

Lomonosov Moscow State University

Faculty of Computational Mathematics and Cybernetics

Department of Automation Systems of Computer Systems

OpenStack platform

Vasilenko Anatoly

521 group

[](https://en.wikipedia.org/wiki/File:The_OpenStack_logo.svg)

Moscow, 2015

1. Contents

[1 Contents 2](#_Toc433055781)

[2 Introduction 3](#_Toc433055782)

[3 Cloud computing 3](#_Toc433055783)

[4 OpenStack capabilities 4](#_Toc433055784)

[5 History 5](#_Toc433055785)

[6 Governance 5](#_Toc433055786)

[7 Components 5](#_Toc433055787)

[7.1 Compute (Nova) 6](#_Toc433055788)

[7.2 Image Service (Glance) 7](#_Toc433055789)

[7.3 Object Storage (Swift) 7](#_Toc433055790)

[7.4 Dashboard (Horizon) 8](#_Toc433055791)

[7.5 Identity Service (Keystone) 8](#_Toc433055792)

[7.6 Networking (Neutron) 8](#_Toc433055793)

[7.7 Block Storage (Cinder) 9](#_Toc433055794)

[7.8 Orchestration (Heat) 10](#_Toc433055795)

[7.9 Telemetry (Ceilometer) 10](#_Toc433055796)

[7.10 Database (Trove) 10](#_Toc433055797)

[7.11 Elastic Map Reduce (Sahara) 11](#_Toc433055798)

[7.12 Bare Metal Provisioning (Ironic) 11](#_Toc433055799)

[7.13 Multiple Tenant Cloud Messaging (Zaqar) 11](#_Toc433055800)

[7.14 Shared File System Service (Manila) 12](#_Toc433055801)

[7.15 DNSaaS (Designate) 12](#_Toc433055802)

[7.16 Security API (Barbican) 12](#_Toc433055803)

[7.17 Components Relationships 13](#_Toc433055804)

[8 Cloud Example 14](#_Toc433055805)

[9 Deployment models 14](#_Toc433055806)

[10 OpenStack Features 15](#_Toc433055807)

[11 Conclusion 15](#_Toc433055808)

[12 References 16](#_Toc433055809)

1. Introduction

OpenStack is a free and open-source cloud-computing software platform. Users primarily deploy it as an “infrastructure as a service” (IaaS), but it can be used to provide all three types of cloud computing services: IaaS, PaaS (“platform as a service”) and SaaS (“software as a service”).

The technology consists of a group of interrelated projects that control pools of processing, storage, and networking resources in a data center. Users can manage their resource using a web-based application, command-line tools or through a RESTful API, that supports JSON and XML formats. OpenStack was released under the terms of the Apache License, most of it has been written in Python.

OpenStack began in 2010 as a joint project of Rackspace Hosting and of NASA. Now it is a product managed by the OpenStack Foundation, a non-profit corporate entity established in September 2012 to promote OpenStack software and its community. More than 500 companies have joined the project, including Arista Networks, AT&T, AMD, Cisco, Dell, Ericsson, Fujitsu, Go Daddy, Google, Hewlett-Packard, Huawei, IBM, Intel, Juniper Networks, NEC, Oracle, Pure Storage, Red Hat, and VMware. OpenStack project has its own site at [www.openstack.org](http://www.openstack.org).

The OpenStack community collaborates around a six-month, time-based release cycle with frequent development milestones. During the planning phase of each release, the community gathers for an OpenStack Design Summit to facilitate developer working-sessions and to assemble plans.

The summit in May 2014 in Atlanta drew 4,500 attendees — a 50% increase from the Hong Kong Summit six months earlier.

1. Cloud computing

Cloud computing is a concept for enabling ubiquitous, convenient, on-demand access to a shared pool of configurable computing resources. Cloud computing provide users (people and companies) with various capabilities to store and process their data in third-party data centers. This is part of the concept of division of labour: sharing of resources to achieve synergy and economies of scale. At the basis of cloud computing is the concept of shared services.

Cloud computing, or “the cloud”, focuses on maximizing the effectiveness of the shared resources. Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand. This approach helps maximize the use of computing power while reducing the overall cost of resources by using less electric power, air conditioning, rack space, replacement of broken details etc. to maintain the system. With cloud computing, multiple users can access a single server to retrieve and update their data without purchasing licenses for different applications, because people will pay not per user, but per installed instances of application.

The term "moving to cloud" also refers to an organization moving away from its own dedicated hardware to the use of a shared cloud infrastructure. This will decrease CAPEX, because the company does not need to buy hardware any more, and will not need to store it and spend money (OPEX) for its supporting when it is not in use.

The present availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture, and autonomic and utility computing have led to a growth in cloud computing. Companies can scale up if loading is increasing and then scale down again if the loading decrease. They can do it event very fast according to the daily changes. Cloud vendors are experiencing growth rates of 50% per year.

