Fuel User Guide

version 8.0

Contents

Preface

This documentation provides information on how to use Fuel to deploy OpenStack environments. The information is for reference purposes and is subject to change.

Intended Audience

This documentation is intended for OpenStack administrators and developers; it assumes that you have experience with network and cloud concepts.

Documentation History

The following table lists the released revisions of this documentation:

Revision Date	Description
February, 2016	8.0 GA

Introduction to the User Guide

The Fuel User Guide provides instructions on how to configure, test, and operate OpenStack environments using Fuel web UI and CLI.

If you have already deployed OpenStack environments using earlier versions of the Fuel software, see the *Upgrading Fuel* section in the Fuel Installation Guide for instructions on upgrading your existing OpenStack distribution and the Fuel software.

Before you read this document, you must install the Fuel Master node as described in the *Fuel Installation Guide*.

Create a new OpenStack environment

After you install the Fuel Master node, your Fuel Slave nodes appear as **Unallocated nodes** in the Fuel web UI. You can now create, configure, and deploy your first OpenStack environment. You can deploy and manage multiple OpenStack environments using one Fuel Master node. However, you must create each environment separately.

This section includes the following topics:

Create an OpenStack environment in the deployment wizard

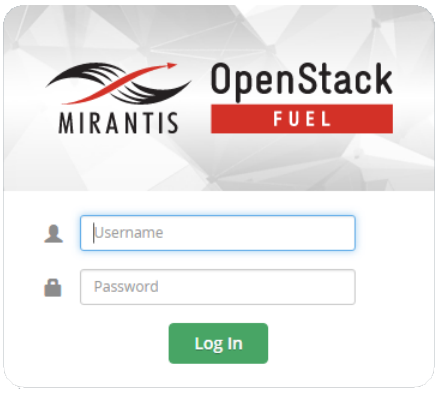
Before you deploy an OpenStack environment, you must decide and prepare hardware for the network topology, types of storage, hypervisor, OpenStack release version, additional OpenStack services, and other components that you want to deploy. Additional OpenStack programs and Fuel plugins may have additional hardware requirements and architectural limitations. For more information, see: *System requirements* in the *Fuel Installation Guide*.

If you are deploying a Mirantis OpenStack environment that is integrated with VMware vSphere, follow the instructions in *Deploying an OpenStack environment with VMware vCenter as a hypervisor*.

To create an OpenStack environment:

1. Access the Fuel web UI by pointing your web browser to http://10.20.0.2:8443.

   The Fuel login screen appears:



2. Log in to the Fuel web UI as *admin*:
    1. If you did not change the default password for *admin*, use the default password *admin*.
    2. If you changed the default password for *admin*, use that password.

    > ***Warning***
    >
    > For your security, change the default password. See: *Change the Fuel Master node administrative password*.

3. If you are logging in to the Fuel Web UI for the first time, select whether you want to send usage statistics or not by clicking *Connect now* or *Connect later*.
4. Click *New OpenStack Environment*.

    The deployment wizard starts.
5. In the *Name and Release* screen, type a name of the OpenStack environment and select an OpenStack release and an operating system on which you want to deploy your OpenStack environment.
6. In the *Compute* screen, select a hypervisor.

    By default, Fuel uses QEMU with KVM acceleration.
7. In the *Networking Setup* screen, select a network topology.

    By default, Fuel deploys Neutron with VLAN segmentation.
8. In the *Storage Backends*, select options for the storage back ends.

    By default, Fuel deploys Logical Volume Management (LVM) for Cinder, local disk for Swift, and Swift for Glance.
9. In the *Additional Services*, select additional OpenStack programs that you want to deploy.
10. In the *Finish* screen, click *Create*.

    Fuel creates an OpenStack environment. Before you can use the environment you must add nodes, verify network settings, and complete other configuration tasks.
11. Proceed to *Configure your Environment*.

Change the Fuel Master node administrative password

We highly recommend that you change the default password for the *admin* user to the one that meets your company's security requirements.

To change the Fuel Administrator password:

1. In the Fuel web UI, click the user icon:

1. Click *Change Password*.

2. Type the current password, the new password, and then confirm the new password. To display what you type, click the eye icon.
3. Click *Apply*.

Seealso

- *Fuel Access Control*

Configure your Environment

After you create an OpenStack environment as described in *Create a new OpenStack environment*, you must add nodes, assign roles, and verify your network configuration. Additionally, you can modify some of the settings you have already configured, as well as configure other important settings, such as disk partitioning, network interface bonding, and so on.

This section includes the following topics:

- *Add a node to an OpenStack environment*
- *Configure network settings*
- *Verify network configuration*
- *Map a logical network to a physical interface*
- *Edit the offloading mode*
- *Configure disk partitioning*
- *Configure network bonding*
- *Configure the OpenStack environment settings*
- *Install the Hadoop cluster service*
- *Install the OpenStack Bare Metal service*
- *Deploy changes*

Add a node to an OpenStack environment

To deploy an OpenStack environment using the Fuel web UI, you must add at least one controller node. However, for a fully functional cloud you must allocate a sufficient number of compute and storage nodes.

An OpenStack node, or a node, is a physical or virtual server that you provision to run a specific set of OpenStack services.

A role is a functional set of services that Fuel installs as a whole on a node, usually in its own disk partition. Some roles can be combined in one node.

The number of discovered unallocated and total nodes is displayed in the upper right corner in the Fuel web UI.

For more information, see *System Requirements* in *Fuel Installation Guide*.

To add a node to an OpenStack environment:

1. Log in to the Fuel web UI.
2. In the *Dashboard* tab, click *Add nodes*.
3. Assign a role or roles to the node by selecting the corresponding option.
4. In the list of discovered nodes, select a physical or virtual node to provision.
5. Optionally, display the node hardware configuration by clicking the *Settings* icon next to the discovered node.
6. Click *Apply Changes*.
7. Repeat step 2 - step 6 for all nodes that you want to include into this OpenStack environment.

   After you add nodes, Fuel enables you to deploy your OpenStack environment. However, you may need to apply additional changes to fully address your infrastructure requirements.

Seealso

- *System requirements* in *Fuel Installation Guide*
- *Label an OpenStack node*
- *Modify the Fuel Slave node host name*

Label an OpenStack node

In large deployments, sorting nodes by roles may not be efficient. Therefore, Fuel provides the capability to add custom labels to OpenStack nodes and later sort and display the nodes with that label. For example, you can label nodes located in one rack as *rack #1* and so on. Labels can be added and removed before or after you deploy an OpenStack environment.

Label an OpenStack node:

1. Log in to the Fuel web UI.

2. Click *Nodes*.
3. Select a node or nodes that you want to label.
4. Click the label icon.
5. Click *Add label*.
6. Type a *Name* and *Value*.

   Example:

   - **Name:** Row
   - **Value:** 1

   > ***Note***
   >
   > You can have multiple labels with identical names and different values. However, you cannot assign labels with identical names and different values to one node. For example, you cannot assign label *Row 1* and *Row 2* to one node, but you can assign them to different nodes.

7. Click *Apply*.

Change the role of a node

If you have assigned a wrong role or want to add additional roles to a node, you can modify this setting before you deploy an OpenStack environment, as well as after the deployment.

To change the role of a node:

1. In the Fuel web UI, click *Nodes*.
2. Select a node.
   - If the OpenStack environment is not yet deployed:
     1. Click *Edit Roles*.
     2. Modify the role as required.
   - If the OpenStack environment has been already deployed:
     1. Click *Delete*.

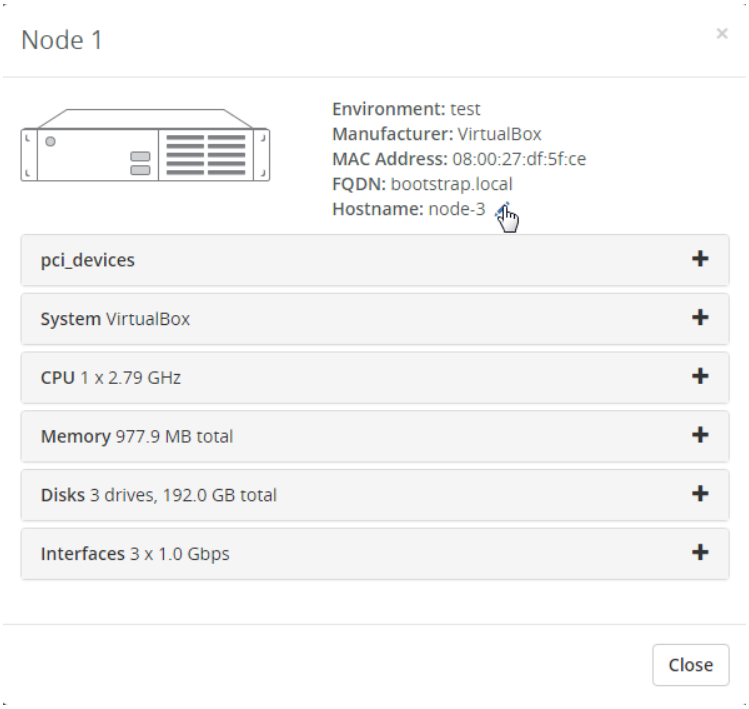
        Fuel changes the node's status to *Unallocated*.
     2. Click *Add Node*.
     3. Select the node and assign a new role or roles to the node as described in *Add a node to an OpenStack environment*.

Modify the Fuel Slave node host name

You can modify host names of the Fuel Slave nodes before you deploy an OpenStack environment. This functionality enables you to assign host names that match your corporate standards or a naming convention of your choice. You cannot change a host name of a Fuel Slave node after you deploy an OpenStack environment.

To modify the Fuel Slave node host name using Fuel web UI:

1. Log in to the Fuel web UI.
2. Click the *Nodes* tab.
3. Click the settings icon next to the corresponding node.
4. Click the edit icon:



5. Type the new host name.
6. Click *Close*.

To modify the Fuel Slave node host name using Fuel CLI:

1. Log in to the Fuel Master node CLI.
2. Type:

```
fuel node --node <NODE_ID> --hostname <NODE_HOSTNAME>
```

Value	Description

<NODE_ID>	A specific node indentificator. You can get the information about the node ID by typing: `fuel nodes`
<NODE_HOSTNAME>	A new host name for the selected node.

Configure network settings

After you select a network topology in the deployment wizard, you can further configure Neutron L2 and L3 settings, as well as modify the node network group configuration.

Fuel creates the *default* node network group that includes the Public, Storage, Management, Private, and Baremetal networks if you installed the OpenStack Bare Metal service. The `default` node group also uses the shared Admin network. You can modify parameters of the Admin network for all other node network groups, if any, except the `default` node network group.

If you configure multiple node network groups, Fuel configures a gateway for all networks.

> ***Note***
>
> When you deploy an environment using Fuel, you can exclude specific IP addresses so that Fuel does not assign them. This helps to avoid network conflicts in case these IP addresses were previously reserved by other network entities.
>
> To prevent IP address collisions, set the IP address range to be used by Fuel. For example, for nodes and VIPs, excluding the reserved IPs. In addition, you can specify multiple IP address ranges. If you have an IP address in use in the middle of the network IP address range, you can split the range to exclude the IP addresses in use.

To configure network settings:

1. In the Fuel web UI, click the *Network* tab.
2. Select a setting and modify as needed.

Seealso

- *Configure network bonding*
- *Edit the offloading mode*
- *Map a logical network to a physical interface*

Configure network bonding

Network bonding, or network aggregation, or NIC bonding is a network technology that enables you to maximize throughput by aggregating multiple physical links into a single high-speed aggregated network interface. In addition to increasing bandwidth, network bonding provides fault tolerance.

You must configure NIC bonding before or in the scope of mapping logical networks to physical network interfaces.

The following tables describe the types of one-side and two-side bonding that Fuel supports.

