Bright Cluster Manager 7.2

OpenStack Deployment Manual

Revision: f6f97b6

Date: Fri Sep 13 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 7.2.

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 7.2 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 *Hadoop Deployment Manual* describes how to deploy Hadoop with Bright Cluster Manager.
- The *UCS Deployment Manual* describes how to deploy the Cisco UCS server 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 goes via standard e-mail. Section 11.2 of the *Administrator Manual* has more details on working with support.

1

Introduction

OpenStack is an open source implementation of cloud services. It is currently (2015) undergoing rapid development, and its roadmap is promising.

A relatively stable implementation of OpenStack, based on the OpenStack Liberty release (`https://www.openstack.org/software/liberty/`) is integrated into the Bright Cluster Manager 7.2 for OpenStack edition. It is supported for versions of RHEL7 onwards.

By relatively stable it is meant that OpenStack itself is usable and stable for regular use in common configurations, but not quite production-ready when carrying out some less common configuration changes. 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 to become convenient and stable for regular use. This process benefits all participants in the OpenStack software ecosystem.

OpenStack consists of subsystems, developed as 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. As far as the user is concerned the 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 because it is not yet production-ready.

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.

2

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 3) 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 demonstration purposes.

However, if running OpenStack with Bright Cluster Manager, then a standard demonstration configuration can be considered to be a head node, a network node, and several regular nodes. Regular nodes are commonly called compute nodes, while the network node is typically a re-purposed regular node. For such a standard configuration, recommended hardware specifications for useful demonstration purposes are:

- A head node with 8GB RAM, 4 cores and two network interfaces. In a standard configuration this is typically configured as a controller, which means it runs RabbitMQ services.
- A network node with 2GB RAM and two network interfaces
- Three regular nodes with 2GB RAM per core. Each regular node has a network interface.
  - In larger clusters, it may be a good idea to offload the controller type from the head node to a regular node. If a regular node is configured as a controller, it must have at least 6GB RAM.

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 run into 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.

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 2.1). This is the recommended installation method.
- Using the text-based `cm-openstack-setup` utility (section 2.2). 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.

2.1 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 connected to the distribution repositories.

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 2.1.11). 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 2.1.11, figure 2.12).

  - 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 2.2) 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 2.1). 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.

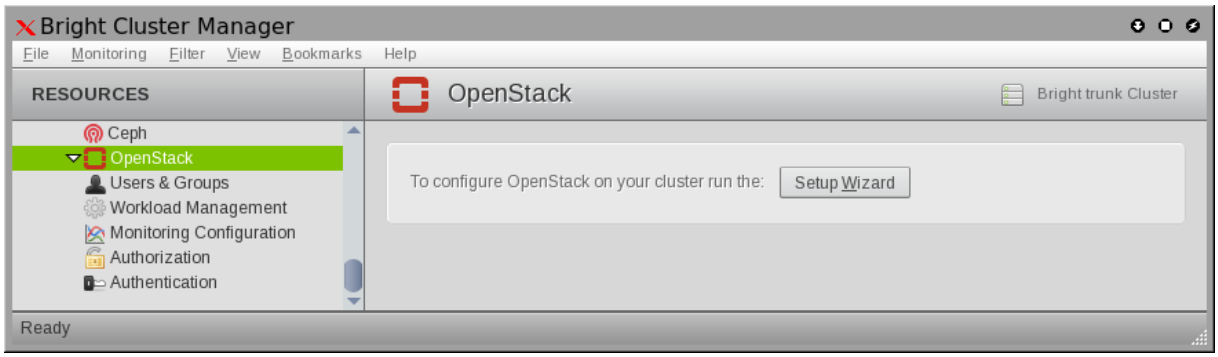

Figure 2.1: The `Setup Wizard` Button In `cmgui`'s OpenStack Resource

The wizard runs through the screens in sections 2.1.1-2.1.15, described next.

2.1.1 OpenStack Setup Wizard Overview

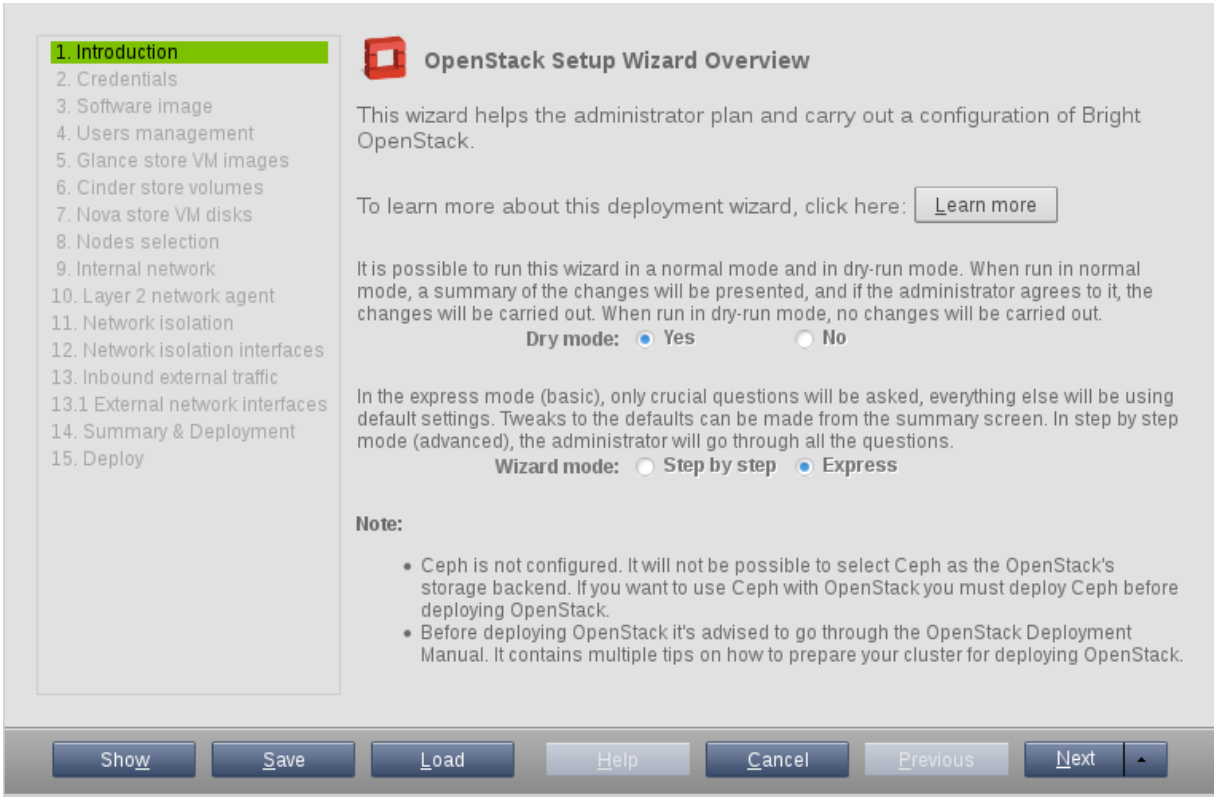

Figure 2.2: OpenStack Setup Wizard Overview Screen

The main overview screen (figure 2.2) 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 the following questions:

- Should a dry-run be done?
  - In a dry-run, the wizard pretends to carry out the installation, but the changes are not really implemented. This is useful for getting familiar with options and their possible consequences. A dry run is enabled as the default.
- Should the wizard 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, 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 2.1.15).

- 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 2.1.2-2.1.15.

2.1.2 OpenStack admin User Screen

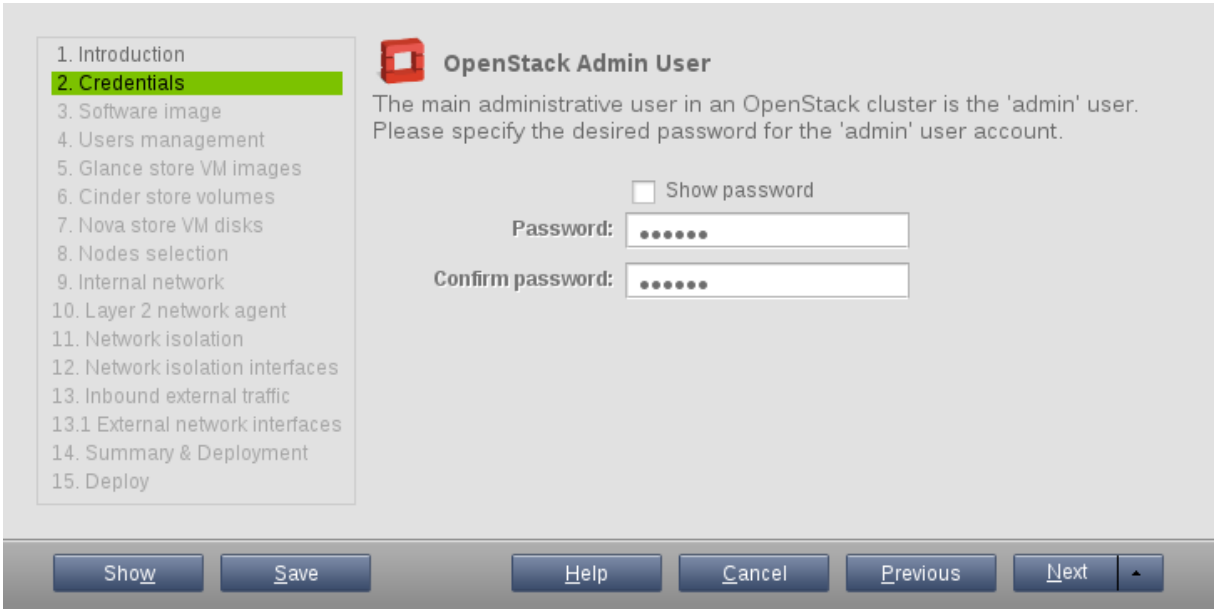

Figure 2.3: OpenStack admin User Screen

The OpenStack credentials screen (figure 2.3) 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.

2.1.3 OpenStack Software Image Selection

Figure 2.4: OpenStack Software Image Selection Screen

The OpenStack software image selection screen (figure 2.4) 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.

2.1.4 User Management

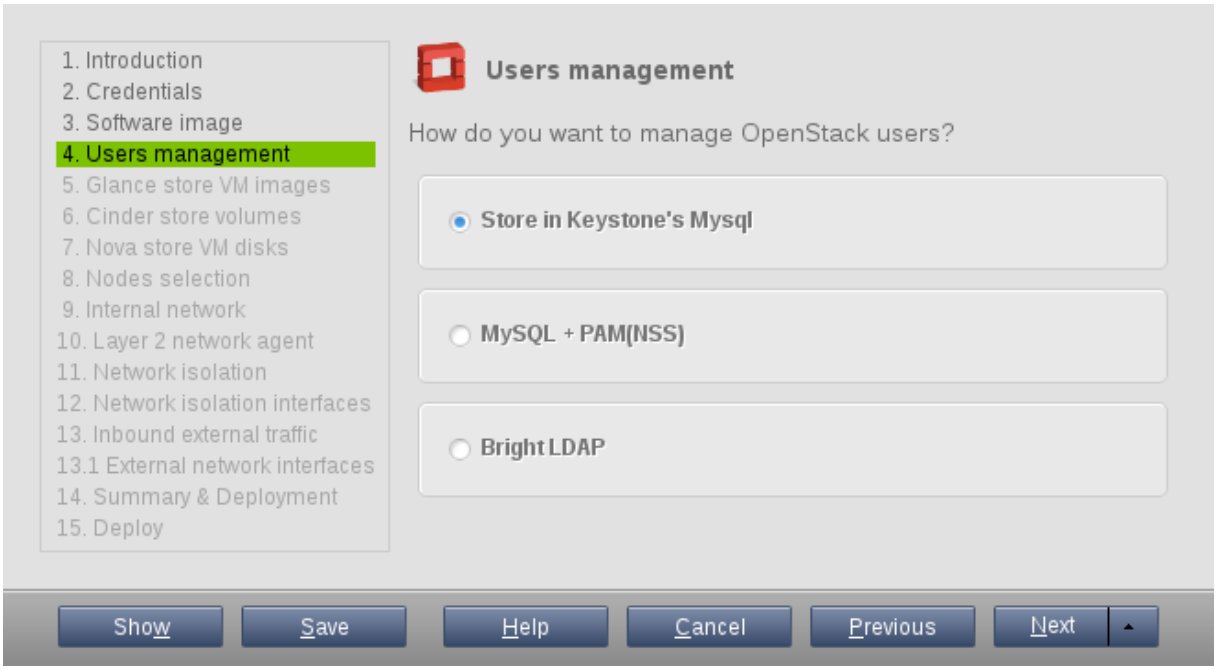

Figure 2.5: OpenStack User Management Screen

The User Management screen (figure 2.5) 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.
- Store in a MySQL database managed by Keystone, and use PAM (NSS). Bright Cluster Manager users are accessible via Keystone only if they have have been initialized, with Keystone roles assigned to them for a project. Only such users can access OpenStack. Initialization can be set with an initialization script (section 4.1.2)
- Use Bright Cluster Manager LDAP authentication. As in the preceding case, Bright Cluster Manager users are accessible via Keystone only if they have have been initialized, with Keystone roles assigned to them for a project. Only such users can access OpenStack. Initialization can be set with an initialization script.

Keystone can also be set to authenticate directly with an external LDAP or AD server.

2.1.5 Glance VM Image Storage

Figure 2.6: OpenStack Glance VM Image Storage Screen

The `Glance VM Image Storage` screen (figure 2.6) 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

2.1.6 Cinder Volume Storage

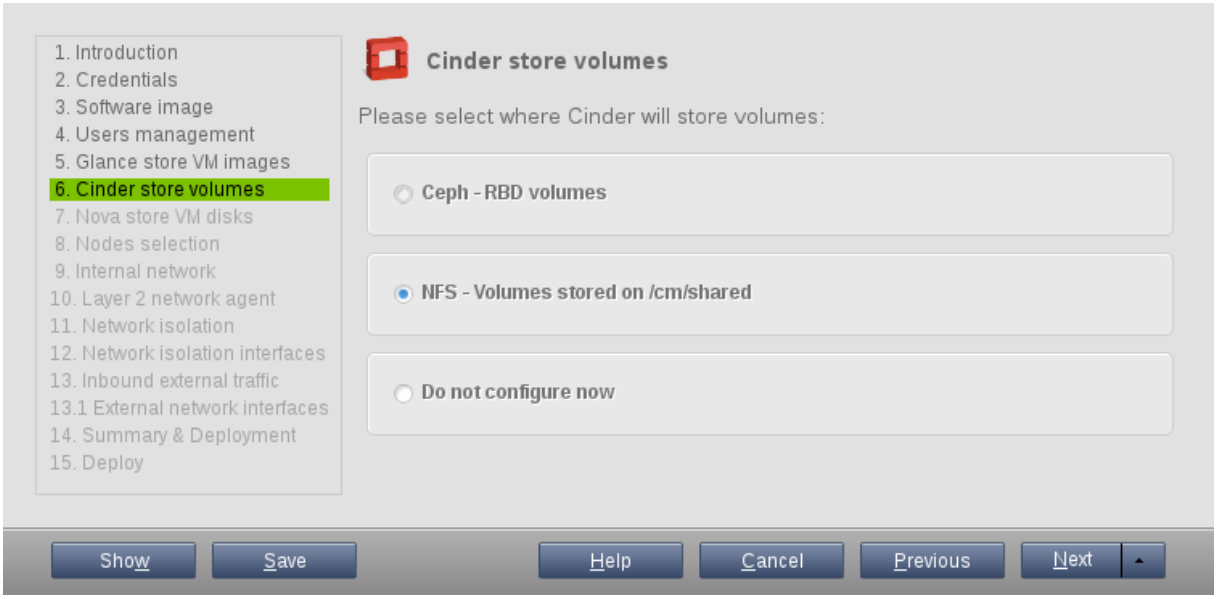

Figure 2.7: OpenStack Cinder Volume Storage Screen

The `Cinder Volume Storage` screen (figure 2.7) 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`

2.1.7 Nova VM Disks Storage

Figure 2.8: OpenStack Nova VM Disks Storage Screen

The `Nova VM Disks Storage` screen (figure 2.8) 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.

2.1.8 OpenStack Nodes Selection

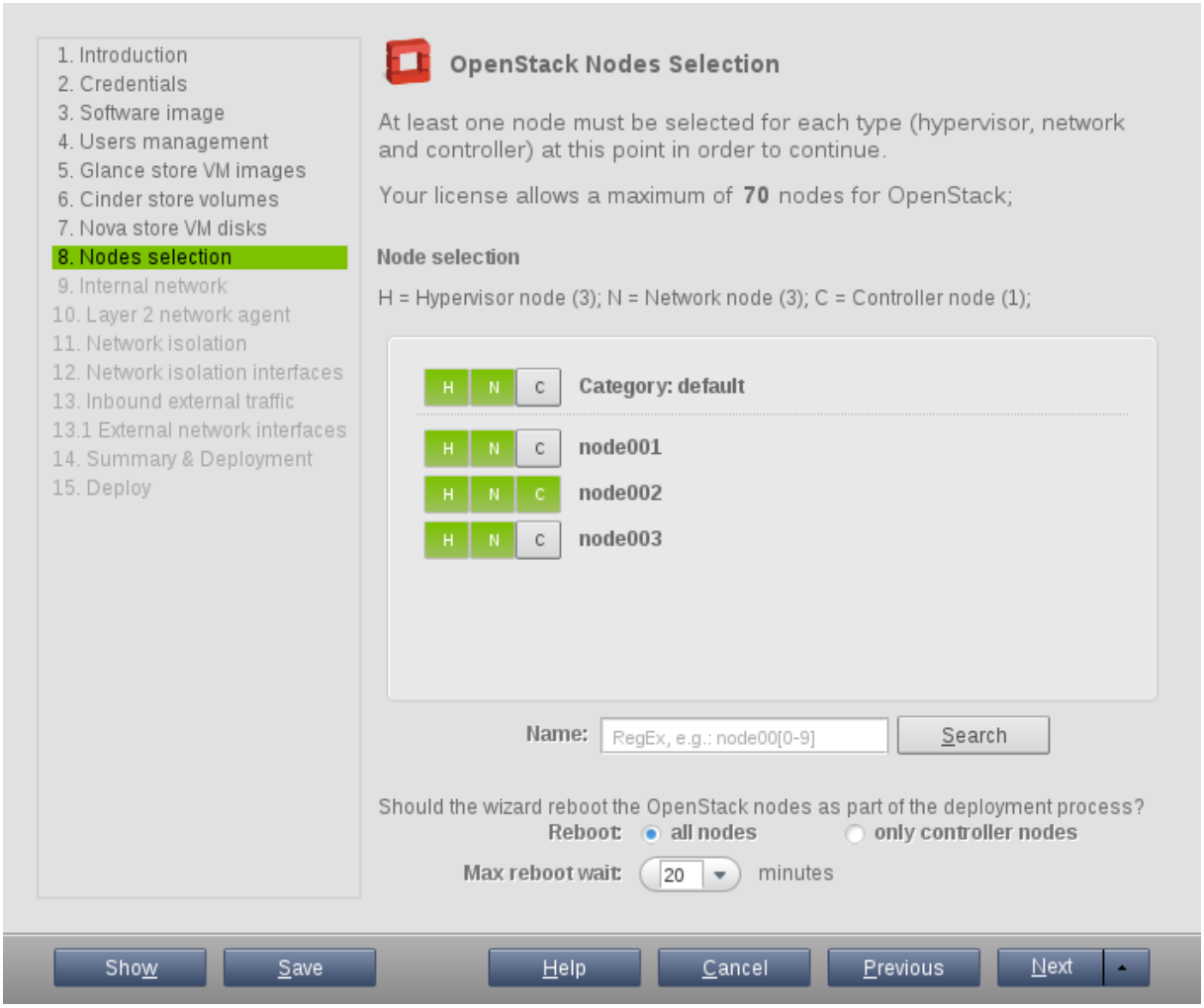

Figure 2.9: OpenStack Nodes Selection

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.
- 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 2.9 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.