Cloud Computing increases following characteristics: agility/availability, lowering cost (reductions based on the division of labour), device and location independence, maintenance (for cloud computing it is much easier because of orchestration and dashboard systems), multitenancy, performance, productivity (resources can be accessed simultaneously), reliability (based on redundancy, which is much cheaper for clouds), scalability and security (can improve because of centralization of data, but it is harder to make secure distributed systems).

Cloud computing can handle three service models:

1. Infrastructure as a service (IaaS). This is the name for services provided by companies in the form of physical or more often virtual machines. Cloud providers typically bill IaaS services on a utility computing basis: cost reflects the amount of resources allocated and consumed.
2. Platform as a service (PaaS). In the PaaS models, cloud providers deliver a computing platform, typically including operating system, programming-language execution environment, database, and web server. Examples of such service are Google App Engine.
3. Software as a service (SaaS). Cloud providers manage the infrastructure and platforms that run the applications. SaaS is sometimes referred to as "on-demand software" and is usually priced on a pay-per-use basis or using a subscription fee. Often cloud users access the software from cloud clients. Examples of such services are GitHub project or Google Sheets.

Cloud can be one of three types: private (for internal use of organization), community (share infrastructure between several organizations with common concerns like security), public (for hosting services that will be opened to public), hybrid cloud (the composition of private, public or community clouds), distributed cloud, inter cloud and multi cloud.

1. OpenStack capabilities

OpenStack can manage different operating systems; can cooperate with different hypervisors and even Linux containers. The main goal of this project it to create community and open source product to manage clouds in all kinds of datacenters. In the world already exists some similar projects, but some of them are more specific for some purposes and some of them are not so well thought-out. Apart from OpenStack exists OpenShift, Cloud Foundry, AppScale, CloudStack, Eucalyptus, Flexiant Limited, Nimbus, OpenNebula, OpenQRM, OpApp, Jelastic, PetiteCloud. In our days, the leader is Amazon Web Services provider, but its cloud platform is used for internal usage only.

Anyway, the Amazon platform is one of the best and one of the most widespread, so it will be good to take it into account, while developing OpenStack. 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. This is very important because it give a huge lift up for OpenStack on the way to conquer the world.

OpenStack is five-year old and already created its own ecosystem. There is many services around it, vendors, marketing, developers, meetups, conferences and other activities.

1. History

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 offer cloud-computing services running on standard hardware. The community's 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 NASA's Nebula platform as well as from Rackspace's Cloud Files platform.

In 2011, developers of the Ubuntu Linux distribution adopted OpenStack with an unsupported technology preview of the OpenStack Bexar release for Ubuntu 11.04. Ubuntu's sponsor Canonical then introduced full support for OpenStack clouds, starting with OpenStack's Cactus release.

OpenStack became available in Debian Sid from the OpenStack "Cactus" release in 2011, and the first release of Debian including OpenStack was Debian 7.0 Wheezy, including OpenStack 2012.1 Essex.

And it started to grow up as a snowball:

1. In October 2011, SUSE announced the preview of the OpenStack.
2. In 2012, Red Hat announced a preview of their OpenStack distribution.
3. In July 2013, NASA announced that they are going to focus on the use of public clouds.
4. In December 2013, Oracle announced that it has joined OpenStack and will implement it in its products.
5. In May 2014, HP announced HP Helion and released a preview of HP Helion OpenStack Community.
6. At the 2014 Interop and Tech Field Day, software defined networking (SDN) was demonstrated in the context of using OpenStack, showing capabilities of automation.

The last one was one of the most important of all events, because it proves that OpenStack is a live project, which has its own place in our world and can be used jointly with other progressive IT technologies.

1. Governance

OpenStack is governed by a non-profit foundation and its board of directors, a technical committee and a user committee. The board of directors is made up of eight members from each of the eight platinum sponsors, eight members from the 24 defined maximum allowed Gold sponsors, and eight members elected by the Foundation individual members [14].

The Foundation's stated mission is “*providing shared resources to help achieve the OpenStack Mission by Protecting, Empowering, and Promoting OpenStack software and the community around it, including users, developers and the entire ecosystem*”. Though, it is not about the development of the software, which is managed by the technical committee – “*an elected group that represents the contributors to the project, and has oversight on all technical matters*” [15].

1. Components

OpenStack has a modular architecture with various code names for its components. This is one of its main benefits, because each component provide certain services. At the same time, components are isolated from each other and uses RESTful API for communication. Therefore, this leads us to very important abilities:

1. Separate development
2. API unification
3. Efficient utilization by user, because you can use only those components, which is required by your applications.

The core components of the OpenStack are Compute (Nova), Object and Block Storage (Swift and Cinder) and Networking (Neutron). These are services, which were needed for every instance of datacenter based on OpenStack.

