Cloud Management
Software/Platforms:
OpenStack

Cloud Strategy Partners, LLC

Sponsored by: IEEE Educational Activities and IEEE Cloud Computing
This IEEE Cloud Computing tutorial has been developed by Cloud Strategy Partners, LLC. Cloud Strategy Partners, LLC is an expert consultancy firm that specializes in Technology and Strategy relating to Cloud Computing.
This tutorial begins with an overview of cloud management software stacks and moves into the history of OpenStack. We will discuss the OpenStack architecture and OpenStack infrastructure components. Next we will review load balancing in OpenStack as well as examine the OpenStack dashboard (Horizon) and user interfaces. Finally, we will discuss the OpenStack identity (Keystone) and cloud federation.
Outline
In this Lesson, we will cover the following topics

- Overview cloud management software stacks
- History of OpenStack
  - OpenStack community
- OpenStack Architecture
- OpenStack infrastructure components
  - OpenStack Compute (Nova)
  - OpenStack Block Storage (Cinder)
  - OpenStack Object Storage (Swift)
  - OpenStack Image Management (Glance)
  - OpenStack Networking (Neutron)
  - OpenStack Metering (Ceilometer)
  - OpenStack Orchestration (Heat)
- Load balancing in OpenStack
- OpenStack Dashboard (Horizon) and user interfaces
- OpenStack Identity (Keystone) and cloud federation
- Wrap up and Take away

Cloud Middleware ("CloudOS") Stacks
There are several different Cloud Middleware stacks. These "stacks" are often called a "CloudOS", even though they are actually Middleware. Each of the servers in a Cloud is actually running an Operating System such as Linux or Windows, and includes a virtualization component such as Xen, KVM, ESX, or Hyper-V. In this lesson, we are going to study OpenStack.

History of OpenStack
OpenStack came about in a very round-about fashion, as the illustration portrays. This is actually a little known story.

The inspiration for OpenStack came from the University of California, Santa Barbara Eucalyptus Project, which NASA has done significant work on. However, NASA was unable to give their changes back to the Eucalyptus project, as it had become a spinoff company already and was pursuing a "closed source" strategy. As NASA was required by law to make the improvements available somehow (they were owned by the citizens of the United States,
as NASA is funded with taxpayer money), when Rackspace was approached to host a repository, they decided to throw in their code as well and create OpenStack as we largely know it today.

In July 2010 Rackspace Hosting and NASA jointly launched an open-source cloud-software initiative known as OpenStack. The OpenStack project intended to help organizations which offer cloud-computing services running on standard hardware. The first official release, code-named Austin, appeared four months later, with plans to release regular updates of the software every few months. The early code came from the NASA Nebula platform and from the Rackspace Cloud Files platform. In July 2011, Ubuntu Linux developers adopted OpenStack.

**OpenStack Statistics**

The OpenStack Foundation, established September 2012, is an independent body providing shared resources to help achieve the OpenStack Mission by protecting, empowering, and promoting OpenStack software and the community around it. This includes users, developers and the entire ecosystem.

Founded by Rackspace Hosting and NASA, OpenStack has grown to be a global software community of developers collaborating on a standard and massively scalable open source cloud operating system. The OpenStack Foundation promotes the development, distribution and adoption of the OpenStack cloud operating system. As the independent home for OpenStack, the Foundation has already attracted more than 7,000 individual members from 100 countries and 850 different organizations. It has also secured more than $10 million in funding. All of the code for OpenStack is freely available under the Apache 2.0 license. Some OpenStack users include:

- PayPal / eBay
- NASA
- CERN
- Yahoo!
- Rackspace Cloud
- HP Public Cloud
- MercadoLibre.com
- AT&T
- KT (formerly Korea Telecom)
- Deutsche Telekom
- Wikimedia Labs
• Hostalia of Telefónica Group
• SUSE Cloud solution
• Red Hat OpenShift PaaS solution
• Zadara Storage
• Mint Services
• GridCentric

OpenStack Releases and Facts
OpenStack is based on a coordinated 6-month release cycle with frequent development milestones. The creation of OpenStack took an estimated 249 years of effort (COCOMO model). In a nutshell, OpenStack has:

• 64,396 commits made by 1,128 contributors, with its first commit made in May, 2010.
• 908,491 lines of code. OpenStack is written mostly in Python with an average number of source code comments.
• A code base with a long source history.
• Increasing Y-O-Y commits.
• A very large development team comprised of people from around the world.