2.1.9 OpenStack Internal Network Selection Screen

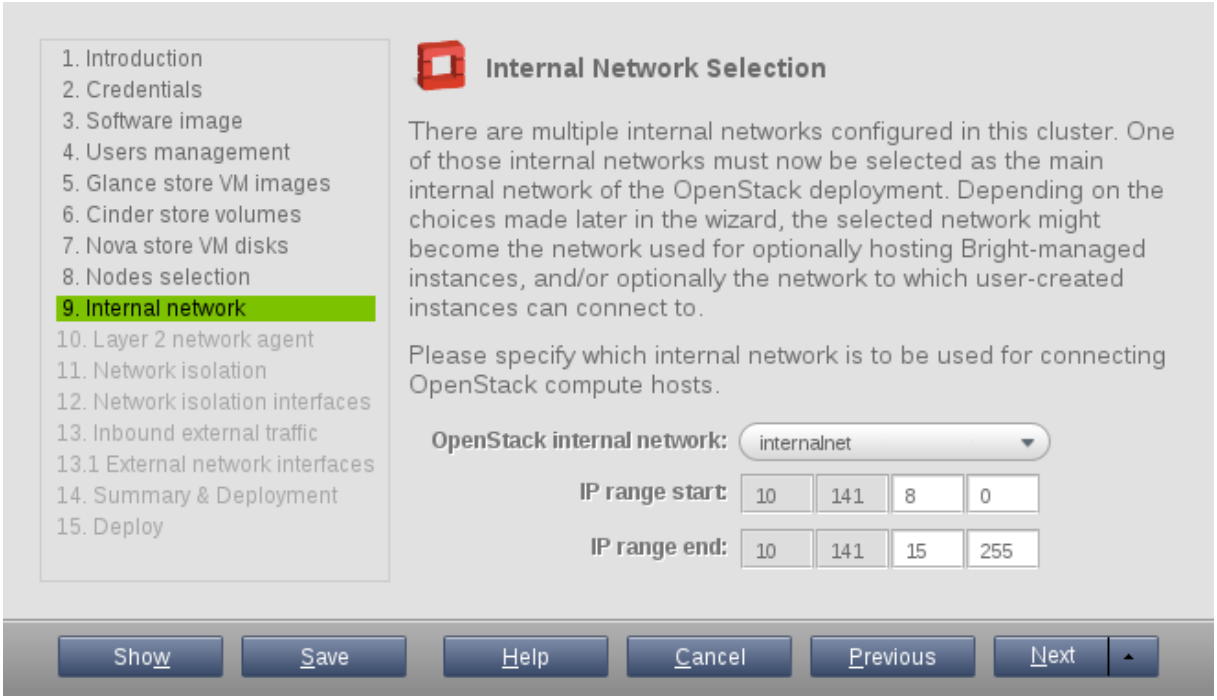

Figure 2.10: OpenStack Internal Network Selection

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.

2.1.10 OpenStack Layer 2 Network Agent Selection Screen

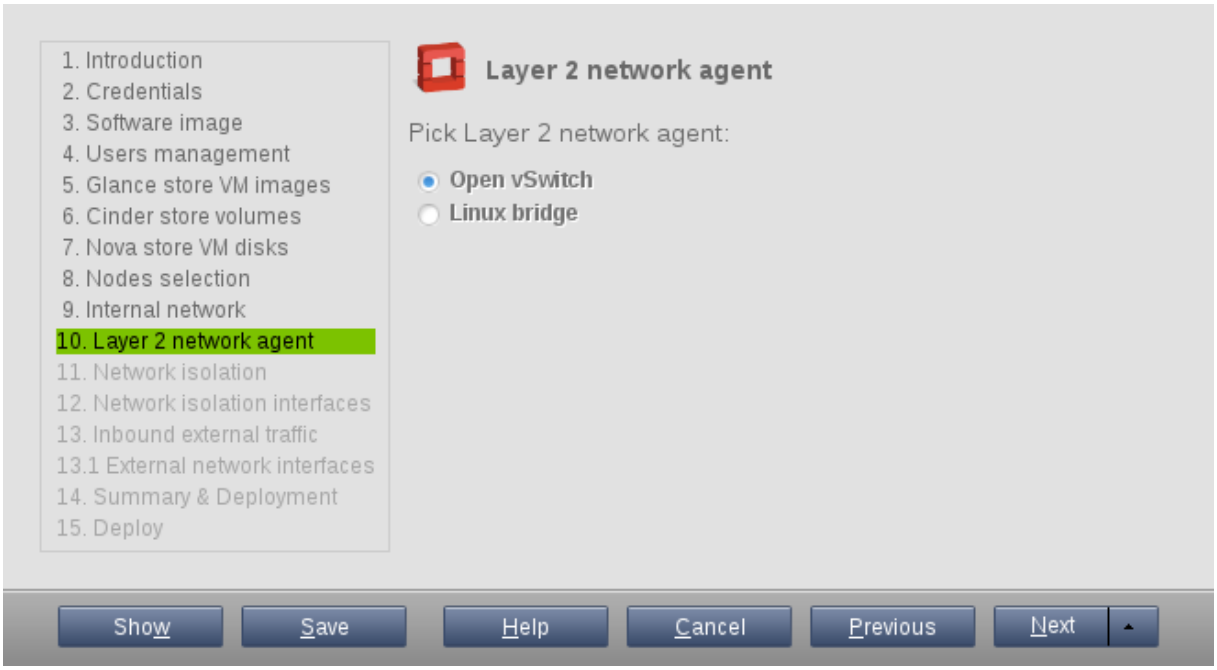

Figure 2.11: OpenStack Internal Network Selection

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

- Linux bridge: simpler, but not as versatile.
- 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).

2.1.11 OpenStack Network Isolation And VLAN/VXLAN Configuration

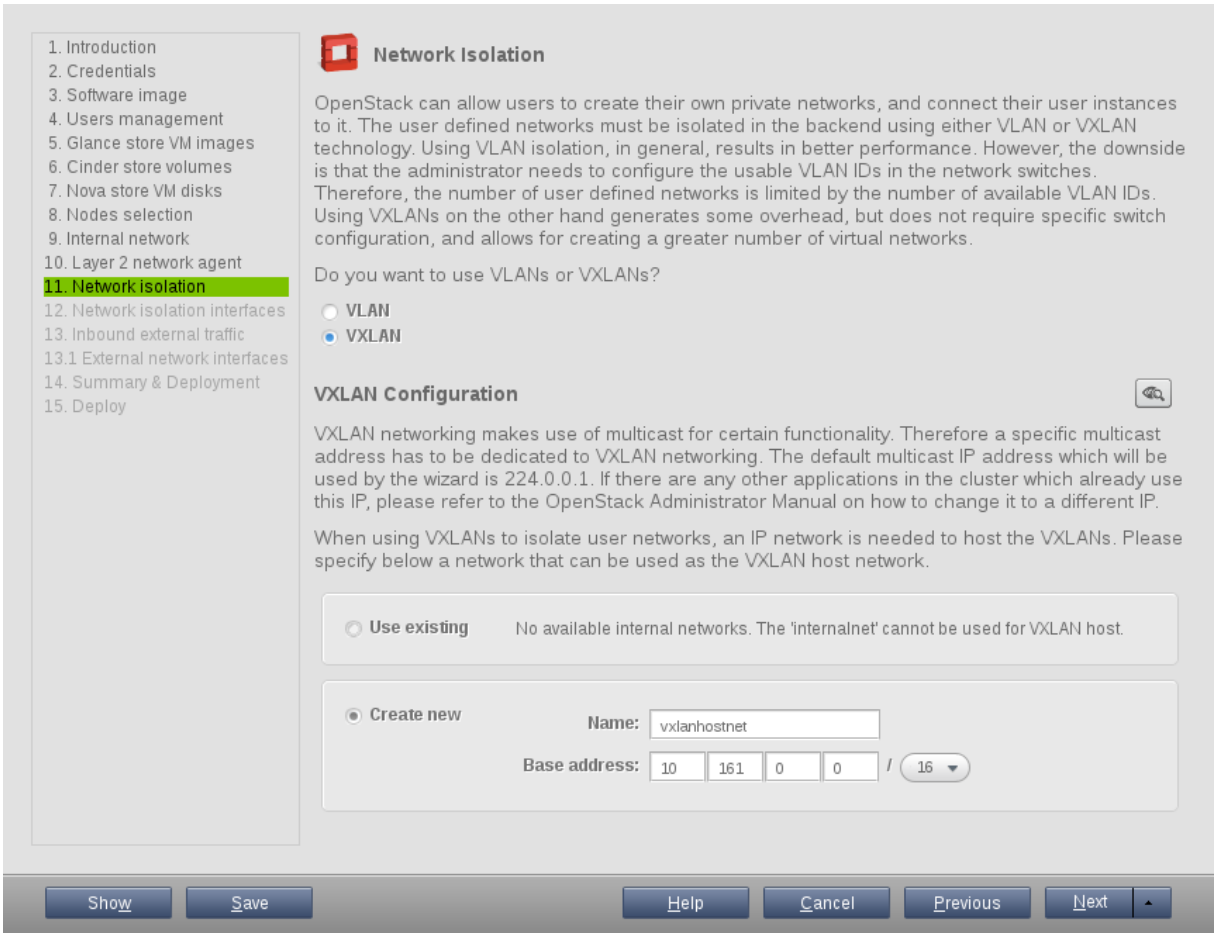

Figure 2.12: OpenStack Network Isolation And VXLAN Configuration Screen

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 580 of the *Administrator Manual*).

2.1.12 OpenStack Network Isolation interface For Network And Hypervisor Nodes

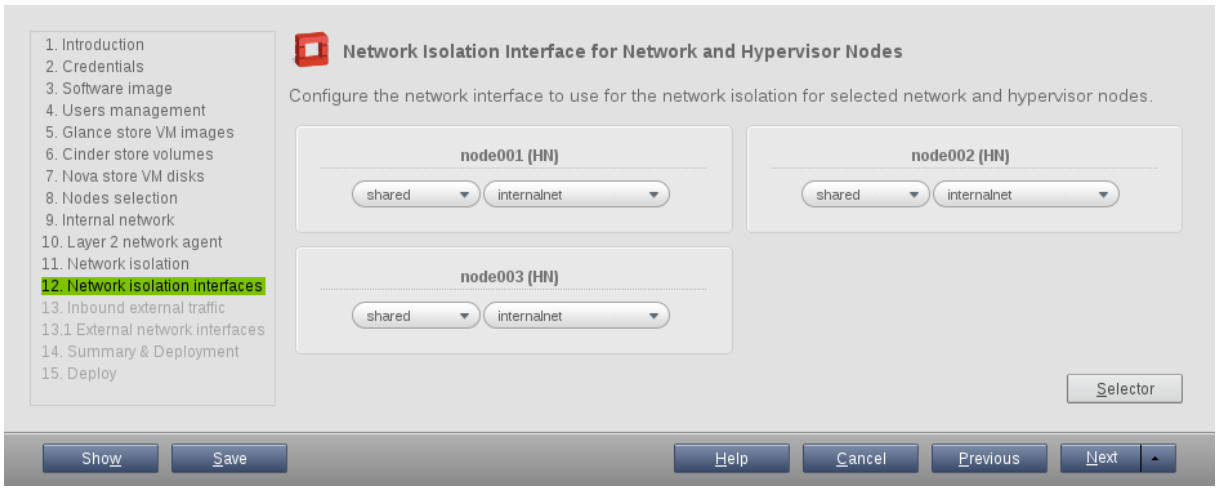

Figure 2.13: OpenStack Network Isolation interface For Network And Hypervisor Nodes Screen

The `Network Isolation interface For Network And Hypervisor Nodes` screen (figure 2.13) 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.

2.1.13 OpenStack Inbound External Traffic

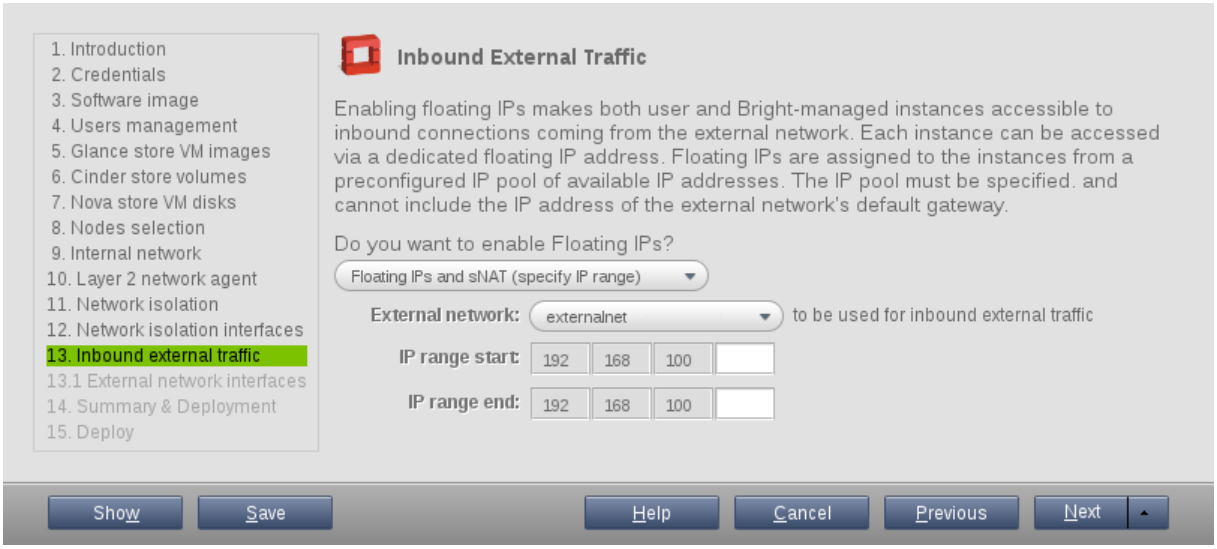

Figure 2.14: OpenStack Inbound External Traffic Screen

The `OpenStack Inbound External Traffic` screen (figure 2.14) 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.

2.1.14 OpenStack External Network Interface For Network Node

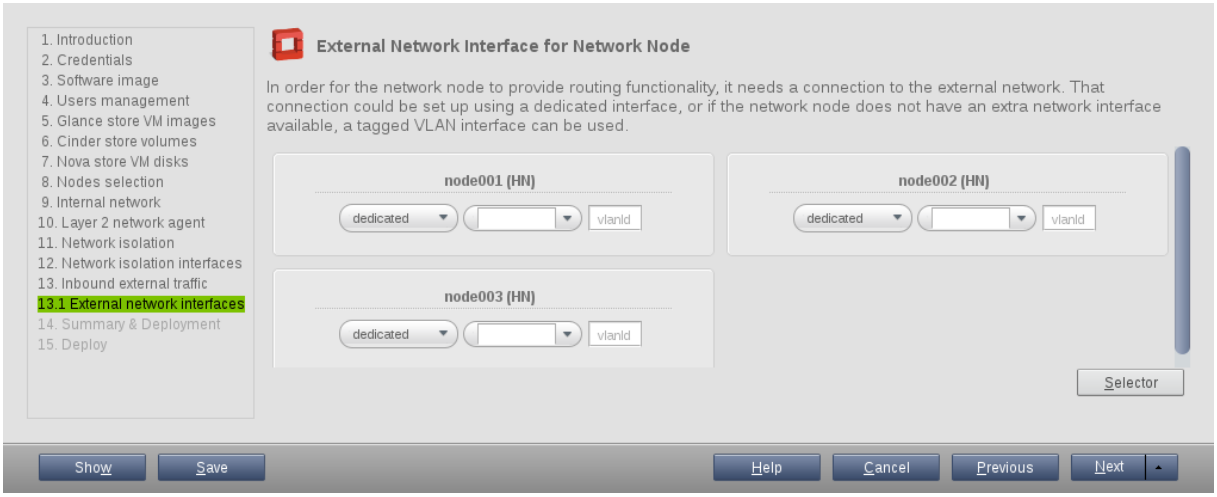

Figure 2.15: OpenStack External Network Interface For Network Node Screen

The `OpenStack External Network Interface For Network Node` screen (figure 2.15) 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.

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

2.1.15 Summary

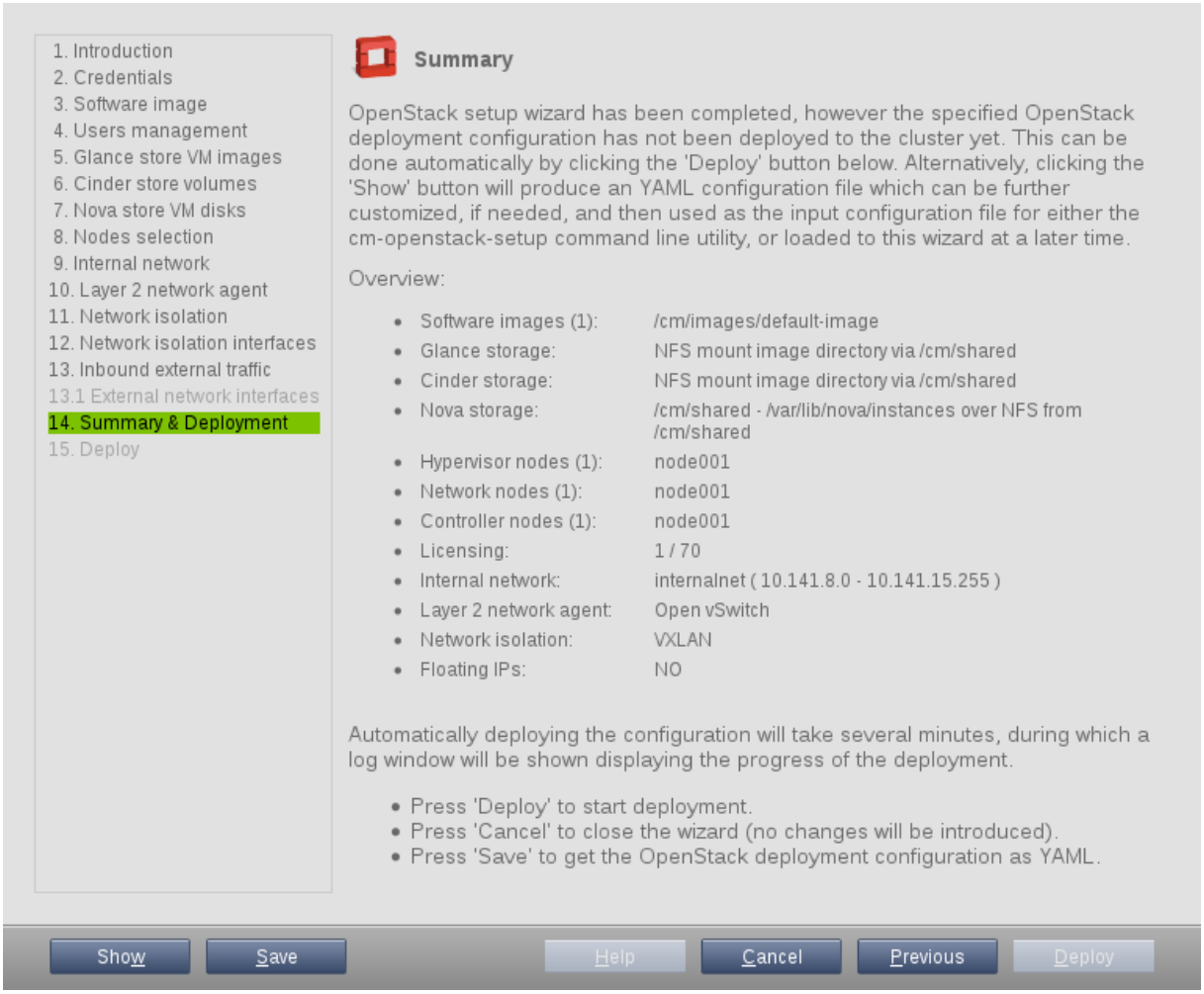

Figure 2.16: Summary Screen

Viewing And Saving The Configuration

The summary screen (figure 2.16) 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 2.17.

