SIAULIAI UNIVERSITY

FACULTY OF TECHNOLOGY, PHYSICAL AND BIOMEDICAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS

Andrii Sokorenko

THE CLOUD COMPUTING: SERVICE-ORIENTED SOLUTIONS RESEARCH

DEBESŲ KOMPIUTERIJA: Į PASLAUGAS ORIENTUOTŲ SPRENDIMŲ TYRIMAS

Master Thesis

Supervisor: dr. Liudvikas Kaklauskas

Reviewer: prof. Genadijus Kulvietis

Šiauliai, 2017

Andrii Sokorenko. The cloud computing: service-oriented solutions research.
Course's thesis, supervisor doc. Liudvikas Kaklauskas. The Department of Informatics, The Faculty of Technology, Physical and Biomedical Sciences, Šiauliai University.

2017, 64 p.

Summary

In this master's thesis researcher described cloud computing architecture, deployment models, such us private cloud, community cloud, public cloud and hybrid cloud. Deeply explain main characteristic of each service models. Also described cloud computing technologies, namely virtualization, service-oriented architecture, grid computing and utility computing. Made overview of service platforms such us OpenStack, CloudStack, Eucalyptus and vCloud Director. According to received information about architecture and main futures of these technologies researcher provided general, functional and property comparison between all of them, chose which service platform is worth to use and implement it in virtual environment with description of all executed commands.

Andrii Sokorenko. Debesis kompiuterija: į paslaugas orientuota sprendimai tyrimai.
Kursas darbas, Vadovas doc. Liudviko Kaklauskas. Informatikos fakultetas, technologijos, fizinių ir biomedicinos mokslai, Šiaulių universiteto fakultetas.

2017, 64 p.

Santrauka

Šiame Magistro rašto darbe apibūdinama skaitmeninės duomenų bazės architektūra, diegimo modeliai – privati, bendruomenės, viešoji ir hibridinė skaitmeninė duomenų bazės. Detaliau paaiškintos kiekvienos paslaugos modelio pagrindinės savybės. Taip pat aprašytos skaitmeninės duomenų bazės technologijos – jų virtualizacija, į paslaugas orientuota architektūra, skaičiavimo ir paslaugų tinklai. Apžvelgtos paslaugų platformos, tokios kaip - OpenStack, CloudStack, Eucalyptus ir vCloud Director. Atsižvelgiant į gautą informaciją apie architektūrą ir pagrindines ateities technologijas, pateiktas bendras bei funkcinis savybių palyginimas , siūloma, kurios paslaugų platformos tinkamos naudoti ir įgyvendinti jas virtualioje aplinkoje, aprašant visas vykdomas komandas.

CONTENT

INTRODUCTION

Cloud Computing became to represents a new technology that uses internet, isolated servers to keep up data and applications. Cloud computing permits castomers and businesses to use applications without installation and access to their personal files at any personal computer with internet access. This technology permits to save resources of companies in computing by centralizing their data in one cloud storage, processing and bandwidth.

Cloud computing used for computing resources such us hardware and software, that are delivered as a service through the network. The term cloud started to use because of cloud-shaped symbol, like an abstraction for the advanced infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's information, applications and computation.

This new model brings together all disciplines, technologies and business models to deliver Information Technology resources on-demand. This is a new trend that well fits in an environment where resources are provisioned dynamically and exposed as a service on the Internet.

The main goal of this master thesis is to discover the cloud computing architecture, make overview of layers and types of clouds, such us SaaS, PaaS, IaaS.

According to discovered information build own IaaS. Make comparison between known nowdays technologies. After received results find out which service platform is worth to use and describe process of it implementation. Summurise received experience of building own Infrastructure as a Service.

Currently, many corpanies are involved in cloud computing and many cloud computing platforms have been put forward. In this context, open source cloud technologies such as OpenStack, CloudStack, Eucalyptus and vCloud Director have gained significant momentum in the last few years.

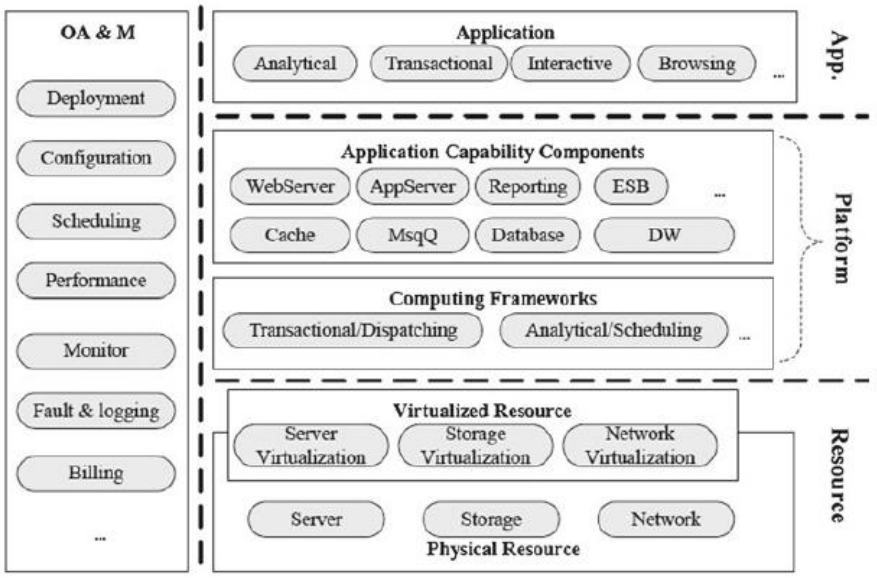
Each has its own characteristics and advantages. For a researcher, they represent a unique opportunity to analyze, contribute, and innovate in new services using these technologies.

I CLOUD COMPUTING ARCHITECTURE OVERVIEW

According to the National Institute of Standards and Technology definition, cloud computing is a model designed for on-demand network access to a shared pool of computing resources like servers, applications, networks, storage and services, that can be released in short period of time with minimal management effort or service provider interaction.

Many organizations and researchers have defined the architecture for Cloud Computing. Basically, the whole system can be divided into the core stack and the management. In the core stack, there are three layers: Resource, Platform and Application. The resource layer is the infrastructure layer which is composed of physical and virtualized computing, storage and networking resources. The platform layer is the most complex part which could be divided into many sub-layers; e.g. a computing framework manages the transaction dispatching and/or task scheduling. A storage sub-layer provides unlimited storage and caching capability. The application server and other components support the same general application logic as before with either on-demand capability or flexible management, such that no components will be the bottle neck of the whole system.

Based on the underlying resource and components, the application could support large and distributed transactions and management of huge volume of data. All the layers provide external service through web service or other open interfaces. Cloud Architecture is depicted in Pic. 1.



Pic. 1. Cloud computing architecture

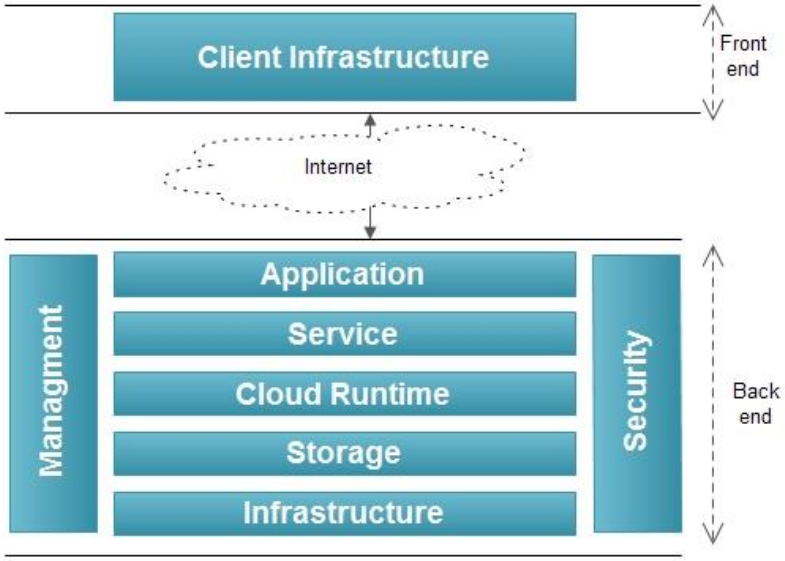
(source: International Journal of Trend in Research and Development, Volume 4(1), ISSN: 2394-9333: A review on Cloud Computing)

Cloud Computing architecture refers to the various components and sub-components of cloud that builds the structure of the system. Broadly, this architecture can be classified into two sections (Pic. 2): source:

1) Front-end
2) Back-end

The front-end and back-end is connected to each other via virtual network or the internet. Besides, there are other components like Middleware, Cloud Resources etc., that is included in the Cloud Computing architecture.

Front-end is the side that is visible for the client, customer or the user. It includes the client's computer system or network that is used for accessing the cloud system. Different Cloud Computing system has different user interfaces. For email programs, the support is driven from web browsers like Firefox, Chrome, and Internet Explorer etc. On the other hand, for other systems there are unique applications shared between the client and the service provider.



Pic. 2. Two sections of Cloud Computing architecture

Back-end is the side used by the service provider. It includes various servers, computers, data storage systems, virtual machines etc., that builds together the cloud of computing services. This system can include different types of computer programs. Each application in this system is managed by its own dedicated server. The back-end side has some responsibilities to fulfill towards the client:

1) To provide security mechanisms, traffic control and protocols
2) To employ protocols that connects networked computers for communication

One central server is used to manage the entire Cloud Computing system. This server is responsible for monitoring the traffic and making each end run smoothly without any disruption. This process is followed with a fixed set of rules called Protocols. Also, a special software named as Middleware is used to perform the processes. Middleware connects networked computers to each other.

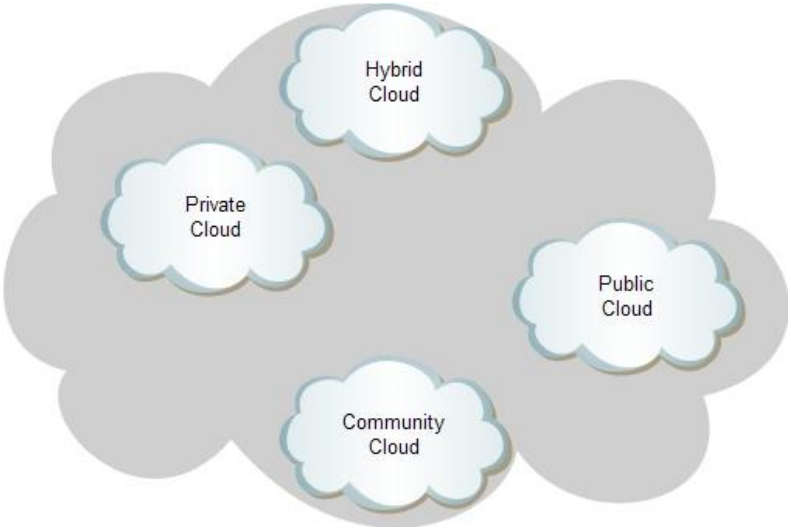
Depending on the demand of client, the storage space is provided by the Cloud Computing service provider. While some companies require huge number of digital storage devices, few others require not as many. Cloud Computing service provider usually holds twice the number of storage space that required by the client. This is to keep a copy of client's data secured during the hours of system breakdown. Building copies of data for backup is called as Redundancy.

1.1 Deployment models

The NIST definition lists five essential characteristics of cloud computing: on-demand self-service, broad network access, resource pooling, rapid elasticity or expansion, and measured service. Deployment models divides into four groups:

1) Private cloud
2) Community cloud
3) Public cloud
4) Hybrid cloud

Together they categorize ways to deliver cloud services (Pic.3). The definition is intended to serve as a means for broad comparisons of cloud services and deployment strategies, and to provide a baseline for discussion from what is cloud computing to how to best use cloud computing.