Types of one-side bonding

Name	Description
`balance-rr`	Implements the round-robin policy. This mode provides load balancing and fault tolerance.
`Active-backup`	Implements the active-backup policy. In this mode, one network interface is active and other network interface is passive. When an active network interface fails, a failover occurs and the previously passive NIC becomes active.
`balance-xor`	Implements the XOR policy. Transmit network packets are based on the selected transmit hash policy. This mode provides load balancing and fault tolerance.
`Broadcast`	Implements the broadcast policy. Transmits network traffic on all slave interfaces. This mode provides fault tolerance.
`balance-tlb`	Adaptive transmit load balancing based on the link utilization. This mode provides load balancing and fault tolerance.
`balance-alb`	Adaptive transmit and receive load balancing based on the link utilization. This mode provides load balancing and fault tolerance.
`balance-slb`	Modification of the `balance-alb` mode. SLB bonding enables a limited form of load balancing. The mode does not require information about the remote switch. SLB assigns each source MAC and VLAN pair to a link and transmits all packets from the MAC and VLAN pair through that link.
`balance-tcp`	Adaptive transmit load balancing among network interfaces.

Types of one-side bonding

Name	Description
`layer2`	Uses XOR of hardware MAC addresses to generate the hash.
`layer2+3`	Uses a combination of layer2 and layer3 protocol information to generate the hash.
`layer3+4`	Uses the upper layer protocol information, when available, to generate the hash.
`encap2+3`	Uses the same formula as `layer2+3`, but relies on `skb_flow_dissect` to obtain the header fields which may result in the use of inner headers if an encapsulation protocol is used.
`encap3+4`	Similar to `encap2+3`, but uses``layer3+4``.

To configure network interfaces:

1. In the Fuel web UI, click the *Nodes* tab.

2. Select nodes.
3. Click *Configure Interfaces*
4. Select network interfaces that you want to aggregate.
5. Click *Bond Network Interfaces*.
6. In the *Mode* drop-down list, select an appropriate bonding mode.

> ***Note***
>
> When bonding an Admin interface, you can select the `balance-rr` and `Active Backup` modes. Fuel supports Admin interface bonding in LACP mode as an experimental feature. For the `802.3ad (LACP)` bond, you can also select an LACP rate. The values of the LACP rate include: `fast` and `slow`.

7. Create and configure additional network interfaces, if needed.
8. Click *Apply*.

Edit the offloading mode

Fuel assigns the default offloading mode to all network interfaces automatically. You may want to modify this setting to meet your network requirements. The number of available offloading types depends on network hardware and the kernel version that you use.

Fuel automatically detects offloading modes for any physical network interface.

To edit the offloading mode using Fuel web UI:

1. Log in to the Fuel web UI.
2. Click *Nodes*.
3. Select a node.
4. Click *Interface configuration*.
5. Click *Offloading Modes: Default* to disable offloading.

To edit the offloading mode using CLI:

1. Log in to the Fuel Master node CLI.
2. Verify the node ID:

```
fuel nodes
```

3. Download the information about network interfaces:

```
fuel node --node <NODE_ID> --network --download
```

4. Open the `/root/node_<NODE_ID>/interfaces.yaml` file for editing.
5. Disable or leave the default value next to the `state` field:
   - true - enable offloading modes
   - false - disable offloading modes
   - null - default offloading modes
6. Upload the modified file:

```
fuel node --node <NODE_ID> --network --upload
```

Map a logical network to a physical interface

You may want to allocate specific network interfaces to handle different types of network traffic to achieve better performance in your OpenStack environment. Fuel enables you to modify mappings for your entire network, except for the *Admin* network for which you can make changes only during the Fuel Master node installation.

Network interface mapping can be modified after you deploy an OpenStack environment. The `net-config` task updates the networking configuration.

To map a logical network to a physical interface:

1. In the Fuel web UI, click *Nodes*.
2. Select a node.
3. Click *Configure Interfaces*.
4. Drag and drop a logical network to the corresponding physical interface or bond.

Verify network configuration

After you configure network settings, verify your network configuration. Network verification tests connectivity between nodes through configured VLANs on the configured host interfaces. Additionally, Fuel verifies that no external DHCP servers interfere with the OpenStack environment deployment. If network verification fails, the possible reasons may include incorrect network configuration, hardware misconfiguration, such as VLAN tagging is disabled on the switch port, and so on.

You must resolve all errors before you deploy an OpenStack environment.

> ***Note***
>
> Network verification does not test bond network interfaces.

To verify network configuration:

1. In the Fuel web UI, click *Networks*.
2. Click *Connectivity Check*.
3. Click *Verify Networks*.
4. Resolve any network conflicts.
5. Run the network verification again.

Configure disk partitioning

By default, Fuel allocates a balanced amount of disk space for all components that will be installed on the corresponding node. After you assign a role or roles to a node, you can modify the disk partition as needed.

If you modify node roles, Fuel resets the disk partition to default settings.

The following table describes the partition types that you can configure for each node.

Disk partition types

Type of partition	Description
Base system	Comprehensive of swap space, includes operating system and basic software option.
Virtual storage	Storage for virtual instances.
Image storage	Glance stores virtual instance images in this partition.
Cinder	Storage allocated for Cinder.
Ceph and Ceph Journal	Storage allocated for Ceph.
MongoDB	Storage used for Ceilometer information stored in MongoDB.
MySQL	Storage for Zabbix statistics.

To configure disk partitioning:

1. In the Fuel web UI, click *Nodes*.
2. Select a node or nodes.

   > ***Note***
   >
   > You can select multiple nodes of the same role. The nodes with the same role must have identical hardware configuration. You cannot change configuration of multiple nodes with different roles or hardware in one transaction.

3. Click *Disk Configuration*.
4. Click on a partition that you want to modify.
5. In the *Volume Groups*, adjust the partition size using the slider.
6. Alternatively, type the partition size in the corresponding field.

   Fuel adjusts the number you type to satisfy block size boundary requirements. The display adjusts to show the new allocation.
7. Click *Apply*.

Configure the OpenStack environment settings

You can configure security, compute, storage, logging, and other settings in the *Settings* tab. Most of these settings you have already configured in the deployment wizard.

Additionally, you can configure some of the advanced OpenStack settings by editing the corresponding configuration files.

To configure the OpenStack environment settings:

1. In the Fuel web UI, click *Settings*.
2. Select a coresponding tab and edit as required:

OpenStack environment settings

Name	Description
General settings	• **Access** Enables you to modify access permissions for Horizon. By default, Fuel assigns user name, password, and tenant *admin*. • **Repositories** Fuel includes default repositories from which it downloads the packages required to install and update Fuel and OpenStack components. If you do not have an Internet connection, you can set up a local repository and provide the URL to the repository on this page. • **Kernel parameters** Enables you to modify kernel parameters. This field does not set kernel parameters for the Fuel Master node or for nodes that have already been deployed.

	The kernel parameters for OpenStack and Fuel include: • `ttys0=<speed>` Enables serial console for videoless servers. • `console=ttyS0,9600` Enables serial console. • `nofb` Disables Linux framebuffer. • `nomodeset` Disables the video card kernel handling. This parameter may be required for old integrated server video chips. • `intel_iommu and amd_iommu` Enables/disables physical-to-virtual address translation for peripheral devices. Some devices, such as Mellanox cards, require this parameter to be enabled. Other peripheral devices may be incompatible with device virtual address space and may only work with real address space. If you are unable to boot a node or the node has a kernel panic soon after being booted, setting this parameter to "off" may resolve the issue. • `unsupported_hardware` Instructs the operating system to boot even if it does not recognize some of the configured hardware. Failure to set this parameter may result in inability for Linux to boot. This typically happens with the latest CPU models. Because most hardware provides backward compatibility with older versions, setting this kernel parameter may enable the system to boot. However, if no backward compatibility is provided, the system may panic or fail in other ways even with this parameter set.
Security settings	Modify security access settings, such as TLS for OpenStack public checkpoints, HTTPS for Horizon, SSH Public to access Fuel Slave nodes. You can use a self-signed certificate or upload a pre-generated key and certificate.
Compute settings	• **Hypervisor** Enables you to modify the previously selected option. • **Nova quotas** Sets tenant quotas on CPU and memory usage. • **Resume guests state on host boot** Controls whether to preserve the state of virtual instances across reboots.

Storage settings	• **Use qcow format for images** If you select this option, ephemeral volumes will be created as a copy-on-write layer of the base image. If you do not select this option, ephemeral volumes will be full copies of the base image. The default setting is to use copy-on-write for ephemeral volumes. If you select to use Ceph RBD as a storage back end for ephemeral volumes, this setting is ignored. • **Storage Backends** Modify storage options you have previously selected in the deployment wizard. The storage options that you select must match the roles you assign to a node. For example, if you select Ceph as a storage back end, you must configure the appropriate number of nodes with the *Storage - Ceph OSD* role. • **Ceph object replication factor** Determines the minimum number of Ceph OSD nodes that Fuel must deploy. For a production environment, deploy at least three Ceph OSD nodes.
Logging settings	Configure the Puppet and OpenStack debug logging and syslog settings. • **Common** Typically, you do not need to enable debug logging. Enable debug logging to analyze the problems in your system. • **Syslog** Fuel deploys an OpenStack environment with the standard Linux syslog message logging tool. Syslog logs activity of all OpenStack services. By default, `rsyslog` is configured to use the Fuel Master node as a remote syslog server that contains all logs generated on all nodes in the OpenStack environment. If you want to use an external server for `rsyslog`, specify an IP address and port number of the server in the *Syslog* field.
OpenStack services	Select additional OpenStack services to deploy. Some OpenStack services may have additional network and storage requirements. For more information, see: *Configure additional components*.

3. Click *Save Settings*.

Change the DNS and NTP server settings

If the Fuel Master node does not have access to the Internet or if you plan to disable Internet access after deployment, you may want to change the NTP and DNS servers for the nodes and omit routing through the Fuel Master node.

To change the DNS and NTP server settings:

1. In the Fuel web UI, click the guilabel:*Networks* tab.

2. Click *Other*.
3. Type the DNS server IP address or NTP server IP address or FQDN.
4. Click *Save Settings*.

> ***Note***
>
> Fuel does not verify if the specified DNS and NTP services are available. Verify that you specify correct values.

Install additional components

If you want to install additional components, such as the OpenStack Application Catalog service (Murano), the Telemetry service (Ceilometer), the Bare Metal service (Ironic), or the Hadoop cluster (Sahara), you must select a corresponding checkbox in the deployment wizard. However, some components require additional configuration before installation. This section describes the installation process for the OpenStack programs that require additional attention.

Follow the steps described in the corresponding sub-sections of this section.

This section includes the following topics:

Install the OpenStack Bare Metal service

Before you install the OpenStack Bare Metal service (Ironic) verify that your environment meets *Prerequisites for physical machines* and *Bare Metal service limitations* in the *Fuel Installation Guide*.

You install the OpenStack Bare Metal service when you deploy an OpenStack environment. Follow the steps described in *Create a new OpenStack environment* and *Configure your Environment* to configure other components and settings of your OpenStack environment. Then, follow the steps described in this section to configure Ironic in the deployment wizard.

To install the OpenStack Bare Metal service:

1. Configure the new environment as described in *Create a new OpenStack environment*.

   For Ironic you must select:

   - In *Networking Setup - Neutron with VLAN segmentation*.
   - In *Additional Services - Install Ironic*.
2. In the *Dashboard*, click *Add nodes*.
3. In the *Nodes* tab, add nodes with the *Ironic* role.

   We recommend that you assign separate nodes for the Ironic program and do not combine the *Ironic* role with any other roles. However, if you do not have sufficient hardware, you can combine the *Ironic* and *Controller* roles in one node.
4. Add all other nodes required for your environment as described in *Configure your Environment*.
5. Select nodes with the *Ironic* and *Controller* roles.
6. Click *Configure Interfaces*.
7. Assign network interfaces that will be used for *baremetal* network by dragging the *baremetal* network to the required NIC.

