PROJEKT

Multitenantné operačné centrum kybernetickej bezpečnosti riešené ako otvorená cloudová služba s prvkami strojového učenia

Previazané s balíkom KPB6 – Publikačné výstupy

Financované Európskou úniou
NextGenerationEU

PLÁN OBNOVY

ÚRAD VLÁDY
SLOVENSKEJ REPUBLIKY

OpenStack – Architecture overview and comparison of deployment options

Marek Moravcik
Faculty of Management Science and Informatics
University of Zilina
Zilina, Slovakia
marek.moravcik@fri.uniza.sk

Ivana Bridova
Faculty of Management Science and Informatics
University of Zilina
Zilina, Slovakia
ivana.bridova@fri.uniza.sk

Martin Kontsek
Faculty of Management Science and Informatics
University of Zilina
Zilina, Slovakia
martin.kontsek@fri.uniza.sk

Pavel Segec
Faculty of Management Science and Informatics
University of Zilina
Zilina, Slovakia
pavel.segec@fri.uniza.sk

***Abstract*—The paper deals with the OpenStack cloud platform used in the Department of Information Networks.**

OpenStack is an open-source solution for private, public, and hybrid clouds, offering a lot of flexibility and scalability in managing compute resources. The paper also describes the main parts of the OpenStack platform architecture. It also focuses on the analysis of various tools for automating the deployment of OpenStack, such as OpenStack-Ansible, Kolla-Ansible, and OpenStack Charms, which simplify its configuration process and subsequent management in the software layer. Special attention is given to the integration of MAAS and Juju tools, which enable effective management of physical infrastructure.

The results of this work provide an overview of the possibilities for implementing OpenStack in an academic environment and emphasize the importance of automating these steps.

***Index Terms*—Cloud Computing, OpenStack, MAAS, Juju**

I. Introduction

OpenStack is a popular platform for deploying of private clouds.As an open-source cloud computing standard it supports variety of compute resources such as containers or virtual machines. This makes OpenStack well-suited for academic environments. At our faculty, OpenStack serves as the core platform for managing and delivering cloud-based infrastructure. It provides the flexibility and scalability required to meet the diverse needs of our projects and different research initiatives.

The essence of OpenStack lies in its modular design. For its run it needs only a few basic modules, while additional functionality can be achieved by adding another components based on the administrators needs. These components, often referred to as services or modules, interact through well-defined APIs to deliver specific functionalities, such as virtual machine orchestration, network management, or storage provisioning.

OpenStack provides a vast numbers of modules, but only three basic ones are needed to run this platform. Those modules are Nova (compute), Neutron (networking), and Ceilometer (monitoring and metering). These core modules work together to provide essential cloud computing functions. Additional modules in the departments's OpenStack setup include for example Cinder (block storage), Keystone (identity service), Horizon (dashboard), and Heat (orchestration module), among others. In the next chapter, we will provide a more information about these modules.

II. Openstack modules

A. Nova

Nova is the primary compute engine for OpenStack, responsible for managing virtual machines in the cloud. It provides the essential components for creating, deploying, and managing virtual instances. The Nova service is highly flexible and can be deployed on various hardware platforms, enabling broad compatibility with different virtualization technologies. Within our department's cloud environment, we primarily use it for creating instances that simulate network topologies for educational purposes.[1, 2]

The process of creating such an instance is relatively simple. First, we need to select an image (representing a real operating system), then create a network to which the instance will be assigned. Next, we add a security group to protect it from unauthorized manipulation, choose the appropriate resources like RAM or memory, and finally, we can launch the instance.

B. Swift

Swift is a module used for storing and managing objects in the cloud. These objects represent unstructured data, such as documents, images, videos, or various backups. It is optimized

for working with large volumes of such data, while also providing high scalability and replicating data across multiple nodes in a cluster. This replication ensures fault tolerance. The system is built on an API compatible with OpenStack, making it easier to integrate with other modules, especially for the needs of the Nova module.[3]

C. Glance

It is a module in OpenStack responsible for providing virtual machine images. It is especially important for the Nova module, as it is used when creating instances that utilize stored images. The module supports various image formats, such as VMDK and VDI, and can use different backends for storage, including Swift or Ceph. In addition to managing the images themselves, it also allows for storing metadata and categorizing images, making searching for them easier.[4]

D. Keystone

Keystone is the identity service for OpenStack, providing authentication and authorization. It is responsible for managing user accounts and roles, ensuring that only authorized users and services can access the cloud resources or projects. Keystone supports different types of authentication, offering specific control over permissions and resource access.[2]