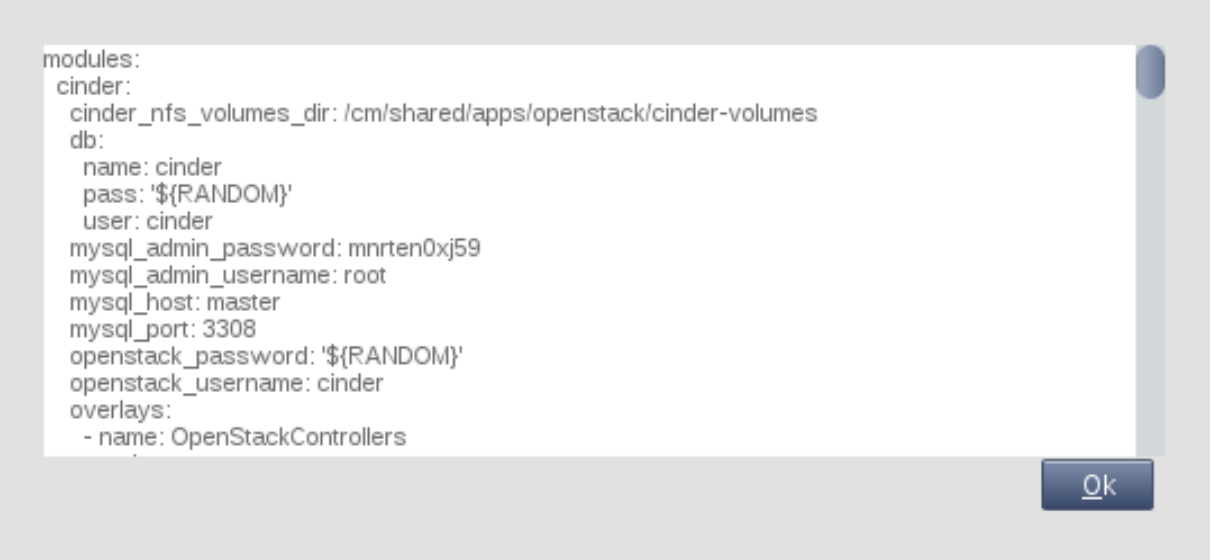

Figure 2.17: OpenStack Configuration Screen

The configuration can be saved with the `Save` button of figure 2.16.

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 2.2), if run as:

  ```
  [root@bright72~]# 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 first screen (figure 2.2). After loading the configuration, a `Deploy` button appears.

Clicking the `Deploy` button that appears in figure 2.2 after loading the YAML file, or clicking the `Deploy` button of figure 2.16, 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 2.2). Some log excerpts from the script are displayed within a `Deployment Progress` window (figure 2.18).

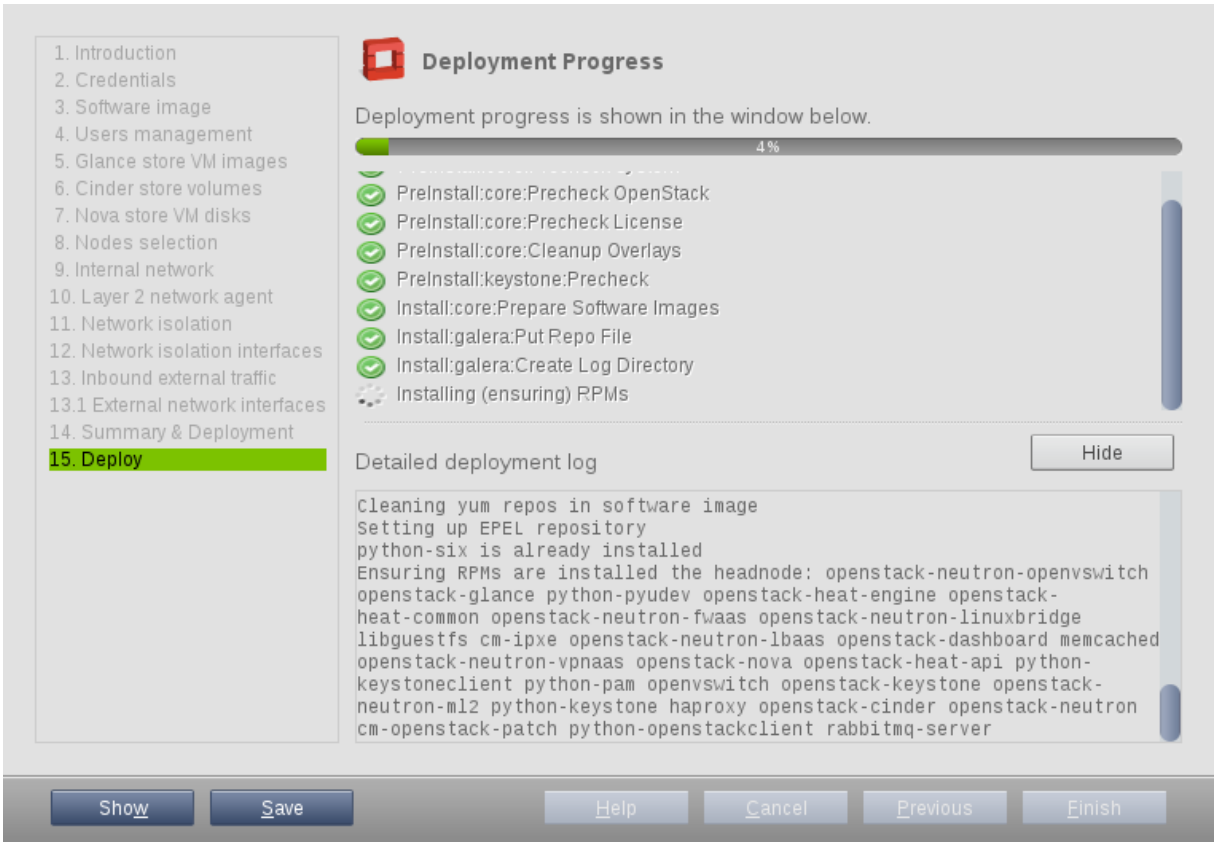

Figure 2.18: OpenStack Deployment Progress Screen

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.

2.2 Installation Of OpenStack From The Shell

The `cmgui` OpenStack installation (section 2.1) 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 2.2.1. 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`>.

2.2.1 Start Screen

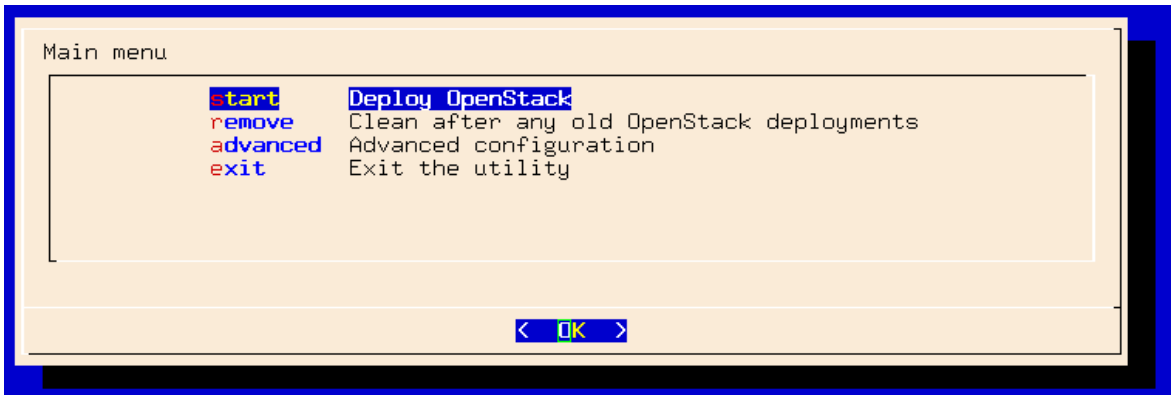

Figure 2.19: Start Screen

The start screen (figure 2.19) lets the administrator:

- deploy Bright Cluster Manager OpenStack.
- remove Bright Cluster Manager's OpenStack if it is already on the cluster.
- carry out some more advanced configuration tasks
- 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@bright72 ~]# cmsh
[bright72]% openstack
[bright72->openstack[default]]% set preventremoval no; commit; quit
[root@bright72 ~]# cm-openstack-setup --remove
Please wait...
Connecting to CMDaemon
###### WARNING: Setup will attempt to remove the following objects:
...
```

2.2.2 Controller Node Selection

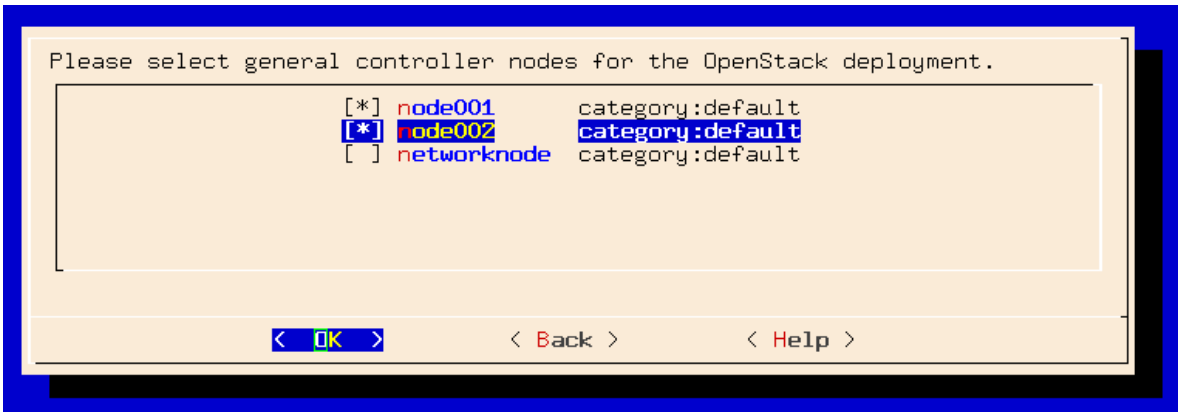

Figure 2.20: Controller Nodes Selection

The controller nodes selection screen (figure 2.20) 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 Liberty are
  - Nova (compute)
  - Neutron (networking)
  - Swift (object storage)
  - Cinder (block storage)
  - Keystone (identity)
  - Glance (image)

  Each controller node is required to have a minimum of 2 cores and 4096MB RAM.

2.2.3 Setting The Cloud `admin` Password

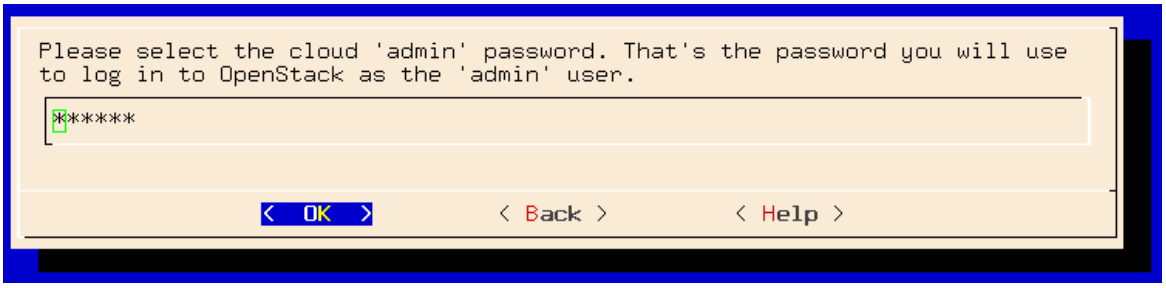

Figure 2.21: Cloud `admin` Password Screen

The OpenStack cloud `admin` password screen (figure 2.21) 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.

2.2.4 User Management Configuration Of OpenStack Users

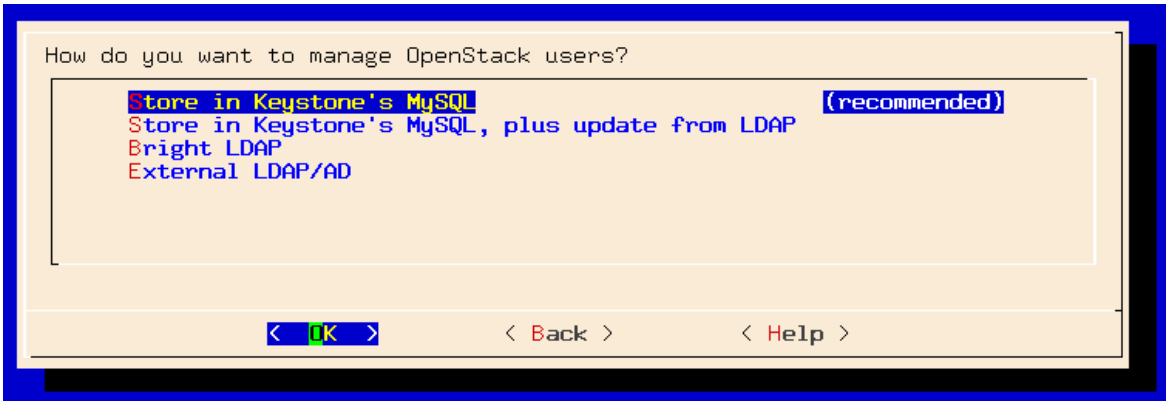

Figure 2.22: User Management Configuration Of OpenStack Users Screen

The user management configuration of OpenStack users screen (figure 2.22) allows the administrator to choose how OpenStack users are to be managed. Options are:

- Managing via Keystone MySQL
- Managing via Keystone MySQL, and updating from LDAP
- Using LDAP provided by Bright Cluster Manager
- Using an external LDAP or Active Directory.

2.2.5 Informative Text Prior To Deployment

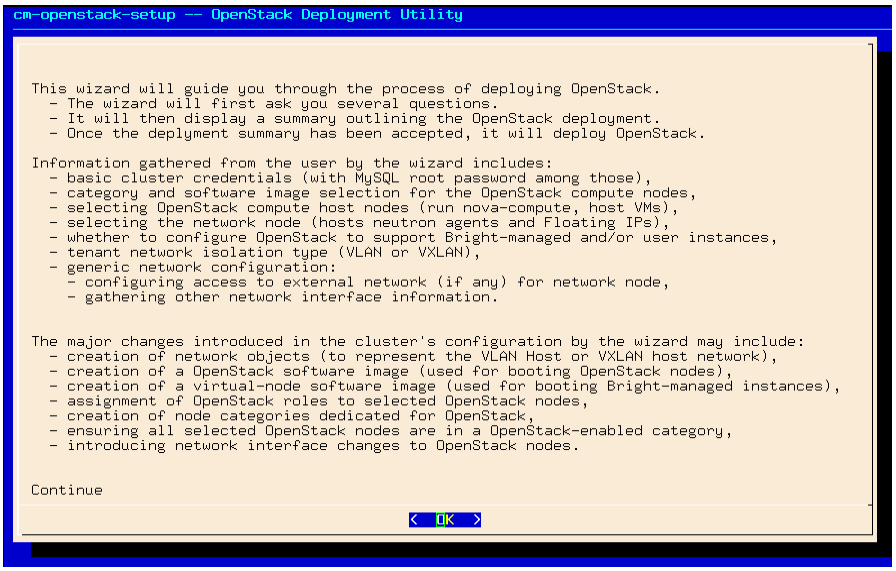

Figure 2.23: Informative Text Prior To Deployment

If deployment is selected in the preceding screen, an informative text screen (figure 2.23) gives a summary of what the script does.

2.2.6 Pre-Setup Suggestions

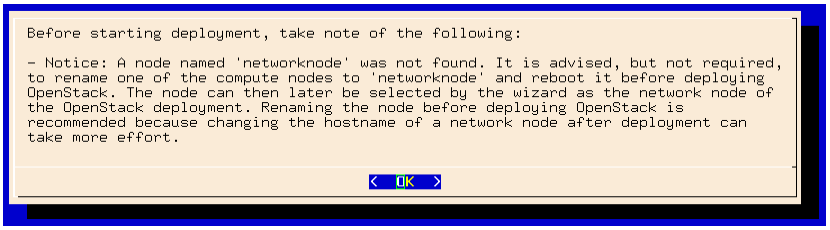

Figure 2.24: Pre-Setup Suggestions

The pre-setup suggestions screen (figure 2.24) suggests changes to be done before going on.

2.2.7 MySQL `root` And OpenStack `admin` Passwords

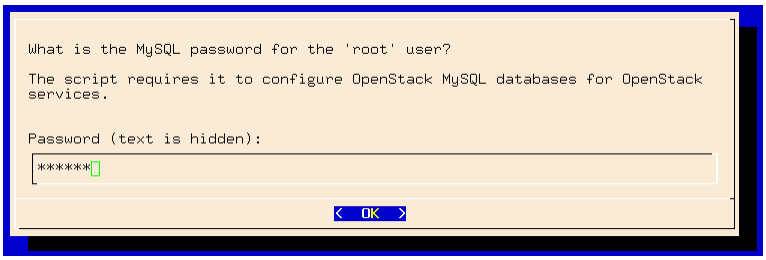

Figure 2.25: MySQL `root` Password Screen

The MySQL `root` password screen (figure 2.25) prompts for the existing root password to `MySQL` to be entered.