   Example:

8. In the *Network* tab, configure the *Baremetal network*.
    - For the OpenStack nodes:
        1. Click *Neutron L2*.
        2. Specify CIDR of the *baremetal* network.
        3. Type the IP range that will be assigned to OpenStack service nodes in the *baremetal* network.
        4. Specify whether to use VLAN tagging or not.
    - For the bare-metal nodes:
        1. Click *Neutron L3*.
        2. Specify an IP range for the nodes on which you will deploy physical machines.

           Assign the IP range from the CIDR you configured for of the *baremetal* network in the previous step.
        3. Assign a gateway IP address.
9. Configure other settings for your OpenStack environment as described in *Configure your Environment*.
10. Proceed to *Configure the Bare Metal service*.

Install the Hadoop cluster service

You install the OpenStack Hadoop cluster service, or Sahara, when you deploy an OpenStack environment. Follow the steps described in *Create a new OpenStack environment* and *Configure your Environment* to configure other components and settings of your OpenStack environment. Then, follow the steps described in this section to configure Ironic in the deployment wizard.

To install the Hadoop cluster service:

1. Create and configure your environment as described in *Create a new OpenStack environment*.
2. On the *Additional Services* page, select *Install Sahara*.
3. Configure and deploy your environment.
4. Proceed to *Configure the Hadoop cluster service*.

Deploy an Openstack Environment

After finishing configuration, you can deploy your OpenStack environment.

This section includes the following topics:

Deploy changes

When you have completed configuring your OpenStack environment as described in *Create a new OpenStack environment* and *Configure your Environment*, you can start the deployment.

> ***Warning***
>
> After you deploy an OpenStack environment, you will not be able to modify many of the OpenStack parameters, such as network topology, disk partitioning, and so on. Verify that you have applied correct settings.

To deploy an OpenStack environment:

1. In the Fuel web UI, select the *Dashboard* tab.
2. Click *Deploy changes*.

   Fuel deploys your OpenStack environment. Depending on the configuration of the environment, the deployment may take from fifteen minutes to an hour.

Interrupt the OpenStack environment deployment

You may need to interrupt the deployment of your OpenStack environment if you have applied incorrect settings.

Depending on the status of deployment, deployment interruption may result in various outcomes.

To interrupt the OpenStack environment deployment:

1. In the Fuel web UI, click the *Dashboard* tab.
2. In the deployment progress bar area, click *Stop*.
3. Click the *Nodes* tab.

   - If no nodes have finished deployment, all nodes are rebooted to the bootstrap state and appear as *Offline*. Fuel resets the environment to the state before you have started the deployment.

     1. Wait until Fuel reboots the nodes. The nodes must appear as *Online*. All settings in all tabs must be unlocked.
     2. Apply your required changes to the OpenStack environment configuration.
     3. Deploy your OpenStack environment.

- If some nodes have already been deployed and have the *Ready* status, Fuel reboots only the nodes that have not finished deployment. Settings remain locked.

    1. Reset the OpenStack environment as described in *Reset an OpenStack environment after deployment*.
    2. Configure your OpenStack environment.
    3. Deploy your OpenStack environment.

Reset an OpenStack environment after deployment

You may want to reset an OpenStack environment after it was successfully deployed, failed to deploy with an error, or you have interrupted the deployment to modify the settings. After you reset an OpenStack environment, Fuel reboots all Fuel Slave nodes and returns them to the *Unallocated* state.

To reset an OpenStack environment:

1. In the Fuel web UI, click the *Dashboard* tab.
2. Click *Reset*.
3. Wait while Fuel reboots the nodes. The nodes must have the status *Online*.
4. Configure and deploy a new environment.

Next Steps

After you successfully deploy your OpenStack environment, you must verify that your environment is operational, as well as configure additional components. After completing the verification, log in to the Horizon dashboard or the OpenStack CLI to operate your environment.

Configure and verify the following:

- If you have installed additional OpenStack components, such as the OpenStack application catalog (Murano), the Telemetry service (Ceilometer), the Bare Metal Service (Ironic), or the Hadoop cluster (Sahara), complete the tasks described in the corresponding sub-section in *Configure additional components*.
- Set up *shell access* to the Fuel Master and target nodes to ensure you can use OpenStack CLI.
- Run tests to ensure that the deployed environment is operational:
    - Run *Verify Networks*. Although you may have already run this test before you deployed the OpenStack environment, you may want to verify the network configuration again. For more information, see: Network Troubleshooting
    - Run the *Verify your OpenStack environment* tests, including "Additional Checks".
    - If you have installed additional components, run the post-deployment tests:
        - *Sahara*
        - *Murano*
    - Check other components:
        - If you have deployed ceph as a storage back end, follow the Verifying the deployment instructions to check the deployment.
- Create a backup of your environment and store it in a safe location.
- To rename, reset, or delete an environment, click the *Actions* tab on the Fuel dashboard and follow the instructions on the screen.
- Manage your environment using the Horizon dashboard or the command-line tools:
    - For more information about using the Horizon dashboard, see the OpenStack User Guide.
    - Create projects/tenants and users; see Managing Projects and Users.
    - If you deployed an OpenStack environment that is integrated with the VMware vSphere, see *Running vCenter*.

Configure additional components

If you have installed additional components, such as the OpenStack application catalog (Murano), the Telemetry service (Ceilometer), the Bare Metal service (Ironic), or the Hadoop cluster (Sahara), you may need to complete post- deployment steps that will ensure your OpenStack environment functions correctly.

If you installed any of these components, complete the steps described in the corresponding sections.

If you installed Ironic, complete the tasks described in the following sections:

- *Configure the Bare Metal service*
- *Prepare a physical machine image*

Configure the Bare Metal service

After you deploy an OpenStack environment as described in *Install the OpenStack Bare Metal service*, you must configure the components required for the Ironic program. Execute all actions described in this section as an OpenStack user with administrative privileges.

To configure the Bare Metal service:

1. Define the memory, CPU, and disk size of physical instances that you will deploy by creating a nova flavor that matches the server hardware on which you plan to run instances. Use the following command:

   ```
   nova flavor-create <flavor_name> <flavor_id> <RAM> <disk_size> <CPU>
   ```

2. Optionally, specify additional parameters using the `nova flavor-key` command.

   Example:

   ```
   nova flavor-key baremetal-flavor set cpu_arch=x86_64
   ```

3. View and remember the list of UUIDs for bootstrap images:

   ```
   glance image-list | grep <bootstrap kernel, ramdisk and squashfs image
   name>
   ```

4. Enroll the nodes on which you plan to boot instances into the OpenStack Bare Metal service.

   ```
   ironic node-create [-c <chassis>] -d <driver> [-i <key=value>] [-p
   <key=value>] [-e <key=value>] [-u <uuid>] [-n <name>]
   ```

Fuel drivers

Driver	Description
`fuel_ssh`	Enables communication between the Fuel Master node and other nodes.
`fuel_ipmitool`	Enables communication through IPMI. Use with the nodes that require IPMI, such as nodes that you use for bare-metal deployments.

`fuel-libvirt`	Ensures operation of virtual ironic instances hosted on `libvirt`.
`fake`	Used for testing Fuel APIs.

Use the values from step 1 and step 2. The following text is an example for the `fuel_ipmitool` driver.

Example:

```
ironic node-create [-n <node name>] [-u <node uuid>] -d fuel_ipmitool
                          -p memory_mb=<node RAM> -p cpu_arch=<node cpu_arch>
                          -p local_gb=<node disk size> -p cpus=<node N of cpus>
                          -i deploy_kernel=<uuid of bootstrap kernel image>
                          -i deploy_ramdisk=<uuid of bootstrap initramfs image>
                          -i deploy_squashfs=<uuid of bootstrap squashfs image>
                          -i ipmi_address=<node IPMI address or hostname>
                          -i ipmi_username=<node IPMI user name>
                          -i ipmi_password=<node IPMI pasword>
```

5. Communicate the node's network interface cards to the Bare Metal service by creating a port with MAC addresses of each network interface:

```
ironic port-create -a <MAC address of the node NIC> -n <node UUID>
```

6. Prepare images that you plan to use to deploy physical machines as described in *Prepare a physical machine image*.

Prepare a physical machine image

If you use default Fuel drivers for Ironic, you must build and upload a physical machine image into Glance, as well as configure the image with specific parameters.

To prepare a physical machine image:

1. Build a physical machine image.

   You can build images for a physical machine using a method of your personal preference. For example, using Disk Image builder (DIB):

```
disk-image-create -a amd64 -p <packages> -o <image_name> <base_image>
```

   Example:

```
disk-image-create -a amd64 -p grub2-common,grub-pc, \
grub-gfxpayload-lists,emacs24-nox,parted baremetal ubuntu-minimal
```

2. Upload the image to Glance using the `glance image-create` command.

   Example:

```
glance image-create --name test --disk-format raw --container-format bare
          --file test [--visibility public]
```

3. Tag the image with the corresponding metadata.

   Example:

```
glance image-update <image-id> --property cpu_arch=x86_64
                               --property hypervisor_type="baremetal"
                               --property fuel_disk_info=DISK_INFO
```

   The `DISK_INFO` value is a structure that describes the partition layout required by the image.

   Example:

```
‘[{"name": "sda", "extra": [], "free_space": 11000, "type": "disk",
"id": "vda", "size": 11000, "volumes": [{"mount": "/", "type":
"partition", "file_system": "ext4", "size": 10000}]}]’
```

> ***Warning***
>
> Only extended file systems are supported!

Seealso

- `glance help image-create`
- *Disk Image Builder Documentation*

If you installed Sahara, complete the tasks described in the following sections:

Configure the Hadoop cluster service

After you deploy your OpenStack environment with the Hadoop cluster service, upload the Sahara image to Glance. You must use a pre-built image for a corresponding Linux distribution. You can build your own image as described in the OpenStack Sahara documentation or use one of the pre-built images available at docs.openstack.org/developer/Sahara If you use a pre-built image, use the corresponding user name provided on the same page.

Sahara supports the following Hadoop platforms:

- Vanilla Apache Hadoop
- Hortonworks Data Platform (HDP)
- Cloudera Hadoop Distribution (CDH)
- Apache Spark
- MapR

Typically, the Sahara images are provided in the `qcow2` format that is compatible with the default OpenStack hypervisor KVM. If you install your environment with VMware vSphere, you must convert the image to an appropriate format before you upload the image to Glance.

To configure the Hadoop cluster service:

1. Download or build an appropriate image for the Hadoop cluster.
2. Upload the image to Glance.
3. Log in to Horizon.
4. Register the image in the Sahara Image Registry:
    1. Click *Project* ⊠ *Data processing* ⊠ *Image Registry*.
    2. Click *Register Image*.
    3. Specify the username appropriate to the image you use.
    4. Specify a correpsonding plugin and version.
    5. Click *Add plugin tags*.
    6. Add appropriate tags.
5. Verify that you have an adequate pool of floating IP addresses available:
    - If you run Neutron networking with **auto_assign_floating_ip** parameter set to `False`, provide a floating IP pool in each Node Group Template.

Verify your OpenStack environment

After you have succefully deployed your OpenStack environment, run the build-in health tests to ensure your environment is fully operational.

Health tests have the following advantages:

- Using post-deployment checks helps you identify potential issues which may impact the health of a deployed system.
- All post-deployment checks provide detailed descriptions about failed operations and tell you which component or components are not working properly.
- Performing these checks manually would consumed a great deal of time, but it only take a few minutes to run the full suite of tests from the Fuel console.
- Aside from verifying that everything is working correctly, the process also determines how quickly your system works.
- Post-deployment checks continue to be useful after you initially deploy your environment. For example, after sizable changes are made in the environment, you can use the checks to determine if any new failure points have been introduced.