E. Horizon

Horizon is a web-based graphical user interface (GUI) for OpenStack platform. It uses a REST-based API front end to enable easy communication with these services through a web browser. For users, Horizon serves as a platform to manage resources, including created topologies and components such as instances, networks, and more.[5]

F. Neutron

Neutron serves as the networking component of OpenStack, providing flexibility and network management in a cloud environment. It allows users to efficiently configure and manage network elements through an API interface and a set of agents responsible for managing the software-defined networking (SDN) infrastructure. Neutron enables the creation of various types of networks, which is very useful in an academic environment, where students work on creating network topologies and gain practical experience with them. This OpenStack component allows for flexible customization of network settings according to specific needs, which is essential when building complex network architectures. Neutron also provides an interface for integration with other OpenStack services, such as Nova, Cinder, and others, enabling the creation of fully integrated solutions within the cloud.[1]

G. Cinder

Cinder is the block storage service for OpenStack, offering volumes to Nova virtual machines or containers through a REST API. It also allows users to manage snapshots and backups, enabling them to easily restore data or create new volumes from snapshots. With Cinder, users can scale storage dynamically based on their needs or configure different storage policies. All these actions can be easily performed via the Horizon user interface.[6]

H. Heat

Heat is the module responsible for orchestration, which enables their users to automate deployment of cloud applications by defining their infrastructure needs as code. It implements an orchestration engine responsible for creating applications based on two different templates, Heat Orchestration Template (HOT) or AWS CloudFormation (CFN). Those templates describe the relationships between resources like instances, networks with their volumes or security groups.[5]

Heat makes easy way how to setup even complex topologies or applications without a lot of manual work. This module also manages the whole lifecycle of the application e.g. when you need to change it for some reasons, you just need to simply modify the template and paste it in OpenStack. Heat then on his own compare the new template with the previous one and apply all founded changes. Even though Heat primarily manages infrastructure it also integrates with software configuration.[5]

I. Ceilometer

Ceilometer is the telemetry service within OpenStack. Its main strength lies in capability to collect metering data from various OpenStack services, such as Nova, Neutron or Cinder and providing a unified view of resource consumption across the cloud. The whole concept of Ceilometer is built on collecting and processing data samples. All those samples are being recorded on regular basis and together creating stastistics. Those statistics then enabling administrators to gain insights into resource usage and system performance. Ceilometer allows us also to use samples as alarms. Those alarms watch for a certain criterion to be met and then perform specific action.[7, 1]

III. Deployment options

OpenStack, as a cloud computing platform, offers several deployment models, each fullfilling different organizational needs and operational strategies. The primary models include the public cloud, private cloud, and hybrid cloud.

A. Public cloud

The public cloud model provides resources over the internet in form of already existing Openstack deployment, that is managed by the providing party. This is allowing organizations to scale flexibly without heavy investments in physical infrastructure and administrators, which can be difficult to find and train - as Openstack isn't as widely spread as many of its public cloud alternatives. While this is advantageous for many, it may not suit organizations with strict compliance or security requirements. [8]

B. Private cloud

Conversely, the private cloud model offers dedicated resources and greater control, appealing to enterprises that prioritize data security and customization. This option allows organizations to tailor their environment according to specific operational demands while maintaining full oversight of their infrastructure. This is also the approach that was chosen on the Department of Information Networks on Faculty of Management Science and Informatics, where such solution was deployed - mainly because of the already available hardware and many opportunities it opens for education of students and future projects. However, it requires more upfront investment and ongoing management, that can get difficult in terms of man-hours required for such maintenance. [8, 9]

C. Hybrid cloud

The hybrid cloud model combines elements of both public and private clouds, enabling organizations to balance workloads between on-premise facilities and off-site services. This flexibility is particularly beneficial for oragnizations aiming to optimize costs while managing fluctuating workloads or sensitive data - though it should be dully noted that hybrid cloud bring the extra complexity to an already wide set of systems that form together such cloud infrastructure. As organizations navigate these models, they must consider factors such as cost, scalability, and compliance to determine which deployment strategy aligns best with their goals and resources. [8, 9]

IV. Overview and Comparison of most prominent Deployment Tools and Frameworks

After deciding on the type of deployment, there comes an entirely different problem - choosing the right deployment tool, as the on-premise deployment of OpenStack can be greatly facilitated by various tools and frameworks, each offering unique advantages and disadvantages:

