International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 04, April-2022

A Review of OpenStack Cloud Projects

Koffka Khan
Department of Computing and Information Technology
The University of the West Indies,
Trinidad and Tobago, W.I

***Abstract*— OpenStack is a collection of software modules called projects that work together to create and manage cloud infrastructures. OpenStack delivers infrastructure as a service functionality. It pools provisions and manages compute, storage and network resources. It's one of the many open source technologies that help IT teams create and manage cloud workloads and is often seen as an alternative to cloud platforms like AWS, and Azure. In order to run multiple operating systems and applications, organizations often turn to virtualization, which abstracts computing resources from physical hardware such as servers. Installing OpenStack on top of the virtualized environment forms a cloud operating system and broadens this pool of resources to support many uses from web and application hosting to big data tasks. With OpenStack, software components called projects are selected to build out the features of an enterprise's cloud setups vary, but typically start with a handful of central components, like a compute Virtual Machine (VM) image, networking, storage, identity management and resource management. It business can add other components to further build up this infrastructure to fit its growing needs. OpenStack boasts many benefits for businesses like affordability as it's freely available under the Apache 2.0 license. Reliability, with almost a decade of development in use. In vendor neutrality, as its open source nature makes it attractive to businesses trying to avoid vendor lock in, but adopters also must consider some drawbacks including complexity, requiring IT staff with significant knowledge of the platform support as it's not owned by any one vendor or team and relies on the open source, community and consistency. The OpenStack components suite is always in flux as components are added and depreciated. OpenStack adoption is a process where organizations looking to build a private cloud based on OpenStack need time financial investment and support from upper management. (*Abstract*)**

Keywords*— *OpenStack; software; projects; cloud; workloads, big data, virtual; machine; reliability

I. INTRODUCTION

In OpenStack [25], there are many projects, each of them with a different adoption status. And if you want to learn OpenStack, it is important to become familiar with the different products that are around. You should also realize that at the OpenStack Summit, that happens twice a year, decisions are made on new projects. So, new projects are being added on a regular basis. I want to show you the project navigator website. On this website, you can get an overview of all the current projects that currently are existing, including their current status. So, here's the project navigator website. So, you can see that the OpenStack Foundation distinguishes between core services and optional services.

The core services are Nova [9], Neutron [10], Swift [2], Cinder [21], Keystone [7], and Glance [14]. I will explain in a while what these services are all about. The interesting thing here is that you can see the current adoption. So, Nova appears to be a very essential part of OpenStack, because 93% of all OpenStack clouds are currently using it. And it appears to be very mature, because it has a maturity score of eight out of eight. And it has been around for a long time - six years, as you can see here ... which is the case for Swift, as well, and which is the case for Glance, as well. Now, you can also see that there are optional services. And some optional services are doing quite well, like Horizon, for example, that has been around for five years and is adopted for 86% of all OpenStack deployments. But there's also new services, like Project Magnum and Project Congress. Congress is an interesting one; it is being used by 1% of all OpenStack clouds only.

As you can see, this is how the OpenStack Foundation decides upon the majority score. And, as you can see, the only majority indicator is that this project team has achieved corporate diversity and everything else is still being worked upon. That doesn't necessarily mean that it's a bad project, it's just a project that has just been started. So, if you want to know before starting to use a specific project in OpenStack, if you want to know about its current status, this is where you can find it.

Now, let me give you an overview of the core OpenStack projects. So, currently there are six core OpenStack projects, as you have been able to see in the previous lesson. To start with, there is Nova Compute. Nova Compute is the interface to the hypervisor. It makes sure that virtual machines can be running somewhere in the cloud. Then, there is Neutron Networking. Neutron Networking is providing Software Defined Networking to the cloud. Then, we have Swift Object Storage. This is a smart way of using storage in a cloud, storage in a way that it is not bound to physical devices, but it is organized in binary objects that can be scattered all over the cloud in a distributed and replicated way. Then, there is Cinder Block Storage. Cinder Block Storage is what you can use as an administrator to provide persistent storage to the virtual machine that you are deploying in the cloud. We also have Keystone Identity. Keystone Identity is the glue that is matching everything. In Keystone Identity you will create users, roles, and tenants, and services and Keystone decides which user has access to which specific service and how specific services can communicate to one another. And there is Glance Image. Glance Image is the image service and you need it, because you don't want to install an instance in the cloud, you want to deploy it. And in order to deploy it, that's the Glance Image service.

