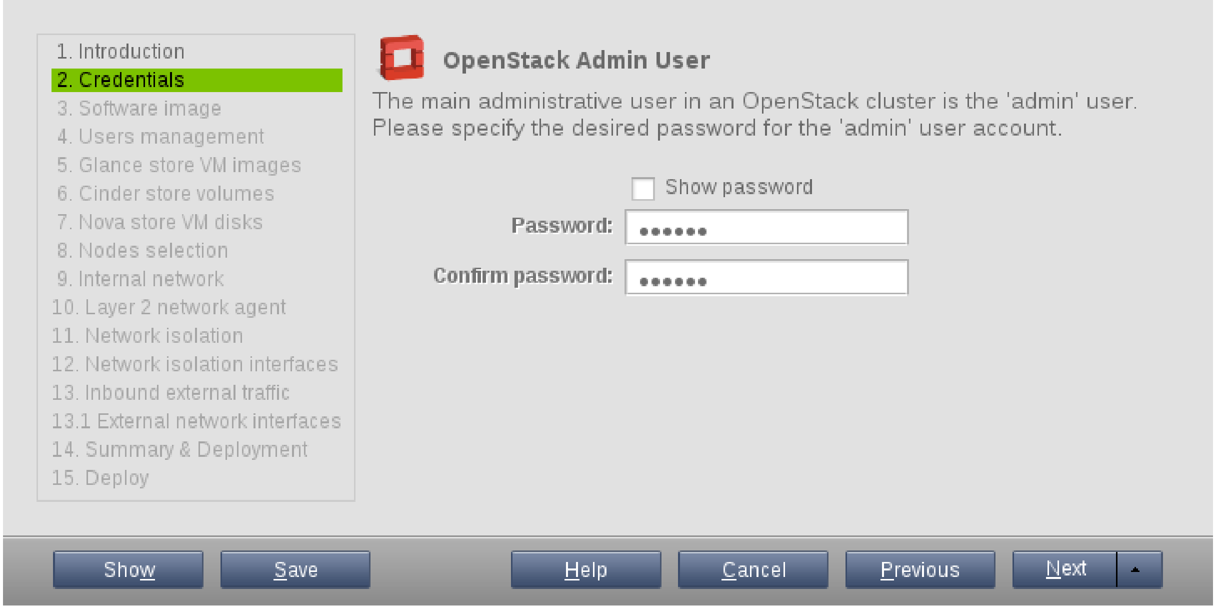
During the wizard procedure, buttons are available at the bottom of the screen. Among other options, in the main overview screen, the buttons allow a previously-saved configuration to be loaded, or allow the current configuration to be saved. The configurations are loaded or saved in a YAML format.

On clicking the Next button:

- If the express mode has been chosen, then the wizard skips the in-between steps, and jumps ahead to the Summary screen (section 3.1.15).
- Otherwise, if the step-by-step mode has been chosen, then each time the Next button is clicked, the wizard goes to the next screen in the series of in-between steps. Each screen allows options to be configured.

  The steps are described in the following sections 3.1.2-3.1.15.

3.1.2 OpenStack admin User Screen



OpenStack Admin User

The main administrative user in an OpenStack cluster is the 'admin' user. Please specify the desired password for the 'admin' user account.

□ Show password

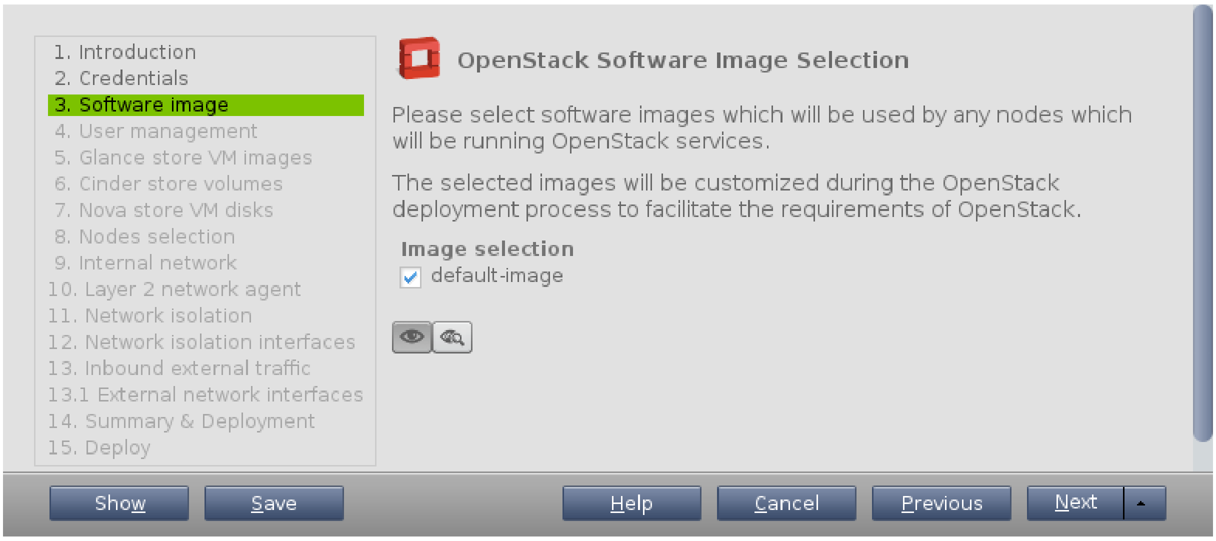
Password: ••••••

Confirm password: ••••••

Figure 3.3: OpenStack admin User Screen

The OpenStack credentials screen (figure 3.3) allows the administrator to set the password for the OpenStack admin user. The admin user is how the administrator logs in to the Dashboard URL to manage OpenStack when it is finally up and running.

3.1.3 OpenStack Software Image Selection



OpenStack Software Image Selection

Please select software images which will be used by any nodes which will be running OpenStack services.

The selected images will be customized during the OpenStack deployment process to facilitate the requirements of OpenStack.

Image selection

☑ default-image

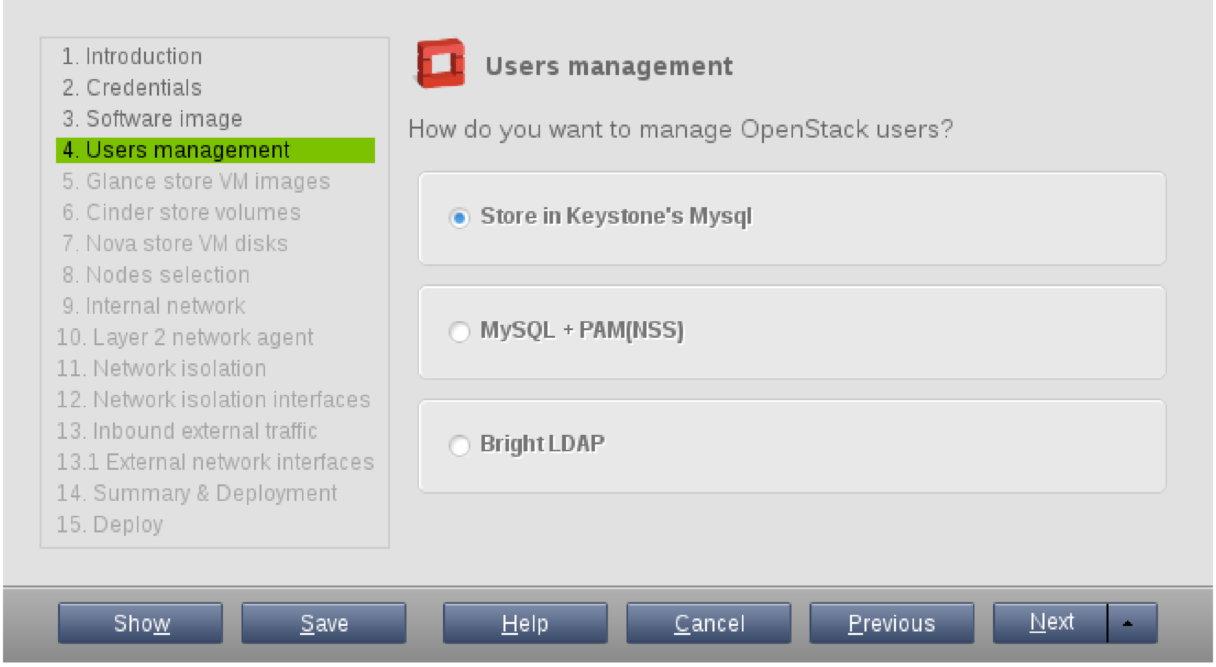
Figure 3.4: OpenStack Software Image Selection Screen

The OpenStack software image selection screen (figure 3.4) lets the administrator select the software image that is to be modified and used on the nodes that run OpenStack.

The administrator can clone the default-image before running the wizard and modifying the image, in order to keep an unmodified default-image as a backup.

The administrator should take care not to move a node with OpenStack roles to another category that contains a different image without OpenStack roles. OpenStack nodes behave quite differently from non-OpenStack nodes.

3.1.4 User Management



Users management

How do you want to manage OpenStack users?

● Store in Keystone's Mysql

○ MySQL + PAM(NSS)

○ Bright LDAP

Figure 3.5: OpenStack User Management Screen

The User Management screen (figure 3.5) allows the administrator to select how OpenStack users are to be managed. Choices available are:

- Store in a MySQL database managed by Keystone, and by default isolate users from the non-OpenStack part of the cluster.

  Thus, in this case, the OpenStack users are managed by Keystone, and isolated from the LDAP users managed by Bright Cluster Manager.

- Store in a MySQL database managed by Keystone, and use PAM (NSS). Further details on this can be found in the background note on page 74.

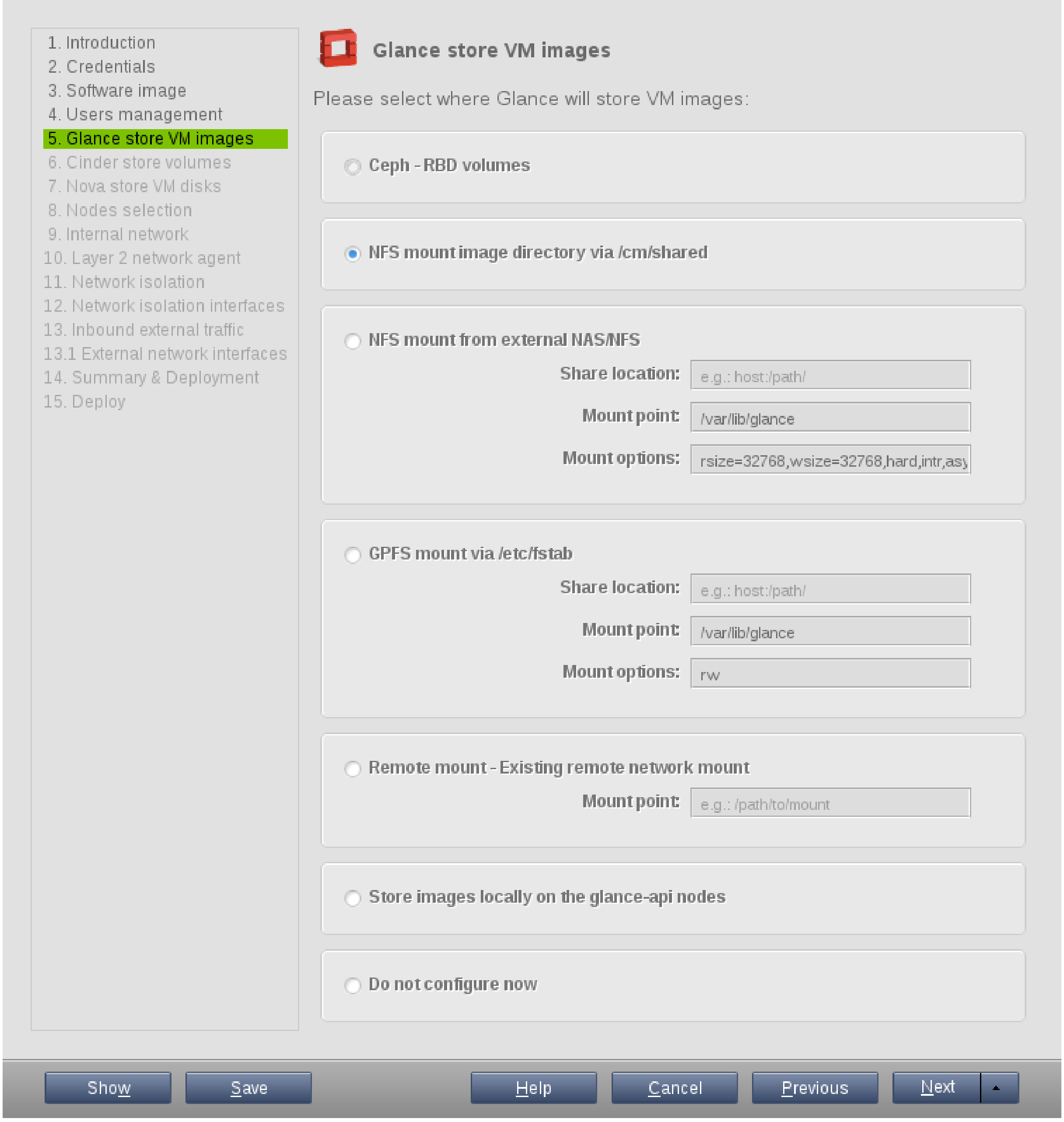
- Use Bright Cluster Manager LDAPS authentication. Further details on this can be found in the background note on page 74.

Keystone can also be set to authenticate directly with an external LDAP or AD server, but this requires manual configuration in Bright Cluster Manager. In cmsh this configuration can be done as follows:

Example

```
[root@bright73 ~]# cmsh
[bright73]% openstack settings default
[bright73->openstack[default]->settings]% authentication
[bright73->...->settings->authentication]% set custompublicauthhost <external authentication server>
```

3.1.5 Glance VM Image Storage



Glance store VM images

Please select where Glance will store VM images:

○ Ceph - RBD volumes

● NFS mount image directory via /cm/shared

○ NFS mount from external NAS/NFS
Share location: e.g.: host:/path/
Mount point: /var/lib/glance
Mount options: rsize=32768,wsize=32768,hard,intr,as

○ GPFS mount via /etc/fstab
Share location: e.g.: host:/path/
Mount point: /var/lib/glance
Mount options: rw

○ Remote mount - Existing remote network mount
Mount point: e.g.: /path/to/mount

○ Store images locally on the glance-api nodes

○ Do not configure now

Figure 3.6: OpenStack Glance VM Image Storage Screen

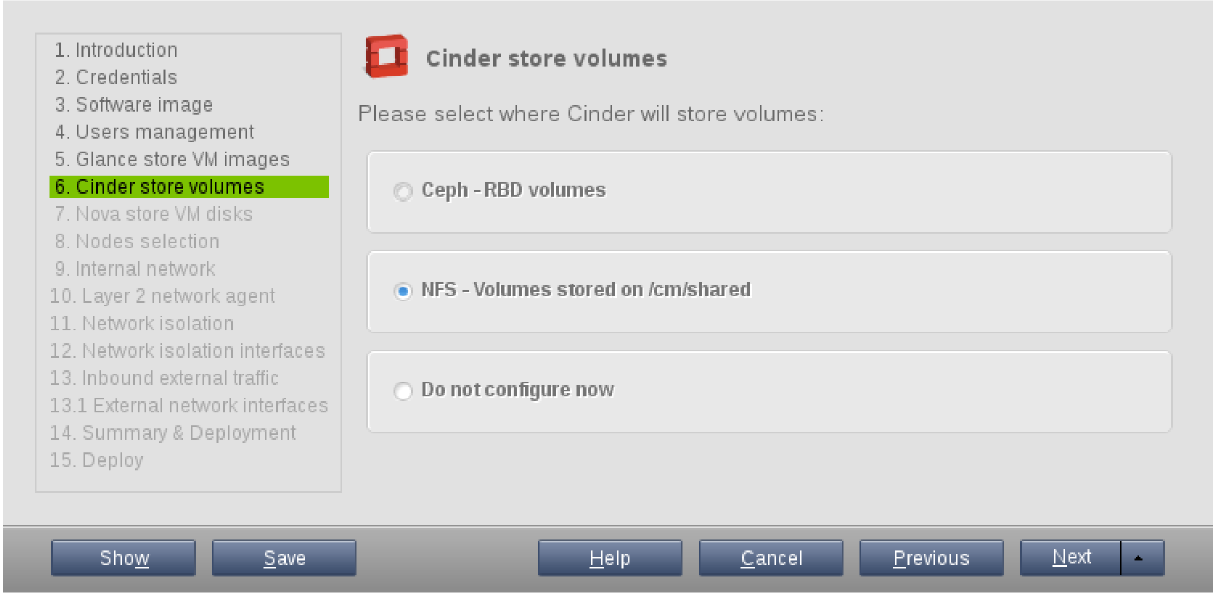
The `Glance VM Image Storage` screen (figure 3.6) allows the administrator to select where virtual machine images are stored. Choices are:

- As Ceph-RBD volumes
- Within an NFS image directory, using the internal NFS. This is using a directory under `/cm/shared`
- Within an NFS image directory, using an external NAS/NFS. The share location, mount point and mount options should be specified.
- Within a GPFS image directory, mounted via `/etc/fstab`. The share location, mount point, and

mount options should be specified.

- Using a remote mount from another network file system. The mount point should be specified.
- As images stored locally on the glance-api nodes.

3.1.6 Cinder Volume Storage



Cinder store volumes

Please select where Cinder will store volumes:

○ Ceph - RBD volumes

● NFS - Volumes stored on /cm/shared

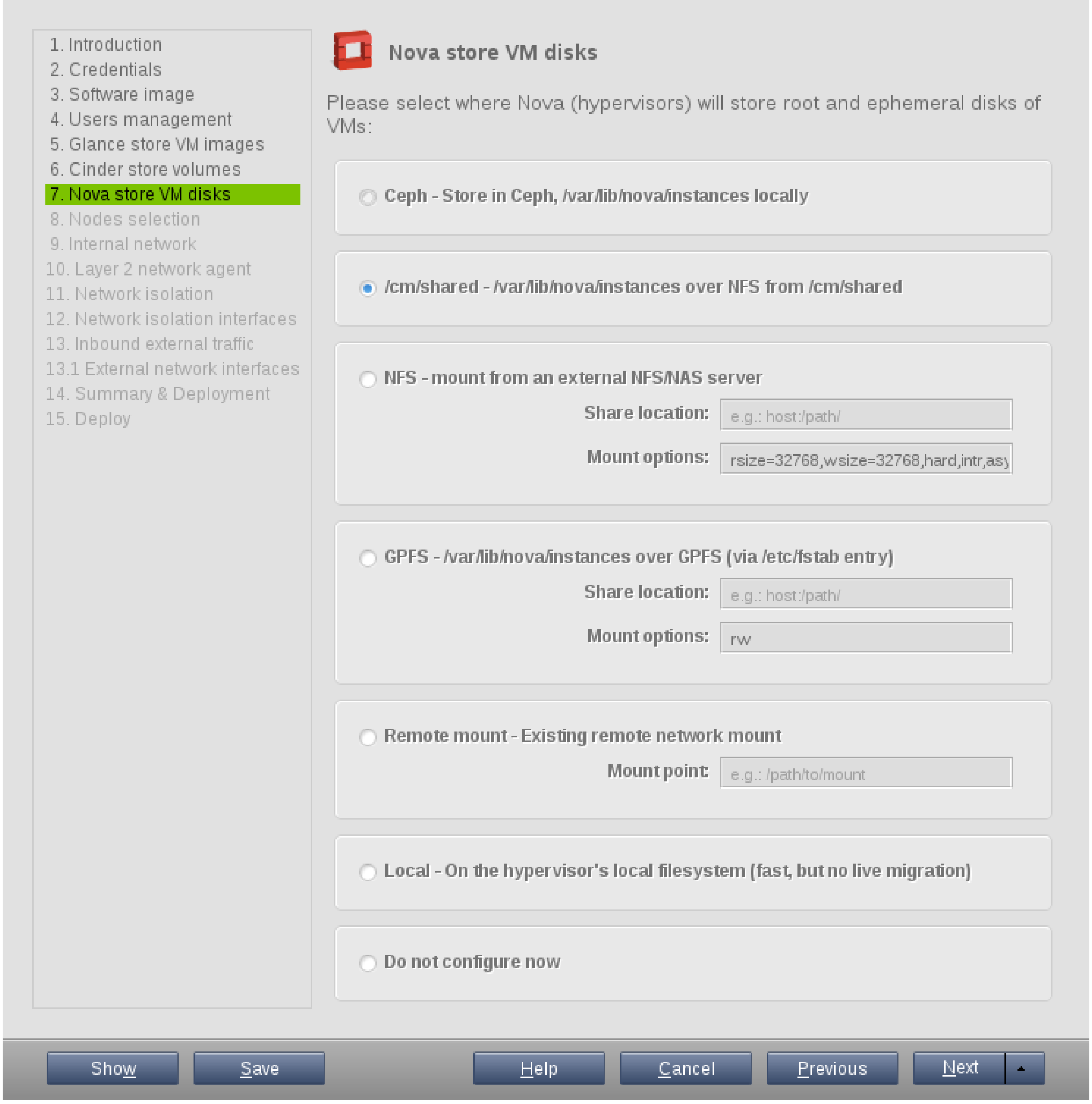
○ Do not configure now

Figure 3.7: OpenStack Cinder Volume Storage Screen

The `Cinder Volume Storage` screen (figure 3.7) allows the administrator to choose how Cinder volumes are to be stored. Options are:

- As Ceph-RBD volumes
- Within an NFS directory, using the internal NFS. This is using a directory under `/cm/shared`
- Within a GPFS volume. The mount point should be specified.

3.1.7 Nova VM Disks Storage



Nova store VM disks

Please select where Nova (hypervisors) will store root and ephemeral disks of VMs:

○ Ceph - Store in Ceph, /var/lib/nova/instances locally

● /cm/shared - /var/lib/nova/instances over NFS from /cm/shared

○ NFS - mount from an external NFS/NAS server
Share location: e.g.: host:/path/
Mount options: rsize=32768,wsize=32768,hard,intr,asy

○ GPFS - /var/lib/nova/instances over GPFS (via /etc/fstab entry)
Share location: e.g.: host:/path/
Mount options: rw

○ Remote mount - Existing remote network mount
Mount point: e.g.: /path/to/mount

○ Local - On the hypervisor's local filesystem (fast, but no live migration)

○ Do not configure now

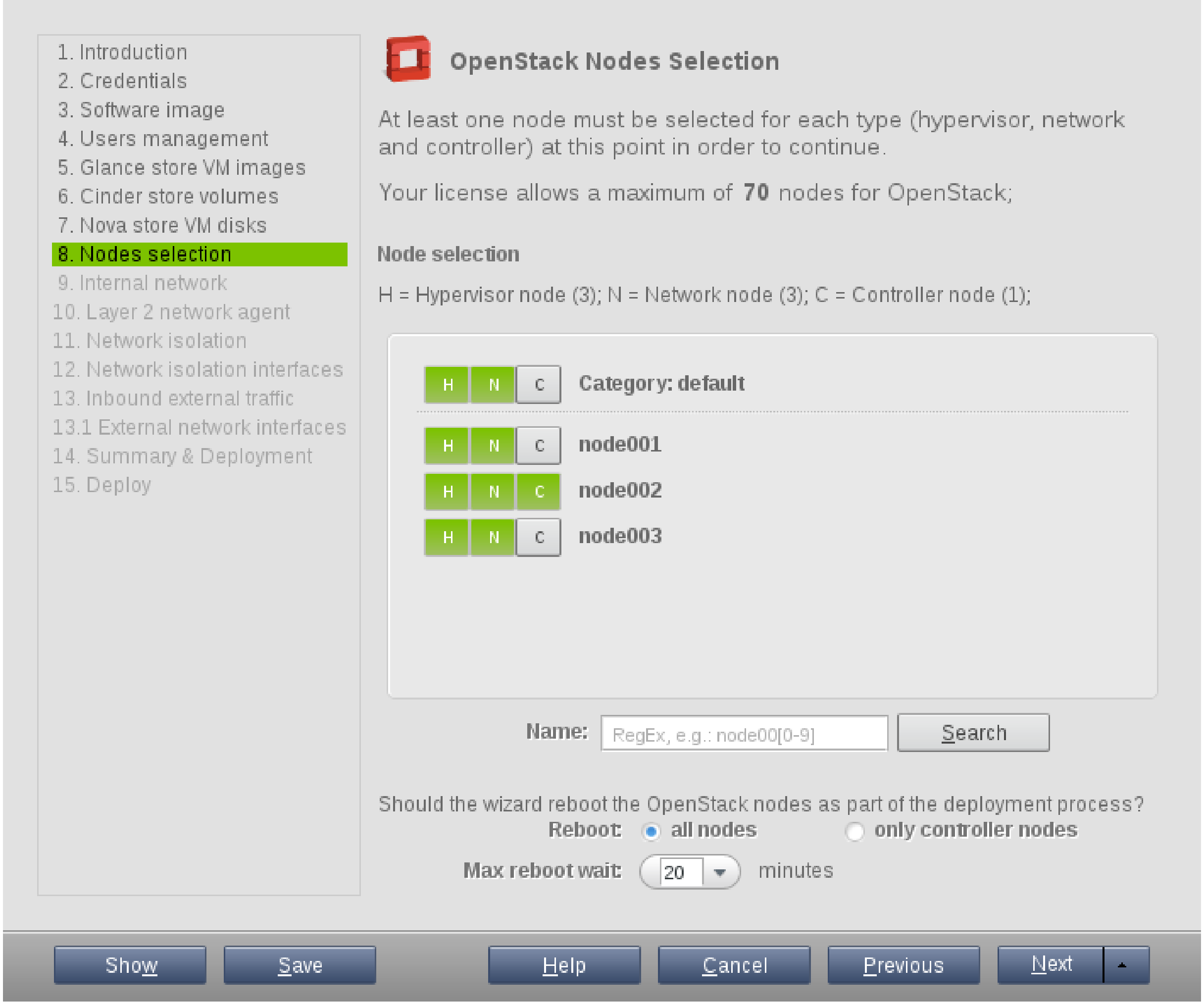
Figure 3.8: OpenStack Nova VM Disks Storage Screen

The `Nova VM Disks Storage` screen (figure 3.8) allows the administrator to choose how Nova hypervisors store the root and ephemeral disks of VMs. Options are:

- Ceph: Stored locally under `/var/lib/nova/instances`
- An NFS directory, using the internal NFS. This is using a directory served from `/cm/shared` as `/var/lib/nova/instances`.
- An NFS directory, using an external NAS/NFS. The share location, and mount options should be specified.
- A GPFS directory, mounted via `/etc/fstab`. The directory is served as `/var/lib/nova/instances`

- A remote mount from another network file system. The mount point should be specified.
- A local filesystem on the hypervisor itself, under /var/lib/nova. This is fast, but does not support live migration.

3.1.8 OpenStack Nodes Selection



OpenStack Nodes Selection

At least one node must be selected for each type (hypervisor, network and controller) at this point in order to continue.

Your license allows a maximum of **70** nodes for OpenStack;

Node selection

H = Hypervisor node (3); N = Network node (3); C = Controller node (1);

H	N	C	**Category: default**
H	N	C	**node001**
H	N	C	**node002**
H	N	C	**node003**

Name: RegEx, e.g.: node00[0-9] Search

Should the wizard reboot the OpenStack nodes as part of the deployment process?
Reboot: ● all nodes ○ only controller nodes

Max reboot wait: 20 minutes

Figure 3.9: OpenStack Nodes Selection

The OpenStack Nodes Selection screen allows the administrator to toggle whether a node takes on the function type of hypervisor node, network node, or controller node.

- A hypervisor node hosts virtual nodes. Typically a hypervisor node has many cores. The more hypervisors there are, the more VMs can be run.
- A network node runs DHCP and legacy routing services. At least one is required, and two are recommended for high availability DHCP and routing for production systems. In the reference architecture (page 15) the set of network nodes is the same as the set of controller nodes. This means that in the reference architecture case each of the controller nodes is running on a machine which is also running a network node within that same machine, which means the resulting hybrid machine can be called a controller/network node. There are therefore 3 controller/network nodes in the reference architecture.

- A controller node runs RabbitMQ services. At least one is required, and three are recommended for high-availability production systems.

Each of these three function types must exist at least once in the cluster. Each node can have multiple functions types, and each function type can be allocated to many nodes. Combining hypervisor nodes with controller nodes is however usually not recommended, due to the high CPU load from controller services.

An often convenient way to set the function types is by category first, and individually next. For example nodes that are to be hypervisors and controllers can have their function type set at category level, by clicking on the category toggles. An individual node in a category can then have a missing function type added to it as a variation on the category-level configuration in this screen.

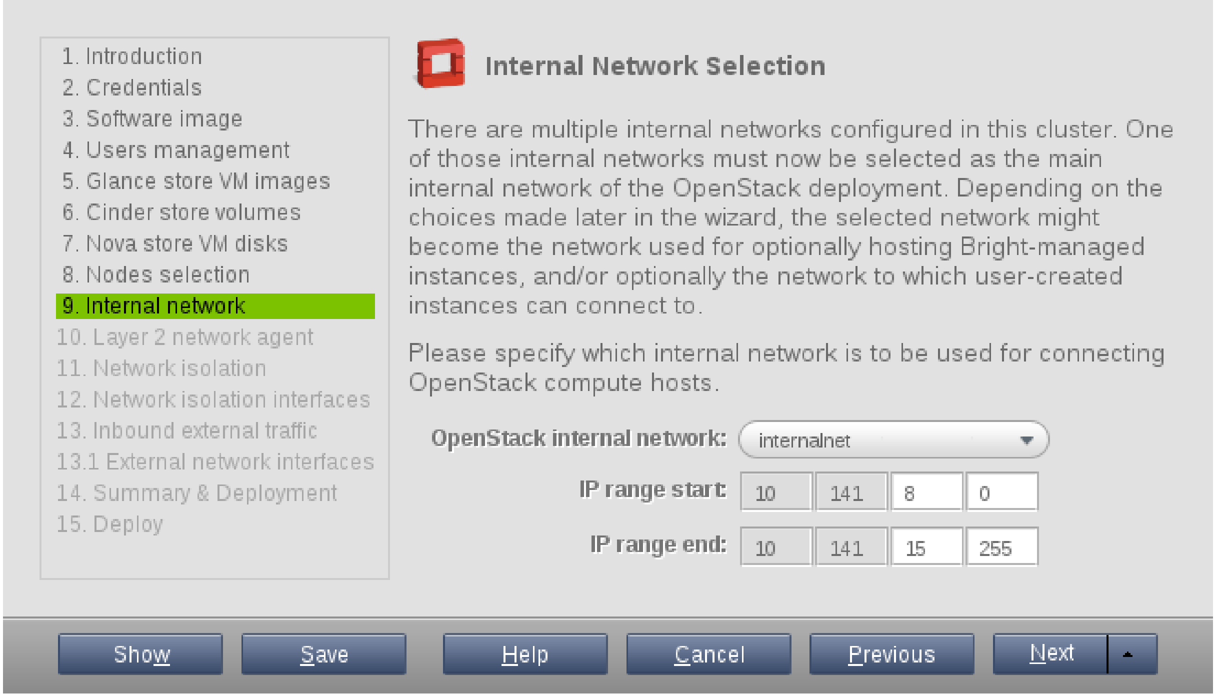
For example, in figure 3.9, the category level has the hypervisor node and network node function types set. This means that node001, node002, and node003 all inherit these function types in their configuration. In addition, node002 has individually had the controller function type added to it.

Within the `OpenStack Nodes Selection` screen, the full list of nodes can be searched through with a regex search. This highlights the searched-for list of nodes.

When the OpenStack installation wizard completes, and configuration is deployed, the OpenStack nodes are all set to reboot by default. However, the `OpenStack Nodes Selection` screen also allows the rebooting of just the controller nodes, which is often sufficient.

When a node reboots, it can take some time to be provisioned. The time to wait for reboot is configurable in the `OpenStack Nodes Selection` screen.

3.1.9 OpenStack Internal Network Selection Screen



Internal Network Selection

There are multiple internal networks configured in this cluster. One of those internal networks must now be selected as the main internal network of the OpenStack deployment. Depending on the choices made later in the wizard, the selected network might become the network used for optionally hosting Bright-managed instances, and/or optionally the network to which user-created instances can connect to.

Please specify which internal network is to be used for connecting OpenStack compute hosts.

OpenStack internal network: internalnet

IP range start: 10 141 8 0

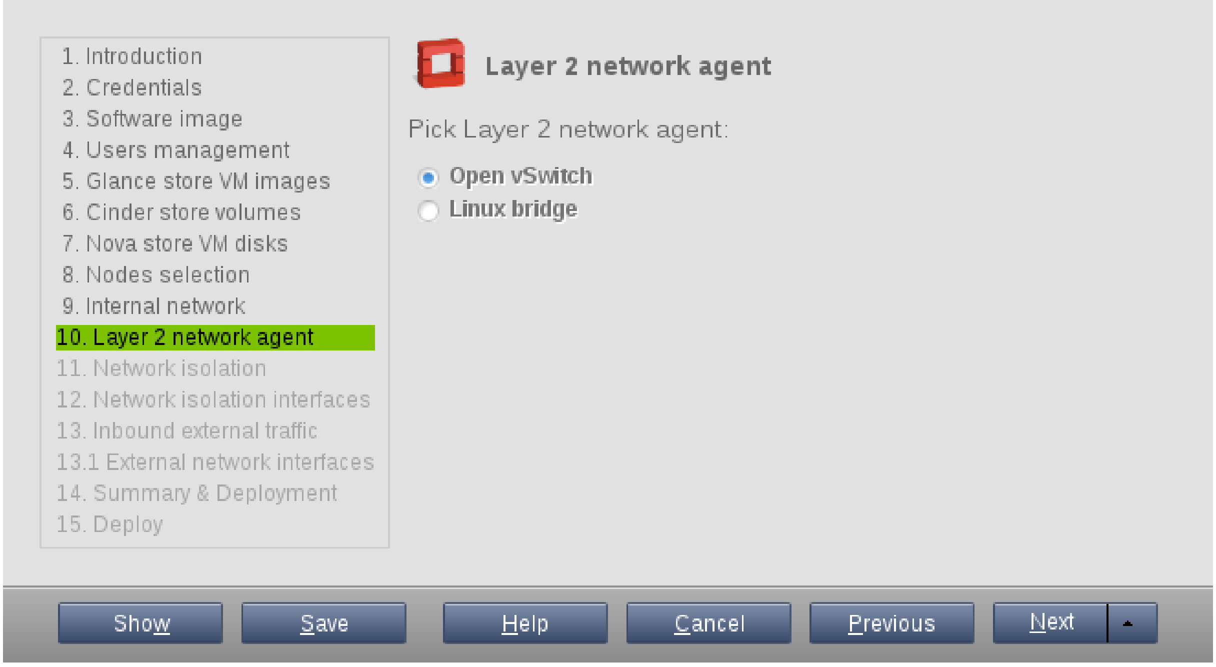
IP range end: 10 141 15 255

Figure 3.10: OpenStack Internal Network Selection

The `OpenStack Internal Network Selection` screen allows the administrator to set the main internal network of the OpenStack nodes. This network is the network that is used to host Bright-managed instances and is also the network that user-created instances can connect to.