Pic. 3. “Deployment models”

Private Cloud - This is a secured IT infrastructure, controlled and operated by single organization. This organization can manage a private cloud on its own. Infrastructure can be located either in the customer’s building, or at external operator’s building, or partly at customer and partly at operator. The perfect private cloud is a cloud deployed only on territory of organization, managed and controlled by its employees.

Public Cloud – is an IT infrastructure used by many companies and services. Users of these clouds do not have permission to manage and maintain this cloud, all responsibility for these actions is controlled by owner of this cloud. Any company and individual user can become a user of the offered services. Cloud providers offer an easy way to deploy web sites or business systems, with great scalability. Examples: online services Amazon EC2 and Simple Storage Service (S3), Google Apps / Docs, Salesforce.com, Microsoft Office Web.

Community Cloud – Community cloud shares infrastructure between several organizations from a specific community with common concerns (security, compliance, jurisdiction, etc.), whether managed internally or by a third-party and hosted internally or externally. The costs are spread over fewer users than a public cloud (but more than a private cloud), so only some of the cost savings potential of cloud computing are realized (The Definitive Guide to Modern Supply Chain Management: Chad W. Autry,Thomas J. Goldsby,John Bell,Mark A. Moon,Chuck Munson,Michael Watson,Sara Lewis,Peter Cacioppi,Jay Jayaraman).

Hybrid cloud – an IT infrastructure that contain of the best parts of a public and private cloud in time of resolving some tasks. Usually this type of clouds is used when the organization has seasonal activity periods, some of resources are transferred to the public cloud. (Source: Science, Engineering & Education, 1, (1), 2016, 83-88: Big data and cloud computing – issues and problems)

1.2 Service Models

Cloud Computing is gaining popularity to the extent that the new XaaS service category introduced will gradually take the place of many types of computational and storage resources used today. Cloud Computing delivers infrastructure, platform, and software (application) as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. These services in industry are respectively referred to as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Table 1 summarizes the nature of these categories and lists some major players in the field.

Table 1. Cloud computing services classification

Category	Characteristics	Product type	Vendors and products
SaaS	Customers are provided with applications that are accessible anytime and from anywhere	Web applications and services Web (2.0)	SalesForce.com(CRM) Google documents Clarizen.com (Project management) Google mail (automation)
PaaS	Customers are provided with a platform for developing applications hosted in the Cloud	Programming APIs and frameworks; Deployment system	Google AppEngine Microsoft Azure Manjrasoft Aneka
IaaS/ HaaS	Customers are provided with virtualized hardware and storage on top of which they can build their infrastructure	Virtual machines management infrastructure, Storage management	Amazon EC2 and S3; GoGrid; Nirvanix

(Source: Security, trust, and regulatory aspects of Cloud Computing is Business Environments. S. Srinivasan Texas Southern University, USA)

1.2.1 Infrastructure-as-a-Service

Infrastructure-as-a-Service, also called Hardware-as-a-Service was coined possibly in 2006. As the result of rapid advances in hardware virtualization, IT automation and usage metering and pricing, users could buy IT hardware, or even an entire data center, as a pay-as-you-go subscription service.

Infrastructure-as-a-Service (IaaS) or Hardware-as-a-Service (HaaS) solutions deliver IT infrastructure based on virtual or physical resources as a commodity to customers. These resources meet the end user requirements in terms of memory, CPU type and power, storage, and, in most of the cases, operating system as well. Users are billed on a pay-per-use basis. They have to set up their applications on top of these resources that are hosted and managed in data centers owned by the vendor. Amazon is one of the major players in providing IaaS solutions.

Amazon Elastic Compute Cloud (EC2) provides a large computing infrastructure and a service based on hardware virtualization. By using Amazon Web Services, users can create Amazon Machine Images (AMIs) and save them as templates from which multiple instances can be run. It is possible to run either Windows or Linux virtual machines, for which the user is charged per hour for each of the instances running. Amazon also provides storage services with the Amazon Simple Storage Service (S3), users can use Amazon S3 to host large amount of data accessible from anywhere.

(Source: Cloud Computing: First International Conference, CloudCom 2009, Beijing)

1.2.2 Platform-as-a-Service

Platform-as-a-Service provides an application or platform for developing where users can create their own application that will be executed and run in the Cloud. PaaS provides an application framework and API which can be used to create and develop applications for the Cloud.

Google AppEngine is a platform for developing scalable web applications that run on top of data centers maintained by Google. It defines an application model and provides a set of APIs that allow developers to take advantage of additional services such as Mail, Datastore, Memcache, and others. AppEngine manages the execution of applications and automatically scales them up/down as required. Google provides a free but limited service, while utilizes daily and per minute quotas to meter and price applications requiring a professional service.

Azure is a Cloud service operating system that serves as the development, runtime, and control environment for the Azure Services Platform. By using the Microsoft Azure SDK, developers can create services that leverage the .NET Framework. These services have to be uploaded through the Microsoft Azure portal in order to be executed on top of Windows Azure. Additional services, such as workflow execution and management, web services orchestration, and access to SQL data stores, are provided to build enterprise applications.

Aneka, commercialized by Manjrasoft, is a pure PaaS implementation and provides end users and developers with a platform for developing distributed applications for the Cloud by using .NET technology. The core value of Aneka is a service oriented runtime environment that is deployed on both physical and virtual infrastructures and allows the execution of applications developed by means of various programming models. Aneka provides a Software Development Kit (SDK) helping developers to create applications and a set of tools for setting up and deploying Clouds on Windows and Linux based systems. Aneka does not provide an IT hardware infrastructure to build computing Clouds, but system administrators can easily set up Aneka Clouds by deploying Aneka containers on clusters, data centers, desktop PCs, or even bundled within Amazon Machine Images.

(Source: Cloud Computing: First International Conference, CloudCom 2009, Beijing)

1.2.3 Software-as-a-Service

Software or an application is hosted as a service and provided to customers across the Internet. This mode eliminates the need to install and run the application on the customer's local computers. SaaS therefore alleviates the customer's burden of
software maintenance, and reduces the expense of software purchases.

Software-as-a-Service solutions are at the top end of the Cloud Computing stack and they provide end users with an integrated service comprising hardware, development platforms, and applications. Users are not allowed to customize the service but get access to a specific application hosted in the Cloud. Examples of SaaS implementations are the services provided by Google for office automation, such as Google Mail, Google Documents, and Google Calendar, which are delivered for free to the Internet users and charged for professional quality services. Examples of commercial solutions are SalesForce.com and Clarizen.com, which provide online CRM (Customer Relationship Management) and project management services, respectively.

(Source: Inter-cooperative Collective Intelligence: Techniques and Applications, Authors: Fatos Xhafa, Nik Bessis, 2014)

1.2.4 Data-as-a-Service

Data in various formats and from multiple sources could be accessed via services by users on the network. Users could, for example, manipulate the remote data just like operate on a local disk or access the data in a semantic way in the Internet. Amazon Simple Storage Service (S3) provides a simple Web services interface that can be used to store and retrieve, declared by Amazon, any amount of data, at any time, from anywhere on the Web. The DaaS could also be found at some popular IT services, e.g., Google Docs and Adobe Buzzword. ElasticDrive is a distributed remote storage application which allows users to mount a remote storage resource such as Amazon S3 as a local storage device.

(Source: Inter-cooperative Collective Intelligence: Techniques and Applications, Authors: Fatos Xhafa, Nik Bessis, 2014)

II CLOUD COMPUTING TECHNOLOGIES

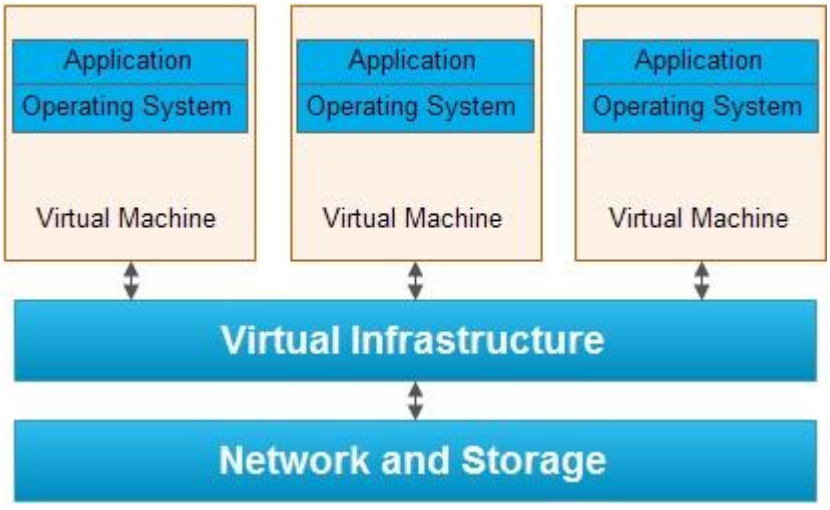
There are certain technologies that are working behind the cloud computing platforms making cloud computing flexible, reliable and usable. These technologies are listed below:

1) Virtualization
2) Service-Oriented Architecture (SOA)
3) Grid Computing
4) Utility Computing Virtualization

2.1 Virtualization model

The idea of virtualizing a computer system's resources, including processors, memory, and I/O devices, has been well established for decades, aiming at improving sharing and utilization of computer systems. Hardware virtualization allows running multiple operating systems and software stacks on a single physical platform. (Source: Cloud Computing: Principles and Paradigms. Author: Rajkuma Buyya, James Broberg, Andrzej Goscinski)

Virtual machine techniques, such as VMware, and Xen offer virtualized IT infrastructures on demand. Virtual network advances, such as Virtual Private Network (VPN), support users with a customized network environment to access Cloud resources. Virtualization techniques are the bases of the Cloud Computing since they render flexible and scalable hardware services. (Pic. 4)
(Source: Scientific Cloud Computing: Early Definition and Experience. Author: Lizhe Wang, Jie Tao, Marcel Kunze)



Pic. 4. Virtualization model

2.2 Service-Oriented Architecture, Web Services, Web 2.0, Mashups

The emergence of Web services open standards has significantly contributed to advances in the domain of software integration. Web services can glue together applications running on different messaging product platforms, enabling information from one application to be made available to others, and enabling internal applications to be made available over the Internet.

Over the years a rich WS software stack has been specified and standardized, resulting in a multitude of technologies to describe, compose, and orchestrate services, package and ransport messages between services, publish and discover services, represent quality of service (QoS) parameters, and ensure security in service access.

WS standards have been created on top of existing ubiquitous technologies such as HTTP and XML, thus providing a common mechanism for delivering services, making them ideal for implementing a service-oriented architecture (SOA). The purpose of a SOA is to address requirements of loosely coupled, standards-based, and protocol-independent distributed computing. In a SOA, software resources are packaged as “services,” which are well-defined, self-contained modules that provide standard business functionality and are independent of the state or context of other services. Services are described in a standard definition language and have a published interface.

