Information technology for the deployment of the OpenStack cloud environment

Information technology for the deployment of the OpenStack cloud environment

V I Shevchenko, O V Chengar and T A Kokodey

Sevastopol State University, st. Universitetskaya 33, Sevastopol, Russia

E-mail: tanya.kokodey@gmail.com

Abstract. This article discusses the information technology of implementing and adapting cloud services for a research laboratory using the OpenStack open environment as an example. The structure of the cloud computing environment is developed. An example of the implementation of the installation and configuration of client and server applications using three computers is given. The proposed technology will allow the user to deploy and configure the OpenStack environment with basic services for installing virtual machines with an internal network and a set of external addresses.

1. Introduction

Cloud computing is a computing model in which resources such as computing power, data storage, network and software are abstracted and provided as services over the Internet in remote access. Billing models for these services are usually similar to those adopted for utility services. Availability, ease of resource allocation, dynamic and almost infinite scaling are key attributes of cloud computing [1]. Setting up an infrastructure using a cloud computing model is usually referred to as a “cloud." The main categories of services available in the cloud are infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) [2].

OpenStack is a collection of open source software that enterprises / service providers can use to configure and run their cloud computing and storage infrastructure. There are five main service families in OpenStack:

- Nova – calculation service;
- Swift – data storage service;
- Glance – image service;
- Keystone – authorization service;
- Horizon – User Interface (UI) service.

The basic architecture of OpenStack is shown in figure 1. Three elements of architecture interact with all components of the system. Horizon is a graphical user interface that allows administrators to manage all projects with maximum ease. Keystone manages authorized users, and Neutron allows you to define networks that provide communications between components. Nova distributes workloads. Computing machines usually require some form of persistent storage, which can be either block (Cinder) or object (Swift). In addition, Nova requires an image to run the instance. Glance processes this request and can use Swift as internal storage.

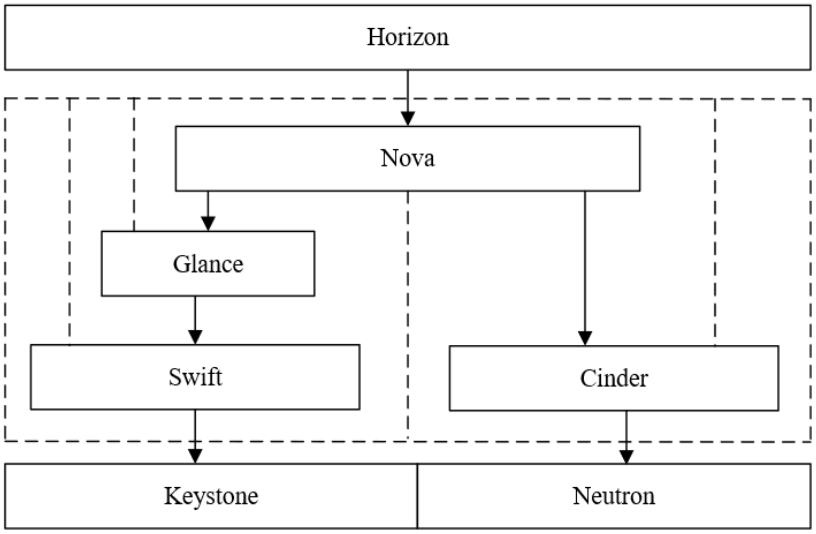

Figure 1. Generalized Architecture of OpenStack.

The OpenStack architecture encourages maximum independence for each project. This allows users to deploy only the subset of functionality they require and integrate this subset with other systems and technologies that offer similar or complementary functions. Nevertheless, this independence should not mask the fact that for unhindered operation a fully functional private cloud is likely to require almost all functionality, while all elements must be closely integrated [3].

This article discusses the deployment and configuration of a minimum cloud infrastructure on the OpenStack platform. The main task is to configure the environment for testing the research laboratory software, with the ability to host N virtual machines (VMs) with various operating systems (OS) on a minimum number of "real" devices.