- openstack-ansible
- kolla-ansible
- openstack-charms

For instance, tools like Ansible automate the installation and configuration processes, allowing for quicker deployment and consistency across environments. This tool is used as such in ‘openstack-ansible‘ and ‘kolla-ansible‘ projects, each aiming to install and manage Openstack in various ways - These automation tools reduce human error and streamline the management of numerous servers, which is crucial in complex cloud infrastructures. However, they require a certain level of expertise to set up and maintain, posing a challenge for organizations without dedicated resources. While both openstack-ansible and kolla-ansible leverage Ansible for deployment and management of Openstack, the main distiction between the two is their approach to underlying runtime environment for Openstack components - openstack-ansible is leveraging power of Linux Containers (LXC). In contrast, kolla-ansible uses Docker containers, emphasizing much more popular approach, making it simpler for a lot of admins already familiar with the technology. [10, 11, 12, 13]

Entirely another deployment stack/tool worth mentioning is OpenStack Charms, which leverages multiple tools managed by Canonical Ltd. - those mainly being Juju and MAAS (Metal as a Service). These tools work together to simplify management of servers (including installation of operating system) and deployment of applications onto them in the form of packages, entirely getting rid of most of the manual and tedious processes. In comparison to Ansible equipped tools like openstack-ansible and kolla-ansible, OpenStack Charms follow more model-driven approach, where everything is abstracted behind high-level objects - making the entire system more approachable even to less experienced admins. On the other hand, system like this can and has its apparent caveats - while the tools offer a vast amount of configuration options, it can’t provide such granular control as Ansible equipped tools do - making it more difficult to customize Openstack Charms in edge case scenarios. [12, 14]

V. MAAS

MAAS (Metal as a Service) is a tool developed by Canonical Ltd. for managing the physical infrastructure of an on-premise OpenStack deployment. It allows administrators to manage a large fleet of physical servers as if they were virtual machines. The central function of MAAS in OpenStack deployment at our department is to handle the provisioning of bare-metal machines by turning them into a pool of available resources. It achieves this by automating processes such as power management, PXE booting, operating system installation, and network configuration - all this is achieved through machine-specific out-of-band management interface based on IPMI (Intelligent Platform Management Interface). Through its intuitive web interface, MAAS enables administrators to add, commission, and deploy machines seamlessly. Commissioning a machine also involves testing its hardware (such as CPU, RAM, and disk) and verifying network connectivity before it becomes available for deployment. In the context of OpenStack, MAAS makes the initialization of the hardware layer setup easier, freeing administrators from manually commissioning all the hardware manually. [15, 16]

To describe how the usual workflow looks like, the initialization and commissioning process in MAAS begins when a physical machine is added to the system. When commissioning the machine, the machine is powered on through IPMI, instructed to boot from PXE (Preboot Execution Environment), where it loads the provided (in this case Ubuntu) Linux distribution into its RAM. MAAS then uses this ephemeral operating system to scan the machine and determine its hardware: i.e. CPUs, RAM, storage drives, PCI and USB devices, and so forth. As next, MAAS carries out some basic test, just to make sure that all the hardware works just as expected. [16]

Once the hardware is validated, MAAS marks the machine as ready for deployment and user is allowed to configure the desired machine status - basic configuration, mainly involving networking setup. After that, user can initiate the deployment

itself, where the destination system will be loaded and installed with the desired operating system, network configuration and other such configuration user provided. [16]