By default for a default Bright Cluster Manager installation, `internalnet` is used. A subset of the network is configured for OpenStack use by setting appropriate IP ranges.

3.1.10 OpenStack Layer 2 Network Agent Selection Screen



Layer 2 network agent

Pick Layer 2 network agent:

● Open vSwitch

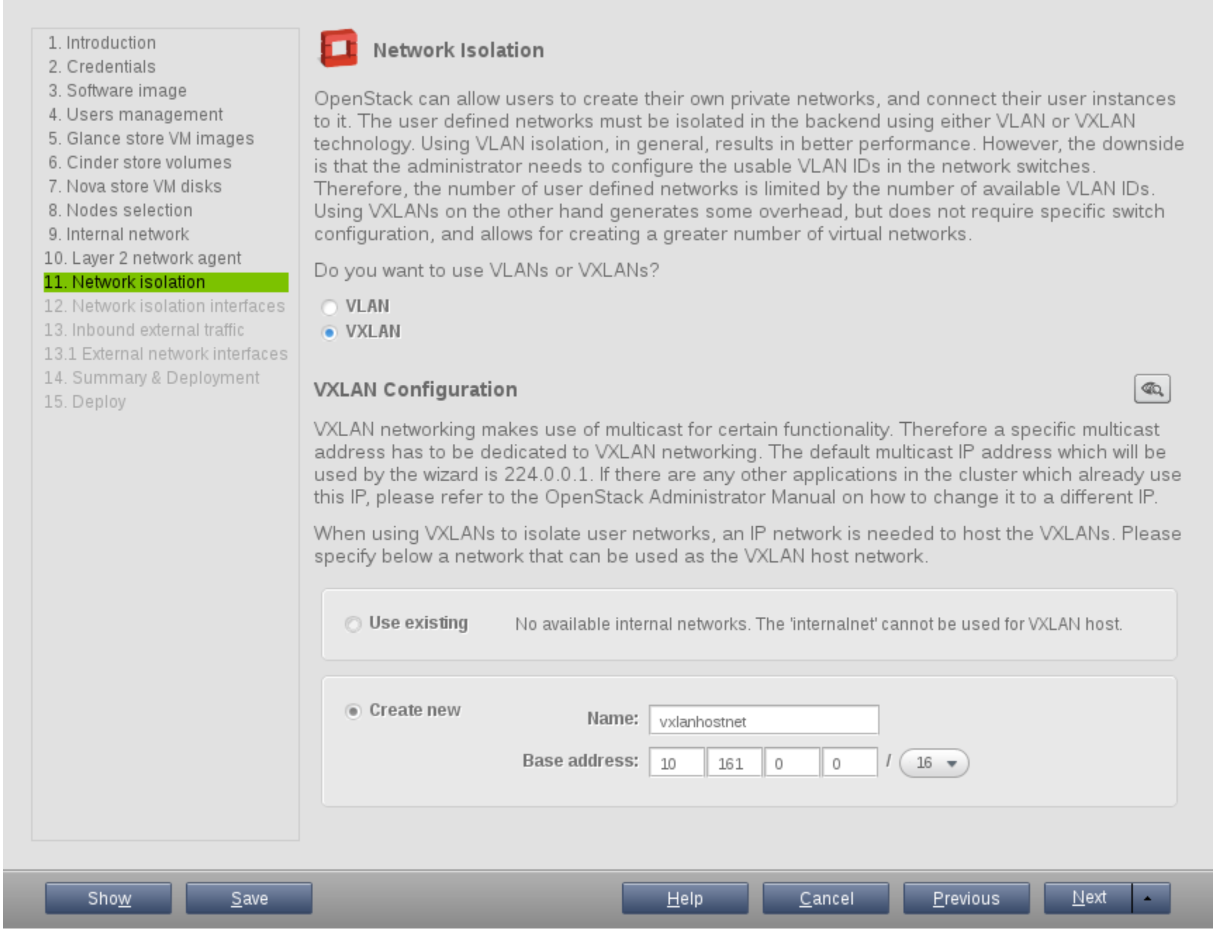
○ Linux bridge

Figure 3.11: OpenStack Layer 2 Network Agent Selection

The OpenStack Layer 2 Network Agent Selection screen allows the administrator to set the network agent that OpenStack is to use for its OSI Layer 2 networking. The two options are:

- Open vSwitch: more complex, and more versatile. It is developing rapidly and is now recommended in preference to Linux bridge networking. A useful feature that Open vSwitch supports, and that Linux Bridge does not, is Distributed Virtual Routers (DVR).
- Linux bridge: simpler, but not as versatile.

3.1.11 OpenStack Network Isolation And VLAN/VXLAN Configuration



Network Isolation

OpenStack can allow users to create their own private networks, and connect their user instances to it. The user defined networks must be isolated in the backend using either VLAN or VXLAN technology. Using VLAN isolation, in general, results in better performance. However, the downside is that the administrator needs to configure the usable VLAN IDs in the network switches. Therefore, the number of user defined networks is limited by the number of available VLAN IDs. Using VXLANs on the other hand generates some overhead, but does not require specific switch configuration, and allows for creating a greater number of virtual networks.

Do you want to use VLANs or VXLANs?

○ VLAN
● VXLAN

VXLAN Configuration

VXLAN networking makes use of multicast for certain functionality. Therefore a specific multicast address has to be dedicated to VXLAN networking. The default multicast IP address which will be used by the wizard is 224.0.0.1. If there are any other applications in the cluster which already use this IP, please refer to the OpenStack Administrator Manual on how to change it to a different IP.

When using VXLANs to isolate user networks, an IP network is needed to host the VXLANs. Please specify below a network that can be used as the VXLAN host network.

○ Use existing   No available internal networks. The 'internalnet' cannot be used for VXLAN host.

● Create new   Name: vxlanhostnet
Base address: 10 161 0 0 / 16

Figure 3.12: OpenStack Network Isolation And VXLAN Configuration Screen

The OpenStack Network Isolation And VXLAN Configuration screen allows the administrator to decide on the network isolation technology that is to be used for the private network of OpenStack user instances. The options, selectable by radio buttons, are either VLANs or VXLANS. Accordingly, a VLAN subscreen, or a closely similar VXLAN subscreen, is then displayed. VXLANs are recommended by default due to their greater ease of use.

VLAN Subscreen

The VLAN range defines the number of user IP networks that can exist at the same time. This must match the VLAN ID configuration on the switch, and can be up to around 4000.

In the VLAN configuration subscreen a network must be selected by:

- either choosing an existing network that has already been configured in Bright Cluster Manager, but not `internalnet`
- or it requires specifying the following, in order to create the network:
  - A new network `Name`: default: `vlanhostnet`
  - VLAN Range start: default: `5`
  - VLAN Range end: default: `100`

VXLAN Subscreen

The VXLAN range defines the number of user IP networks that can exist at the same time. While the range can be set to be around 16 million, it is best to keep it to a more reasonable size, such as 50,000, since a larger range slows down Neutron significantly.

An IP network is needed to host the VXLANs and allow the tunneling of traffic between VXLAN endpoints. This requires

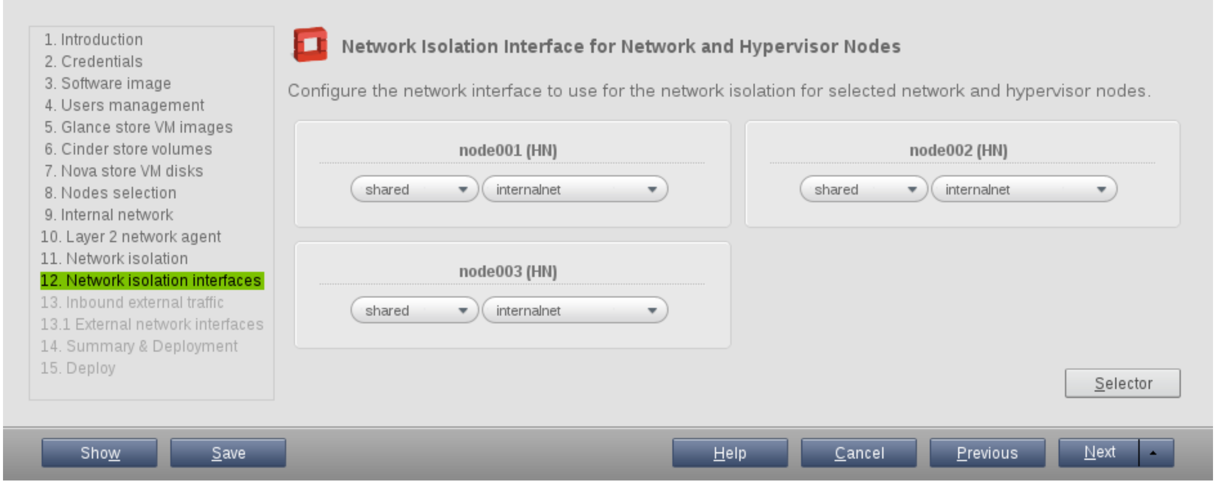
- either choosing an existing network that has already been configured in Bright Cluster Manager, but not `internalnet`
- or it requires specifying the following, in order to create the network:
  - A new network `Name`: default: `vxlanhostnet`
  - `Base address`: default: `10.161.0.0`
  - `Netmask bits`: default: `16`

In the VXLAN configuration subscreen, if the icon to view details is clicked, then the following extra options are suggested, with overrideable defaults as listed:

- `VXLAN Range start`: default: `1`
- `VXLAN Range end`: default: `50000`

VXLAN networking uses a multicast address to handle broadcast traffic in a virtual network. The default multicast IP address that is set, `224.0.0.1`, is unlikely to be used by another application. However, if there is a conflict, then the address can be changed using the CMDaemon `OpenStackVXLANGroup` directive (Appendix C, page 604 of the *Administrator Manual*).

3.1.12 OpenStack Network Isolation interface For Network And Hypervisor Nodes



Network Isolation Interface for Network and Hypervisor Nodes

Configure the network interface to use for the network isolation for selected network and hypervisor nodes.

node001 (HN): shared, internalnet

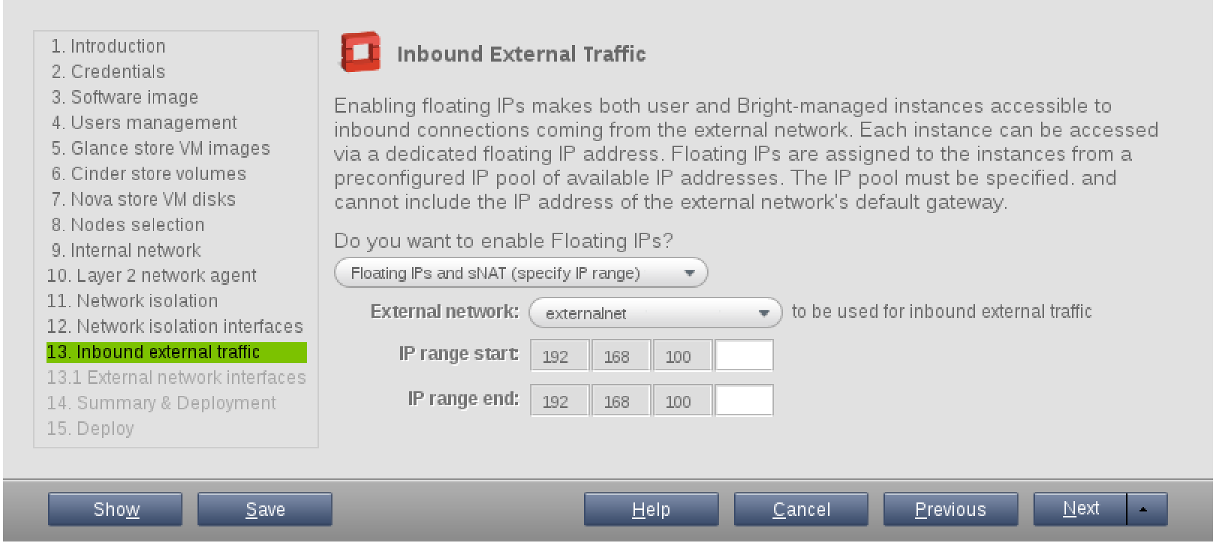
node002 (HN): shared, internalnet

node003 (HN): shared, internalnet

Figure 3.13: OpenStack Network Isolation interface For Network And Hypervisor Nodes Screen

The `Network Isolation interface For Network And Hypervisor Nodes` screen (figure 3.13) sets the network that will be used for the network nodes and hypervisor nodes. These are classed according to whether the network will be shared or dedicated, and the `Selector` button allows advanced filtering, which is useful when dealing with a large number of nodes.

3.1.13 OpenStack Inbound External Traffic



Inbound External Traffic

Enabling floating IPs makes both user and Bright-managed instances accessible to inbound connections coming from the external network. Each instance can be accessed via a dedicated floating IP address. Floating IPs are assigned to the instances from a preconfigured IP pool of available IP addresses. The IP pool must be specified. and cannot include the IP address of the external network's default gateway.

Do you want to enable Floating IPs?

Floating IPs and sNAT (specify IP range)

External network: externalnet to be used for inbound external traffic

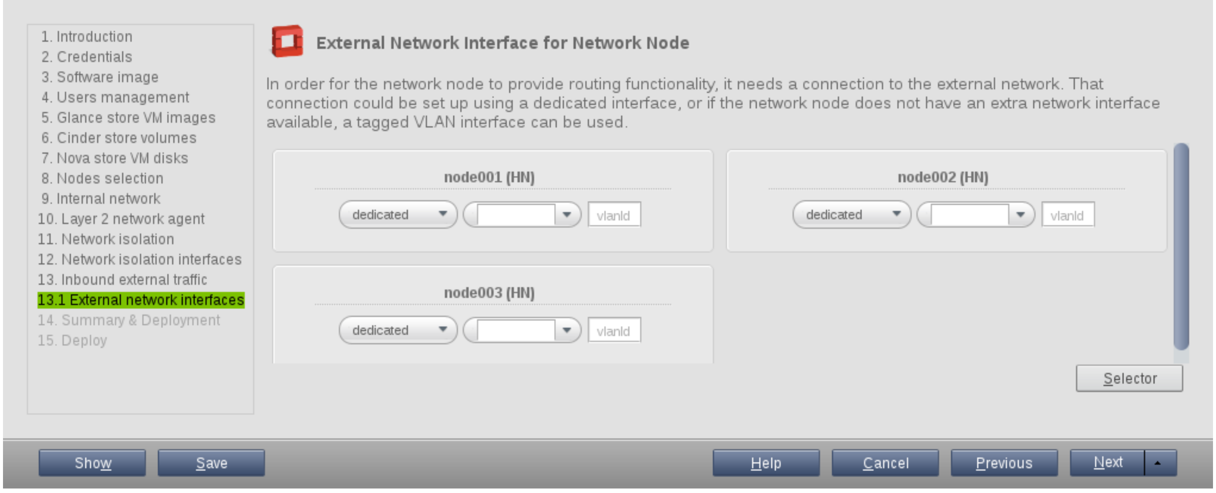
IP range start: 192 168 100

IP range end: 192 168 100

Figure 3.14: OpenStack Inbound External Traffic Screen

The `OpenStack Inbound External Traffic` screen (figure 3.14) allows the administrator to set floating IP addresses. A floating IP address is an address on the external network that is associated with an OpenStack instance. The addresses "float" because they are assigned from an available pool of addresses, to the instance, when the instance requests an address.

3.1.14 OpenStack External Network Interface For Network Node



External Network Interface for Network Node

In order for the network node to provide routing functionality, it needs a connection to the external network. That connection could be set up using a dedicated interface, or if the network node does not have an extra network interface available, a tagged VLAN interface can be used.

node001 (HN): dedicated vlanId

node002 (HN): dedicated vlanId

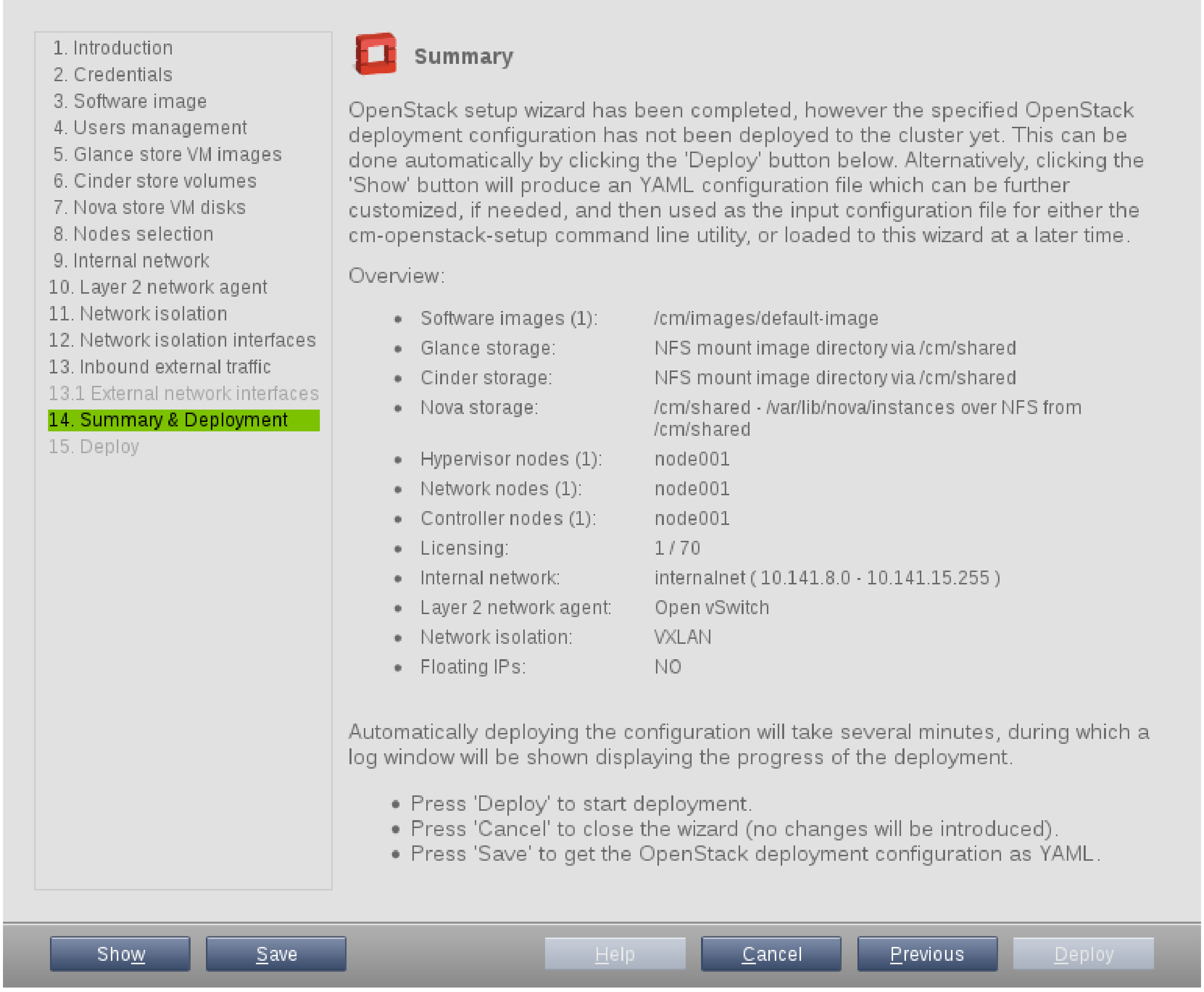
node003 (HN): dedicated vlanId

Figure 3.15: OpenStack External Network Interface For Network Node Screen

The `OpenStack External Network Interface For Network Node` screen (figure 3.15) allows the administrator to provide routing between the external network and the network nodes. It can be set up on a dedicated interface. If no spare interface is available on the network node, then if the switch supports it, a tagged VLAN interface can be configured instead.

The `Selector` button allows advanced filtering, which is useful when dealing with a large number of nodes.

3.1.15 Summary



Summary

OpenStack setup wizard has been completed, however the specified OpenStack deployment configuration has not been deployed to the cluster yet. This can be done automatically by clicking the 'Deploy' button below. Alternatively, clicking the 'Show' button will produce an YAML configuration file which can be further customized, if needed, and then used as the input configuration file for either the cm-openstack-setup command line utility, or loaded to this wizard at a later time.

Overview:

Software images (1):	/cm/images/default-image
Glance storage:	NFS mount image directory via /cm/shared
Cinder storage:	NFS mount image directory via /cm/shared
Nova storage:	/cm/shared - /var/lib/nova/instances over NFS from /cm/shared
Hypervisor nodes (1):	node001
Network nodes (1):	node001
Controller nodes (1):	node001
Licensing:	1 / 70
Internal network:	internalnet (10.141.8.0 - 10.141.15.255)
Layer 2 network agent:	Open vSwitch
Network isolation:	VXLAN
Floating IPs:	NO

Automatically deploying the configuration will take several minutes, during which a log window will be shown displaying the progress of the deployment.

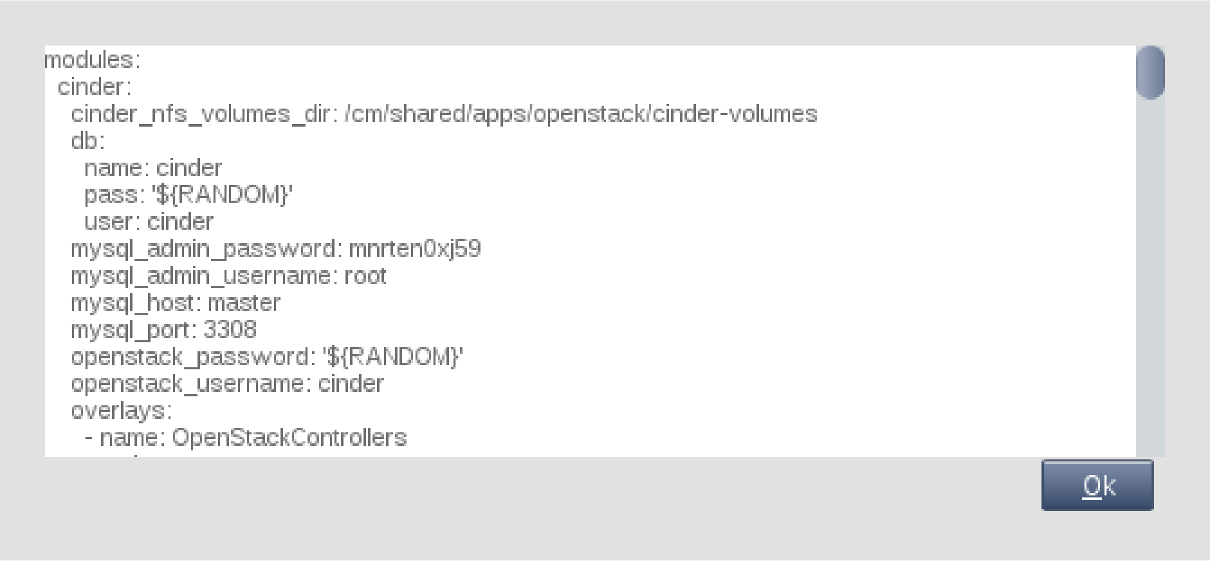
- Press 'Deploy' to start deployment.
- Press 'Cancel' to close the wizard (no changes will be introduced).
- Press 'Save' to get the OpenStack deployment configuration as YAML.

Figure 3.16: Summary Screen

Viewing And Saving The Configuration

The summary screen (figure 3.16) gives a summary of the configuration. The configuration can be changed in cmgui if the administrator goes back through the screens to adjust settings.

The full configuration is kept in an YAML file, which can be viewed by clicking on the Show button. The resulting read-only view is shown in figure 3.17.



```
modules:
  cinder:
    cinder_nfs_volumes_dir: /cm/shared/apps/openstack/cinder-volumes
    db:
      name: cinder
      pass: '${RANDOM}'
      user: cinder
    mysql_admin_password: mnrten0xj59
    mysql_admin_username: root
    mysql_host: master
    mysql_port: 3308
    openstack_password: '${RANDOM}'
    openstack_username: cinder
    overlays:
      - name: OpenStackControllers
```

Figure 3.17: OpenStack Configuration Screen

The configuration can be saved with the Save button of figure 3.16.

After exiting the wizard, the YAML file can be directly modified if needed in a separate text-based editor.

Using A Saved Configuration And Deploying The Configuration

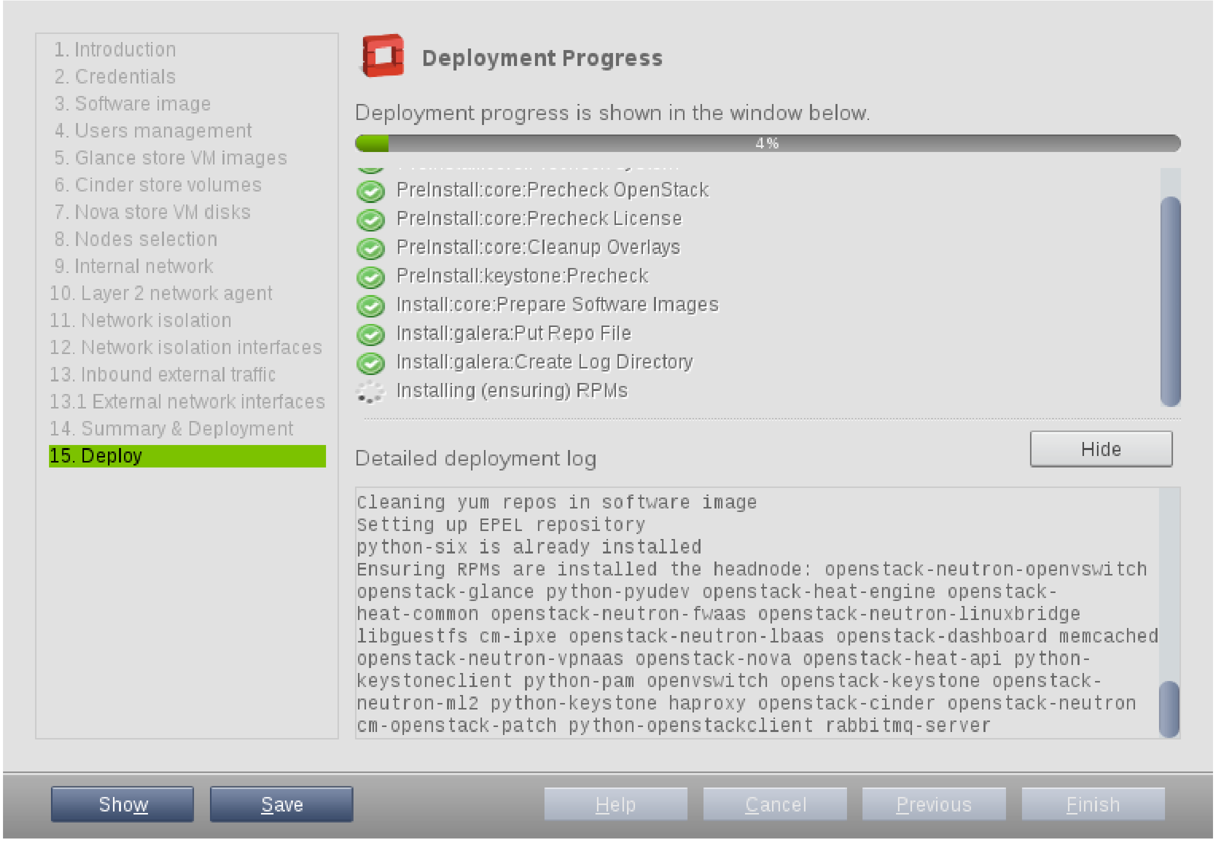
Using a saved YAML file is possible.

- The YAML file can be used as the configuration starting point for the text-based cm-openstack-setup utility (section 3.2), if run as:

  ```
  [root@bright73~]# cm-openstack-setup -c <YAML file>
  ```

- Alternatively, the YAML file can be deployed as the configuration by launching the cmgui wizard, and then clicking on the Load button of the first screen (figure 3.2). After loading the configuration, a Deploy button appears.

Clicking the Deploy button that appears in figure 3.2 after loading the YAML file, or clicking the Deploy button of figure 3.16, sets up OpenStack in the background. The direct background progress is hidden from the administrator, and relies on the text-based cm-openstack-setup script (section 3.2). Some log excerpts from the script can be displayed within a Deployment Progress window (figure 3.18).



Deployment progress is shown in the window below.

```
PreInstall:core:Precheck OpenStack
PreInstall:core:Precheck License
PreInstall:core:Cleanup Overlays
PreInstall:keystone:Precheck
Install:core:Prepare Software Images
Install:galera:Put Repo File
Install:galera:Create Log Directory
Installing (ensuring) RPMs
```

```
Cleaning yum repos in software image
Setting up EPEL repository
python-six is already installed
Ensuring RPMs are installed the headnode: openstack-neutron-openvswitch
openstack-glance python-pyudev openstack-heat-engine openstack-
heat-common openstack-neutron-fwaas openstack-neutron-linuxbridge
libguestfs cm-ipxe openstack-neutron-lbaas openstack-dashboard memcached
openstack-neutron-vpnaas openstack-nova openstack-heat-api python-
keystoneclient python-pam openvswitch openstack-keystone openstack-
neutron-ml2 python-keystone haproxy openstack-cinder openstack-neutron
cm-openstack-patch python-openstackclient rabbitmq-server
```

Figure 3.18: OpenStack Deployment Progress Screen

At the end of its run, the cluster has OpenStack set up and running in an integrated manner with Bright Cluster Manager.

The administrator can now configure the cluster to suit the particular site requirements.

3.2 Installation Of OpenStack From The Shell

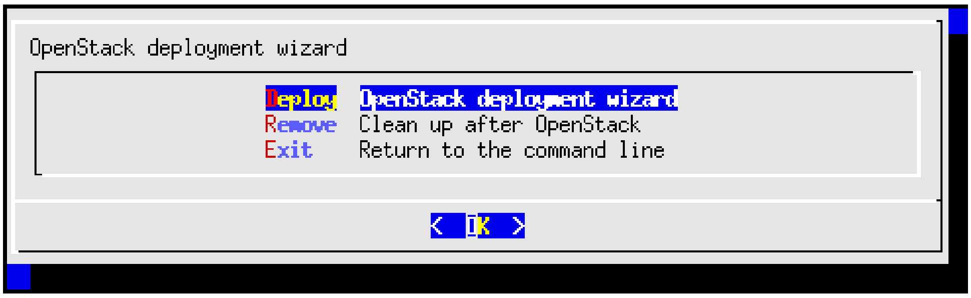
The `cmgui` OpenStack installation (section 3.1) uses the `cm-openstack-setup` utility during deployment, hidden from normal view. The installation can also be done directly with `cm-openstack-setup`. The `cm-openstack-setup` utility is a less-preferred alternative to the installation of OpenStack from `cmgui`.

The `cm-openstack-setup` utility is a part of the standard `cluster-tools` package. Details on its use are given in its manual page (`man (8) cm-openstack-setup`). When run, the regular nodes that are to run OpenStack instances are rebooted by default at the end of the dialogs, in order to deploy them.

A prerequisite for running `cm-openstack-setup` is that the head node should be connected to the distribution repositories.

A sample `cm-openstack-setup` wizard session is described next, starting from section 3.2.1. The session runs on a cluster consisting of one head node and one regular node. The wizard can be interrupted gracefully with a <`ctrl-c`>.

3.2.1 Start Screen



```
OpenStack deployment wizard

        Deploy  OpenStack deployment wizard
        Remove  Clean up after OpenStack
        Exit    Return to the command line
```

Figure 3.19: Start Screen

The start screen (figure 3.19) lets the administrator:

- deploy Bright Cluster Manager OpenStack.
- remove Bright Cluster Manager's OpenStack if it is already on the cluster.
- exit the installation.

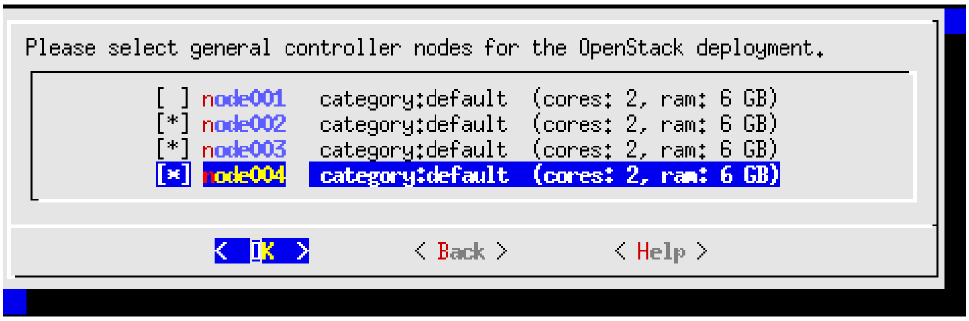
Removal removes OpenStack-related database entries, roles, networks, virtual nodes, and interfaces. Images and categories related to OpenStack are however not removed.