The problem under consideration is relevant, since the introduction and adaptation of OpenStack open-source cloud services in the processes occurring as part of research activities in the laboratory will allow: reduction of costs associated with the purchase of expensive equipment, emulation of computing complex functions which feature complex architecture necessary to ensure research activities and provide scalability and flexibility in architecture and information technology.

2. Overview of existing technologies for the implementation and adaptation of cloud services

At present, there are various approaches and technologies for implementing and adapting the cloud services of the OpenStack open environment.

In research [1], the process of deploying the OpenStack cloud environment on two computers (Controller and Compute) in the following configuration was considered: Keystone, Glance, Nova, Neutron, Cinder, Horizon. This approach enables the user to; run many virtual machines (as much as RAM and CPU will allow), create virtual networks, create virtual disks and connect them to virtual machines, and to control and administer through UI Horizon service.

In research [5], the technology of creating an OpenStack "all-in-one" system was proposed, which can be further expanded by creating additional nodes for the computation server — nova-compute and the storage server — Swift.

In research [6], a cloud storage implementation technology is considered that transforms a set of disconnected public servers into a scalable, durable, easy-to-manage storage system based on OpenStack Swift.

The main disadvantage of the above technologies is the lack of a clear schematic representation of the deployment processes of the OpenStack environment. In addition, the technologies presented do

not have a sufficient degree of information for use as a basic approach for deploying a full OpenStack environment in a research laboratory.

3. Description of the structure of the cloud computing environment

To solve this problem, three computers (VMs) were used: Server1, Server2 and Client1. VM Server1 is designed to run all components of the system, namely: Nova, Glance, Swift, Keystone and Horizon (OpenStack Dashboard). VM Server2 is designed to run only nova-compute. During installation, a feature of the OpenStack environment is used, which is associated with the absence of a common direct interaction policy between components; each component or group of components can be installed on any server. VM Client1 is an optional component. In our case, it is used to complete images as a client to the web interface and runs OpenStack commands to manage the infrastructure. The presence of this client ensures that users do not need to interfere with the server for tasks such as configuration.

Using the approach proposed in this article, we can deploy an OpenStack environment with basic services for deployment of VM with an internal network and a set of external addresses.

The generalized structure of the cloud computing environment is presented in figure 2.

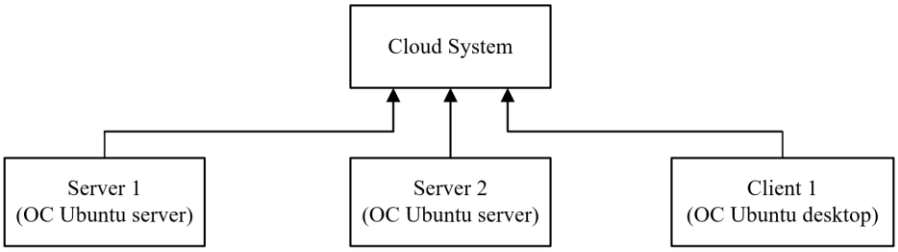

Figure 2. Generalized Cloud Computing Environment.

The Ubuntu Server was chosen as the operating system (OS) for Server1 and Server2, since this OS is free, it has high stability, performance and security. During installation, you only need to select OpeSSH-Server (a freeware version of the server tools family for remote computer management and file transfer using the Secure Shell (SSH) protocol). For Client1, the base version of the Ubuntu Desktop operating system was chosen, which is also a free, high-performance OS based on the Linux family OS. The choice of OS based on the architecture of the common Linux kernel will significantly simplify the process of integration and interaction of systems in the future.

All computers included in the system were connected to a local network with Internet access to download the necessary software packages. In the local network, all computers have static IP addresses. This is necessary for the correct configuration and functioning of the OpenStack modules.

The deployed cloud environment on three computers is described using the UML diagram, which is shown in fiure 3.

Figure 3 shows the three computers used in the system. The attributes are OS versions. The diagram also shows on which computers the specific OpenStack modules and other necessary software packages are installed.