This paper consists of eleven Sections. In Sections II to VIII we present details of individual OpenStack projects. They

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 04, April-2022

are NOVA (Section II), NEUTRON (Section III), SWIFT (Section IV), GLANCE (Section V), CINDER (Section VI), KEYSTONE (Section VII) and HORIZON (Section VIII). In Section IX we give a brief description of other important OpenStack projects. In Section X we give a discussion of OpenStack services. Finally, in Section XI the conclusion is given.

II. NOVA

Nova is maybe the most important core project in OpenStack. It is responsible for managing compute instance lifecycle. Now, what is a compute instance? Compute is a different name for Nova, and an instance is a virtual machine. And Nova takes care of running the virtual machine. Some people think that Nova is the hypervisor. That is not the case. Nova interfaces to the hypervisor. It is not the hypervisor itself. And that makes it very interesting. No matter which hypervisor you are running, it may be Xen, or KVM, or VMware, or vSphere, or you imagine, and it can be dealt with by Nova [13].

Nova will install an agent on the hypervisor to make sure that it's supported in your OpenStack environment. Nova itself is responsible for spawning, scheduling, and decommissioning of virtual machines on demand. And it includes Nova service processes that are running on the cloud controller, as well as Nova agents, that are running on the hypervisor. And you will see more frequently, Nova is using a distributed architecture, where some agents are addressed that are running there where work needs to be done, and some service processes are running on the centralized cloud controller. Figure 1 shows the icon used to represent OpenStack NOVA.

Figure 1: Icon for OpenStack NOVA

III. NEUTRON

Neutron enables Software Defined Networking. Now, let's talk about Software Defined Networking - what is it all about? Well, it allows users to define their own networking between the instances that are deployed. Let me make a small drawing to explain what this is about. So, imagine your typical OpenStack environment. Let's say that in this environment there are two different compute nodes. So, that will be c for compute node and there is another c for compute node. These compute nodes will be connected by using a physical network [28]. We call that the underlay network. And this physical network may involve routing and stuff. And that is typically what the user of OpenStack won't be aware of. So, at the usage level, that will be the higher level, that is what we call the Overlay Network.

At the user level, there may be an instance running here, and there may be another instance running here. But the user may want to deploy these instances as being used in the same broadcast domain. So, this might be a Logical Network, right? In a broadcast domain, the instances will be in the same network. Now, there's an issue here. How can this be the same broadcast domain if, in the underlay network, we have different physical networks? And that is exactly what Neutron is taking care of by using Software Defined Networking.

In order to do this, Neutron needs to interface the physical network architecture. And to do that, Neutron is using a pluggable architecture. And this pluggable architecture supports many networking vendors and technologies. So, if you have made a large investment in some proprietary networking technology, chances are that there's a good Neutron plugin available. Most network vendors do have Neutron plugins, so you won't have an issue there. And also, Neutron provides an API for users to define networks and the attachments into them. And that means that, for a programmer, as well as an administrator, it is relatively easy to create their own Software Defined Networking. Figure 2 shows the icon used to represent OpenStack NEUTRON.

Figure 2: Icon for OpenStack NEUTRON

IV. SWIFT

The next core OpenStack project that we'll be talking about is Swift. Swift is designed to provide scalability at the storage level. Now, how does it do that? It works with binary objects to store data in a distributed, replicated way. Let me explain why we need this. Storage ultimately ends on a hard drive. And a hard drive is a physical device, right? And the problem with physical devices is that they are limited and they are not very scalable. So, if all your cloud storage will be on a server that is providing hard drives, that isn't very scalable. Now, Swift is doing that by providing the object-based storage model. Let me sketch what this looks like. So, the idea about Swift is that we have an application. And this application normally, in order to write data, would write a file. Now, in an OpenStack environment, the application doesn't write a file to hard drive.