A shortcut to carry out a removal from the shell prompt is to run `cm-openstack-setup --remove`. The `preventremoval` setting must be set to `no` for this to work:

Example

```
[root@bright73 ~]# cmsh
[bright73]% openstack
[bright73->openstack[default]]% set preventremoval no; commit; quit
[root@bright73 ~]# cm-openstack-setup --remove
Please wait...
Connecting to CMDaemon
###### WARNING: Setup will attempt to remove the following objects:
...
```

3.2.2 Controller Node Selection



```
Please select general controller nodes for the OpenStack deployment.

          [ ] node001   category:default  (cores: 2, ram: 6 GB)
          [*] node002   category:default  (cores: 2, ram: 6 GB)
          [*] node003   category:default  (cores: 2, ram: 6 GB)
          [*] node004   category:default  (cores: 2, ram: 6 GB)
```

Figure 3.20: Controller Nodes Selection

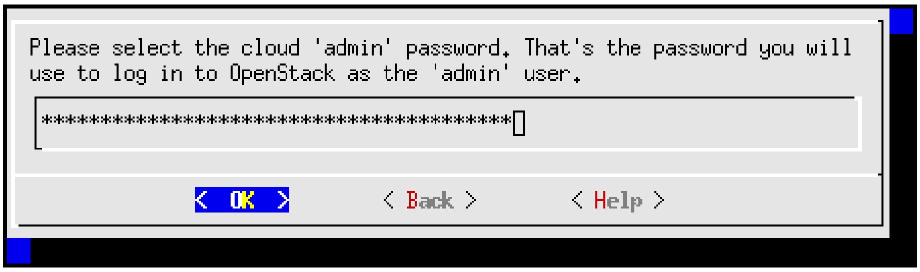
The controller nodes selection screen (figure 3.20) allows the selection of nodes on which the following services are to run:

- the OpenStack database service
- the OpenStack core services. The core OpenStack services in Mitaka are

- Nova (compute)
- Neutron (networking)
- Swift (object storage)—not deployed by default in Bright Cluster Manager OpenStack edition
- Cinder (block storage)
- Keystone (identity)
- Glance (image service)

Each controller node is required to have a minimum of 2 cores, and 8GB of RAM is recommended.

3.2.3 Setting The Cloud `admin` Password

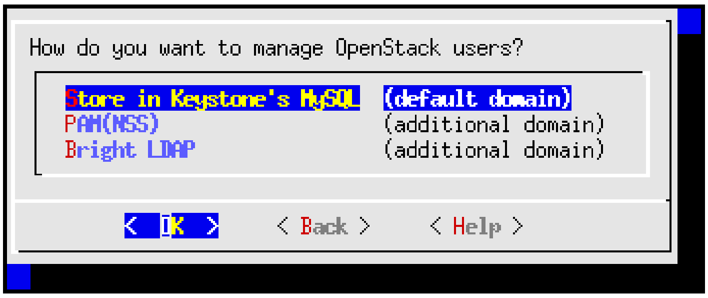


```
Please select the cloud 'admin' password. That's the password you will
use to log in to OpenStack as the 'admin' user.
****************************************
```

Figure 3.21: Cloud `admin` Password Screen

The OpenStack cloud `admin` password screen (figure 3.21) prompts for a password to be entered, and then re-entered, for the soon-to-be-created `admin` user of OpenStack. The `admin` user is mandatory. The password can be changed after deployment.

3.2.4 User Management Configuration Of OpenStack Users



```
How do you want to manage OpenStack users?
  Store in Keystone's MySQL   (default domain)
  PAM(NSS)                    (additional domain)
  Bright LDAP                 (additional domain)
```

Figure 3.22: User Management Configuration Of OpenStack Users Screen

The user management configuration of OpenStack users screen (figure 3.22) allows the administrator to choose how OpenStack users are to be managed. Options are:

- Managing via Keystone MySQL (default domain)
- Managing via PAM(NSS)
- Using LDAPS as provided by Bright Cluster Manager

Managing via Keystone's MySQL means that OpenStack users, in the default OpenStack domain, are independent of the pre-existing Bright Cluster Manager users.

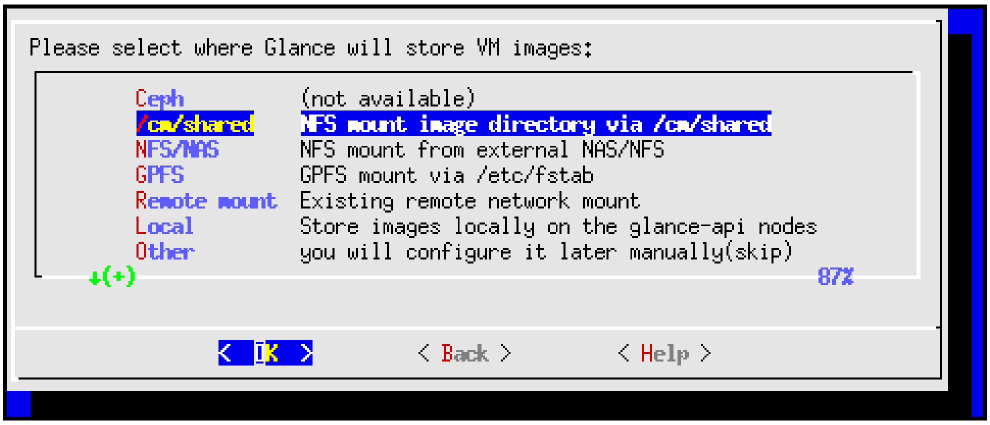
Managing via PAM(NSS) additionally allows Keystone to use PAM as an identity backend for additional domains. For external identity authentication, PAM(NSS) can be run in a read-only mode.

Managing via Bright Cluster Manager's LDAPS means that OpenStack users, stored in the default OpenStack domain, and independent of the pre-existing Bright Cluster Manager users, are used, and Bright Cluster Manager users are also visible to Keystone, via a read-only access.

3.2.5 Ceph And Other Storage Options

This section (3.2.5) covers the Ncurses `cm-openstack-setup` wizard configuration of Ceph options.

Glance VM Image Storage



```
Please select where Glance will store VM images:

        Ceph          (not available)
        /cm/shared    NFS mount image directory via /cm/shared
        NFS/NAS       NFS mount from external NAS/NFS
        GPFS          GPFS mount via /etc/fstab
        Remote mount  Existing remote network mount
        Local         Store images locally on the glance-api nodes
        Other         you will configure it later manually(skip)
     ↓(+)                                                  87%

          <  OK  >        < Back >        < Help >
```

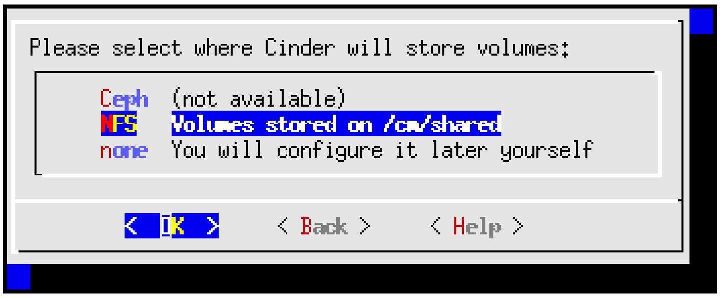
Figure 3.23: Image Storage Options

The image storage screen (figure 3.23) can be used to set the virtual machine storage used.

The storage options are:

- `Ceph` - This is only available as an image storage option, if set up as in Chapter 4.
- `/cm/shared` - The standard Bright Cluster Manager shared NFS directory
- `NFS/NAS` - An external NAS NFS directory
- `GPFS` - A GPFS mount as defined in the `/etc/fstab` configuration.
- `Remote mount` - An existing remote network mount
- `Local` - Images are stored locally on Glance API nodes.
- `Other` - to be configured later (skips this screen)
- `More` - Other backends that are not listed in this menu

Cinder Volume Storage



```
Please select where Cinder will store volumes:

      Ceph  (not available)
      NFS   Volumes stored on /cm/shared
      none  You will configure it later yourself

     <  OK  >     < Back >     < Help >
```

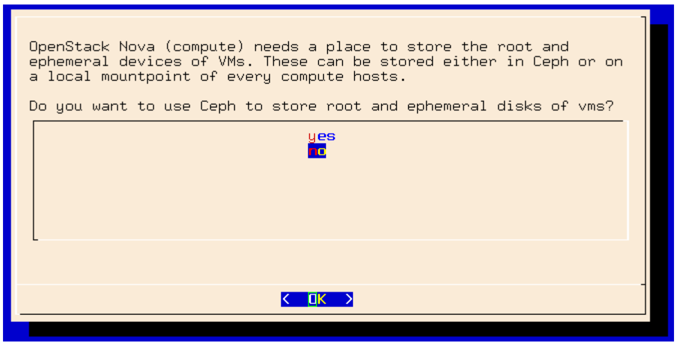
Figure 3.24: Volume Storage Options

The OpenStack Cinder volume storage screen (figure 3.24) allows the setting of persistent block volume read and write storage.

The storage options are:

- Ceph - This is only available as a volume storage option, if set up as in Chapter 4. If set, it uses Ceph's RBD volume driver, and configures a "volume backup" driver to use Ceph.
- NFS - Storage is done on /cm/shared using the Cinder reference driver. This is not recommended for large-scale production use.
- None - to be configured later (skips this screen)

Root And Ephemeral Device Storage With Ceph

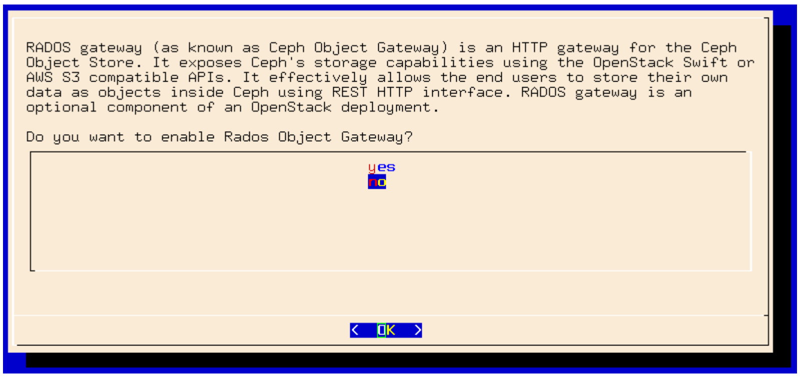


```
OpenStack Nova (compute) needs a place to store the root and
ephemeral devices of VMs. These can be stored either in Ceph or on
a local mountpoint of every compute hosts.

Do you want to use Ceph to store root and ephemeral disks of vms?
```

Figure 3.25: Root And Ephemeral Device Storage With Ceph

Data storage with Ceph can be enabled by the administrator by using the Ceph for OpenStack root and ephemeral device storage screen (figure 3.25).

Ceph Object Gateway (Ceph RADOS Gateway)

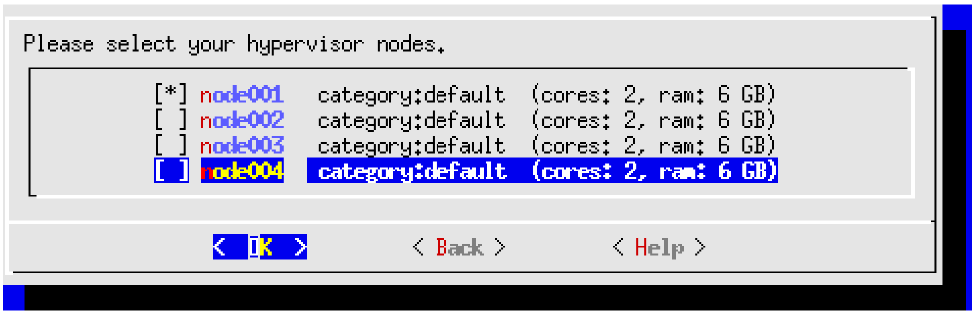


```
RADOS gateway (as known as Ceph Object Gateway) is an HTTP gateway for the Ceph
Object Store. It exposes Ceph's storage capabilities using the OpenStack Swift or
AWS S3 compatible APIs. It effectively allows the end users to store their own
data as objects inside Ceph using REST HTTP interface. RADOS gateway is an
optional component of an OpenStack deployment.

Do you want to enable Rados Object Gateway?
```

Figure 3.26: Root And Ephemeral Device Storage With Ceph

The Ceph RADOS gateway screen (figure 3.26) lets the administrator set the Ceph RADOS gateway service to run when deployment completes.

3.2.6 Hypervisor Nodes Selection For OpenStack

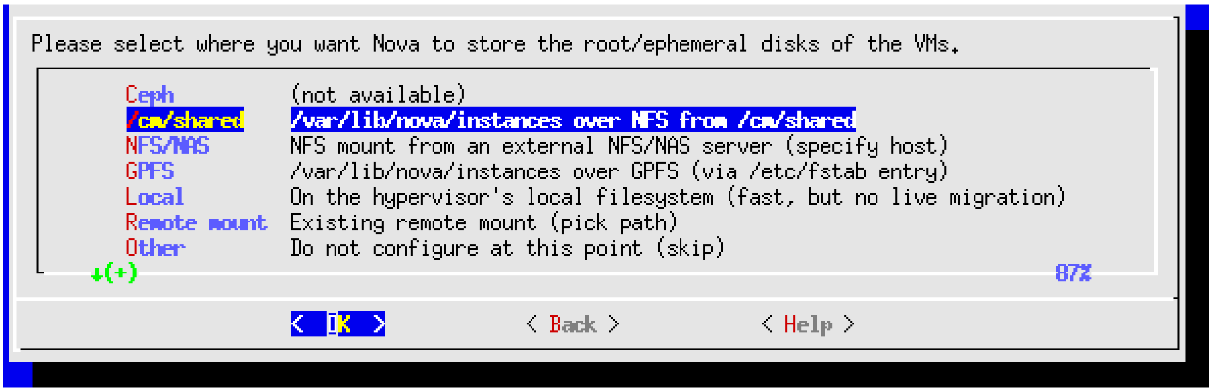


```
Please select your hypervisor nodes.

    [*] node001   category:default  (cores: 2, ram: 6 GB)
    [ ] node002   category:default  (cores: 2, ram: 6 GB)
    [ ] node003   category:default  (cores: 2, ram: 6 GB)
    [ ] node004   category:default  (cores: 2, ram: 6 GB)

         < OK >         < Back >         < Help >
```

Figure 3.27: Hypervisors To Be Used For OpenStack

The hypervisor nodes selection screen (figure 3.27) lets the administrator set the nodes that will be hypervisors. These are the machines that host the compute nodes, and which are assigned the `OpenStackNovaCompute` role. The set of nodes can be changed on a cluster later on, by managing the node list of the `OpenStackHyperVisors` configuration overlay.

3.2.7 VM Root/Ephemeral Disk Storage



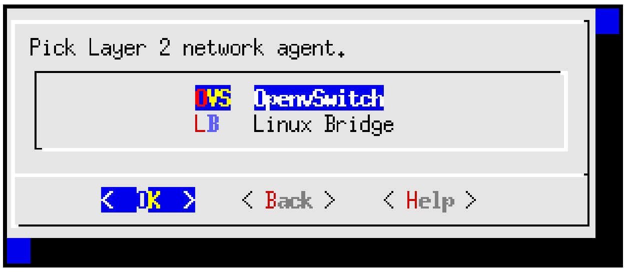
```
Please select where you want Nova to store the root/ephemeral disks of the VMs.

    Ceph          (not available)
    /cm/shared    /var/lib/nova/instances over NFS from /cm/shared
    NFS/NAS       NFS mount from an external NFS/NAS server (specify host)
    GPFS          /var/lib/nova/instances over GPFS (via /etc/fstab entry)
    Local         On the hypervisor's local filesystem (fast, but no live migration)
    Remote mount  Existing remote mount (pick path)
    Other         Do not configure at this point (skip)
 ↓(+)                                                                 87%

         < OK >         < Back >         < Help >
```

Figure 3.28: Setting Root/Ephemeral VM Disk Storage Location

The VM root/ephemeral disk storage screen allows the administrator to tell Nova where to store the root/ephemeral disks. The options are

- `Ceph`: This option is available if Ceph has been configured. By default, `/var/lib/nova/instances` is used.
- `/cm/shared`: The disks can be stored on the hypervisor nodes under the NFS shared directory in `/var/lib/nova/instances`
- `NFS/NAS`: An external NFS/NAS host can be used
- `GPFS`: The disks can be stored on the hypervisor nodes via a GPFS directory specified for `/var/lib/nova/instances` in `/etc/fstab`
- `Local`: The disks can be stored on the local filesystem of the hypervisor. This is avoids network lag, but also does not permit migration.
- `Remote mount`: A path to an existing remount mount point.
- `Other`: Skip (configure later maybe)
- `More`: Suggests alternatives

3.2.8 OpenStack Layer 2 Network Agent Selection Screen



Pick Layer 2 network agent.

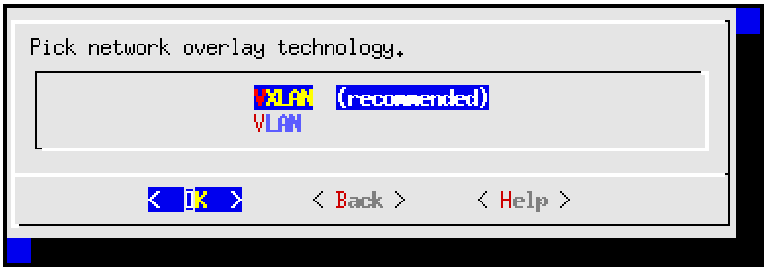
OVS OpenvSwitch
LB Linux Bridge

Figure 3.29: Layer 2 Network Agent Selection For OpenStack

The OSI layer 2 Network Agent Selection screen (figure 3.29) allows the administrator to choose the network agent that OpenStack is to use for its OSI Layer 2 networking. The two options are:

- Open vSwitch: more complex, and more versatile. It is developing rapidly and is now recommended in preference to Linux bridge networking. A useful feature that Open vSwitch supports, and that Linux Bridge does not, is Distributed Virtual Routers (DVR).
- Linux bridge: simpler, but not as versatile.

3.2.9 Network Overlay Technology Used For OpenStack



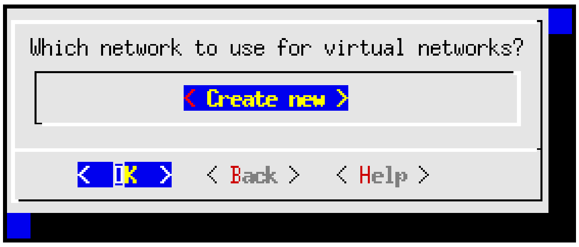
Pick network overlay technology.

VXLAN (recommended)
VLAN

Figure 3.30: Network Overlay Used For OpenStack

The network overlay technology screen (figure 3.30) allows the administrator to choose what kind of network isolation type should be set for the user networks.

3.2.10 Setting The Virtual Network Name

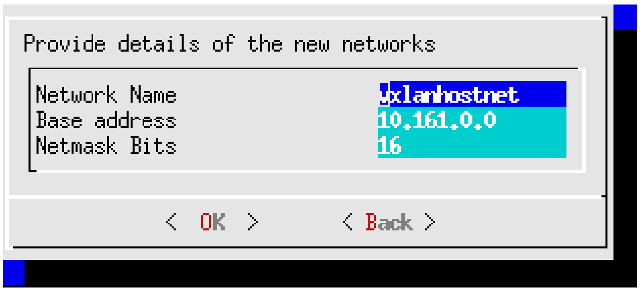


Which network to use for virtual networks?

Figure 3.31: Creating The Virtual Network

The virtual network is the hosting network for OpenStack end user networks. The virtual networks screen (figure 3.31) allows the administrator to configure a virtual network to host the end user networks. By default, if needed, the network to be created is named `vlanhostnet` for a VLAN network, and `vxlanhostnet` for a VXLAN network. An existing VXLAN or VLAN network can be selected.

3.2.11 Setting The Network Details For The Virtual Network

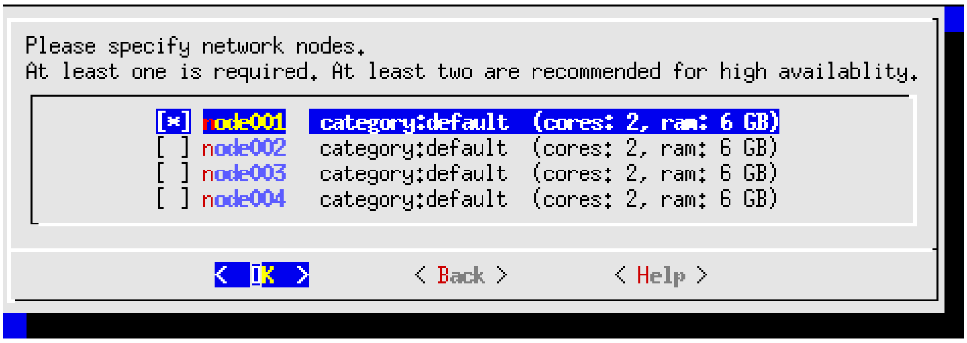


```
Provide details of the new networks

Network Name              vxlanhostnet
Base address              10.161.0.0
Netmask Bits              16
```

Figure 3.32: Setting The Network Details Of The Virtual Network

The virtual network is the hosting network for end user networks. If it does not have its details configured as yet, then the network details screen (figure 3.32) allows the administrator to set the base address and netmask bits for the virtual network.

3.2.12 Setting The Network Nodes



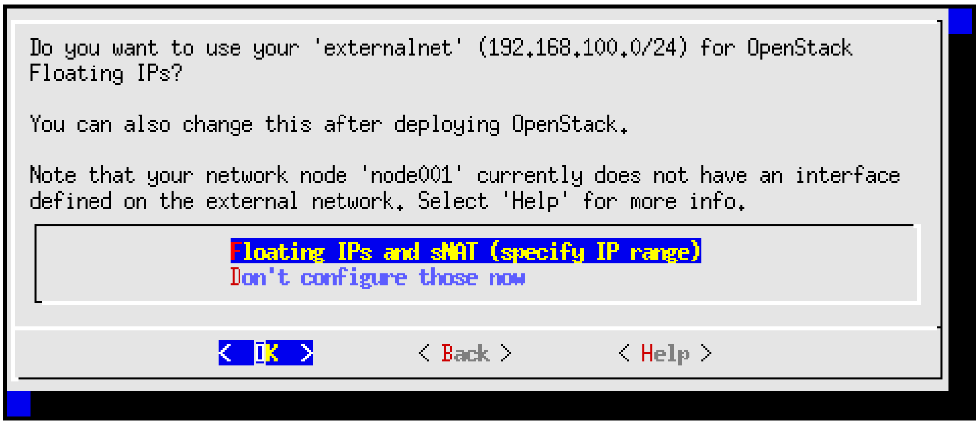
```
Please specify network nodes.
At least one is required. At least two are recommended for high availablity.

[*] node001  category:default  (cores: 2, ram: 6 GB)
[ ] node002  category:default  (cores: 2, ram: 6 GB)
[ ] node003  category:default  (cores: 2, ram: 6 GB)
[ ] node004  category:default  (cores: 2, ram: 6 GB)
```

Figure 3.33: Setting The Network Nodes

The network node selection screen (figure 3.33) allows the administrator to set network nodes. The network nodes run OpenStack networking components from OpenStack neutron. A reasonable rule-of-thumb is to have 1 network node per 10 hypervisor nodes. Network nodes and compute nodes can be combined.

To use Floating IPs or sNAT, network nodes must be connected to the external network.

3.2.13 Floating IPs And sNAT



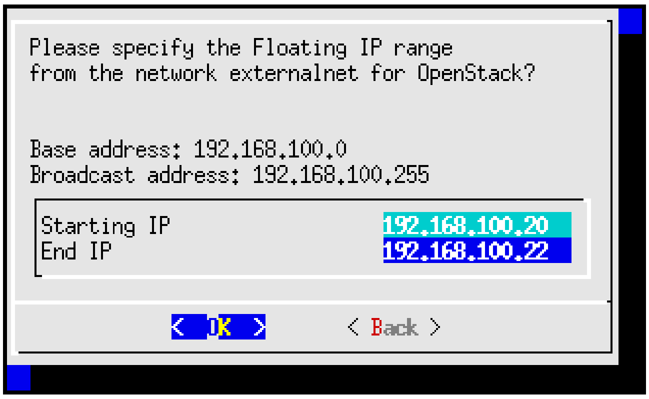
```
Do you want to use your 'externalnet' (192.168.100.0/24) for OpenStack
Floating IPs?

You can also change this after deploying OpenStack.

Note that your network node 'node001' currently does not have an interface
defined on the external network. Select 'Help' for more info.

                 Floating IPs and sNAT (specify IP range)
                 Don't configure those now

              < OK >          < Back >          < Help >
```

Figure 3.34: Floating IPs

The floating IPs screen (figure 3.34) lets the administrator allow floating IPs to be configured on the external network. This allows instances within OpenStack to be accessed from the external network. Floating IPs can also be configured after OpenStack has been set up.

A note is shown in the dialog if the network node does not have an external network interface. Creating the external network interface is possible at this point or later, using cmsh for example.

3.2.14 External Network Floating IP Range

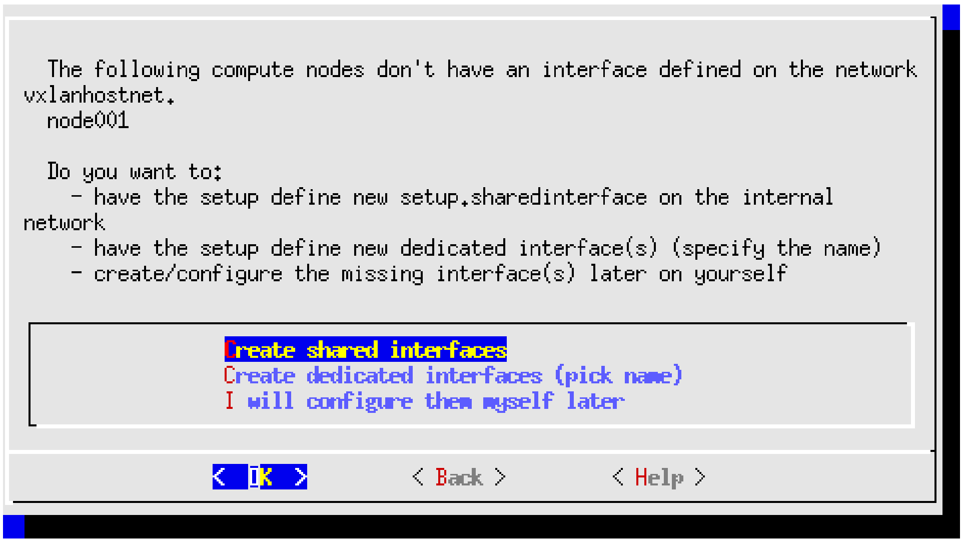


```
Please specify the Floating IP range
from the network externalnet for OpenStack?


Base address: 192.168.100.0
Broadcast address: 192.168.100.255

Starting IP                       192.168.100.20
End IP                            192.168.100.22

              < OK >          < Back >
```

Figure 3.35: External Network: Floating IP Range

If floating IPs are to be configured by the wizard, then the floating IP range screen figure 3.35 allows the administrator to specify the floating IP address range on the external network.

3.2.15 External Network Interface Creation



```
  The following compute nodes don't have an interface defined on the network
vxlanhostnet.
  node001

  Do you want to:
    - have the setup define new setup.sharedinterface on the internal
network
    - have the setup define new dedicated interface(s) (specify the name)
    - create/configure the missing interface(s) later on yourself
```

Create shared interfaces
Create dedicated interfaces (pick name)
I will configure them myself later
< OK >
< Back >
< Help >

Figure 3.36: External Network: Interface Creation

If floating IPs are to be configured by the wizard, then the external network interface creation screen (figure 3.36) allows the administrator to create a network interface. The interface is created on each network node that is missing an interface to the external network.

The interface can be

- a shared interface: this uses the internal network for virtual networking
- a dedicated interface: this uses a dedicated network with its associated dedicated interface. The device must exist on the network node in order for the interface to be created.

The interface creation step can be skipped and carried out after OpenStack deployment, but OpenStack may not run properly because of this. Alternatively, if each network node has special needs, then each interface can be set up before running the wizard.

3.2.16 Saving The Configuration

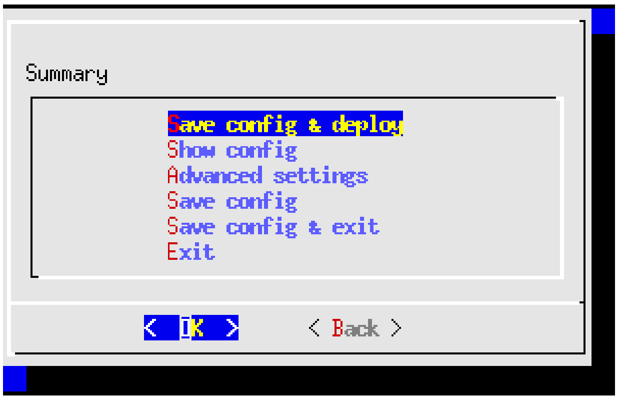


Figure 3.37: Viewing And Saving The Configuration

The screen for saving the configuration (figure 3.37) allows the administrator to view the configuration with the Show option. The configuration that has been prepared by the wizard can be seen with the Show config option, and using the <Page Up> and <Page Down> keys to scroll up and down.

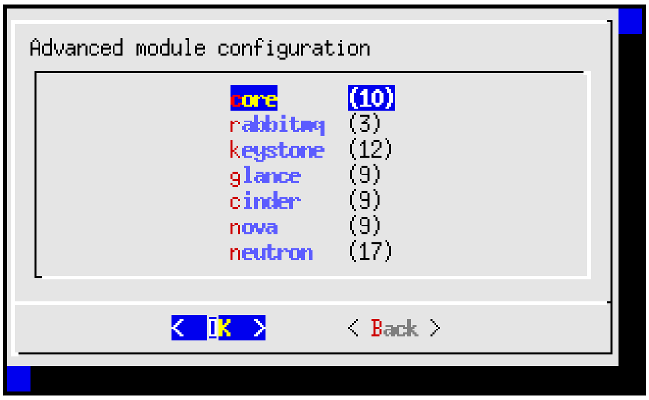
The configuration options can also be saved with the various save options:

- `Save config & deploy`: Saves, and after saving carries out the text-based deployment stage of the installation.
- `Save`: Saves, and stays within the Ncurses dialog. The deployment can be carried out later from a saved configuration.
- `Save config & exit`: Saves, and then exits the Ncurses dialog. The deployment can be carried out later from a saved configuration.

Saving saves the configration as a YAML configuration file, by default `cm-openstack-setup.conf`, in the directory under which the wizard is running. This file can be used as the input configuration file for the `cm-openstack-setup` utility using the `-c` option.

Most administrators run `Save config & deploy`, and the deployment run takes place (section 3.2.17). Some administrators may however wish to modify some OpenStack component settings.

The OpenStack Components Advanced Settings Screens

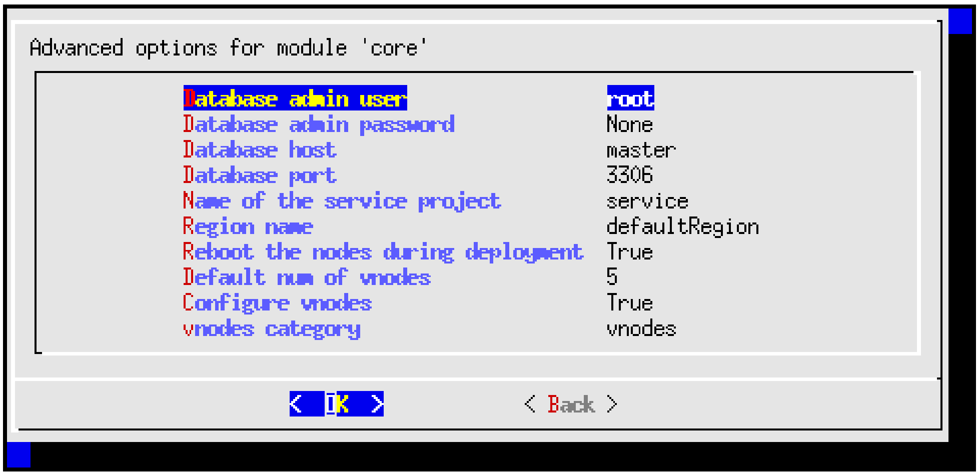


```
Advanced module configuration

        core       (10)
        rabbitmq   (3)
        keystone   (12)
        glance     (9)
        cinder     (9)
        nova       (9)
        neutron    (17)
```

Figure 3.38: Advanced Options

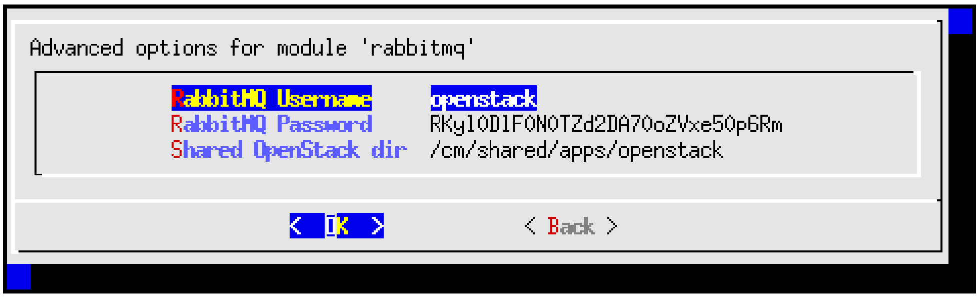
The advanced settings screen (figure 3.38) allows an administrator to set up OpenStack components with some advanced options. For example, values for the passwords and ports used by various OpenStack services can be modified. These values can also be altered from within `cmsh` after deployment.

The components that can be dealt with in the advanced settings screen are `core`, `rabbitmq`, `keystone`, `glance`, `cinder`, `nova`, and `neutron` (figures 3.39– 3.45).