OpenStack shared services are Identity service (Keystone), Image service (Glance), Telemetry service (Ceilometer), Orchestration service (Heat) and Database service (Trove). In addition, there is some other services exists in OpenStack project. The important point here is that new services can be added without essential changes to other services. It make a lot of space for a maneuver. Shared 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 view for users as they process different cloud resources.

After studying the developer pages of the OpenStack components, it was striking how non-overlapping the teams of developers are. I have seen it, because they form their code and write supporting information in different styles even within the scope of following the common pattern. This is one of the many examples, how module architecture work. Each command can concentrate on certain problem and solve it faster and better.

* 1. Compute (Nova)

OpenStack Compute (Nova) is a cloud computing fabric controller, which is the main part of an IaaS system, and enables service providers to offer on-demand computing resources, by managing large amounts of virtual machines. It is designed to manage and automate pools of computer resources and can work with widely available virtualization technologies (KVM, VMware, Xen, Hyper-V, LXC (Linux containers)), as well as bare metal and high-performance computing (HPC) configurations. KVM and XenServer are the most popular choices for hypervisor technology. LXC is supported for users who wish to minimize virtualization overhead and achieve greater performance, it is still not fully supported and have some particular drawbacks related to the Linux containers, but LXC is young technology and still in the process of active development. As Nova is used with different hypervisors, it makes no difference between software it is running, what is the part of the OpenStack flexibility. It even supports different hardware architectures.

It is written in Python, very comfortable programing language, but have many external libraries such as Eventlet, Kombu, and SQLAlchemy. 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.

Nova is such an important component of the OpenStack, that Nova developer teams meets weekly, they have their own ecosystem and gives a lot of significance to the code-review, saying - “Code reviews are the life blood of the Nova developer community”. They are also very fond of new developers, which are getting involved into the Nova project and tries to help them with everything they can.

* 1. Image Service (Glance)

Glance provides discovery, registration, and delivery services for disk and server images. Stored images can be used as a template for starting new instances (operating systems or containers with services) in future. In addition, templates can be created beforehand by administrators and been provided to the users. Glance 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. Templates may have wide variety of formats: raw, vhd (hyper-V), VDI (Virtual Box (Oracle)), qcow2 (QEMU/KVM), VMDK (VMWare), OVF. The Image Service API providing a standard REST interface lets clients streaming the images between locations and servers.

Glance adds many enhancements to existing legacy infrastructures. For example, if integrated with VMware, Glance introduces advanced features to the vSphere family such as, vMotion. This feature is the live migration of a running VM, from one physical server to another, without service interruption. Thus, it enables a dynamic and automated self-optimizing datacenter

OpenStack's image is an operating system installed on a virtual machine (VM). If a developer adds a variation to an image (as a configuration job), the result is an instance of that image. Subsequently, that instance is an image that developers can add more variations to.

Glance - is a compute module, it does not store images, but holds their metadata from Swift (Object Storage) or a storage backend datastore. Other modules must communicate with the images metadata through Glance. Nova can change some information about the images, but all its commands will be addressed to Glance, Glance is the only module that can add, delete, share, or duplicate images.

* 1. Object Storage (Swift)

Object Storage is a scalable redundant storage system, a lot of storage data can be stored and retrieved with a simple API. 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. Usually there is three copies of the data in each datacenter, but this value is configurable. It must be clear, that object storage is not an ordinary file system, but a storage having no central "brain" or master point.

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 unreliable hard drives and servers can be used. Cost efficiency is a very valuable parameter in datacenters.

Swift is built to be scalable and optimized for durability, availability and concurrency. Its main usage is for storing an unstructured data.

In August 2009, Rackspace started the development of the predecessor to Swift, as a complete replacement for the *Cloud Files* product. The initial development team consisted of nine developers. SwiftStack, an object storage software company, is currently the leading developer for Swift. This is a classic example of benefit gained from modular architecture and well-designed API.

* 1. Dashboard (Horizon)

OpenStack Dashboard provides administrators and users a graphical interface to access, manage, and automate cloud-based resources. The design permits third party products and services, such as billing, monitoring, and additional management tools.

The dashboard is one of several ways users can interact with OpenStack resources. Developers can automate access or build tools to manage resources using the native OpenStack API or the EC2 compatibility API (An API used by Amazon – the biggest hosting company in the world).

Horizon consists of three central dashboards, a “User Dashboard”, a “System Dashboard”, and a “Settings” dashboard. Between these three they cover the core OpenStack applications.

* 1. Identity Service (Keystone)

Keystone is an authentication system for the datacenter users and regulate user’s rights for different OpenStack services they can access. It acts as a common authentication system and can integrate with different backend services like LDAP for keeping its metadata. It supports different types of authentication: username and password credentials, token-based systems, ssh keys and Amazon Web Services logins (AWS-style logins).