As can be seen from figure 3, all computers contain an NTP server. It provides time synchronization for all OpenStack components. Also, on Server1 and Server2 the Bridge-utils software package is installed, which is necessary to support network bridges.

The diagram shows that the MySQL database is installed only on Server1. At the same time, a separate database has been created for the Nova, Glance and Keystone modules.

The Keystone module includes only Server1 (figure 3). Keystone provides authentication and access policy service for all components in the OpenStack family. This is implemented by its own RETS based on the API (Authentication API). The Keystone module provides authentication and authorization for the Swift, Glance, and Nova components. Authentication verifies that the request

really came from the one who actually does it. Authorization checks if the authorized user really has access to the services that he requests.

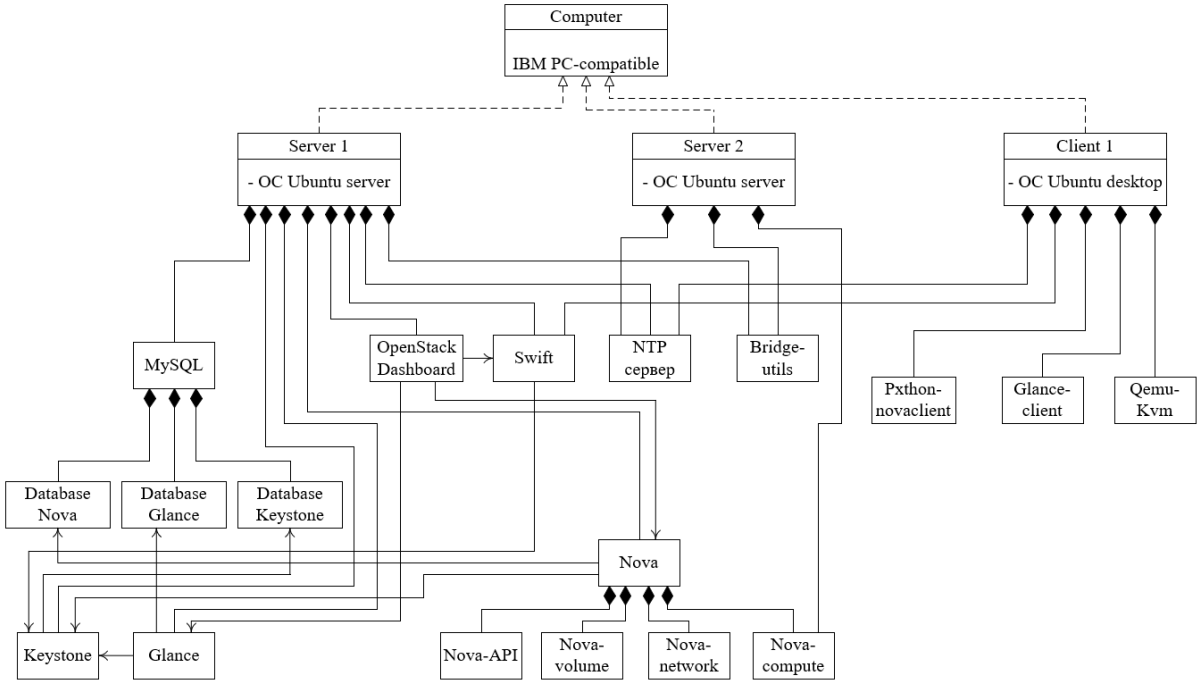

Figure. 3. Diagram of classes of for the deployed cloud environment.

The Glance module, like Keystone, includes only Server1 (figure 3). Glance is an OpenStack image service. It provides system mapping and recovery for a virtual machine image.

Nova is the computing factory controller for the OpenStack cloud. All events necessary to support the life cycle of events in the OpenStack cloud are handled by Nova. This is done by the Nova control platform, which manages computing, network, and authorization resources. But Nova does not provide virtualization capabilities per se; instead, it uses the Libvirt API to interact with supported hypervisors. Nova unleashes its full potential through the web services API, compatible with Amazon's EC2 Web Services APIs.