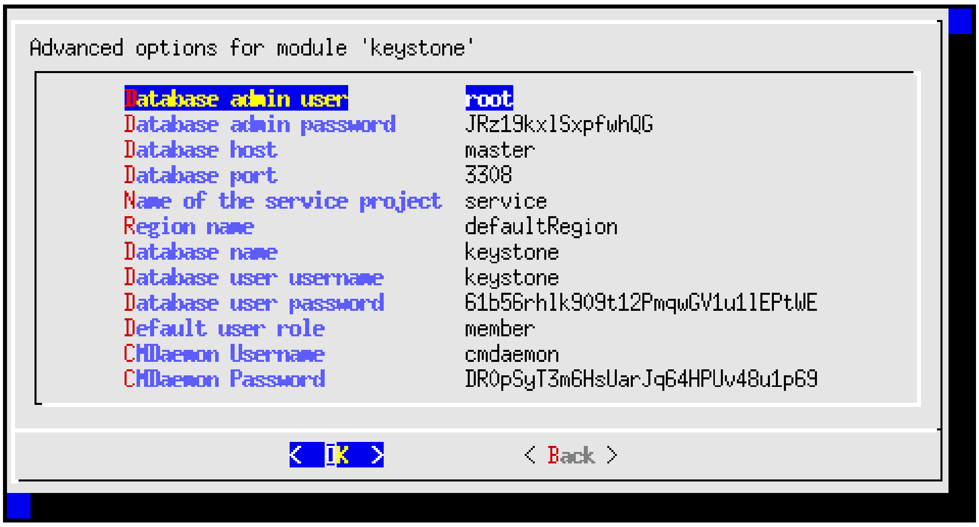
```
Advanced options for module 'core'

     Database admin user                    root
     Database admin password                None
     Database host                          master
     Database port                          3306
     Name of the service project            service
     Region name                            defaultRegion
     Reboot the nodes during deployment     True
     Default num of vnodes                  5
     Configure vnodes                       True
     vnodes category                        vnodes
```

Figure 3.39: Advanced Options: Core



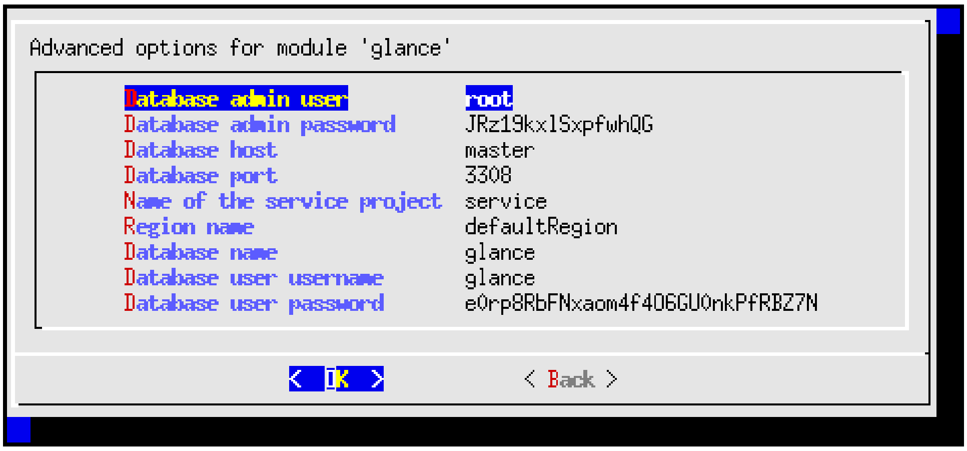
```
Advanced options for module 'rabbitmq'

        RabbitMQ Username      openstack
        RabbitMQ Password      RKyl0DlF0N0TZd2DA70oZVxe50p6Rm
        Shared OpenStack dir   /cm/shared/apps/openstack
```

Figure 3.40: Advanced Options: RabbitMQ



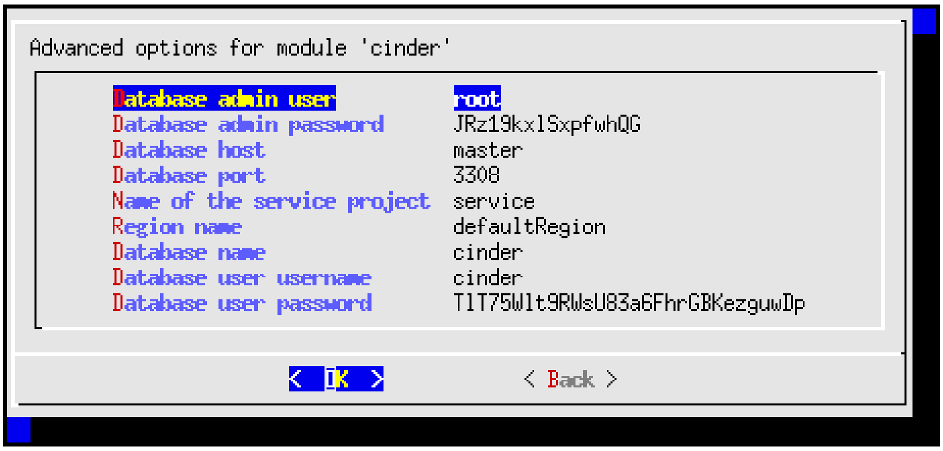
```
Advanced options for module 'keystone'

     Database admin user              root
     Database admin password          JRz19kxlSxpfwhQG
     Database host                    master
     Database port                    3308
     Name of the service project      service
     Region name                      defaultRegion
     Database name                    keystone
     Database user username           keystone
     Database user password           61b56rhlk909t12PmqwGV1u1lEPtWE
     Default user role                member
     CMDaemon Username                cmdaemon
     CMDaemon Password                DR0pSyT3m6HsUarJq64HPUv48u1p69
```

Figure 3.41: Advanced Options: Keystone



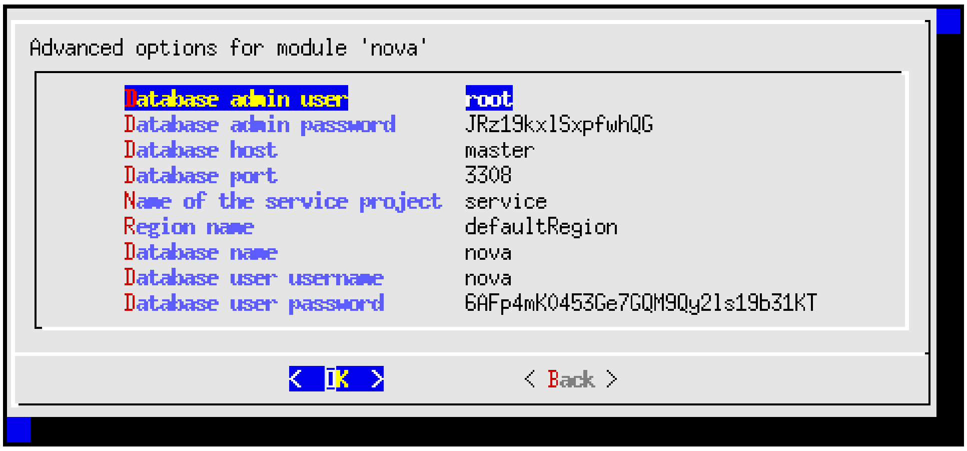
```
Advanced options for module 'glance'

     Database admin user              root
     Database admin password          JRz19kxlSxpfwhQG
     Database host                    master
     Database port                    3308
     Name of the service project      service
     Region name                      defaultRegion
     Database name                    glance
     Database user username           glance
     Database user password           e0rp8RbFNxaom4f406GU0nkPfRBZ7N
```

Figure 3.42: Advanced Options: Glance



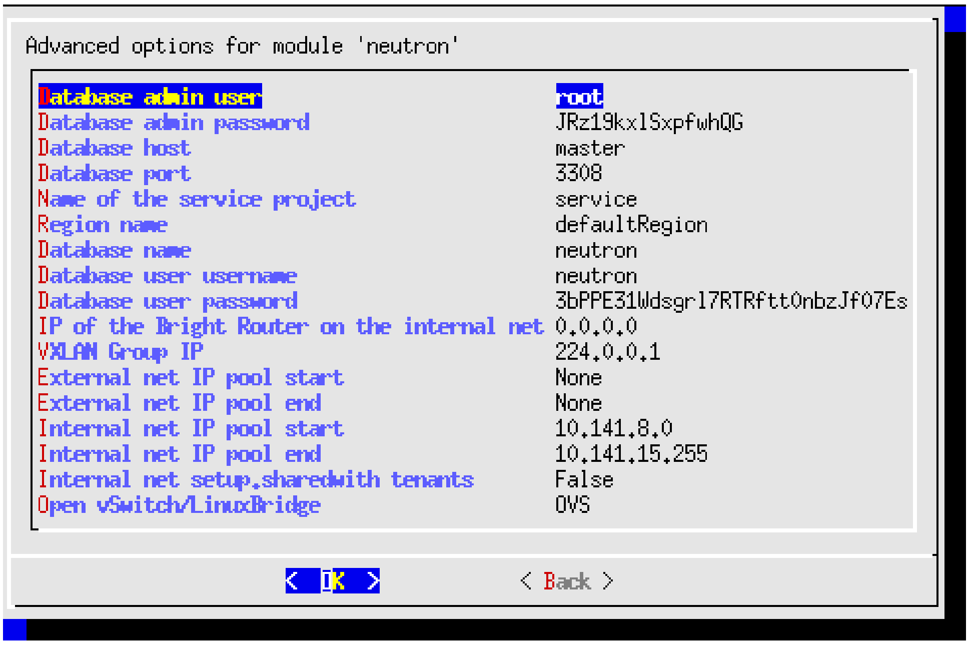
```
Advanced options for module 'cinder'

        Database admin user           root
        Database admin password       JRz19kxlSxpfwhQG
        Database host                 master
        Database port                 3308
        Name of the service project   service
        Region name                   defaultRegion
        Database name                 cinder
        Database user username        cinder
        Database user password        TlT75Wlt9RWsU83a6FhrGBKezguwDp

                  < OK >              < Back >
```

Figure 3.43: Advanced Options: Cinder



```
Advanced options for module 'nova'

        Database admin user           root
        Database admin password       JRz19kxlSxpfwhQG
        Database host                 master
        Database port                 3308
        Name of the service project   service
        Region name                   defaultRegion
        Database name                 nova
        Database user username        nova
        Database user password        6AFp4mK0453Ge7GQM9Qy2ls19b31KT

                  < OK >              < Back >
```

Figure 3.44: Advanced Options: Nova



```
Advanced options for module 'neutron'

Database admin user                          root
Database admin password                      JRz19kxlSxpfwhQG
Database host                                master
Database port                                3308
Name of the service project                  service
Region name                                  defaultRegion
Database name                                neutron
Database user username                       neutron
Database user password                       3bPPE31Wdsgrl7RTRfttOnbzJf07Es
IP of the Bright Router on the internal net  0.0.0.0
VXLAN Group IP                               224.0.0.1
External net IP pool start                   None
External net IP pool end                     None
Internal net IP pool start                   10.141.8.0
Internal net IP pool end                     10.141.15.255
Internal net setup.sharedwith tenants        False
Open vSwitch/LinuxBridge                     OVS
```

Figure 3.45: Advanced Options: Neutron

3.2.17 The Deployment Run—An Overview

The deployment displays a lengthy text run. An elided version follows:

```
Checking overlay names OpenStackHAProxies
Checking overlay names OpenStackControllers
Checking overlay names GaleraNodes
Checking overlay names OpenStackControllers
...
Executing 228 stages
################## Starting execution for 'bright73 OpenStack'
  - core
  - galera
  - rabbitmq
  - keystone
  - glance
  - cinder
  - nova
  - neutron
  - horizon
  - heat
  - radosgw
## Progress: 0
#### stage: core: Resolve Special Hostnames In Config
#### stage: core: Precheck System
Checking system configuration
## Progress: 1
#### stage: core: Precheck OpenStack
#### stage: core: Check Networking
## Progress: 2
#### stage: core: Precheck License
Your license allows for 70 OpenStack nodes.
#### stage: core: Precheck License Node Count
...
```

```
## Progress: 42
#### stage: AggregatedStages: Reboot Nodes
All affected nodes: ['node001', 'node002', 'node003', 'node004']
All nodes to be rebooted: node001, node002, node003, node004
Node has been rebooted node001
Node has been rebooted node002
Node has been rebooted node003
Node has been rebooted node004
Press ctrl+c to abort waiting and continue with deployment
Waiting for nodes to start reboot
Going to wait up to 1200 seconds for nodes to come back UP.
Waiting for 4 nodes to come back up
Waiting for 1 node to come back up
All 4 nodes came back up.
...
## Progress: 96
#### stage: core: Add Image
#### stage: core: Add Image
#### stage: core: Get Image UUID
## Progress: 97
#### stage: core: Configure Monitoring
Setting up monitoring for OpenStack
#### stage: core: Configure Sec Groups
## Progress: 98
#### stage: galera: Configure Monitoring
#### stage: keystone: Configure CMDaemon Post Deployment
## Progress: 99
#### stage: nova: Wait For Service To Be Operational
Waiting for nova
#### stage: nova: Patch Flavors
## Progress: 100
#### stage: horizon: Running: 'systemctl restart httpd'

Took:     26:38 min.
Progress: 100/100
################### Finished execution for 'bright73 OpenStack', status: completed
The following problems were encountered during execution:

Bright Cluster Manager OpenStack finished!

[root@bright73 ~]#
```

3.2.18 The State After Running `cm-openstack-setup`

At this point, the head node has OpenStack installed on it.

However, a regular node that has been configured with the OpenStack compute host role, ends up with OpenStack deployed on it only after the operating system running on the node is updated with the installed OpenStack software, and the newly-configured interfaces are set up according to the specified configuration.

For simplicity, it is best to simply reboot the regular nodes to update the interfaces and software on the regular nodes.

Trying to do it without a reboot by using `imageupdate` (section 5.6 of the *Administrator Manual*) is not recommended, because interfaces typically do change along with the updates, except for some specially configured cases. In the case of these special configurations, the setup wizard can be set to reboot only the controler node using the `cmgui` (figure 3.4) software image selection advanced view screen, but it should only be set to do it like that if the interfaces configuration has not changed.

The reboot action is therefore carried about by default, as shown in the preceding output, in the text that follows "Progress 42".

The administrator can further configure the cluster to suit requirements. Setting up a secondary node for high availability is discussed in section 3.3, while the rest of the manual describes other configurations.

3.3 Adding A Secondary Node To An Existing OpenStack Cluster For High Availability

On an existing OpenStack Bright cluster, the public endpoints point to the public IP address of the head node.

If a secondary head node is added to the cluster to provide high availability (Chapter 13 of the *Administrator Manual*), then some downtime is required. This is because after the secondary head node is synced from the primary and finalized, the public endpoints need to be changed to point to the shared public IP address, instead of the public IP address of the primary head node, and the OpenStack services then need to be restarted.

The endpoints can viewed and changed from `cmsh`, with a session similar to the following:

Example

```
[bright73->openstack[default]->endpoints]% list -f name:20,service:10,url:40,interface:10
name (key)           service    url                                      interface
-------------------- ---------- ---------------------------------------- ----------
volume:adminv1       cinder (a+ http://master:8776/v1/$(tenant_id)s      Admin
volume:internalv1    cinder (a+ http://master:8776/v1/$(tenant_id)s      Internal
volume:publicv1      cinder (a+ http://10.2.61.67:8776/v1/$(tenant_id)s  Public
volume:adminv2       cinderv2 + http://master:8776/v2/$(tenant_id)s      Admin
volume:internalv2    cinderv2 + http://master:8776/v2/$(tenant_id)s      Internal
volume:publicv2      cinderv2 + http://10.2.61.67:8776/v2/$(tenant_id)s  Public
glance:admin         glance (c+ http://master:9292                       Admin
glance:internal      glance (c+ http://master:9292                       Internal
glance:public        glance (c+ http://10.2.61.67:9292                   Public
heat:admin           heat (e19+ http://master:8004/v1/$(tenant_id)s      Admin
heat:internal        heat (e19+ http://master:8004/v1/$(tenant_id)s      Internal
heat:public          heat (e19+ http://10.2.61.67:8004/v1/$(tenant_id)s  Public
keystone:admin       keystone + http://master:35357/v3                   Admin
keystone:internal    keystone + http://master:5000/v3                    Internal
keystone:public      keystone + http://10.2.61.67:5000/v3                Public
networking:admin     neutron (+ http://master:9696/                      Admin
networking:internal  neutron (+ http://master:9696/                      Internal
networking:public    neutron (+ http://10.2.61.67:9696/                  Public
compute:admin        nova (632+ http://master:8774/v2/$(tenant_id)s      Admin
compute:internal     nova (632+ http://master:8774/v2/$(tenant_id)s      Internal
compute:public       nova (632+ http://10.2.61.67:8774/v2/$(tenant_id)s  Public
[bright73->openstack[default]->endpoints]% use volume:publicv1
[bright73...endpoints[volume:publicv1]]% set url "http://<shared external IP>:8776/v2/$(tenant_id)s"
[bright73->openstack[default]->endpoints*[volume:publicv1*]]% commit
```

In the preceding example, the `-f` option is used to reduce the list output format to something easier to look over. Also, when setting the `publicv1` URL, the italicized text *shared external IP* should be replaced with the actual value of the shared public external IP address.

4

Ceph Installation

4.1 Ceph Introduction

Ceph, at the time of writing, is the recommended storage software for OpenStack for serious use. The Ceph RADOS Gateway is a drop-in replacement for Swift, with a compatible API. Ceph is the recommended backend driver for Cinder, Glance and Nova.

The current chapter discusses

- The concepts and required hardware for Ceph (section 4.1)
- Ceph installation and management (section 4.2)
- RADOS GW installation and management (section 4.4)

4.1.1 Ceph Object And Block Storage

Ceph is a distributed storage software. It is based on an object store layer called RADOS (Reliable Autonomic Distributed Object Store), which consists of Ceph components called OSDs (Object Storage Devices) and MONs (Monitoring Servers). These components feature heavily in Ceph. OSDs deal with storing the objects to a device, while MONs deal with mapping the cluster. OSDs and MONs, together carry out object storage and block storage within the object store layer. The stack diagram of figure 4.1 illustrates these concepts.

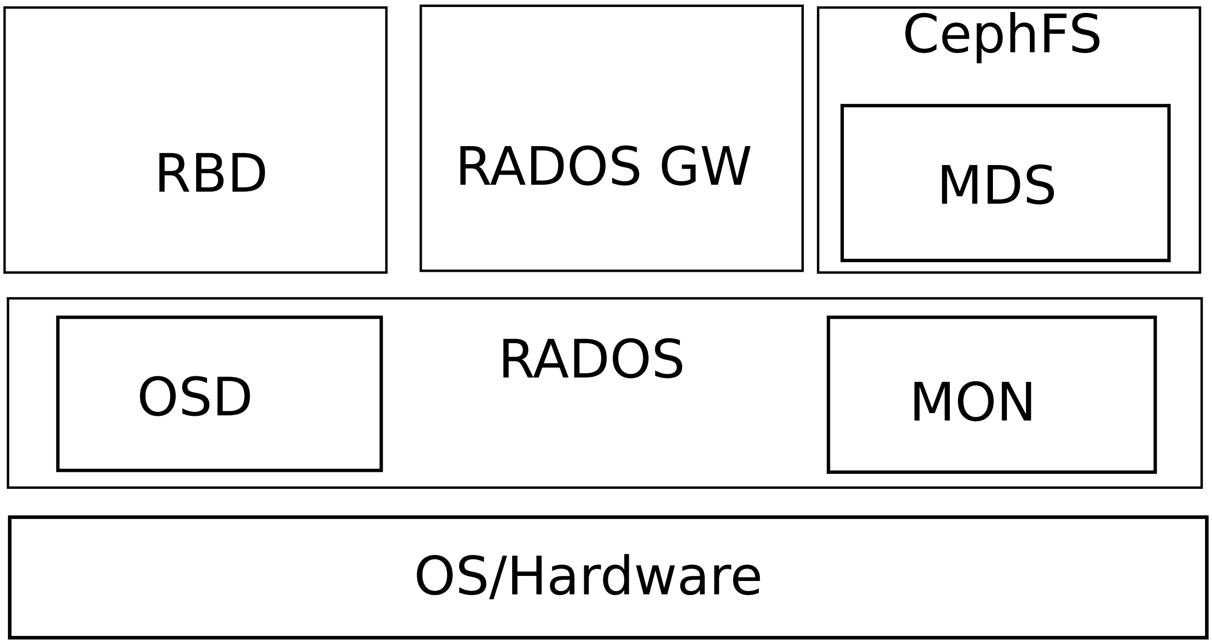


Figure 4.1: Ceph Concepts

On top of the object store layer are 3 kinds of access layers:

1. **Block device access:** RADOS Block Device (RBD) access can be carried out in two slightly different ways:

   (i) via a Linux kernel module based interface to RADOS. The module presents itself as a block device to the machine running that kernel. The machine can then use the RADOS storage, that is typically provided elsewhere.

   (ii) via the `librbd` library, used by virtual machines based on `qemu` or `KVM`. A block device that uses the library on the virtual machine then accesses the RADOS storage, which is typically located elsewhere.

2. **Gateway API access:** RADOS Gateway (RADOS GW) access provides an HTTP REST gateway to RADOS. Applications can talk to RADOS GW to access object storage in a high level manner, instead of talking to RADOS directly at a lower level. The RADOS GW API is compatible with the APIs of Swift and Amazon S3.

3. **Ceph Filesystem access:** CephFS provides a filesystem access layer. A component called MDS (Metadata Server) is used to manage the filesystem with RADOS. MDS is used in addition to the OSD and MON components used by the block and object storage forms when CephFS talks to RADOS. The Ceph filesystem was declared production-ready in Ceph Jewel, but at the time of writing (July 2016) this access layer is not yet supported by Bright Cluster Manager.

4.1.2 Ceph Software Considerations Before Use

Recommended Filesystem For Ceph Use

The storage forms of Ceph (object, block, or filesystem) can use a filesystem for storage. For production use of Ceph, XFS is currently the recommended filesystem option due to its stability, ability to handle extreme storage sizes, and its intrinsic ability to deal with the significant sizes of the extended attributes required by Ceph.

The nodes that run OSDs are typically regular nodes. Within the nodes, the storage devices used by the OSDs automatically have their filesystems configured to be of the XFS type during the installation of Ceph with Bright Cluster Manager.

Use Of datanode For Protection Of OSD Data

Typically, a filesystem used for an OSD is not on the same device as that of the regular node filesystem. Instead, typically, OSD storage consists of several devices that contain an XFS filesytem, with the devices attached to the node. These devices need protection from being wiped during the reprovisioning that takes place during a reboot of regular nodes .

The recommended way to protect storage devices from being wiped is to set the datanode property of their node to yes (page 183 of the *Administrator Manual*).

Use Of Slurm On OSD Nodes

Ceph can be quite demanding of the network and I/O. Running Slurm jobs on an OSD node is therefore not recommended. In addition, if Slurm roles are to be assigned to nodes that have OSD roles, then the default ports 6817 and 6818 used by Slurm can conflict with the default range 6800-7300 used by the Ceph OSD daemons. If there is a need to run Slurm on an OSD node then it is necessary to arrange it so that the ports used do not conflict with each other. During installation, a warning is given when this conflict is present.

4.1.3 Hardware For Ceph Use

An absolute minimum installation: can be carried out on two nodes, where:

- 1 node, the head node, runs one Ceph Monitor and the first OSD.
- 1 node, the regular node, runs the second OSD.

This is however not currently recommended, because the first OSD on the head node requires its own Ceph-compatible filesystem. If that filesystem is not provided, then Ceph on the cluster will run, but in a degraded state. Using such a system to try to get familiar with how Ceph behaves in a production environment with Bright Cluster Manager is unlikely to be worthwhile.

A more useful minimum: if there is a node to spare, installing Ceph over 3 nodes is suggested, where:

- 1 node, the head node, runs one Ceph Monitor.
- 1 node, the regular node, runs the first OSD.
- 1 more node, also a regular node, runs the second OSD.

In this case the OSD pool default size should be set to 2 in the Global OSD Settings (figure 4.8).

For production use: a redundant number of Ceph Monitor servers is recommended. Since the number of Ceph Monitoring servers must be odd, then at least 3 Ceph Monitor servers, with each on a separate node, are recommended for production purposes. The recommended minimum of nodes for production purposes is then 5:

- 2 regular nodes running OSDs.
- 2 regular nodes running Ceph Monitors.
- 1 head node running a Ceph Monitor.

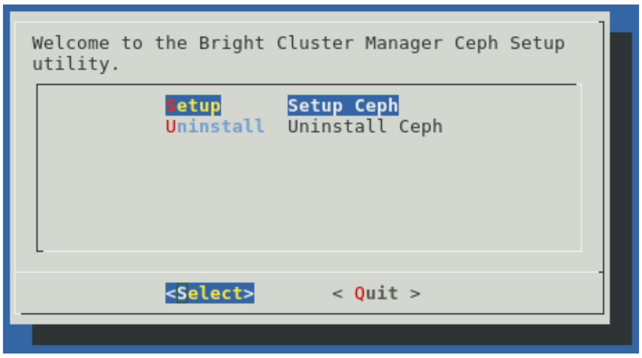
Drives usable by Ceph: Ceph OSDs can use any type of disk that presents itself as a block device in Linux. This means that a variety of drives can be used.

4.2 Ceph Installation With cm-ceph-setup

Ceph installation for Bright Cluster Manager can be carried out with the Ncurses-based cm-ceph-setup utility. It is part of the cluster-tools package that comes with Bright Cluster Manager. If the Ceph packages are not already installed, then the utility is able to install them for the head and regular nodes, assuming the repositories are accessible, and that the package manager priorities are at their defaults.

4.2.1 Ceph Installation: The Configuration Stage

The cm-ceph-setup utility can be run as root from the head node.

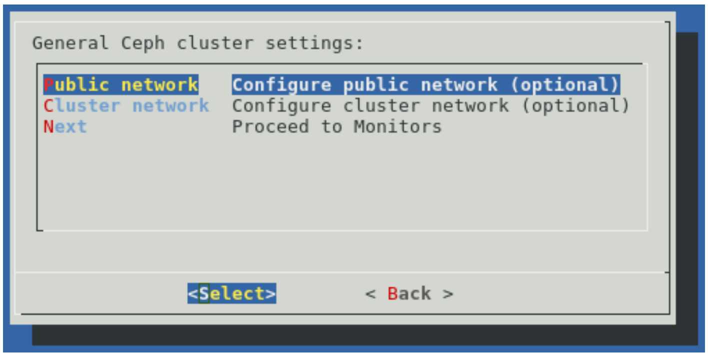


Welcome to the Bright Cluster Manager Ceph Setup utility.

Figure 4.2: Ceph Installation Welcome

At the welcome screen (figure 4.2), the administrator may choose to

- Set up Ceph
- Remove Ceph if it is already installed.



General Ceph cluster settings:

Public network Configure public network (optional)
Cluster network Configure cluster network (optional)
Next Proceed to Monitors
<Select>
< Back >

Figure 4.3: Ceph Installation General Cluster Settings

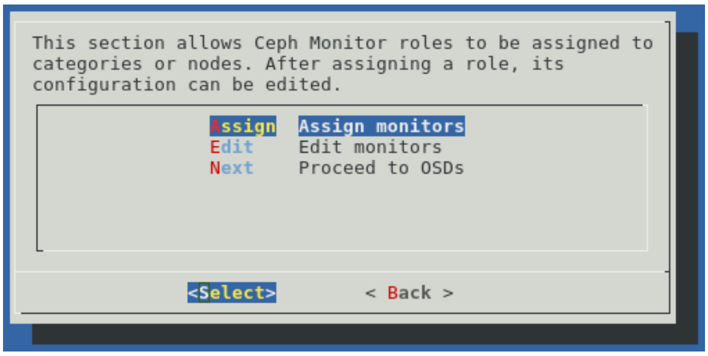
If the setup option is chosen, then a screen for the general Ceph cluster settings (figure 4.3) is displayed. The general settings can be adjusted via subscreens that open up when selected. The possible general settings are:

- Public network: This is the network used by Ceph Monitoring to communicate with OSDs. For a standard default Type 1 network this is internalnet.
- Cluster network: This is the network used by OSDs to communicate with each other. For a standard default Type 1 network this is internalnet.

Network Types are discussed in section 3.3.6 of the *Installation Manual*.

Selecting the Next option in figure 4.3 continues on with the next major screen of the setup procedure, and displays a screen for Ceph Monitors configuration (figure 4.4).

Ceph Monitors Configuration



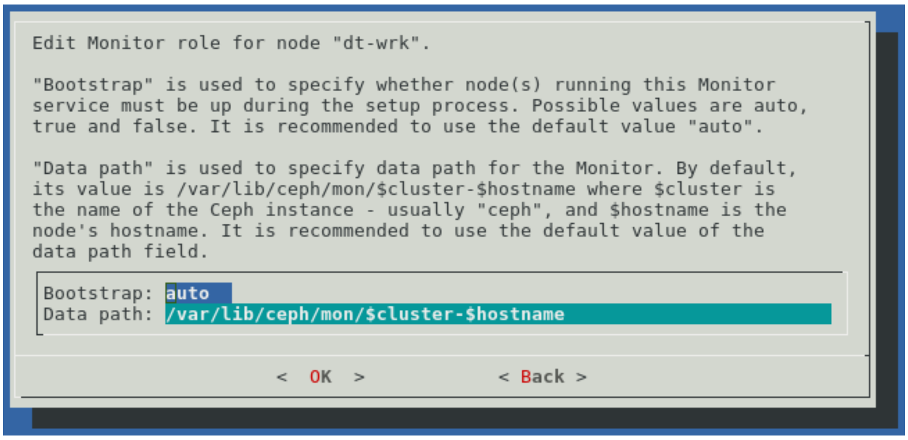
```
This section allows Ceph Monitor roles to be assigned to
categories or nodes. After assigning a role, its
configuration can be edited.

            Assign    Assign monitors
            Edit      Edit monitors
            Next      Proceed to OSDs

          <Select>          < Back >
```

Figure 4.4: Ceph Installation Monitors Configuration

In this screen:

- Ceph Monitor roles can be assigned to, or removed from, nodes or categories. On assigning a role, the nodes running the monitors can have their properties edited with the edit menu option.
- Existing Ceph Monitors can have their properties edited (figure 4.5), after selecting the nodes or categories.
- The OSD configuration screen can be reached after making changes, if any, to the Ceph Monitor configuration.

Typically in a first run, the head node has a Ceph Monitor added to it.



```
Edit Monitor role for node "dt-wrk".

"Bootstrap" is used to specify whether node(s) running this Monitor
service must be up during the setup process. Possible values are auto,
true and false. It is recommended to use the default value "auto".

"Data path" is used to specify data path for the Monitor. By default,
its value is /var/lib/ceph/mon/$cluster-$hostname where $cluster is
the name of the Ceph instance - usually "ceph", and $hostname is the
node's hostname. It is recommended to use the default value of the
data path field.

 Bootstrap: auto
 Data path: /var/lib/ceph/mon/$cluster-$hostname

                <  OK  >             < Back >
```

Figure 4.5: Ceph Installation Monitors Editing: Bootstrap And Data Path

Editing Ceph Monitors: The Edit option in figure 4.4 opens up a screen, figure 4.5, that allows the editing of existing or newly-added Ceph Monitors for a node or category:

- The bootstrap option can be set. The option configures initialization of the maps on the Ceph Monitors services, prior to the actual setup process. The bootstrap option can take the following values:
  - auto: This is the default and recommended option. If the majority of nodes are tagged with auto during the current configuration stage, and configured to run Ceph Monitors, then
    - If they are up according to Bright Cluster Manager at the time of deployment of the setup process, then the Monitor Map is initialized for those Ceph Monitors on those nodes.

    * If they are down at the time of deployment of the setup process, then the maps are not initialized.

  - `true`: If nodes are tagged `true` and configured to run Ceph Monitors, then they will be initialized at the time of deployment of the setup process, even if they are detected as being down during the current configuration stage.
  - `false`: If nodes are tagged `false` and configured to run Ceph Monitors, then they will not be initialized at the time of deployment of the setup process, even if they are detected as being up during the current configuration stage.

- The data path is set by default to:

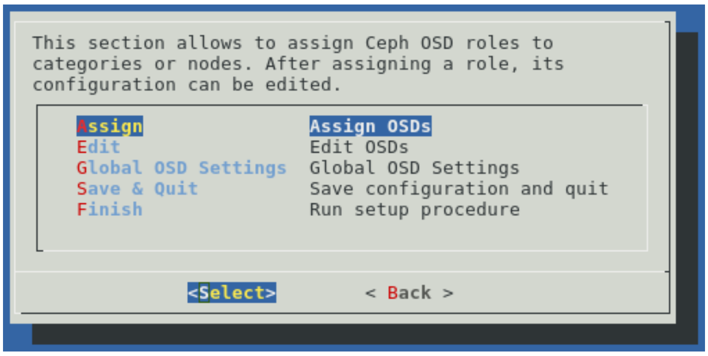
  ```
  /var/lib/ceph/mon/$cluster-$hostname
  ```

  where:

  - `$cluster` is the name of the Ceph instance. This is `ceph` by default.
  - `$hostname` is the name of the node being mapped.

- The `Back` option can be used after accessing the editing screen, to return to the Ceph Monitors configuration screen (figure 4.4).