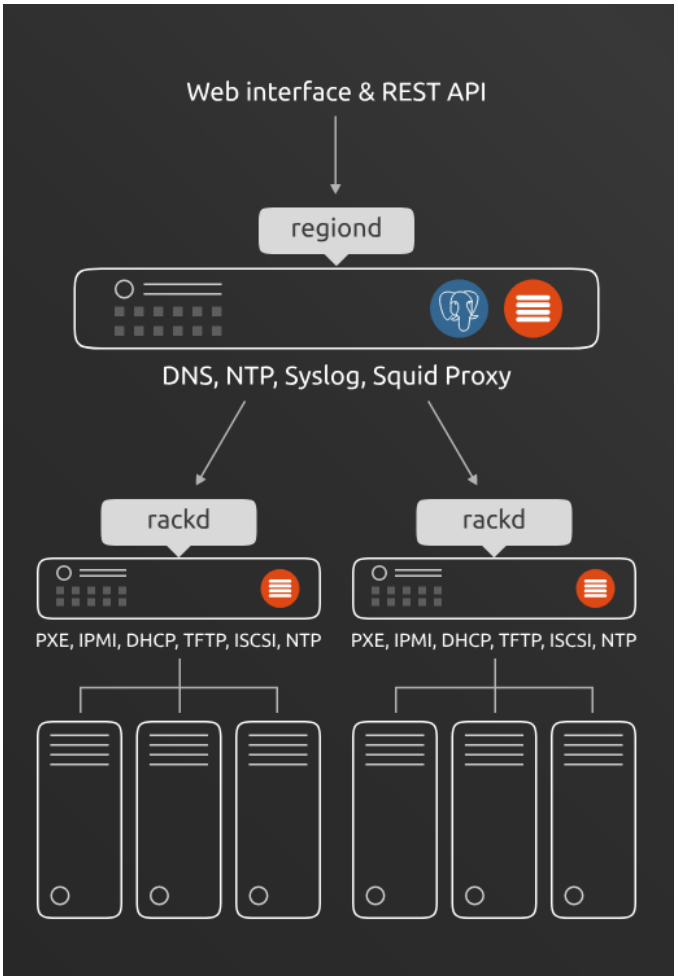

Fig. 1. MAAS Architecture [17]

VI. Juju

Juju is a tool that works with MAAS to manage the lifecycle of OpenStack components. It is a service orchestration and management tool that automates the deployment, scaling, and operation of software services, which we utilize in our OpenStack on-premise deployment. The main concept of Juju is to package different application deployments into ”charms,” which are reusable bundles of application files accompanied with scripts for installation, upgrades and other such tasks, overall encapsulating best practices for deploying and managing specific software application. In the case of OpenStack, Juju provides charms for each of the major components — such as Nova for compute, Neutron for networking, Cinder for block storage, or Keystone for identity management. When combined with MAAS, Juju allows these services to be deployed across a collection of bare-metal machines - which Juju controls through MAAS itself - completely automatically. This automation reduces the complexity of deploying OpenStack, which usually involves a more complex installation process. [18, 19]

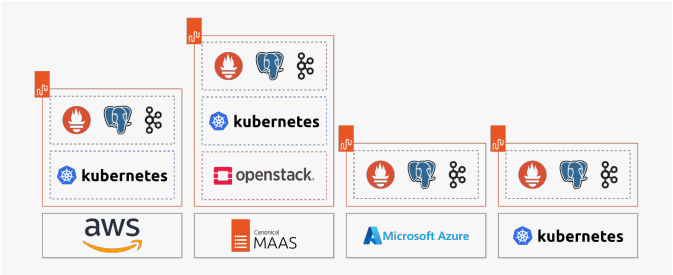

Fig. 2. Juju Architecture [17]

Juju also offers the ability to create and manage relationships between these OpenStack components dynamically. For example, when deploying an OpenStack cloud, Juju can link Nova (compute) to Neutron (networking), ensuring that these services are correctly configured to communicate with each other - all this is accomplished through integrated logic in the Juju controller. That means, when you link charms together, Juju ensures that the required endpoints, credentials and similar settings are exchanged and applied correctly on both sides. [19]

This modular approach juju is utilizing also supports horizontal scaling - as demand increases and currently deployed services are over-saturated, additional compute or storage nodes can be provisioned with MAAS, and Juju will automatically deploy and configure the necessary services on these new nodes. This design also enables Juju to simplify upgrade and maintenance processes. It simplifies the upgrade by allowing services to be upgraded without significant downtime, maintaining high availability. Juju also provides monitoring and logging for all the deployed services, allowing administrators to track application status and health in real time from a single place with just one utility. [20] To provide an useful example, Juju’s status output for deployed services also allowed us to write our own alerting service to send notification to our administrators in case of any application failure thanks to the Juju’s ability to parse all the status output in JSON format, which allows its users to parse any required information.

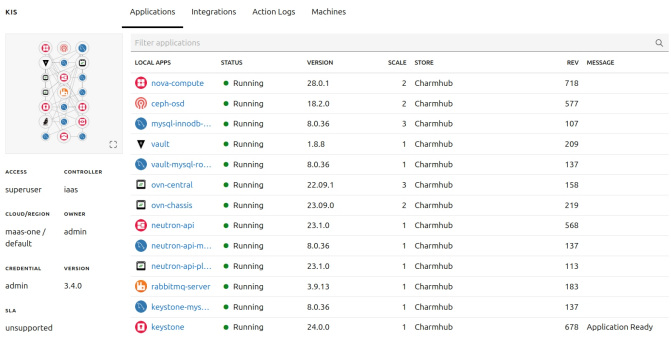

Fig. 3. Juju Dashboard [17]

By combining Juju and MAAS, organizations can create a fully automated and scalable infrastructure for deploying OpenStack on bare-metal hardware. This combination provides the benefits of public clouds on your own hardware, giving businesses and institutions full control over their infrastructure while maintaining the flexibility to scale and evolve their environments over time.