In an OpenStack environment, an application can interface with Swift object storage. And the Swift object storage is talking to many, many storage nodes. Let me just draw three items that represent storage nodes. You know what? Let's draw four of them. Now Swift is using a proxy. And when the Swift proxy received data from the application, it is going to create binary objects. And these binary objects are the chunks of data. So, let's say we have binary objects A, B, and C. Let's

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 04, April-2022

make it very small. In Swift, binary objects A may be stored here, and binary object B may be stored here, and binary object C may be stored here. But if it's just stored once, that is not very fault tolerant, right? And that is why Swift includes a replication algorithm - to store the binary objects on multiple servers. By default, that will be three times, there's nothing against storing the binary objects four times, if that is needed.

So, how does this make your storage more efficient? Well, it makes storage more efficient because, at the moment that the application needs to retrieve the data, it will address the Swift proxy again. And the Swift proxy is using an advanced algorithm to determine exactly where the binary objects resides. And it will send calls to all the storage nodes that are involved. And because of all these storage nodes that will be able to work in parallel, the data will be arriving at the Swift proxy, and hence, at the application, in a very fast way. And this is how you make storage scalable in OpenStack [1]. Because, imagine that these storage nodes are one terabyte each. When you are running out of available storage, well, then it is pretty easy to add a couple of more Swift storage nodes, and you can even rebalance the binary objects that are written in the Swift storage configuration. There's one thing about Swift, and that is the application.

For the application, in order to talk to Swift, it is using a Restful API. And REST is a standard way of communicating in an OpenStack environment. And that means that the application is not writing a file to a filesystem, it is using a RESTful API call, which is understood by Swift proxy [18]. RESTful API is the native language of OpenStack, and that makes Swift the native choice for object storage in OpenStack. Let's get back to the slide, where we can see that Swift is using access via the RESTful API, and it is designed to be very fault tolerant. Figure 3 shows the icon used to represent OpenStack SWIFT.

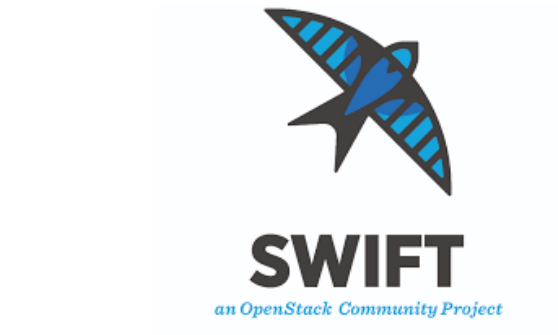

Figure 3: Icon for OpenStack SWIFT

V. GLANCE

The next important OpenStack project is Glance. Glance is used to store virtual machine disk images. So what is it about virtual machine disk images? Well, virtual machines, which are the instances, are not installed, they are spawned off from an image. It's like booting Linux from a live CD. And images can be easily downloaded or they can be created to match specific needs within an organization. This is a link where you can find a list of ready-to-download images. Let's have a look at it. So, at this web page, you can see that different cloud images are provided for different operating systems, that include CentOS, and Debian, and Fedora, so basically, all the major Linux distributions, and it even includes Microsoft Windows.

An interesting cloud image is the CirrOS (test) image. And this one is interesting because it is very small. Download of the CirrOS (test) image is only like thirteen megabytes, and this is a minimal Linux operating system. But as you can see, you can download other instances as well, such as Windows cloud-based images, which is a little bit specific, by the way, and not as easy to deploy in OpenStack, as is the case for the Linux-based cloud images, because they are running a free and open operating system [3].

Glance is the image store. So, that means that, if an administrator wants to boot an instance, he will boot the instance from the Glance image store. And that is why Glance is a very important part of OpenStack. And to make it all scalable, the Glance image store typically is using Swift or Ceph object storage [6] as a backend. Of course, you don't have to use Swift or Chef object storage as a backend, but it makes it more scalable. In a small all-in-one deployment, you are more than welcome to use local storage as a backend to Glance, but, as you can imagine, if you are using local storage, then everything you do is bound to the physical server that contains the physical images. And that's not very scalable. Figure 4 shows the icon used to represent OpenStack GLANCE.