This section includes the following topics:

Health checks

Fuel provides capabilities to analyze your OpenStack environment functionality through health checks. Fuel's health checks provide status information about the most commonly used components and the most recently performed actions.

The following table describes the Fuel automated health checks.

Fuel automated health checks

Category	Description	Tests
Sanity tests	Verify overall system operability. If these tests fail, you may need to restart some OpenStack services. Sanity tests will likely be the point on which the success of your deployment pivots, but it is critical to pay close attention to all information collected from theses tests.	Include multiple tests that requests the lists of various OpenStack objects, configurations, and services. If you deploy additional OpenStack services, include tests specific to these services.
Functional tests	Reveal networking, system-requirements, and functionality issues. Functional tests verify basic OpenStack operations in normal conditions.	Include multiple tests that create or launch various OpenStack objects and virtual instances.
High-availability tests	Verify that the high-availability architecture functions correctly. These tests include verification of RabbitMQ availability, HAproxy back ends status and so on.	Include tests that verify that various components, such as RabbitMQ, Pacemaker, the Galera cluster, and son on are highly-available and operational.
Platform services functional tests	Verify basic functionality of the Orchestration service (Heat), Hadoop service (Sahara), Telemetry service (Ceilometer), and Application Catalog service (Murano).	Include multiple tests that verify functionality of additional OpenStack components. Some services, such as Sahara and Murano, require additional preparation before running a test. For more information, see: • *Preparing the Hadoop cluster service for testing* • *About the Hadoop cluster service test* • *Preparing the OpenStack Application Catalog for testing* • *Murano platform test details* • *Overview of the OpenStack Orchestration service platform tests*
Cloud validation tests	Verify that your cloud functions correctly. These tests verify that your nodes have enough free space, as well as various cloud settings, such as log rotation and so on.	Cloud validation tests include: • Check the disk space outages on the Controller and Compute nodes. • Check log rotation configuration on all nodes.

Configuration tests	Verify Fuel configuration. For example, one of the tests verifies that you have changed the default password and suggests to change it if you did not.	Configuration tests include: • Check usage of the default credentials (password) for root user to SSH on the Fuel Master node. If the default password has not been not changed, the test fails with a recommendation to change it. • Check usage of the default credentials for the OpenStack environment. If you use the default values for the admin user, the test fails with a recommendation to change the password and user name for the OpenStack user with the admin role.

Run a health check

We recommend that you run all health tests immediately after you deploy your OpenStack environment, so you can promptly address any issues with your environment configuration.

Each test contains information on its estimated and actual duration. Information about test processing time is based on the tests conducted in our lab. Therefore, actual time for the test to complete may vary for different environments.

After a test is complete, the results appear in the *Status* column. If a test fails, Fuel displays an error message. To assist in troubleshooting, the test scenario is displayed under the failure message and the failed step is highlighted.

To run a health check:

1. In the Fuel web UI, click the *Health Check* tab.
2. Select the tests that you want to run.
3. Click *Run Tests*.

Resolve a problem

If a test fails, there are several ways to investigate the problem. You can search for the information about the problem in the logs of each OpenStack component, as well as in the test logs.

To resolve a health check issue:

1. Verify that all OpenStack services are up and running.
    - In the Fuel web UI:
        1. Click *Health Check*.
        2. Run the Sanity tests.
    - In the Fuel CLI:

1. View the list of services:

```
nova-manage service list
```

2. If any of the services have the *XXX* status, restart these services:

```
service openstack-<service name> restart
```

3. Analyze error messages in *Dashboard*, *Networks*, and other tabs, if any.

   For example, a test may fail for the following reasons:

   - A quota has been exceeded
   - Network configuration is incorrect
   - A general lack of resources, such as memory or disk space.

4. Analyze the log files.

Preparing the OpenStack Application Catalog for testing

Fuel runs the platform tests in the tenant you have previously specified in the *Settings* ☒ *OpenStack Settings*. By default, Fuel selects the *admin* tenant.

You must download and prepare a corresponding image to test Murano. For example, for Linux-based services deployment testing, add a Linux based image to Murano.

To prepare the OpenStack Application Catalog for testing, add a Linux-based image to Murano. You can download the pre-built Murano image or build your own as described in Murano documentation.

> ***Note***
>
> The base operating system of the Murano image does not have to match the base operating system of the OpenStack environment.

Prepare the OpenStack Application Catalog for testing:

1. Download or build a Murano image for testing.
2. Upload the image to the *admin* tenant in the OpenStack Image Service (Glance).
3. Open the *Murano* tab.
4. Navigate to the *Manage* submenu.
5. Click *Images*.
6. Click *Mark Image*.

   The Image registration window displays.

7. Select the Linux image with the Murano Agent.

8. In the *Title* field, set the title for this image.
9. Select the *Generic Linux* type.
10. Click *Mark*.

    Proceed with testing the OpenStack Application Catalog.

Murano platform test details

If you have installed the OpenStack Application Catalog (Murano) in your OpenStack environment, you can test that Murano functions correctly by running the Fuel platform tests.

The following table describes the details of the OpenStack Application Catalog tests.

Murano platform tests

Name	Description	Scenario
Murano environment with the WordPress application deployment	The test verifies that the user can deploy the WordPress application in the Murano environment.	1. Send a request to create an OpenStack environment. 2. Send a request to create a session for the OpenStack environment. 3. Send a request to create MySQL. 4. Send a request to create the Linux-based Apache service. 5. Send a request to create WordPress. 6. Request to deploy a session. 7. Check the environment status. 8. Check the deployment status. 9. Check ports availability. 10. Check the WordPress path. 11. Send a request to delete the OpenStack environment.

Murano environment with the Linux Apache service deployment	The test verifies that the Murano service can create and deploy the Linux Apache service.	1. Verify the Linux image with Murano agent is installed in Glance. 2. Send a request to create an OpenStack environment. 3. Send a request to create a session for the OpenStack environment. 4. Send a request to create the Linux-based Apache service. 5. Request to deploy the session. 6. Check the environment status. 7. Check the deployment status. 8. Send a request to delete the OpenStack environment.

Preparing the Hadoop cluster service for testing

You can run the platform tests to verify that the Hadoop cluster (Sahara) functions correctly. To run the tests, Sahara must be deployed and configured.

You run the tests in the tenant you specified in the *OpenStack Settings* tab during the OpenStack installation. By default, the *admin* tenant is used for the tests.

You must have at least 4096 MB RAM available in the tenant for Sahara platform tests. Otherwise, some tests may fail.

> ***Note***
>
> Sahara uses auto-security groups for opening required ports for each plugin. For more information, see the corresponding plugin documentation.

To prepare Hadoop cluster for testing

1. Download the image with Hadoop for Sahara
2. If you use VMware vSphere as a hypervisor for your OpenStack environment, convert the *qcow2* image format to *vmdk*:
3. Register the image with Sahara:
    1. Upload the image into Glance into the *admin* tenant.
    2. Name the image *sahara*.
    3. In Horizon, navigate to *Projects* ⊠ *Data Processing*.
    4. Switch to the *admin* tenant.

5. Select *Image Registry*.
6. Click *Register Image*.

   The *Image registration* dialog appears.
7. Select the image you have just uploaded.
8. Set username to `ubuntu`.
9. Select the tags: `plugin=vanilla` and `version=<version>`.
10. Click *Add plugin tags*.
11. Click *Done*.

4. Proceed with testing the Hadoop cluster.

About the Hadoop cluster service test

The Hadoop cluster service test verifies that Sahara can launch a Hadoop cluster using the Vanilla plugin.

The following table describes the details of the Hadoop cluster tests.

Sahara platform tests

Name	Description	Scenario

Test that Sahara can launch a Hadoop cluster using the Vanilla plugin.	The test verifies successful launch of the Hadoop cluster.	1. Log in to the OpenStack dashboard.
		2. Register an image:
		  1. Select *Project* ⊠ *Data Processing* ⊠ *Image Registry*.
		  2. Click *Register Image*.
		  3. In the *Image* field, select an image.
		  4. Specify the `User Name` value for the selected OS.
		  5. Set the following values:
		  • `Plugin=vanilla`
		  • `Version=2.6.0`
		  6. Click *Add plugin tags*.
		  7. Click *Done*.

		1. Create a master node group template: 1. Select *Project* ☒ *Data Processing* ☒ *Node Group Templates*. 2. Click *Create Template*. 3. In the *Create Node Group template* dialog, set the following values: • `Plugin name=Vanilla Apache Hadoop` • `Hadoop version=2.6.0` 4. Click *Create* to proceed. 5. In the second *Create Node Group template* dialog, set the following values: • `Template Name=vanilla2-master` • `OpenStack Flavor=m1.small` • `Floating IP pool=(external network)` 6. In the *Process* section, select: • `namenode` • `secondarynamenode` • `resourcemanager` • `historyserver` • `oozie` 7. Click *Create*.

		1. Create a worker node group template: 1. Select :menuselection: *Project* --> *Data Processing* --> *Node Group Templates*. 2. Click *Create Template*. 3. In the *Create Node Group template* dialog, set the following values: • `Plugin name=Vanilla Apache Hadoop` • `Hadoop version=2.6.0` 4. Click *Create* to proceed. 5. In the second *Create Node Group template* dialog, set the following values: • `Template Name=vanilla2-worker` • `OpenStack Flavor=m1.small` • `Floating IP pool=(external network)` 6. In the *Process* section, select: • `datanode` • `nodemanager` 7. Click *Create*.

		1. Create a cluster template: 1. Select *Project* ⊠ *Data Processing* ⊠ *Cluster Templates*. 2. Click *Create Template*. 3. In the *Create Cluster Template* dialog, set the following values: • `Plugin name=Vanilla Apache Hadoop` • `Hadoop version=2.6.0` 4. Click *Create*. 5. In the second *Create Cluster Template* dialog, set the following values: • In the *Details* tab, specify `Template Name=vanilla2-template`. • In the *Node Groups* tab, specify `vanilla2-master` and `vanilla2-worker`. • In the *HDFS Parameters* tab, specify `dfs.replication=1`. 6. Click *Create*.
		1. Launch the cluster: 1. Select *Project* --> *Data Processing* --> *Clusters*. 2. Click *Launch Cluster*. 3. In the *Launch Cluster* dialog, set the following values: • `Plugin name=Vanilla Apache Hadoop` • `Hadoop version=2.6.0` 4. Click *Create* to proceed. 5. In the second *Launch Cluster* dialog, set :guilabel:`Cluster Name=vanilla2-cluster`. 6. Click *Create*. 7. Wait until the cluster has the `Active` status.

		1. Delete the cluster: 1. In the *Clusters* page, select the `vanilla2-cluster` cluster. 2. Click *Delete Cluster*. 2. Delete the templates: 1. Select *Project* ☒ *Data Processing* ☒ *Cluster Templates*. 2. Select the *vanilla2-template* template. 3. Click *Delete Templates*. 4. Select *Project --> Data Processing --> Node Group Templates*. 5. Select *vanilla2-master* and *vanilla2-worker* templates. 6. Click *Delete Templates*.

Overview of the OpenStack Orchestration service platform tests

If you have installed the OpenStack Orchestration service (Heat) in your OpenStack environment, Fuel provides the automatic tests to verify its functionality.

The following table describes the details of the Heat tests.

Heat platform tests

Name	Description	Scenario

Test basic stack operations.	The test verifies that the Heat service can create, update, and delete a stack, as well as shows details of the stack and its resources, events, and template.	1. Create a stack. 2. Wait for the stack status to change to `CREATE_COMPLETE`. 3. Get details of the created stack by its name. 4. Get the list of resources for the created stack. 5. Get details of the stack resources. 6. Get the list of events for the created stack. 7. Get the details of the stack event. 8. Update the stack. 9. Wait for the stack to update. 10. Get the stack template details. 11. Get the list of resources for the updated stack. 12. Delete the stack. 13. Wait for the stack to delete.