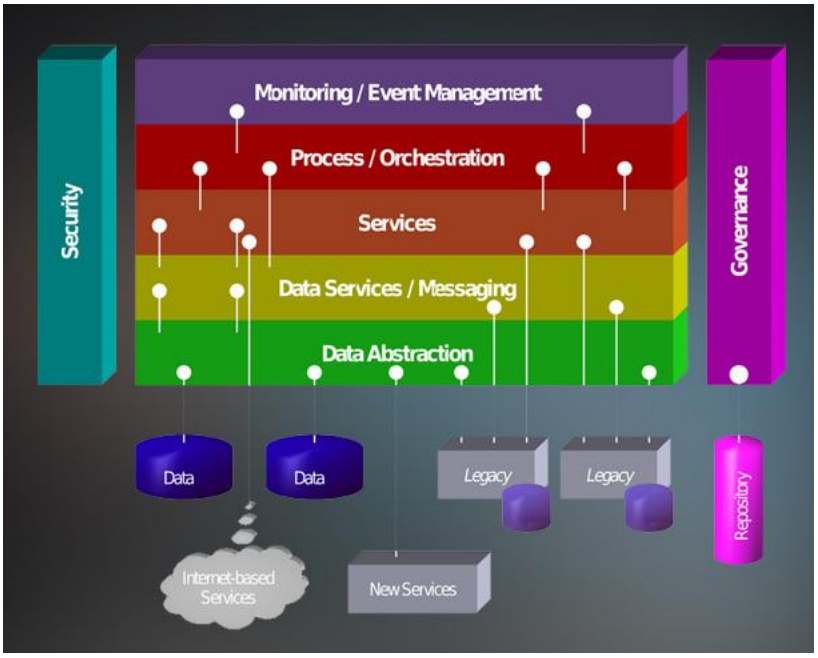
The maturity of WS has enabled the creation of powerful services that can be accessed on-demand, in a uniform way. While some WS are published with the intent of serving end-user applications, their true power resides in its interface being accessible by other services. An enterprise application that follows the SOA paradigm is a collection of services that together perform complex business logic.

This concept of gluing services initially focused on the enterprise Web, but gained space in the consumer realm as well, especially with the advent of Web 2.0. In the consumer Web, information and services may be programmatically aggregated, acting as building blocks of complex compositions, called service mashups. Many service providers, such as Amazon, del.icio.us, Facebook, and Google, make their service APIs publicly accessible using standard protocols such as SOAP and REST. Consequently, one can put an idea of a fully functional Web application into practice just by gluing pieces with few lines of code.

In the Software as a Service (SaaS) domain, cloud applications can be built as compositions of other services from the same or different providers. Services such user authentication, e-mail, payroll management, and calendars are examples of building blocks that can be reused and combined in a business solution in case a single, ready-made system does not provide all those features. Many building blocks and solutions are now available in public marketplaces. For

example, Programmable Web1 is a public repository of service APIs and mashups currently listing thousands of APIs and mashups. Popular APIs such as Google Maps, Flickr, YouTube, Amazon eCommerce, and Twitter, when combined, produce a variety of interesting solutions, from finding video game retailers to weather maps. Similarly, Salesforce.com's offers AppExchange,2 which enables the sharing of solutions developed by third-party developers on top of Salesforce.com components. (Pic. 5).

(Source: Rajkumar Buyya, James Broberg - Cloud Computing Principles and Paradigms - 2011)



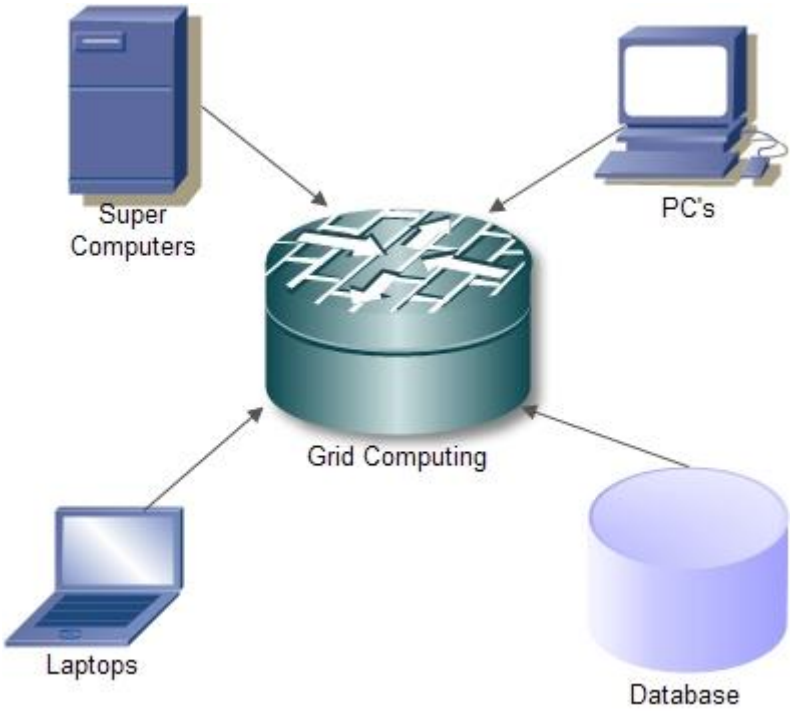
Pic. 5. SOA (source: http://www.wilmertech.com/service-oriented-architecture.php)

2.3 Grid Computing

Grid computing - is a form of distributed computing where different types of computing units working together to perform a huge number of tasks. This technology is used to solve scientific, mathematical problems that require significant computing resources. Grid computing is also used in commercial infrastructure to solve economic tasks, seismic analysis, development and discovering of new medicine.

From network organization side grid is open and standardized environment that provides a flexible, secure, coordinated separation of computing resources and storage resources that are part of this environment within a single virtual organization.

Grid computing can be organized on the basis of old PC connected in hierarchical Ethernet network with servers. These computers are heterogeneous and geographically dispersed. Grid Computing breaks complex task into smaller pieces. These smaller pieces are distributed to CPUs that reside within the grid. (Pic. 6)



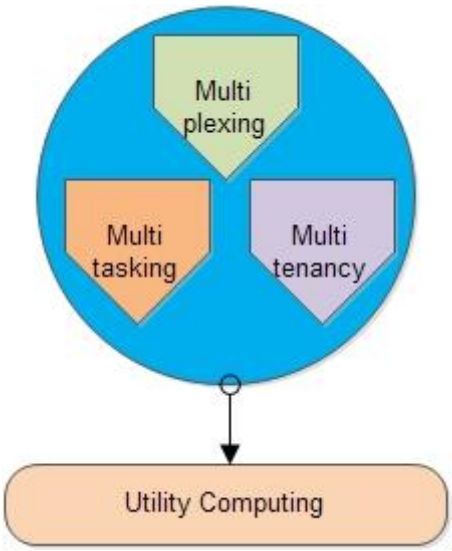
Pic. 6. Grid Computing

2.4 Utility Computing

With increasing popularity and usage, large grid installations have faced new problems, such as excessive spikes in demand for resources coupled with strategic and adversarial behavior by users. Initially, grid resource management techniques did not ensure fair and equitable access to resources in many systems. Traditional metrics (throughput, waiting time, and slowdown) failed to capture the more sutale requirements of users. There were no real incentives for users to be flexible about resource requirements or job deadlines, nor provisions to accommodate users with urgent work.

In utility computing environments, users assign a “utility” value to their jobs, where utility is a fixed or time-varying valuation that captures various QoS constraints (deadline, importance, satisfaction). The valuation is the amount they are willing to pay a service provider to satisfy their demands. The service providers then attempt to maximize their own utility, where said utility may directly correlate with their profit. Providers can choose to prioritize high yield (i.e., profit per unit of resource) user jobs, leading to a scenario where shared systems are viewed as a marketplace, where users compete for resources based on the perceived utility or value of their jobs. Further information and comparison of these utility computing environments are available in an extensive survey of these platforms. (Pic. 7).

(Source: Cloud Computing: Principles and Paradigms. Author: Rajkuma Buyya, James Broberg, Andrzej Goscinski)

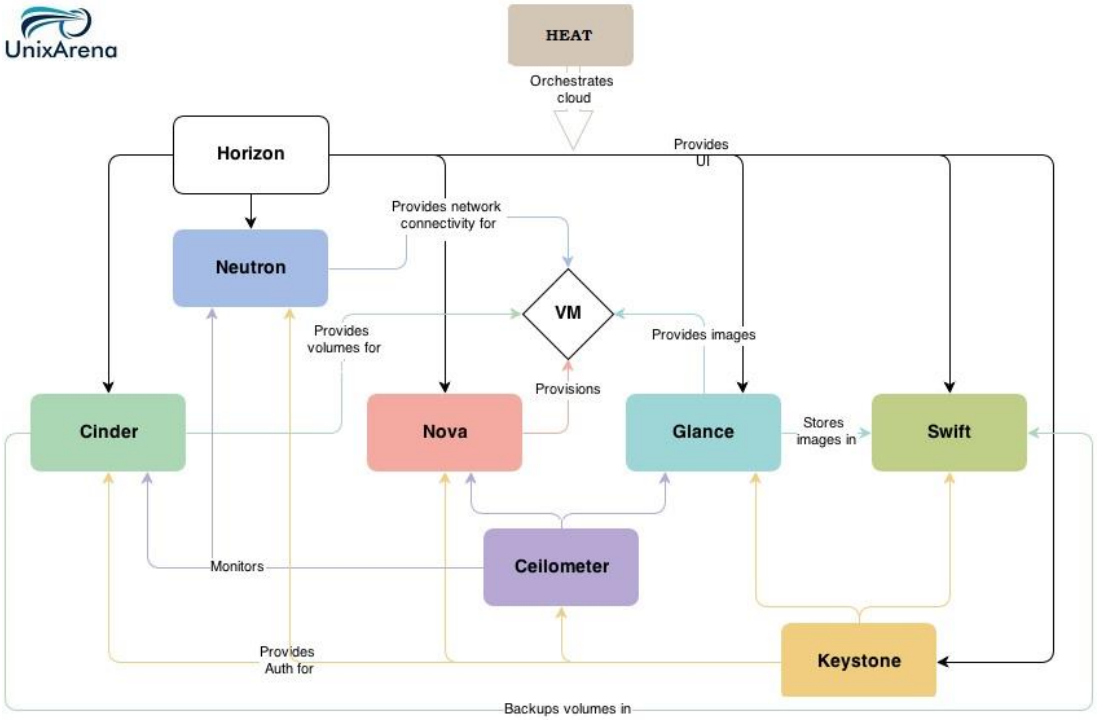


Pic. 7. Utility Computing

III SERVICE PLATFORMS

3.1 Open Stack

OpenStack is an open-source platform for deploying clouds. Release Grizzly platform OpenStack consists of seven basic projects: Compute (Nova), Networking (Neutron / Quantum), Identity Management (Keystone), Object Storage (Swift), Block Storage (Cinder), Image Service (Glance), User Interface Dashboard (Horizon) (Pic. 8).



Pic. 8 OpenStack architecture (source: http://www.unixarena.com/2015/08/openstack-architecture-and-components-overview.html)

The OpenStack Compute (Nova) module manages the "factory" of cloud computing (this is the basic component of infrastructure services). The OpenStack Compute module, written on Python, creates an abstraction layer for resources virtualizing (such as processors, memory, network adapters and hard disks) and supports the appropriate functions for increasing utilization factor and automation.

OpenStack Compute provides active management of virtual machines with functions such as start, resize, pause, stop and restart. In addition, there is a mechanism for caching images of virtual machines on the Compute nodes in order to speed up the initialization of these machines. In time of executing these images, it is possible to programmatically save files and manage them through the API interface.

(Source: http://www.informit.com/articles/article.aspx?p=2764999&seqNum=2)

The Networking (Neutron) module provides local networks management with virtual networks (VLANs), DHCP and IPv6 support. Users can define networks, subnets and routers to configure their topology. The floating IP address mechanism allows users to assign (and reassign) fixed external IP addresses of virtual machines.

The OpenStack Identity Management module (Keystone) manages the user directory, as well as the directory of OpenStack services that these users can access. The main goal is to maintain a centralized authentication mechanism for all components of OpenStack. Keystone module is able to integrate with various other directory services, including the Pluggable Authentication Module, Lightweight Directory Access Protocol (LDAP), OAuth. With the help of appropriate plug-ins several forms of authentication are supported, including simple login (using login and password) or multifactor authentication.

The OpenStack Object Storage (Swift) module is based on the Rackspace Cloud Files product. It is a reserved storage system, which is perfect for horizontal storage resources scaling. OpenStack ensures data replication and distribution among devices in its pool, so users can use hard drives and servers instead of more expensive hardware. The OpenStack system is able to get content from other active systems and transfer it to new cluster elements. Swift mostly focused on static data, such as virtual machine images, backups, and archives. Swift software writes files and other objects to a set of disk drives, which can be distributed among several servers in one or more data centers.