As you can see from the diagram (figure 3), Nova consists of four components, with Server1 including all components, and Server2 only Nova-compute. Consider each component.

The Nova API server provides an interface to the outside world for interacting with the cloud infrastructure. The API server is the only component of the outside world used to manage the infrastructure.

Nova-volume is used to manage LVM volumes. Nova-volume performs volume-related functions, such as creating, deleting, and/or expanding a volume, and disconnecting a volume from an instance. Volumes provide persistent storage for instances which not permanent as the root partition, and any changes to it are lost when the instance is deleted. When the volume is separated from the instance, or when the instance to which the volume is attached is deleted, it saves the data stored in it. This data may be available to reattach the volume to the same instance or by attaching it to other instances.

Nova-network is a network controller. It interacts with the host network configuration. Nova-network performs operations such as allocating IP addresses, configuring VLANs for projects, implementing security groups, and configuring networks.

Nova-compute is a Nova compute component. It works with a lifecycle management instance. Nova-compute receives various requests. For example, life cycle management through a message queue, and performance of corresponding operations. Thus, in accordance with the diagram (figure 3),

Server2 is a computing server. To ensure maximum performance, only one OpenStack module is installed on it.

Swift is a storage infrastructure that provides distribution, ultimately matching a virtual storage object for OpenStack. Swift is similar to Amazon's Simple Storage Service. Swift is capable of storing billions of objects distributed across nodes. Swift has built-in redundancy and resiliency management, as well as the ability to archive and stream media. This is a significantly scalable system in terms of size (several petabytes) and volume (number of objects). The diagram (figure 3) shows that Swift components are installed not only on Server1, but also on Client1.

OpenStack dashboard is an administrative web interface. The Horizon module is based on a dashboard that can be used to manage and administer OpenStack services. It can be used to manage events and images, create a key pair, connect volumes to events, manage Swift containers, and so forth. The dashboard even gives the user access to the event console and can connect to events via VNC. As you can see in the class diagram (figure 3), OpenStack dashboard includes only Server1.

Client1 includes four Swift software packages, Pxthon-novaclient, Glance-client, and Qemu-Kvm. Swift was reviewed above. Qemu-Kvm is a hardware emulation and virtualization software. It allows OS and software designed for one hardware architecture (for example, ARM) to run on another (for example, PC) as a separate process. And Pxthon-novaclient and Glance-client are the client components of the Nova and Glance modules, respectively.

The UML diagram shown in figure 3 illustrates the structure of the cloud system. It shows the interconnections of all system components, which allows you to determine the installation and configuration of the cloud modules.

4. Testing the model of organizational and technological processes of machining

OpenStack is a set of open source technology projects, so the software packages of all its modules are available in standard Ubuntu repositories. Additional software packages that have been used are also available in these repositories.

All modules are configured using the appropriate commands and files described in the original documentation for the software packages.

The installation sequence of all components is presented in the form of a diagram in figure 4.

4.1. Install additional packages

Before installing OpenStack modules, you need to verify that support for network bridges, the MySQL database, and the NTP time server are installed. This ensures that machines and network nodes are synchronized.

To provide bridge support for Server1 and Server2 from the standard Ubuntu OS repositories, the bridge-utils package must be installed. After that, you can configure the network settings and restart the network.

The time for all OpenStack components must be synchronized. To do this, you must install the NTP package on all computers. In this case, Server1 is configured as an NTP server. If the Internet connection drops, the NTP server uses its own hardware clock as a backup.

Next, install the mysql-server and python-mysqldb packages and create MySQL databases for use with nova, glance and keystone. We also create users for them and provide privileges for the created databases.

4.2. Installing and configuring the Keystone Module

The first OpenStack module to install is Keystone for Server1. It is an authentication service used by OpenStack. For its operation, the packages Keystone, python-keystone and python-keystonclient should be installed. Python-keystone and python-keystonclient are a library of components and API client for the OpenStack authentication system developed in Python. Next, in the configuration files, you should specify the MySQL database and restart Keystone.