Test the stack autoscaling.	The test verifies that the Heat service can scale the stack capacity up and down automatically according to the changes in the configuration. Image with cfntools package should be imported.	1. Create a flavor. 2. Create a keypair. 3. Save the generated private key to a file on the controller node. 4. Create a security group. 5. Create a stack. 6. Wait for the stack status to change to 'CREATE_COMPLETE'. 7. Create a floating IP. 8. Assign the floating IP to the instance of the stack. 9. Wait for the `cloud_init` procedure to complete on the instance. 10. Load CPU of the instance to initiate the stack scaling up. 11. Wait for the second instance to launch. 12. Release CPU of the instance to initiate the stack scaling down. 13. Wait for the second instance to be terminated. 14. Delete the file with the private key on the controller node. 15. Delete the stack. 16. Wait for the stack to delete.
Test the stack rollback functionality.	The test verifies that the Heat service can rollback the stack if its creation failed.	1. Start stack creation with rollback enabled. 2. Verify that the stack appears with status `CREATE_IN_PROGRESS`. 3. Wait for the stack to be deleted as a result of the rollback after the expiration of the timeout defined in the `WaitHandle` resource of the stack. 4. Verify that the instance of the stack has been deleted.

Test advanced stack operations.	The test verifies that the Heat service can suspend and resume the stack.	1. Create a stack. 2. Wait until the stack status changes to `CREATE_COMPLETE`. 3. Call the stack suspend action. 4. Wait until the stack status changes to `SUSPEND_COMPLETE`. 5. Call the stack resume action. 6. Wait until the stack status changes to `RESUME_COMPLETE`. 7. Call the stack check action. 8. Wail until the stack status changes to `CHECK_COMPLETE`. 9. Delete the stack. 10. Wait for the stack to be deleted.

Role operations

Role object

Beginning with Fuel 6.1, you can create, update or delete roles using *Nailgun* REST API and Fuel Client.

For Fuel CLI command reference, see *Role operations* section.

This section provides the Controller role example:

```
id: 9
meta:
  conflicts:
  - compute
  description: The controller initiates orchestration activities and provides an external
    API.  Other components like Glance (image storage), Keystone (identity management),
    Horizon (OpenStack dashboard) and Nova-Scheduler are installed on the controller
    as well.
  has_primary: true
  limits:
    min: 1
    overrides:
    - condition: cluster:mode == 'multinode'
      max: 1
      message: Multi-node environment can not have more than one controller node.
    - condition: cluster:mode == 'ha_compact'
      message: At least 3 controller nodes are recommended for HA deployment.
      recommended: 3
  name: Controller
  update_required:
  - compute
  - cinder
name: controller
volumes_roles_mapping:
    - allocate_size: min
      id: os
    - allocate_size: all
      id: image
```

The following fields are mandatory:

```
name: controller
meta:
  name: Controller
  description: Description goes here

# at least one volume is required
volumes_roles_mapping:
```

```
  - allocate_size: min
    id: os
```

Primary behaviour for node can be enabled with `has_primary: true` option. If this option is set to `during orchestration`, you will be able to assign separate tasks for primary-controller and controller.

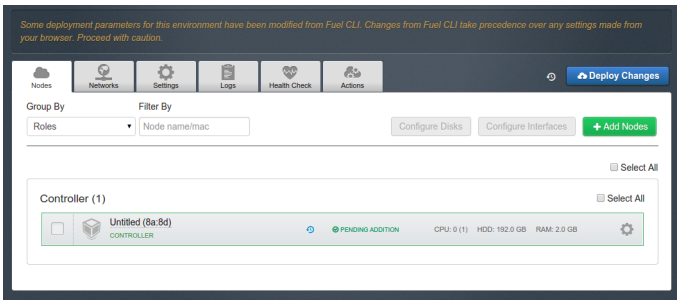
Using Fuel CLI

Fuel CLI is a powerful tool that allows you to:

- Operate with environments using the text console only.
- Modify directly the internal data that you can't modify via the Fuel Web UI.
- Avoid data verifications done by the Fuel Web UI logic.

Fuel CLI may break your environment if not used carefully.

It is necessary to understand that any modifications done using Fuel CLI take precedence over the settings made from the Fuel Web UI. Fuel shows a special message to inform you:



Basic usage

Fuel CLI has the following usage pattern:

```
fuel [global optional args] <namespace> [action] <optional args>
```

Example:

```
fuel --env-id=1 node set --node-id=1,4,5 --role=controller
```

where `--env-id=1` is a global optional argument pointing to the specific environment, `node` - is a namespace for all node control functions, `set` is an action that assigns specific nodes to some environments in certain roles.

To get the list of all global optional arguments and namespaces, run:

```
fuel --help
```

To get the list of actions and optional arguments for a namespace, run:

```
fuel <namespace> --help
```

CLI commands reference

Interpretation of acronyms in CLI commands

CLI commands contain a number of acronyms. For better understanding of those, see the example command output below.

> ***Note***
>
> *Nailgun* populates the database with hardware configuration information about all the managed *nodes* it discovers as well as the configuration and status of each node.

The `fuel node list` command is used on the Fuel Master node to list out the current information about the nodes for the environment:

```
[root@fuel ~]# fuel nodes

id | status   | name              | cluster | ip        | mac               | ...
---|----------|-------------------|---------|-----------|-------------------| ...
4  | ready    | Untitled (b0:77) | 1       | 10.20.0.6 | 56:40:fa:cc:cf:45 | ...
1  | ready    | Untitled (ca:9a) | 1       | 10.20.0.4 | ca:03:e6:b1:13:46 | ...
3  | ready    | Untitled (0e:64) | 2       | 10.20.0.7 | 26:1f:eb:91:d8:49 | ...
2  | ready    | Untitled (c1:ef) | 2       | 10.20.0.3 | 22:2a:45:36:5d:42 | ...
5  | discover | Untitled (e1:c4) | None    | 10.20.0.5 | 08:00:27:1a:e1:c4 | ...


id | status   | name              |...| roles      | pending_roles | online | group_id
---|----------|-------------------|...|------------|---------------|--------|---------
4  | ready    | Untitled (b0:77) |...| compute    |               | True   | 1
1  | ready    | Untitled (ca:9a) |...| controller |               | True   | 1
3  | ready    | Untitled (0e:64) |...| compute    |               | True   | 2
2  | ready    | Untitled (c1:ef) |...| controller |               | True   | 2
5  | discover | Untitled (e1:c4) |...|            |               | True   | None
```

The meaning of these fields is:

id: The node identifier, assigned incrementally when the node is first discovered (when the Fuel agent sends its first request to the Fuel Master node).

This ID is the Primary Key for this record in the database; it is unique and is never reassigned; when you delete a node from the environment, that node's ID is deleted; the next node added to the environment is assigned a new ID that is higher than the highest-numbered ID in the database.

status: Current state of the node:

- **ready:** Node is deployed and provisioned, ready to use
- **discover:** Node is not deployed and not provisioned
- **provisioning:** Node is in the process of being provisioned (operating system is being installed)
- **provisioned:** Node is provisioned but not deployed
- **deploying:** Node is being deployed (OpenStack is being installed and configured)
- **error:** Deployment/provisiong of the node has failed

name: Name of the node as displayed on the screen when you assign roles. By default, this is "Untitled" with the final digits of the MAC address used by the Admin interface for that node. You can double-click on that title to change the name.

cluster: ID of the environment to which the node is assigned.

ip: IP address of the admin interface, which is the IP address for the default route.

mac: MAC address of the admin interface, determined the same way as the IP address.

roles: *Role(s)* that the node has; populated only after deployment.

The following two columns appear at the right end of this display; they are not shown here:

pending_roles: Before deployment, lists the roles that have been assigned to this node. When deployment is complete, the contents of this field are moved to the **roles** column

For Release 6.x and later, this field can also contain the **primary** value to indicate that this node is the Primary Controller node. The **primary** value is persisted in the database through the use of the **has_primary** field in the *openstack.yaml* file.

online: Status of the node:

- **False:** Node is offline.
- **True:** Node is available via the Fuel admin network.

group_id: The group node identifier. When you assign roles to your target nodes, Fuel tries to automatically determine the node's group based on the DHCP address.

Release

For acronyms meaning, see *What stands for acronyms in CLI commands*.

Get list of all available releases:

```
fuel release
```

or short version

```
fuel rel
```

for specific release

```
fuel rel --rel <release_number>
```

Version

To get all the details on the Fuel environment installed, run the following command:

```
fuel fuel-version
```

> ***Warning***
>
> The argument `--fuel-version` will be deprecated since the Fuel 7.0 release. Please use **fuel-version** command instead.

Networks configuration

Download a network configuration for a specific environment:

```
fuel --env <ENV_ID> network --download --dir <PATH>
```

where:

- `<ENV_ID>` - an environment ID
- `<PATH>` - a path to directory

For example, download the network configuration for the environment with ID `1` to the current directory:

```
fuel --env 1 network --download
```

Upload a network configuration for a specific environment:

```
fuel --env <ENV_ID> network --upload --dir <PATH>
```

For example, upload the network configuration for the environment with ID `1` from the current directory:

```
fuel --env 1 network --upload
```

> ***Note***
>
> The **fuel network** command can update a configuration for all networks in an environment and Neutron parameters, but it does not update VIPs and network templates. You have to update VIPs and network templates additionally using corresponding Fuel CLI commands.

Verify a network configuration for a specific environment:

```
fuel --env <ENV_ID> network --verify --dir <PATH>
```

For example, verify the network configuration for the environment with ID 1 from the current directory:

```
fuel --env 1 network --verify
```

> ***Note***
>
> Verification does not work for multiple cluster networks, when an environment has more than one node group.

To see interaction with Nailgun API, run the commands with the *--debug* option:

```
fuel --env <ENV_ID> network --download --dir <PATH> --debug
fuel --env <ENV_ID> network --upload --dir <PATH> --debug
fuel --env <ENV_ID> network --verify --dir <PATH> --debug
```

Environment

For acronyms meaning, see *What stands for acronyms in CLI commands*.

To list environments:

```
fuel env
```

To create an environment, run the following command using `--name` and `--rel` (release) options:

```
fuel env create --name <env_name> --rel <release_number>
```

By default it creates environment in `multinode` mode, and `nova` network mode. To specify other modes, you can add optional arguments; for example:

```
fuel env create --name <env_name> --rel <release_number> \
--mode ha --network-mode neutron --net-segment-type vlan
```

Use the `set` action to change the name, mode, or network mode for the environment; for example:

```
fuel --env <env_id> env set --name <NewEmvName> --mode ha_compact
```

To delete the environment:

```
fuel --env <env_id> env delete
```

To update the Mirantis OpenStack environment to a newer version (available since Fuel 5.1):

```
fuel env --update --env <env_id> --rel <release_number>
```

To roll back a failed update, use this same command but modify the release ID.

Node

For acronyms meaning, see *What stands for acronyms in CLI commands*.

To list all available nodes run:

```
fuel node list
```

and filter them by environment:

```
fuel --env-id <env_id> node list
```

Assign some nodes to environment with with specific roles

```
fuel node set --node <node_id> --role controller --env <env_id>
fuel node set --node <node1_id>,<node2_id>,<node3_id> \
--role compute,cinder --env <env_id>
```

Remove some nodes from environment

```
fuel node remove --node <node1_id>,<node2_id> --env <env_id>
```

Also you can do it without `--env` or `--node` to remove some nodes without knowing their environment and remove all nodes of some environment respectively.

```
fuel node remove --node <node1_id>,<node2_id>
fuel node remove --env <env_id>
```

Delete nodes from Fuel DB.

