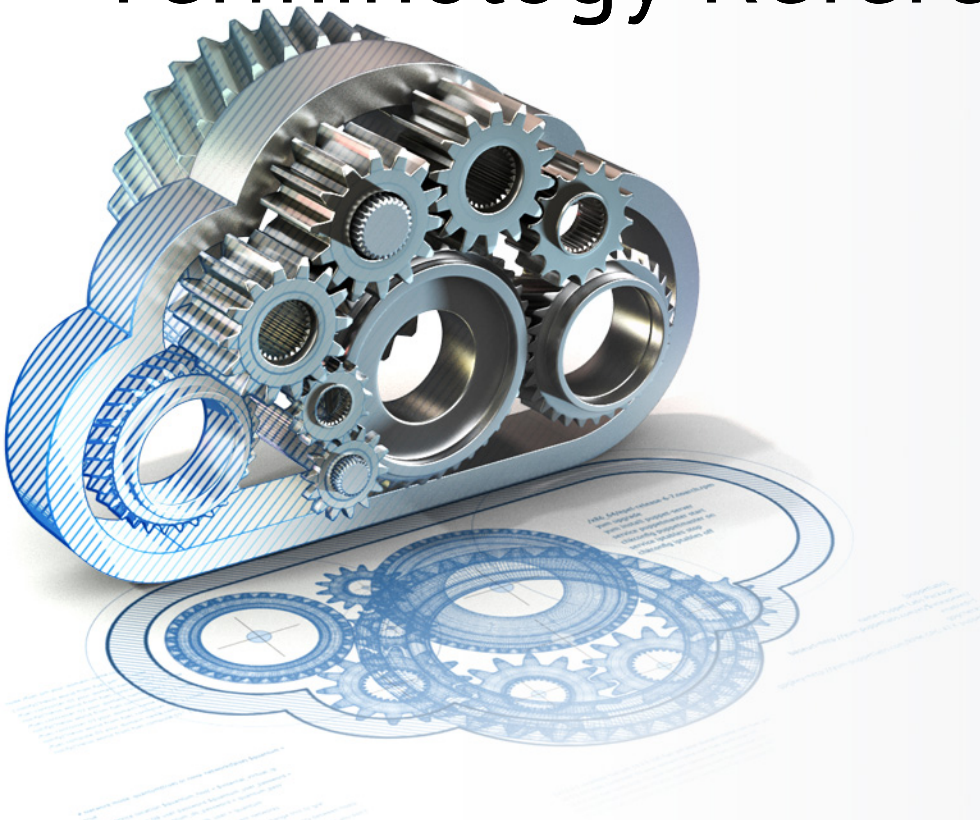
MIRANTIS

BUILT FOR openstack®

Mirantis OpenStack®

version 6.0

Terminology Reference



Contents

Preface

This documentation provides information on how to use Mirantis Fuel to deploy OpenStack environment. 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
October, 2014	6.0 Technical Preview
December, 2014	6.0 GA

Terminology Reference

AMQP (Advanced Message Queuing Protocol)

AMQP is an international standard for message-oriented middleware. AMQP is a wire-level protocol, meaning that it describes the format of the data that is transmitted, so any tool that is AMQP compliant can generate and interpret messages for any other AMQP compliant tool.

Mirantis OpenStack and Fuel use *RabbitMQ* as the AMQP compliant messaging interface.

For more information:

- The RabbitMQ site provides a nice tutorial about AMQP.
- For an architectural overview, see AMQP and Nova.
- ISO/IEC 19464 is the official standard.

Astute

Astute processes are worker processes used by *Fuel* to deploy an OpenStack environment. The *Nailgun* service creates a JSON file that contains information about the nodes and their roles and puts this file into the *RabbitMQ* queue. An Astute worker process then receives this information and deploys the environment by setting the configuration for each node in *Cobbler*. This same Astute worker process monitors the deployment process and periodically reports the progress to Nailgun through its RabbitMQ queue.

See Fuel Architecture for more information.

Availability Zone

An Availability Zone is a name given to a *host aggregate*. An Availability Zone can be used to segregate your cloud environment; for example, to identify hosts that are equipped for heavy computational activities or that have some sort of specialized equipment attached to them. Users can specify an Availability Zone when launching an instance in Horizon; the *Nova scheduler* then uses the Availability Zone as a criteria when allocating an instance to a node.

Bonding

NIC Bonding (also called NIC Aggregation) aggregates multiple physical links to one link to increase speed and provide fault tolerance. Mirantis OpenStack uses *OVS (Open vSwitch)* to implement NIC bonding.

See *Setting NIC bonding (NIC aggregation)* for instructions about implementing bonding through the Fuel UI. Mirantis OpenStack supports bonding in the following modes: Active-backup, Balance SLB (Source Level Bonding), and Balance TCP with LACP (Link Aggregation Control Protocol). The Fuel UI requires LACP to implement the Balance TCP mode.

OVS bonding is an *experimental feature* in Fuel 6.0. You can instead run Linux bonding; follow the instructions in *Types of Bonding*. See LP1401260.

Ceilometer (OpenStack Telemetry)

Ceilometer aggregates usage and performance data across all the services deployed in an OpenStack environment. This data provides visibility and insight into the usage of the cloud across dozens of data points and allows cloud operators to view metrics globally or by individual deployed resources. The information gathered can be used for monitoring and capacity planning, or to create an alarming service, or can serve data to external software for a customer billing system. The framework can be expanded to collect measurements for other needs.

Fuel can install Ceilometer on systems running either the CentOS or Ubuntu operating system; check the appropriate box on the *Related projects* screen when configuring your environment.

Note that Ceilometer collects a great deal of data and performs a large volume of database writes; with 100 resources and default configs Ceilometer collects around 16k samples per hour.

Mirantis OpenStack 5.x defaults to installing *MongoDB* as the recommended back-end database for OpenStack Telemetry. The Fuel Master Node enables you to choose the installation of MongoDB as a role onto a node; see *Assign a role or roles to each node server* for instructions. This resolves the Ceilometer performance issues caused by the volume of concurrent read/write operations.

For more information, see

- *Ceilometer and MongoDB*
- *Running Ceilometer* discusses how to configure Ceilometer and work with its output.
- Ceilometer wiki
- Ceilometer blob

Ceph

An open source storage platform that provides unified object, block, and file storage. For more information, see the Ceph documentation.

For information about deploying Ceph in Mirantis OpenStack, see *Choosing the Storage Model*.

Cinder

Cinder is the code name for the OpenStack Block Storage project. Cinder was originally part of the Nova project but is now an independent core OpenStack project. For more information, see the Cinder developer documentation.

Cinder can be deployed on its own OpenStack storage node (often called a "Cinder node") or can share a Controller node. For information about deploying Cinder in Mirantis OpenStack, see Storage Architecture.

The *VMDK* driver provides vCenter datastores for Cinder block storage.

Cobbler

Cobbler is the provisioning service used by *Fuel* in Mirantis OpenStack 5.x and earlier releases. The *Astute* worker processes populate user-defined configuration information for each OpenStack node. When each node is rebooted, Cobbler starts the operating system installer with the user-configured settings.

For more information about how Cobbler is used in Fuel, see Fuel Architecture. Also see the Cobbler Web Page.