2.2.8 Reboot After Configuration

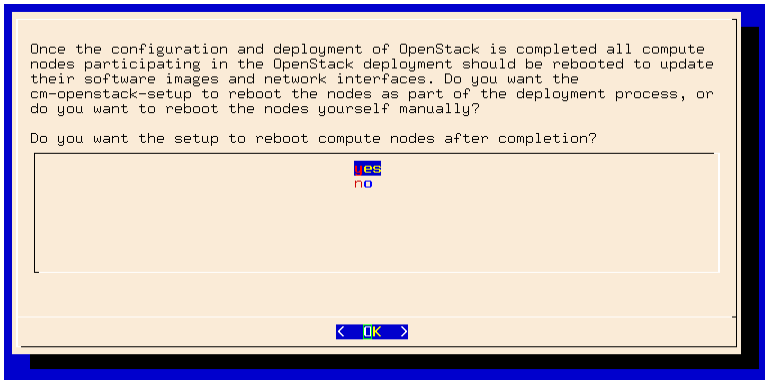

Figure 2.26: Reboot After Configuration Screen

A screen is shown asking if the compute host nodes, that is, the nodes used to host the virtual nodes, should be re-installed after configuration (figure 2.26). A re-install is usually best.

2.2.9 Ceph Options

This section (2.2.9) covers the Ncurses `cm-openstack-setup` wizard configuration of Ceph options.

Glance Image Storage With Ceph

Figure 2.27: Image Storage With Ceph

Ceph can be set for storing virtual machine images, instead of the OpenStack reference Glance, using the Ceph image storage screen (figure 2.27).

Block Storage With Ceph

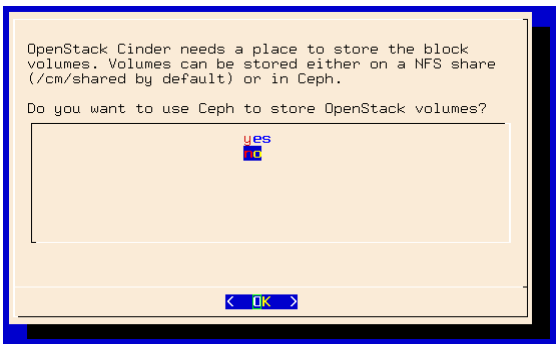

Figure 2.28: Block Storage With Ceph

Ceph can be set for handling block volume storage read and writes, instead of the OpenStack reference Cinder, by using the Ceph for OpenStack volumes screen (figure 2.28).

Root And Ephemeral Device Storage With Ceph

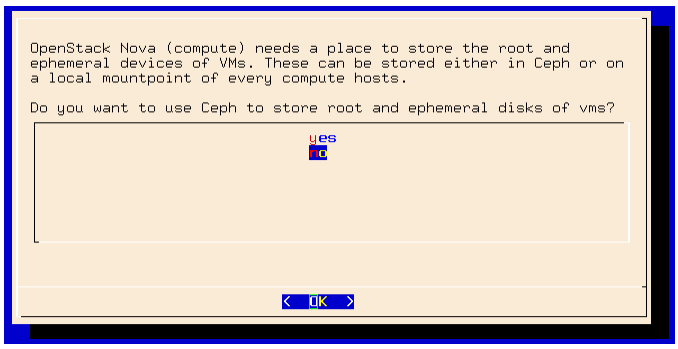

Figure 2.29: Root And Ephemeral Device Storage With Ceph

Data storage with Ceph can be enabled by the administrator by using the Ceph for OpenStack root and ephemeral device storage screen (figure 2.29).

Ceph Object Gateway (Ceph RADOS Gateway)

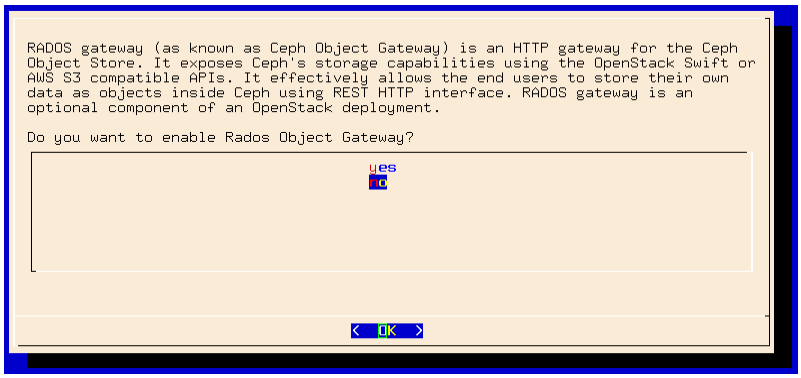

Figure 2.30: Root And Ephemeral Device Storage With Ceph

The Ceph RADOS gateway screen (figure 2.30) lets the administrator set the Ceph RADOS gateway service to run when deployment completes.

2.2.10 Internal Network To Be Used For OpenStack

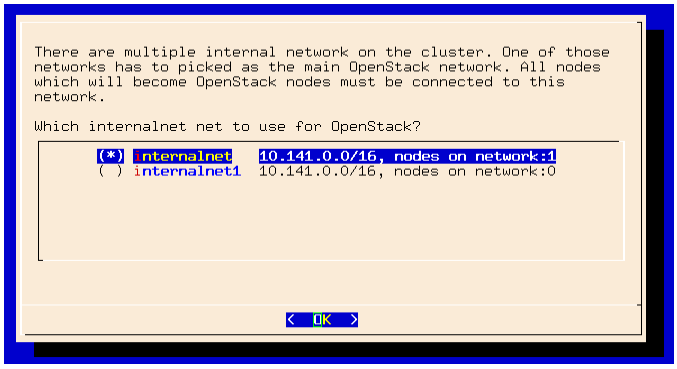

Figure 2.31: Internal Network To Be Used For OpenStack

If there are multiple internal networks, then the internal network selection screen (figure 2.31) lets the administrator choose which of them is to be used as the internal network to which the OpenStack nodes are to be connected.

2.2.11 User Instances

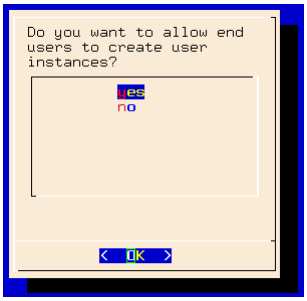

Figure 2.32: User Instances

The user instances screen (figure 2.32) lets the administrator decide if end users are to be allowed to create user instances.

2.2.12 Virtual Instance Access To Internal Network

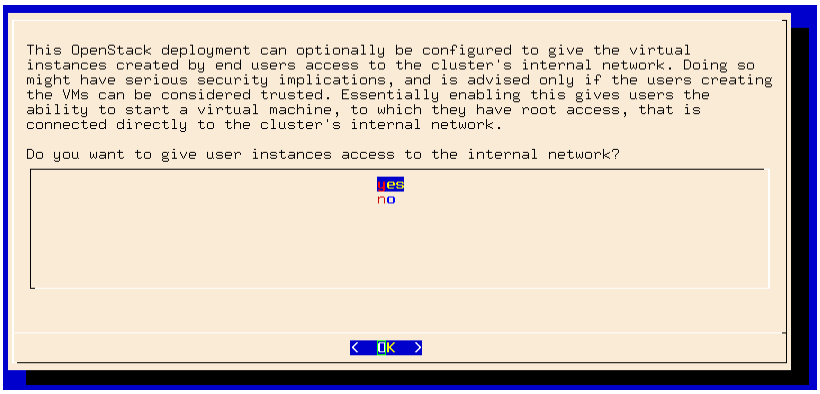

Figure 2.33: Virtual Instance Access To Internal Network

The screen in figure 2.33 lets the administrator allow virtual instances access to the internal network. This should only be allowed if the users creating the instances are trusted. This is because the creator of the instance has root access to the instance, which is in turn connected directly to the internal network of the cluster, which means all the packets in that network can be read by the user.

2.2.13 Network Isolation Type

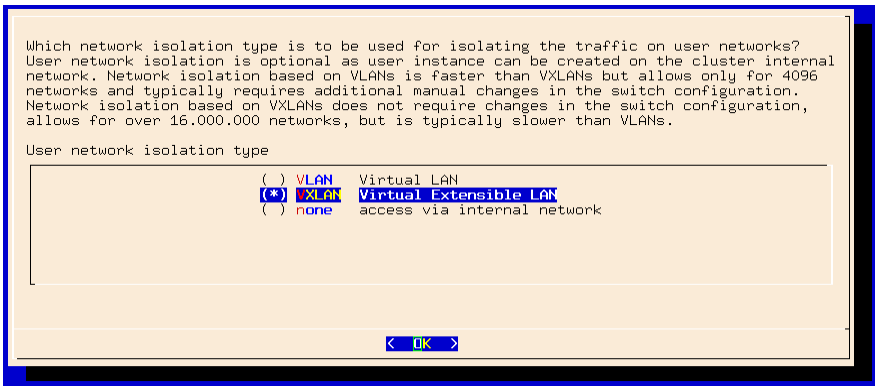

Figure 2.34: Network Isolation Type

The network isolation type screen (figure 2.34) allows the administrator to choose what kind of network isolation type, if any, should be set for the user networks.

2.2.14 Choosing The Network That Hosts The User Networks

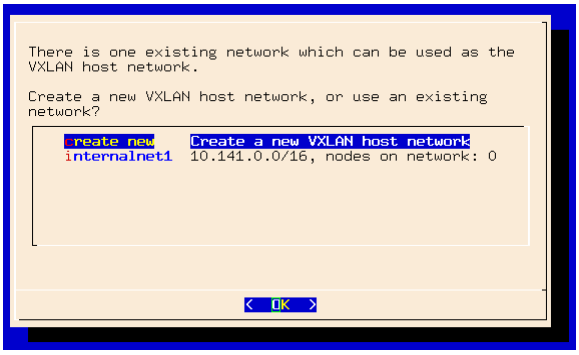

Figure 2.35: User Networks Hosting

If the user networks has their type (VXLAN, VLAN, or no virtual LAN) chosen in section 2.2.13, then a screen similar to figure 2.35 is displayed. This allows one network to be set as the host for the user networks.

If there are one or more possible networks already available for hosting the user networks, then one of them can be selected. Alternatively, a completely new network can be created to host them.

2.2.15 Setting The Name Of The Hosting Network For User Networks

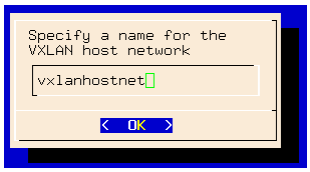

Figure 2.36: Setting The Name Of The Network For User Networks

If a network to host the user networks is chosen in section 2.2.14, then a screen similar to figure 2.36 is displayed. This lets the administrator set the name of the hosting network for user networks.

2.2.16 Setting The Base Address Of The Hosting Network For User Networks

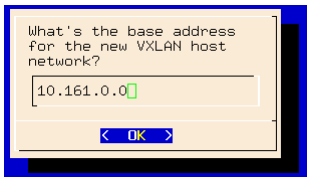

Figure 2.37: Setting The Base Address Of The Network For User Networks

If the network name for the network that hosts the user networks is chosen in section 2.2.15, then a screen similar to figure 2.37 is displayed. This lets the administrator set the base address of the hosting network for user networks.

2.2.17 Setting The Number Of Netmask Bits Of The Hosting Network For User Networks

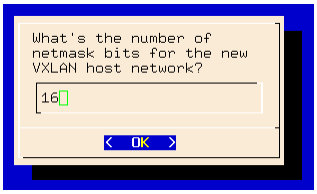

Figure 2.38: Setting The Number Of Netmask Bits Of The Network For User Networks

If the base address for the network that hosts the user networks is set in section 2.2.16, then a screen, similar to figure 2.38 is displayed. This lets the administrator set the number of netmask bits of the hosting network for user networks.

2.2.18 Enabling Support For Bright-managed Instances

Figure 2.39: Enabling Support For OpenStack Instances Under Bright Cluster Manager

There are two kinds of OpenStack instances, also known as virtual nodes, that can run on the cluster. These are called user instances and Bright-managed instances. The screen in figure 2.39 decides if Bright-managed instances are to be configured, and be ready to run. Bright-managed instances are actually a special case of user instances, just managed much more closely by Bright Cluster Manager.

Only if permission is set in the screen of section 2.2.12, can an end user access Bright-managed instances.

The screens from figure 2.40 to figure 2.42 are only shown if support for Bright-managed instances is enabled.

2.2.19 Starting IP Address For Bright-managed Instances

Figure 2.40: Starting IP Address For Bright-managed Instances

A starting IP address must be set for the Bright-managed instances (figure 2.40)

2.2.20 Ending IP Address For Bright-managed Instances

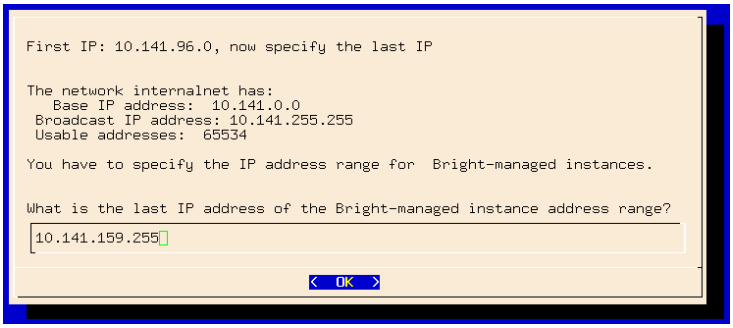

Figure 2.41: Ending IP Address For Bright-managed Instances

An ending IP address must be set for the Bright-managed instances (figure 2.41).

2.2.21 Number Of Virtual Nodes For Bright-managed Instances

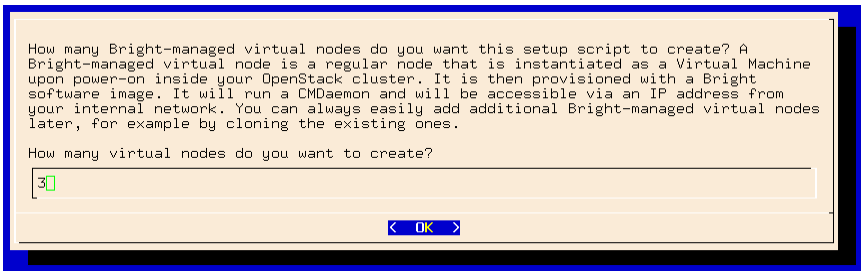

Figure 2.42: Number Of Virtual Nodes For Bright-managed Instances

The number of Bright-managed virtual machines can be set (figure 2.42). The suggested number of instances in the wizard conforms to the defaults that OpenStack sets. These defaults are based on an overcommit ratio of virtual CPU:real CPU of 16:1, and virtual RAM:real RAM of 1.5:1. The instance flavor chosen then determines the suggested number of instances.

2.2.22 DHCP And Static IP Addresses

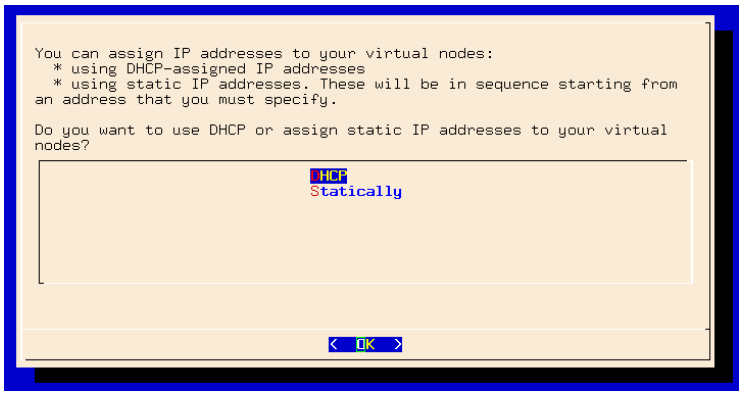

Figure 2.43: DHCP Or Static IP Address Selection

The instances can be configured to obtain their IP addresses either via DHCP, or via static address assignment (figure 2.44).

2.2.23 Floating IPs

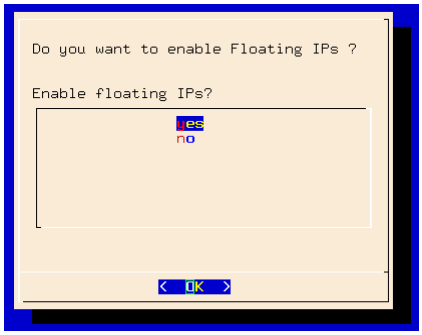

Figure 2.44: Floating IPs

The Floating IPs screen (figure 2.44) lets the administrator enable Floating IPs on the external network, so that instances can be accessed using these.

2.2.24 External Network Starting Floating IP

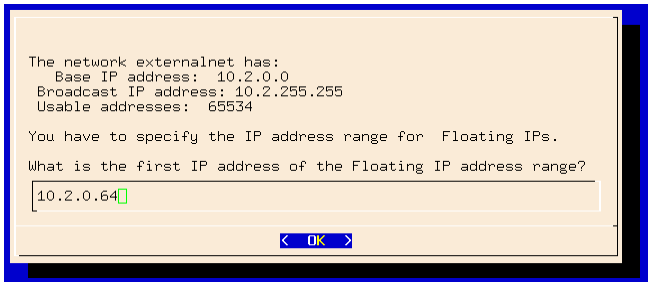

Figure 2.45: External Network: Starting Floating IP

A screen similar to figure 2.45 allows the administrator to specify the starting floating IP address on the external network.

2.2.25 External Network Ending Floating IP

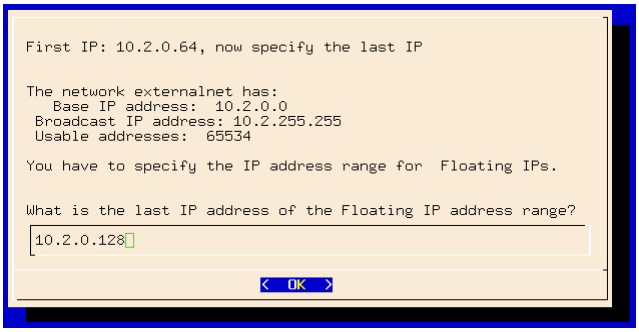

Figure 2.46: External Network: Ending Floating IP

A screen similar to figure 2.46 allows the administrator to specify the ending floating IP address on the external network.

2.2.26 VNC Proxy Hostname

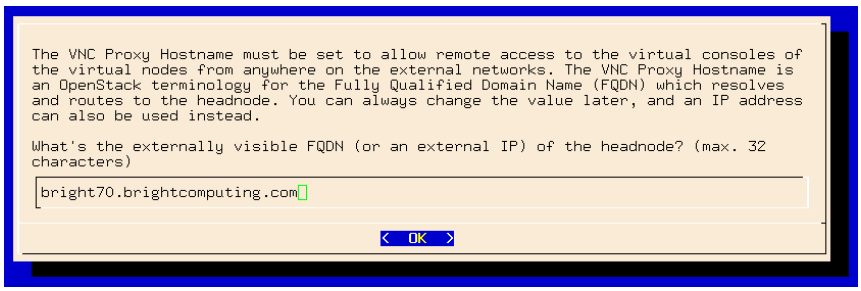

Figure 2.47: VNC Proxy Hostname

The VNC Proxy Hostname screen (figure 2.47) lets the administrator set the FQDN as seen from the external network. An IP address can be used instead of the FQDN.