Figure 4: Icon for OpenStack GLANCE

VI. CINDER

The next essential OpenStack project is Cinder. It provides persistent storage to instances. Now, what is the idea? Well, the idea is that instant storage, by default, is ephemeral. That is also like booting from a Linux live CD. If you are working from a Linux live CD, if you ever try to change the configuration file, where would you change it to? Because the live CD image is loaded to RAM and it doesn't really exist on any writable hard disk. And that's the same problem with Glance images, and that is why, in OpenStack, instant storage is ephemeral.

Cinder is what allows the administrators to attach additional persistent block devices to instances [20]. So, if you want to make sure that it will be saved, then you will be using Cinder. Cinder can use different backends, as well. It can be local storage, which, by default, will be local Linux LVM [Logical Volume Manager], or it can include Swift and Ceph object storage as well for increased scalability. Figure 5 shows the icon used to represent OpenStack CINDER.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 04, April-2022

Figure 5: Icon for OpenStack CINDER

VII. KEYSTONE

No OpenStack cloud would be able to exist without the services provided by Keystone service. So, Keystone is used for authentication and authorization and it lists all current services and endpoints [11]. A service is one of the projects that is implemented in OpenStack. So, if you want to be able to access Nova, Nova needs to be defined as a service in Keystone. And the endpoint is providing a URL that provides access to the specific service. As an administrator, you will figure that you can query these services and endpoints, as well. And I will show you that in an upcoming lesson in this course.

Keystone is a core element in OpenStack. In Keystone, also, the users and roles are created and they are assigned to projects, which are also known as tenants. A project, also known as tenant, typically, is a customer of OpenStack. If OpenStack is used as a public cloud, that can be different companies that are hiring cloud space in OpenStack. And, in order to distinguish between the resources available to one customer and another customer, OpenStack is using this notion of projects or tenants.

Within a project environment, you will typically be creating user accounts - user accounts that will be assigned specific roles and, depending on the role that is assigned to a user account, users will have more or less options to do in OpenStack. And Keystone is taking care of all that. So, it's a central repository for all services and endpoints results, and we will see that there are nice commands that allow you to query this repository. By default, Keystone is using a database, which is the MariaDB database [24], to store information. The default is MariaDB, but that doesn't mean that you can't change it. OpenStack is very flexible. So, if you have an OracleDB [31] that you want to use, that's fine as well. Or, if you want to put it in an Lightweight Directory Access Protocol (LDAP) directory [8], that can be done also. Figure 6 shows the icon used to represent OpenStack KEYSTONE.

Figure 6: Icon for OpenStack KEYSTONE

VIII. HORIZON

Horizon is the Dashboard and it provides a web interface for easy management of instances and other OpenStack properties [29]. So, Horizon is one of the most popular components of OpenStack. It's used very frequently, because it's a lot easier than using the powerful command line interface, and that is why end-uses like using Horizon.

This is what anybody can use, or any application can use, to request information directly from OpenStack. And the answer is provided in the json format. That's not very readable. So, if you are not a developer, probably you are not going to like the API too much. That is why, from the command line, you can use OpenStack commands, as well. For example, OpenStack endpoint list, which is presenting more or less the same information as the API call, that I've just shown using curl. But still, this is not something that many users will get very enthusiastic from, because you need to be able to work with complicated commands, and that is why there is Horizon.

So, login to the Horizon interface on OpenStack. Let me login as admin for instance. You'll notice when working from Horizon that the perspective that you get as the admin user is different from the perspective that you will see as a tenant user. The admin user is responsible for managing tenants, as well as cloud infrastructure. And a tenant user is responsible for managing the tenant environment. So, here we can see that currently three tenants exist. There's the admin tenant that's the scope-free administrator, the demo tenant, which on the CentOS PackStack deployment has been created automatically, providing you a demo environment that you can use to get started in the easy way with OpenStack. And there's a services tenant. Now, within this admin environment, you can do generic stuff, like monitoring resource usage, like getting an overview of what your cloud is doing ... So, here you can see the different resources and what they are currently doing. Now, let me login as an end user, as well. So, here we can see the environment that the demo user can use. For example, there's some networking. That's networking that has been created automatically.