Main OpenStack Components
OpenStack has a modular architecture with various code names for its components. OpenStack has several shared services that span the three pillars of compute, storage and networking, making it easier to implement and operate your cloud. These services -including identity, image management and a web interface -integrate the OpenStack components with each other as well as external systems to provide a unified experience for users as they interact with different cloud resources. These are

• Nova: OpenStack Compute: Provision and manage large networks of virtual machines
• Swift and Cinder: OpenStack Storage: Object and Block storage for use with servers and applications
• Neutron: OpenStack Networking: Pluggable, scalable, API-driven network and IP management
• Horizon :A Dashboard
• Keystone: Identity Services
• Glance: Image Services
OpenStack and Components Detail

Compute (Nova)
The OpenStack cloud operating system enables enterprises and service providers to offer on-demand computing resources, by provisioning and managing large networks of virtual machines. Compute resources are accessible via APIs for developers building cloud applications and via web interfaces for administrators and users. The compute architecture is designed to scale horizontally on standard hardware.

Object Storage (Swift)
In addition to traditional enterprise-class storage technology, many organizations now have a variety of storage needs with varying performance and price requirements. OpenStack has support for both Object Storage and Block Storage, with many deployment options for each depending on the use case.

Block Storage (Cinder)
OpenStack Block Storage (Cinder) provides persistent block level storage devices for use with OpenStack compute instances. The block storage system manages the creation, attaching and detaching of the block devices to servers.

Networking (Neutron)
Today's data center networks contain more devices than ever before. From servers, network equipment, storage systems and security appliances, many of which are further divided into virtual machines and virtual networks. The number of IP addresses, routing configurations and security rules can quickly grow into the millions. Traditional network management techniques fall short of providing a truly scalable, automated approach to managing these next-generation networks. At the same time, users expect more control and flexibility with quicker provisioning.

Dashboard (Horizon)
OpenStack Dashboard (Horizon) provides administrators and users a graphical interface to access, provision and automate cloud-based resources.

Identity Service (Keystone)
OpenStack Identity (Keystone) 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 like LDAP.

Image Service (Glance)
OpenStack Image Service (Glance) provides discovery, registration and delivery services for disk and server images.
OpenStack supports multiple API’s

OpenStack APIs are compatible with Amazon EC2 and Amazon S3 and thus client applications written for Amazon Web Services can be used with OpenStack with minimal porting effort.

OpenStack Conceptual Architecture

Conceptual Architecture

The OpenStack project as a whole is designed to deliver a massively scalable cloud operating system. To achieve this, each of the constituent services is designed to work together to provide a complete Infrastructure-as-a-Service (IaaS). This integration is facilitated through public application programming interfaces (APIs) that each service offers (and in turn can consume). While these APIs allow each of the services to use another service, it also allows an implementer to switch out any service as long as they maintain the API. These are (mostly) the same APIs that are available to end users of the cloud.

- Dashboard (“Horizon”) provides a web front end to the other OpenStack services
- Compute (“Nova”) stores and retrieves virtual disks (“images”) and associated metadata
- Image (“Glance”)
- Network (“Neutron”) provides virtual networking for Compute.
- Block Storage (“Cinder”) provides storage volumes for Compute.
- Image (“Glance”) can store the actual virtual disk files in the Object Store (“Swift”)
- All the services authenticate with Identity (“Keystone”)
- End users can interact through a common web interface (Horizon) or directly to each service through their API
- All services authenticate through a common source (facilitated through keystone)
- Individual services interact with each other through their public APIs (except where privileged administrator commands are necessary)

Architecture Diagrams Legend

The subsequent slides contain architectural diagrams of OpenStack internals.
Very specific conventions are used to indicate the type of element. Please refer to this diagram when studying the following diagrams.