Ceph OSDs Configuration

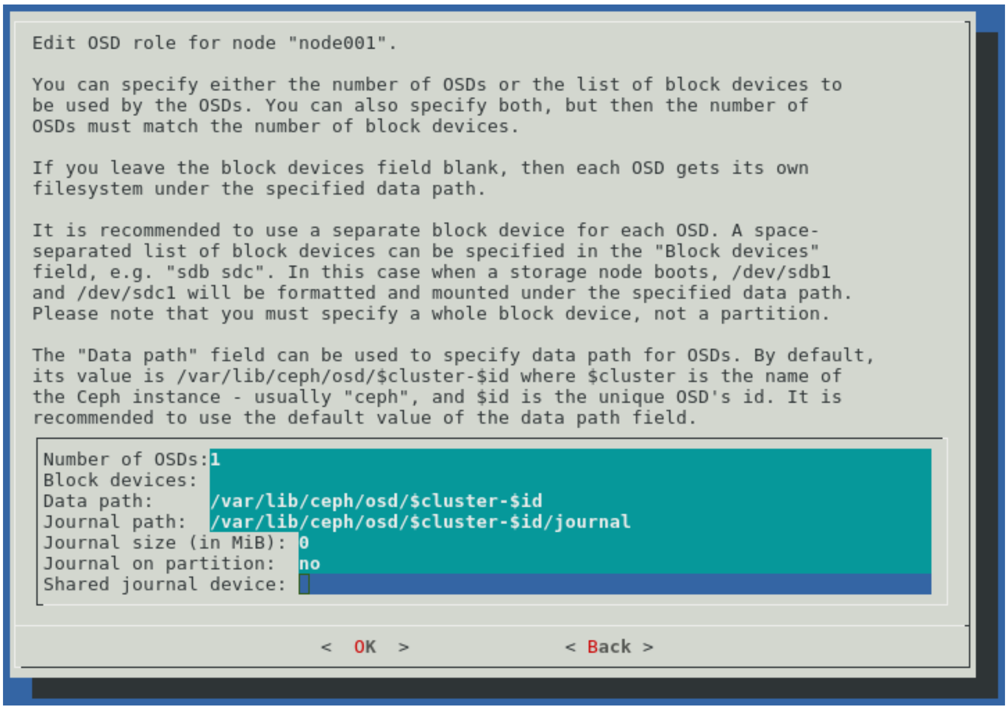


```
This section allows to assign Ceph OSD roles to
categories or nodes. After assigning a role, its
configuration can be edited.

    Assign                Assign OSDs
    Edit                  Edit OSDs
    Global OSD Settings   Global OSD Settings
    Save & Quit           Save configuration and quit
    Finish                Run setup procedure

            <Select>          < Back >
```

Figure 4.6: Ceph OSDs Configuration

If `Proceed to OSDs` is chosen from the Ceph Monitors configuration screen in figure 4.4, then a screen for Ceph OSDs configuration (figure 4.6) is displayed, where:

- OSDs roles can be assigned or removed from nodes or categories.
- Existing OSDs can be edited (figure 4.7) from nodes or categories.
- Global OSD settings can be edited (figure 4.8).
- The configuration can be saved for later with the `Save & Quit` option.
- To finish up on the installation, the `Finish` option runs the Ceph setup procedure itself.



```
Edit OSD role for node "node001".

You can specify either the number of OSDs or the list of block devices to
be used by the OSDs. You can also specify both, but then the number of
OSDs must match the number of block devices.

If you leave the block devices field blank, then each OSD gets its own
filesystem under the specified data path.

It is recommended to use a separate block device for each OSD. A space-
separated list of block devices can be specified in the "Block devices"
field, e.g. "sdb sdc". In this case when a storage node boots, /dev/sdb1
and /dev/sdc1 will be formatted and mounted under the specified data path.
Please note that you must specify a whole block device, not a partition.

The "Data path" field can be used to specify data path for OSDs. By default,
its value is /var/lib/ceph/osd/$cluster-$id where $cluster is the name of
the Ceph instance - usually "ceph", and $id is the unique OSD's id. It is
recommended to use the default value of the data path field.

 Number of OSDs:1
 Block devices:
 Data path:     /var/lib/ceph/osd/$cluster-$id
 Journal path:  /var/lib/ceph/osd/$cluster-$id/journal
 Journal size (in MiB): 0
 Journal on partition:  no
 Shared journal device:

                  <  OK  >                 < Back >
```

Figure 4.7: Ceph Installation OSDs Editing: Block Device Path, OSD Path, Journals For Categories Or Nodes

Editing Ceph OSDs: The Edit option in figure 4.6 opens up a screen, figure 4.7, that allows the editing of the properties of existing or newly-added Ceph OSDs for a node or category. In this screen:

- When considering the Number of OSDs and the Block devices, then it is best to set either

  - the Number of OSDs

  or

  - the Block devices

  Setting *both* the number of OSDs and block devices is also possible, but then the number of OSDs must match the number of block devices.

- If only a number of OSDs is set, and the block devices field is left blank, then each OSD is given its own filesystem under the data-path specified.

- Block devices can be set as a comma- or a space-separated list, with no difference in meaning.

  Example

  /dev/sda,/dev/sdb,/dev/sdc

  and

  /dev/sda /dev/sdb /dev/sdc

  are equivalent.

- For the OSD Data path, the recommended, and default value is:

  /var/lib/ceph/osd/$cluster-$id

  Here:

  - `$cluster` is the name of the head node of the cluster.
  - `$id` is a number starting from `0`.

- For the `Journal path`, the recommended, and default value is:

  ```
  /var/lib/ceph/osd/$cluster-$id/journal
  ```

- The `Journal size`, in MiB, can be set for the category or node. A value set here overrides the default global journal size setting (figure 4.3). This is just the usual convention where a node setting can override a category setting, and a node or category setting can both override a global setting.

  Also, just like in the case of the global journal size setting, a journal size for categories or nodes must always be greater than zero. Defining a value of 0 MiB means that the default that the Ceph software itself provides is set. At the time of writing (July 2016), Ceph software provides a default of 5GiB.

  The `Journal size` for a category or node is unset by default, which means that the value set for `Journal size` in this screen is determined by whatever the global journal size setting is, by default.

- Setting `Journal on partition` to `yes` means that the OSD uses a dedicated partition. In this case:

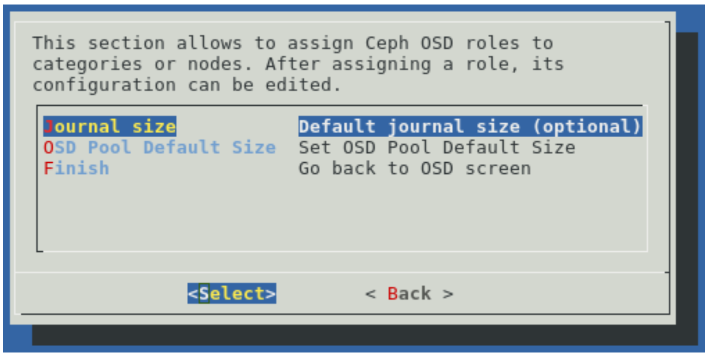
  - The disk setup used is modified so that the first partition, with a size of `Journal size` is used
  - A value of `0` for the `Journal size` is invalid, and does not cause a Ceph default size to be used.

  The default value of `Journal on partition` is `no`.

- The `Shared journal device` path must be set if a shared device is used for all the OSD journals in the category or node for which this screen applies. If it is used, then the device is partitioned automatically, and the available space is divided equally among the OSD journals.

  The path is unset by default, which means the device is not used by default.

The `Back` option can be used after accessing the editing screen, to return to the Ceph OSDs configuration screen (figure 4.6).



```
This section allows to assign Ceph OSD roles to
categories or nodes. After assigning a role, its
configuration can be edited.
```

Journal size   Default journal size (optional)
OSD Pool Default Size Set OSD Pool Default Size
Finish   Go back to OSD screen
<Select>
< Back >

Figure 4.8: Ceph Installation OSD Global Settings Editing: Default Journal Size, Default OSD Pool Size

Editing Ceph Global OSD Settings: The Global OSD Settings option can be selected from the Ceph OSDs main configuration screen (figure 4.6). The screen then displayed (figure 4.8) allows the following options to be modified:

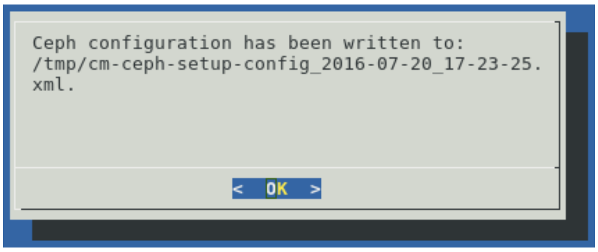
- Journal size: The default OSD journal size, in MiBs, used by an OSD. The actual size must be greater than zero. The value can be overridden by a category or node setting later on.

  Defining a value of 0 MiB here means that the default that the Ceph software itself provides is set. At the time of writing (July 2016), Ceph software provides a default of 5GiB.

- OSD Pool Default Size: The default OSD pool size. This sets the number of replicas for objects in the pool. It should be less than or equal to the number of OSD nodes. If unsure the administrator can just leave it at the default value.

The Back or Finish options can then be used to return to the Ceph OSDs configuration screen (figure 4.6).

Save Configuration And Quit



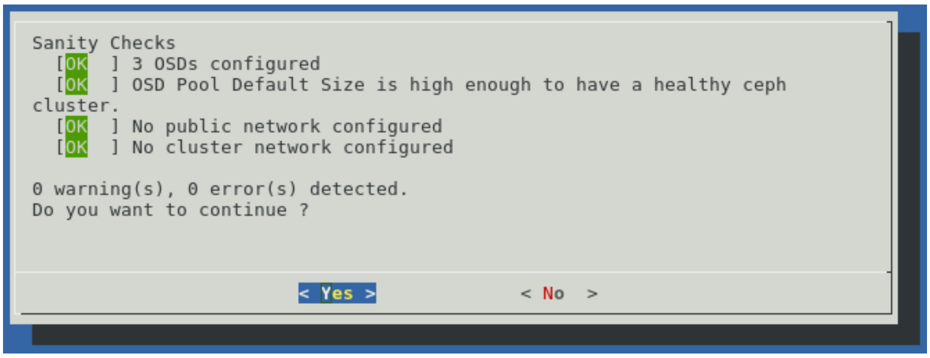
```
Ceph configuration has been written to:
/tmp/cm-ceph-setup-config_2016-07-20_17-23-25.
xml.
```

Figure 4.9: Ceph Installation Configuration Was Saved

After selecting the Save & Quit option of figure 4.6, the Ceph setup configuration file is saved, (figure 4.9), and the configuration part of the cm-ceph-setup script is completed. The deployment stage of the installation is next.

4.2.2 Ceph Installation: The Deployment Stage

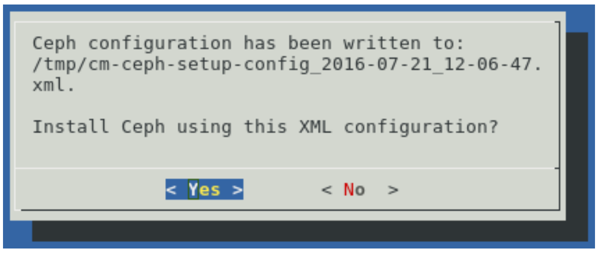
After selecting the Finish option of figure 4.6, the Ceph setup proceeds. First, sanity checks results are displayed:



```
Sanity Checks
  [OK   ] 3 OSDs configured
  [OK   ] OSD Pool Default Size is high enough to have a healthy ceph
cluster.
  [OK   ] No public network configured
  [OK   ] No cluster network configured

0 warning(s), 0 error(s) detected.
Do you want to continue ?
```

Figure 4.10: Ceph Installation Sanity Checks Result

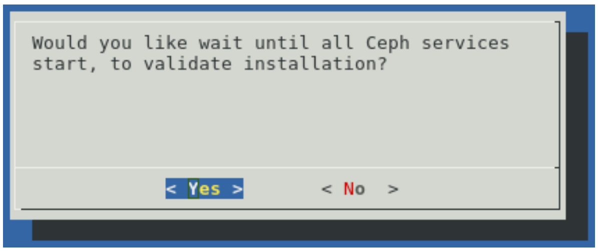
Then installation confirmation is displayed:



```
Ceph configuration has been written to:
/tmp/cm-ceph-setup-config_2016-07-21_12-06-47.
xml.

Install Ceph using this XML configuration?
```

Figure 4.11: Ceph Installation Confirmation

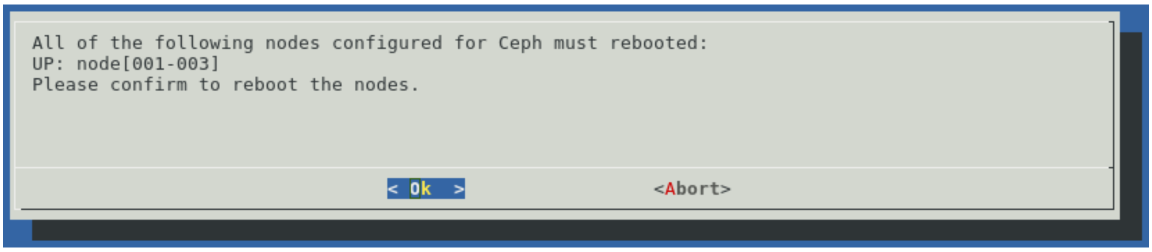
In the next dialog validation can be selected. In this case the installer waits until all Ceph services come up:



```
Would you like wait until all Ceph services
start, to validate installation?
```

Figure 4.12: Ceph Installation Validation Dialog

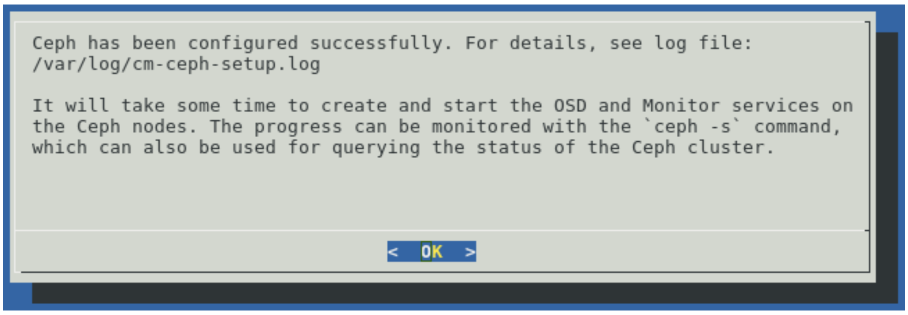
After that, the installation process starts. It eventually asks to confirm that the OSD nodes may be rebooted:



```
All of the following nodes configured for Ceph must rebooted:
UP: node[001-003]
Please confirm to reboot the nodes.
```

Figure 4.13: Nodes Reboot Confirmation

A successful run displays a screen as in figure 4.14:



```
Ceph has been configured successfully. For details, see log file:
/var/log/cm-ceph-setup.log

It will take some time to create and start the OSD and Monitor services on
the Ceph nodes. The progress can be monitored with the `ceph -s` command,
which can also be used for querying the status of the Ceph cluster.
```

Figure 4.14: Ceph Installation Completion

4.3 Checking And Getting Familiar With Ceph Items After `cm-ceph-setup`

4.3.1 Checking On Ceph And Ceph-related Files From The Shell

The status of Ceph can be seen from the command line by running:

Example

```
[root@bright73 ~]# ceph -s
   cluster d9422c23-321e-4fa0-b510-ca8e09a0a1fc
     health HEALTH_OK
     monmap e1: 1 mons at bright73=10.141.255.254:6789/0, election ep\
och 2, quorum 0 bright73
     osdmap e6: 2 osds: 2 up, 2 in
      pgmap v9: 192 pgs, 3 pools, 0 bytes data, 0 objects
            2115 MB used, 18340 MB / 20456 MB avail
                 192 active+clean
```

The -h option to ceph lists many options. Users of Bright Cluster Manager should usually not need to use these, and should find it more convenient to use the cmgui or cmsh front ends instead.

Generated XML Configuration File

By default, an XML configuration file is generated by the cm-ceph-setup utility, and stored after a run in the current directory as:

```
./cm-ceph-setup-config.xml
```

The name of the Ceph instance is by default ceph. If a new instance is to be configured with the cm-ceph-setup utility, then a new name must be set in the configuration file, and the new configuration file must be used.

Using An XML Configuration File

The -c option to cm-ceph-setup allows an existing XML configuration file to be used.

Example

```
[root@bright73 ~]# cm-ceph-setup -c /root/myconfig.xml
```

A Sample XML Configuration File

A Ceph XML configuration schema, with MONs and OSDs running on different hosts, could be as follows:

Example

```
<cephConfig>
  <networks>
    <public>internalnet</public>
    <cluster>internalnet</cluster>
  </networks>
  <journalsize>0</journalsize>
  <monitor>
    <hostname>raid-test</hostname>
    <monitordata>/var/lib/ceph/mon/$cluster-$hostname</monitordata>
  </monitor>
  <osd>
    <hostname>node001</hostname>
    <osdassociation>
      <name>osd0</name>
      <blockdev>/dev/sdd</blockdev>
      <osddata>/var/lib/ceph/osd/$cluster-$id</osddata>
      <journaldata>/var/lib/ceph/osd/$cluster-$id/journal</journaldata>
      <journalsize>0</journalsize>
    </osdassociation>
    <osdassociation>
      <name>osd1</name>
      <blockdev>/dev/sde</blockdev>
      <osddata>/var/lib/ceph/osd/$cluster-$id</osddata>
      <journaldata>/var/lib/ceph/osd/$cluster-$id/journal</journaldata>
      <journalsize>0</journalsize>
    </osdassociation>
    <osdassociation>
      <name>osd2</name>
      <blockdev>/dev/sdf</blockdev>
      <osddata>/var/lib/ceph/osd/$cluster-$id</osddata>
      <journaldata>/var/lib/ceph/osd/$cluster-$id/journal</journaldata>
      <journalsize>0</journalsize>
    </osdassociation>
  </osd>
</cephConfig>
```

A disk setup (section 3.9.3 of the *Administrator Manual*) can be specified to place the OSDs on an XFS device, on partition `a2` as follows:

Example

```
<diskSetup>
  <device>
    <blockdev>/dev/sda</blockdev>
    <partition id="a1">
      <size>10G</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
    <partition id="a2">
      <size>10G</size>
      <type>linux</type>
      <filesystem>xfs</filesystem>
      <mountPoint>/var</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
    <partition id="a3">
      <size>2G</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/tmp</mountPoint>
      <mountOptions>defaults,noatime,nodiratime,nosuid,nodev</mountOptions>
    </partition>
    <partition id="a4">
      <size>1G</size>
      <type>linux swap</type>
    </partition>
    <partition id="a5">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/local</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
  </device>
</diskSetup>
```

Installation Logs

Installation logs to Ceph are kept at:

```
/var/log/cm-ceph-setup.log
```

4.3.2 Ceph Management With cmgui And cmsh

Only one instance of Ceph is supported at a time. Its name is ceph.

Ceph Overview And General Properties

From within cmsh, ceph mode can be accessed:

Example

```
[root@bright73 ~]# cmsh
[bright73]% ceph
[bright73->ceph]%
```

From within ceph mode, the overview command lists an overview of Ceph OSDs, MONs, and placement groups for the ceph instance. Parts of the displayed output are elided in the example that follows for viewing convenience:

Example

```
[bright73->ceph]% overview ceph
Parameter                         Value
--------------------------------- ----------------------------
Status                            HEALTH_OK
Number of OSDs                    2
Number of OSDs up                 2
Number of OSDs in                 2
Number of mons                    1
Number of placements groups       192
Placement groups data size        0B
Placement groups used size        10.07GB
Placement groups available size   9.91GB
Placement groups total size       19.98GB

Name                              Used       Objects     ...
--------------------------------- ---------- ----------  ...
bright73:.rgw                     1B         0           ...
bright73:data                     0B         0           ...
bright73:metadata                 0B         0           ...
bright73:rbd                      0B         0           ...
...
```

The cmgui equivalent of the overview command is the Overview tab, accessed from within the Ceph resource.

Some of the major Ceph configuration parameters can be viewed and their values managed by CMDaemon from ceph mode. The show command shows parameters and their values for the ceph instance:

Example

```
[bright73->ceph]% show ceph
Parameter                      Value
------------------------------ ----------------------------------------
Admin keyring path             /etc/ceph/ceph.client.admin.keyring
Bootstrapped                   yes
Client admin key               AQDkUM5T4LhZFxAA/JQHvzvbyb9txH0bwvxUSQ==
Cluster networks
Config file path               /etc/ceph/ceph.conf
Creation time                  Thu, 25 Sep 2014 13:54:11 CEST
Extra config parameters
Monitor daemon port            6789
Monitor key                    AQDkUM5TwM2lEhAA0CcdH/UFhGJ902n3y/Avng==
Monitor keyring path           /etc/ceph/ceph.mon.keyring
Public networks
Revision
auth client required cephx     yes
auth cluster required cephx    yes
auth service required cephx    yes
filestore xattr use omap       no
fsid                           abf8e6af-71c0-4d75-badc-3b81bc2b74d8
mon max osd                    10000
mon osd full ratio             0.95
mon osd nearfull ratio         0.85
name                           ceph
osd pool default min size      0
osd pool default pg num        8
osd pool default pgp num       8
osd pool default size          2
version                        0.80.5
[bright73->ceph]%
```

The cmgui equivalent of these settings is the Settings tab, accessed from within the Ceph resource.

Ceph extraconfigparameters setting: The Extra config parameters property of a ceph mode object can be used to customize the Ceph configuration file. The Ceph configuration file is typically in /etc/ceph.conf, and using extraconfiparameters settings, Ceph can be configured with changes that CMDaemon would otherwise not manage. After the changes have been set, CMDaemon manages them further.

Thus, the following configuration section in the Ceph configuration file:

```
[mds.2]
host=rabbit
```

could be placed in the file via cmsh with:

Example

```
[root@bright73 ~]# cmsh
[bright73]% ceph
[bright73->ceph[ceph]]% append extraconfigparameters "[mds.2] host=rabbit"
[bright73->ceph*[ceph*]]% commit
```

If a section name, enclosed in square brackets, [], is used, then the section is recognized at the start of an appended line by CMDaemon.

If a section that is specified in the square brackets does not already exist in /etc/ceph.conf, then it will be created. The \n is interpreted as a new line at its position. After the commit, the extra configuration parameter setting is maintained by the cluster manager.

If the section already exists in /etc/ceph.conf, then the associated key=value pair is appended. For example, the following appends host2=bunny to an existing mds.2 section:

```
[bright73->ceph[ceph]]% append extraconfigparameters "[mds.2] host2=bunny"
[bright73->ceph*[ceph*]]% commit
```

If no section name is used, then the key=value entry is appended to the [global] section.

```
[bright73->ceph[ceph]]% append extraconfigparameters "osd journal size = 128"
[bright73->ceph*[ceph*]]% commit
```

The /etc/ceph.conf file has the changes written into it about a minute after the commit, and may then look like (some lines removed for clarity):

```
[global]
auth client required = cephx
osd journal size=128

[mds.2]
host=rabbit
host2=bunny
```

As usual in cmsh operations (section 2.5.3 of the *Administrator Manual*):

- The set command clears extraconfigparameters before setting its value
- The removefrom command operates as the opposite of the append command, by removing key=value pairs from the specified section.

There are similar extraconfigparameters for Ceph OSD filesystem associations (page 64) and for Ceph monitoring (page 65).

Ceph OSD Properties

From within ceph mode, the osdinfo command for the Ceph instance displays the nodes that are providing OSDs along with their OSD IDs:

Example

```
[bright73->ceph]% osdinfo ceph
OSD id       Node                     OSD name
------------ ----------------------- ------------
0            node001                  osd0
1            node002                  osd0
```

Within a device or category mode, the roles submode allows parameters of an assigned cephosd role to be configured and managed.

Example

```
[bright73->category[default]->roles]% show cephosd
Parameter                     Value
----------------------------- ------------------------------
Name                          cephosd
OSD associations              <1 in submode>
Provisioning associations     <0 internally used>
Revision
Type                          CephOSDRole
```

Within the cephosd role the templates for OSD filesystem associations, osdassociations, can be set or modified:

Example

```
[bright73->category[default]->roles]% use cephosd
[bright73...[default]->roles[cephosd]]% osdassociations
[bright73...osd]->osdassociations]% list -f name:10,osddata:30
name (key) osddata
---------- ------------------------------
osd0       /var/lib/ceph/osd/$cluster-$id
[bright73...osd->osdassociations]% list -f journaldata:38,journalsize:11
name (key) journaldata                            journalsize
---------- -------------------------------------- -----------
osd0       /var/lib/ceph/osd/$cluster-$id/journal 0
```

The -f option is used here with the list command merely in order to format the output so that it stays within the margins of this manual.

The cmgui equivalent of the preceding cmsh settings is accessed from within a particular Nodes or Categories item in the resource tree, then accessing the Ceph tab, and then choosing the OSD checkbox. The Advanced button allows cephosd role parameters to be set for the node or category.

OSD filesystem association extraconfigparameters **setting:** Extra configuration parameters can be set for an OSD filesystem association such as ods0 by setting values for its extraconfigparameters option. This is similar to how it can be done for Ceph general configuration (page 62):

Example

```
[bright73...osd]->osdassociations]% use osd0
[bright73...osdassociations[ods0]]% show
Parameter                        Value
-------------------------------- --------------------------------------
...
Automatically adjust weight      off
Extra config parameters
...
[bright73...osdassociations[osd0]]% set extraconfigparameters "a=b"
...
```

Ceph Monitoring Properties

Similarly to Ceph OSD properties, the parameters of the cephmonitor role can be configured and managed from within the node or category that runs Ceph monitoring.

Example

```
[bright73]% device use bright73
[bright73->device[bright73]]% roles ; use cephmonitor
[ceph->device[bright73]->roles[cephmonitor]]% show
Parameter                        Value
-------------------------------- ------------------------------------
...
Extra config parameters
Monitor data                     /var/lib/ceph/mon/$cluster-$hostname
Name                             cephmonitor
Provisioning associations        <0 internally used>
Revision
Type                             CephMonitorRole
```

Ceph monitoring `extraconfigparameters` setting: Ceph monitoring can also have extra configurations set via the `extraconfigparameters` option, in a similar way to how it is done for Ceph general configuration (page 62).

Monitors are similarly accessible from within `cmgui` for nodes and categories, with an `Advanced` button in their `Ceph` tab allowing the parameters for the Monitor checkbox to be set.

Ceph `bootstrap`

For completeness, the `bootstrap` command within `ceph` mode can be used by the administrator to initialize Ceph Monitors on specified nodes if they are not already initialized. Administrators are however not expected to use it, because they are expected to use the `cm-ceph-setup` installer utility when installing Ceph in the first place. The installer utility carries out the bootstrap initialization as part of its tasks. The `bootstrap` command is therefore only intended for use in the unusual case where the administrator would like to set up Ceph storage without using the `cm-ceph-setup` utility.

4.4 RADOS GW Installation, Initialization, And Properties

4.4.1 RADOS GW Installation And Initialization

If Ceph has been installed during `cm-ceph-setup`, then RADOS is installed and initialized so that it provides a REST API, called the RADOS GW service.

4.4.2 Setting RADOS GW Properties

RADOS GW Properties In `cmsh`

RADOS GW properties can be managed in `cmsh` by selecting the device, then assigning the `radosgateway` role to the device. The properties of the role can then be seen and altered:

```
[bright73]% device use node004
[bright73->device[node004]]% roles
[bright73->device[node004]->roles]% assign radosgateway; commit
[bright73->device[node004]->roles[radosgateway]]% show
Parameter                         Value
--------------------------------- --------------------------------
Name                              radosgateway
Provisioning associations         <0 internally used>
Revision
Type                              RadosGatewayRole
Server Port                       7480
Enable Keystone Authentication    yes
Keystone Accepted Roles
Keystone Revocation Interval      600
Keystone Tokens Cache Size        500
NSS DB Path                       /var/lib/ceph/nss
```

4.4.3 Turning Keystone Authentication On And Off For RADOS GW

Keystone authentication can be disabled or enabled using `cmsh` to set the `enablekeystoneauthentication` property. The property can be set from within the `radosgateway` role of the node running the RADOS GW service.

For example, setting `enablekeystoneauthentication` to `no` on a RADOS GW node, and committing it makes RADOS GW services unavailable to OpenStack instances.

Example

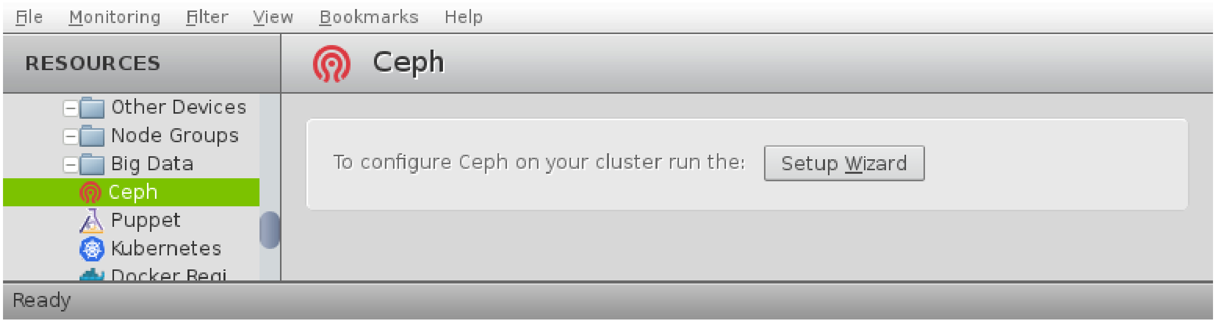
```
[bright73->device[node004]->roles[radosgateway]]% set enablekeystoneauthentication no
[bright73->device*[node004*]->roles*[radosgateway*]]% commit
```

4.5 Installation Of Ceph From cmgui

Ceph can be installed from Bright Cluster Manager in the following two ways:

- Using the text-based `cm-ceph-setup` utility (section 4.2). The utility is a part of the standard cluster-tools package.
- Using the GUI-based Ceph `Setup Wizard` button from within `cmgui` (this section). This is the recommended installation method.

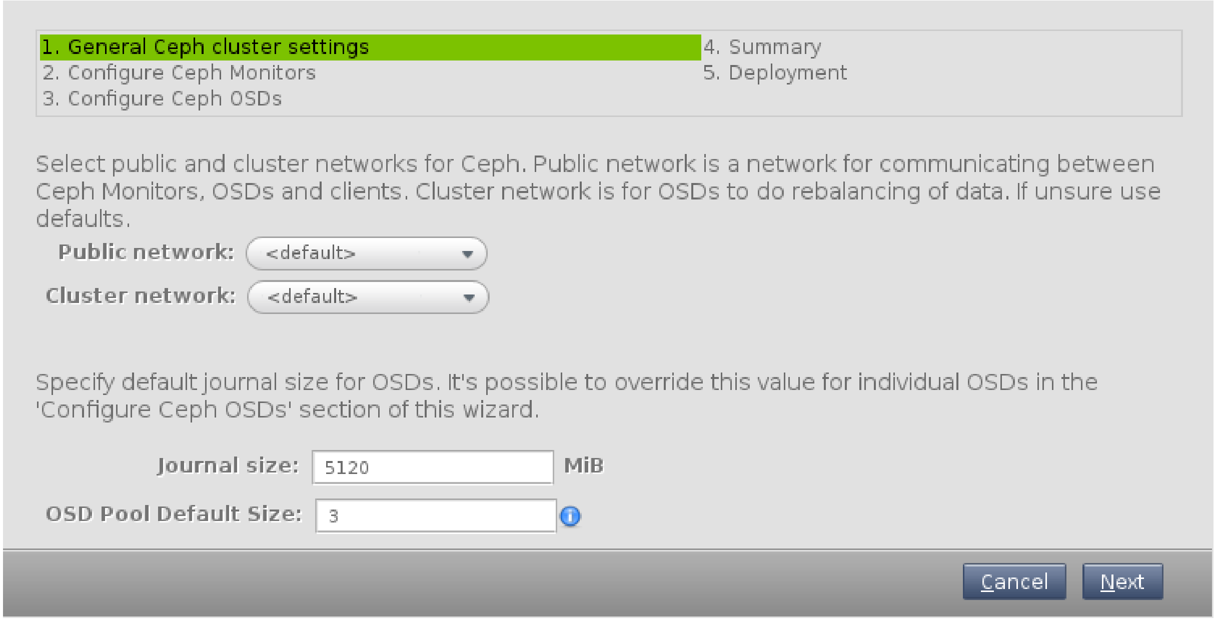
If Ceph is not yet installed on the cluster, then it can be installed with `cmgui` by clicking on the `Ceph` resource folder. This brings up the `cmgui` setup wizard Ceph Installation start screen (figure 4.15).



To configure Ceph on your cluster run the:

Figure 4.15: Ceph Wizard Installation: Start Screen

Clicking on the `Setup Wizard` button launches the run through the setup wizard. The first page of the wizard is the Ceph Installation General cluster settings screen (figure 4.16).



Select public and cluster networks for Ceph. Public network is a network for communicating between Ceph Monitors, OSDs and clients. Cluster network is for OSDs to do rebalancing of data. If unsure use defaults.

Public network: <default>

Cluster network: <default>

Specify default journal size for OSDs. It's possible to override this value for individual OSDs in the 'Configure Ceph OSDs' section of this wizard.

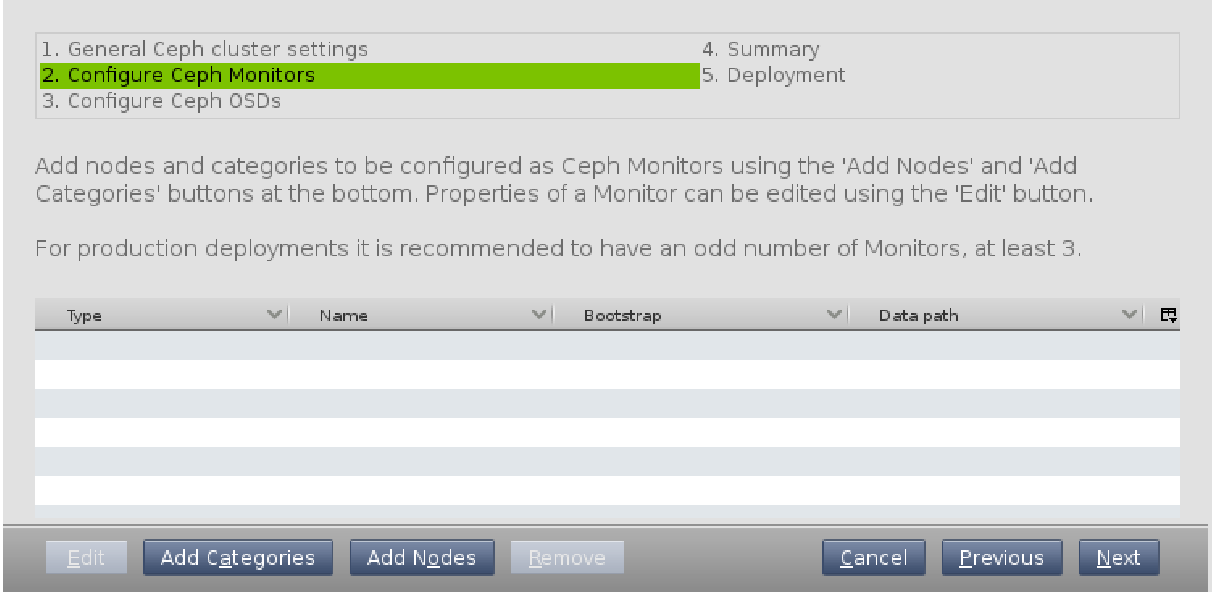
Journal size: 5120 MiB

OSD Pool Default Size: 3

Figure 4.16: Ceph Wizard Installation: General Cluster Settings

The GUI screen of figure 4.16 is a combination of the Ncurses Ceph Installation General Cluster Settings screen figure 4.3 (page 66), together with the Ncurses OSD journal settings of figure 4.8, (page 56). The settings of the `cmgui` screen are explained in the texts in the section for figures 4.3 and 4.8.