Compute nodes (OpenStack Compute)

A Compute node is a *node* to which the Compute *role* is assigned; see *Assign a role or roles to each node server*. Compute nodes are the workhorses of your installation; they are the servers on which your users' virtual machines are created. *nova-compute* controls the life cycle of these VMs.

The Neutron *Open Vswitch* agent also runs on Compute nodes; other Neutron agents run only on Controller nodes. The Ceilometer Compute Agent may also run on Compute nodes.

The Fuel UI does not allow you to assign the Compute role to a node that runs the Controller role in most cases. It is possible, although not recommended, to run both the Compute and Controller roles on a single server by editing the *openstack.yaml* file.

> ***Note***
>
> In environments that use vCenter as the hypervisor, the Nova-compute service can run **only** on Controller nodes. Because of this, Fuel does not allow you to *assign* the "Compute" role to any node when vCenter is selected as the hypervisor.

Also see:

- *Choosing the Storage Model*
- *Compute Node Architecture*.
- Compute web site

Controller node

A Controller *node* is a server with the Controller *role* assigned to it, usually using the *Assign a role or roles to each node server* screen.

The Controller manages all activities in the environment. *nova-controller* maintains the life cycle of the Controller. along with *RabbitMQ*; *HAProxy*; *MySQL* and *Galera*; the *Pacemaker Cluster* (which includes *Corosync*); *Keystone*; *Glance*; and *Cinder*. Other services that may optionally run on the Controller include *Heat*, *Neutron*, *Swift*, *Ceph* Monitor, *Ceilometer*, *Sahara*, and *Murano*. See *Controller Node Architecture*.

To achieve *High Availability* for the environment, you must deploy a cluster of at least three Controller nodes.

The Fuel UI does not allow you to assign the *Compute* role to the server that runs the Controller role in most cases. It is possible, although not recommended, to run both the Compute and Controller roles on a single server by editing the *openstack.yaml* file.

Note

In environments that use vCenter as the hypervisor, the Nova-compute service can run **only** on Controller nodes. Because of this, Fuel does not allow you to *assign* the "Compute" role to any node when vCenter is selected as the hypervisor.

Corosync

The Corosync Cluster Engine is a Group Communication System with additional features for implementing high availability within applications. See the Corosync home page for more information. For a good discussion about Pacemaker and Corosync and how they are related, see What is Pacemaker?.

Mirantis OpenStack uses Corosync with the Pacemaker cluster resource manager as the communication and quorum service.

CRM (Cluster Resource Manager)

CRM is the main *Pacemaker* utility; use it to query, start/stop, or reset a resource and other maintenance tasks for the cluster.

For more information:

- *crm - Cluster Resource Manager* discusses some of the more common CRM commands.
- *How to verify that Neutron HA is working* discusses how to use CRM to analyze a Neutron high availability cluster.
- *HowTo: Enable/Disable Galera Cluster Autorebuild Mechanism* for information about using CRM to control the *Galera Cluster for MySQL* autorebuild feature.
- CRM interactive help and documentation.

DevStack

An OpenStack package that can be installed and deployed on your laptop or inside a VM on a cloud or other machine for evaluation purposes. DevStack runs in a virtual machine so does not give the same performance as OpenStack running on dedicated hardware and it only runs a single node. See the DevStack web page for installation instructions.

Fuel is a more powerful tool that supports a virtualized deployment including multiple controllers and support for HA. Fuel can also be used to deploy a bare-metal cloud installed on bare metal and appropriate for a production environment.

Docker containers and dockerctl

Docker and dockerctl provide tools used to deploy applications in *LXC (Linux containers)* containers. Docker brings LXC to the foreground by wrapping it with user-friendly commands. Coupled with volume and port linking capabilities, Docker lets you replace your complex server install with smaller, more manageable containers. Other benefits of Docker include:

- Docker's image format allows you to build once and deploy everywhere.
- Docker's image and Dockerfile format allow you to design containers that behave predictably on any system.
- Multiple copies of the same image can run at the same time, enabling upgrades to take place without showing any service interruptions to an end user.

See the docker home page.

Containers inside Docker do not include a copy of the kernel; instead, they use the same kernel as your host operating system, which means the disk footprint for a container is smaller and has less memory overhead than a traditional Virtual Machine. However, containers have some limitations compared to kernel VMs:

- Process limits cannot be modified
- Custom or different kernel modules cannot be loaded
- If the parent process of a container exits, the container is considered off
- No pause or live migration functionality is supported
- Linked ports or volumes cannot be used during the docker build process

Dockerctl is a simple wrapper for Docker that improves Docker's offline image handling and container versioning. Dockerctl augments basic Docker with the following capabilities:

- Ability to use the same idempotent identifiers across application versions (such as RabbitMQ)
- Ability to expose DHCP broadcast traffic to an LXC container running Cobbler and dnsmasq.
- Utility for running arbitrary commands or exposing a shell in a Docker container (via lxc-attach).
- Implement image management that really works in a deployment without Internet access.

Fuel 5.0 and later use Docker containers for the Postgres database, Cobbler, Puppet, and other components. See *System changes for Docker affecting Fuel 5.0 and later* for more information.

See

- *Docker Containers and Dockerctl* for a list of commands that are useful for managing LXC containers on the Fuel Master node.
- *Fuel Master and Docker disk space troubleshooting* for details about resolving Docker disk space problems on the Fuel Master node.

Ephemeral storage

Ephemeral storage is used for scratch space and as temporary storage for the operating system in a guest VM. Ephemeral storage is allocated for an instance; its size is determined by the flavor of the instance. Ephemeral storage persists through a reboot of the guest operating system but is deleted when the instance is deleted.

The Nova Compute service manages ephemeral storage.

- By default, ephemeral drives are stored locally on Compute nodes, in the Virtual Storage partition.
- If Ceph is configured for the environment and the Ceph RBD backend for ephemeral drives is *enabled*, Nova-compute stores ephemeral drives in Ceph.
- Other storage options are possible, such as an NFS share that is mounted from a SAN.

ESXi

ESXi is a Type 1 *Hypervisor* that serves as a virtualization server for *vSphere* and *vCenter*. All virtual machines and guest operating systems are installed on ESXi servers. VMware vCenter/ESXi datastore is now supported as a backend for Glance. To enable this option, see *vSphere deployment notes*.

Experimental features

Mirantis OpenStack ships with experimental features that you can enable and try out. Experimental features provide functionality that may be useful to some customers but has not been subjected to the rigorous testing that is required for environments that need high levels of stability.

The following features are currently defined as experimental:

- Zabbix integration; see *Zabbix monitoring tool*.
- Ability to apply minor updates to OpenStack environments within the same OpenStack major release; see *Upgrading and Updating from Earlier Releases*. Mirantis OpenStack 6.0 is the first release based on Juno and so no upgrade is possible.
- vCenter with NSX. See *Preparing for VMware NSX integration* and *Deploying NSX*.
- OVS *bonding*.
- Image-based provisioning.
- Ceilometer can use an external MongoDB installation.

Instructions for enabling experimental features on a running Fuel Master node are provided in *Enable Experimental Features*.

Fencing

Process of locking resources away from a node whose status is uncertain. Ceph supports fencing but you must ensure that no controllers host both the Ceph OSD and Ceph Mon roles.