**OpenStack Compute (Nova)**

OpenStack Compute (Nova) is a cloud computing fabric controller (the main part of an IaaS system). It is written in Python and uses many external libraries such as Eventlet (for concurrent programming), Kombu (for AMQP communication), and SQLAlchemy (for database access). Nova's architecture is designed to scale horizontally on standard hardware with no proprietary hardware or software requirements and provide the ability to integrate with legacy systems and third party technologies. It is designed to manage and automate pools of computer resources and can work with widely available virtualization technologies, as well as bare metal and high-performance computing (HPC) configurations. KVM and XenServer are available choices for hypervisor technology, together with Hyper-V and Linux container technology such as LXC. In addition to different hypervisors, OpenStack runs on ARM.

Popular Use Cases:
- Service providers offering an IaaS compute platform or services higher up the stack
- IT departments acting as cloud service providers for business units and project teams
- Processing big data with tools like Hadoop
- Scaling compute up and down to meet demand for web resources and applications
- High-performance computing (HPC) environments processing diverse and intensive workloads.

Nova is the most complicated and distributed component of OpenStack. A large number of processes cooperate to turn end user API requests into running virtual machines. Below is a list of these processes and their functions:

- **nova-api**: accepts and responds to end user compute API calls. It supports OpenStack Compute API, Amazon's EC2 API and a special Admin API (for privileged users to perform administrative actions). It also initiates most of the orchestration activities (such as running an instance) as well as enforces some policy (mostly quota checks).

- The nova-compute process is primarily a worker daemon that creates and terminates virtual machine instances via hypervisor's APIs (XenAPI for XenServer/XCP, libvirt for KVM or QEMU, VMwareAPI for VMware, etc.). The process by which it does so is fairly complex but the basics are simple: accept actions from the queue and then perform a series of system commands (like launching a KVM instance) to carry them out while updating state in the database.
Nova-volume manages the creation, attaching and detaching of Z volumes to compute instances (similar functionality to Amazon’s Elastic Block Storage). It can use volumes from a variety of providers such as iSCSI or Rados Block Device in Ceph. A new OpenStack project, Cinder, will eventually replace nova-volume functionality. In the Folsom release, nova-volume and the Block Storage service will have similar functionality.

The nova-network worker daemon is very similar to nova-compute and nova-volume. It accepts networking tasks from the queue and then performs tasks to manipulate the network (such as setting up bridging interfaces or changing iptables rules). This functionality is being migrated to Neutron, a separate OpenStack project. In the Folsom release, much of the functionality will be duplicated between nova-network and Neutron.

The nova-schedule process is conceptually the simplest piece of code in OpenStack Nova: it takes a virtual machine instance request from the queue and determines where it should run (specifically, which compute server host it should run on).

The queue provides a central hub for passing messages between daemons. This is usually implemented with RabbitMQ today, but could be any AMQP message queue (such as Apache Qpid). New to the Folsom release is support for ZeroMQ.

The SQL database stores most of the build-time and runtime state for a cloud infrastructure. This includes the instance types that are available for use, instances in use, networks available and projects. Theoretically, OpenStack Nova can support any database supported by SQL-Alchemy but the only databases currently being widely used are SQLite3 (only appropriate for test and development work), MySQL and PostgreSQL.

Nova also provides console services to allow end users to access their virtual instance’s console through a proxy. This involves several daemons (nova-console, nova-novncproxy and nova-consoleauth).