This is Software Defined Networking [32]. It will be presented in an easy way to the tenant user. So we can see there's a router, there's a private network, we can see the subnet range on the private network, we can see the public network, as well, which is the external network, and we can see the router that exists, as well. And like this, there is everything the user wants to be able to use. For example, the instances - no instances have been created yet, and I will explain this later, but, what you can see here is a limited environment, where tenant users can create whatever environment they need to be creating. Figure 7 shows the icon used to represent OpenStack HORIZON.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 04, April-2022

Figure 7: Icon for OpenStack HORIZON

IX. OTHER OPENSTACK PROJECTS

Heat is used for deploying stacks of instances in an OpenStack environment. This allows multiple related virtual machines to be deployed in an easy way. Heat uses the HOT (Heat Orchestration Template) or the AWS CloudFormation template format to define the stacks. There are many other third party solutions offering additional options for working with templates [23]. Figure 8 shows the icon used to represent OpenStack HEAT.

Figure 8: Icon for OpenStack HEAT

Trove is Database-as-a-Service for OpenStack, provisioning relational and non-relational Database engines. Its purpose is to offer database functionality to users without the need to deal with complex administration tasks [22]. As a result, users, as well as administrators, can easily deploy database instances. Figure 9 shows the icon used to represent OpenStack TROVE.

Figure 9: Icon for OpenStack TROVE

Sahara is big data in OpenStack. It was configured to offer easy Hadoop deployment on top of OpenStack. Sahara takes care of Hadoop parameters, such as version, cluster topology, and hardware. The user provides these parameters and Sahara will then automatically deploy the cluster [12]. After the cluster is deployed, Sahara can easily add and remove nodes to the cluster, as necessary. Figure 10 shows the icon used to represent OpenStack SAHARA.

Figure 10: Icon for OpenStack SAHARA

Ironic is the OpenStack bare metal provisioning program and was developed to deploy physical machines (and not virtual machines). Ironic is a bare metal hypervisor API with a set of plugins which interact with the bare metal hypervisors. It uses PXE (Preboot Execution Environment) or IPMI (Intelligent Platform Management Interface) in order to provision and turn machines on or off. Ironic also works with vendor-specific plugins to provide additional functionality [19]. Figure 11 shows the icon used to represent OpenStack IRONIC.

Figure 11: Icon for OpenStack IRONIC

Zaqar is the OpenStack Messaging Service. It is an alternative to a stand-alone installation of the messaging service, such as AMQP (Advanced Message Queuing Protocol) or ZeroMQ. Zaqar can use an HTTP-based REST API, as well as a websocket-based API for communication. Zaqar enables the handling of multi-tenant queues and provides internal high availability and scalability (which is challenging in the stand-alone messaging services) [26]. Figure 12 shows the icon used to represent OpenStack ZAQAR.

Figure 12: Icon for OpenStack ZAQAR

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 04, April-2022

Originally based on Cinder, Manila is the OpenStack shared file system service. It offers an alternative to local storage or object storage, thus providing a flexible storage solution. The file access is share-based, not block-based, like running Samba in the cloud [17].

The Designate project provides DNS as a Service for OpenStack:

- REST API access for domain and record management
- Usable in a multi-tenant environment
- Integrated with Keystone for authentication
- Integrates with Nova and Neutron for auto-generation of DNS records
- Native support for Bind9 and PowerDNS [4].

Figure 13 shows the icon used to represent OpenStack DESIGNATE.

Figure 13: Icon for OpenStack DESIGNATE

Barbican implements security and key management in the cloud, providing a REST API. It is designed for storage, provisioning, and management for all kinds of secrets, including passwords, encryption keys, and X.509 certificates [16]. Figure 14 shows the icon used to represent OpenStack BARBICAN.