The OpenStack Block Storage (Cinder) module manages the block-level storage of Compute instances. Block storage uses in case of strict performance requirements (databases and file systems). Together with the Cinder module, most used storage is one, which based on Linux, but there are also some plug-ins for other platforms, including Ceph, NetApp, Nexenta and SolidFire. Users of cloud resources can manage their storage resources using the toolbar. The ability to create backup copies of Cinder volumes via the snapshot mechanism is also supported.

The OpenStack Image Service (Glance) module provides support for images of virtual machines. This module supports the ability to create snapshots and backups. Glance-images is used as templates for fast and consistent deployment of new servers. The server API uses a RESTful interface (Representational State Transfer), so users can view and select images on the virtual disk assigned to an extensible set of internal storage (including OpenStack Object Storage).

Currently, Nova fully supports two hypervisors: KVM and XEN. The platform is being developed rapidly and soon will be provided with a broader functionality. The technology is popular among a large community of specialists and is backed by such companies as Cisco, Dell, NASA, Intel, AMD, Citrix, Rackspace, and RightScale. The core of this product is developed by NASA.

(Source: http://www.informit.com/articles/article.aspx?p=2764999&seqNum=2)

Main features of OpenStack:

- Ability to manage virtualized commodity server resources
- Ability to manage local area networks
- Virtual machine image management
- Security groups
- Role-based access control
- Projects & quotas
- VNC proxy through a Web browser

OpenStack is open-source and can be downloaded for free. The project is developed by various contributors and exists mainly on user donations. In comparison to other products mentioned in this research, OpenStack seems to have the largest and the most active community. The community members are always willing to help others find solutions to any arising problems. However, OpenStack documentation is somewhat incomplete. Due to the rapid development of the product, the documentation fails to cover all the current issues and new features in time.

This open-source platform is free and is being developed very rapidly. It demonstrates a lot of progress, but still lots of development efforts are required before it can be used for production. OpenStack is already compatible with Amazon API and the dashboard project is currently under consideration.

(Source: http://www.informit.com/articles/article.aspx?p=2764999&seqNum=2)

3.2 vCloud Director

vCloud Director represents a platform for creating clouds developed by VMware. vCloud Director make it possible to building hybrid clouds and. It's easy to migrate virtual machines between private and public clouds using VMware vCloud Connector.

Main features of vCloud Director:

- Virtual data centers
- vShield security technologies
- Infrastructure service catalog
- Multi-tenant organizations
- Self-service portal
- VMware vCloud API, open virtualization format, and callouts

vCloud Director is VMware's cloud management software. VMware is one of the oldest developers in virtualization. vCloud has existed since 2008 as a cloud management system, allowing interoperability with on premise workloads virtualized with VMware vSphere. As a VMware product, vCloud works best with the vSphere hypervisor, but it does also support Hyper-V, Xen, and Red Hat virtualization.

Interoperability with other clouds can also be achieved via export to OVF or open virtualization format. Much like CloudStack, vCloud is designed to manage multiple virtual infrastructures across different locations or data centers.

vCloud also supports snapshots, volumes, live migration, templates, virtual networking, virtual firewalls, virtual load balancers, and local storage support. It can handle up to 50,000 total virtual machines, with 30,000 powered on. vCloud can manage up to 2,000 host servers for these machines. While it is more expensive (after all, CloudStack is free) and somewhat more limited, VMware provides extensive support and documentation, making setup and troubleshooting easier, especially for smaller teams.

vCloud is more appealing to shops with legacy VMware infrastructure, as they can manage and migrate existing machines into a larger hybrid cloud system. Choosing CloudStack, there is a good chance to be using the vSphere hypervisor also, and perfect packaged combination of vSphere and vCloud could make administration smoother.

In other hand there are no free editions of the product. VMware is one of the leaders in the market and has a very large community. There is also a rich knowledge base, which can be used as a free support service. The product comes with a support package and the company offers additional paid support on demand. (Source: https://www.greenhousedata.com/blog/cloudstack-vs.-vcloudwhich-is-right-for-your-organization)

Proprietary software usually comes with high quality documentation, and this platform is no exception. There will be no difficulties, in case of carefully following all the instructions provided in the guides.

According to others experience this platform was successfully installed and configured vCloud Director. It should be mentioned, that Red Hat is required in order to install this platform. Other things will be needed for installation include vCenter (with clusters and DRS), and vShield.

Obviously vCloud Director uses the vCenter API. This means a user of vCloud Director has access to the full functionality implemented in vSphere.

This is a commercial product and that is a big disadvantage for some users. However, if VMware for virtualization is already in use, vCloud Director will be the most appropriate choice.
(Source: https://www.greenhousedata.com/blog/cloudstack-vs.-vcloudwhich-is-right-for-your-organization)

3.3 Apache CloudStack

CloudStack is an open source software for cloud computing. It uses for creating and management of IaaS. Can use such hypervisors like VMware VSphere and XenServer. In addition to its own API, CloudStack also supports the Web Services (AWS) API Amazon and the Open Cloud Computing Interface from the Open Grid Forum.

Apache CloudStack allows to automate creating, configuration and management of a private, hybrid or public cloud infrastructure (IaaS, infrastructure as a service). The CloudStack platform was transferred to the Apache Foundation by Citrix, which received the project after the acquisition of Cloud.com. Installation packages are prepared for RHEL / CentOS and Ubuntu. (Source: http://en.bmstu.wiki/Apache_CloudStack)

CloudStack does not depend on the type of hypervisor and allows to use Xen (XenServer and Xen Cloud Platform), KVM, Oracle VM (VirtualBox) and VMware in the same cloud infrastructure. CloudStack allows to create public IaaS-service similar to Amazon EC2, and a private cloud-infrastructure deployed on local servers only for needs of a particular company.

In minimal implementation the CloudStack-based cloud infrastructure consists of a single management server and a set of compute nodes that can host guest OS in virtualization mode. In more complex systems it’s possible to a cluster of several management servers and additional load balancers. At the same time, the infrastructure can be divided into segments, each of which operates in a separate data center.

To manage user’s database, storage, computing and network resources, CloudStack offers web-interface and a special API.

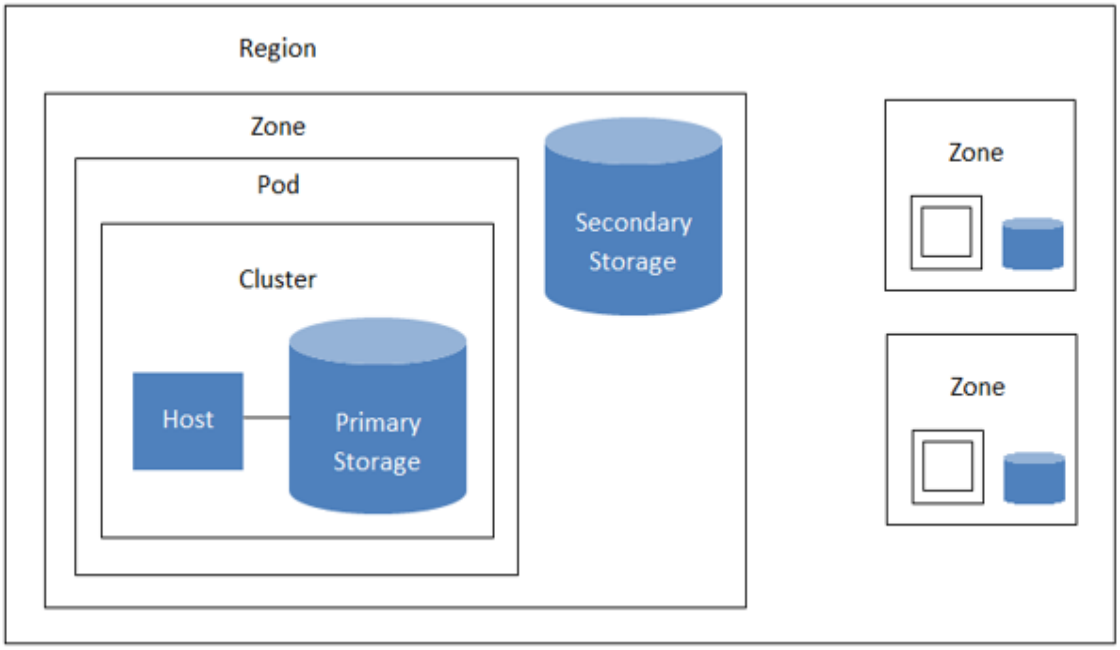
The infrastructure of CloudStack is hierarchical. The highest level is the Zone - the data center layer structure (Pic. 9).

Zone is the largest structure in the CloudStack hierarchy. A zone usually corresponds to one data center, although it is acceptable to have multiple zones within a single data center. Zones organizational infrastructure provides physical isolation. For example, each zone can have its own power source and network connection, zones can be separated geographically. Within the same zone, hosts can easily work under control of different hypervisors.

Pod - is the second largest structure in the CloudStack hierarchy and is an analog of the physical rack with servers. Racks are contained in areas. Each zone can include one or more racks.

Cluster is the third largest structure in CloudStack. Clusters are placed into racks and racks at the same time into zones. Cluster - is a group of physical servers (hosts) with the same configuration located in one rack. All hosts must operate under the same hypervisor, located on the

same subnet, and have access to at least one shared storage. Live VM migration from one host to another, without interrupting maintenance, can only be performed within a single cluster.



Pic.9 A region with multiple zones(source: http://docs.cloudstack.apache.org/en/latest/concepts.html)

Cluster is the third largest structure in CloudStack. Clusters are placed into racks and racks at the same time into zones. Cluster - is a group of physical servers (hosts) with the same configuration located in one rack. All hosts must operate under the same hypervisor, located on the same subnet, and have access to at least one shared storage. Live VM migration from one host to another, without interrupting maintenance, can only be performed within a single cluster.

Host is a physical server with hypervisor installed (KVM, Xen or ESXi) on board. Nodes provides computing resources on which the VM operates. Nodes are the smallest unit in the CloudStack infrastructure.

Storages – can be Primary and Secondary. Secondary storages can only be on NFS servers, while primary storage can be optionally connected via iSCSI or using local CloudStack server disks.

Primary storage is connected to cluster and shared by all cluster hosts. The VMs are located on the primary storage. Primary storage should have increased reliability because in case of its failure the functioning of the cloud will be completely broken.

Secondary storage does not connect to any cluster or node, it exists at zone level and used to store VM ISO images and VM snapshots.

To install CloudStack MS in the base version it’s necessary to have only one server, physical or virtual is not important, running CentOS, RHEL or Ubuntu operation systems.

Supporting of hardware virtualization on the server where CloudStack MS will be installed is not required.

Server requirements are the following: CPU x86-64, 4 GB RAM, 50 GB per disk. If the CloudStack is to be used as a secondary NFS storage it's recommended 500GB of disk space for storing VM and ISO images, one network controller, static IP address and a full FQDN name.

In practice, CloudStack is able to work on the server with any parameters. Perhaps the only requirement is to processor, it should support x86-64 architecture. In case of database CloudStack use MySQL Installation is performed using packages available from the CloudStack repositories. Can be installed quickly and smoothly.

Resources allocation like processor, memory, disk space or network can be implemented by applying templates created by system administrator. There are several categories of templates:

- Compute Offerings - describes the main VM resources, such as the maximum frequency and number of CPUs, amount of memory, storage location (local or on shared storage).
- System Offerings - describes sets of resources for system VMs.
- Disc Offerings - presets of VM discs (sizes and description)
- Network Offerings - bandwidth restriction can be set, VLAN, allow or deny a certain type of traffic.