- Remove offline nodes:

```
fuel node --node-id <id> --delete-from-db
fuel node --node-id <id1> <id2> --delete-from-db
```

- Remove nodes with any status (`--force` option forces deletion of nodes regardless of their state):

```
fuel node --node-id <id> --delete-from-db --force
```

Node group

For acronyms meaning, see *What stands for acronyms in CLI commands*.

To list all available node groups:

```
fuel nodegroup
```

and filter them by environment:

```
fuel --env <env_id> nodegroup
```

Create a new node group

```
fuel --env <env_id> nodegroup --create --name "group 1"
```

Delete the specified node groups

```
fuel --env <env_id> nodegroup --delete --group <group_id>
fuel --env <env_id> nodegroup --delete --group <group1_id>,<group2_id>,<group3_id>
```

Assign nodes to the specified node group:

```
fuel --env <env_id> nodegroup --assign --node <node_id> --group <group_id>
fuel --env <env_id> nodegroup --assign --node <node1_id>,<node2_id>,<node3_id> --group <gro
```

Network Template

To upload a network template, run the following command on the Fuel Master node:

```
fuel --env <ENV_ID> network-template --upload --dir <PATH>
```

where:

- <ENV_ID> - an ID of your OpenStack environment that you can get by running the **fuel environment** command

- <PATH> - a path to a directory where your template is located

For example:

```
fuel --env 1 network-template --upload --dir /home/stack/
```

Download a network template to the current directory:

```
fuel --env <ENV_ID> network-template --download
```

For example:

```
fuel --env 1 network-template --download
```

Delete an existing network template:

```
fuel --env <ENV_ID> network-template --delete
```

For example:

```
fuel --env 1 network-template --delete
```

Network Group

List all available network groups:

```
fuel network-group
```

List network groups in a particular node group:

```
fuel network-group --node-group <GROUP_ID>
```

For example:

```
fuel network-group --node-group 1
```

Create a new network group:

```
fuel network-group --create --node-group <NODE_GROUP_ID> --name <NAME> \
     --release <RELEASE_ID> --vlan <VLAN_ID> --cidr <CIDR> --gateway <GATEWAY_IP> \
     --meta <META_INFO>
```

where:

- <NODE_GROUP_ID> - an ID of a node group
- <NAME> - a name of a new network group
- <RELEASE_ID> - a release ID this network group belongs to
- <VLAN_ID> - a VLAN of a network
- <CIDR> - a CIDR of a network
- <GATEWAY_IP> - a gateway of a network
- <META_INFO> - meta information in JSON format

For example:

```
fuel network-group --create --node-group 1 --name "new network" \
             --release 2 --vlan 100 --cidr 10.0.0.0/24

fuel network-group --create --node-group 2 --name "new network" \
            --release 2 --vlan 100 --cidr 10.0.0.0/24 --gateway 10.0.0.1 \
            --meta 'meta information in JSON format'
```

Set parameters for the specified network group:

```
fuel network-group --set --network <ID> --<PARAMETER> <NEW_VALUE>
```

where:

- <ID> - an ID of a network group
- <PARAMETER> - a parameter you want to set or update. See the `fuel network-group --create` command for the list of parameters.
- <NEW_VALUE> - a new value for the specified parameter

For example:

```
fuel network-group --set --network 1 --name new_name
```

Delete network groups:

```
fuel network-group --delete --network <GROUP_ID>
```

For example:

```
fuel network-group --delete --network 1
```

You can also delete multiple groups:

```
fuel network-group --delete --network 2,3,4
```

Roles operations

CLI basically implements standard CRUD for operating on a role.

- List a role:

```
fuel role --rel 2

 name          | id
 --------------|---
 controller    | 9
 compute       | 10
 cinder        | 11
 cinder-vmware | 12
 ceph-osd      | 13
 mongo         | 14
 zabbix-server | 15
 base-os       | 16
```

- Create a new role.
    - In this example, we first create a swift role in `swift.yaml`:

```
meta:
  description: Installs swift server.
  has_primary: true # we need primary-swift and swift during orchestration
  name: Swift
name: swift
volumes_roles_mapping:
  - allocate_size: min
    id: os
```

    - Then use `--create` flag to proceed. When created, you can start using a new role for your own tasks:

```
 fuel role --rel <2> --create --file <swift.yaml>

 fuel role --rel <2>
```

```
 name          | id
--------
 swift         | 17
```

- Update role data:

```
fuel role --rel <2> --update --file <swift.yaml>
```

- Delete the role:

```
fuel role --rel <2> --delete --role <swift>
```

Configuring

Configuration of the environment or some node is universal and done in three stages:

1. Download current or default configuration. Works for (`network`, `settings`, `node --disk`, `node --network`). Operations with `deployment` and `provisioning` can be node specific. (e.g. `fuel --env 1 deployment --node-id=1,2`)

Example:

```
fuel --env 1 network download
fuel --env 1 settings download
fuel --env 1 deployment default
fuel --env 1 provisioning download
fuel node --node-id 2 --disk --download
```

2. Modify the downloaded *YAML* files with your favorite text editor.
3. Upload files to nailgun server

After redeploying your environment with the new configuration, you should create a new *backup* of the Fuel Master node. You may also want to delete the YAML files since you can easily regenerate them at any time. Some of the generated YAML files contain unencrypted passwords whose presence on disk may constitute a security threat.

Example:

```
fuel --env 1 provisioning upload
fuel node --node-id 2 --disk --upload
```

Note

To protect yourself when using the Fuel CLI to modify configurations, note the following:

- *Back up* all of your configurations before you begin any modifications.
- If you remove something from a configuration file, be sure you do not need it; Fuel CLI overwrites the old data with the new rather than merging new data with existing data.
- If you upload any changes for provisioning or deployment operations, you freeze the configuration for the entire environment; any changes you later make to the networks, cluster settings, or disk configurations using the Fuel Web UI are not implemented. To modify such parameters, you must edit the appropriate section of each node's configuration and apply the changes with Fuel CLI.

Changing the configuration of Nova, Neutron, and Keystone

Using CLI, you can override the hardcoded, or provided by Nailgun, configuration values, as well as introduce new configuration options for OpenStack services.

You can change the Nova, Neutron, and Keystone configuration for:

- A single node
- All nodes with a certain role
- An environment

You can change the configuration before or after the environment deployment.

The services the configuration of which you change restart automatically after applying the changes.

The `override_resources` Puppet resource applies changes to the existing configuration resources and creates the new ones that were not previously defined.

To change the configuration of OpenStack Services

1. Log in to the Fuel Master node.
2. Edit the YAML file with the configuration options of the services that you are going to change. Example:

```
configuration:
  nova_config:
    DEFAULT/debug:
      value: True
    DEFAULT/amqp_durable_queues:
      value: False
  keystone_config:
    DEFAULT/default_publisher_id:
      ensure: absent
```

```
    DEFAULT/crypt_strength:
      value: 6000
```

3. Upload the YAML file:

    - To upload the changes for the environment:

      ```
      fuel openstack-config --env 1 --upload file.yaml
      ```

    - To upload the changes for all nodes with a role:

      ```
      fuel openstack-config --env 1 --role compute --upload file.yaml
      ```

    - To upload the changes for certain nodes:

      ```
      fuel openstack-config --env 1 --node 1,2,3 --upload file.yaml
      ```

4. Execute the changes:

    - To execute the changes for the environment:

      ```
      fuel openstack-config --env 1 --execute
      ```

    - To execute the changes for all nodes with a role:

      ```
      fuel openstack-config --env 1 --role compute --execute
      ```

    - To execute the changes for certain nodes:

      ```
      fuel openstack-config --env 1 --node 1,2,3 --execute
      ```

The services will restart automatically.

Additional commands

- List the configuration changes history:

  ```
  fuel openstack-config --env 1 --list
  ```

  This command returns a list of configuration changes, each of them with a respective ID record.

- Download a previously uploaded YAML file with the configuration changes:

```
fuel openstack-config --id 1 --download
```

The `id` parameter is the record number from the changes history that you can get with the **fuel openstack-config --env 1 --list** command.

Workflow of the configuration change override

The `override_resources` Puppet resource overrides the already existing resources and creates the previously not defined resources.

> ***Note***
>
> `override_resources` must always be used as the first resource in manifests.

Example:

```
keystone_config {
  'DEFAULT/debug': {value => True}
}
override_resource {'keystone_config':
  data => {
     'DEFAULT/debug': {'value' => False},
     'DEFAULT/max_param_size': {'value' => 128}
  }
}
```

The Nova, Keystone, and Neutron top-level granular tasks use `override_resources`. The new parameter hash used in the Puppet resources is passed to `override_resources` from hiera.

The three following hiera files cover the hierarchical configuration overrides:

- `/etc/hiera/override/config/%{::fqdn}`
- `/etc/hiera/override/config/role`
- `/etc/hiera/override/config/cluster`

Hiera delivers the hierarchical structure of data.

The top-level granular tasks used to override the configuration have the `refresh_on` parameter.

Example:

```
- id: keystone
  type: puppet
```

```
  groups: [primary-controller, controller]
  required_for: [openstack-controller]
  requires: [openstack-haproxy, database, rabbitmq]
  refresh_on: [keystone_config]
  parameters:
    puppet_manifest:
       /etc/puppet/modules/osnailyfacter/modular/keystone/keystone.pp
    puppet_modules: /etc/puppet/modules
    timeout: 3600
  test_pre:
    cmd: ruby
       /etc/puppet/modules/osnailyfacter/modular/keystone/keystone_pre.rb
  test_post:
    cmd: ruby
       /etc/puppet/modules/osnailyfacter/modular/keystone/keystone_post.rb
```

Nailgun uses the `refresh_on` parameter to run the respective task when user changes the OpenStack configuration.

Deployment

For acronyms meaning, see *What stands for acronyms in CLI commands*.

You can deploy environment changes with:

```
fuel --env <env_id> deploy-changes
```

Also, you can deploy and provision only some nodes like this

```
fuel node --provision --node <node1_id>,<node2_id>
fuel node --deploy --node <node1_id>,<node2_id>
```

Change and Set Fuel password

You can change the Fuel Master Node password with either of the following:

```
fuel user --change-password --new-pass=<new_password>
```

Note that **change-password** option can also be used without preceding hyphens.

You can use flags to provide username and password to other fuel CLI commands:

```
--user=admin --password=test
```

See *Fuel Access Control* for more information about Fuel authentication.

Add network ranges

To add network ranges, edit the network configuration file: add the IP network range to `ip_ranges` and change `notation` from `cidr` to `ip_ranges`.

Step-by-step:

1. On the Fuel Master node, download the network configuration file:

   ```
   fuel network --env <ENV-ID> -d
   ```

   where <ENV_ID> is the ID of the environment (a number) that you can get by issuing the `fuel env` command.

   For example:

   ```
   fuel network --env 1 -d
   ```

2. Open the downloaded **/root/network_<ENV-ID>.yaml** file for editing.
3. Add your list of IP network ranges under the `ip_ranges` parameter.