Nova interacts with many other OpenStack services: Keystone for authentication, Glance for images and Horizon for web interface. The Glance interactions are central. The API process can upload and query Glance while nova-compute will download images for use in launching images.

OpenStack Block Storage (Cinder)
Block storage volumes are fully integrated into OpenStack Compute and the Dashboard allowing for cloud users to manage their own storage needs. In addition to local Linux server storage, it can use storage platforms including Ceph, CloudByte, Coraid, EMC (VMAX and
VNX), GlusterFS, IBM Storage (Storwize family, SAN Volume Controller, and XIV Storage System), Linux LIO, NetApp, Nexenta, Scality, SolidFire and HP (Store Virtual and StoreServ 3Par families). Block storage is appropriate for performance sensitive scenarios such as database storage, expandable file systems, or providing a server with access to raw block level storage. Snapshot management provides powerful functionality for backing up data stored on block storage volumes. Snapshots can be restored or used to create a new block storage volume. A few points on OpenStack Block Storage:

OpenStack provides persistent block level storage devices for use with OpenStack compute instances.

The block storage system manages the creation, attaching and detaching of the block devices to servers. Block storage volumes are fully integrated into OpenStack Compute and the Dashboard allowing for cloud users to manage their own storage needs. In addition to using simple Linux server storage, it has unified storage support for numerous storage platforms including Ceph, NetApp, Nexenta, SolidFire, and Zadara. Block storage is appropriate for performance sensitive scenarios such as database storage, expandable file systems, or providing a server with access to raw block level storage.

Snapshot management provides powerful functionality for backing up data stored on block storage volumes. Snapshots can be restored or used to create a new block storage volume.

Cinder separates out the persistent block storage functionality that was previously part of OpenStack Compute (in the form of nova-volume) into its own service. The OpenStack Block Storage API allows for manipulation of volumes, volume types (similar to compute flavors) and volume snapshots. cinder-api accepts API requests and routes them to cinder-volume for action.

cinder-volume acts upon the requests by reading or writing to the Cinder database to maintain state, interacting with other processes (like cinder-scheduler) through a message queue and directly upon block storage providing hardware or software. It can interact with a variety of storage providers through a driver architecture. Currently, there are drivers for IBM, SolidFire, NetApp, Nexenta, Zadara, linux iSCSI and other storage providers.

Much like nova-scheduler, the cinder-scheduler daemon picks the optimal block storage provider node to create the volume on.

Cinder deployments will also make use of a messaging queue to route information between the cinder processes as well as a database to store volume state. Like Neutron, Cinder will mainly interact with Nova, providing volumes for its instances.
OpenStack Object Storage (Swift)

OpenStack Object Storage (Swift) is a scalable redundant storage system. Objects and files are written to multiple disk drives spread throughout servers in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters scale horizontally simply by adding new servers. Should a server or hard drive fail, OpenStack replicates its content from other active nodes to new locations in the cluster. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used. Object Storage is ideal for cost effective, scale-out storage. It provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention. Block Storage allows block devices to be exposed and connected to compute instances for expanded storage, better performance and integration with enterprise storage platforms, such as NetApp, Nexenta and SolidFire.

A few details on OpenStack’s Object Storage

OpenStack provides redundant, scalable object storage using clusters of standardized servers capable of storing petabytes of data.

Object Storage 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. Having no central "brain" or master point of control provides greater scalability, redundancy and durability.

Objects and files are written to multiple disk drives spread throughout servers in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster.

Storage clusters scale horizontally simply by adding new servers. Should a server or hard drive fail, OpenStack replicates its content from other active nodes to new locations in the cluster. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used in lieu of more expensive equipment. The swift architecture is very distributed to prevent any single point of failure as well as to scale horizontally.

It includes the following components:
Proxy server (swift-proxy-server) accepts incoming requests via the OpenStack Object API or just raw HTTP. It accepts files to upload, modifications to metadata or container creation. In addition, it will also serve files or container listing to web browsers. The proxy server may utilize an optional cache (usually deployed with memcache) to improve performance. Account servers manage accounts defined with the object storage service.