Floating IP address

A floating IP address is an IP address, usually public, that is dynamically associated with a running virtual instance to make the VM accessible to the outside world. Note that external ports are associated with tenant routers, not with virtual instances.

To associate a floating IP address with your VM:

1. Add a floating IP address to the VM using either the Horizon dashboard or the command line.
2. Configure the *Security Group* to enable the kind of access required for the VM. See Configure access and security for instances.
3. Select the appropriate rule for ICMP and SSH access; preconfigured rules are available from a drop-down menu inside the Security Group configuration dialogue.

4. If you want to connect to the instance using SSH, set up one of the following mechanisms:
    - Use a specific password-enabled image or cloud-init scripts.
    - Inject your public key into the instance. Instructions are in the document referenced in Step #3.

For more information:

- To set the IP range when configuring your environment with Fuel, see *Network settings*
- For details about the OpenStack requirements for configuring floating IP addresses, see *Public and Floating IP address requirements*.
- Configure public (floating) IP addresses
- L3 routing and NAT

Fuel

Fuel is an open source, software life cycle management application that deploys multiple OpenStack environments from a single interface and then enables you to manage those environments post deployment.

Fuel includes:

- A web-based UI that runs on the *Fuel Master Node* The Fuel UI performs the following tasks:
    - create and manage multiple OpenStack environments
    - discover and configure hardware
    - configure network and storage for the environment
    - validate the network configuration before and after deployment
    - view logs in realtime from the UI

  The *Fuel CLI* is a shell tool that performs many of the same functions as the Fuel UI plus does some advanced, specialized configuration tasks.
- Reference architectures that we have tested and certified to ensure that your deployed clouds are scalable, reliable, and production quality.
- An open and flexible library (REST API) that enables customers to make configuration changes that are more advanced or focused than the default choices within Fuel. This library can also be used to fold additional drivers or integrations into the deployed environment.
- A complete, command-line interface for Fuel; see *Fuel CLI* that can be used for almost all tasks that are done from the Fuel UI as well as some advanced configuration tasks.

  The REST API communicates with *Nailgun*, which then manages the other activities to deploy the OpenStack environment.

Beginning with Mirantis OpenStack 6.0, Fuel features the ability to install plug-ins when you deploy your environment. Plug-ins are downloadable software components that extend the functionality of your environments in a flexible, repeatable, and reliable manner. There is no need to install drivers and patches manually after Fuel deploys your cloud; plug-ins do this for you.

See

- Fuel Architecture.

- Sequence diagrams provides details about how Fuel provisions the operating system, verifies the networking, then provisions and deploys the OpenStack environment.

Fuel Master Node

A server with the *Fuel* application installed, also commonly referred to as the Fuel server. The Fuel Master node runs on its own server or VM to deploy and manage OpenStack environments. It assigns IP addresses to the OpenStack nodes, performs PXE boot and initial configuration, and provisions the nodes according to their roles in the environment.

For more information, see

- *Download and Install Fuel* has instructions for downloading and installing the Fuel Master Node.
- *Create a new OpenStack environment* gives instructions for creating a new OpenStack environment from the Fuel Master Node.
- Fuel Architecture for details about how Fuel is implemented.

Galera Cluster for MySQL

Galera is a synchronous multi-master cluster for the MySQL database. Mirantis OpenStack uses MySQL/Galera for HA deployments; see *MySQL and Galera* in Reference Architecture for more information.

Additional information is available:

- *HowTo: Backport Galera Pacemaker OCF script* explains how to backport the Galera Pacemaker OCF script to pre-5.1 environments.
- *HowTo: Enable/Disable Galera Cluster Autorebuild Mechanism*
- Galera Cluster documentation.
- Monitoring the Galera Cluster.
- Troubleshooting the Galera Cluster.

Glance

Glance is the OpenStack Image Storage project. Glance provides a scaleable, RESTful API for VM images and metadata.

Fuel can deploy either of the following as the storage backend for Glance:

- *Swift*, the standard OpenStack object storage component
- *Ceph RBD*, the distributed block device provided by the Ceph storage platform.
- In multi-node mode, Glance uses local file storage instead of Swift when configured with the default option (in other words, if Ceph is not enabled as the storage backend).

Glance can only be deployed on controller nodes in Fuel.

For more information, see the Glance developer documentation.

HA

HA (High Availability or Highly Available architecture) replicates servers and services to minimize downtime and data loss in the event of a hardware failure. OpenStack is architected to support HA; see the OpenStack High Availability Guide for more information about the OpenStack implementation.

When you create an OpenStack environment with Fuel, you are asked to choose either an HA or non-HA deployment mode on the *High-availability (HA) or non-HA mode* screen. Before Mirantis OpenStack 5.1, different internal architectures were used for the two modes; if you deployed a non-HA environment, you had to redeploy the environment to convert it to HA.

Now you can deploy the Highly Available architecture (also known as Multi-node HA) onto any number of controllers, including just one. The environment is not considered highly available if installed on a single controller because no secondary controller is present for failover, but the same internal mechanisms are used so you can later add replica servers and transform your OpenStack environment into an HA environment without having to redeploy it. but controllers can be added to the environment at a later time and will be properly configured for High Availability in conjunction with the initially deployed controller.

At least three controllers must be configured to have a reliable HA environment; this allows for failover and the *Pacemaker* quorum. The controller cluster can include more than three servers to increase the level of reliability. For more information about how Fuel deploys HA, see *Multi-node with HA Deployment*.

The multi-node mode, previously used to deploy an environment with only one controller, is retained as a legacy option in Mirantis OpenStack for 5.1 but it is not recommended for deploying new environments; we expect to remove this option in a future release.

Hadoop (Apache Hadoop)

The Apache Hadoop project develops software for processing "Big Data". See the Apache Hadoop homepage.

The OpenStack *Sahara* project provides extensions to OpenStack that enable it to run Hadoop clusters.

HAProxy

HAProxy provides load balancing, proxying, and high availability in a Mirantis OpenStack environment. Each OpenStack controller runs HAProxy, which manages a single External Virtual IP (VIP) for all controller nodes and provides HTTP and TCP load balancing of requests going to OpenStack API services, RabbitMQ, and MySQL.

See:

- HA Proxy documentation

Hardened packages in Mirantis OpenStack

Hardened packages in Mirantis OpenStack include the core OpenStack projects, updated with each stable release of OpenStack, and supporting a broad range of operating systems, hypervisors, and deployment topologies. Also included are:

- Packages to ensure High Availability.
- Fixes for defects reported by our customers that may not yet have been merged into the community source.
- Mirantis-driven premium OpenStack projects such as Sahara (which provides a simple means to provision a Hadoop cluster on top of OpenStack) and Murano (an application catalog that can be used to publish apps and compose reliable environments out of them.)

- Mirantis-certified partner plug-ins, drivers, and integrations.

By default, Mirantis OpenStack, enables features that Mirantis has certified and confirmed as production-ready. It also includes some "experimental" features, which are new features that are less-hardened but are integrated into the product for customers who can tolerate some risk. See *Enable Experimental Features* for information about enabling these experimental features.