Keystone can provide list of different services allowed to the user, so third-party programs can determine which services is allowed for them to use.

For administrator Keystone enables to configure users and systems policies, create users and grant them permissions relative to different OpenStack services. Users is allowed to manage resources to their discretion.

* 1. Networking (Neutron)

OpenStack Networking (Neutron) is a system for managing networks and IP addresses, in general, it provides networking as a service between interfaces, and enables other OpenStack components to configure itself. It is very unusual for old IT world to freely enable networking configuration, but it is as it is, in new IT innovations, it became perfectly normal. Neutron main mission is to implement network services and to provide scalable and technology-agnostic network abstraction.

OpenStack Networking enables to create different network topologies and provide different services for various applications or user groups. It is not based on VLANs as standard models, so is it easily configurable. OpenStack Networking manages IP addresses, allowing sticking static IP address to the machine or to assign dynamic IP address via DHCP. Dynamic IP addresses let traffic to be dynamically rerouted to any resources in the IT infrastructure.

Users can create their own networks, control traffic, connect servers and devices to one or more networks and configure advanced network polices. OpenStack Networking provides an extension framework that can be used to manage additional services such as intrusion detection systems (IDS), firewalls, virtual private networks (VPN), load balancing, etc.

Administrators can use software defined networking (SDN) technology like OpenFlow to support high levels of multi-tenancy and massive scale. OpenStack Neutron, with its plugin architecture, provides the ability to integrate SDN controllers into the OpenStack. This integration of SDN controllers into Neutron using plugins provides centralized management, and facilitates the network programmability. With this integration of OpenStack Neutron and SDN controllers, the changes to the network and network elements triggered by the OpenStack user, are translated into Neutron APIs, and handled by neutron plugins, applying the changes to interfaces.

The main benefit of the OpenStack Networking is that it is pluggable, scalable and API-driven, both: administrators and users can use it. Scalability is very important in datacenters in our days, because the number of IP addresses, routing configurations and security rules can quickly grow into the millions.

* 1. Block Storage (Cinder)

OpenStack Block Storage 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 different servers by means of different hypervisors. Block storage volumes are fully integrated into other OpenStack services allowing cloud users to manage their own storage needs by means of graphical interface or using console commands. In addition to local Linux server storage (root file system of the operating system, which is usually small in datacenters), it can use different storage platforms like Ceph, NetApp, Nexenta, SolidFire, xZadara, EMC (ScaleIO, VMAX and VNX), IBM Storage, SAN Volume Controller, Pure Storage, etc. Block storage is appropriate for performance sensitive operations, such as database storage or providing a server with access to raw block level storage. Snapshot management provides functionality for backing up data stored on block storage volumes and restore them.

Cinder has some main features, which makes it very useful in context of OpenStack:

1. Redundancy makes it very reliable, and in case of some breakage, the system will restore the data. User can configure the replication parameter. Therefore, it makes no need for special RAID 0-7 systems to be used any more.
2. No limits for the size of the storage. Several hard drives can be represented to the clients system as one single block device.
3. Absence of central database, leads to the scalability benefits.
4. Detection of drive failures, preventing data corruption.
5. Enable of direct access to the data from the browser.
6. Real-time monitoring of the client activities concerned the block storage.
7. The access to the block storage can be limited for different users.

In total, Cinder virtualizes pools of block storage devices and provides them to the end users without requiring any knowledge of where their storage is actually deployed or on what type of device.

* 1. Orchestration (Heat)

To understand the main idea of the Heat, firstly we must discus what is “orchestration”. Orchestration is the concept of unified coordination, management and handling exceptions of the complex computer system as a whole. It is something like implicitly autonomic control of the system. However, in reality, after taking into account technical details, orchestration is only the effect achieved by automation of the computer system processes. In cloud computing, orchestration consists of three parts:

1. Composing of architecture, tools and processes used to deliver a defined service
2. Stitching of software and hardware component to deliver a defined service
3. Connecting and automating of workflows to deliver a defined service

Heat is a service to orchestrate many different cloud applications using templates in form of the text files, through an OpenStack native REST API and a CloudFormation compatible Query API. To set up Heat you must describe the infrastructure in a text file, its resources and relationships between them (this will enable Heat to create the entire infrastructure in the correct order). As Heat always knows the infrastructure, it is capable of updating it in a correct way after changing the template. As an additional feature Heat configuration files is compatible with configuration management tools such as Puppet and Chef. It enables to write the unified configuration files for setting up the infrastructure and to set up applications inside infrastructure. Through integration with the Ceilometer (telemetry service), the orchestration engine can also perform auto-scaling of certain infrastructure elements.

Heat mission consist of creation a human and machine accessible service for managing the entire lifecycle of infrastructure and services within the cloud.