   Sample:

   ```
   ip_ranges:
   - - 192.168.0.1
     - 192.168.0.90
   - - 192.168.0.100
     - 192.168.0.254
   ```

4. In the same network configuration file, change `notation: cidr` to `notation: ip_ranges`.

   Sample:

   ```
   meta:
     cidr: 192.168.0.0/24
     configurable: true
     map_priority: 2
     name: management
     notation: ip_ranges
     render_addr_mask: internal
   ```

5. Upload the edited network configuration file:

   ```
   fuel network --env <ENV-ID> -u
   ```

Fuel Plugins CLI

- To install a Fuel plugin:

1. Select from the following options:

- If you install a Fuel plugin from an *.fp* package, type:

```
fuel plugins --install <fuel-plugin-file>
```

- If you install a Fuel plugin from an *.rpm* package, select from the following options:
  - Using `yum install`:
    1. Install the Fuel plugin:

```
yum install <fuel-plugin-file>
```

    2. Register the plugin in *Nailgun*:

```
fuel plugins --register <fuel-plugin-name>==<fuel-plugin-version>
```

  - Using the same command you used to install a Fuel plugin from the *.fp* package:

```
fuel plugins --install <fuel-plugin-file>
```

2. View the list of installed plugins:

```
fuel plugins --list

id |    name                    | version  | package_version
---|----------------------------|----------|----------------
1  | <fuel-plugin-name>         | 1.0.0    | 2.0.0
```

- To remove a plugin, type:

```
fuel plugins --remove <fuel-plugin-name>==<fuel-plugin-version>
```

- To upgrade a Fuel RPM plugin, type:

```
fuel plugins --update <fuel-plugin-file>
```

> ***Note***
>
> Upgradess are *not* supported for:
>
> - fp plugins
> - major versions of RPM plugins
>
>   For example, you can only upgrade from version 1.0.0 to 1.0.1.

To see the list of all available options, use `fuel plugins --help` command.

Rollback

You can use the rollback feature to return a node to its original state (e.g. the state before the node failed). This can be used to revert changes during a failed upgrade or other node malfunction.

The rollback is done in two major steps:

1. Partition preservation -- prevent the node redeployment process from deleting data on the partition. This way you will not have to manually back up and restore the data to perform the rollback.
2. Node reinstallation -- restore the node to its original state.

Partition preservation

With partition preservation you can keep any type of data that meets the following criteria:

- The data is stored on a dedicated partition.
- The partition is not a root partition. (The root partition is always erased during deployment).

The following data types are a good example of what can be preserved:

- Ceph data
- Swift data
- Nova instances cache
- Database, custom partition types

> ***Note***
>
> Do not change the partition size as this will make the the rollback impossible.

1. On the Fuel Master node, dump the disks information using this *fuel CLI* command:

   ```
   fuel node --node-id <NODE_ID> --disk --download
   ```

   where <NODE_ID> points to a specific node identified by its ID (a number) that you can get by issuing the `fuel nodes` command.

   For example:

   ```
   fuel node --node-id 1 --disk --download
   ```

2. Edit the `/root/node_1/disks.yaml` file to enable partition preservation by using the `keep_data: true` flag. Also note that all partitions with the same name need to have the same flag value.

   Example of the flag in a `disks.yaml` file:

```
- extra:
   - disk/by-id/scsi-SATA_QEMU_HARDDISK_QM00001
   - disk/by-id/ata-QEMU_HARDDISK_QM00001
   id: disk/by-path/pci-0000:00:01.1-scsi-0:0:0:0
   name: sdc
   size: 101836
   volumes:
   - name: mysql
     size: 101836
     keep_data: true
```

3. Upload the modified file:

```
fuel node --node-id <NODE_ID> --disk --upload
```

where <NODE_ID> points to a specific node identified by its ID (a number) that you can get by issuing the `fuel nodes` command.

For example:

```
fuel node --node-id 1 --disk --upload
```

Node reinstallation

1. On the Fuel Master node, issue the following command to reprovision the node:

```
fuel node --node-id <NODE_ID> --provision
```

where <NODE_ID> points to a specific node identified by its ID (a number) that you can get by issuing the `fuel nodes` command.

For example:

```
fuel node --node-id 1 --provision
```

2. Then issue the following command to redeploy the node:

```
fuel node --node-id <NODE_ID> --deploy
```

where <NODE_ID> points to a specific node identified by its ID (a number) that you can get by issuing the `fuel nodes` command.

Virt role reinstallation

Follow the steps below to reinstall the virt role if you have the *Reduced Footprint feature* enabled.

1. On the Fuel Master node, dump the disks information using the *fuel CLI* command:

   ```
   fuel node --node-id <NODE_ID> --disk --download
   ```

   where <NODE_ID> points to the node with virt role identified by its ID (a number) that you can get by issuing the `fuel nodes` command. For example:

   ```
   fuel node --node-id 1 --disk --download
   ```

2. Edit the */root/node_1/disks.yaml* file to enable the partition preservation of the volume with `vm` name using the `keep_data: true` flag of the corresponding volumes. Note that all partitions with the same name need to have the same flag value.

   Example of the flag in a *disks.yaml* file:

   ```
   - extra:
     - disk/by-id/wwn-0x5000c5007a287855
     - disk/by-id/scsi-SATA_ST2000DM001-1ER_Z4Z1WH2V
     - disk/by-id/ata-ST2000DM001-1ER164_Z4Z1WH2V
     id: disk/by-path/pci-0000:00:1f.2-scsi-0:0:0:0
     name: sda
     size: 1907037
     volumes:
     - keep_data: false
       name: os
       size: 67584
     - keep_data: false
       name: cinder
       size: 919726
     - keep_data: true
       name: vm
       size: 919727
   ```

3. Upload the modified file:

   ```
   fuel node --node-id <NODE_ID> --disk --upload
   ```

   where <NODE_ID> points to a specific node identified by its ID (a number) that you can get by issuing the `fuel nodes` command.

   For example:

   ```
   fuel node --node-id 1 --disk --upload
   ```

4. On the Fuel Master node, reprovision the node:

```
fuel node --node-id <NODE_ID> --provision
```

where <NODE_ID> points to a specific node identified by its ID (a number) that you can get by issuing the `fuel nodes` command.

For example:

```
fuel node --node-id 1 --provision
```

5. Provision the bare-metal node with the virtual role and spawn virtual machines:

```
fuel2 env spawn-vms <CLUSTER_ID>
```

For example:

```
fuel2 env spawn-vms 1
```

6. Redeploy the spawned node:

```
fuel node --node-id <NODE_ID> --deploy
```

where <NODE_ID> points to a specific node identified by its ID (a number) that you can get by issuing the `fuel nodes` command.

Create diagnostic snapshot using shotgun

Shotgun is a tool that you can use to generate diagnostic snapshots for Fuel. Although, Fuel API for diagnostic snapshots provides similar functionality, you may prefer to use Shotgun due to the following limitations of Fuel API:

- When the size of log files is too big, Fuel drops a timeout exceptions.
- When you use Fuel API, you may run out of space in the */var/* partition.

Therefore, use Shotgun from the Fuel Master node directly and fetch the default configuration from the Fuel Client.

Shotgun stores temporary snapshots in `/var/www/nailgun/dump/fuel-snapshot`. A symlink to the last compressed snapshot is located in `/var/www/nailgun/dump/last`.

With Shotgun you can use standard commands, such as **dir**, **command**, and **file**:

```
- command: brctl show
  to_file: brctl_show.txt
  type: command
- path: /etc/sysconfig/network-scripts
  type: dir
```

To use Shotgun:

1. Install Shotgun on the Fuel Master node:

   ```
   yum install -y shotgun
   ```

2. Fetch the default configuration:

   ```
   fuel snapshot --conf > dump_conf.yaml
   ```

3. Provide the configuration to Shotgun and execute it:

   ```
   shotgun -c dump_conf.yaml
   ```

VMware integration

Deploying an OpenStack environment with VMware vCenter as a hypervisor

Before you provision your environment with VMware vCenter as a hypervisor, verify that you have completed the following:

- Installed and configure the Fuel Master node as described in the Fuel Installation Guide.
- Installed and configured VMware vSphere.
- Verified that VMware vCenter is accessible from the Fuel Master node.
- Verify the login credentials.

To deploy an OpenStack environment that is integrated with the VMware vSphere, click on the "New OpenStack environment" icon to launch the wizard that creates a new OpenStack environment.

Seealso

- VMware vSphere in the OpenStack documentation.

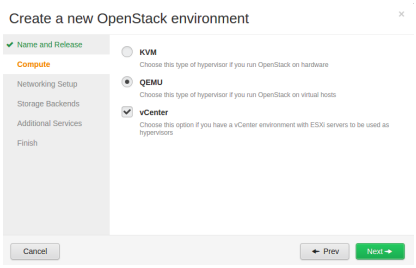
Create Environment

Select an OpenStack release and type a name for the OpenStack environment:

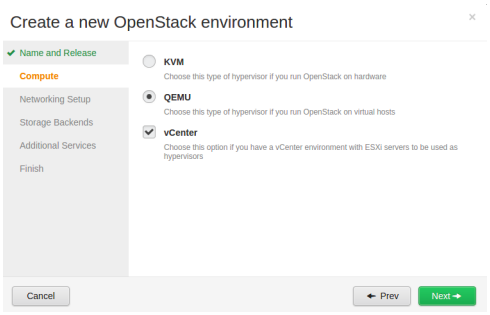
Select a hypervisor for VMware vCenter

You can select two hypervisors for your OpenStack environment.. That means, you now have three options:

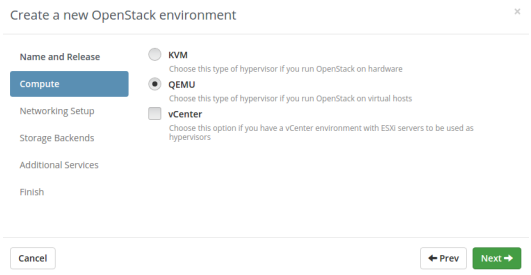
1. enable vCenter only - select vCenter checkbox and leave the radio button as is; you will then finish this configuration using VMware tab of the Fuel web UI. See the *VMware tab* for more details.



2. enable both vCenter and KVM/QEMU - select vCenter checkbox and choose between KVM and QEMU radio buttons.

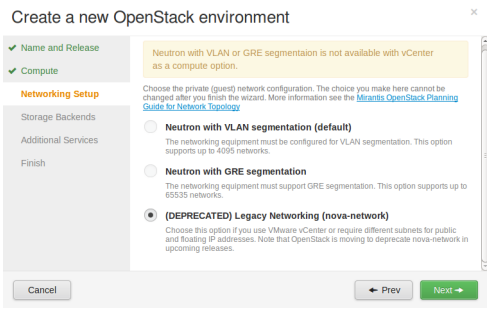


3. enable KVM/QEMU only - click the corresponding radio button and leave vCenter checkbox empty.



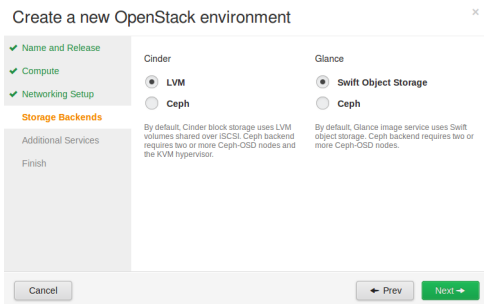
Select Network Service for vCenter

Currently, the only support network option for vCenter is nova-network.



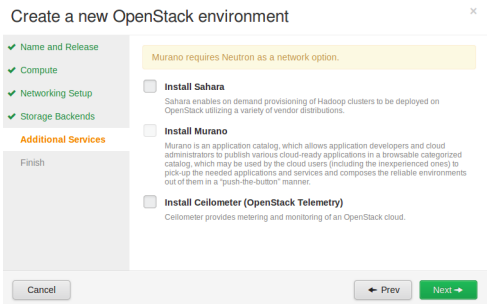
Choose Backend for Cinder and Glance with vCenter

At this step you should select storage backend for Cinder that is going to be used with KVM/QEMU if you deploy compute nodes. You can choose Ceph as the backend for Cinder and Glance with vCenter. If you would like to use Glance with VMware datastore, enable it on the *Settings* tab of the Fuel web UI and finish backend configuration at the VMware tab.



Related projects for vCenter

Nova-network does not support Murano, so you cannot run Murano in the OpenStack environment with vSphere integration.