Figure 14: Icon for OpenStack BARBICAN

The Magnum project was developed to enable container management in OpenStack. Its purpose is to integrate container orchestration engines as first-class resources in OpenStack [5]. This makes it possible to run containers in a similar way as instances do on top of Nova. OpenStack Magnum is a relatively new project, and it's final direction is not yet clear. Figure 15 shows the icon used to represent OpenStack MAGNUM.

Figure 15: Icon for OpenStack MAGNUM

The Murano project provides an Application Catalog to OpenStack. The purpose of this project is to enable application developers and cloud administrators to publish available applications in a catalog [30]. This allows users to consult a list of cloud applications in an easy browsable way. Figure 16 shows the icon used to represent OpenStack MURANO.

Figure 16: Icon for OpenStack MURANO

Congress is Policy as a Service. This project provides Governance and Compliance as services in the cloud. The administrator can define policies and Congress will take care of implementing these policies [27].

X. DISCUSSION: OPENSTACK SERVICES

So, we have just talked about these core OpenStack services. Behind the core OpenStack services there's something else as well. Some vital services are always required, and these include, to start with, time synchronization. That is required because many services are using time stamps for communication. In Keystone, Keystone is issuing tickets that are based on time stamps and, if the services don't agree on the right time, then, there will be no communication. There's a database. The MariaDB database that we've been talking about when we talked about Keystone. This is used for storing all of the cloud-related information. That's pretty interesting, because, as an administrator, you will be able to query the database and get the information about what's happening out of the database. And there is the message queue.

The message queue is an essential component that services access to pass messages in an orderly way between services. It is like an Simple Mail Transfer Protocol (SMTP) mail server in email handling. It is a service that takes care of delivering the message to the right destination. So, in an OpenStack deployment that is automated, all of this will be created automatically for you and you will notice that it is created as one of the first things. If you want to deploy OpenStack manually, you need to take care of these three essential services yourself.

To understand OpenStack, you should understand RESTful API. Representational state transfer (REST) is a generic method to provide access to all OpenStack components. All OpenStack APIs are RESTful, which provides uniform access. And this makes work for developers easy, as the same standards are used throughout. And a developer can use the same language throughout everything he's doing. RESTful API access can be used while implementing commands, but also directly using curl. And I want to give you a short demo of what the RESTful API makes possible.

XI. CONCLUSION

I have certainly not listed every project available for OpenStack nor is this paper an endorsement of a specific project, but it might give everyone a context of some of those major projects that you tend to hear about related to OpenStack. For example, Neutron is the networking core. Swift is a storage technology that we can do direct puts and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 04, April-2022

gets HTTP calls directly into that object storage environment. Virtualization is done through Nova and that image management. The repository of images used for the virtual machines is Glance. Zaqar helps with messaging and moving messages throughout the infrastructure and then Designate assists with our domain name services. Service identity management is handled by Keystone. Key management is enabled by Barbican where you're keeping your tokens your secret data within the environment that needs to be highly secure. Orchestration and dashboard services are handled by Heat and Horizon. Governance which would typically be external to an operating system is maintained by Congress. OpenStack is very API-oriented is handled by Manila. The database-as-a-services is managed by Trove which allows you to connect to multiple data structures within your environment, both SQL and noSQL. Future OpenStack projects can include computational intelligence techniques as shown in [15] to expand its services range.

REFERENCES

[1] Anilkumar, Chunduru, and Sumathy Subramanian. "A novel predicate based access control scheme for cloud environment using open stack swift storage." Peer-to-Peer Networking and Applications 14, no. 4 (2021): 2372-2384.

[2] Arnold, Joe. Openstack swift: Using, administering, and developing for swift object storage. " O'Reilly Media, Inc.", 2014.

[3] Benjamin, Bruce, Joel Coffman, Hadi Esiely-Barrera, Kaitlin Farr, Dane Fichter, Daniel Genin, Laura Glendenning et al. "Data protection in OpenStack." In 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), pp. 560-567. IEEE, 2017.

[4] Benomar, Zakaria, Francesco Longo, Giovanni Merlino, and Antonio Puliafito. "A Stack4Things-based web of things architecture." In 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), pp. 113-120. IEEE, 2020.