Havana

Code name for the eighth release of the OpenStack software. For more information, see the Havana web site. Mirantis OpenStack version 4.0 and Fuel 4.1 incorporate and support the Havana code base. The following improvements in Havana are not deployed by Fuel 4.x although they are included in Mirantis OpenStack 4.x:

- Nova Compute: Cells, Availability zones, Host aggregates
- Neutron: Load Balancer as a Service (LBaaS),
- Multiple L3 and DHCP agents per cloud
- Keystone: Multi-factor authentication, PKI authentication
- Swift: Regions, Adjustable replica count, Cross-project ACLs
- Cinder: Cinder-backup service, Support for Fibre Channel over Ethernet (FCoE)
- linux-iscsi.org (LIO) is now supported as an Internet Small Computer System Interface (iSCSI) backend

Heat

Heat is the OpenStack Orchestration service. It implements a framework that manages the lifecycle of your infrastructure inside the OpenStack cloud. Mirantis OpenStack 4.0 and later deploys Heat into each environment by default. Heat uses templates to describe the deployment process of instances, but also supports the AWS CloudFormation template format.

Heat uses a template that can be maintained under source code control. See the Heat wiki for more information.

The Heat components are:

heat-api -- provides a native REST API and processes API requests.

heat-api-cfn -- similar to the *heat-api* but also provides an AWS CloudFormation compatible API.

heat-engine -- main component of the Heat framework. It does all the work of reading templates, launching instances and providing events to the API users.

See also:

- The official documentation of the Heat project
- Development documentation
- Mirantis blog record

Horizon

Horizon is the OpenStack dashboard application that provides a web-based GUI used to configure, monitor, and manage core OpenStack services such as Nova, Swift, and Keystone.

See http://docs.openstack.org/developer/horizon/ for more information.

Host Aggregates

A host aggregate is a grouping of hosts; one host can be in more than one host aggregate. Host aggregates are only exposed to cloud administrators.

You can assign a *availability zone*` name to a host aggregate. Users can then request that availability zone when creating an instance.

For more information, see:

- Segregating your Cloud provides good conceptual information, although it is out-dated in that it still requires configuration changes to the *nova.conf* file.
- Host Aggregates
- Russell Bryant's Availability Zones and Host Aggregates in OpenStack Compute (Nova) article.

Hypervisor

A hypervisor creates and runs virtual machines. OpenStack runs a hypervisor as a component of the compute node. You can select the KVM, QEMU, or *VMware vCenter* hypervisor from the *Hypervisor* screen; other hypervisors are available through Mirantis Services. See the HypervisorSupportMatrix web page for information about the status of other hypervisors for OpenStack.

Icehouse

Code name for the ninth release of the OpenStack software. For more information, see the Icehouse web site.

Mirantis OpenStack version 5.0 and later incorporates and supports the Icehouse code base. The following features in Icehouse are not deployed by Fuel 5.x but can be configured manually using the standard OpenStack practices:

- Neutron: Load Balancer as a Service (LBaaS),
- Multiple L3 and DHCP agents per cloud
- Keystone: Multi-factor authentication, PKI authentication
- Swift: Regions, Adjustable replica count, Cross-project ACLs
- Cinder: Cinder-backup service, Support for Fibre Channel over Ethernet (FCoE)

Identity Service

The OpenStack Identity service provides a central directory of users mapped to the OpenStack services they can access. It acts as a common authentication system across the cloud operating system and can integrate with existing backend directory services such as LDAP. It supports multiple forms of authentication including standard username and password credentials, token-based systems and AWS-style logins.

The catalog also provides a queryable list of all of the services deployed in an OpenStack cloud in a single registry. Users and third-party tools can programmatically determine which resources they can access.

As an administrator, OpenStack Identity enables you to:

- Configure centralized policies across users and systems.
- Create users and tenants and define permissions for compute, storage and networking resources using role-based access control (RBAC) features.
- Integrate with an existing directory such as LDAP, allowing for a single source of identity authentication across the enterprise

As a user, OpenStack Identity enables you to:

- Get a list of the services that you can access
- Make API requests or log into the web dashboard to create resources owned by your account

Pacemaker uses an OpenStack Resource Agent to manage the Identity Service in OpenStack High Availability deployments. For information about the High Availability Identity service, see Highly Available OpenStack Identity.

Image Service

The OpenStack Image Service provides discovery, registration and delivery services for disk and server images. The ability to copy or snapshot a server image and immediately store it away is a powerful capability of the OpenStack cloud operating system. A stored image can be used as a template to get new servers up and running quickly and consistently; it can also be used to store and catalog an unlimited number of backups.

The Image Service can store disk and server images in a variety of back-ends, including OpenStack Object Storage. The Image Service API provides a standard REST interface for querying information about disk images and lets clients stream the images to new servers.

Capabilities of the Image Service include:

- Administrators can create base templates from which their users can start new compute instances
- Users can choose from available images, or create their own images from existing servers
- Snapshots can be stored in the Image Service so that virtual machines can be backed up quickly
- A multi-format image registry, the image service allows uploads of private and public images in a variety of formats, including Raw, Machine (kernel/ramdisk outside of image, a.k.a. AMI), VHD (Hyper-V), VDI (VirtualBox), qcow2 (Qemu/KVM), VMDK (VMWare), and OVF (VMWare, others).

Pacemaker uses an OpenStack Resource Agent to manage the Image Service in OpenStack High Availability deployments. For information about the High Availability Image service, see Highly Available OpenStack Image API.

Ironic

Ironic is an API to manage and provision physical machines; it is currently an incubated project that is forked from and will replace the Nova "baremetal" driver used in the Grizzly and Havana releases. See the Ironic wiki page.

iSER

iSER stands for "iSCSI Extensions for *RDMA*". It is an extension of the data transfer model of iSCSI, a storage networking standard for TCP/IP. iSER enables the iSCSI protocol to take advantage of the RDMA protocol suite to supply higher bandwidth for block storage transfers (zero time copy behavior). To that fact, it eliminates the TCP/IP processing overhead while preserving the compatibility with iSCSI protocol.

Juno

Code name for the tenth release of the Openstack software. Mirantis OpenStack version 6.0 incorporates and supporte the Juno code base. The following improvements in Juno are not deployed by Fuel 6.x although they are included in Mirantis OpenStack 6.x.

Keystone

Keystone is the OpenStack *Identity Service*. It is installed on the target nodes and used by OpenStack.

Beginning with Mirantis OpenStack 5.1, Fuel creates a separate Keystone instance that is installed in a container on the Fuel Master node and manages access to the Fuel UI. See *Fuel Access Control* for more information. In Release 5.1.1 and later, expired tokens are automatically cleaned from the Keystone database on the Fuel Master node.

- Keystone web page
- *Details of Multi-node with HA Deployment* discusses how Keystone works in the OpenStack Environment.
- *Keystone Token Cleanup* discusses how to manage expired Keystone tokens in the databases installed on the Controller nodes to avoid performance degradation in the OpenStack environment.

KVM

One of the *hypervisors* that can be selected from the *Hypervisor* screen in the Fuel UI.

L3 Agents

An L3 agent is a *Neutron* agent that allows administrators and tenants to create routers that interconnect L2 networks and floating IPs that create ports on private networks to make them publicly accessible.