All new templates that describe various resources are assigned by understandable names. In the future, the creation of a VM is reduced to the selection of certain templates that form, as a result, its productivity and limitations. It's impossible to change any resource template, when VM is running. The only thing that is possible with a working VM is to add or remove certain rules in one of the existing Network Offerings templates.

There are two types of accounts: "User" - for users and "Admin" - for administrators. At the domain level it's possible to define maximum amount of VM (Instance Limits), limit of public IP (Public IP Limits), maximum number of snapshots (Snapshot Limits) and etc.

All administrative accounts are divided into root administrators (Admins) and domain administrators (Domain-admins). The official documentation says that Domain-admins privileges allows to manage user accounts of their domain. In practice it's clear that Domain-admins can not create, delete or edit their domain accounts. In this case, it is not entirely clear the purpose of domains, except for imposing quantitative restrictions on a group of users.
(Source: http://docs.cloudstack.apache.org/en/latest/concepts.html)

3.4 Eucalyptus

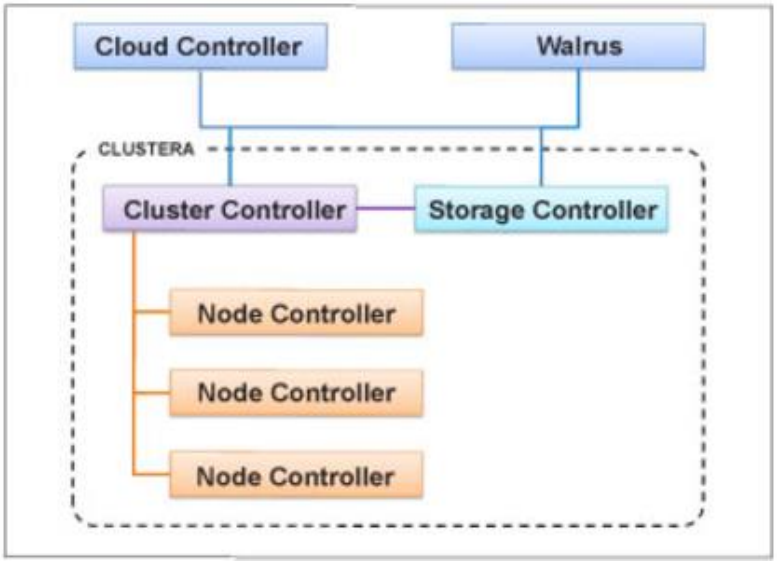
Eucalyptus is another popular cloud platform. Sony, Puma, NASA, Trend Micro and other companies have chosen it to deploy their private clouds. Eucalyptus has a free version and a commercial edition. Obviously, the commercial edition comes with much more extended functionality.

One of the greatest advantages making this platform truly convenient to work with is that the Eucalyptus API is fully compatible with the Amazon API. As a result, all the scripts and software products based on the Amazon API can be easily employed for your private cloud. Eucalyptus supports three hypervisors: XEN, KVM and ESXi. The last one is only available to the users of the Enterprise Cloud edition.

Main features:

- Roles (assigning and managing permissions)
- Hypervisor agnostic
- Clustering and zoning
- Flexible network management, security groups, and traffic isolation

Eucalyptus implements the IAAS (Infrastructure-as-a-Service) scheme, the "lower" level of the cloud. It allows to raise virtual machines after request of higher-level applications. The capabilities of Eucalyptus are mostly borrowed from Amazon EC2, so Eucalyptus can be viewed as an open source alternative to Amazon's services with certain limitations and assumptions. The hierarchical structure of Eucalyptus is shown in Pic.10



Pic.10 Eucalyptus architecture (source: http://opensourceforu.com/2014/03/build-private-cloud-eucalyptus/)

Walrus is a component, which provides permanently storage for all VM in Ecaliptus cluoud. This is a container where users can upload using simpe HTTP put/get queries data and store it.

The Node Controller controls the starting, working process and shutdown of virtual machines on node. A node is a machine with a working hypervisor (for example, Xen), which implements virtual machines (instances in the terminology of Eucalyptus).

The Cluster Controller manages subordinate node controllers: collects information about the workload of the nodes and decides which nodes will run the virtual machines.

The Storage Controller is a place for storing images of virtual machines. Warlus represents repository, which is similar to Amazon S3 service.

The Cloud Controller represents an entry point. Requests for running virtual machines are received from end-user (or higher-level application). Data of cloud nodes load arrives from cluster controllers.

Summary: in Table 2 briefly described all features which is concerned to specific service platform. (Source: Cloud Platform Comparison. Author: Vadim Truksha. 2015)

Table 2

Features	CloudStack	Eucalyptus	OpenStack	vCloud Director
AD Integration	+/-	-	-	+
Management Console	+	-	+/-	+
Web accesss to VM console	+	-	+	+
API	+	+	+	+
Multi-role Support	+	+	+	+
VLANs	+	+	+	+
Hypervisors	KVM, XEN, ESXi, OVM, BareMetal	KVM, XEN, ESXi	XEN, KVM	ESXi
Easy Template Creation Process	+	-	-	+
Snapshots	+	+	+	+
Resource Over Provisioning and Limits	+	+	+	+
Alerts and Notifications	+	-	-	+
Volumes	+	+	+	+
Guest OS Preferences	like hypervisors	linux	like hypervisors	like hypervisors
Host Maintenance with Live Migration	+	-	-	+
Free	+	+/-	+	-
Amazon API Compability	+	+	+	-
Rightscale	+	+	+/-	-
Hight Availability cloud companent	+	+	-	+
Implementation complexity	-	+	+	-

After analyzing all advantages and disadvantages of service platforms described in Table 1 researcher made a decision to build IaaS based on OpenStack platform using KVM hypervisor.

IV HYPERVISORS OVERVIEW

4.1 XEN hypervisor

A modern service provider, which provide services based on VPS and VDS cannot exist without virtualization technologies. To simplify managing of virtual machines service providers uses hypervisors – specialized programs, which make it possible to create, manage, start, stop and transport physical servers. In UNIX based systems it's very common to use XEN hypervisor.

To date, XEN is a cross-platform hypervisor with a huge number of functions and advanced capabilities, which makes it possible to use it even in a corporate environment. One of the main features of the XEN is the support for paravirtualization - a special OS kernel mode which allows to work together with XEN. Unlike emulating a separate isolated environment, this mode makes it possible to achieve much more performance. Of course, there are some limitations in paravirtualization mode:

- privileged operations are prohibited
- switching between 32-bit mode to 64-bit is not supported when running

Another feature of Xen is that the code of the hypervisor itself, starting with the 3d version, includes only the most necessary set of functions: virtual memory, processor clock, DMA and real-time clock management. All other functions, including work with the disk subsystem, peripheral devices, input and output are made in domains - currently running virtual machines. Thanks to this approach, Xen remains the lightest hypervisor - in versions 4.x, for example, the binary code takes only about 600 KB.

The history of Xen is long enough. Initially, it was a research project of one of Cambridge students, who made it a commercial version after few years. The first release was in 2003, and in four years the source code was purchased by Citrix, which was a turning point in the development of the hypervisor. The funding allowed this project to be developed vary fast and made completely free and open source with GPL license. To date, Xen is developing under Linux Foundation. The code embedded in the Linux kernel and updated regularly.

Main advantages of XEN:

- the speed of virtual machines running XEN due to paravirtualization is comparable to the performance without virtualization, directly on the hardware;

- migration of working machines between hosts is possible without stopping or suspending their work;

- each guest OS can control up to 32 processors, can change numbers of used processors, use other resources, without stopping or suspending their work;

- wide support of various platforms: x86, x64, ARM, PPC and others;

- ability to run guest OS in hardware virtualization mode (required for Windows and other non-modifiable operating systems);

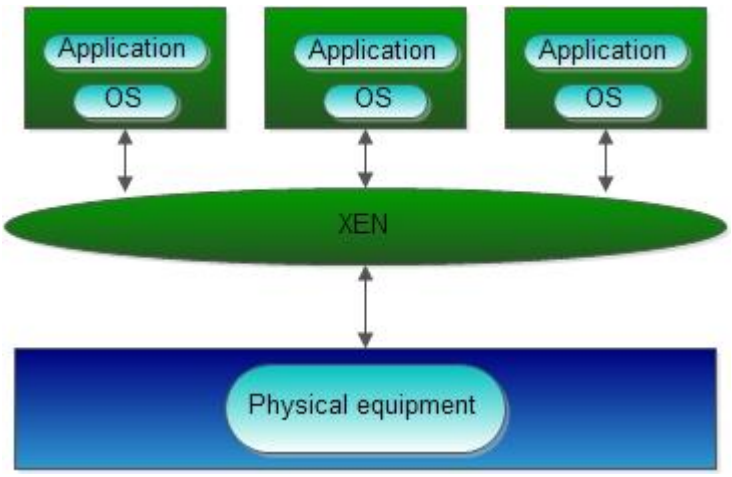
- thanks to driver support in the Linux kernel excellent compatibility with a wide range of devices

- complete free, including for commercial use.

The long history of XEN project practically avoid those problems such us instability and unpredictability in various scenarios, which is typical for new and free hypervisors.

The most often mentioned disadvantages of XEN is the high cost of deploying guest systems, which does not allow to optimize the density of virtual machines. Due to this, many providers prefer to use other solutions at the OS level (for example, OpenVZ, Virtuozzo, FreeBSD Jail).

Due to the similar file structure of virtual machines, XEN (Pic.11) supports migration to other hypervisors, for example, KVM. The only one thing is needed: to stop the virtual machine and create a disk image in VPS format. It's possible to make all those steps with native XEN tools. When the image is created, it needs to be moved to where it should be implemented. It's even easier to migrate between two different servers with the XEN hypervisor. Due to the so-called "live" migration, there is no need even to stop the machine - it can be transferred dynamically, and it is absolutely transparent to the client, without failures or disruptions in the operation of the software. (Source: https://www.xenproject.org/)



Pic.11 Xen architecture

4.2 KVM hypervisor

There are two main types of hypervisors. Type 1 hypervisors run directly on the host's hardware to control the hardware and to manage virtual machines. These are often referred to as "native" or "bare-metal" because they require no other underlying operating system. Type 1 examples include VMware vSphere ESXi hypervisor, Citrix XenServer, and the open source KVM (Kernel-based Virtual Machine).

Type 2 hypervisors are hosted, which means they must run inside an operating system that, in turn, is running on the physical hardware. Type 2 hypervisor examples include VMWare Workstation and Oracle VM VirtualBox.

KVM (Kernel-based Virtual Machine) is the leading open source complete virtualization solution on x86 hardware and it supports all major operating systems including Linux and Windows. KVM enables organizations to be agile by providing robust flexibility and scalability that fit their specific business demands. KVM converts the Linux kernel into a bare metal hypervisor and it leverages the advanced features of Intel VT-X and AMD-V x86 hardware, thus delivering unsurpassed performance levels. In addition, KVM incorporates Linux security features including SELinux(Security-Enhanced Linux) developed by the US Security Agency to add access controls, multi-level and multi-category security as well as policy enforcement. As a result, organizations are protected from compromised virtual machines which are isolated and cannot be accessed by any other processes.

Much like Linux itself, server virtualization technology is following the now-familiar trajectory toward open and standard implementations. Just as industry-standard x86 processors have steadily replaced proprietary processors, and Linux has replaced many proprietary operating systems, so too open source KVM hypervisor technology now competes directly with other virtualization solutions. This progress has been aided by the fact that industry- standard x86 processors and systems have grown increasingly more powerful, in terms of processing, memory, and I/O—and now represent attractive shared resources. In addition, the integration of key virtualization technology at the processor level by both Intel (Intel® VT) and AMD (AMD-V) has enabled virtualization to be deeply integrated at the Linux kernel level, yielding significant benefits in terms of performance, scalability, and security.
(Source: https://software.intel.com/sites/default/files/OVM_KVM_wp_Final7.pdf)

V BUILDING INFRASTRUCTURE AS A SERVICE

Basic components to build IaaS:

- OpenStack - platform for deploying clouds
- KVM hypervisor

In test environment will be created 2 machines based on Ubuntu operating system. These machines will represent Controller and Compute including:

- Keystone
- Glance
- Nova
- Neutron
- Cinder
- Horizon

This system will allow to start some quantity of virtual machines (depends on RAM and CPU which is installed on Compute machine), create virtual networks and storage. To manage all those resources will be used OpenStack Dashboard.