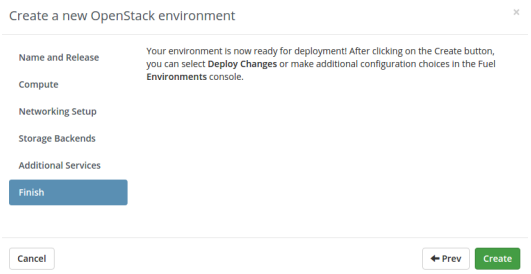
Previously, when you selected vCenter, Compute and Controller roles were assigned to the same node, while Ceilometer compute agent was not present on this node.

In Fuel 6.1 release, this logic was changed to provide metrics collection for the instances.

For previous Fuel releases, support for collecting polling meters from instances on vCenter was not implemented: only central agent polled services like Glance and Swift on Controller node. No metrics was collected from compute nodes.

In 6.1, Ceilometer support for vCenter is implemented according to 1-1 mapping principle (the one done between *nova-compute and vSphere cluster*). Now Ceilometer compute service is available for each vSphere cluster. That means,every agent polls resources about instances from those that only relate to their vSphere cluster. Every agent uses its own configuration file with authentication parameters for its specific vSphere cluster. What is more, monitoring under Pacemaker is introduced for every Ceilometer compute service to avoid failures (for example, failover of primary controller node) and missing polling data as the result.

Complete the creation of your vCenter environment



Select *Create* and click on the icon for your named environment.

Configuring a vCenter environment

Select your OpenStack environment integrated with vSphere to display the configuration screen. Set up the environment proceeding with the following steps.

Assign a role to a node server

Procedure:

1. In the configuration screen, select the *Nodes* tab.
2. Assign a role to the node server by checking the required option from the available options list.

Node roles for the OpenStack environments that support vCenter

Role	Description
Controller	Controller node initiates orchestration activities and provides an external API. Fuel installs components such as Glance, Keystone, Horizon, and Nova-Scheduler on Controller node as well. By default, for VMware vCenter integration, a nova-compute service with VCDriver runs on Controller node. The nova-compute service on Controller node manages VMs running on ESXi hosts through vCenter.
Compute	Compute node creates, manages, and terminates VMs. The nova-compute service on Compute node manages locally running VMs through KVM/QEMU.
Storage - Cinder	Cinder node provides scheduling of block storage resources, typically delivered over iSCSI or other compatible backend storage systems. You can use Block storage for database storage, expandable file systems, or to provide a server with access to raw block level devices. This node role can be enabled for the Cinder with LVM or Ceph environment.
Storage - Cinder Proxy to VMware Datastore	Cinder-VMware provides scheduling of block storage resources delivered over VMware vCenter.
Compute VMware	Compute VMware node runs nova-compute with VCDriver that manages ESXI computing resources through VMware vCenter. It enables you to deploy a nova-compute service on a standalone node rather than Controller node.

Deploy nova-compute on a standalone node

To distribute nova-compute services among available nodes and deploy nova-compute on a standalone node, proceed with the following steps.

Procedure:

1. Verify that the *Compute VMware* option for a specific node server is selected in the *Nodes* tab.
2. In the *VMware tab* for a specific cluster, select this node as the *Target node* for the `nova-compute` service.

> ***Note***
>
> Known limitations for the *Compute VMware* node role:
>
> - you cannot combine it with any other role;
> - you cannot add a Compute-VMware node to a deployed OpenStack environment.

Seealso

- *Nova-Computes* configuration.

Configure the Network

You should choose either the Nova-network FlatDHCP or the VLAN manager:

- VLAN manager provides better virtual machine isolation, i.e. enables segregating virtual machine tenants into separate broadcast domains.
- FlatDHCP manager uses a single IP subnet. Select it if you do not want to configure VLANs on your network equipment.

Please, note that nova-network will be working in a single-host mode (that means, the process runs on one of the Controllers) if you are using vCenter. When nova-network crashes it will be restarted by *pacemaker* on the same Controller or on another live Controller, during this period of time, all virtual machines will lose connectivity with external networks. Without vCenter, each compute node holds its own nova-network process (multi-host mode).

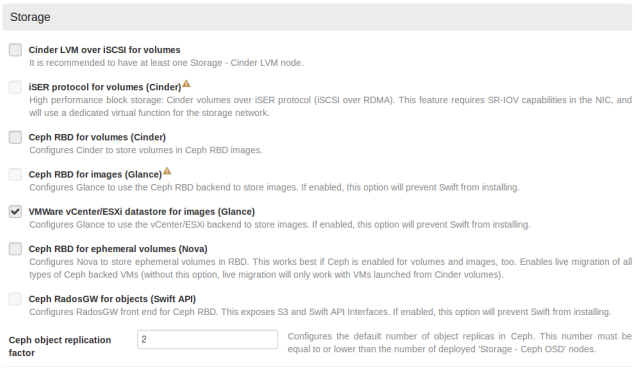
- To enable *FlatDHCP manager*, follow these steps:
    1. In the *Networks* tab, click the *FlatDHCP manager* radio button.
    2. In the *Nova-network configuration*, enable the *Use VLAN tagging for fixed networks* checkbox.
    3. Type the VLAN tag you selected for the VLAN ID in the ESXi host network configuration.
- To enable *VLAN manager*, follow these steps:
    1. In the *Networks* tab, select the *VLAN manager* radio button.

2. In the *Nova-network configuration*, select *Fixed network size* using the drop-down menu.
3. Specify *Number of fixed networks* and type *Fixed VLAN ID range*.
4. Click the **Verify Networks** button to verify if networks are configured correctly.
5. Click the **Save settings** button to continue.

Settings

To enable VMware vCenter for volumes, you should add a node and assign *Storage - Cinder Proxy to VMware Datastore* role to it, see *Assign a role to a node server* for details.

To enable VMware vCenter managed datastore as a backend for Glance, select *VMWare vCenter/ESXi datastore for images (Glance)* checkbox.



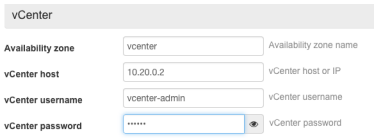
VMware tab

Beginning with Fuel 6.1 release, all vCenter-related settings are consolidated on the VMware tab of the Fuel web UI.

vCenter

In this section, you should enter not only vCenter credentials (previously found on the Fuel UI wizard and *Settings* of the Fuel web UI tab), but also specify Availability zone:

- For KVM/QEMU nova-compute services, availability zone is *nova*. You cannot edit its name, because it is the default availability zone used by OpenStack.
- For vCenter nova-compute services, the availability zone name is set to *vcenter* by default, but it can be changed.



Nova-Computes

Each nova-compute service controls a single vSphere cluster. For each vSphere cluster, you need to configure a separate nova-compute service that will be running either on the Controller node, or on a Compute-VMware host.

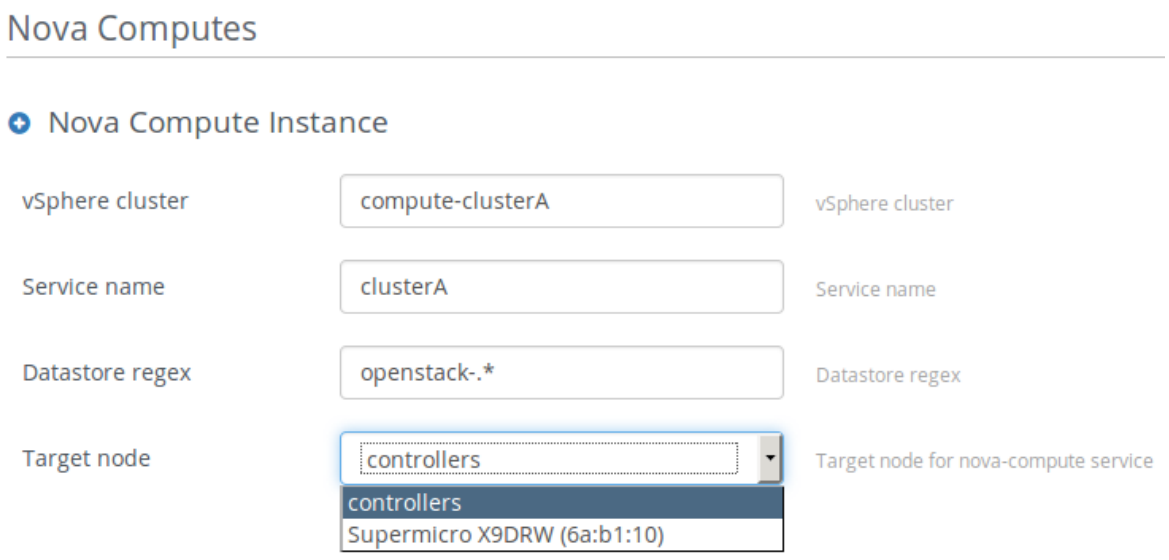
Select from the following options:

- for vCenter only environment - do not add any compute nodes.
- for dual hypervisors environments - configure the following:
  - **vSphere cluster** - specifies the name of the cluster that this nova-compute service manages.
  - **Service name** - specifies the service name to reference to your cluster. It is a string that should not contain any non-ASCII characters.
  - **Datastore regexp** - specifies the names of the data stores to be used with the Nova Compute instance.

    To use a specific data store, type its name in the *Datastore regexp* field. Verify that you enter the valid data store name.

> ***Note***
>
> If you provide an invalid data store name, you can still successfully deploy a vCenter environment. However, you will encounter difficulties later when creating instances in VMware vSphere.
>
> You can also specify a group of data stores. For example, type `openstack-.*` to select all the data stores which names start with `openstack-`.

  - **Target node** - a dropdown list with the following items:
    - the *controllers* option is selected by default. It deploys the nova-compute service on the Controller nodes.
    - names of all nodes with the compute-vmware role assigned. Select one of the available nodes if you decide to run the compute-service on that standalone node.



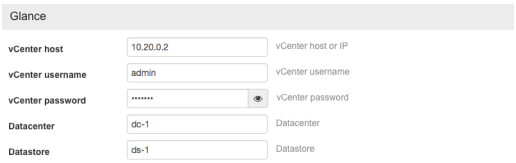
If required, configure more nova-compute instances by clicking +.

Network

If you decided to use VLAN Manager, enter the interface of ESXi hosts on which VLANs will be provisioned.

Glance

To enable Glance, you should first select the checkbox on the *Settings* tab (see *VMware vCenter/ESXi datastore for images (Glance)*). Then, you should enter the information for Glance.



Prepare Murano images for VMware vSphere

Typically, murano images provided in the Community App Catalog have the `qcow2` format. If you want to use murano images with VMware vCenter, you must convert the images to the `vmdk` format.

To prepare murano images for VMware vSphere:

1. Convert a `qcow2` image to the `vmdk` format. See Converting between image formats for details.
2. Upload the `vmdk` image into the Image service.

   If you use Docker or Kubernetes, set the image name to:

   - `ubuntu14.04-x64-docker` for Docker
   - `ubuntu14.04-x64-kubernetes` for Kubernetes
3. Configure the image metadata, so that it works with VMware vCenter proceeding with one of the following options:

   - In the OpenStack Dashboard, go to *Project* -> *Compute* -> *Images*, select the *Update Metadata* action from the *Actions* drop-down list. Update image resource metadata as follows:

Resource metadata name	Value
`vmware_adaptertype`	`lsiLogic`
`vmware_disktype`	`sparse`
`hypervisor_type`	`vmware`

   - In CLI, use the **glance image-update** command:

```
glance image-update IMAGE_ID --property vmware_adaptertyper="lsiLogic" --property vmwar
```

4. Mark the image with murano metadata proceeding with one of the following options:

    - In the OpenStack Dashboard, go to *Murano -> Manage -> Images*, click *Mark Image* to select the image to be marked.
    - In CLI, set the `murano_image_info` property with the **glance image-update** command:

    ```
    glance image-update <imageName> --property murano_image_info='{"title": "Windows 20
    ```

Seealso

- Tag VMware images
- Upload an image into glance with murano metadata

Index