In Mirantis OpenStack 6.0 and later, multiple L3 agents are configured, one on on each Controller. In earlier releases, each environment had a single L3 agent that was located on one of the Controllers. This helps avoid the performance bottlenecks that could occur with only one L3 agent running in the environment. Rescheduling of networks is moved to Neutron server code, which accomplishes this by automatically reassigning routers to L3 agents when it detects that a particular L3 agent is dead.

These multiple L3 agents are actually clones that cam be addressed with the **clone_p_neutron-l3-agent** resource; the **p_neutron-l3-agent** resource is still provided, to act on a specific resource on a specific node.

Logging

OpenStack and Fuel use the standard Linux **syslog** and **rsyslog** facilities for logging events on the Fuel Master Node and on the nodes in the OpenStack environment.

See the following:

- *Configuring syslog* tells how to configure your environment to use a remote server as the **rsyslog** server rather than use the Fuel Master Node as the **rsyslog** server for the environment.

- *Setting debug level for the environment* tells how to configure debug logging for the environment.
- *Logs and messages* discusses

For more information about OpenStack logs, see the community documentation:

- Manage Logs
- Logging and Monitoring
- Compute service logging

LVM

LVM (Logical Volume Manager) is the default storage backend for *Cinder* block storage. You can instead choose *Ceph* as the storage backend for Cinder on the *Storage backend for Cinder and Glance* screen.

See *Choosing the Storage Model* for information about choosing the storage backend for Cinder.

LXC (Linux containers)

LXC (Linux containers) are peacemeal containers that can be modified, upgraded, and backed up independently. LXC is a user space interface to the Linux kernel containment features. See LXC for more information. OpenStack is phasing in support for LXC containers using *Docker containers and dockerctl*.

Mirantis OpenStack

Hardened OpenStack distribution plus additional services for high availability deployed by Fuel. Fuel deploys Mirantis OpenStack with an operating system based on either the Ubuntu or CentOS Linux distro.

ML2

The ML2 (Modular Layer 2) plug-in is a framework that allows OpenStack Neutron networking to simultaneously utilize a variety of Layer 2 networking technologies. Fuel 5.1 and later releases support ML2 drivers.

Two types of ML2 drivers can be implemented:

- **type drivers** each manage one network type. They manage type-specific network state information, validate the network, and allocate tenant networks. Network types that are currently supported include local, flat, vlan, gre, and vxlan.
- **mechanism drivers** manage network mechanisms. They create, update, and delete network and port resources. Two methods are exposed for each action:
  - ACTION_RESOUCE__precommit -- called within the context of the database transaction to validate the action being taken and modify the mechanism driver's private database. This method cannot communicate with anything outside Neutron because it should not block.
  - ACTION_RESOURCE_postcommit -- called after the database transaction is complete to push the resource change to the entity that is responsible for applying that change. For example, it pushes the change to an external network controller that updates the network resources.

For more information, see:

- *Creating and Configuring ML2 Drivers for Neutron* gives instructions for implementing an ML2 driver.

- Richard Boswell's article includes a detailed example of implementing an ML2 driver; look for the "ML2 Configuration" section about half-way through the article.
- Neutron/ML2 Wiki gives an overview of ML2.

MongoDB

MongoDB is an open source NoSQL document database optimized to host very large tables on commodity hardware. See the MongoDB web site.

Mirantis OpenStack 5.0 and later (OpenStack *Icehouse* and later) implements MongoDB to use as a backend for *Ceilometer (OpenStack Telemetry)*. The MongoDB role should be activated only if you are running Ceilometer; other OpenStack services continue to use *MySQL*. Fuel requires that the MongoDB roll be activated in order to install Ceilometer; see *Assign a role or roles to each node server*. It is possible (although not recommended) to revert Ceilometer to use MySQL after deployment.

Murano

The Murano project provides an application catalog that application developers and cloud administrators can use to publish various cloud-ready applications in a browsable, categorized catalog. Users can then select the applications and services they need and install them in a "push-the-button" manner. See the Murano wiki.

Fuel can deploy all Murano components, including the Murano Dashboard. See *Murano Deployment Notes* for more information about deploying Murano with Fuel.

Murano requires one of the *Neutron* network topologies; if you choose nova-network as the network type when deploying your OpenStack environment with Fuel, you will not be able to deploy Murano in that environment.

MySQL

The database most frequently used in OpenStack deployments. The MySQL database runs on the controller node; MySQL client software must be installed on other nodes that access the MySQL database.

For *HA* deployments, Mirantis OpenStack uses *Pacemaker*/*Corosync* to provide redundancy and failover capabilities for MySQL. Mirantis OpenStack uses MySQL/Galera for database replication in HA deployments that use the CentOS or Ubuntu kernel; see Preparing MySQL for Pacemaker high availability.

Note that *Ceilometer (OpenStack Telemetry)* uses *MongoDB* rather than MySQL.

Nailgun

Nailgun is the configuration and management service used as the backend for the Fuel UI and CLI. It contains the logic for creating an environment and editing its settings; assigning roles to the discovered nodes; an starting the deployment process for a new OpenStack cluster.

Nailgun stores all of its data in a PostgreSQL database. that contains the hardware configuration of all managed nodes it discovers, plus the roles, environment settings, current deployment status and progress of running deployments.

Note that Nailgun in Fuel is not in any way related to the Nailgun that provides a JVM in which Java programs can be run without incurring the standard JVM startup overhead.

- For a detailed explanation of Nailgun's role in the Fuel architecture, see the Fuel Architecture Overview.

- For details about developing and customizing Nailgun, see Nailgun Development and Customization Instructions.

Native VLAN

An untagged VLAN on a tagged port.

Networking

The nodes in the OpenStack environment communicate with each other over using one of the network topologies:

- With *Neutron* networking, GRE tunnels or VLANs can be used for network segmentation.
- With *Nova Network*, FlatDHCP and VLAN modes are available.

The following documents provide information:

- For general information to help you select the best network topology for your environment, see *Choose Network Topology*.
- For a list of the logical networks used in OpenStack (Public, Storage, Administrative, and so forth), see *Logical Networks*.
- For diagrams, detailed discussions, and instructions for deploying the different networking models, see *Neutron Network Topologies* and *Nova Network Topologies*.
- For information about calculating the hardware required for your deployment, see *Calculating Network*.
- *Configure your Environment* includes instructions for using the Fuel UI to change network parameters during and after installation.
- *Setting NIC bonding (NIC aggregation)* gives instructions for using the Fuel UI to set up *Bonding*.
- *Advanced Network Configuration using Open VSwitch* describes *OVS (Open vSwitch)* and includes instructions for adjusting the network configuration by editing configuration files and using the command-line tools.
- *Changing PXE Network Parameters During Installation*

Neutron

Neutron (formerly know as Quantum) is one of the network models used in OpenStack deployments. It is an OpenStack Core project to provide networking as a service between interface devices such as vNICS that are managed by other OpenStack services such as Nova. See the Neutron web page for more information.

See

- *Choose Network Topology* for information about choosing a network topology
- *Neutron Network Topologies* for more detailed information about the neutron implementation and how to configure it.
- *How to verify that Neutron HA is working* for instructions about using *CRM (Cluster Resource Manager)* to verify that the Neutron *HA* cluster is working properly.

NIC (Network Interface Card)

The physical networking port and hardware used for networking. In a virtualized deployment, NIC can also refer to the software interfaces between virtual machines.