In test environment was used this configuration for each component:

- Controller: CPU i3-540, RAM 8Gb, 2 SSD drives 120Gb, 2 HDD 500 Gb
- Compute: CPU i7 6700K, RAM 32Gb, 2 HDD 500Gb
- Operating system: Ubuntu 14.04
- OpenStack: Kilo

Network also was divided in 4 groups:

- Management — 10.0.0.0/24 — VLAN 10
- Tunnel — 10.0.1.0/24 — VLAN 11
- Storage — 10.0.2.0/24 — VLAN 12
- External — 192.168.1.0/24

DNS server was configured for each component, it follows that Controller has 10.0.0.11 IP address and Compute 10.0.0.31 as well.

On Controller component network interfaces is assigned in this order:

```
auto p2p1.10

iface p2p1.10 inet static

   address 10.0.0.11
```

```
        netmask 255.255.255.0
        gateway 10.0.0.1
        dns-nameservers 10.0.0.1
    auto p2p1.11
    iface p2p1.11 inet static
        address 10.0.1.11
        netmask 255.255.255.0
    auto p2p1.12
    iface p2p1.12 inet static
        address 10.0.2.11
        netmask 255.255.255.0
    auto p3p1
    iface p3p1 inet manual
            up ip link set dev $IFACE up
            down ip link set dev $IFACE down
```

On Compute component network interfaces is assigned in this order:

```
auto p2p1.10
iface p2p1.10 inet static
   address 10.0.0.31
   netmask 255.255.255.0
   gateway 10.0.0.1
   dns-nameservers 10.0.0.1


auto p2p1.11
```

iface p2p1.11 inet static

address 10.0.1.31

netmask 255.255.255.0

auto p2p1.12

iface p2p1.12 inet static

address 10.0.2.31

netmask 255.255.255.0

To configure Network Time Protocol for each component was executed these commands:

- Controller:
  # apt-get install ntp -y
  # cat /etc/ntp.conf
  server 1.europe.pool.ntp.org iburst
  restrict -4 default kod notrap nomodify
  restrict -6 default kod notrap nomodify
  # service ntp stop
  # ntpdate 1.europe.pool.ntp.org
  # service ntp start
- Compute:
  # apt-get install ntp -y
  # cat /etc/ntp.conf
  server controller iburst
  # service ntp stop
  # ntpdate controller
  # service ntp start

Next step was setup Kilo repository from ubuntu-cloud.archive.canonical.com:

apt-get install ubuntu-cloud-keyring

echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu" "trusty-updates/kilo main" > /etc/apt/sources.list.d/cloudarchive-kilo.list

In role of SQL server it could be MySQL, PostgreSQL, Oracle and etc. According to official manual was used MariaDM and configured in this way:

```
# apt-get install mariadb-server python-mysqldb -y

# cat /etc/mysql/conf.d/mysqld_openstack.cnf

[mysqld]

bind-address = 10.0.0.11

default-storage-engine = innodb

innodb_file_per_table

collation-server = utf8_general_ci

init-connect = 'SET NAMES utf8'

character-set-server = utf8
```

Also RabbitMQ:

```
# apt-get install rabbitmq-server

# rabbitmq-plugins enable rabbitmq_management

# service rabbitmq-server restart
```

To use RabbitMO it’s necessary to create user and set relevant permissions:

```
rabbitmqctl add_user openstack RABBIT_PASS

rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

5.1 Installing and configuring Keystone

Keystone is an authorization center for OpenStack platform. Keystone data is stored in SQL database and in memcache. So in this way was created keystone database:

CREATE DATABASE keystone;

GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'KEYSTONE_DBPASS';

GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'KEYSTONE_DBPASS';

In configuration file /etc/keystone/keystone.conf:

[DEFAULT]

admin_token = ADMIN_TOKEN

[database]

connection = mysql://keystone:KEYSTONE_DBPASS@controller/keystone

[memcache]

servers = localhost:11211

[token]

provider = keystone.token.providers.uuid.Provider

driver = keystone.token.persistence.backends.memcache.Token

[revoke]

driver = keystone.contrib.revoke.backends.sql.Revoke

ADMIN_TOKEN was generated by openssl: openssl rand –hex 16

To configure apache web server it’s necessary to make some changes in apache2.conf and wsgi-keystone.conf files:

cat /etc/apache2/apache2.conf

```
ServerName controller

# cat /etc/apache2/sites-available/wsgi-keystone.conf

Listen 5000

Listen 35357

<VirtualHost *:5000>

   WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone display-name=%{GROUP}

   WSGIProcessGroup keystone-public

   WSGIScriptAlias / /var/www/cgi-bin/keystone/main

   WSGIApplicationGroup %{GLOBAL}

   WSGIPassAuthorization On

   <IfVersion >= 2.4>

     ErrorLogFormat "%{cu}t %M"

   </IfVersion>

   LogLevel info

   ErrorLog /var/log/apache2/keystone-error.log

   CustomLog /var/log/apache2/keystone-access.log combined

</VirtualHost>

<VirtualHost *:35357>

   WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone display-name=%{GROUP}

   WSGIProcessGroup keystone-admin

   WSGIScriptAlias / /var/www/cgi-bin/keystone/admin

   WSGIApplicationGroup %{GLOBAL}

   WSGIPassAuthorization On
```

```
    <IfVersion >= 2.4>
      ErrorLogFormat "%{cu}t %M"
    </IfVersion>
    LogLevel info
    ErrorLog /var/log/apache2/keystone-error.log
    CustomLog /var/log/apache2/keystone-access.log combined
</VirtualHost>
# ln -s /etc/apache2/sites-available/wsgi-keystone.conf /etc/apache2/sites-enabled
# mkdir -p /var/www/cgi-bin/keystone
# curl http://git.openstack.org/cgit/openstack/keystone/plain/httpd/keystone.py?h=stable/kilo | tee
/var/www/cgi-bin/keystone/main /var/www/cgi-bin/keystone/admin
# chown -R keystone:keystone /var/www/cgi-bin/keystone
# chmod 755 /var/www/cgi-bin/keystone/*
# service apache2 restart
# rm -f /var/lib/keystone/keystone.db
```

It was also necessary to configure endpoints, which will give OpenStack access to other services.

To make it easier global values was created with these names:

```
# export OS_TOKEN=ADMIN_TOKEN
# export OS_URL=http://controller:35357/v2.0
```

At this step was created services with name “keystone” and API endpoint:

```
# openstack service create  --name keystone --description "OpenStack Identity" identity
# openstack endpoint create  --publicurl http://controller:5000/v2.0 --internalurl
http://controller:5000/v2.0 --adminurl http://controller:35357/v2.0 --region RegionOne  identity
```

Creating admin users:

```
# openstack project create --description "Admin Project" admin
```

openstack user create --password-prompt admin

openstack role create admin

openstack role add --project admin --user admin admin

Creating Service Project and demo:

openstack project create --description "Service Project" service

openstack project create --description "Demo Project" demo

openstack user create --password-prompt demo

openstack role create user

openstack role add --project demo --user demo user

Environment scripts for Admin and Demo in admin-openrc.sh and demo-openrc.sh as well.

cat admin-openrc.sh

export OS_PROJECT_DOMAIN_ID=default

export OS_USER_DOMAIN_ID=default

export OS_PROJECT_NAME=admin

export OS_TENANT_NAME=admin

export OS_USERNAME=admin

export OS_PASSWORD=ADMIN_PASS

export OS_AUTH_URL=http://controller:35357/v3

export OS_IMAGE_API_VERSION=2

export OS_VOLUME_API_VERSION=2

cat demo-openrc.sh

export OS_PROJECT_DOMAIN_ID=default

export OS_USER_DOMAIN_ID=default

```
export OS_PROJECT_NAME=demo

export OS_TENANT_NAME=demo

export OS_USERNAME=demo

export OS_PASSWORD=DEMO_PASS

export OS_AUTH_URL=http://controller:5000/v3

export OS_IMAGE_API_VERSION=2

export OS_VOLUME_API_VERSION=2

# source admin-openrc.sh
```

5.2 Installing and configuring Glance

First of all, glace database should be created:

```
# mysql -u root -p

CREATE DATABASE glance;

GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY
'GLANCE_DBPASS';

GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY
'GLANCE_DBPASS';
```

The user glance should exist with admin rights, because all services will execute with glace role:

```
# openstack user create --password-prompt glance

# openstack role add --project service --user glance admin

# openstack service create --name glance  --description "OpenStack Image service" image

# openstack endpoint create --publicurl http://controller:9292 --internalurl http://controller:9292 --
adminurl http://controller:9292 --region RegionOne image
```

Glance installation and configuration:

```
# apt-get install glance python-glanceclient

# cat /etc/glance/glance-api.conf

[DEFAULT]

...

notification_driver = noop



[database]

connection = mysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]

auth_uri = http://controller:5000
```

auth_url = http://controller:35357

auth_plugin = password

project_domain_id = default

user_domain_id = default

project_name = service

username = glance

password = GLANCE_PASS

[paste_deploy]

flavor = keystone

[glance_store]

default_store = file

filesystem_store_datadir = /var/lib/glance/images/

File /etc/glance/glance-registry.conf should have same information in default, paste_deploy, keysone_authtoken and database fields.

Data base should be also synchronized:

su -s /bin/sh -c "glance-manage db_sync" glance

Restart services and delete local data base:

service glance-registry restart

service glance-api restart

rm -f /var/lib/glance/glance.sqlite

Images should be downloaded for Ubuntu OS:

mkdir /tmp/images

wget -P /tmp/images http://cloud-images.ubuntu.com/releases/14.04.2/release/ubuntu-14.04-server-cloudimg-amd64-disk1.img

glance image-create --name "Ubuntu-Server-14.04.02-x86_64" --file /tmp/images/ubuntu-14.04-server-cloudimg-amd64-disk1.img --disk-format qcow2 --container-format bare --visibility public --progress

rm -r /tmp/images

At this point service Glance is installed and configured.

5.3 Installing and configuring Nova.

Nova is a main part of IaaS based on OpenStack, because it creates virtual machines automatically. Nova can be based on hypervisors like KVM, Xen, Hyper-V and etc. According to comparison shown before KVM was a best decision.

Again, nova database should be created:

```
# mysql -u root -p

CREATE DATABASE nova;

GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY
'NOVA_DBPASS';

GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY 'NOVA_DBPASS';
```

Adding information about nova into keystone and installing necessary packets:

```
# openstack user create --password-prompt nova

# openstack role add --project service --user nova admin

# openstack service create --name nova --description "OpenStack Compute" compute

# openstack endpoint create --publicurl http://controller:8774/v2/%\(tenant_id\)s --internalurl
http://controller:8774/v2/%\(tenant_id\)s --adminurl http://controller:8774/v2/%\(tenant_id\)s --
region RegionOne compute

# apt-get install nova-api nova-cert nova-conductor nova-consoleauth nova-novncproxy nova-
scheduler python-novaclient
```

Nova configuration file (nova.conf) should be also changed:

```
[DEFAULT]

rpc_backend = rabbit

auth_strategy = keystone

my_ip = 10.0.0.11

vncserver_listen = 10.0.0.11

vncserver_proxyclient_address = 10.0.0.11
```

[database]

connection = mysql://nova:NOVA_DBPASS@controller/nova

[oslo_messaging_rabbit]

rabbit_host = controller

rabbit_userid = openstack

rabbit_password = RABBIT_PASS

[keystone_authtoken]

auth_uri = http://controller:5000

auth_url = http://controller:35357

auth_plugin = password

project_domain_id = default

user_domain_id = default

project_name = service

username = nova

password = NOVA_PASS

[glance]

host = controller

[oslo_concurrency]

lock_path = /var/lib/nova/tmp

Data base synchronization, service restarting and deleting local database:

su -s /bin/sh -c "nova-manage db sync" nova

service nova-api restart

service nova-cert restart

service nova-consoleauth restart

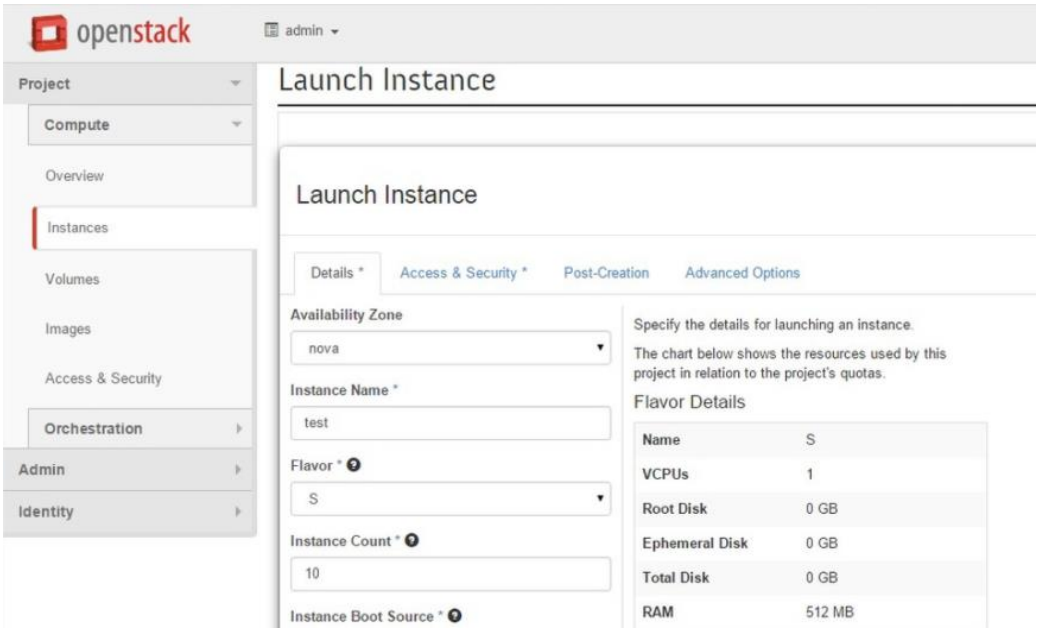
service nova-scheduler restart

service nova-conductor restart

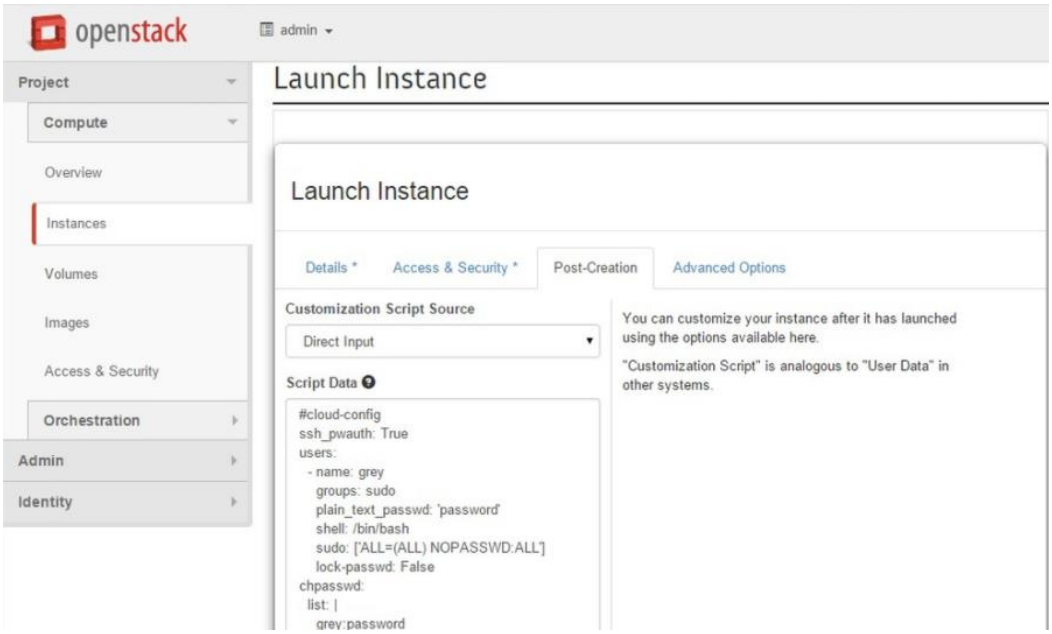
service nova-novncproxy restart

rm -f /var/lib/nova/nova.sqlite

At this step service Nova is installed and configured. Configuring of Compute, Neutron, Cinder, Horison and Network implementation is described in Attachments.



Pic. 12 Web interface



Pic. 13 Web interface

CONCLUSIONS

In today's environment, companies can quickly leverage a combination of many different cloud services to get new innovative products to market faster and cheaper than before. Now that enterprises and goverments are investing heavily in cloud technologies, hybrid models are becoming more mature. With the increase of trust in hybrid models, cloud adoption is quikly rising and the barriers to entry are lowering. Procuring and managing infrastructure is becoming less of a bottleneck now that provisioning infrastructure can be done through code. And given that infrastructure can be treated as code, practitioners are looking at new ways of building and managing software to increase agility.

In case company will decide to move their business infrastructure to the cloud, the first task they will face with is to choose a platform which will satisfy all company's requirements. It may be difficult to find what is behind the vendor's promises.

In this master's thesis researcher described cloud computing architecture, deployment models, such us private cloud, community cloud, public cloud and hybrid cloud. Deeply explain main characteristic of each service models. Also described cloud computing technologies, namely virtualization, service-oriented architecture, grid computing and utility computing. Made overview of service platforms such us OpenStack, CloudStack, Eucalyptus and vCloud Director. According to received information about architecture and main futures of these technologies researcher provided general, functional and property comparison between all of them, chose which service platform is worth to use and implement it in virtual environment with description of all executed commands.

There is no solution, and even the most effective cloud platform that will fully satisfy all of the stress and use cases for each company. For a specific type of business, company need to understand and specify all the factors involved, describe their typical tasks, find outpossible risks, grant a budget, and compare it against the platforms' capabilities and license prices.

REFERENCES

1. Fundamentals on building a reliable cloud-based saas architecture http://usersnap.com/blog/cloud-based-saas-architecture-fundamentals/

2. Использование виртуализации на основе KVM http://www.ibm.com/developerworks/ru/library/l-using-kvm/

3. NIST SP 800-145: A NIST Definition of Cloud Computing; refer to: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

4.NIST SP 800-145, "A NIST definition of cloud computing", http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf

5. NIST SP 800-146, "NIST Cloud Computing Synopsis and Recommendations", http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf

6. Cloud Security Alliance: TCI – Reference Architecture; refer to: https://cloudsecurityalliance.org/wp-content/uploads/2011/10/TCI-Reference-Architecture-v1.1.pdf.

7. Jiamei Tang, Sangwook Kim. (2015) A Service-oriented device selection solution based on user satisfaction and device performance in a ubiquitous environment. Multimedia Tools and Applications

8. Ville Alkkiomäki, Kari Smolander. (2015) Anatomy of one service-oriented architecture implementation and reasons behind low service reuse. Service Oriented Computing and Applications.

9. Fu Hou, Xinjun Mao, Wei Wu, Lu Liu, John Panneerselvam. (2014) A Cloud-Oriented Services Self-Management Approach Based on Multi-agent System Technique. 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing

10. Michael Hahn, Santiago Gomez Saez, Vasilios Andrikopoulos, Dimka Karastoyanova, Frank Leymann. (2014) Development and Evaluation of a Multi-tenant Service Middleware PaaS Solution.

11. Rustem Dautov, Iraklis Paraskakis, Mike Stannett. (2014) Utilising stream reasoning techniques to underpin an autonomous framework for cloud application platforms. Journal of Cloud Computing

12. Lucas Bueno R. Oliveira, Diogo Brandao Martins, Felipe Augusto Amaral, Flavio Oquendo, Elisa Yumi Nakagawa. (2014) Automating Cataloging and Discovery of Services for Service-Oriented Robotic Systems.

13. E. del Val, M. Rebollo, V. Botti. (2014) Combination of self-organization mechanisms to enhance service discovery in open systems. Information Sciences

14. Elisa Yumi Nakagawa, Flavio Oquendo, José Carlos Maldonado. 2014. Reference Architectures. Software Architecture

15. Hamza Chehili, Lionel Seinturier, Mahmoud Boufaida. (2013) FASOAD: A Framework for Agile Service-Oriented Architectures Development. 2013 24th International Workshop on Database and Expert Systems Applications

16. Information Technology: New Generations (ITNG), 2010 Seventh International Conference on. Service-Oriented Cloud Computing Architecture

17. High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities

18. Internet Computing, IEEE (Volume:13 , Issue: 5). Cloud Computing: Distributed Internet Computing for IT and Scientific Research

19. CISCO, "Cisco Cloud Computing - Data Center Strategy, Architecture, and Solutions", http://www.cisco.com/web/strategy/docs/gov/CiscoCloudComputing_WP.pdf

20. SNIA, "Cloud Storage for Cloud Computing", www.snia.org/cloud/CloudStorageForCloudComputing.pdf

21. https://docs.openstack.org/ "OpenStack Documentation"

22. Robert G. Siebeck et al., "Cloudbased Enterprise Mashup Integration Services for B2B Scenarios", MEM2009 workshop, Spain, 2009

23. Arista, "Cloud Networking: Design Patterns for 'Cloud Centric' Application Environments", January 2009.

24. VMWare Inc., VMWare, http://www.vmware.com

25. KVM Project, Kernel based virtual machine, http://www.linux-kvm.org

26. https://www.xenproject.org/ Xen prject

27.http://www.dxc.technology/cloud/offerings/140041/140149-eucalyptus_software_support_services Eucalyptus Software Support Services

28. http://masters.donntu.org/2008/fvti/dzeba/library/lib3.htm Перспективы грид: грид-компьютинг – следующее поколение распределённого компьютинга

29. Science, Engineering & Education, 1, (1), 2016, 83-88: Big data and cloud computing – issues and problems

30. Security, trust, and regulatory aspects of Cloud Computing is Business Environments. S. Srinivasan Texas Southern University, USA

31. The Definitive Guide to Modern Supply Chain Management: Chad W. Autry,Thomas J. Goldsby,John Bell,Mark A. Moon,Chuck Munson,Michael Watson,Sara Lewis,Peter Cacioppi,Jay Jayaraman

32. Cloud Computing: First International Conference, CloudCom 2009, Beijing

33. http://en.bmstu.wiki/Apache_CloudStack

33. Inter-cooperative Collective Intelligence: Techniques and Applications, Authors: Fatos Xhafa, Nik Bessis, 2014

34. Cloud Computing: Principles and Paradigms. Author: Rajkuma Buyya, James Broberg, Andrzej Goscinski

35. Scientific Cloud Computing: Early Definition and Experience. Author: Lizhe Wang, Jie Tao, Marcel Kunze

36. Cloud Platform Comparison. Author: Vadim Truksha. 2015

37. http://www.fastcat.co/document/openstack/openstack_install_controller

38. http://docs.cloudstack.apache.org/en/latest/concepts.html

39. https://software.intel.com/sites/default/files/OVM_KVM_wp_Final7.pdf

40. International Journal of Trend in Research and Development, Volume 4(1), ISSN: 2394-9333: A review on Cloud Computing

ATTACHMENT 1. Installing and configuring Compute

First of all, installation and configuration:

apt-get install nova-compute sysfsutils

[DEFAULT]

verbose = True

rpc_backend = rabbit

auth_strategy = keystone

my_ip = 10.0.0.31 #MANAGEMENT_INTERFACE_IP_ADDRESS

vnc_enabled = True

vncserver_listen = 0.0.0.0

vncserver_proxyclient_address = 10.0.0.31 #MANAGEMENT_INTERFACE_IP_ADDRESS

novncproxy_base_url = http://controller:6080/vnc_auto.html

[oslo_messaging_rabbit]

rabbit_host = controller

rabbit_userid = openstack

rabbit_password = RABBIT_PASS

[keystone_authtoken]

auth_uri = http://controller:5000

auth_url = http://controller:35357

auth_plugin = password

project_domain_id = default

user_domain_id = default

project_name = service

username = nova

password = NOVA_PASS

[glance]

host = controller

[oslo_concurrency]

lock_path = /var/lib/nova/tmp

[libvirt]

virt_type = kvm

MANAGEMENT_INTERFACE_IP_ADDRESS is an IP address from VLAN 10 field. In novncproxy_base_url controller parameter should be vnc address, in other case it won't be possible to access VNC console from Horizon.

After this step Nova service should be restarted and local data base should be deleted also:

service nova-compute restart

rm -f /var/lib/nova/nova.sqlite

To check if all services in UP status this command have to be executed:

nova service-list

Basically, at this point IaaS is built, but network part is still not configured. So next step will be installing and configuring Neutron service

ATTACHMENT 2. Installing and configuring Neutron

In this test environment network kernel will be installed on controller, but in official manual 3d node is used. In case if computing nodes will be more than 10 and/or big network traffic, it's better to migrate network-server on separate note.

As before, creating database, creating information about nova into keystone and installing necessary packets:

mysql -u root -p

CREATE DATABASE neutron;

GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY 'NEUTRON_DBPASS';

GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'NEUTRON_DBPASS';

openstack user create --password-prompt neutron

openstack role add --project service --user neutron admin

openstack service create --name neutron --description "OpenStack Networking" network

openstack endpoint create --publicurl http://controller:9696 --adminurl http://controller:9696 --internalurl http://controller:9696 --region RegionOne network

apt-get install neutron-server neutron-plugin-ml2 python-neutronclient neutron-plugin-openvswitch-agent neutron-l3-agent neutron-dhcp-agent neutron-metadata-agent

Make changes in sysctl.conf, neutron.conf, ml2_conf.ini, l3_agent_ini, dhcp_agent.ini, dnsmasq-neutron.conf, metadata_agent.ini, nova.conf:

cat /etc/sysctl.conf

net.ipv4.ip_forward=1

net.ipv4.conf.all.rp_filter=0

net.ipv4.conf.default.rp_filter=0

sysctl –p

cat /etc/neutron/neutron.conf