* 1. Telemetry (Ceilometer)

The OpenStack Telemetry Service (Ceilometer) aggregates usage and performance data across the services settled up in an OpenStack cloud. This enables monitoring the usage of the cloud across the OpenStack services and allows cloud operators to view metrics globally or by individual deployed resources.

This service is very important for the hosting companies, because it is used to establish customer billing. As this is directly connected to the money, it is very important for the telemetry to be reliable and faultless. Its purpose is to measure, monitor and alarm in case of fire (because nobody will read and notice this, and I will be a little bit surprised if anyone does). Ceilometer readouts are logged and can be viewed if necessary or if the customer will request the statistics of the resource usage, he must pay for.

* 1. Database (Trove)

Trove is a database as a service for OpenStack. It has two diagonal types of engines: relational and non-relational database. The service goal is to permit users quickly utilize database without necessity to handle complex administrative tasks. Of course, Trove focuses on high performance, while performing administration tasks: deployment, configuration, backups, restores and monitoring. Trove is still actively developed, and people are focused on improving its internal mechanisms.

* 1. Elastic Map Reduce (Sahara)

In this section, first of all, I would like to introduce you to the Hadoop clusters. Hadoop is a Map-Reduce distributed database system. It is based on the concept of two operations: map and reduce. This concept became very popular for the task of processing huge amount of data in an undertime. This concept widely used by such IT giants as Google, Yandex, and some other less known companies. Hadoop is one of the realization of the concept, brought into world by Apache. The Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly available service on top of a cluster of computers, each of which may be prone to failures.

As it is formulated in the main project: “Sahara aims to provide users with simple means to provision Hadoop clusters by specifying several parameters like Hadoop version, cluster topology, nodes hardware details and a few more”. After a user fills all the parameters, Sahara deploys the cluster in a few minutes; it also is capable of adding and removing worker nodes on demand.

Sahara’s architecture consists of “Vendor Plugins” – mechanism responsible for configuring and launching data processing units, Elastic Data Processing (EDP) – mechanism responsible for scheduling and managing data processing jobs and other Hadoop parts that are responsible for communication with other OpenStack components, such as Horizon, Keystone, Glance, etc.

* 1. Bare Metal Provisioning (Ironic)

Ironic is an integrated OpenStack project, which aims to use bare metal machines instead of virtual machines. It is best thought of as a bare metal hypervisor **API** and a set of plugins, which interact with the bare metal hypervisors. By default, it will use PXE (preboot execution environment (on the client side it requires only a PXE-capable network interface controller)) and IPMI (intelligent platform management interface (set of computer interface specifications for an autonomous management and monitoring independent from bare computer resources, such as CPU or firmware)) in concert to provision and turn on/off machines. Ironic also supports vendor-specific plugins, which may implement additional functionality.

In real world this service is usually not used, because of the virtualization technology that is adjusted nearly everywhere. Virtualization gives some valuable benefits like scalability and that is why usually preferable.

There is an interesting thing about the Ironic. Because of the service purpose “bare metal”, the mascot “bear metal” was created, you can find the reference to the picture at the end [12].

* 1. Multiple Tenant Cloud Messaging (Zaqar)

Zaqar (used to be known as Marconi) is a multi-targeted cloud messaging service for web developers. It combines the ideas pioneered by Amazon's SQS (Simple Queue Service) product with additional semantics to support event broadcasting apart from the other services aspects like reliable, simple, scalable, secure and inexpensive firewall-friendly messaging from one pier to another. Underlying the most important properties: security and scalability is very important for web applications, because it must obtain agility and hacker resistance. All these requests lead to next design principles: component-based architecture (enables to add new abilities operatively), availability and scalability, fault tolerance, recoverability (failures is easy to debug and repair), open standards (important step for security).

The service supports a fully RESTful API, which developers can use to send messages between various components of their SaaS and mobile applications by using a variety of communication patterns. Any OpenStack components can integrate with Zaqar to communicate with other components and users, but in order to use the service, component must authenticate itself. At the time of writing this paper, Zaqar was ready to use in production, but only for small and medium-sized workloads.

* 1. Shared File System Service (Manila)

Manila was abstracted from the Block Storage service and now it is a shared filesystem service. Just like the other OpenStack controllers, this one can permit to create, delete, and give/deny access to a share. However, this service is not as ordinary as others are. The problem of the shared filesystem is a long-standing problem and there is still no best answer how to solve it. Every decision has its benefits and drawbacks. It is very hard to allow write into the shared file system instances (there is several instances because of the redundancy) and at the same time give equal results to other users, who is reading data from the same file system.

The design and prototype implementation provide extensibility for multiple backends (to support file system specific nuances (for example EMC, NetApp, HP, IBM, Oracle or Red Hat GlusterFS)) but is intended to be sufficiently abstract to support any of a variety of shared file systems.