Container servers manage a mapping of containers (i.e. folders) within the object store service.

Object servers manage actual objects (i.e. files) on the storage nodes.

There are also a number of periodic processes which run to perform housekeeping tasks on the large data store. The most important of these is the replication services, which ensures consistency and availability through the cluster. Other periodic processes include auditors, updaters and reapers.

Authentication is handled through configurable WSGI middleware (which will usually be Keystone).

---

**OpenStack Image Management (Glance)**

OpenStack Image Service (Glance) provides discovery, registration and delivery services for disk and server images. Stored images can be used as a template. They 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 from existing servers

Snapshots can also 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, also known as AMI)
VHD (Hyper-V)
VDI (VirtualBox)
qcow2 (Qemu/KVM)
VMDK (VMWare)

OVF (VMWare, others)
The Glance architecture has stayed relatively stable since the Cactus release. The biggest architectural change has been the addition of authentication, which was added in the Diablo release. Just as a quick reminder, Glance has four main parts to it:

1. glance-api accepts Image API calls for image discovery, image retrieval and image storage.

2. glance-registry stores, processes and retrieves metadata about images (size, type, etc.).

3. A database to store the image metadata. Like Nova, you can choose your database depending on your preference (but most people use MySQL or SQLite).

4. A storage repository for the actual image files. In the diagram above, Swift is shown as the image repository, but this is configurable. In addition to Swift, Glance supports normal filesystems, RADOS block devices, Amazon S3 and HTTP. Be aware that some of these choices are limited to read-only usage.

There are also a number of periodic processes which run on Glance to support caching. The most important of these is the replication services, which ensures consistency and availability through the cluster. Other periodic processes include auditors, updaters and reapers. As you can see from the diagram in the Conceptual Architecture section, Glance serves a central role to the overall IaaS picture. It accepts API requests for images (or image metadata) from end users or Nova components and can store its disk files in the object storage service, Swift.

OpenStack Networking (Neutron, formerly Quantum) is a pluggable, scalable and API-driven system for managing networks and IP addresses. Like other aspects of the cloud operating system, it can be used by administrators and users to increase the value of existing data center assets. OpenStack Networking ensures the network will not be the bottleneck or limiting factor in a cloud deployment and gives users real self-service, even over their network configurations.
OpenStack Networking (Neutron)

OpenStack Networking is a system for managing networks and IP addresses. Like other aspects of the cloud operating system, it can be used by administrators and users to increase the value of existing data center assets. OpenStack Networking ensures the network will not be the bottleneck or limiting factor in a cloud deployment and gives users real self-service, even over their network configurations.

OpenStack Neutron provides networking models for different applications or user groups. Standard models include flat networks or VLANs for separation of servers and traffic. OpenStack Networking manages IP addresses, allowing for dedicated static IPs or DHCP. Floating IPs allow traffic to be dynamically rerouted to any of your computer resources, which allows you to redirect traffic during maintenance or in the case of failure. Users can create their own networks, control traffic and connect servers and devices to one or more networks. Administrators can take advantage of software-defined networking (SDN) technology like OpenFlow to allow for high levels of multi-tenancy and massive scale. OpenStack Networking has an extension framework allowing additional network services, such as intrusion detection systems (IDS), load balancing, firewalls and virtual private networks (VPN) to be deployed and managed.

Networking Capabilities
OpenStack provides flexible networking models to suit the needs of different applications or user groups. Standard models include flat networks or VLANs for separation of servers and traffic.

OpenStack Networking manages IP addresses, allowing for dedicated static IPs or DHCP. Floating IPs allow traffic to be dynamically re-routed to any of your computer resources, which allows you to redirect traffic during maintenance or in the case of failure. Users can create their own networks, control traffic and connect servers and devices to one or more networks.