Node

A node is a physical or virtual server that is provisioned to run a specific set of OpenStack services. Nodes are often identified by the *role* that they run (for example, Controller node or Compute node) but internally they are identified by a node ID number, such as node-1, node-2, and so forth. Use the **fuel node list** command on the Fuel Master node to see the node IDs that are assigned for your environment. See *How Nodes are Defined and Managed* for details about the information displayed by this command.

Fuel assigns node numbers sequentially, in the order that the hardware is discovered. The node ID is a primary key so node IDs are never reused unless the environment is redeployed. For example, if you deploy an environment with ten nodes, then delete node-6 and add a new node, the new node will be node-11.

The node-id can be used to SSH to a node and is used to identify a specific node from *fuel CLI*.

The term "node" is also used to refer to a member of a *Galera Cluster*, a *Pacemaker* Cluster, a *RabbitMQ* Cluster, a *Hadoop* Cluster, and so forth.

Node group

A node group is a set of nodes, grouped by networks; a cluster is a set of node groups. This is a *Nailgun* concept that is introduced to support the *Multiple Cluster Networks* feature in Fuel 6.0 and later.

See:

- *Configuring Multiple Cluster Networks*
- *Implementing Multiple Cluster Networks*
- *Node group* lists the Fuel CLI commands for managing node groups.

Nova

OpenStack Core project used for compute nodes; all major Nova components can be run on multiple servers and use message queues for communication between components. See the Nova web page for more information.

The Baremetal driver used for provisioning in Nova has recently been forked into its own project; see "Ironic".

Nova Network

The Nova-network model was the original networking model for OpenStack. It supports two topologies -- the FlatDHCPManager and VLAN Manager -- that can be used to deploy private networks for tenants; see *Nova Network Topologies* for more information about using Nova-network in Mirantis OpenStack.

Nova network is scheduled for deprecation in favor of the *Neutron* network model. See Deprecation of Nova Network for more details.

Choose Network Topology gives considerations for choosing a network topology.

NSX

VMware NSX is a network virtualization platform that can be used as network provider for *Neutron*.

Fuel 5.1 and later can deploy Mirantis OpenStack environments that use VMware NSX as a backend for Neutron.

- For more details about the VMware NSX platform, see VMware NSX Network Virtualization Platform whitepaper
- For information about planning your OpenStack deployment with VMware NSX integration, see *Preparing for VMware NSX integration*.
- For information about deploying Mirantis OpenStack with NSX integration, see *Deploying NSX*.
- For background information about how NSX support is implemented in Mirantis OpenStack, see *Neutron with VMware NSX*.

Object Storage

Provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving, and data retention. This is not a traditional file system but rather a distributed storage system for static data such as virtual machine images, photo storage, email storage, backups, and archives. Objects and files are written to multiple disk drives spread across different servers in the data center; the OpenStack software ensures data replication and integrity across the cluster.

OpenStack

Open source software that can be used to deliver a massively scalable cloud operating system for private and public clouds. For more information, see the OpenStack web page and OpenStack documentation.

The Mirantis OpenStack distribution packages a stable version of the open source pieces into an installable package that deploys an operating system based on either Ubuntu or CentOS. and adds Fuel to simplify the deployment and management tasks.

Orchestration Service

OpenStack Orchestration is a template-driven engine that allows application developers to describe and automate the deployment of infrastructure. The flexible template language can specify compute, storage and networking configurations as well as detailed post-deployment activity to automate the full provisioning of infrastructure as well as services and applications. Through integration with the Telemetry service, the Orchestration engine can also perform auto-scaling of certain infrastructure elements.

Overcommit ratio

The overcommit ratio defines the amount of virtual CPU, memory, and disk resources that can be allocated to instances on a Compute node, relative to the amount of physical resources available on that node. Often, instances do not fully utilize all resources allocated to them; overcommittment allows you to better utilize the available resources in light of this fact. You must carefully monitor resource utilization to ensure that adequate resources are available on the system. If this ratio is set too high for your workload, customers may suffer performance degradation for the instances; if CPU and memory resources are exhausted, instances may be destroyed; if a host runs out of disk space, instances may suffer spurious disk I/O errors.

Overcommit ratios can be set for CPUs, RAM, and disk. By default, Fuel sets the overcommit ratio for CPUs at 8:1; This means that, if your physical node has 12 cores, the Filter *Scheduler* sees 96 available virtual cores and so could provision 24 4-core instances on that physical node.

Fuel sets the overcommit ratio for CPUs at 8:1, and RAM and disks at 1:1, meaning that the scheduler only sees the actual amount of physical memory and physical disk space that is allocated. Note that OpenStack sets the overcommit ratio for CPUs at 16:1 and the overcommitment ratio for RAM at 1.5:1. The default CPU overcommit rate of 8:1 is necessary for the Mirantis CI tools but may not be the best default rate for customer environments. If you do not know what your VM workload is going to be, reset this ratio to 1:1 to avoid CPU congestion. You can then use the *atop* service and other tools to monitor activity in your environment to determine whether a different overcommit ratio is appropriate for your environment.

To modify the overcommit ratio(s):

- Log into each Compute node.
- Edit the */etc/nova/nova.conf* file to change the values.
- Restart the nova-compute service.
- Log into each Controller node and restart the nova-scheduler service.

Note that the overcommit ratio is not recognized by the traditional Simple "naive" Scheduler.

For more information, see Overcommitting.

OVS (Open vSwitch)

Multilayer virtual switch that the Neutron networking model uses to create a flexible network setup and to isolate tenants from each other on L2 and L3 layers. You can do some basic configuration of OVS on the Fuel UI beginning with Fuel 4.1; additional customization can be done by editing configuration files and using the command-line tools; see *Advanced Network Configuration using Open VSwitch*.

Pacemaker

High-availability resource manager for Linux based clusters. Fuel uses Packemaker with *Corosync* to implement *HA* for OpenStack services.

See:

- *CRM (Cluster Resource Manager)* is the utility used to manage Pacemaker
- *How HA with Pacemaker and Corosync Works* discusses how Mirantis OpenStack uses Pacemaker and Corosync to implement *HA*.
- What is Pacemaker? for a good discussion about Pacemaker and Corosync and how they are related.
- The Pacemaker Cluster Stack discusses how Pacemaker is used with OpenStack.
- Pacemaker web page contains more in-depth information about Pacemaker.

Persistent Storage

Persistent storage is storage that exists outside an instance, in contrast to *ephemeral storage*.

Fuel deploys storage for two types of persistent data:

- *Glance*, for image data, which can use either *Swift* or *Ceph RBD* as the storage backend
- *Cinder*, for block data, which can use either *LVM* or *Ceph RBD* as the storage backend

Plugin

Beginning with Mirantis OpenStack 6.0, Fuel features the ability to install plugins before you deploy your environment. Plugins are downloadable software components that extend the functionality of your environments in a flexible, repeatable and reliable manner.

For example, Neutron LBaaS provides Load-Balancing-as-a-Service for Neutron, OpenStack Network Service.

You can download Fuel plugins, read the documentation and perform required installation steps using Fuel Plugins Catalog. For common instructions on installation, see *Install Fuel Plugins*.

If you would like to develop you own plugin for Fuel, enter Fuel Plugins SDK.