The next screen is the Ceph Monitors settings screen (figure 4.17). The Ceph Monitors settings screen allows items to be selected for use as Ceph Monitors. The items to be selected can be categories (using the `Add Categories` button) or nodes (using the `Add Nodes` button):



1. General Ceph cluster settings
2. Configure Ceph Monitors
3. Configure Ceph OSDs
4. Summary
5. Deployment

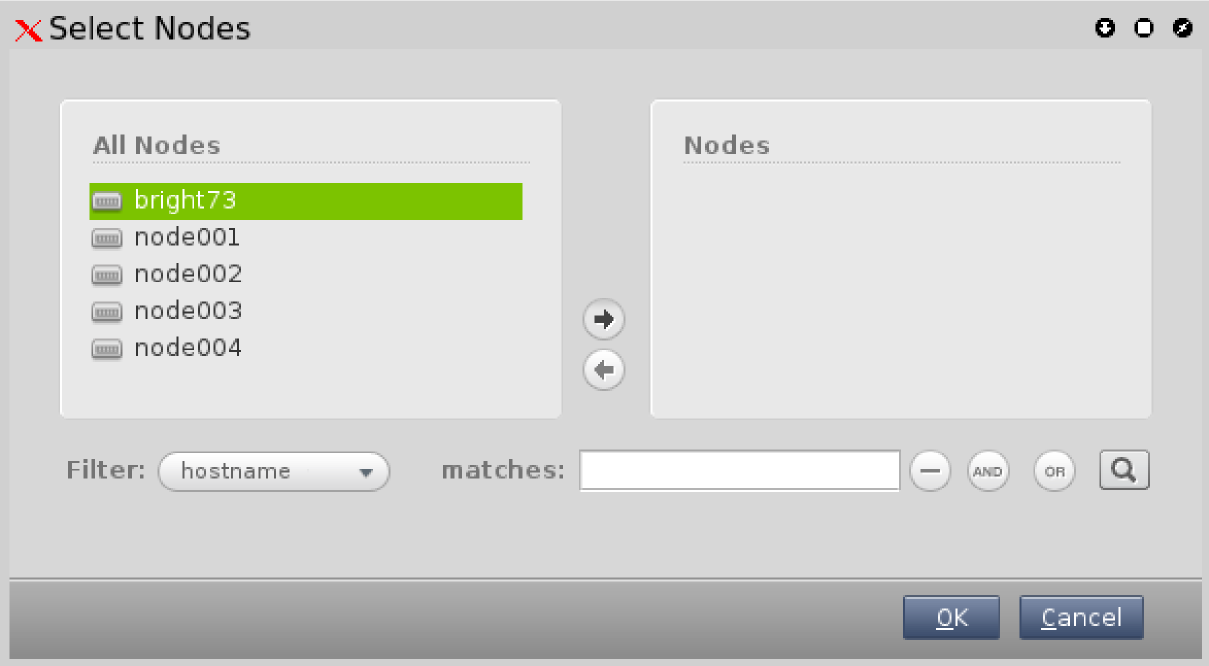
Add nodes and categories to be configured as Ceph Monitors using the 'Add Nodes' and 'Add Categories' buttons at the bottom. Properties of a Monitor can be edited using the 'Edit' button.

For production deployments it is recommended to have an odd number of Monitors, at least 3.

Type	Name	Bootstrap	Data path

Figure 4.17: Ceph Wizard Installation: Ceph Monitors Settings

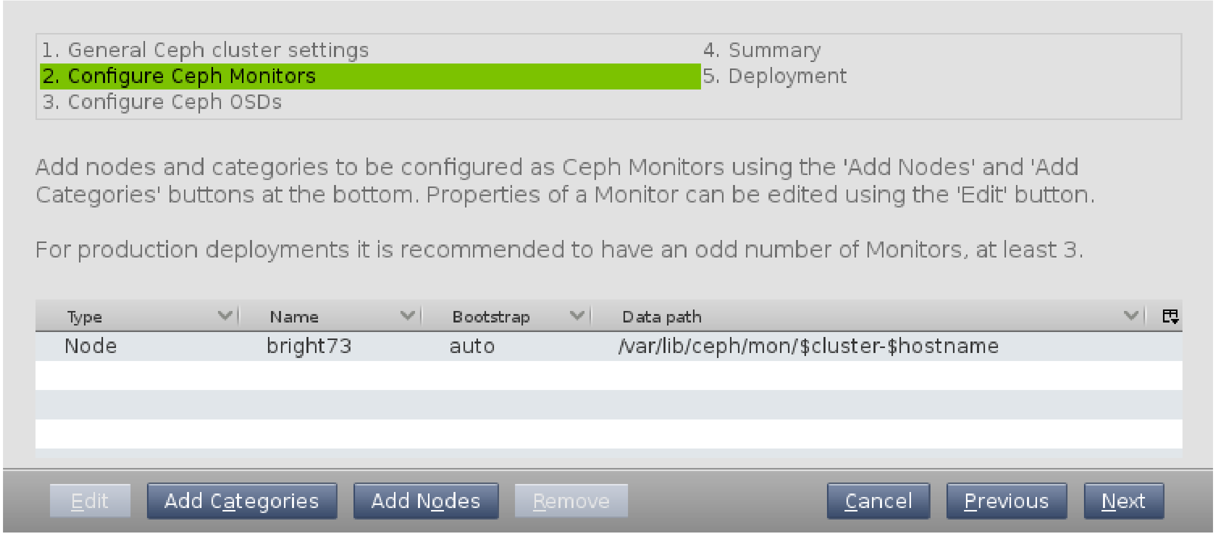
For example, the Add Nodes button opens up a Select Nodes dialog screen (figure 4.18):



Filter: hostname matches:

Figure 4.18: Ceph Wizard Installation: Select Nodes For Monitors Dialog

After selecting the item or items, and then clicking on the OK button, the Ceph Monitors settings screen is displayed again. This time the Ceph Monitors settings screen shows the selected item or items (figure 4.19):



1. General Ceph cluster settings
2. Configure Ceph Monitors
3. Configure Ceph OSDs
4. Summary
5. Deployment

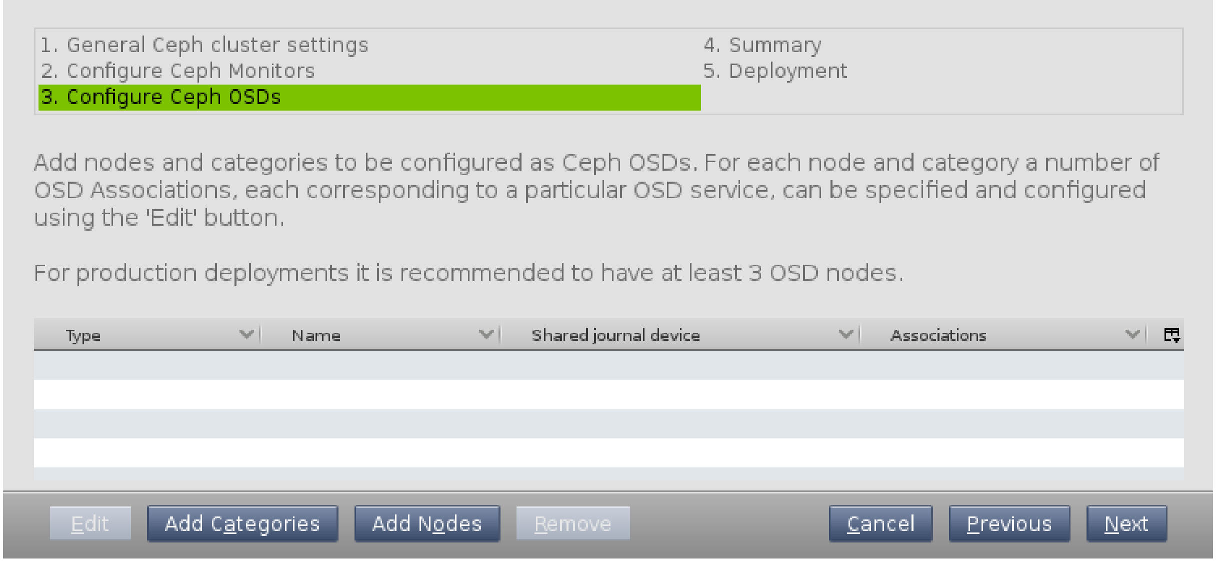
Add nodes and categories to be configured as Ceph Monitors using the 'Add Nodes' and 'Add Categories' buttons at the bottom. Properties of a Monitor can be edited using the 'Edit' button.

For production deployments it is recommended to have an odd number of Monitors, at least 3.

Type	Name	Bootstrap	Data path
Node	bright73	auto	/var/lib/ceph/mon/$cluster-$hostname

Figure 4.19: Ceph Wizard Installation: Ceph Monitors Selection

The next screen is the Ceph OSDs settings screen (figure 4.20). The Ceph OSDs settings screen allows items to be selected for use as Ceph OSDs. The items to be selected can be categories (using the Add Categories button) or nodes (using the Add Nodes button):



1. General Ceph cluster settings
2. Configure Ceph Monitors
3. Configure Ceph OSDs
4. Summary
5. Deployment

Add nodes and categories to be configured as Ceph OSDs. For each node and category a number of OSD Associations, each corresponding to a particular OSD service, can be specified and configured using the 'Edit' button.

For production deployments it is recommended to have at least 3 OSD nodes.

Type	Name	Shared journal device	Associations

Figure 4.20: Ceph Wizard Installation: Ceph OSDs Settings

For example, the Add Nodes button opens up a Select Nodes dialog screen (figure 4.21):

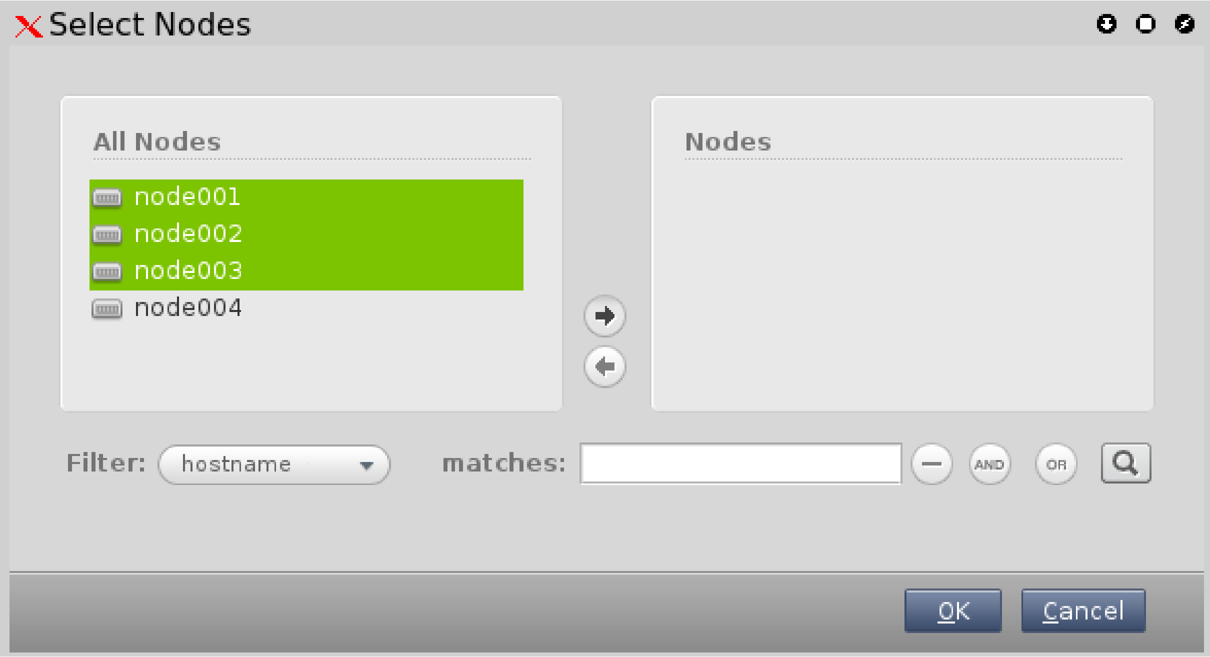
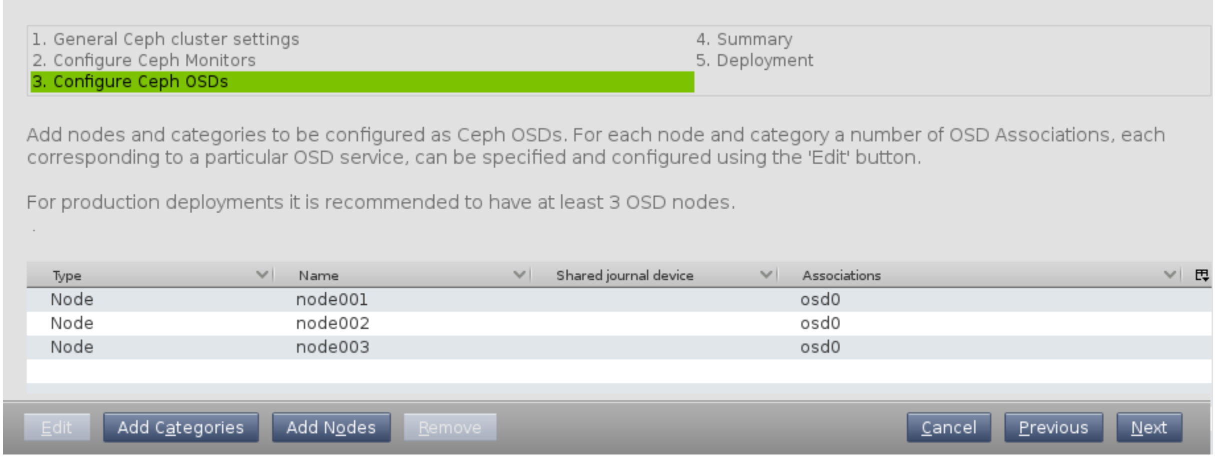


Figure 4.21: Ceph Wizard Installation: Select Nodes For OSDs Dialog

The dialogs shown in figure 4.18 for Ceph Monitors, and in figure 4.21 for Ceph OSDs, are both dialogs that look and work in the same way. Care should be taken not to confuse them with each other, because Monitors and OSDs have different purposes.

After using the Ceph OSDs `Select Nodes` dialog screen to select the item or items, the `OK` button can be clicked. The Ceph OSDs settings screen is the displayed again. This time the Ceph OSDs settings screen shows the selected item or items (figure 4.22):



Add nodes and categories to be configured as Ceph OSDs. For each node and category a number of OSD Associations, each corresponding to a particular OSD service, can be specified and configured using the 'Edit' button.

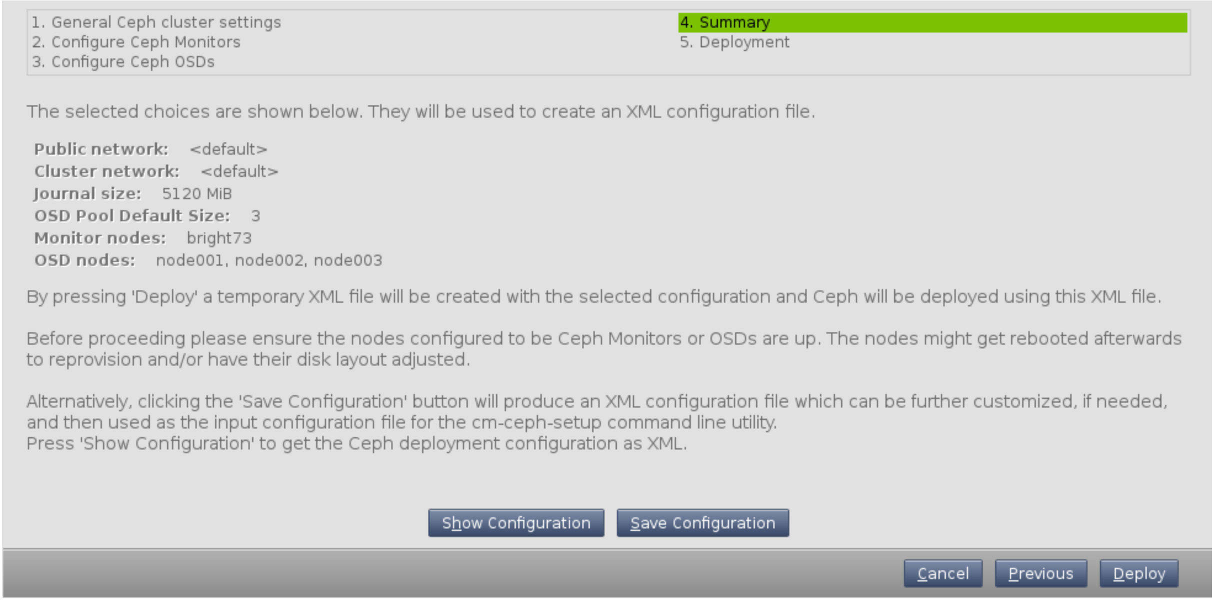
For production deployments it is recommended to have at least 3 OSD nodes.

Type	Name	Shared journal device	Associations
Node	node001		osd0
Node	node002		osd0
Node	node003		osd0

Figure 4.22: Ceph Wizard Installation: Ceph OSDs Selection

The next screen is the configuration summary screen (figure 4.23). Further details can be seen by clicking on the `Show Configuration` button, which shows the underlying raw XML configuration (figure 4.24).

In figure 4.23 the `Save Configuration` option can be used to save the Ceph setup configuration to a file. The `Deploy` button proceeds with deploying Ceph according to the configuration specified in the wizard.



The selected choices are shown below. They will be used to create an XML configuration file.

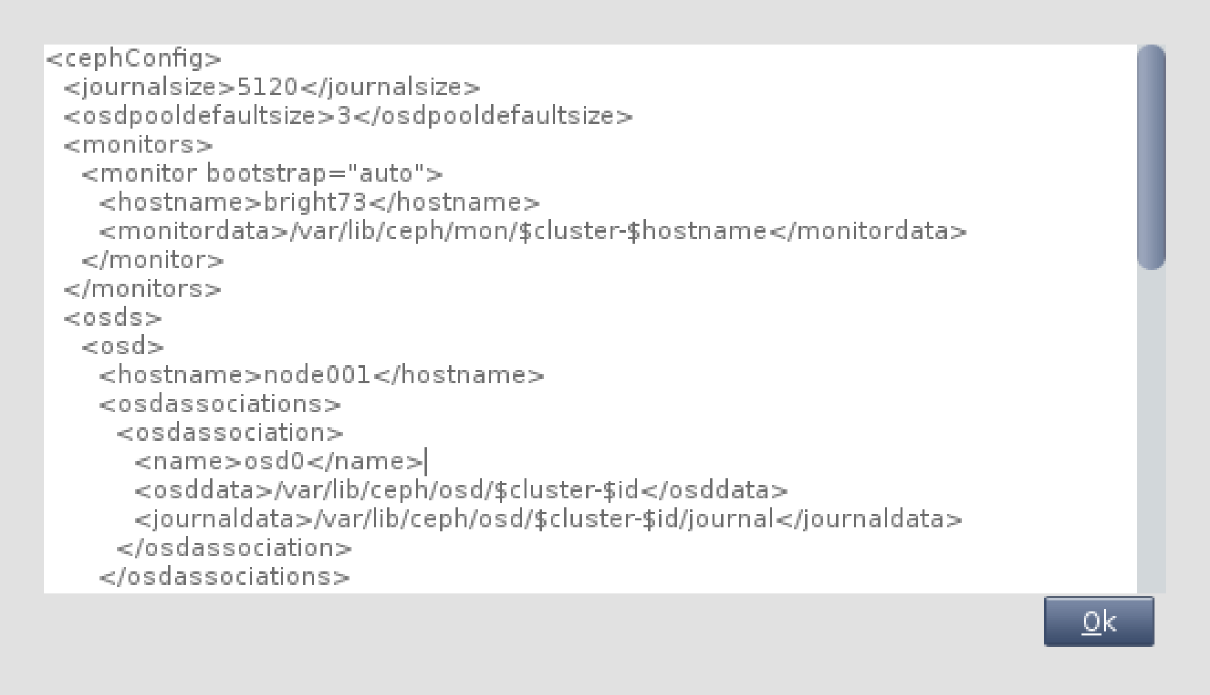
Public network: <default>
Cluster network: <default>
Journal size: 5120 MiB
OSD Pool Default Size: 3
Monitor nodes: bright73
OSD nodes: node001, node002, node003

By pressing 'Deploy' a temporary XML file will be created with the selected configuration and Ceph will be deployed using this XML file.

Before proceeding please ensure the nodes configured to be Ceph Monitors or OSDs are up. The nodes might get rebooted afterwards to reprovision and/or have their disk layout adjusted.

Alternatively, clicking the 'Save Configuration' button will produce an XML configuration file which can be further customized, if needed, and then used as the input configuration file for the cm-ceph-setup command line utility.
Press 'Show Configuration' to get the Ceph deployment configuration as XML.

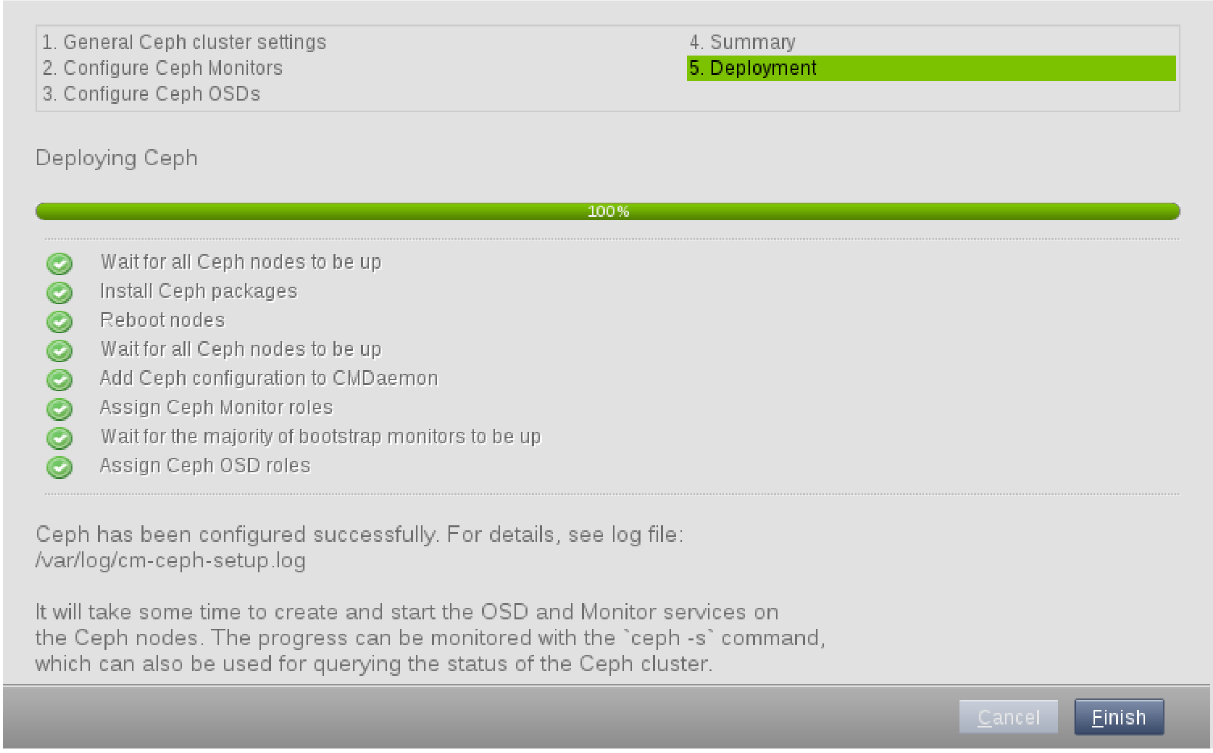
Figure 4.23: Ceph Wizard Installation: Configuration Summary



```
<cephConfig>
  <journalsize>5120</journalsize>
  <osdpooldefaultsize>3</osdpooldefaultsize>
  <monitors>
    <monitor bootstrap="auto">
      <hostname>bright73</hostname>
      <monitordata>/var/lib/ceph/mon/$cluster-$hostname</monitordata>
    </monitor>
  </monitors>
  <osds>
    <osd>
      <hostname>node001</hostname>
      <osdassociations>
        <osdassociation>
          <name>osd0</name>
          <osddata>/var/lib/ceph/osd/$cluster-$id</osddata>
          <journaldata>/var/lib/ceph/osd/$cluster-$id/journal</journaldata>
        </osdassociation>
      </osdassociations>
```

Figure 4.24: Ceph Wizard Installation: Show XML Configuration

During the deployment process, the progress is displayed (figure 4.25).



Deploying Ceph

- ✓ Wait for all Ceph nodes to be up
- ✓ Install Ceph packages
- ✓ Reboot nodes
- ✓ Wait for all Ceph nodes to be up
- ✓ Add Ceph configuration to CMDaemon
- ✓ Assign Ceph Monitor roles
- ✓ Wait for the majority of bootstrap monitors to be up
- ✓ Assign Ceph OSD roles

Ceph has been configured successfully. For details, see log file:
/var/log/cm-ceph-setup.log

It will take some time to create and start the OSD and Monitor services on the Ceph nodes. The progress can be monitored with the `ceph -s` command, which can also be used for querying the status of the Ceph cluster.

Figure 4.25: Ceph Wizard Installation: Deployment Progress

The event viewer in cmgui also shows the changes taking place.

After deployment, the OSD and Monitor services take some time be created and to start up. When all is up and running, the status of a healthy system, according to the output of the ceph -s command, should look something like the following:

Example

```
[root@bright73 ~]# ceph -s
    cluster 163589aa-c50e-46f3-8514-6f053cae5f2a
     health HEALTH_OK
     monmap e1: 1 mons at bright73=10.141.255.254:6789/0
            election epoch 3, quorum 0 bright73
     osdmap e7: 3 osds: 3 up, 3 in
            flags sortbitwise,require_jewel_osds
      pgmap v1005: 64 pgs, 1 pools, 0 bytes data, 0 objects
            25291 MB used, 5392 MB / 30684 MB avail
                  64 active+clean
```

A nearly full Ceph system, which is still functioning properly, would show something like:

Example

```
[root@bright73 ~]# ceph -s
    cluster 3404ee8a-1c3d-44b0-bbb8-3d2c5ae539f2
     health HEALTH_WARN
            3 near full osd(s)
     monmap e1: 1 mons at bright73=10.141.255.254:6789/0
            election epoch 3, quorum 0 bright73
     osdmap e9: 3 osds: 3 up, 3 in
            flags nearfull,sortbitwise,require_jewel_osds
```

```
pgmap v38: 64 pgs, 1 pools, 0 bytes data, 0 objects
      26407 MB used, 4276 MB / 30684 MB avail
            64 active+clean
```

5

User Management And Getting OpenStack Instances Up

In this chapter:

- Section 5.1 describes Bright Cluster Manager's user management integration with OpenStack.
- Section 5.2 describes how a user instance can be run with OpenStack under Bright Cluster Manager. A user instance is an instance that is not a tightly-integrated Bright-managed instance. A Bright-managed instance is a special case of an user instance. Bright-managed nodes are treated by Bright Cluster Manager very much like a regular nodes.
- Section 5.3 describes how a Bright-managed instance is managed in Bright Cluster Manager

5.1 Bright Cluster Manager Integration Of User Management In OpenStack

User management in Bright Cluster Manager without OpenStack is covered in Chapter 6 of the *Administrator Manual*. Users managed in this way are called *Bright users*.

OpenStack allows a separate set of users to be created within its projects. By default, these *OpenStack users* are set up to be independent of the Bright users.

OpenStack user accounts are of two kinds:

- regular users: these are end users who get to use an OpenStack user instance or a Bright-managed OpenStack instance. These can be managed by Bright Cluster Manager's LDAP, or can also simply be managed within OpenStack, depending on the Keystone backend driver used.
- service users: these user accounts are used to run the OpenStack service components. They are associated with the `service` project and `admin` role. Thus, the Nova service has a `nova` user, the Cinder service has a `cinder` user, and so on, and these are all assigned an `admin` role. The list of service user names can be listed in the default installation as follows:

```
[bright73->openstack[default]->roleassignments]% list -f name:25 | grep service
admin:service:admin
cinder:service:admin
cmdaemon:service:admin
glance:service:admin
heat:service:admin
keystone:service:admin
neutron:service:admin
nova:service:admin
radosgw:service:admin
```

The OpenStack service users are stored in the Keystone database, managed by the OpenStack MariaDB running on the controller nodes.

In `cmsh` the role assignment name field is in the form of:

<OpenStack user name> : *<project>* : *<role>*

The background note on page 79 has some further details on role assignment in Bright Cluster Manager OpenStack edition.

The service user `radosgw` is created only if the RADOS GW is installed (section 4.4).

Regular OpenStack users can be created in several ways, including:

- using `cmsh`, from within `openstack` mode
- using `cmgui`, from within the OpenStack tab
- using the OpenStack Horizon dashboard, where clicking on the `Identity` sidebar resource leads to the `Users` management window
- using the `openstack` command line utility that comes with OpenStack.
- using the Keystone Python API, which is an option that is more likely to be of interest to developers rather than system administrators

The details of how this is carried out depends on user database backend configuration. OpenStack users and Bright users can be given the same name and password in several ways, depending on the database driver used by Keystone (section 3.1.4), and how the administrator configures the users using the initialization and migration scripts (section 5.1.2).

Having the OpenStack service users not be in the Bright Cluster Manager LDAP and thus not be the same as Bright users has some advantages.

Having OpenStack regular users be the same as Bright users is also something that administrators may want.

Background Note: The User Database Drivers, User Migration And Initialization

This section on database drivers is offered as background material to provide a deeper understanding of user management in Bright Cluster Manager with OpenStack. It can be skipped by administrators who have no need to understand how the configuration can be customized, or who have been provided with a customized configuration already.

It should be understood that Bright users are not OpenStack users by default when OpenStack setup is carried out. To make a Bright user able to use OpenStack under the same user name, some configuration must be carried out. The exact configuration depends upon the use case. The main configuration involves the type of backend user database driver used, and can additionally include the option of initialization and migration scripts.

Initialization and migration scripts are scripts that can be used to initialize and migrate Bright users to become OpenStack users, after OpenStack setup has been carried out.

In this background note, two kinds of Bright users are defined:

1. **legacy users**: These are Bright users created from before the OpenStack initialization and migration scripts are working.
2. **fresh users**: These are Bright users created after the OpenStack initialization and migration scripts are working.

The following table displays the driver configuration options that allow the Bright Cluster Manager user to use OpenStack.

Keystone Driver	Legacy And Fresh Bright Access To OpenStack	Use To Create New OpenStack Users In OpenStack?
MySQL	No, but migrating Bright users to become regular OpenStack users often makes sense, and can be done for fresh users automatically by configuring initialization and migration scripts.	Yes, OpenStack does it in a self-contained manner within the `members` project without having to configure OpenStack user initialization or user migration scripts. When the initialization and migration scripts are configured, then creating a fresh Bright user can still create a new OpenStack user with the same name.
MySQL + PAM/NSS (Hybrid)	Yes, via `pam` domain, and using role and project assignments.	Not recommended. Typically set up PAM users instead
Bright LDAP	Yes, via `ldap` domain, and some OpenStack project and OpenStack role assignment	No.

The first Keystone driver, MySQL: has Keystone use only Galera's MySQL database for OpenStack users, that is for both the service OpenStack and the regular OpenStack users. It means that Bright Cluster Manager's regular LDAP user database remains in use as another, independent database for Bright users, and these users cannot be used for OpenStack functionality unless the users are duplicated across from Bright Cluster Manager's regular LDAP into the OpenStack domain. Thus, without that duplication, the regular OpenStack users are created by OpenStack actions and are stored in the Galera MySQL database, in the `default` domain associated with a default OpenStack installation.

Not having unified user databases—having the OpenStack MySQL user database distinct from Bright Cluster Manager's regular LDAP user database—means that using the Keystone MySQL driver is typically used for proof-of-concept deployments, or small deployments, rather than larger scale deployments.

User duplication from the Bright Cluster Manager user names to the OpenStack users can be useful for this driver: If a migration script and an initialization script are configured to run on the Bright Cluster Manager user name in CMDaemon (section 5.1.2), then fresh Bright users, when created, have their names duplicated as OpenStack user names, and these names are stored in Galera as well as in the regular LDAP user database. Legacy Bright users are not migrated or initialized by this configuration. The databases remain independent, which means that passwords for a duplicated user name are not matched. The passwords can of course be matched manually by the end user.

The second driver, MySQL + PAM/NSS (Hybrid): has Keystone using Galera's MySQL and also Bright Cluster Manager's PAM/NSS, and is called a hybrid driver. The driver handles the `admin`, `cmdaemon`, and OpenStack service users via Galera's MySQL in the OpenStack domain called `default`. On the other hand, all other users—Bright PAM/NSS authenticated users, and any other PAM/NSS authenticated users—are authenticated via PAM/NSS through this driver, and access OpenStack via the special OpenStack domain `pam`. The Bright Cluster Manager administrator is therefore normally only concerned with the PAM/NSS part of the driver when it concerns managing users.

A convenience with this driver is that there is only one password per user, so that this driver is typically used for larger deployments. It is also a cleaner deployment, having normal users placed in the pam domain and handling them there. Also, if using Bright Cluster Manager for user management, then the administrator can manage passwords and other properties in the standard Bright Cluster Manager way from the top-level cmsh user mode.

With this driver, Bright users that are authenticated with LDAP, can be authenticated by Keystone via PAM/NSS. The driver assigns the user the OpenStack pam domain. Within the OpenStack pam domain, an assignment must be carried out by the administrator for the OpenStack role and for the OpenStack project. Without these role and project assignments within the pam domain, the users are merely authenticated, but disallowed the use of OpenStack services. Typically, therefore, to manage the PAM users in the pam domain of OpenStack, an administrative user, for example, pamadmin, can be created within the pam domain, and given the OpenStack admin role. Such a pamadmin administrator is a separate user from the admin created by default in the default domain. This pamadmin can then assign an appropriate OpenStack role and OpenStack project to the user in the pam domain.