2.2.27 Nova Compute Hosts

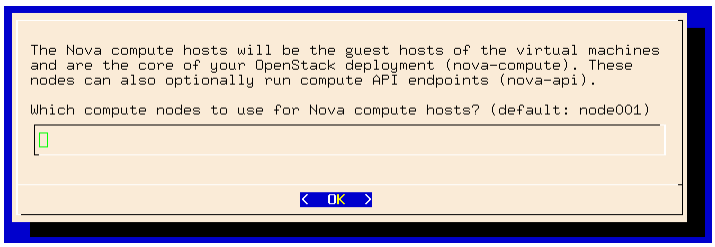

Figure 2.48: Nova Compute Hosts

The Nova compute hosts screen (figure 2.48) prompts the administrator to set the nodes to use as the hosts for the virtual machines.

2.2.28 Neutron Network Node

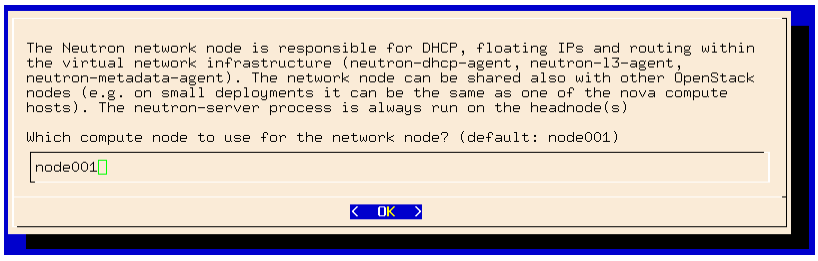

Figure 2.49: Neutron Network Node

The Neutron network node screen (figure 2.49) prompts the administrator to set the node to use for Neutron network node.

2.2.29 Pre-deployment Summary

Figure 2.50: Pre-deployment-summary

The pre-deployment summary screen (figure 2.50) displays a summary of the settings that have been entered using the wizard, and prompts the administrator to deploy or abort the installation with the chosen settings.

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

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

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

A regular node that has been configured with the OpenStack compute host role, ends up with OpenStack installed on it after the operating system running on the node is updated, and the newly-configured interfaces are set up. The update can be carried using `imageupdate` (section 5.6 of the *Administrator Manual*), or by rebooting the regular node. A reboot is however required if the interfaces have been changed, which is normally the case, unless the script is being run after a first run has already set up the changes. A reboot of the regular node is therefore normally a recommended action, because it ensures the updates and the interface changes have been carried out on the node. The reboot action is carried about by default (as shown in the preceding output), while the option is set by the `cm-openstack-setup` script (page 25).

Next, the administrator can further configure the cluster to suit requirements.

3

Ceph Installation

3.1 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 3.1)
- Ceph installation and management (section 3.2)
- RADOS GW installation and management (section 3.4)

3.1.1 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 3.1 illustrates these concepts.

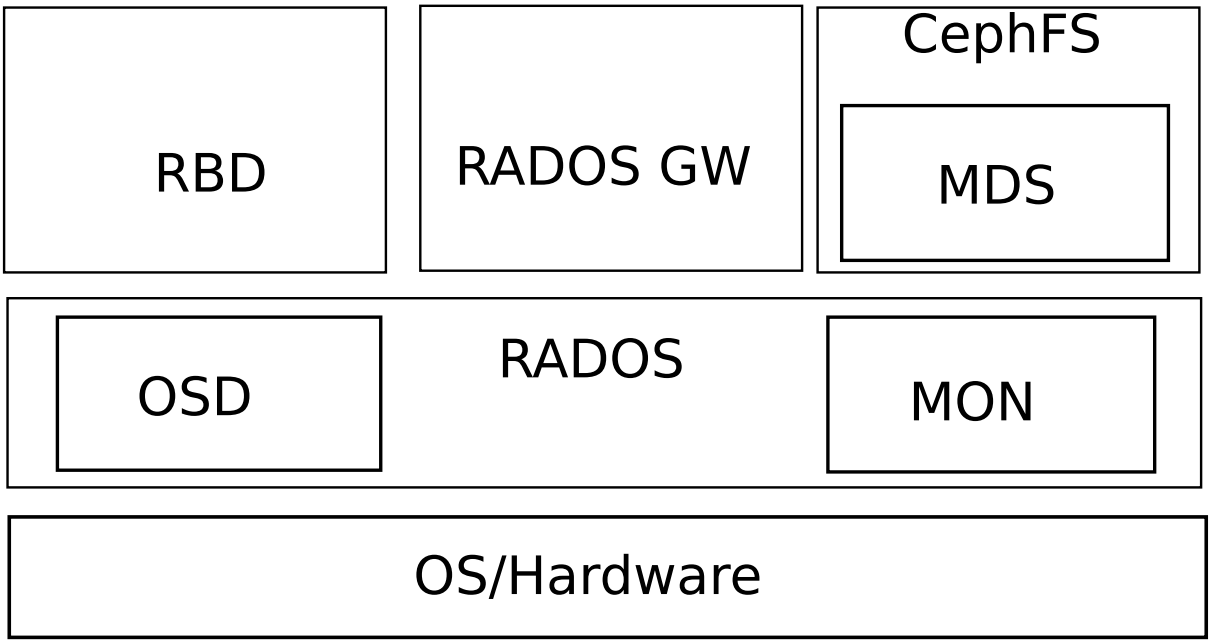

Figure 3.1: Ceph Concepts

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 is not regarded as production-ready by the Ceph project at the time of writing (July 2014), and is therefore not yet supported by Bright Cluster Manager.

3.1.2 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 180 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.

3.1.3 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.

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.

3.2 Ceph Installation With cm-ceph-setup

3.2.1 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.

3.2.2 Starting With Ceph Installation, Removing Previous Ceph Installation

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

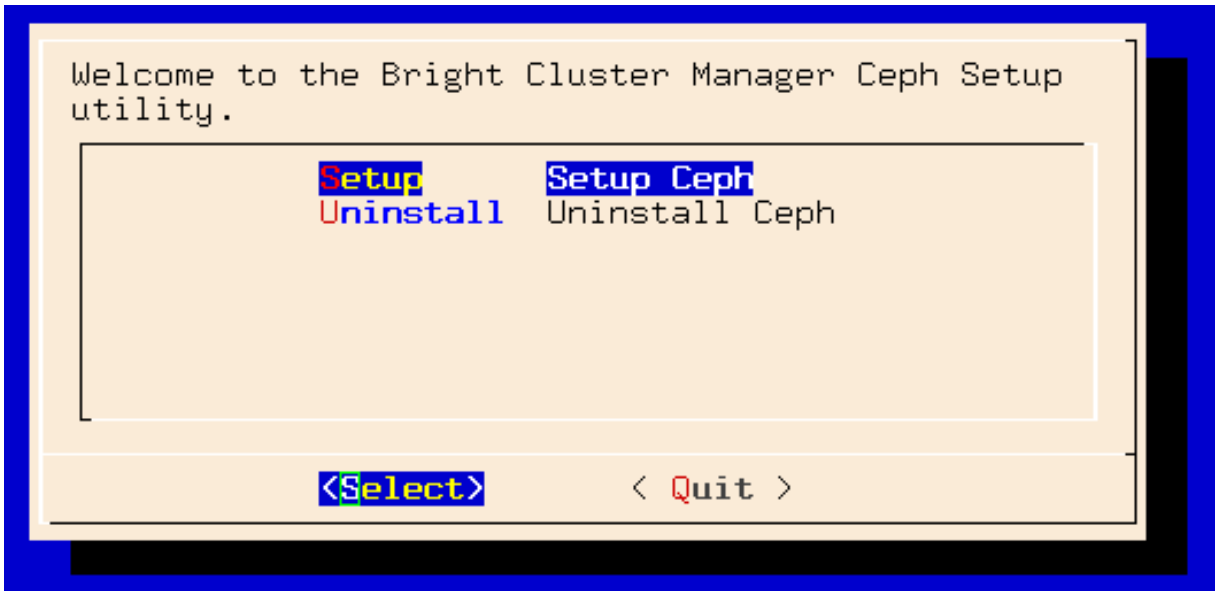

Figure 3.2: Ceph Installation Welcome

At the welcome screen (figure 3.2), the administrator may choose to

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

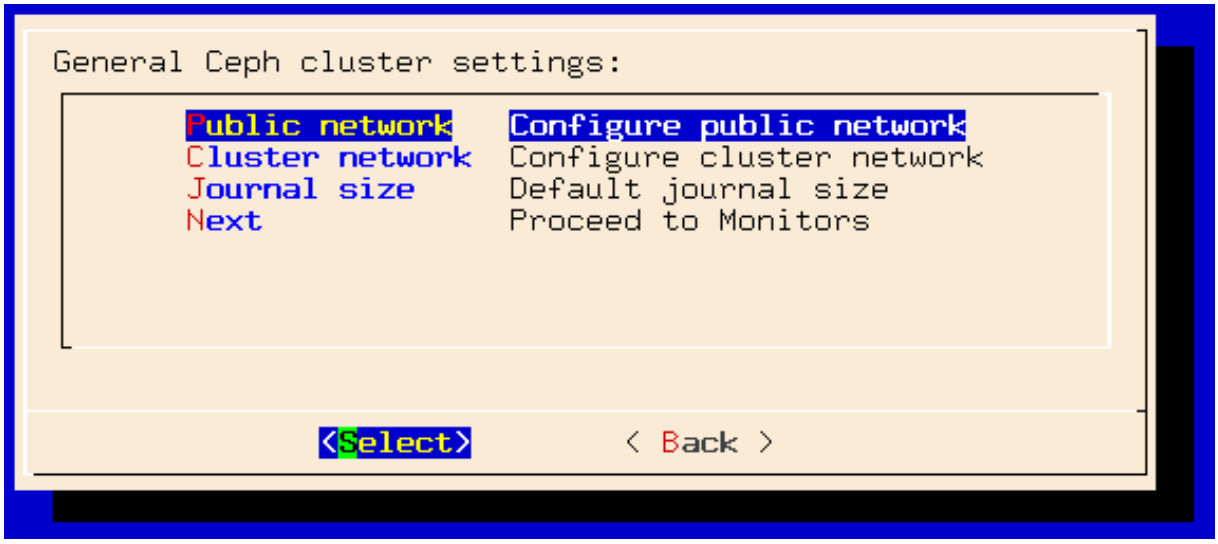

Figure 3.3: Ceph Installation General Cluster Settings

If the setup option is chosen, then a screen for the general Ceph cluster settings (figure 3.3) 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`.

- Private network: This is the network used by OSDs to communicate with each other. For a standard default Type 1 network this is internalnet.
- Journal size: The default OSD journal size, in MiBs, used by an OSD. The actual size must always be greater than zero. This is a general setting, and 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 (March 2015), Ceph software provides a default of 5GiB.

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

Selecting the Next option in figure 3.3 continues on with the next major screen of the setup procedure, and displays a screen for Ceph Monitors configuration (figure 3.4).

3.2.3 Ceph Monitors Configuration

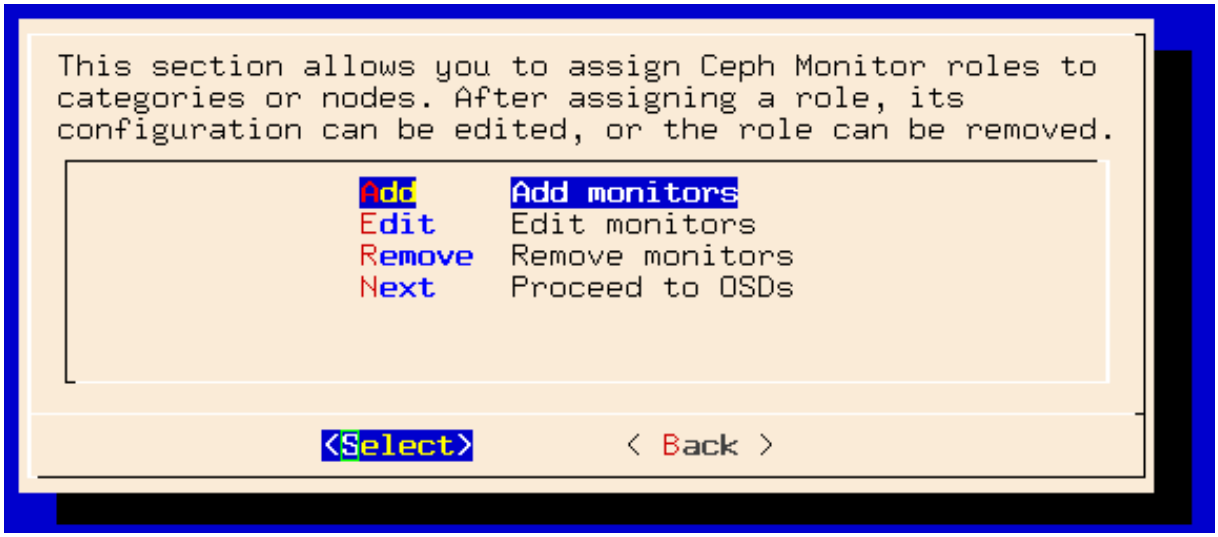

Figure 3.4: Ceph Installation Monitors Configuration

In this screen:

- Ceph Monitors can be added to nodes or categories.
- Existing Ceph Monitors can be edited or removed (figure 3.5), from 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.

Editing Ceph Monitors

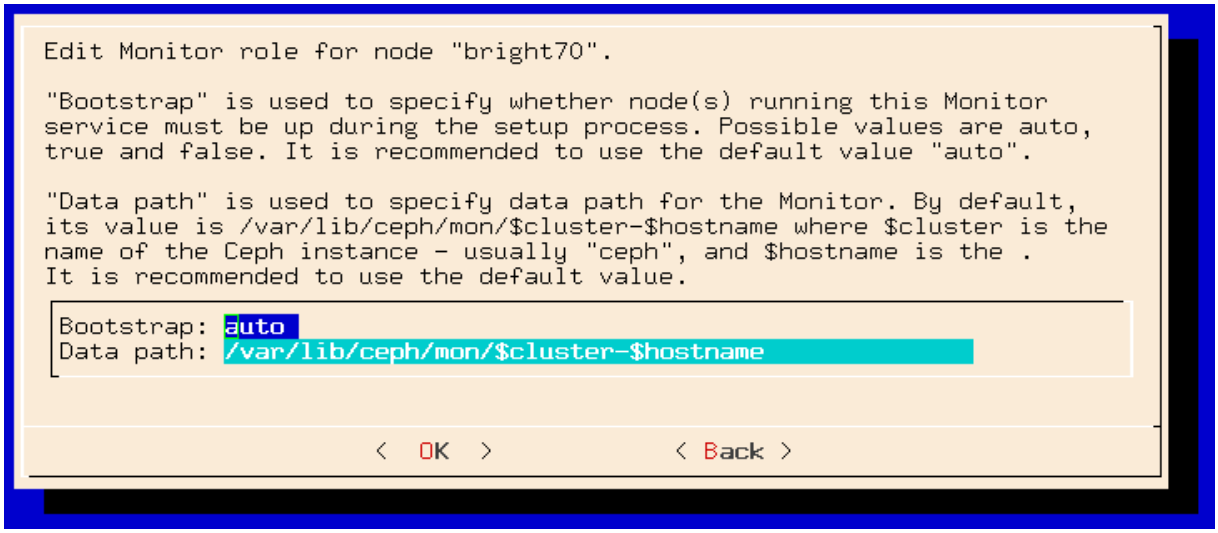

Figure 3.5: Ceph Installation Monitors Editing: Bootstrap And Data Path

The `Edit` option in figure 3.4 opens up a screen, figure 3.5, 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 3.4).

3.2.4 Ceph OSDs Configuration

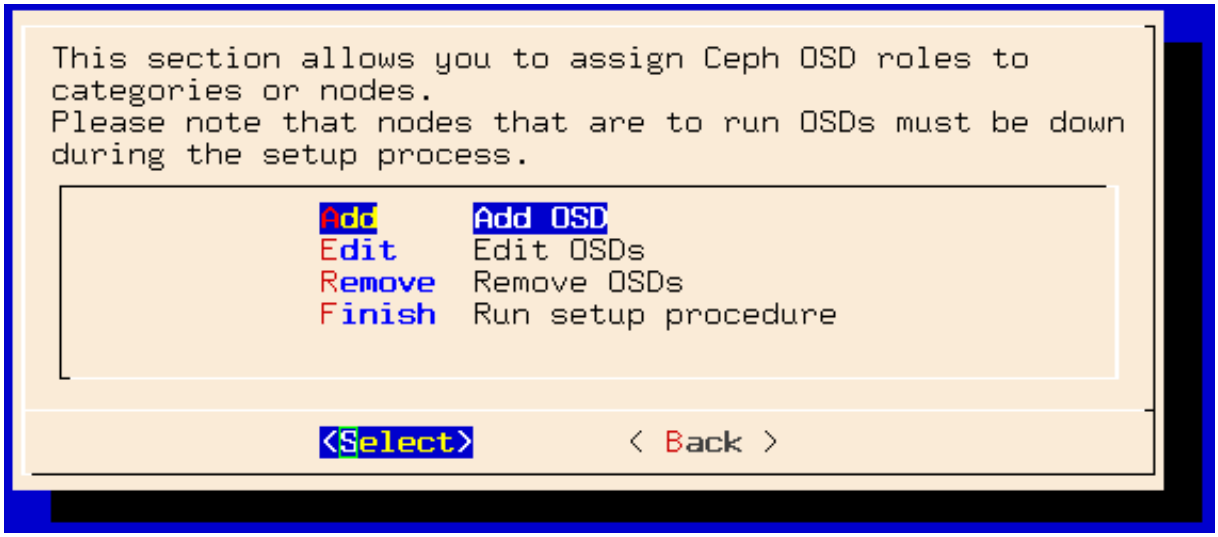

Figure 3.6: Ceph OSDs Configuration

If `Proceed to OSDs` is chosen from the Ceph Monitors configuration screen in figure 3.4, then a screen for Ceph OSDs configuration (figure 3.6) is displayed, where:

- OSDs can be added to nodes or categories. On adding, the OSDs must be edited with the edit menu.
- Existing OSDs can be edited or removed (figure 3.7), from nodes or categories.
- To finish up on the installation, after any changes to the OSD configuration have been made, the `Finish` option runs the Ceph setup procedure itself.