Note, that Mirantis provides Fuel Plugins Certification - a set of technical and business processes that Mirantis uses to verify Fuel plugins built by vendors, allowing customer choice of plugins to lead to a predictable outcome. That means, Mirantis Certification ensures the quality of developed solution.

In terms of Certification, Fuel plugins fall into two categories:

- *Certified* - thoroughly reviewed, tested and supported by Mirantis.
- *Non-Certified* - reviewed, tested in terms of code and installation procedure, but not supported by Mirantis.

See the certification requirements at Fuel Plugin Certification page.

Puppet

Puppet modules bring scalable and reliable IT automation to OpenStack cloud deployments. See the Puppet web page for more details.

Fuel uses Puppet as the configuration management system that compiles a set of instructions for a configurable, reproducible, and sharable installation process. In Fuel 4.1 and later, the Puppet modules and manifests are synchronized between the master nodes and the managed nodes, then applied locally. This solves the security signing, scalability, and performance issues encountered on earlier releases where the Puppet Master Node ran on the Fuel Node Master.

Passing custom attributes can be helpful when you have some Puppet manifests that should be run but are not supported by Fuel itself. See *Using YAML configuration files*.

PXE (Preboot eXecution Environment)

An environment to boot a computer using a network interface independent of storage devices. Fuel uses PXE to boot all nodes in the OpenStack environment. For instructions on managing PXE network parameters, see *Changing PXE Network Parameters During Installation*.

qcow2

qcow2 (QEMU Copy-on-Write) is one of the file formats Fuel supports for Glance image storage and ephemeral volumes. You can set the file format to use for images on the *Choose image format* screen.

- When using *Swift* as the storage backend, qcow2 is the recommended format for storage images. It provides copy-on-write and snapshot functionality for Nova.
- When using *Ceph* as the storage backend for Cinder, you must disable qcow2 so that images are stored in raw format. Ceph includes its own mechanisms that provide copy-on-write capabilities and snapshots; if you use the qcow2 image format with Ceph, images must be converted to raw format in order to be cloned.

QEMU

One of the hypervisors that can be selected from the Fuel UI.

Quotas

OpenStack uses quotas to limit the resources one tenant consumes and thus prevent system capacities from being exhausted without notification. Fuel has a deployment setting to enable/disable quotas; they are disabled by default.

You can set quotas from Horizon; use the "Edit Quotas" form in the tenant panel. You can also use the **nova quota-update** command; For more information about how quotas work and how to use the */nova quota-upate** command, see Manage quotas.

RabbitMQ

RabbitMQ is an AMQP messaging system that OpenStack uses to manage asynchronous communications between OpenStack components.

See:

- RabbitMQ documentation

Rally

Rally is the OpenStack benchmarking suite. You can run Rally against an established OpenStack environment to identify functional failures and performance bottlenecks. See https://wiki.openstack.org/wiki/Rally.

RDMA

RDMA stands for "Remote Direct Memory Access". It is a technology that enables to read and write data from remote server without involving the CPU. It reduces latency and increases throughput. In addition, the CPU is free to perform other tasks.

Resource Agents (RAs)

Resource Agents are used by Pacemaker to interact with applications. Pacemaker includes native that are used in OpenStack high-availability deployments such as those that manage the MySQL databases or virtual IP addresses. Pacemaker also supports third party RAs (such as one that manages RabbitMQ) and native OpenStack RAs (such as those that manage the OpenStack Identity and Image Services.

Role

A role is a functional set of services that Fuel installs as a whole on a *node*.

Fuel can deploy the following roles; see *Assign a role or roles to each node server*:

- *Controller*: No other roles should be deployed on the Controller nodes to avoid resource contention that can lead to poor performance.
    - Fuel prevents the *Compute role* from being installed on Controller nodes although you can disable this restriction by manually editing the *openstack.yaml* file in non-production environments.
    - Fuel allows the *Ceph* OSD to be deployed on the Controller node but this is strongly discouraged for production environments because it degrades the Controller performance.
- *Compute*: Any of the Storage roles can be assigned to the same server(s) that run the Compute role and this is commonly done for small environments. For larger production environments, having dedicated servers for the Compute and Storage nodes usually yields better performance.
- **Storage:**

    See *Storage Node Architecture*.

    Fuel can deploy two types of Storage roles:

    - *Cinder* LVM
    - *Ceph* OSD

    Note that *Swift* storage runs on the Controller node(s) when Ceph is not enabled for image storage in the environment; it is not a separate role that can be deployed on its own server.
- *Telemetry -- MongoDB*: This is the database used for *Ceilometer* (OpenStack Telemetry). It should be deployed on its own node in production environments because it can severely degrade (or even crash) the Controller node it runs on when sampling large amounts of data at short intervals. See *Ceilometer and MongoDB* for more information.
- *Zabbix Server*: Must always be deployed on its own node.

Sahara (formerly known as "Savanna")

Sahara is the OpenStack service that provisions an Apache Hadoop cluster on top of OpenStack. Sahara currently supports Vanilla Apache Hadoop, Hortonworks Data Platform (HDP), Cloudera Hadoop Distribution (CDH) and Apache Spark.

To enable Sahara in your OpenStack environment that is deployed using Fuel, follow the instructions in

See:

- *Planning a Sahara Deployment* discusses considerations for planning for Sahara.
- *Installing Sahara* gives instructions for installing Sahara and running Apache Hadoop in your OpenStack environment.
- Sahara wiki page for more information about Sahara, including a link to its documentation.

Scheduler, Nova

The Nova scheduler determines how to allocate new VM instances among the configured Compute Nodes. The Filter Scheduler uses a set of filters plus a weighting algorithm to determine the best location for a new VM instance. This scheduler is superior to the "Simpler" or "naive" scheduler that older versions of Fuel deployed.

Fuel 5.0 and Fuel 4.1.1 are the first versions of Fuel that allow you to choose which scheduler you use; see *Choose the Compute Node Scheduler*.

The Filter Scheduler uses the *Overcommit ratio* to compute available resources.

For more details, see:

- Scheduling reference gives reference information about the Filter Scheduler.
- Filter Scheduler provides developer information about the Filter Scheduler.
- Grizzly Scheduling gives details about the Simpler ("naive") Scheduler.

Security groups

A Security Group is a set of IP filter rules that is associated with an instance when it is created to define the networking access to an instance. The default OpenStack Security Group does not allow any network access to Instances (so ping and SSH do not work). Security Groups are associated with a project (tenant); most projects provide a "default" security group that is applied to instances that have no security group defined. The project administrator can either add rules to the default Security Group or define alternate Security Groups from which the user can select when creating an instance.

Avoid creating a Security Group that refers to itself as a source. Such a configuration generates N^2 rules in *iptables* (where N is the number of FMs). This significantly impacts networking performance in large deployments.

For more information about creating and using Security Groups, see:

- Security groups web page gives an overview of Security Groups and instructions for defining them.
- Launch an Instance gives instructions for associating a Security Group with an Instance using the OpenStack dashboard (Horizon).
- Configure access and security for instances gives detailed instructions for adding rules to a Security Group.