User duplication from the Bright user names to the OpenStack users, using a migration script and an initialization script, is typically not useful for this driver, since it works against the clean placement described earlier. If a migration script and an initialization script are configured to run on the Bright user name in CMDaemon (section 5.1.2), then fresh Bright users, when created, have their names duplicated as OpenStack user names, and these names are stored in Galera together with the service OpenStack users, as well as in the regular LDAP user database. Legacy Bright users are not migrated or initialized by this configuration. The databases remain independent, which means that passwords for a duplicated user name are not matched. The passwords can of course be matched manually by the end user.

The third driver, Bright LDAP: has Keystone using Bright Cluster Manager's own LDAP database, and does not use the OpenStack user database for regular users. That is, Keystone, when using this driver, handles Bright LDAP users only, ignores any NSS/PAM users, and ignores any regular OpenStack users in Galera. The admin, cmdaemon, and service OpenStack users, on the other hand, are still used by Keystone from Galera in OpenStack.

Creation of a fresh user via OpenStack actions will fail, because the LDAPS access from OpenStack is read-only. There is no account ldapadmin that can be created analogous to pamadmin that has the same abilities that pamadmin had with the second driver. That is, there is no account ldapadmin to assign projects and roles to LDAP users. Current LDAP users can be created via a CMDaemon front-end, such as the top-level user mode of cmsh in Bright, and are automatically go to the domain associated with OpenStack called ldap. OpenStack projects and OpenStack roles can be assigned to a user from the OpenStack command line. The convenience of a single password for users, the simple architecture, and having everything is contained within Bright Cluster Manager, means that this driver is typically useful for small or medium organizations that are using Bright Cluster Manager as is, without authenticating it to an external database via PAM/NSS.

An aside on duplication when using this driver: Duplication is mentioned here for completeness. It is available, but typically pointless for this driver. If a migration script and an initialization script are configured to run on the Bright user name in CMDaemon (section 5.1.2), then a fresh LDAP user name is duplicated during creation, as an OpenStack user name, and also stored in Galera, but not used from Galera. The databases remain independent, which means that passwords for a duplicated user name are not matched. The passwords can of course be matched manually by the end user. Legacy users are not migrated or initialized by this configuration.

Normally one of the three driver types is chosen in the user management screen during the wizard installation (section 3.1.4) or Ncurses installation (section 3.2.4).

However, the driver type can be added or removed after OpenStack installation, within cmsh by using the authbackends submode. For example, adding a name to the chosen driver type adds the driver while assigning it a name in CMDaemon:

Example

```
[bright73->openstack[default]->settings->authentication->authbackends]% add<TAB><TAB>
hybrid  ldap    sql
[bright73->...authbackends]]% add sql sql |#choosing sql as name for type sql backend
```

Further configuration to suit needs can be quite involved. It is therefore recommended to select the appropriate driver during a wizard or Ncurses installation to begin with.

5.1.1 Managing OpenStack Users As Bright Cluster Manager Users

Most administrators should find that the most convenient way to set up Bright Cluster Manager and OpenStack users is using cmsh. For Bright Cluster Manager users this is done from the main user mode, while for OpenStack users, it is done from within the users submodes, under OpenStack mode, in the cmsh hierarchy.

Background Note: Avoiding Confusion About User(s) And (Sub)Modes

The administrator should understand that there is a difference between:

- OpenStack->users submode: OpenStack users are managed from this submode
- OpenStack->settings->users submode: the settings for OpenStack users are managed from this submode
- Bright Cluster Manager user mode: Bright Cluster Manager users are managed from this mode

The following treeview illustrates these user(s) (sub)modes in the cmsh hierarchy:

```
[cmsh]
  |-- ...
  |-- openstack
  |    |-- ...
  |    |-- settings
  |    |    |--...
  |    |    `-- users
  |    |-- ...
  |    `-- users
  |-- ...
  `-- user
```

5.1.2 Synchronizing Users With The OpenStack Initialization And Migration Scripts

Setting the initialization and migration scripts: Bright Cluster Manager provides initialization and migration scripts that can be called after creating a Bright user. When applied to a Bright Cluster Manager user, the OpenStack user of the same name is created as follows:

- The migration script, /cm/local/apps/cluster-tools/bin/cm-user-migration, copies a Bright Cluster Manager user name from the LDAP records over to the OpenStack Keystone records, and by default sets a random password for the OpenStack user.
- The initialization script, /cm/local/apps/cluster-tools/bin/cm-user-init, creates an OpenStack project for the OpenStack user with the same name, if it does not already exist. The user is also assigned the member role. Role assignment here means that the OpenStack user is associated with a project and assigned a role for the purposes of the OpenStack utility (page 79, Background Note: Automated Role Assignment In OpenStack).

The cmsh parameters userinitscript and migrationscript can be set to these initialization and migration script paths. The parameters are initially blank by default. They can be set from within the OpenStack settings submode of cmsh for users as follows:

Example

```
[root@bright73 ~]# cmsh
[bright73]% openstack
[bright73->openstack[default]]% settings ; users
[...settings->users]% set userinitscript /cm/local/apps/cluster-tools/bin/cm-user-init
[...settings*->users*]% set migrationscript /cm/local/apps/cluster-tools/bin/cm-user-migration
[...settings*->users*]% commit
```

In `cmgui` the path parameters can be managed by first clicking on the `OpenStack` resource in the navigator, then going into the `Settings` tabbed pane, and then selecting the `Users` subtab, where the `Migration script:` and `User init script:` fields are displayed.

If the default scripts are set as in the preceding example, then they are automatically executed for the user when creating a regular Bright Cluster Manager user.

The administrator can customize the scripts, according to need, for example by copying them, then modifying the copies and assigning the modified copies to the `userinitscript` and `migrationscript` parameters.

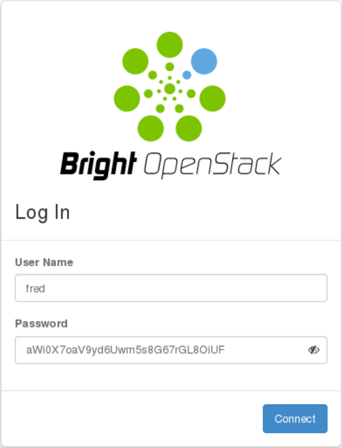
Automated OpenStack user creation: With the initialization and migration scripts set, OpenStack user creation now automatically takes place during regular user creation:

Example

```
[...settings->users]% user
[bright73->user]% add fred
[bright73->user*[fred*]]% set password secret123; commit
```

If Keystone uses the MySQL driver, then the password of the Bright Cluster Manager user and the password for the OpenStack user of the same name are independent. By default, the OpenStack user has a password that is random, and which the migration script places in `~/.openstackrc_password`.

To check that user `fred` can login as an OpenStack user, a login can be attempted via `http://`*<load balancer IP address>*`:10080/dashboard` using the password defined in his `.openstackrc_password` file (figure 5.1):

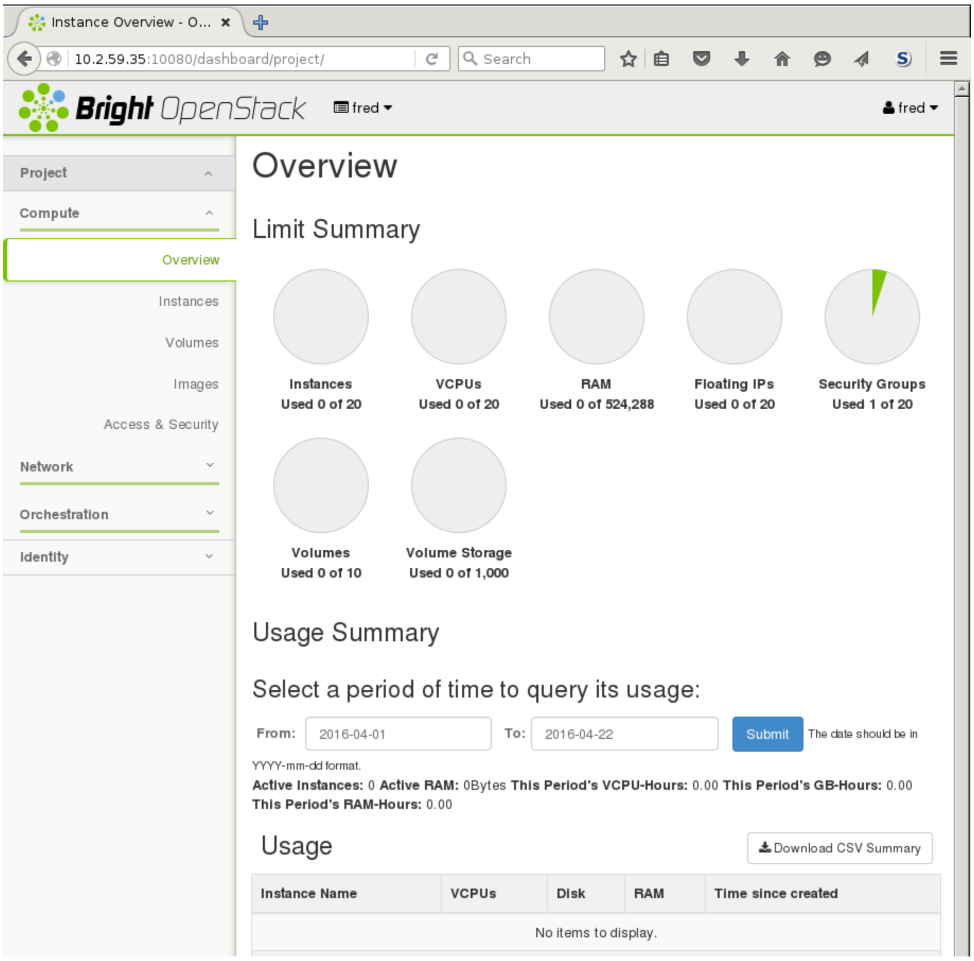


User Name: fred

Password: aWi0X7oaV9yd6Uwm5s8G67rGL8OiUF

Figure 5.1: Login With Horizon At `http://`*<load balancer IP address>*`:10080/dashboard`

If all is well, then the login for the end user succeeds and leads to an overview screen for the user (figure 5.2):



Select a period of time to query its usage:

From: 2016-04-01 To: 2016-04-22 The date should be in YYYY-mm-dd format.

Active Instances: 0 **Active RAM:** 0Bytes **This Period's VCPU-Hours:** 0.00 **This Period's GB-Hours:** 0.00 **This Period's RAM-Hours:** 0.00

Instance Name	VCPUs	Disk	RAM	Time since created
No items to display.				

Figure 5.2: Successful Login With Horizon At `http://`*<load balancer IP address>*`:10080/dashboard`

In an unmodified cluster there should be no instances running yet.

At this point, some background notes to help understand what is going on can be read by simply continuing with reading this chapter sequentially. Alternatively, if an administrator has a sufficiently deep understanding of and familiarity with Bright Cluster Manager and OpenStack, then it is possible to skip ahead to section 5.2, where getting an OpenStack instance up and running is described.

Background Note: Automated Role Assignment In OpenStack

If the default scripts for migration and initialization are in place, then the creation of a Bright user automatically creates an OpenStack user, with a default role assignment in the form of:

<OpenStack user name>:*<project>*:*<role>*

For example, creating the LDAP user `fred` in Bright Cluster Manager, automatically:

- creates an OpenStack user `fred`
- assigns the OpenStack user `fred` the default project `fred`, creating the project if needed
- assigns the OpenStack user `fred` the default role `member`
- assigns the OpenStack user `fred` a key `fred:fred:member` that can be used by the OpenStack utility

Example

```
[bright73->user[fred]]% openstack users
[bright73->openstack[default]->users]% list -f name
name (key)
--------------------
admin
cinder
cmdaemon
fred
glance
heat
keystone
neutron
nova
[bright73->openstack[default]->users]% projects
[bright73->openstack[default]->projects]% list
Name (key)   UUID (key)                        Domain             Enabled MOTD
------------ -------------------------------- ------------------ ------- ------
bright       83b48ea2016c4658b3b1e01a910011d9  Default (default)  yes
fred         b48cd2f6da4645a8886b494ad5f459c6  Default (default)  yes
service      aa239b1f054a470cbe40f74984a9331d  Default (default)  yes
[bright73->openstack[default]->projects]% roleassignments; list -f name,user,project,role
name (key)           user                 project              role
-------------------- -------------------- -------------------- --------------------
admin:bright:admin   admin (bfd7fd66b1ab+ bright (83b48ea2016+ admin (c7b7e8f8c885+
admin:service:admin  admin (bfd7fd66b1ab+ service (aa239b1f05+ admin (c7b7e8f8c885+
cinder:service:admin cinder (e173c5545c8+ service (aa239b1f05+ admin (c7b7e8f8c885+
cmdaemon:bright:adm+ cmdaemon (fae4250c3+ bright (83b48ea2016+ admin (c7b7e8f8c885+
cmdaemon:service:ad+ cmdaemon (fae4250c3+ service (aa239b1f05+ admin (c7b7e8f8c885+
fred:fred:member     fred (80e16841e3df2+ fred (b48cd2f6da464+ member (6cb5e5359b6+
glance:service:admin glance (2a0d739783d+ service (aa239b1f05+ admin (c7b7e8f8c885+
heat:service:admin   heat (7acdc31888534+ service (aa239b1f05+ admin (c7b7e8f8c885+
keystone:service:ad+ keystone (1048db4a5+ service (aa239b1f05+ admin (c7b7e8f8c885+
neutron:service:adm+ neutron (e1b01d92e9+ service (aa239b1f05+ admin (c7b7e8f8c885+
nova:service:admin   nova (634f35b3ee0e4+ service (aa239b1f05+ admin (c7b7e8f8c885+
[bright73->openstack[default]->roleassignments]%
```

Background Note: Automated Writing Out Of The .openstackrc* Files

OpenStack users have a .openstackrc file and a .openstackrc_password file associated with them. The .openstackrc file provides the OpenStack environment, while the .openstackrc_password file provides the OpenStack password. This environment can be used by openstack, the OpenStack utility that an OpenStack user can run to manage instances.

Within the settings submode of OpenStack there is a users submode. Within that users submode the administrator can set the following parameters to configure the .openstackrc* files:

- Write out OpenStack RC for users: This parameter configures how the .openstackrc file is written for an OpenStack user:
  - matchinghomedirectories: writes the file only to home directories that match OpenStack user names
  - allhomedirectories: writes the file to all home directories. That is, even if no OpenStack user matches that name
  - off: does not write out a file
- Write out .openstackrc_password: This parameter can take yes or no as its value. The value decides if the .openstackrc_password file is written for an OpenStack user. This feature

only operates when the user is created. So if this option is made active after user creation, then no password file is written out.

Example

```
[root@bright73 ~]# cmsh
[bright73]% openstack
[bright73->openstack[default]]% settings; users
[...settings->users]% set writeoutopenstackrcforusers matchinghomedirectories
[...settings->users*]% set writeout.openstackrc_password yes
[...settings->users*]% commit
```

With the preceding configuration for the `.openstackrc*` files, if an OpenStack user `fred` is created as in the example on page 78, then the home directory for `fred` would look something like:

Example

```
[root@bright73 ~]# ls -a /home/fred/
.  ..  .bash_logout  .bash_profile  .bashrc  .mozilla  .openstackrc  .openstackrc_password
```

The `.openstackrc*` file contents are similar to the following output:

Example

```
[root@bright73 ~]# cat /home/fred/.openstackrc_password
OS_PASSWORD="LMlr6oRENZoIp0iqaI4304JGNn632P"

[root@bright73 ~]# cat /home/fred/.openstackrc
# This section of this file was automatically generated by cmd. Do not edit manually!
# BEGIN AUTOGENERATED SECTION -- DO NOT REMOVE
# This file has been generated by the CMDaemon and is meant
# to be sourced from within the ~/.bashrc
export OS_TENANT_NAME=""
export OS_USERNAME="fred"
# Public Auth URL (used by users)
export OS_AUTH_URL="http://<load balancer IP address>:5000/v3"

# For keystone v3
export OS_PROJECT_DOMAIN_ID="default"
export OS_USER_DOMAIN_ID="default"
export OS_IDENTITY_API_VERSION=3 # for the 'openstack' utility to work
export OS_CACERT="/etc/keystone/ssl/certs/ca.pem"
# END AUTOGENERATED SECTION   -- DO NOT REMOVE
```

The value of <*load balancer IP address*> in the `.openstackrc` output is a dotted quad value or a resolvable host name, and is the address or name of the HAProxy load balancer that Bright Cluster Manager uses for its OpenStack deployment. The load balancer address is normally the IP address of the head node on the external network on a smaller cluster.

Background Note: Changing The End User OpenStack Password

The end user is given a password for OpenStack user access by the initialization script. This password, stored in `~/.openstackrc_password`, is long, and somewhat random. Most users would therefore like to change it to something that is easier for them to remember. This can be done in the dashboard by, for example, user `fred`, by clicking on the name `fred` in the top right hand corner, then selecting the `Settings` option, and then selecting the `Change Password` option.

The OpenStack APL CLI client `openstack` can be set to use the `.openstackrc` and `.openstackrc_password` files, which were initialized by the `cm-user-init` and `cm-user-migration` scripts earlier on (page 77). The end user can, if required, update the `~/.openstackrc_password` file by hand after a password change is made by the dashboard.

5.2 Getting A User Instance Up

By default, after creating a user as in the example where user fred is created (page 78) the user can log in as an OpenStack user. However, unless something extra has been prepared, a user that logs in at this point has no instances up yet. End users typically want an OpenStack system with running instances.

In this section, getting an instance up and running is used to illustrate the management of OpenStack in Bright Cluster Manager.

5.2.1 Making An Image Available In OpenStack

A handy source of available images is at http://docs.openstack.org/image-guide/obtain-images.html. The URI lists where images for major, and some minor distributions, can be picked up from.

Cirros is one of the distributions listed there. It is a distribution that aims at providing a small, but reasonably functional cloud instance. The Cirros image listed there can therefore be used for setting up a small standalone instance, suitable for an m1.xtiny flavor, which is useful for basic testing purposes.

Installing The Image Using The openstack Utility

If the qcow2 image file cirros-0.3.4-x86_64-disk.img, 13MB in size, is picked up from the site and placed in the local directory, then an image cirros034 can be set up and made publicly available by the administrator or user by using the openstack image create command as follows:

Example

```
[fred@bright73 ~]$ wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
...
2016-05-10 14:19:43 (450 KB/s) - ‘cirros-0.3.4-x86_64-disk.img’ saved [13287936/13287936]
[fred@bright73 ~]$ openstack image create --disk-format qcow2 --file\
 cirros-0.3.4-x86_64-disk.img cirros034
```

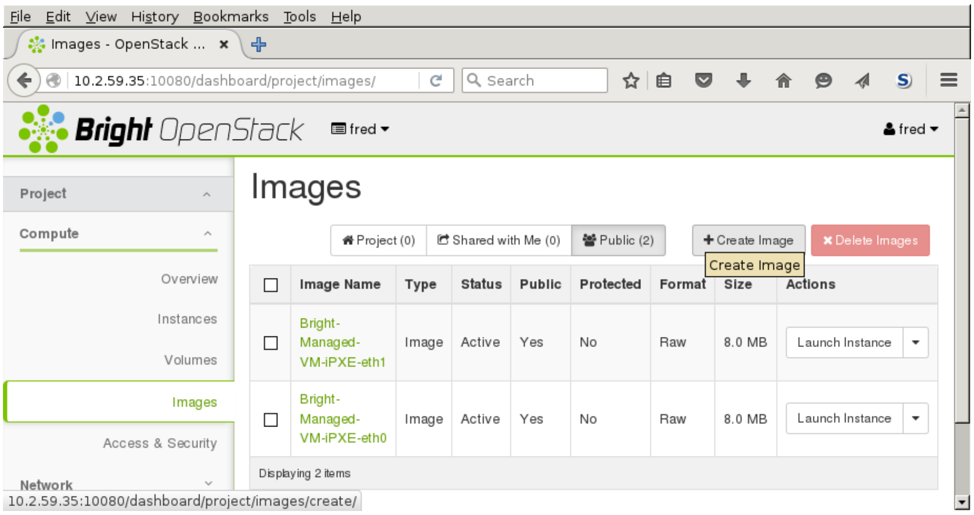
The openstack command in the preceding example assumes that the .openstackrc has been generated, and sourced, in order to provide the OpenStack environment. The .openstackrc file is generated by setting the Write out OpenStack RC for users option (page 80), and it can be sourced with:

Example

```
[fred@bright73 ~]$ . .openstackrc
```

Sourcing in this case means that running the file sets the environment variables in the file, so that after returning to the shell the shell now has these environment variables.

If all goes well, then the image is installed and can be seen by the user or administrator, via OpenStack Horizon, by navigation to the Images pane, or using the URI http://*<load balancer hostname, or IP address>*:10080/dashboard/project/images/ directly (figure 5.3).



Images

□	Image Name	Type	Status	Public	Protected	Format	Size	Actions
□	Bright-Managed-VM-iPXE-eth1	Image	Active	Yes	No	Raw	8.0 MB	Launch Instance
□	Bright-Managed-VM-iPXE-eth0	Image	Active	Yes	No	Raw	8.0 MB	Launch Instance

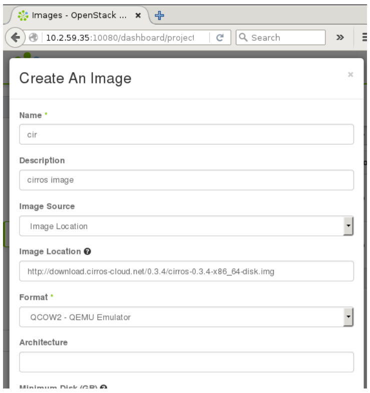
Displaying 2 items

Figure 5.3: Images Pane In Horizon

Installing The Image Using Horizon

Alternatively, instead of using the openstack utility, the image can also be installed by the user or administrator using OpenStack Horizon directly. The Horizon procedure to do this is described next:

Clicking on the Create Image button of the Images pane launches a pop-up dialog. Within the dialog, a name for the image for OpenStack users can be set, the disk format of the image can be selected, the HTTP URL from where the image can be picked up can be specified, and the image can be made public (figure 5.4).



Create An Image

Name *: cir

Description: cirros image

Image Source: Image Location

Image Location: http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img

Format *: QCOW2 - QEMU Emulator

Architecture:

Minimum Disk (GB)

Figure 5.4: Images Pane—Create Image Dialog

The State Of The Installed Image

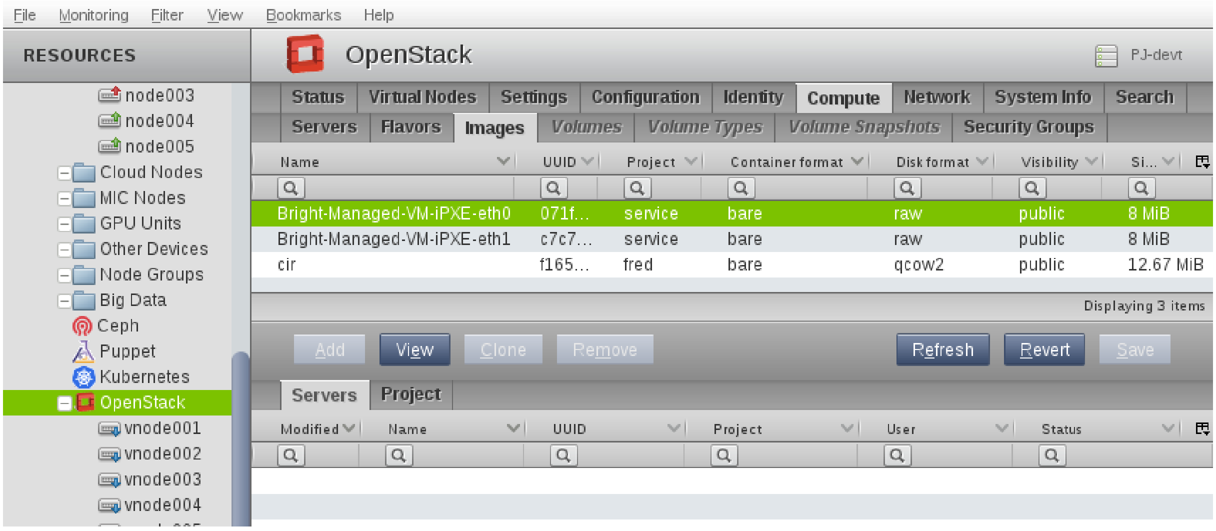
After the image has been installed by user fred, then it is available for launching instances by fred. If the checkbox for Public was ticked in the previous dialog, then other OpenStack users can also use it to launch their instances.

It should however be pointed out that although the image is available, it is not yet ready for launch. The reasons for this are explained shortly in section 5.2.2.

The image properties can be viewed as follows:

- by the authorized OpenStack users with OpenStack Horizon, by clicking through for Image Details
- by cmsh, from within the images submode of openstack mode.

- using cmgui, from within the OpenStack resource tabbed pane, then within the Compute subtab, and then within the Images subsubtab (figure 5.5).



Name	UUID	Project	Container format	Disk format	Visibility	Si...
Bright-Managed-VM-iPXE-eth0	071f...	service	bare	raw	public	8 MiB
Bright-Managed-VM-iPXE-eth1	c7c7...	service	bare	raw	public	8 MiB
cir	f165...	fred	bare	qcow2	public	12.67 MiB

Displaying 3 items

Figure 5.5: OpenStack Image Properties In cmgui

5.2.2 Creating The Networking Components For The OpenStack Image To Be Launched

Launching an image that is installed as in section 5.2.1 needs networking components to be configured with it, so that it can work within OpenStack, and so that it can be managed by OpenStack. An instance that is up, but has no networking set up for it, cannot launch an image to get a virtual machine up and running.

Why Use A New Network For An End User?

If it is the OpenStack administrator, admin that is preparing to launch the instance, as a bright project, then the OpenStack launch dialog by default allows the instance to use the default flat internal network of the cluster, bright-internal-flat-internalnet. As instances are run with root privileges, this means that all the internal network traffic can be read by whoever is running the instance. This is a security risk and would be a bad practice.

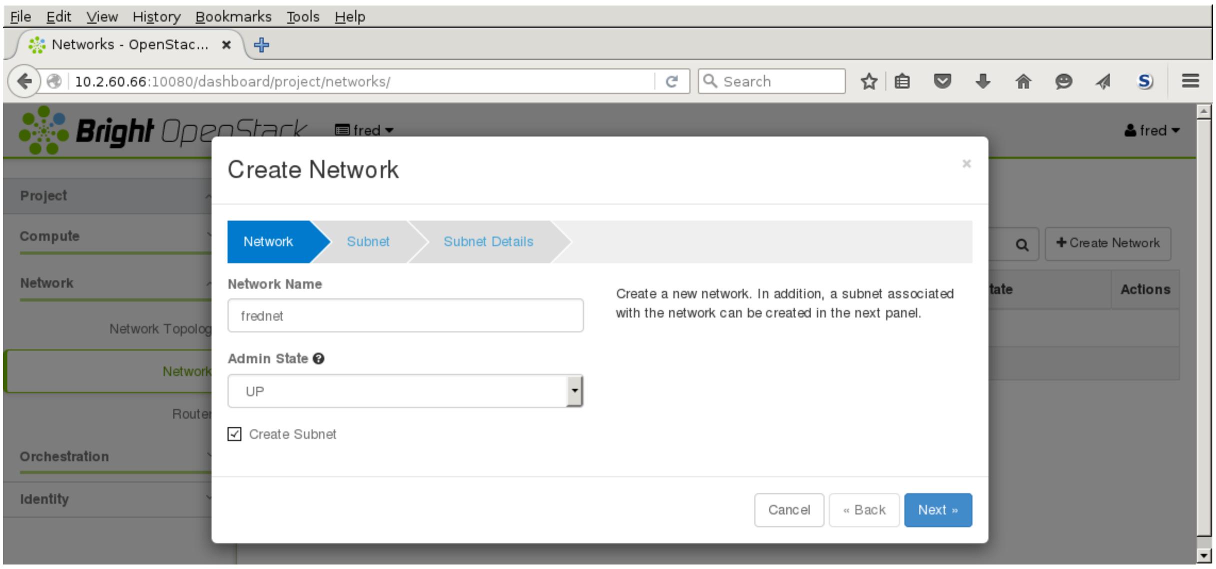
By default, therefore, the non-admin end user cannot launch the instance using the flat internal network of the cluster. The end user therefore typically has to create a new network, one that is isolated from the internal network of the cluster, in order to launch an instance.

This is thus the case for the end user fred, who earlier on had logged into the OpenStack dashboard and created an image by the end of section 5.2.1. User fred cannot run the image in the instance until a network exists for the future virtual machine.

Creating The Network With Horizon

For the sake of this example and clarity, a network can be created in OpenStack Horizon, using the Network part of the navigation menu, then selecting Networks. Clicking on the Create Network button on the right hand side opens up the Create Network dialog box.

In the first screen of the dialog, the network for fred can be given the unimaginative name of frednet (figure 5.6):



Create Network

Network Name: frednet

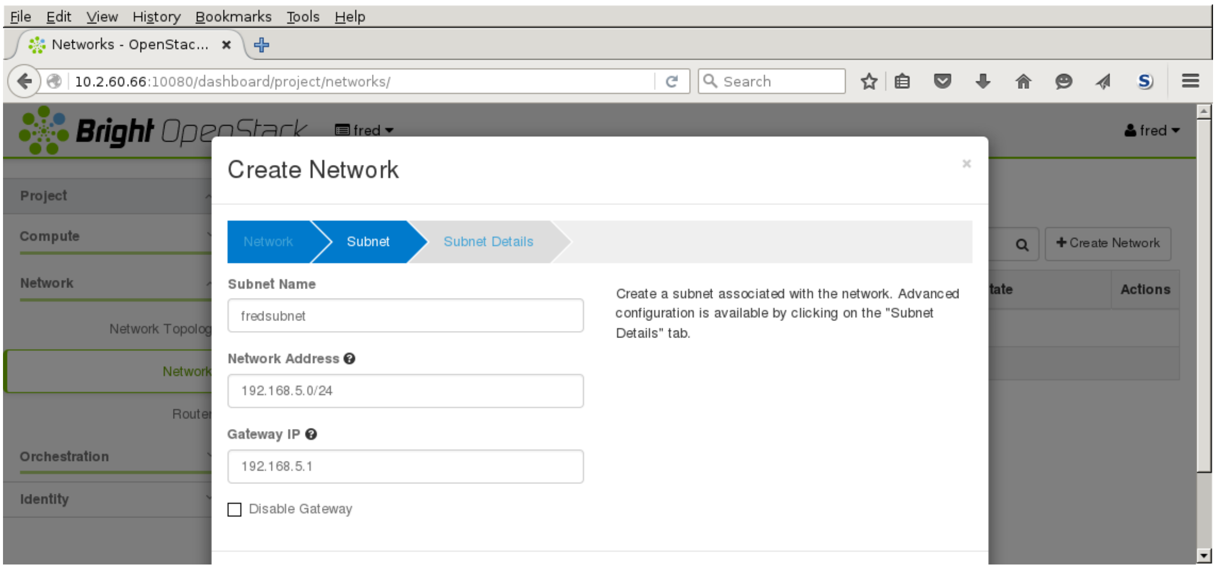
Admin State: UP

☑ Create Subnet

Create a new network. In addition, a subnet associated with the network can be created in the next panel.

Figure 5.6: End User Network Creation

Similarly, in the next screen a subnet called `fredsubnet` can be configured, along with a gateway address for the subnet (figure 5.7):



Create Network

Subnet Name: fredsubnet

Network Address: 192.168.5.0/24

Gateway IP: 192.168.5.1

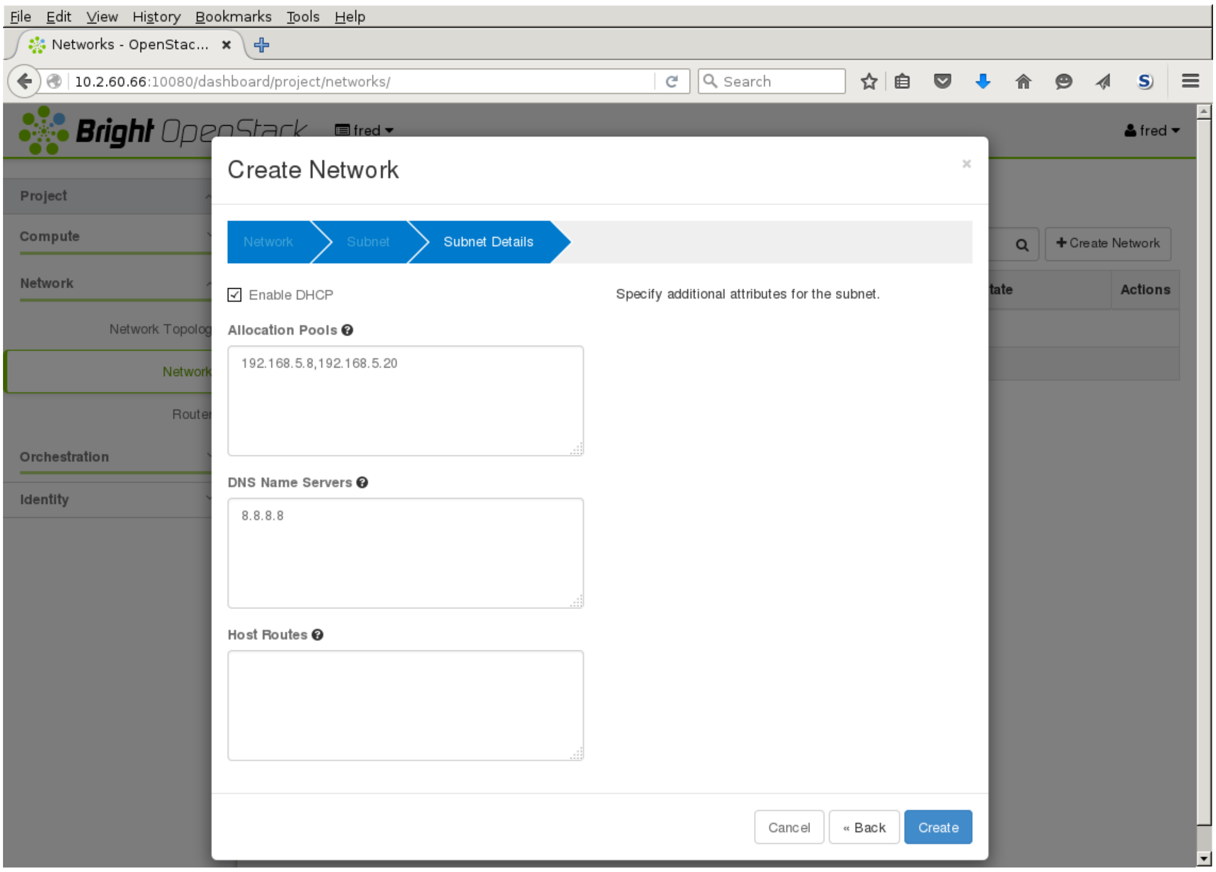
□ Disable Gateway

Create a subnet associated with the network. Advanced configuration is available by clicking on the "Subnet Details" tab.