Keystone setup consists of creating admin and service tenants, admin, nova, glance and swift users, as well as admin and member roles. After that, we add roles for users that have been created and create the corresponding services through which users can authenticate. Keystone setup completes by creating endpoints for each of the services that were created.

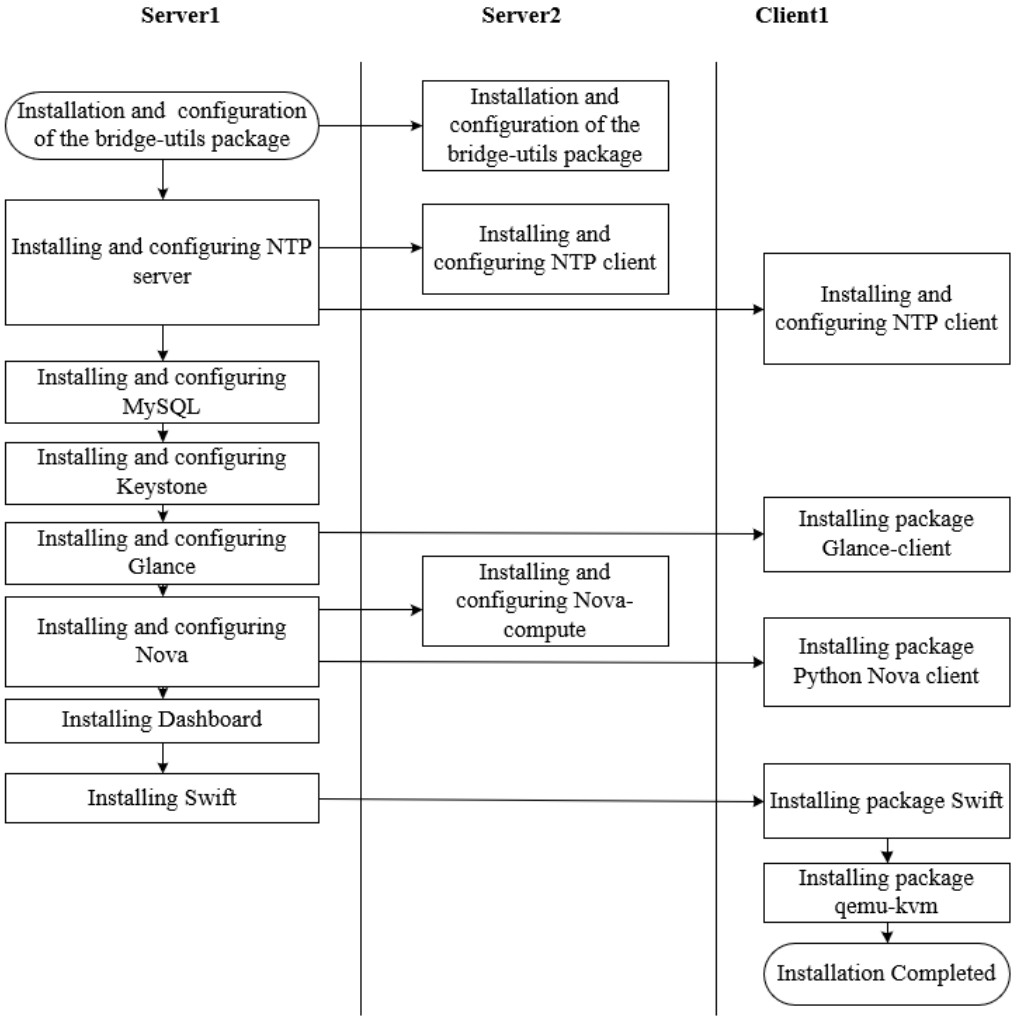

Figure 4. OpenStack Component Installation Sequence Diagram.

4.3. Installing and configuring the Glance module

The Glance module consists of the following software packages: Glance, Glance-API, Glance-client, Glance-common, Glance-registry and Python-Glance. Glance installs on Server1. By default, Glance uses the SQLite database. To work with MySQL, you must specify it in the configuration files, moreover, it is necessary to specify Keystone for authentication in the settings.