* 1. DNSaaS (Designate)

This OpenStack service gives DNS as a Service. It is not a very complicated application, but it has one clear benefit in comparison with traditional DNS services: it has authentication phase, which blocks one of the design vulnerability of DNS servers.

* 1. Security API (Barbican)

Barbican is a REST API designed for the secure storage, provisioning and management of secrets. It is aimed at being useful for all environments, including large ephemeral Clouds. Secrets can be one of the encryption keys, X.509 certificates and passwords.

Security is always very touchy theme for the business, so I will tell some more information under this topic. In modern world, there is a big problem with the key management. While windows have some decent option like Active Directory or Data Protection API, Linux lack any mechanism for key management. For the symmetric and asymmetric keys (but not raw secrets like passwords), Barbican supports full life cycle: management, provisioning, expiration, reporting, etc.

The main idea of the Barbican is to create a central secret-store for keying material, which could be distributed to all types of applications. As this service does not necessary demand the OpenStack, it can be used standalone in any cloud instances and probably even companies that are not related to cloud computing. It means that the developers can try to build its own community around this product.

The picture presents an overall logical diagram for Barbican

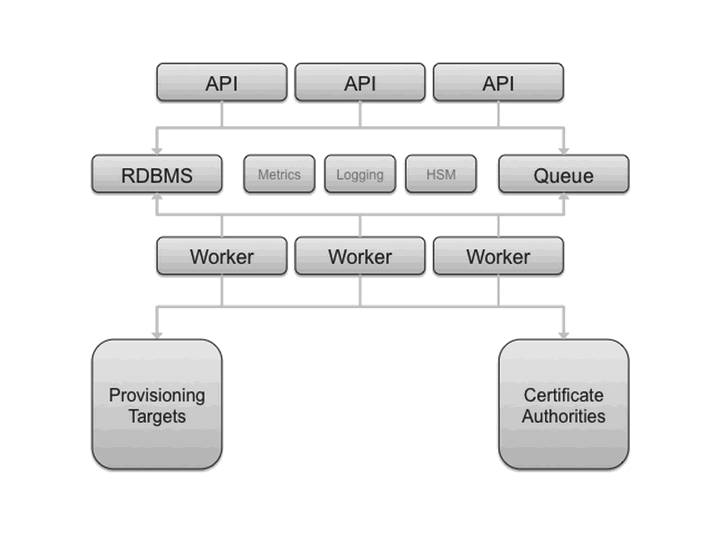


Рисунок 1. Overall logical diagram for Barbican

As you can see, it consists of the Relational Database Management System (RDBMS) to store keying material. The API nodes handle incoming REST requests to Barbican and if the request can be handle asynchronously, it will access RDBMS directly; otherwise, the request will be queued and processed by workers that can communicate with certificate authorities or with provisioning targets module.

* 1. Components Relationships

Now after we have seen all the component of the OpenStack it is time to describe its relationships. There are a lot of them, and the best way to introduce them is to show the graph:

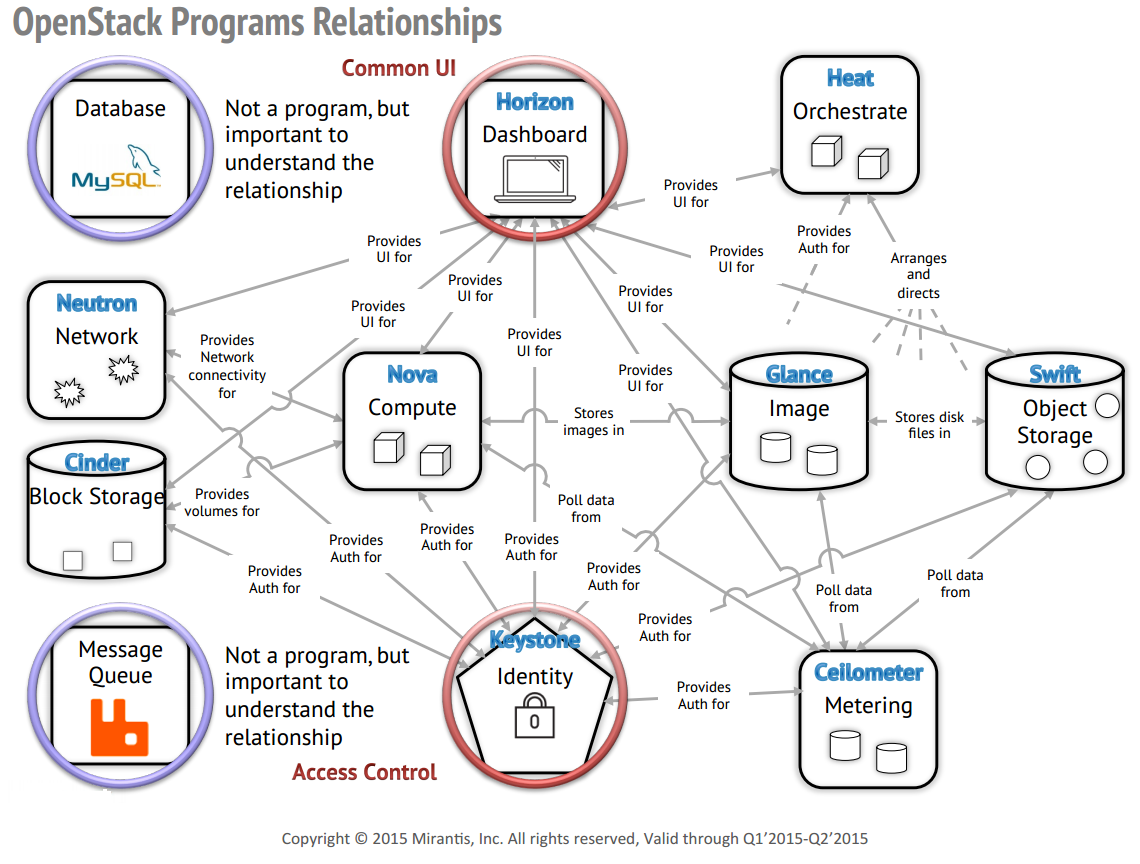


Рисунок 2. Component's Relationships

As you can see the system is very tangled, but as there is standard RESTful API interfaces everywhere, it is easy to maintain and develop.

1. Cloud Example

Now, after we have seen the OpenStack component, I will show a typical example of how the OpenStack cloud can be arranged:

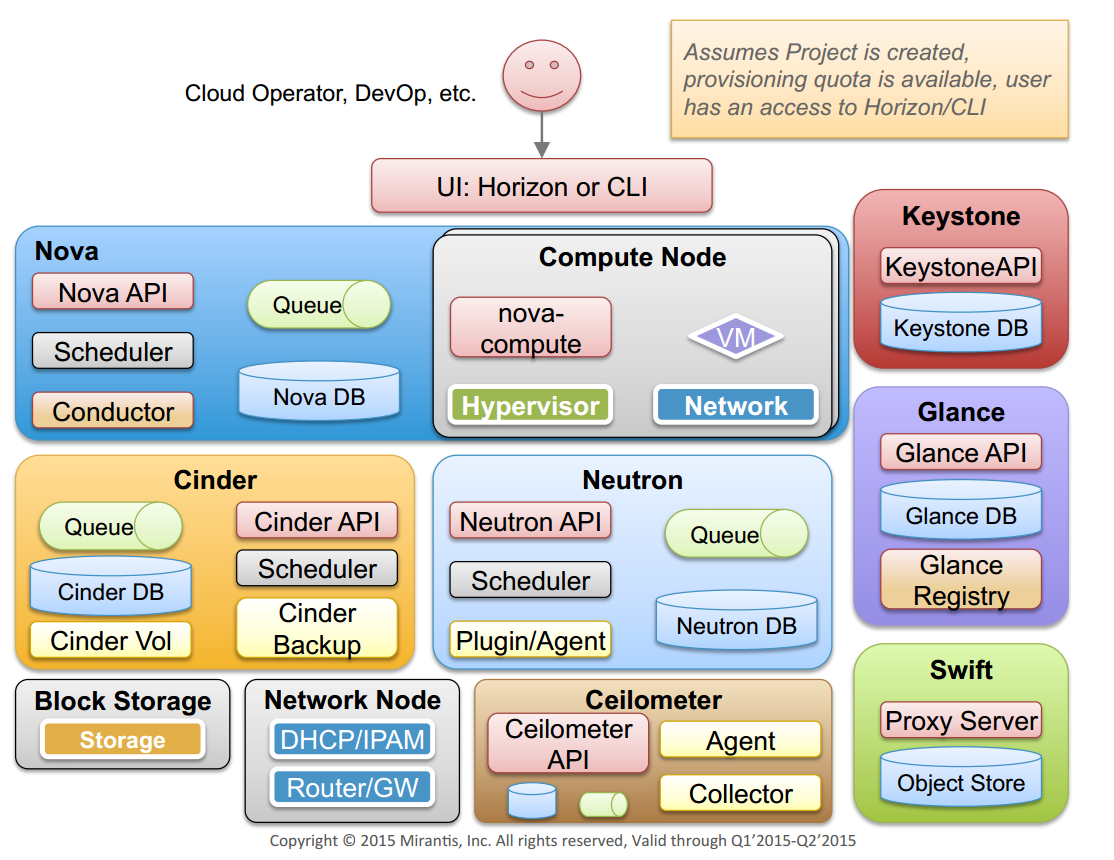


Рисунок 3. OpenStack Example

On this example, you can see the basic OpenStack cloud. Nova is used to manage the compute nodes, Neutron is used to manage the network, Cinder and Swift is used to store the data, Keystone is used to authenticate user, Glance is used to provide the images usually stored on Swift, and Ceilometer is used to gather statistics from the system