VII. Conclusion

In this paper, we presented OpenStack used in the Department of Informatics Netowrk at our faculty, where we focused on describing the architecture and deployed modules that enable flexible and scalable cloud service deployment. We also compared various deployment models, such as public, private, and hybrid clouds, highlighting the variety of options OpenStack offers depending on the specific needs of an institution. Furthermore, we focused on the most suitable tools and frameworks for automating OpenStack deployment. We introduced tools like OpenStack-ansible, Kolla-ansible, and OpenStack Charms, which simplify and speed up the installation and management of complex cloud environments. In addition to managing the software aspect of the cloud, we also addressed the management of its physical part, where we discussed tools like MAAS and Juju, which play a key role in managing our servers.

From an academic perspective, OpenStack is a valuable tool that allows students to work with real cloud technologies and create various topologies using Nova instances, where they can gain further practical experience in configuring and managing them.

Acknowledgment

References

[1] Dan Radez. *Openstack essentials*. Packt Publishing Ltd, 2015.

[2] Huawei Technologies Co., Ltd. “OpenStack”. In: *Cloud Computing Technology*. Springer, 2022, pp. 123–145. ISBN: 978-981-19-3026-3. URL: https://link.springer.com/chapter/10.1007/978-981-19-3026-3_6.

[3] Joe Arnold. *Openstack swift: Using, administering, and developing for swift object storage*. ” O’Reilly Media, Inc.”, 2014.

[4] Ken Pepple. *Deploying openstack*. ” O’Reilly Media, Inc.”, 2011.

[5] Ashish Lingayat et al. “Horizon, a web-based user interface for managing services in openstack: an introspection”. In: *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE. 2018, pp. 1–6.

[6] OpenStack Community. *OpenStack Block Storage (Cinder) Documentation*. Last updated: 2021-09-24 17:25:46. OpenStack Foundation, 2021. URL: https://docs.openstack.org/cinder/latest/.

[7] Dongmyoung Baek and Bumchul Lee. “Analysis of telemetering service in OpenStack”. In: *2015 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE. 2015, pp. 272–274.

[8] Sumit Goyal. “Public vs private vs hybrid vs community-cloud computing: a critical review”. In: *International Journal of Computer Network and Information Security* 6.3 (2014), pp. 20–29.

[9] Mitesh Soni. “Is the private cloud a real cloud?” In: *Linux Journal* 2014.243 (2014), p. 4.

[10] Maciej Siczek) OpenInfra Foundation (Maciej Kucia. *Deploying OpenStack - what options do we have?* Youtube. 2019. URL: https://youtu.be/8ODdvCogwl8?t=473 (visited on 01/11/2025).

[11] *About OpenStack-Ansible - Openstack*. Version 2024.2. URL: https://docs.openstack.org/project-deploy-guide/openstack-ansible/2024.2/app-aboutosa.html (visited on 01/11/2025).

[12] Sai Vivek Gudipati and Vishwa Mithra Tatta. *Investigation of an automatic deployment transformation method for OpenStack*. 2022.

[13] Omar Khedher and Chandan Dutta Chowdhury. *Mastering OpenStack*. Packt Publishing Ltd, 2017.

[14] Anshu Awasthi and P Ravi Gupta. “Comparison of openstack installers”. In: *International Journal of Innovative Science, Engineering & Technology* 2.9 (2015).

[15] *About MAAS - Canonical*. URL: https://maas.io/docs/about-maas (visited on 01/11/2025).

[16] *How it works, MAAS - Canonical*. URL: https://maas.io/how-it-works (visited on 01/11/2025).

[17] *StarlingX*: a fully-featured cloud for the distributed edge — 01.org*. URL: https://01.org/blogs/forrest/2019/starlingx-fully-featured-cloud-distributed-edge.

[18] *What is a charm? (Juju) - Canonical*. URL: https://juju.is/charms-architecture (visited on 01/11/2025).

[19] Dilma Morais and João Paulo Sousa. “Orchestration of Cloud-Based Services and Infrastructure: An Exploratory Analysis of Juju, Kubernetes, and Terraform”. In: *37th International Business Information Management Association (IBIMA)* (2021), pp. 1–8.

[20] *Relation (integration) (Juju) - Canonical*. URL: https://juju.is/docs/juju/relation (visited on 01/11/2025).