On Client1, you need to install the Glance-client software package. This package does not require special settings.

4.4. Installing and configuring the Nova module

The Nova module includes the largest number of software packages: Nova- API, Nova-cert, Nova-compute, Nova-compute-kvm, nova-doc, nova-network, nova-objectstore, nova-scheduler, nova-volume, rabbitmq- server, novnc, nova-consoleauth. On Server1, you need to install all these packages, and on Server2 only nova-compute. Also on Client1, you need to install the client part of

Nova represented by the python-novaclient package. Nova configuration files specify all network parameters. On Server1, you need to create a new physical volume and add it to the nova-volumes volume group. Using the nova-manage package, specify the range of IP addresses and restart the Nova services.

4.5. Installing the openstack-dashboard package

For the Horizon module to work, just install the Openstack-dashboard software package and restart the apache2 web server.

4.6. Installing and configuring the Swift module

The Swift module consists of the following software packages: swift, swift-proxy, swift-account, swift-container, swift-object. The complete Swift module is installed only on Server1, and on Client1 only the swift package is installed. Configuration is required only for Server1.

A file filled with zeros should be created for use as a loop device for the Swift backend repository. In Ubuntu, working with disk images is done through so-called loop devices. The image is attached to the loop-device, after which the system can work with this device, as with conventional block devices. You must use the copy disk command to create a file called swift-disk and the allocated 1KiB million blocks (976.56 MB) to it. A mount point is created for this file. Next, before mounting the backend, you need to create several nodes that are used as storage devices, set the owner of the user 'swift' and the group.

Next, you need to configure the Rsync component. It is responsible for maintaining the replica object; it is used in various swift services to ensure object consistency and update operations. The Rsync component is used for all storage nodes. You also need to configure a proxy server. He takes responsibility for user authentication. During authentication, it is verified that the request actually came from the one who sent it. During authorization, it checks who has access to the resources that are requested. Authorization is carried out by identical services like Keystone.

This does not end the Swift setup. An important component of Swift is Ring. It contains information about the physical location of objects, their replicas, and devices. You must create the builder ring for the appropriate object service, container service, and account service. After that, zones and balance rings are added.

4.7. Finishing installation of OpenStack

After setting up Swift, you must install the qemu-kvm software package for Client1. This package does not require special settings.

If all the settings were done correctly, Nova and Glance commands will be executed on Client1.

5. Conclusion

The proposed information technology for deploying the cloud environment OpenStack can be used in a research laboratory to implement and adapt OpenStack with support for basic services for deploying VMs with an internal network and a set of external addresses.

In the future, it is planned to develop an information technology for scaling the OpenStack environment to a larger number of workplaces in order to deploy additional software services used in the research laboratory.

References

[1] Skatkov A, Shevchenko V, Brykhovetsky A, Voronin D, Mashenko E and Chengar O 2018 *System Modeling Actor Integrarions for Cloud Services ed Arial* vol 1 (Simseropol) pp 15-23
[2] Hogan M, Liu F, Sokol A and Tong J 2011 NIST Cloud Computing Standards Roadmap. *Computer Security Division Information Technology* (Gaithersburg) p 76
[3] Aleksiyants A, Borisenko O, Turdakov D, Sher A and Kuznetsov S 2015 *Proc. of ISP RAS* **27** 35-48

[4] Skatkov A and Shevchenko V 2015 Extension of the reference model of the cloud computing environment in the concept of large-scale scientific research *Proc. of ISP RAS* **6** 285-306
[5] Feoktistov A, Sidorov I, Sergeev V, Kostromin R and Bogdanova V 2017 Virtualization of heterogeneous HPC-clusters based on OpenStack platform *Bulletin of the South Ural State University* **6** 37-48
[6] Duan, Zhi Ying, and Yi Zhen Cao 2014 The Implementation of Cloud Storage System Based on OpenStack Swift *Applied Mechanics and Materials* **644** 2981-4