Editing Ceph OSDs

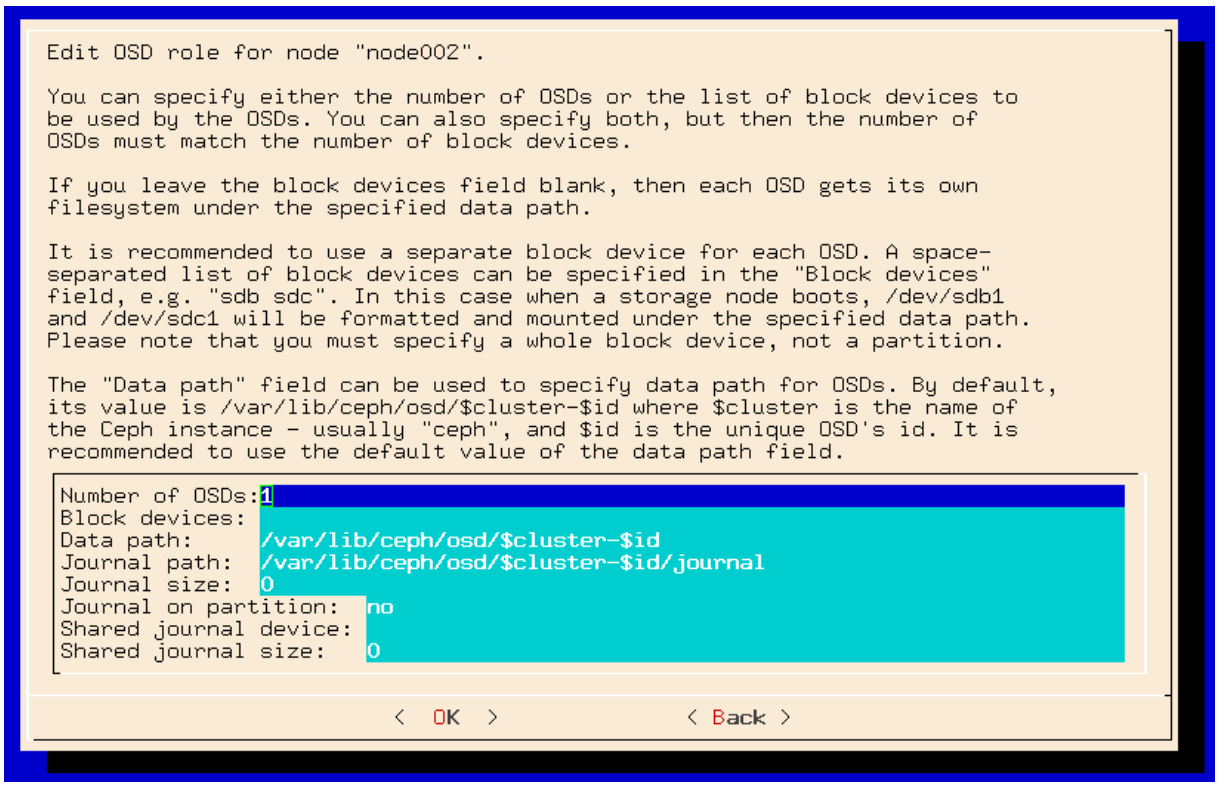

Figure 3.7: Ceph Installation OSDs Editing: Block Device Path, OSD Path, Journals For Categories Or Nodes

The `Edit` option in figure 3.6 opens up a screen, figure 3.7, 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 3.3). 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 (March 2015), 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. The path is unset by default, which means it is not used by default..

- The Shared journal size in MiB can be set. For n OSDs each of size x MiB, the value of Shared journal size is $n \times x$. That is, its value is the sum of the sizes of all the individual OSD journals that are kept on the shared journal device. If it is used, then:

  - The value of Shared journal size is used to automatically generate the disk layout setup of the individual OSD journals.
  - A value of 0 for the Journal size is invalid, and does not cause a Ceph default size to be used.

  The Shared journal size value is unset by default.

The Back option can be used after accessing the editing screen, to return to the Ceph OSDs configuration screen (figure 3.6).

Successful Completion Of The Ceph Installation

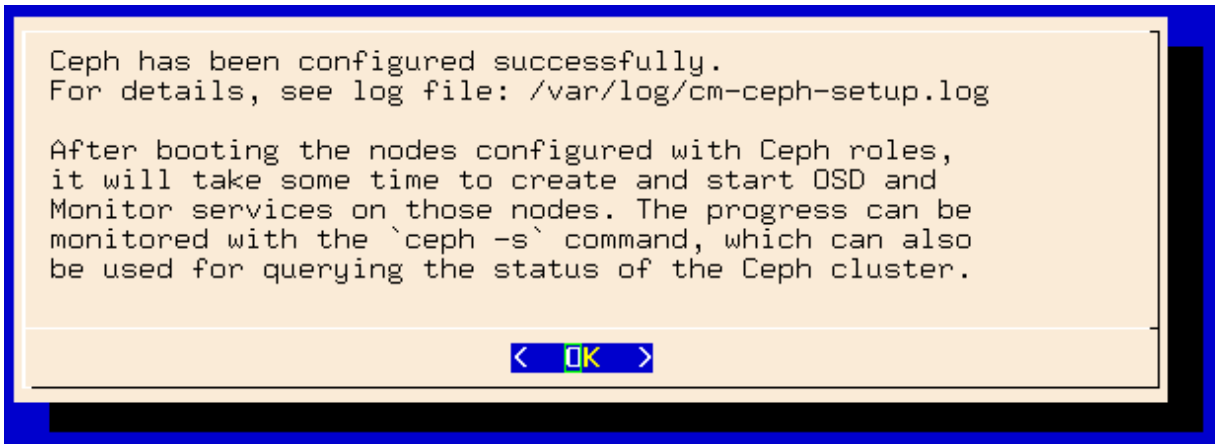

Figure 3.8: Ceph Installation Completion

After selecting the `Finish` option of figure 3.6, the Ceph setup proceeds. On successful completion, a screen as in figure 3.8 is displayed.

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

3.3.1 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@bright72 ~]# ceph -s
   cluster d9422c23-321e-4fa0-b510-ca8e09a0a1fc
     health HEALTH_OK
     monmap e1: 1 mons at bright72=10.141.255.254:6789/0, election ep\
och 2, quorum 0 bright72
     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@bright72 ~]# 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
```

3.3.2 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@bright72 ~]# cmsh
[bright72]% ceph
[bright72->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

```
[bright72->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     ...
-------------------------------- ---------- ----------  ...
bright72:.rgw                    1B         0           ...
bright72:data                    0B         0           ...
bright72:metadata                0B         0           ...
bright72: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

```
[bright72->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
[bright72->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@bright72 ~]# cmsh
[bright72]% ceph
[bright72->ceph[ceph]]% append extraconfigparameters "[mds.2] host=rabbit"
[bright72->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:

```
[bright72->ceph[ceph]]% append extraconfigparameters "[mds.2] host2=bunny"
[bright72->ceph*[ceph*]]% commit
```

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

```
[bright72->ceph[ceph]]% append extraconfigparameters "osd journal size = 128"
[bright72->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 49) and for Ceph monitoring (page 50).

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

```
[bright72->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

```
[bright72->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

```
[bright72->category[default]->roles]% use cephosd
[bright72...[default]->roles[cephosd]]% osdassociations
[bright72...osd]->osdassociations]% list -f name:10,osddata:30
name (key) osddata
---------- ------------------------------
osd0       /var/lib/ceph/osd/$cluster-$id
[bright72...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 48):

Example

```
[bright72...osd]->osdassociations]% use osd0
[bright72...osdassociations[ods0]]% show
Parameter                      Value
------------------------------ ------------------------------------
...
Automatically adjust weight    off
Extra config parameters
...
[bright72...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

```
[bright72]% device use bright72
[bright72->device[bright72]]% roles ; use cephmonitor
[ceph->device[bright72]->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 48).

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.

3.4 RADOS GW Installation, Initialization, And Properties

3.4.1 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.

3.4.2 Setting RADOS GW Properties

RADOS GW Properties In `cmsh`

RADOS GW properties can be managed in `cmsh` by selecting the device, then dropping into the `radosgateway` role:

```
[bright72]% device use bright72
[bright72->device[bright72]]% roles
[bright72->device[bright72]->roles]% use radosgateway
[bright72->device[bright72]->roles[radosgateway]]% show
Parameter                      Value
------------------------------ --------------------------------
Name                           radosgateway
Provisioning associations      <0 internally used>
Revision
Type                           RadosGatewayRole
Module                         mod_fastcgi.so
Server Port                    18888
Server Root                    /var/www
Server Script                  s3gw.fcgi
Server Socket                  /tmp/radosgw.sock
Enable Keystone Authentication yes
Keystone Accepted Roles
Keystone Revocation Interval   600
Keystone Tokens Cache Size     500
NSS DB Path                    /var/lib/ceph/nss
```

For example, setting `enablekeystoneauthentication` to `yes` and committing it makes RADOS GW services available to OpenStack instances, if they have already been initialized (section 3.4.1) to work with Bright Cluster Manager.

RADOS GW Properties In `cmgui`

RADOS GW properties can be accessed in `cmgui` by selecting the device from the resource tree, then selecting the `Ceph` subtab for that device, then ticking the `Rados Gateway` panel, then clicking on the `Advanced` button (figure 3.9).

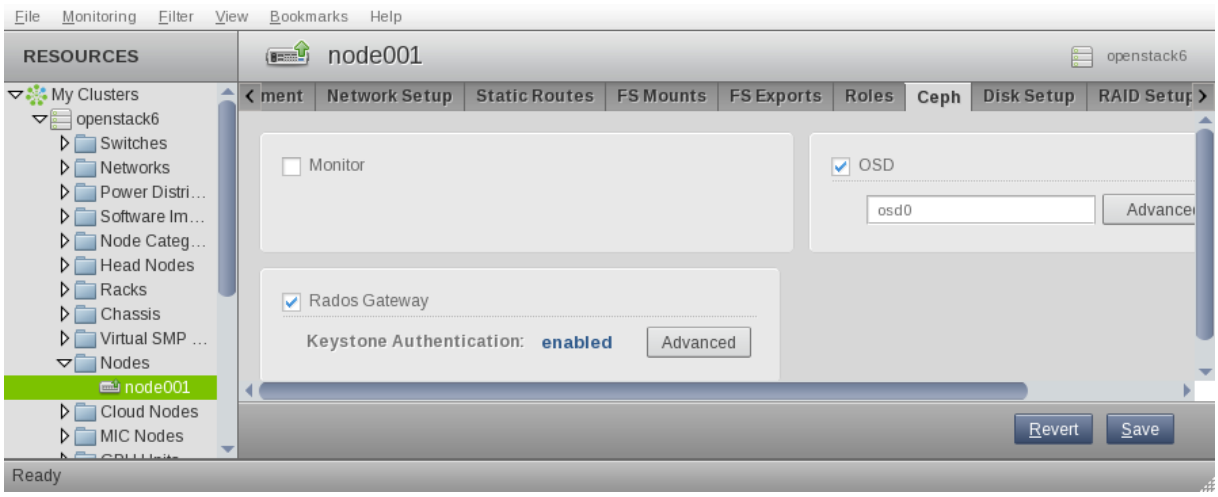

Figure 3.9: Accessing The RADOS GW Properties Button In `cmgui`

This brings up the RADOS GW `Advanced Role Settings` window (figure 3.10), which allows RADOS GW properties to be managed. For example, ticking the `Enable Keystone Authentication` checkbox and saving the setting makes RADOS GW services available to OpenStack instances, if they have already been initialized (section 3.4.1) to work with Bright Cluster Manager.

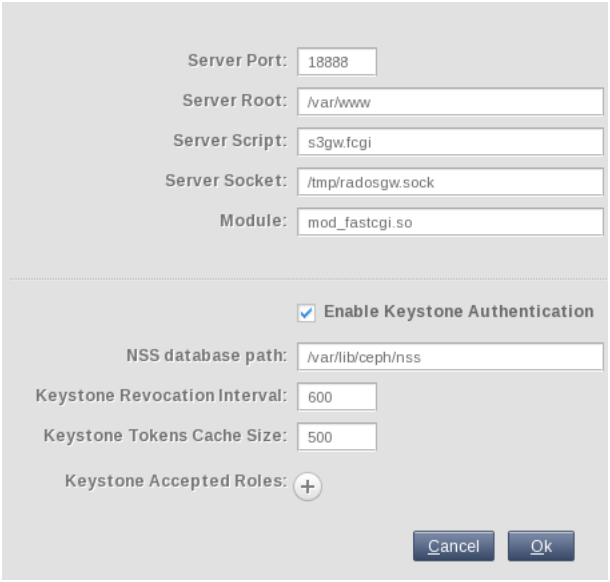

Figure 3.10: RADOS GW Advanced Role Settings In `cmgui`

4

User Management And Getting OpenStack Instances Up

In this chapter:

- Section 4.1 describes Bright Cluster Manager's user management integration with OpenStack.
- Section 4.2 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 4.3 describes how a Bright-managed instance is managed in Bright Cluster Manager

4.1 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 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
- using the Keystone Python API —this last option is more likely to be of interest to developers rather than system administrators

Having OpenStack users be the same as Bright users is often what administrators want. 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 2.1.4), and how the administrator configures the users using the initialization and migration scripts (section 4.1.2).

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

This section on database drivers is offered as background, and can be skipped by most administrators.

In the following table headers, there are two kinds of Bright users.

1. Any users created from before an OpenStack setup is carried out, are for convenience called *old users*.
2. Any users created after an OpenStack setup is carried out, are for convenience called *new users*.

It should be understood that new users are not OpenStack users by default. To make a Bright user able to use OpenStack under the same user name, some configuration must be carried out.

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

Driver used by Keystone	Bright Cluster Manager users can use OpenStack after running:
MySQL	migration script and initialization script
MySQL + PAM/NSS (Hybrid)	initialization script
Bright LDAP	initialization script

The first driver, has Keystone use only Galera's MySQL database for OpenStack users. Bright's regular database remains in use as another, independent database for Bright users. If a migration script and an initialization script is run on the Bright user name, then the user name is duplicated as an OpenStack user name, and also stored into Galera. The databases remain independent, which means that passwords for these names are not matched. They can be matched manually by the end user.

The second driver, has Keystone using Galera's MySQL and Bright Cluster Manager's PAM/NSS, and is called a hybrid driver. This driver can handle both OpenStack users, and Bright users that are authenticated with PAM/NSS, at the same time. With the hybrid driver, only the initialization script needs to be run to give a Bright user the ability to use OpenStack. Running the migration script is not required because the hybrid driver can deal with the existing PAM/NSS users already.

New users that are created with this driver from OpenStack actions are always entered into Galera's MySQL, while new users entered via a CMDaemon action, for example, `cmsh` using the the top-level `user` mode of Bright, remain non-OpenStack-using Bright users, unless explicitly initialized.

The third driver, using Bright's own LDAP database, does not use the OpenStack user database. That is, Keystone, when using this driver, handles Bright users only, and ignores the database of OpenStack users. With the Bright LDAP driver, if an initialization script is run for a user, then Bright users gain the ability to use OpenStack. Creation of a new user via OpenStack actions will fail, because the LDAP access is read-only. LDAP users can be created via a CMDaemon front-end, such as the top-level `user` mode of `cmsh` in Bright.

4.1.1 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 `user` mode, while for OpenStack users, it is done from within the `users` submodes 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:

```
|-- ...
|-- openstack
|   |-- ...
|   |-- settings
|   |   |--...
|   |   `-- users
|   |-- ...
|   `-- users
|-- ...
`-- user
```

4.1.2 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 57, Background Note: Automated Role Assignment In OpenStack).

The `cmsh` parameters `initscriptpath` 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@bright72 ~]# cmsh
[bright72]% openstack
[bright72->openstack[default]]% settings ; users
[...settings->users]% set initscriptpath /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 `Users` tabbed pane, and then selecting the `Users` subtab.

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 `initscript` 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
[bright72->user]% add fred
[bright72->user*[fred*]]% set password secret123; commit
```

For the Keystone + MySQL driver, 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 4.1):

Figure 4.1: 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 4.2):

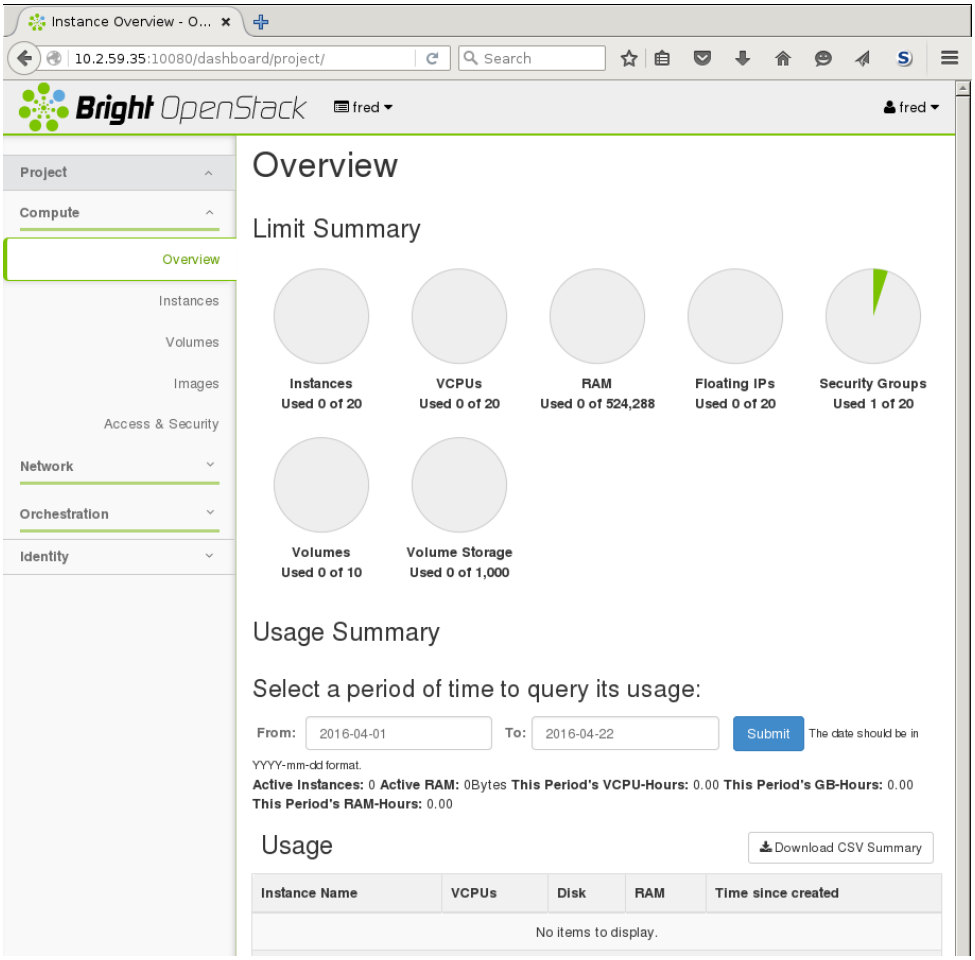

Figure 4.2: Successful Login With Horizon At `http://<load balancer IP address>:10080/dashboard`

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 4.2, 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

```
[bright72->user[fred]]% openstack users
[bright72->openstack[default]->users]% list -f name
name (key)
--------------------
admin
cinder
cmdaemon
fred
glance
heat
keystone
neutron
nova
[bright72->openstack[default]->users]% projects
[bright72->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
[bright72->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+
[bright72->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, the administrator can set the following parameters to configure the `.openstackrc*` files:

- `RC Write Out Mode`: This parameter configures how the `.openstackrc` file is written for an OpenStack user:
  - `matchhome`: writes the file only to home directories that match OpenStack user names
  - `all`: 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 password`: This parameter decides if the `.openstackrc_password` file is written for an OpenStack user. This feature onlyl 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@bright72 ~]# cmsh
[bright72]% openstack
[bright72->openstack[default]]% settings; users
[...settings->users]% set rcwriteoutmode matchhome
[...settings->users*]% set writeoutpassword 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 56, then the home directory for `fred` would look something like:

Example

```
[root@bright72 ~]# 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@bright72 ~]# cat /home/fred/.openstackrc_password
OS_PASSWORD="LMlr6oRENZoIp0iqaI4304JGNn632P"

[root@bright72 ~]# 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, and is the address 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 55). The end user can, if required, update the `~/.openstackrc_password` file by hand after a password change is made by the dashboard.