The pluggable backend architecture lets users take advantage of commodity gear or advanced networking services from supported vendors.

Administrators can take advantage of software-defined networking (SDN) technology like OpenFlow to allow for high levels of multi-tenancy and massive scale.

OpenStack Networking has an extension framework allowing additional network services, such as intrusion detection systems (IDS), load balancing, firewalls and virtual private networks (VPN) to be deployed and managed.
Neutron Topology Model

Neutron provides "network connectivity as a service" between interface devices managed by other OpenS.

neutron-server accepts API requests and then routes them to the appropriate Neutron plug-in for action.

Neutron plug-ins and agents perform the actual actions such as plugging and unplugging ports, creating n

The common agents are L3 (layer 3), DHCP (dynamic host IP addressing) and the specific plug-in agent. Most Neutron installations will also make use of a messaging queue to route information between the ne Neutron will interact mainly with Nova, where it will provide networks and connectivity for its instances.

Network Management with Neutron

Starting from the Folsom release (September 2012), network management is performed by the independent component Neutron (previously called Quantum). Previously network management has been performed by the Network Controller in Nova

Neutron network manager adds the following new functionalities: Give cloud tenants an API to build rich networking topologies, and configure advanced network policies in the cloud; e.g. create multi-tier web application topology Enable innovation plugins (open and closed source) that introduce advanced network capabilities; e.g. use L2-in-L3 tunneling to avoid VLAN limits, provide end-to-end QoS guarantees, used monitoring protocols like NetFlow Allows building advanced network services (open and closed source) that plug into Openstack tenant networks; e.g. VPN-aaS, firewall-aaS, IDS-aaS, data-center-interconnect-aaS Logically Neutron (and Nova) supports two types of IP address: Fixed which are associate with virtual machine instance at creation and remain associated till termination Floating which can be dynamically attached/detached to/from a running virtual machine instance at run-time from Horizon or using the nova-api

For fixed IPs, Neutron (and Nova) supports following three modes of networking: Flat mode provides each virtual machine instance with a fixed IP associated with a default network bridge. This can be manually configured before an instance is booted. This mode is currently applicable to linux operating systems, which manage network configurations in /etc/network/interfaces (Debian & Ubuntu). Flat DHCP mode improves upon Flat mode by creating a DHCP server to provide fixed IPs to virtual machine instances. VLAN DHCP Mode
is the default networking mode in which Nova creates a vlan and bridge for each project. Virtual machine instances in the project are allocated a private IP address from range of IPs. This private IP address is accessible only within the vlan. Users can access these instances by using a special VPN instance called 'cloudpipe' which uses a certificate and key to create a VPN (Virtual Private Network).

**Neutron Networking Example**
This slide illustrates how Neutron based software networking can be applied to result in many different networking topologies for the user. Note the ability to create virtual switch components as well as multiple network segments.

**OpenStack Metering (Ceilometer)**
The Ceilometer project aims to become the infrastructure to collect measurements within OpenStack so that no two agents would need to be written to collect the same data. Its primary targets are monitoring and metering, but the framework should be easily expandable to collect for other needs. To that effect, Ceilometer should be able to share collected data with a variety of consumers.

An agent runs on each OpenStack node (Bare Metal machine) and harvests the data locally. If a meter is available from the existing OpenStack component it should be used. A standalone ceilometer agent implements the meters that are not yet available from the existing OpenStack components. A storage daemon communicates with the agents to collect their data and aggregate them. The agents collecting data are authenticated to avoid pollution of the metering service. The data is sent from agents to the storage daemon via a trusted messaging system (RabbitMQ?). The data/messages exchanged between agents and the storage daemon use a common messages format. The content of the storage is made available thru a REST API providing aggregation. The message queue is separate from other queues (such as the nova queue). The messages in queue are signed and non repudiable.