```
[DEFAULT]

...

rpc_backend = rabbit

auth_strategy = keystone

core_plugin = ml2

service_plugins = router

allow_overlapping_ips = True

notify_nova_on_port_status_changes = True

notify_nova_on_port_data_changes = True

nova_url = http://controller:8774/v2


[oslo_messaging_rabbit]

rabbit_host = controller

rabbit_userid = openstack

rabbit_password = RABBIT_PASS


[database]

connection = mysql://neutron:NEUTRON_DBPASS@controller/neutron


[keystone_authtoken]

auth_uri = http://controller:5000

auth_url = http://controller:35357

auth_plugin = password

project_domain_id = default
```

user_domain_id = default

project_name = service

username = neutron

password = NEUTRON_PASS

[nova]

auth_url = http://controller:35357

auth_plugin = password

project_domain_id = default

user_domain_id = default

region_name = RegionOne

project_name = service

username = nova

password = NOVA_PASS

/etc/neutron/l3_agent.ini

[DEFAULT]

interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver

external_network_bridge =

router_delete_namespaces = True

/etc/neutron/dhcp_agent.ini

[DEFAULT]

interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver

dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq

dhcp_delete_namespaces = True

dnsmasq_config_file = /etc/neutron/dnsmasq-neutron.conf

/etc/neutron/dnsmasq-neutron.conf

dhcp-option-force=26,1454

/etc/neutron/metadata_agent.ini

[DEFAULT]

auth_uri = http://controller:5000

auth_url = http://controller:35357

auth_region = RegionOne

auth_plugin = password

project_domain_id = default

user_domain_id = default

project_name = service

username = neutron

password = NEUTRON_PASS

nova_metadata_ip = controller

metadata_proxy_shared_secret = METADATA_SECRET

/etc/nova/nova.conf

[DEFAULT]

...

network_api_class = nova.network.neutronv2.api.API

security_group_api = neutron

linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver

firewall_driver = nova.virt.firewall.NoopFirewallDriver

[neutron]

url = http://controller:9696

auth_strategy = keystone

admin_auth_url = http://controller:35357/v2.0

admin_tenant_name = service

admin_username = neutron

admin_password = NEUTRON_PASS

service_metadata_proxy = True

metadata_proxy_shared_secret = METADATA_SECRET

Database synchronizing and services restarting:

su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron

service nova-api restart

service neutron-server restart

service openvswitch-switch restart

Now creating bridge and connect with external interface. Restarting:

ovs-vsctl add-br br-ex

ovs-vsctl add-port br-ex p3p1

service neutron-plugin-openvswitch-agent restart

service neutron-l3-agent restart

service neutron-dhcp-agent restart

service neutron-metadata-agent restart

Installing additional plugins and configuration:

apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent

/etc/neutron/neutron.conf

[DEFAULT]

...

rpc_backend = rabbit

auth_strategy = keystone

core_plugin = ml2

service_plugins = router

allow_overlapping_ips = True

[oslo_messaging_rabbit]

rabbit_host = controller

rabbit_userid = openstack

rabbit_password = RABBIT_PASS

[keystone_authtoken]

auth_uri = http://controller:5000

auth_url = http://controller:35357

auth_plugin = password

project_domain_id = default

user_domain_id = default

project_name = service

username = neutron

password = NEUTRON_PASS

/etc/neutron/plugins/ml2/ml2_conf.ini

[ml2]

type_drivers = flat,vlan,gre,vxlan

tenant_network_types = gre

mechanism_drivers = openvswitch

[ml2_type_gre]

tunnel_id_ranges = 1000:2000

[ml2_type_flat]

flat_networks = external

[securitygroup]

enable_security_group = True

enable_ipset = True

firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver

[ovs]

local_ip = 10.0.1.31 #INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS

bridge_mappings = external:br-ex

[agent]

tunnel_types = gre

Openvswitch should be restarted now:

service openvswitch-switch restart

And again make some changes to nova.cong:

[DEFAULT]

...

network_api_class = nova.network.neutronv2.api.API

security_group_api = neutron

linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver

firewall_driver = nova.virt.firewall.NoopFirewallDriver

[neutron]

url = http://controller:9696

auth_strategy = keystone

admin_auth_url = http://controller:35357/v2.0

admin_tenant_name = service

admin_username = neutron

admin_password = NEUTRON_PASS

After all this changes nova-compute and neutron-plugin should be restarted:

service nova-compute restart

service neutron-plugin-openvswitch-agent restart

ATTACHMENT 3. Network implementation

At this step network template would be created. Creating virtual network:

neutron net-create ext-net --router:external --provider:physical_network external --provider:network_type flat

Configuring external network:

```
# neutron subnet-create ext-net 192.168.1.0/24 --name ext-subnet \
  --allocation-pool start=192.168.1.100,end=192.168.1.200 \
  --disable-dhcp --gateway 192.168.1.1
```

External network is 192.168.1.0/24

Default gateway 192.168.1.1.

All external IP addresses is in range 192.168.1.101-200.

Next step – creating internal network for demo project, so first of all, download variables for demo user:

source demo-openrc.sh

Now it’s time to create virtual inner network:

neutron net-create demo-net

neutron subnet-create demo-net 172.16.1.0/24 --name demo-subnet --gateway 172.16.1.1

This virtual network is 172.16.1.0/24.

Default gateway 172.16.1.1 – this is virtual router.

To create this virtual router, it’s necessary to configure demo-subnet interfaces and connect it to external network:

neutron router-create demo-router

neutron router-interface-add demo-router demo-subnet

neutron router-gateway-set demo-router ext-net

At this point, “cloud” is configured and contains network.

ATTACHMENT 4. Installing and configuring Cinder

With Cinder it's possible to manage block devices (virtual drives), connect them to each virtual instances. This virtual drives can be bootable, so in this case it's much easier to transfer VM to another compute instance.

The procedure of installing and configuring is very similar to previous:

mysql -u root -p

CREATE DATABASE cinder;

GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'CINDER_DBPASS';

GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'CINDER_DBPASS';

openstack user create --password-prompt cinder

openstack role add --project service --user cinder admin

openstack service create --name cinder --description "OpenStack Block Storage" volume

openstack service create --name cinderv2 --description "OpenStack Block Storage" volumev2

openstack endpoint create --publicurl http://controller:8776/v2/%\(tenant_id\)s --internalurl http://controller:8776/v2/%\(tenant_id\)s --adminurl http://controller:8776/v2/%\(tenant_id\)s --region RegionOne volume

openstack endpoint create --publicurl http://controller:8776/v2/%\(tenant_id\)s --internalurl http://controller:8776/v2/%\(tenant_id\)s --adminurl http://controller:8776/v2/%\(tenant_id\)s --region RegionOne volumev2

apt-get install cinder-api cinder-scheduler python-cinderclient

cat /etc/cinder/cinder.conf

[DEFAULT]

...

rpc_backend = rabbit

auth_strategy = keystone

my_ip = 10.0.0.11

[oslo_messaging_rabbit]

rabbit_host = controller

rabbit_userid = openstack

rabbit_password = RABBIT_PASS

[database]

connection = mysql://cinder:CINDER_DBPASS@controller/cinder

[keystone_authtoken]

auth_uri = http://controller:5000

auth_url = http://controller:35357

auth_plugin = password

project_domain_id = default

user_domain_id = default

project_name = service

username = cinder

password = CINDER_PASS

[oslo_concurrency]

lock_path = /var/lock/cinder

su -s /bin/sh -c "cinder-manage db sync" cinder

service cinder-scheduler restart

service cinder-api restart

Since controller is also a storage, so next steps is applicable:

apt-get install qemu lvm2

pvcreate /dev/md1

vgcreate cinder-volumes /dev/md1

In this part physical LVM device was created with cinder-volumes lvm-group.

Install necessary packets and change cinder configuration file:

apt-get install cinder-volume python-mysqldb

[DEFAULT]

enabled_backends = lvm

glance_host = controller

[lvm]

volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver

volume_group = cinder-volumes

iscsi_protocol = iscsi

iscsi_helper = tgtadm

And restart services:

service tgt restart

service cinder-scheduler restart

service cinder-api restart

service cinder-volume restart

ATTACHMENT 5. Installing and configuring Horizon

Horizon dashboard is a web interface for OpenStack. It’s written on Python 2.7 and has Django kernel. It helps to manage OpenStack environment: control user/projects/roles, managing images, virtual drives, instances, network and etc.

Installing process (will be installed on controller):

```
# apt-get install openstack-dashboard

OPENSTACK_HOST = "controller"

ALLOWED_HOSTS = '*'

...

CACHES = {

  'default': {

     'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',

     'LOCATION': '127.0.0.1:11211',

  }

}

OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"

TIME_ZONE = "Europe/Vilnius "
```

Now it’s time to restart Apache web server and connect to web interface controller/horizon (Pic. 12-13):

```
# service apache2 reload
```