4.2 Getting A User Instance Up

By default, after creating a user as in the example where user `fred` is created (page 56) 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.

4.2.1 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@bright72 ~]$ 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@bright72 ~]$ openstack image create --disk-format qcow2 --public --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 `rcwriteoutmode` option (page 58), and it can be sourced with:

Example

```
[fred@bright72 ~]$ . .openstackrc
```

Sourcing means running the file so that the environment variables in the file are set in the shell on return.

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 IP address>*`:10080/dashboard/project/images/` directly (figure 4.3).

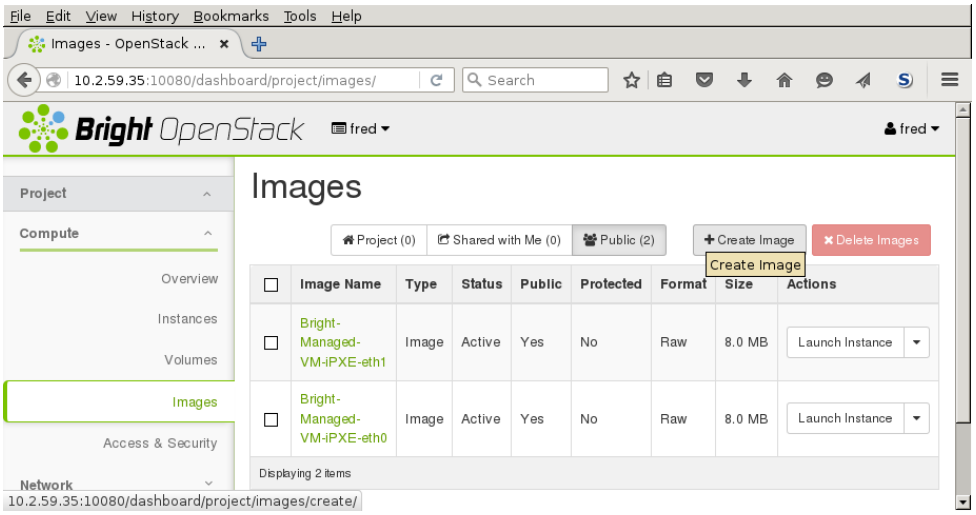

Figure 4.3: Images Pane In Horizon

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 4.4).

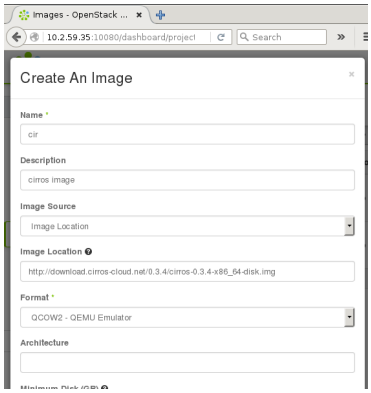

Figure 4.4: Images Pane—Create Image Dialog

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 4.2.2.

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 4.5).

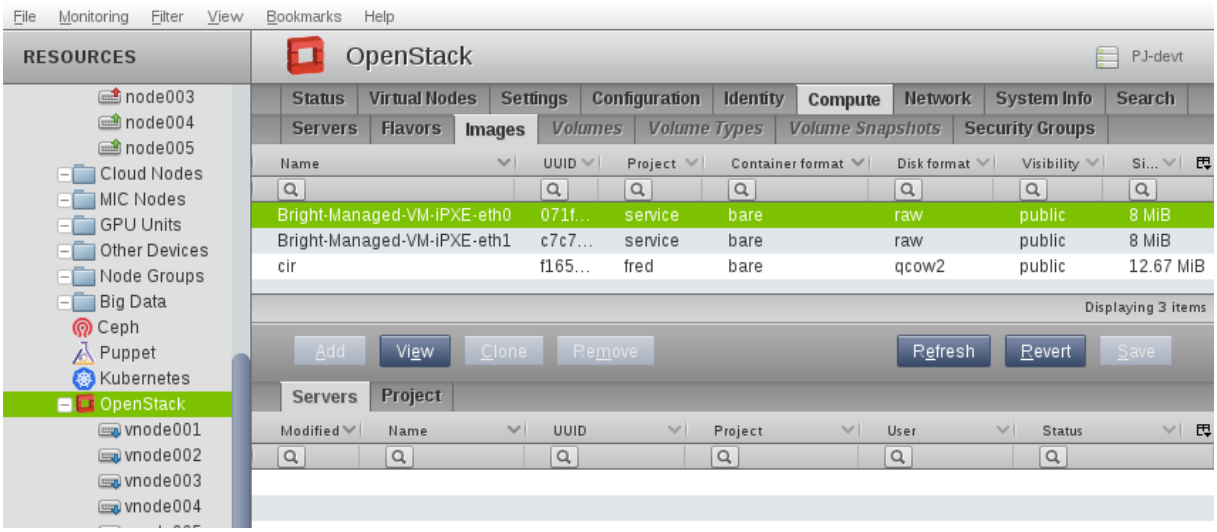

Figure 4.5: OpenStack Image Properties In `cmgui`

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

Launching an image that is installed as in section 4.2.1 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 4.2.1. 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 4.6):

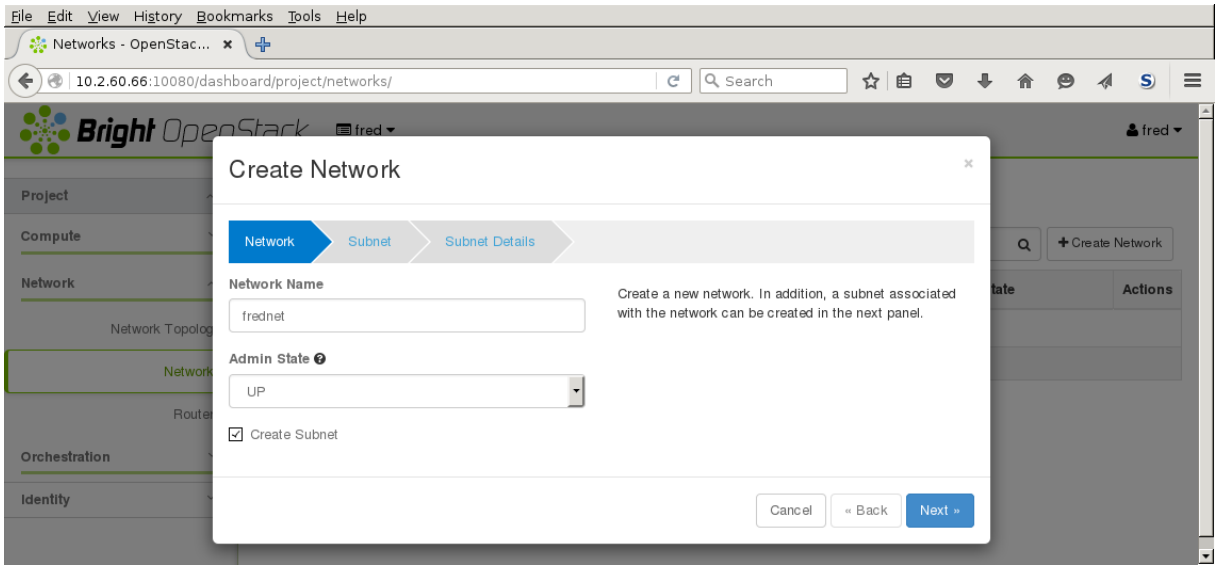

Figure 4.6: End User Network Creation

Similarly, in the next screen a subnet called `fredsubnet` can be configured, along with a gateway address for the subnet (figure 4.7):

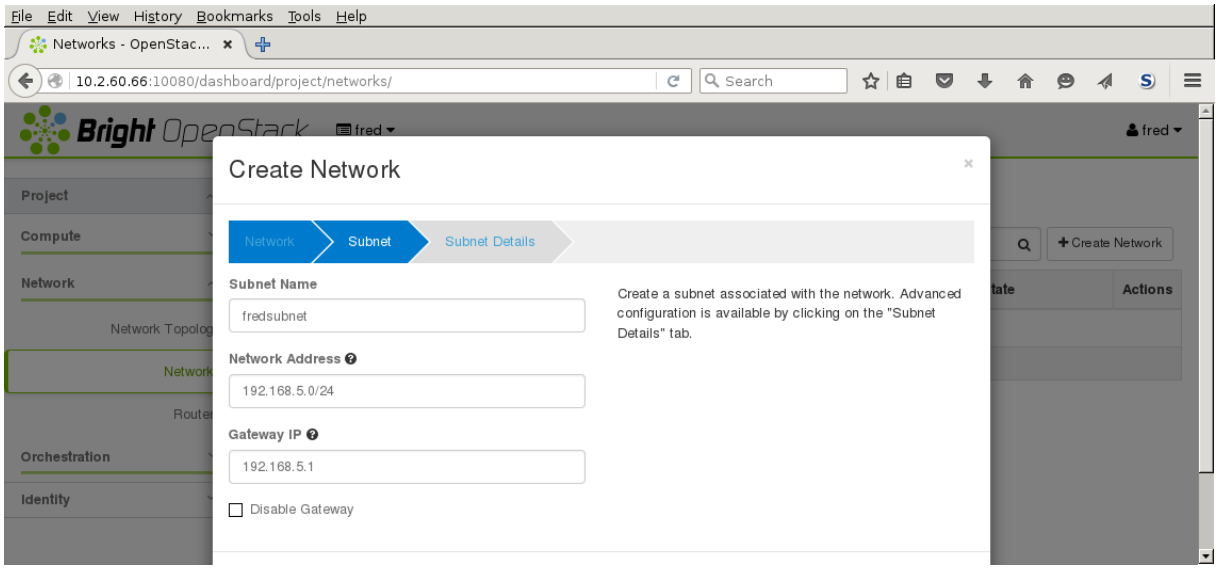

Figure 4.7: End User Subnet Creation

In the next screen (figure 4.8):

- 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

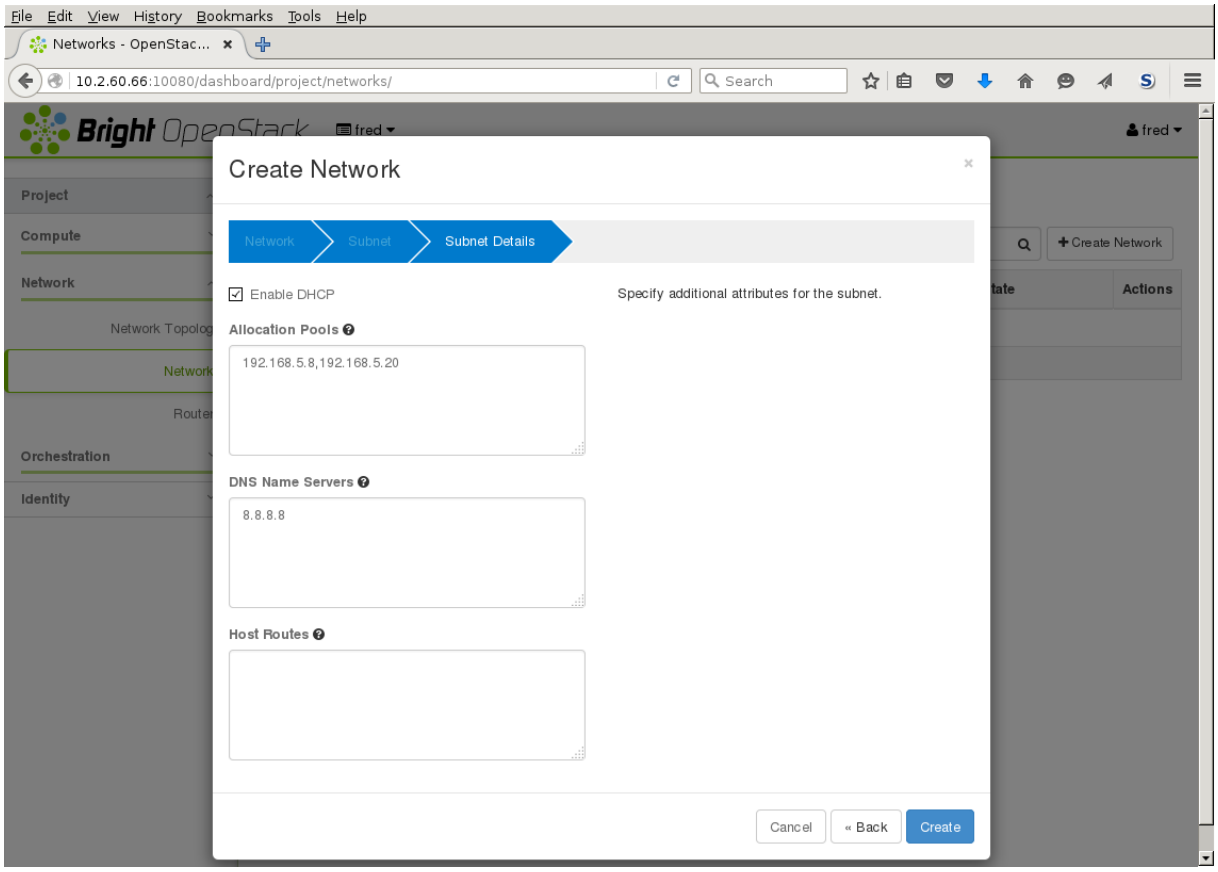

Figure 4.8: End User DHCP, DNS, And Routes

At the end of a successful network creation, when the dialog box has closed, the screen should look similar to figure 4.9:

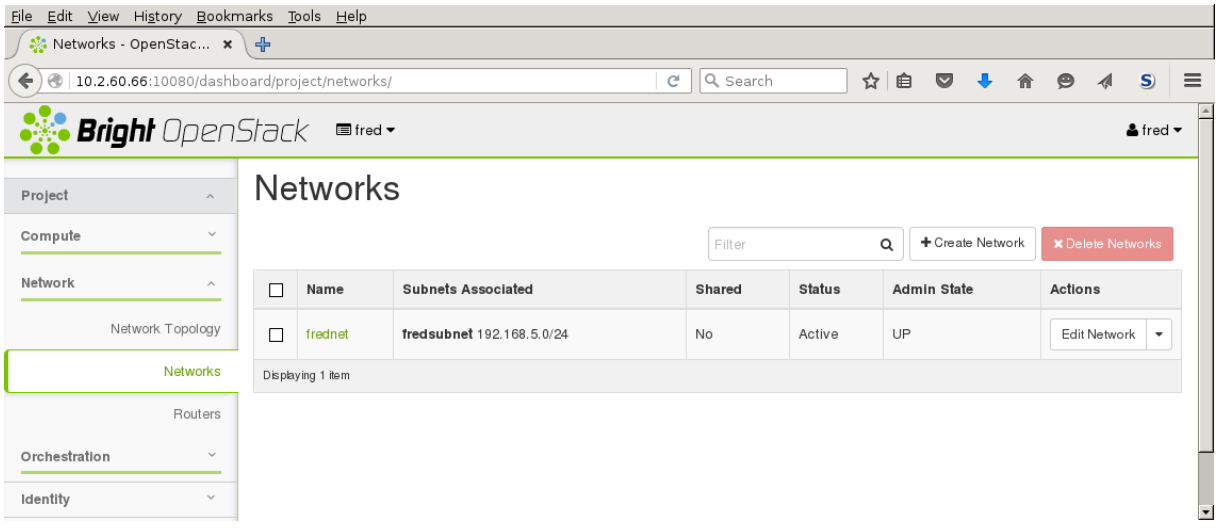

Figure 4.9: End User Node Network Configuration Result

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.

4.2.3 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 2.1.13).