Figure 5.7: End User Subnet Creation

In the next screen (figure 5.8):

- a range of addresses on the subnet is earmarked for DHCP assignment to devices on the subnet
- a DNS address is set
- special routes for hosts can be set



☑ Enable DHCP

Specify additional attributes for the subnet.

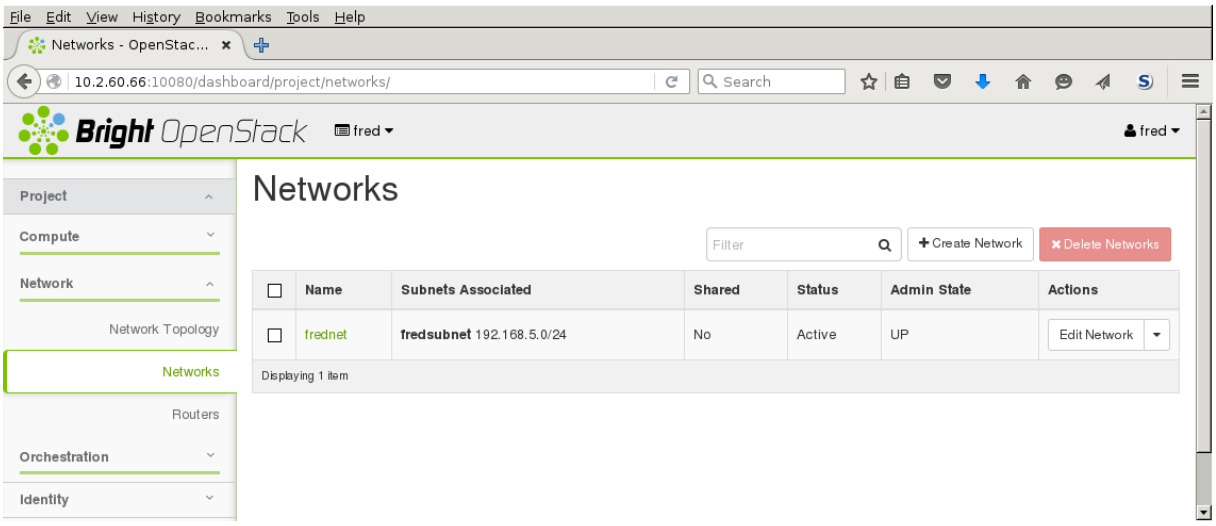
Allocation Pools: 192.168.5.8,192.168.5.20

DNS Name Servers: 8.8.8.8

Host Routes:

Figure 5.8: End User DHCP, DNS, And Routes

At the end of a successful network creation, when the dialog box has closed, the screen should look similar to figure 5.9:



□	Name	Subnets Associated	Shared	Status	Admin State	Actions
□	frednet	**fredsubnet** 192.168.5.0/24	No	Active	UP	Edit Network ▾

Displaying 1 item

Figure 5.9: End User Node Network Configuration Result

The State Of The Image With Its Network Configured

At this point, the image can be launched, for example using Horizon's `Compute` resource in the navigation panel, then choosing the `Instances` pane, and then clicking on the `Launch Instance` button.

On launching, the image will run. However, it will only be accessible via the OpenStack console, which has some quirks, such as only working well in fullscreen mode in some browsers.

It is more pleasant and practical to login via a terminal client such as `ssh`. How to configure this is described next.

5.2.3 Accessing The Instance Remotely With A Floating IP Address

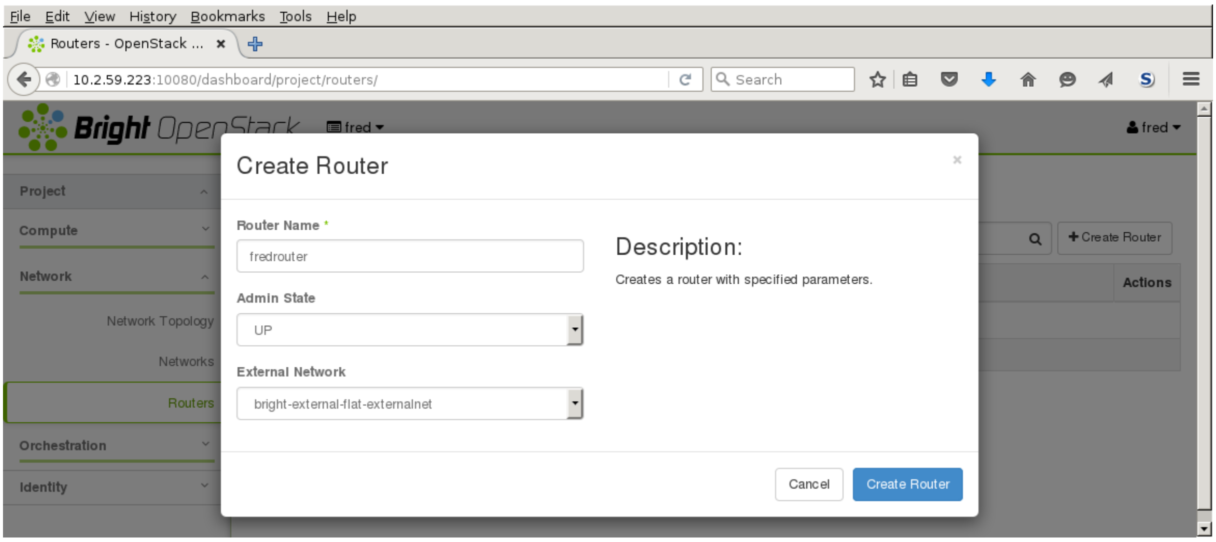
For remote access from outside the cluster, this is possible if a floating IP address, that is from the external network, has been configured for instances on the OpenStack network. The floating IP address is taken from the pool of addresses specified earlier during OpenStack installation (section 3.1.13). The subnet for these addresses needs to be accessible via a router. The configuration of such a router is described in the next subsection.

For remote access from within the cluster, an alternative method to creating a floating IP address, is for the administrator to configure the Bright Cluster Manager internal network to be a shared external network from the point of view of the instance. Sharing the internal network in this way is a security risk due to the reasons given earlier on on page 84. However, it may be appropriate in an isolated cluster with no external network, and with trusted users, in which case the administrator can mark the Bright Cluster Manager internal network from OpenStack Horizon as `shared`.

Remote access from outside the cluster with a floating IP address can be configured as follows:

Router Configuration For A Floating IP Address

Router configuration for a floating IP address with Horizon: A router can be configured from the `Network` part of the navigation menu, then selecting `Routers`. Clicking on the `Create Router` button on the right hand side opens up the `Create Router` dialog box (figure 5.10):



Create Router

Router Name * fredrouter

Admin State UP

External Network bright-external-flat-externalnet

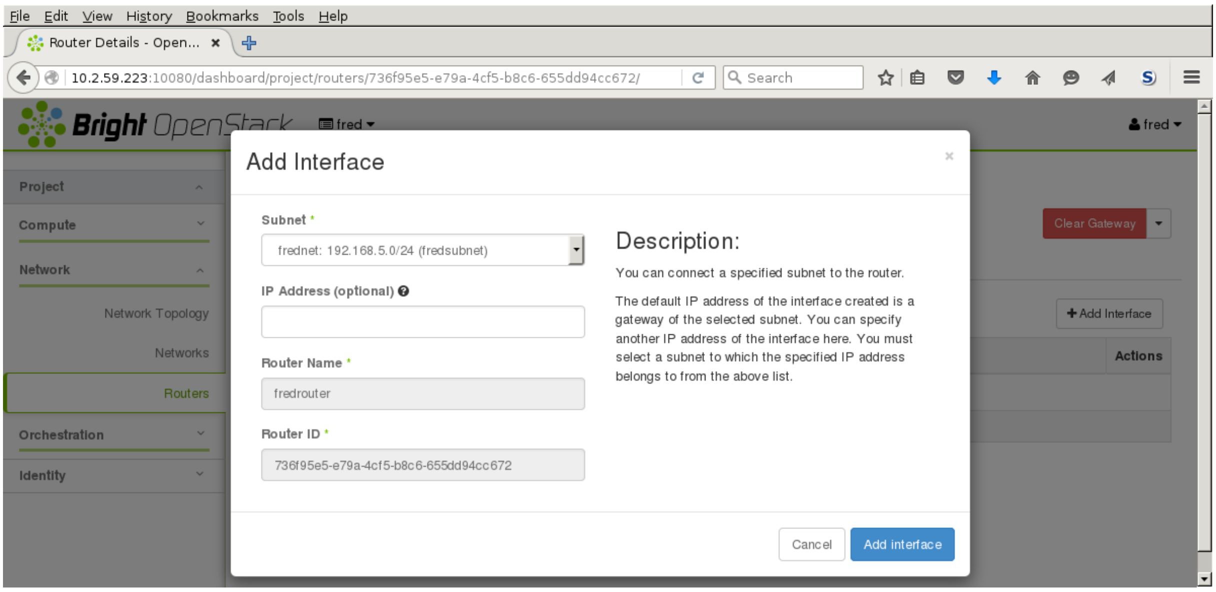
Description:

Creates a router with specified parameters.

Figure 5.10: End User Router Creation

The router can be given a name, and connected to the external network that provides the floating IP addresses of the cluster.

Next, an extra interface for connecting to the network of the instance can be added by clicking on the router name, which brings up the `Router Details` page. Within the `Interfaces` subtab, the `Add Interface` button on the right hand side opens up the `Add Interface` dialog box (figure 5.11):



Add Interface

Subnet *
frednet: 192.168.5.0/24 (fredsubnet)

IP Address (optional)

Router Name *
fredrouter

Router ID *
736f95e5-e79a-4cf5-b8c6-655dd94cc672

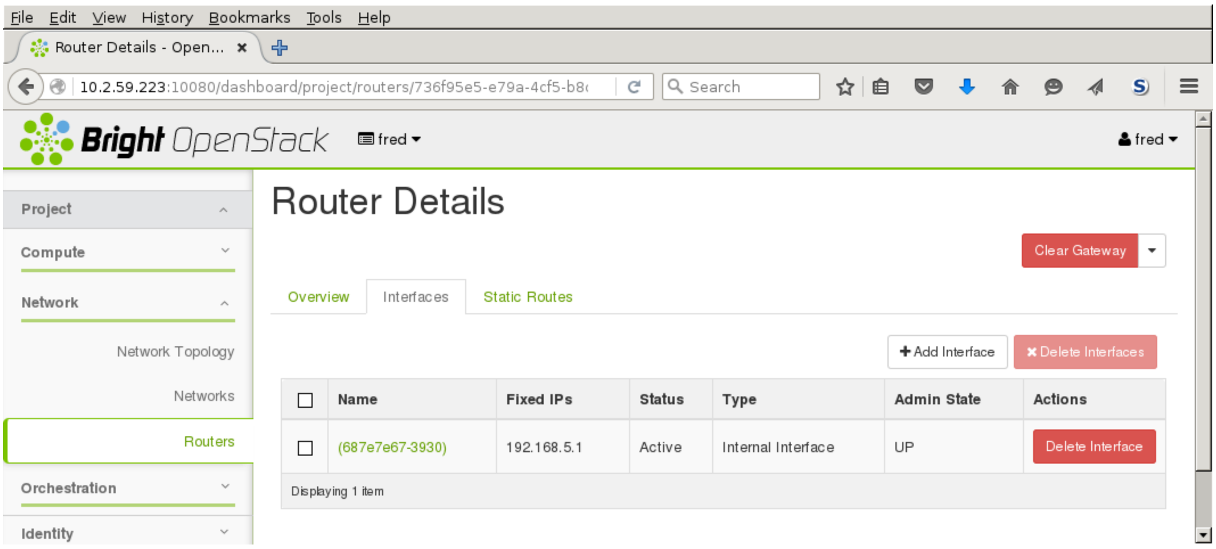
Description:

You can connect a specified subnet to the router.

The default IP address of the interface created is a gateway of the selected subnet. You can specify another IP address of the interface here. You must select a subnet to which the specified IP address belongs to from the above list.

Figure 5.11: End User Router Interfaces Creation

After connecting the network of the instance, the router interface IP address should be the gateway of the network that the instance is running on (figure 5.12):



Router Details

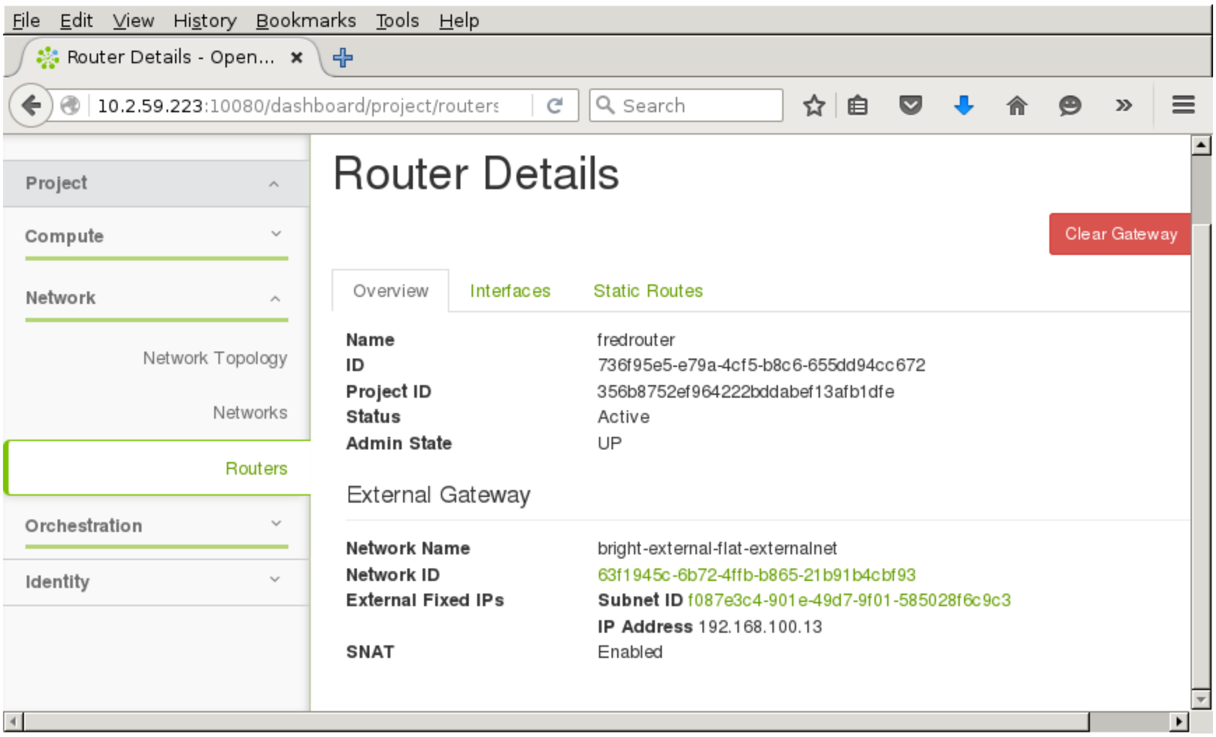
□	Name	Fixed IPs	Status	Type	Admin State	Actions
□	(687e7e67-3930)	192.168.5.1	Active	Internal Interface	UP	Delete Interface

Displaying 1 item

Figure 5.12: End User Router Interface Screen After Router Configuration

The state of the router after floating IP address configuration: To check the router is reachable from the head node, the IP address of the router interface connected to the cluster external network should show a ping response.

The IP address can be seen in the `Overview` subtab of the router (figure 5.13):



Router Details

Name fredrouter
ID 736f95e5-e79a-4cf5-b8c6-655dd94cc672
Project ID 356b8752ef964222bddabef13afb1dfe
Status Active
Admin State UP

External Gateway

Network Name bright-external-flat-externalnet
Network ID 63f1945c-6b72-4ffb-b865-21b91b4cbf93
External Fixed IPs **Subnet ID** f087e3c4-901e-49d7-9f01-585028f6c9c3
IP Address 192.168.100.13
SNAT Enabled

Figure 5.13: End User Router Details After Router Configuration

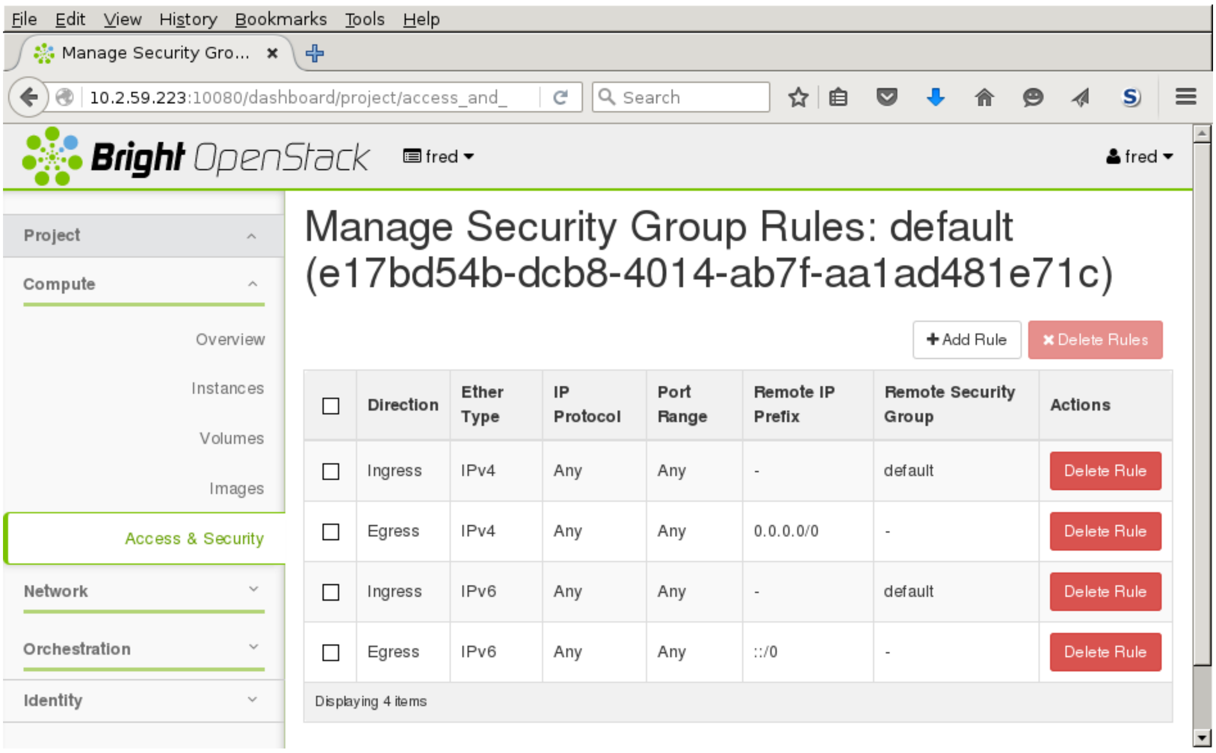
A ping behaves as normal for the interface on the external network:

Example

```
[fred@bright73 ~]$ ping -c1 192.168.100.13
PING 192.168.100.13 (192.168.100.13) 56(84) bytes of data.
64 bytes from 192.168.100.13: icmp_seq=1 ttl=64 time=0.383 ms

--- 192.168.100.13 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.383/0.383/0.383/0.000 ms
```

Security group rules to allow a floating IP address to access the instance: The internal interface to the instance is still not reachable via the floating IP address. That is because by default there are security group rules that set up iptables to restrict ingress of packets across the hypervisor.

The rules can be managed by accessing the `Compute` resource, then selecting the `Access & Security` page. Within the `Security Groups` subtab there is a `Manage Rules` button. Clicking the button brings up the `Manage Security Group Rules` table (figure 5.14):



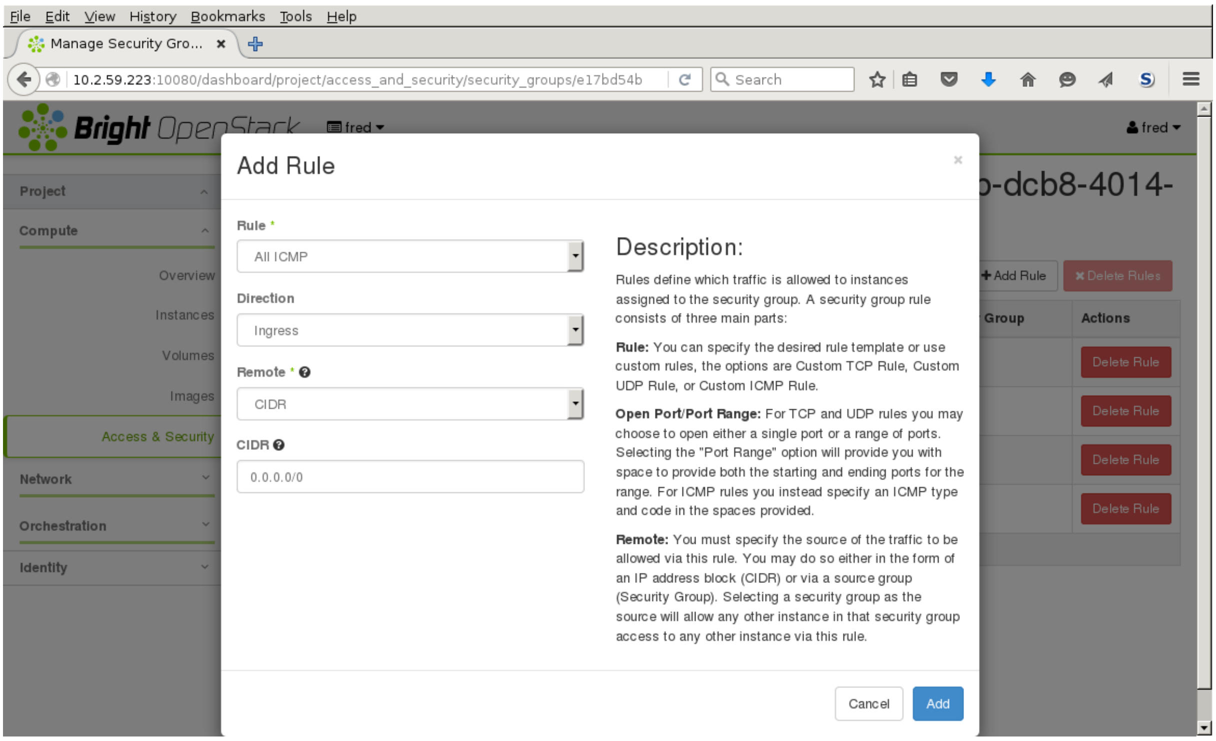
Manage Security Group Rules: default (e17bd54b-dcb8-4014-ab7f-aa1ad481e71c)

□	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
□	Ingress	IPv4	Any	Any	-	default	Delete Rule
□	Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
□	Ingress	IPv6	Any	Any	-	default	Delete Rule
□	Egress	IPv6	Any	Any	::/0	-	Delete Rule

Displaying 4 items

Figure 5.14: Security Group Rules Management

Clicking on the `Add Rule` button brings up a dialog. To let incoming pings work, the rule `All ICMP` can be added. Further restrictions for the rule can be set in the other fields of the dialog for the rule (figure 5.15).



Add Rule

Rule *: All ICMP

Direction: Ingress

Remote *: CIDR

CIDR: 0.0.0.0/0

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

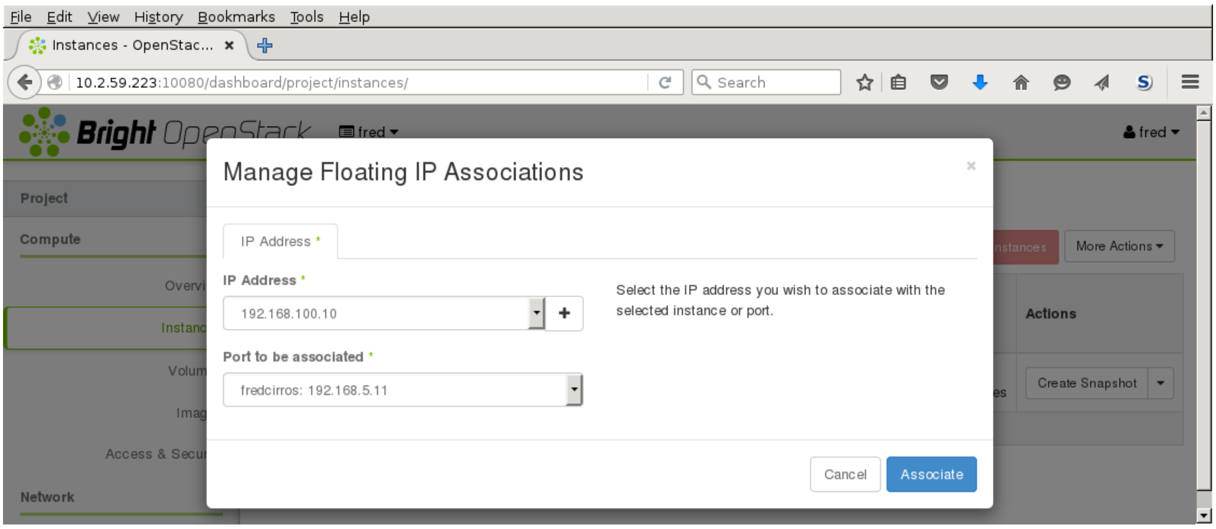
Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Figure 5.15: Security Group Rules Management—Adding A Rule

Floating IP address association with the instance: The floating IP address can now be associated with the instance. One way to do this is to select the Compute resource in the navigation window, and select Instances. In the Instances window, the button for the instance in the Actions column allows an IP address from the floating IP address pool to be associated with the IP address of the instance (figure 5.16).



Manage Floating IP Associations

IP Address * 192.168.100.10

Port to be associated * fredcirros: 192.168.5.11

Select the IP address you wish to associate with the selected instance or port.

Figure 5.16: Associating A Floating IP Address To An Instance

After association, the instance is pingable from the external network of the head node.

Example

```
[fred@bright73 ]$ ping -c1 192.168.100.10
PING 192.168.100.10 (192.168.100.10) 56(84) bytes of data.
64 bytes from 192.168.100.10: icmp_seq=1 ttl=63 time=1.54 ms

--- 192.168.100.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.544/1.544/1.544/0.000 ms
```

If SSH is allowed in the security group rules instead of ICMP, then fred can run ssh and log into the instance, using the default username/password cirros/cubswin:)

Example

```
[fred@bright73 ~]$ ssh cirros@192.168.100.10
cirros@192.168.100.10's password:
$
```

Setting up SSH keys: Setting up SSH key pairs for a user fred allows a login to be done using key authentication instead of passwords. The standard OpenStack way of setting up key pairs is to either import an existing public key, or to generate a new public and private key. This can be carried out from the Compute resource in the navigation window, then selecting the Access & Security page. Within the Key Pairs subtab there are the Import Key Pair button and the Create Key Pair button.

- **importing a key option:** For example, user fred created in Bright Cluster Manager as in this chapter has his public key in /home/fred/.ssh/id_dsa.pub on the head node. Pasting the text of the key into the import dialog, and then saving it, means that the user fred can now login as the user cirros without being prompted for a password from the head node. This is true for images that are cloud instances, of which the cirros instance is an example.

- **creating a key pair option:** Here a pair of keys is generated for a user. A PEM container file with just the private key <*PEM file*>, is made available for download to the user, and should be placed in a directory accessible to the user, on any host machine that is to be used to access the instance. The corresponding public key is stored in Keystone, and the private key discarded by the generating machine. The downloaded private key should be stored where it can be accessed by `ssh`, and should be kept read and write only. If its permissions have changed, then running `chmod 600` <*PEM file*> on it will make it compliant. The user can then login to the instance using, for example, `ssh -i` <*PEM file*> `cirros@192.168.100.10`, without being prompted for a password.

The `openstack keypair` options are the CLI API equivalent for the preceding Horizon operations.

Setting up SSH key pairs in this way relies on a properly functioning `cloud-init`. `cloud-init` is a set of initialization utilities that is part of the image available for the VMs that run under OpenStack (section 5.2.1). It is `cloud-init` that gets the VMs contact the OpenStack metadata server to pick up the public key and place it in the proper location on the VMs.

5.3 Running A Bright-managed Instance

A Bright-managed instance is a special case of the user instance in section 5.2. A Bright-managed instance is a virtual machine that is treated very similarly to a regular node by Bright Cluster Manager, and runs by default as a *vnode*. For example, it runs with the default names of `vnode001`, `vnode002...` rather than a `node001`, `node002` and so on. The default number of `vnodes` that is set, if Bright-managed instances are enabled, is 5, altough this number can be altered during OpenStack installation. The number of vnodes can be modified after installation in several ways, including:

- by adding a vnode as a node of type `virtualnode`
- by cloning an existing vnode and modifying it if needed
- by running the `Create Nodes` wizard in the `Virtual Nodes` tabbed pane. This is accessible from the `OpenStack` resource.

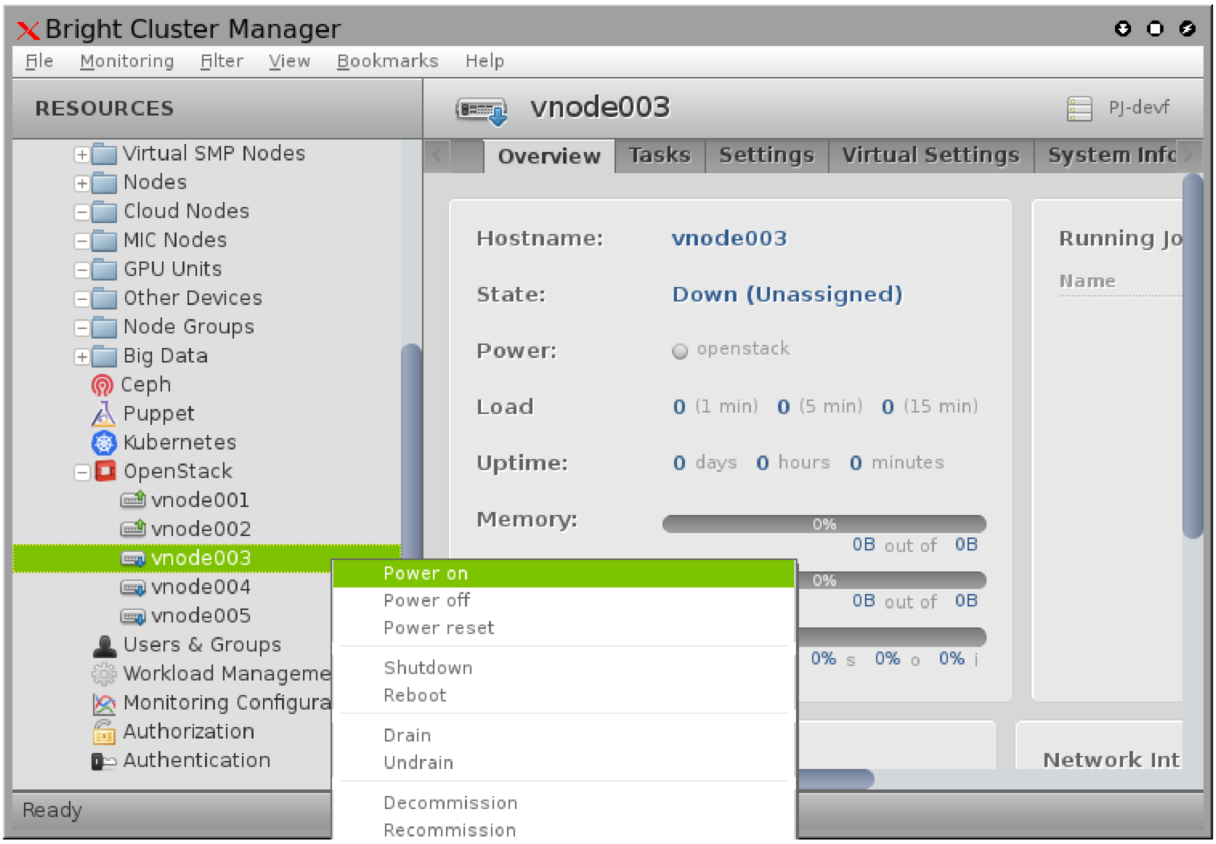
Since Bright Cluster Manager is integrated tightly with vnodes, getting a Bright-managed instance running is much easier than the procedure for user instances described earlier in sections 5.1 and 5.2. It is also a cluster administrator that typically creates Bright-managed instances, which run under the `bright` project, whereas it is end users that typically create regular VM instances, which typically run under a non-`bright` project name.

To get a default vnode up, it can be powered up from `cmsh`:

Example

```
[root@bright73 ~]# cmsh -c "device power on vnode001"
```

or it can be powered up from `cmgui` by right-clicking on the vnodes under the `OpenStack` resource, and selecting the `Power On` option (figure 5.17):



Hostname: vnode003
State: Down (Unassigned)
Power: openstack
Load 0 (1 min) 0 (5 min) 0 (15 min)
Uptime: 0 days 0 hours 0 minutes
Memory: 0% 0B out of 0B
0% 0B out of 0B
0% s 0% o 0% i

Figure 5.17: Powering On A Vnode Instance

Vnode boot can be followed at the console using the View Console button in the Tasks tab, just like with regular nodes. Indeed, most settings are like those for regular nodes.

One such exception is the vnode Virtual Settings tab, that is next to the vnode Settings tab. The Virtual settings allows, among others, a Flavor to be set.

The end user typically notices very little difference between vnodes and regular nodes.