1. Deployment models

OpenStack has a wide variety of users; it is companies needed to manage datacenters. It can be hosting companies, providing PaaS, or the companies that can provide SaaS (company can provide other company for example with 1C) or IaaS (like Google, providing searching, e-mail, cloud storage, etc.). Or the companies that is creating datacenter for its own purposes, but in this case they also provide one of three types of services, and the only difference is that they are customer and the seller at the same time. Here are some companies-users of OpenStack: AT&T, Deutsche Telekom, eBay, HP Public Cloud, Intel, NASA, NSA, DreamHost, PayPal, Sony, Yahoo (terrifying company, yay?), Alcatel-Lucent, etc.

Here is the main theoretical use cases of the OpenStack:

1. Service providers offering an IaaS or services higher up in the stack (PaaS or SaaS)
2. IT departments acting as cloud service providers for business units and project teams.
3. Processing big data with tools like Hadoop.
4. Scaling computeup and down to meet demand for web resources and applications.
5. High-performance computing (HPC) environments processing diverse and intensive workloads.

As you can see now the OpenStack is very promising framework, that is why vendors have worked through multiple ways for customers to deploy it. And here is how it is in practice:

1. Infrastructure as a Service. A vendor hosts an OpenStack-based private cloud, including the underlying hardware and the OpenStack software.
2. Platform as a Service. A vendor provides a public cloud computing system based on the OpenStack project.
3. Software as a Service. A vendor hosts OpenStack management software (without any hardware) as a service. Customers sign up for the service and use it with their internal servers, storage and networks to get a fully operational private cloud.
4. On-premises distribution: In this model, a customer downloads and installs an OpenStack distribution within their internal network. (This is sort of installing OpenStack by your own)
5. Appliance based OpenStack: Nebula was a vendor that sold appliances that could be plugged into a network, which spawned an OpenStack deployment. (This is sort of automatic installation of OpenStack, prepared by some vendor)
6. OpenStack Features

Now I would like to enumerate features in the last version of the OpenStack 2015.1.2 “Kilo”:

1. Managing virtualized server resources (CPU, memory, disk, network)
2. Managing Local Area Networks (LAN) (VLANs, DHCP, IPv4, IPv6)
3. API with rate limiting and authentication (designed to add security between different users)
4. Distributed and asynchronous architecture (scalability and availability of the system leads to decreasing system downtime)
5. Virtual Machine (VM) image management (enables easy store, importing, sharing, and starting images)
6. Security groups (makes it easy to control access to virtual machines)
7. Role based access control (extra security measures)
8. Ability to allocate, track and limit resource utilization
9. Intuitive graphical interface for resource administration and orchestration for automatic deployment.
10. Conclusion

In conclusion, I would like to underline OpenStack essence, main purpose, and main functionality. OpenStack in an open software platform, developed to maintain wide variety of datacenters. It has module structure, which simplifies it and makes more flexible. The main pursued qualities are scalability, flexibility, productivity, cheapness, security, upgradability, multi-functionality.

The core components of the OpenStack are Compute, Storage and Networking. Other component gives additional functionality, such as identity service, database, telemetry, security API, etc. However, the power of OpenStack is not only in the ability to give resources to the user, but also to control them and limit. There is different types of restrictions that can be set onto the user of the cloud.

OpenStack managed to create its own ecosystem with its own “governance”, which helps to propagate the platform further into the IT community making it even better on the way up to the top of the world. Its main goal is probably been achieved and now OpenStack rapidly spreads across the market.

1. References
2. <https://www.openstack.org/software/>
3. <https://wiki.openstack.org/wiki/>
4. <http://docs.openstack.org/developer/openstack-projects.html>
5. <http://thenewstack.io/sdn-controllers-and-openstack-part1/>
6. <https://en.wikipedia.org/wiki/Cloud_computing>
7. <https://en.wikipedia.org/wiki/OpenStack>
8. <https://github.com/openstack/>
9. <https://github.com/swiftstack/swift>
10. <https://en.wikipedia.org/wiki/Orchestration_%28computing%29>
11. <https://hadoop.apache.org/>
12. <https://wiki.openstack.org/wiki/File:Ironic_mascot.svg>
13. <https://aws.amazon.com/sqs/>
14. <https://www.openstack.org/foundation/board-of-directors/>
15. <https://www.openstack.org/legal/bylaws-of-the-openstack-foundation/>
16. <http://www.slideshare.net/mirantis/presentations>
17. <https://en.wikipedia.org/wiki/Software-defined_networking>