SR-IOV

SR-IOV stands for "Single Root I/O Virtualization". It is a specification that allows a PCI device to appear virtually in multiple virtual machines (VMs), each of which has its own virtual function. The specification defines virtual functions (VFs) for the VMs and a physical function for the *hypervisor*. Using SR-IOV in a cloud infrastructure helps reaching higher performance since traffic bypasses the TCP/IP stack in the kernel.

Stateless and Stateful services

A stateless service provides a response to your request and then requires no further attention. To make a stateless service highly available, you need redundant instances that are load balanced. OpenStack services that are stateless include nova-api, nova-conductor, glance-api, keystone-api, neutron-api and nova-scheduler.

A stateful service is one where subsequent requests to the service depend on the results of each previous request. Stateful services are more difficult to manage because a single action typically involves multiple requests, so simply providing additional instances and load balancing does not solve the problem. For example, if the Horizon user interface reset itself every time you went to a new page, it would not be very useful. OpenStack services that are stateful include the OpenStack database and message queue. Implementing high availability for a stateful service depends on whether you choose an active/passive or active/active configuration.

Storage nodes and roles

OpenStack requires block and object storage to be provisioned. Fuel provides the following storage options:

- Cinder LVM provides persistent block storage to virtual machines over the iSCSI protocol
- Swift object stores can be used by Glance to store VM images and snapshots. It may also be used directly by applications.
- Ceph combines object and block storage and can replace either one or both of the above.

You can configure your environment to use either dedicated Storage Nodes or to run the Storage Role on the Compute Nodes.

STP

Spanning Tree Protocol

Swift Object Storage

Swift Object Storage provides a multi-tenant, highly scalable and durable object storage system that can store large amounts of unstructured data at low cost. Fuel can deploy Swift as the storage backend for the *Glance* image service; this is the default in *HA* environments.

Fuel deploys Swift on Controller nodes; it does not provide a separate Swift role that can be installed on other nodes. Support for a separate Swift role is under consideration; see the Separate roles for Swift nodes blueprint for more information.

Some key characteristics of Swift object storage are:

- Each object stored in Swift has its own URL and its own metadata.
- Each object is replicated in the cluster to provide redundancy; the number of replicas is set by the administrator. Each replica is located in as unique a location as possible.
- Libraries are provided for many programming languages; developers can use these libraries to access the data in Swift object stores or they can write directly to the RESTful API. See *Object Storage for Applications*.

See:

- *Glance: Storage for Images* gives information about how to choose the most appropriate storage backend for Glance.
- Swift documentation
- OpenStack Swift Architecture provides a good general introduction to the Swift architecture.

Tagged port

A tagged port (Cisco Trunk Port) is an 802.1q frame from a switch to a server network card used for VLAN networks; untagged ports are used as access ports. Tagged ports are required for the Nova-network VLAN Manager or Neutron with VLAN segmentation.

Telemetry Services

See *Ceilometer (OpenStack Telemetry)*.

Tenant

OpenStack Compute defines a tenant as a group of users.

- In the Dashboard, tenants are represented as projects.
- One tenant can contain zero or more users.
- One user can be associated with one or more tenants.
- A role can be assigned to each tenant and user pairing.

See the OpenStack Compute web page for more information.

Use the keystone command line utility to query and manage a tenant project. The most common commands are:

keystone tenant-list

List all tenants on the system, showing ID, name, and whether they are enabled or disabled.

To list all projects with their ID, name, and whether they are enabled or disabled:

Create a project named new-project:

```
$ keystone tenant-create --name new-project --description 'my new project'
```

vCenter

vCenter is a server that is installed on a Windows or Linux server as a centralized management application to manage virtual machines and *ESXi* hosts. VMWare licenses vCenter as an add-on to the *vSphere* suite.

You use Fuel 5.0 and later to deploy a Mirantis OpenStack environment that utilizes vCenter as a hypervisor over the nova-compute driver and can boot and manage VMs in vCenter.

- For information about planning your OpenStack deployment with vCenter integration, see *Preparing for vSphere Integration*
- For information about deploying Mirantis OpenStack with vCenter integration, see *Deploying vCenter*.
- For background information about how vCenter support is implemented in Mirantis OpenStack, see *VMware vSphere Integration*.

VLAN (Virtual Local Area Network)

A VLAN is a set of ports that form a logical Ethernet segment on an Ethernet switch. The ports of a VLAN form an independent traffic domain in which the traffic generated by the nodes remains within the VLAN. This allows you to use the switch management software to group nodes with related functions into their own separate, logical LAN segments. Fuel deploys Nova-network VLAN Manager topologies, Neutron with VLAN segmentation, and Neutron GRE has VLAN tags on the management, storage, and public networks.

The version of CentOS that Fuel deploys has poor support for VLAN tagged packets so work-arونds are provided to improve the VLAN support when using CentOS; see *VLAN splinters*.

For more information:

- *Neutron with VLAN segmentation and OVS* describes the Neutron VLAN topology. *Neutron VLAN Segmentation Planning* provides examples to help you plan your NIC assignment for this topology.

- *Nova-network VLAN Manager* describes the Nova-network VLAN topology. *Nova-network VLAN Manager* provides examples to help you plan your NIC assignment for this topology.
- *VLAN splinters* gives instructions for setting up VLAN splinters, which improve the performance of VLANs when using the CentOS operating system for your OpenStack nodes.

VMDK

The VMDK driver provides support for the virtual machine disk format used by VMware. This driver is used to enable management of the OpenStack Block Storage volumes on vCenter-managed data stores. Volumes are backed by VMDK files on data stores that use any VMware compatible storage technology such as NFS, iSCSI, FiberChannel, and vSAN. You can use Fuel 5.1 and later to deploy a vCenter integrated Mirantis OpenStack environment that utilizes VMDK support. To enable VMDK driver option, see *vSphere deployment notes*.

For more information, see:

- Virtual Disk Format 5.0
- OpenStack vCenter VMDK configuration documentation

vSphere

vSphere is a VMWare software suite that includes packages such as *vCenter* and *ESXi*. *vCenter* is licensed separately to supplement vSphere.

Zabbix

Zabbix is an open source infrastructure monitoring utility. It is included as an *Experimental package* in Mirantis OpenStack 5.1; Fuel can install Zabbix with your OpenStack environment when the Experimental package is enabled. After deploying your environment with Zabbix enabled, you can access the dashboard using the link that is displayed on the Fuel dashboard.

For Release 5.1, Zabbix can monitor the following:

- Core OpenStack services: *Nova*, *Keystone*, *Neutron*, *Ceilometer (OpenStack Telemetry)*, *Horizon*, *Cinder*, *Glance*, *Swift Object Storage*, and *OVS (Open vSwitch)*.
- Core infrastructure components: *MySQL*, *RabbitMQ*, *HAProxy*, memchached, and libvirtd.
- Operating system statistics: Disk I/O, CPU load, free RAM, et cetera.

For more information, see

- *Zabbix monitoring tool* contains information about planning for Zabbix.
- *Zabbix implementation* describes how Zabbix is implemented in Mirantis OpenStack.
- *Assign a role or roles to each node server* discusses the Fuel screen used to enable the Zabbix server.
- Zabbix documentation.

Zabbix Server

The Zabbix Server role runs the main *Zabbix* processes and hosts the Zabbix database. This role is displayed only when *experimental features* are enabled. Fuel does not allow this role to be assigned to a node that has other roles.