**OpenStack Orchestration (Heat)**
Heat is the main project in the OpenStack Orchestration program. It implements an orchestration engine to launch multiple composite cloud applications based on templates in the form of text files that can be treated like code. A native Heat template format is evolving, but Heat also endeavours to provide compatibility with the AWS CloudFormation template format, so that many existing CloudFormation templates can be launched on OpenStack. Heat provides both an OpenStack-native ReST API and a CloudFormation-compatible Query API.
A Heat template describes the infrastructure for a cloud application in a text file that is readable and writable by humans, and can be checked into version control, diffed, &c. Infrastructure resources that can be described include: servers, floating ips, volumes, security groups, users, etc.

Heat also provides an autoscaling service that integrates with Ceilometer, so you can include a scaling group as a resource in a template.

Templates can also specify the relationships between resources (e.g. this volume is connected to this server). This enables Heat to call out to the OpenStack APIs to create your entire infrastructure in the correct order to completely launch your application. Heat manages the whole lifecycle of the application - when you need to change your infrastructure, simply modify the template and use it to update your existing stack. Heat knows how to make the necessary changes. It will delete all of the resources when you are finished with the application, too.

Heat primarily manages infrastructure, but the templates integrate well with software configuration management tools such as Puppet and Chef. The Heat team is working on providing even better integration between infrastructure and software.

**Load Balancing in OpenStack**

A load balancer is a logical device. It is used to distribute workloads between multiple back-end systems or services called nodes, based on the criteria defined as part of its configuration.

Atlas Load Balancer is a new component in OpenStack that allows users to apply load balancing to an existing configuration instead of adding a custom implementation for a particular application.

Designed to provide functionality similar to Amazon’s ELB (Elastic Load Balancing) and provides a RESTful API for users.

A virtual IP is an Internet Protocol (IP) address configured on the load balancer for use by clients connecting to a service that is load balanced. Incoming connections and requests are distributed to back-end nodes based on the configuration of the load balancer.
A health monitor is a feature of each load balancer. It is used to determine whether or not a back-end node of the load balancer is usable for processing a request. The load balancing service supports two health monitoring modes: passive and active.

**OpenStack Dashboard (Horizon)**

The design allows for third party products and services, such as billing, monitoring and additional management tools. Service providers and other commercial vendors can customize the dashboard with their own brand.

The dashboard is just one way to interact with OpenStack resources. Developers can automate access or build tools to manage their resources using the native OpenStack API or the EC2 compatibility API.

**OpenStack Identity (Keystone)**

OpenStack Identity (Keystone) 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 like LDAP. It supports multiple forms of authentication including standard username and password credentials, token-based systems, and Amazon Web Services log in credentials such as those used for EC2.

Additionally, the catalog provides a query-able 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.

The OpenStack Identity Service enables administrators to:

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

The OpenStack Identity Service enables users to:

- List the services to which they have access
- Make API requests
- Log into the web dashboard to create resources owned by their account
OpenStack Identity Service - Keystone
Keystone provides a single point of integration for OpenStack policy, catalog, token and authentication. Keystone handles API requests as well as providing configurable catalog, policy, token and identity services.

Identity service provides validation of user’s authorization credentials, Roles, Tenants and associated metadata. Token service validates tokens that are used by users or tenants for authentication. Endpoint discovery and endpoint registry services are provided by the Catalog service. Rule based authorization is provided by the Policy service. The Keystone service can use various format of credentials and storages such as file, SQL, PAM or LDAP.

Each Keystone function has a pluggable backend which allows different ways to use the particular service. Most support standard backends like LDAP or SQL, as well as Key Value Stores (KVS). Most people will use this as a point of customization for their current authentication services.

Keystone Identity Server - Sequences
The diagram in this slide shows the Sequences for how Keystone is used.

Wrap up and Take away
OpenStack is one of popular cloud management platforms; it has flexible management functionality and can be easy extended by user. OpenStack is included into the major Linux installations such as Ubuntu, Fedora, CentOS and has growing community.