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 62. 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 4.10):

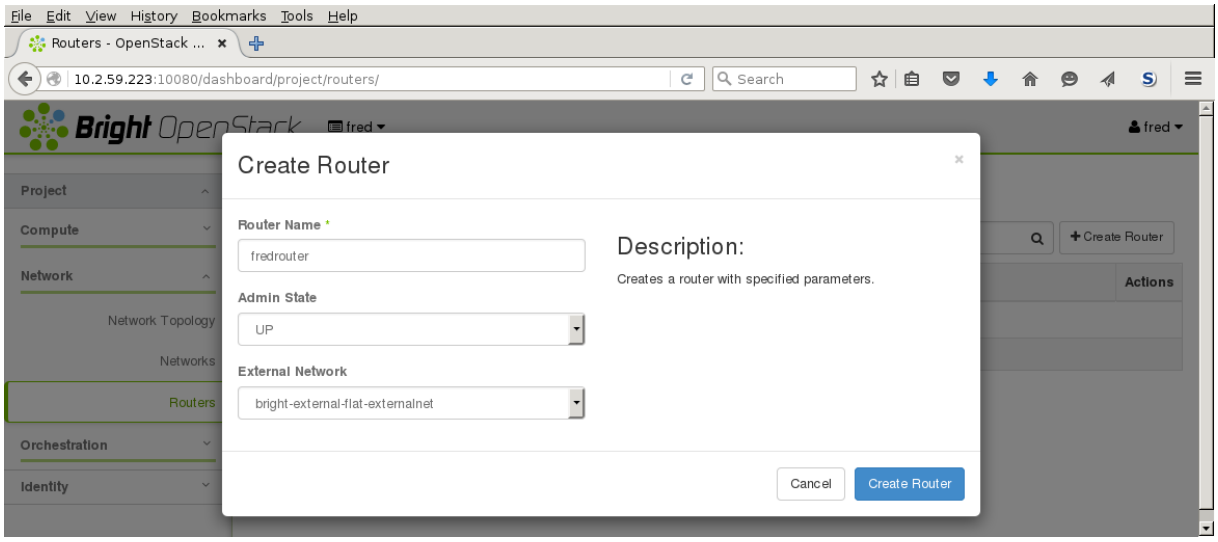

Figure 4.10: End User Router Creation

The router can be given a name, and connected to the external network 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 4.11):

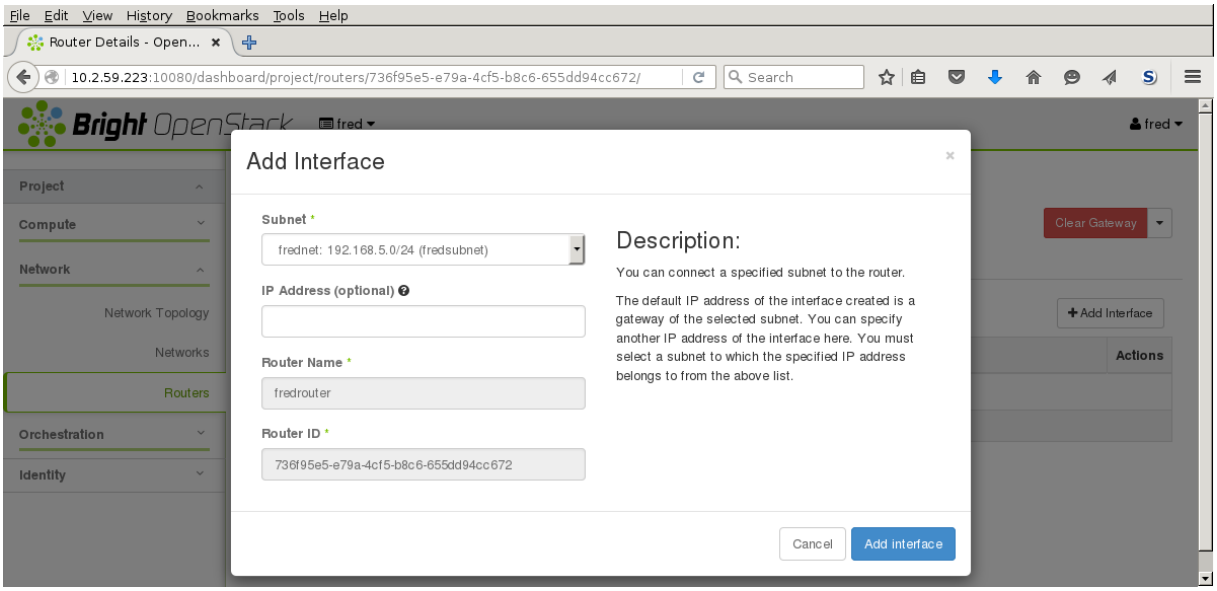

Figure 4.11: End User Router Interfaces Creation

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 4.12):

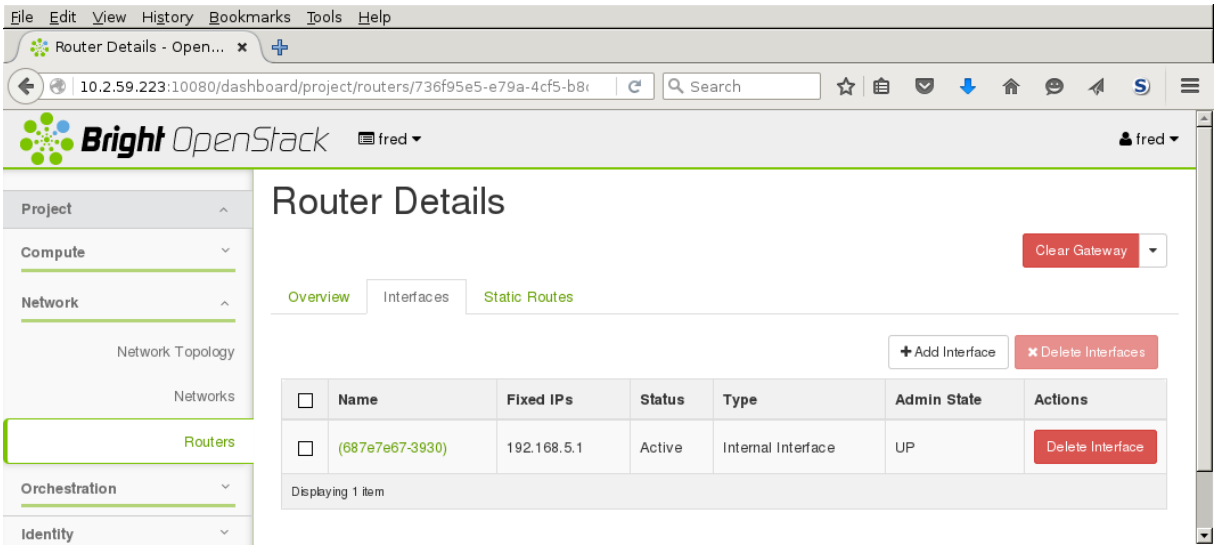

Figure 4.12: End User Router Interface Screen After Router Configuration

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 4.13):

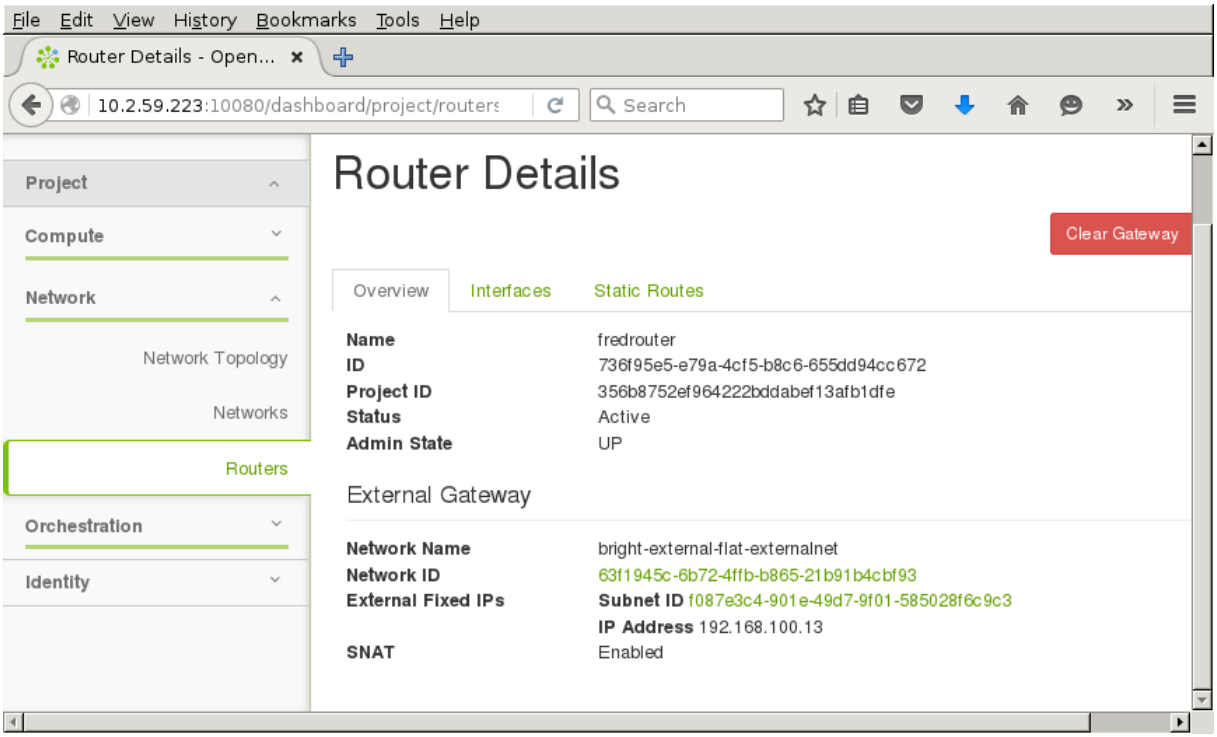

Figure 4.13: End User Router Details After Router Configuration

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

Example

```
[fred@bright72 ~]$ 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 network node.

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 4.14):

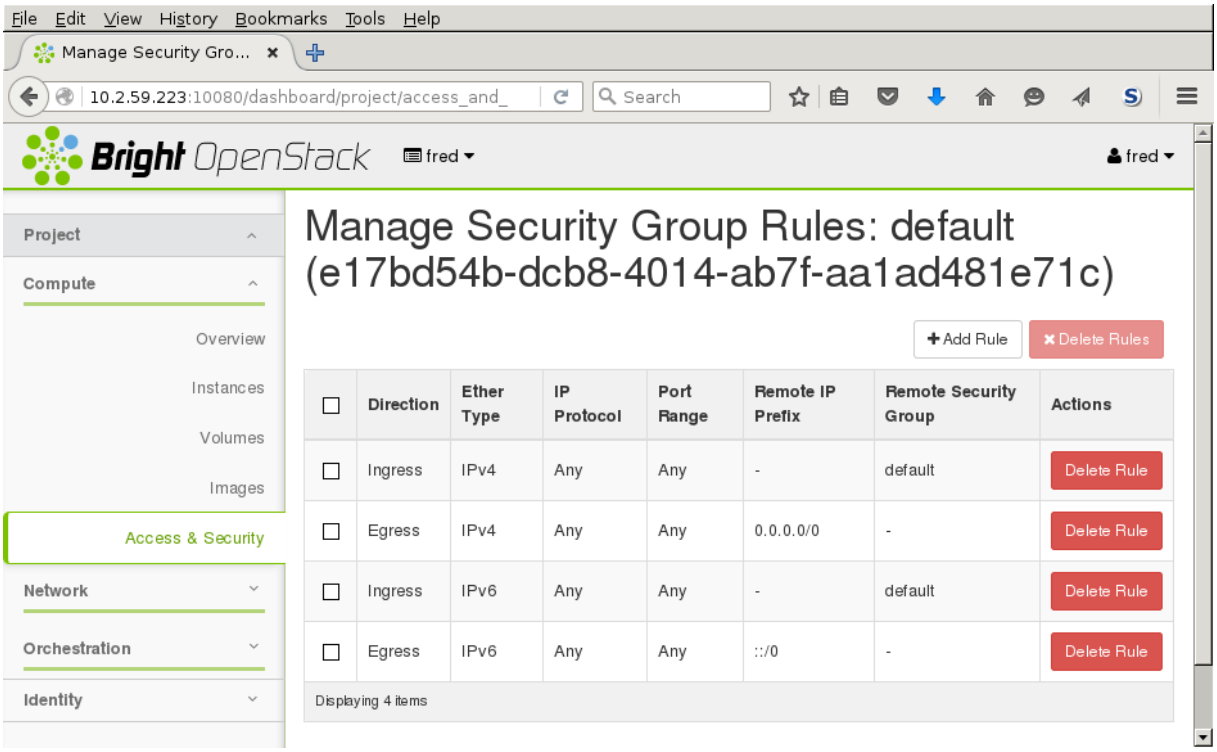

Figure 4.14: Security Group Rules Management

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 4.15).

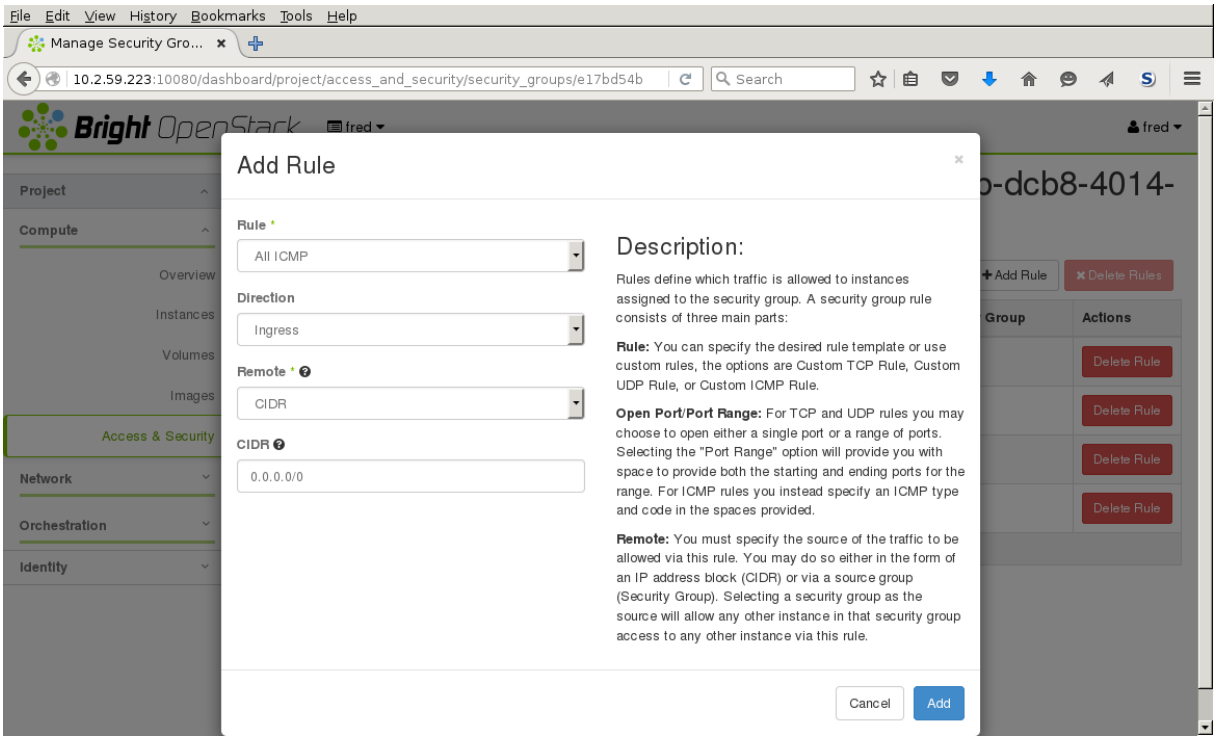

Figure 4.15: Security Group Rules Management—Adding A Rule

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 4.16).

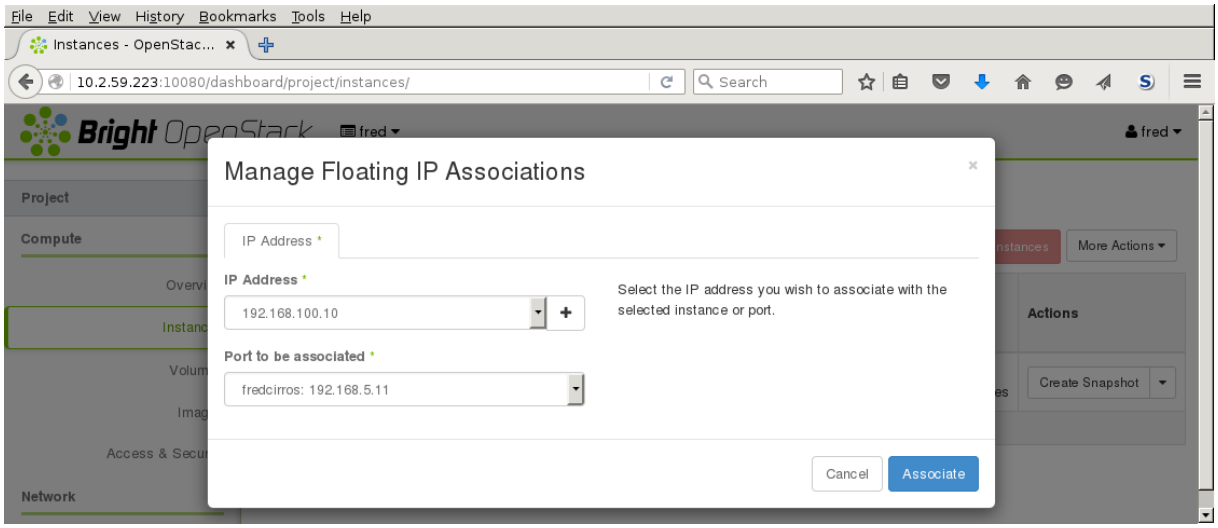

Figure 4.16: Associating A Floating IP Address To An Instance

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

Example

```
[fred@bright72 ]$ 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@bright72 ~]$ 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.

4.3 Running A Bright-managed Instance

A Bright-managed instance is a special case of the user instance in section 4.2. 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 specifications of the vnode can be set in several ways, including:

- during OpenStack installation, where an automated calculation is done to set a default number of vnodes for the cluster (section 2.2.21, figure 2.42)
- 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 4.1 and 4.2.

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

Example

```
[root@bright72 ~]# 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 4.17):

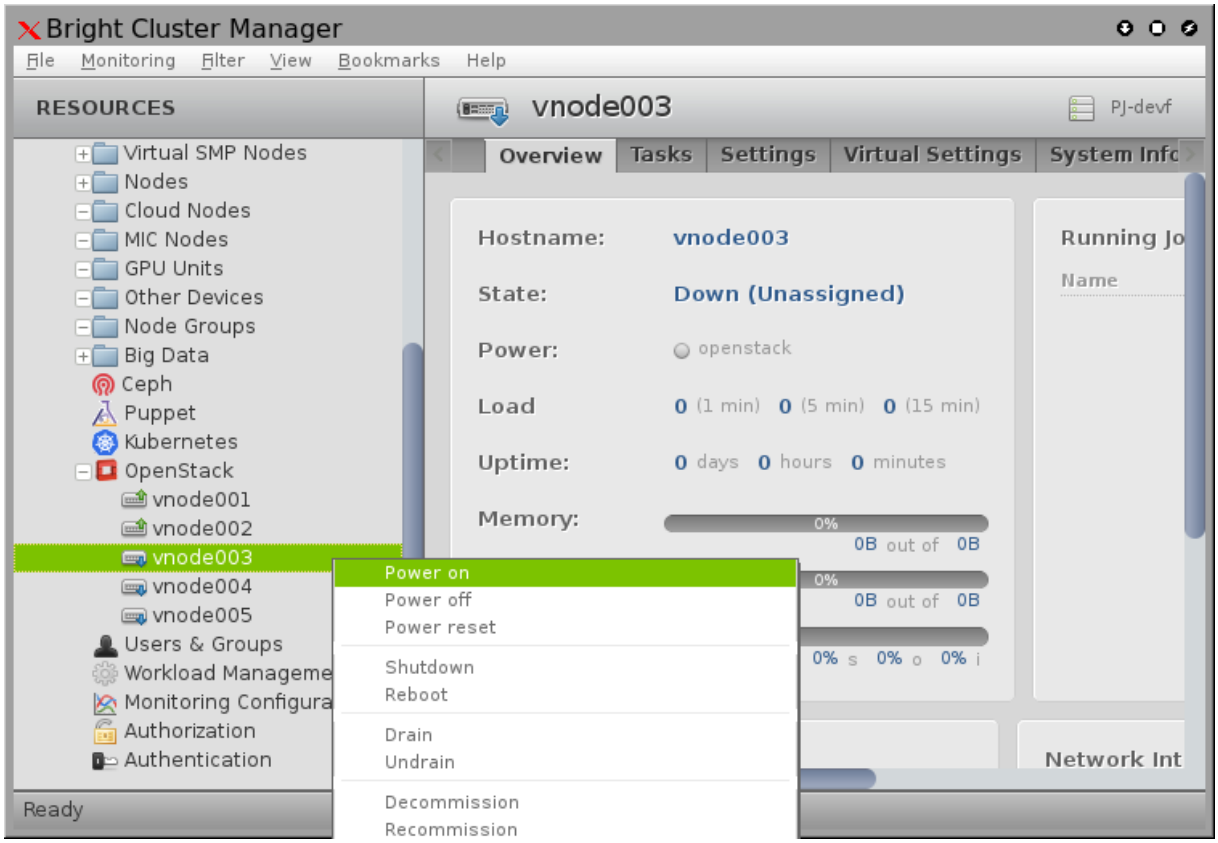

Figure 4.17: Powering On A Vnode Instance

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.