[5] Bolivar, Luis Tomas, Christos Tselios, Daniel Mellado Area, and George Tsolis. "On the deployment of an open-source, 5G-aware evaluation testbed." In 2018 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 51-58. IEEE, 2018.

[6] Bollig, Evan F., Graham T. Allan, Benjamin J. Lynch, Yectli A. Huerta, Mathew Mix, Edward A. Munsell, Raychel M. Benson, and Brent Swartz. "Leveraging openstack and ceph for a controlled-access data cloud." In Proceedings of the Practice and Experience on Advanced Research Computing, pp. 1-7. 2018.

[7] Chadwick, David W., Kristy Siu, Craig Lee, Yann Fouillat, and Damien Germonville. "Adding federated identity management to openstack." Journal of Grid Computing 12, no. 1 (2014): 3-27.

[8] Chitpinityon, Surachai, and Maharat Tossa. "New Approach for Single Sign-on Improvement using Load Distribution Method." In 2021 Research, Invention, and Innovation Congress: Innovation Electricals and Electronics (RI2C), pp. 44-47. IEEE, 2021.

[9] Datt, Aparna, Anita Goel, and S. C. Gupta. "Analysis of infrastructure monitoring requirements for OpenStack Nova." Procedia Computer Science 54 (2015): 127-136.

[10] Denton, James. Learning OpenStack Networking (Neutron). Packt Publishing Ltd, 2015.

[11] Ismail, Salih, Hani Ragab Hassen, Mike Just, and Hind Zantout. "Availability in Openstack: The Bunny that Killed the Cloud." In International Conference on Applied CyberSecurity, pp. 114-122. Springer, Cham, 2021.

[12] Jadhav, Bhushan, and Archana B. Patankar. "A framework for integrating cloud computing and big data analytics into e-governance using Openstack Sahara." In Information and Communication Technology for Intelligent Systems, pp. 705-714. Springer, Singapore, 2019.

[13] Jain, Pragya, Aparna Datt, Anita Goel, and Suresh Chand Gupta. "Cloud service orchestration based architecture of OpenStack Nova and Swift." In 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 2453-2459. IEEE, 2016.

[14] Kengond, Shivaraj, D. G. Narayan, and Mohammed Moin Mulla. "Hadoop as a Service in OpenStack." In Emerging Research in Electronics, Computer Science and Technology, pp. 223-233. Springer, Singapore, 2019.

[15] Khan, Koffka, and Ashok Sahai. A levy-flight neuro-biosonar algorithm for improving the design of eCommerce systems. Journal of Artificial Intelligence, 4(4), pp. 220–232, 2011.

[16] Kuzminykh, Ievgeniia, Bogdan Ghita, and Stavros Shiaeles. "Comparative analysis of cryptographic key management systems." In Internet of Things, Smart Spaces, and Next Generation Networks and Systems, pp. 80-94. Springer, Cham, 2020.

[17] León, José Castro. "Advanced features of the CERN OpenStack Cloud." In EPJ Web of Conferences, vol. 214, p. 07026. EDP Sciences, 2019.

[18] Lima, Stanley, Álvaro Rocha, and Licinio Roque. "An overview of OpenStack architecture: a message queuing services node." Cluster Computing 22, no. 3 (2019): 7087-7098.

[19] Lingayat, Ashish, Avinash Singh, Vinay Naik, Ranjana R. Badre, and Anil Kumar Gupta. "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), pp. 1-6. IEEE, 2018.

[20] Malla, Akash A., Sumedha Shinde, D. G. Narayan, and Mohammed Moin Mulla. "Self-Managed Block Storage Scheduling for OpenStack-based Cloud." Procedia Computer Science 171 (2020): 1439-1448.

[21] Noertjahyana, Agustinus, Henry Novianus Palit, Reynaldo Chandra, Justinus Andjarwirawan, and Lily Puspa Dewi. "Comparative Analysis of NFS and iSCSI Protocol Performance on OpenStack Cinder Technology." Procedia Computer Science 171 (2020): 1498-1506.

[22] Okafor, K. C., J. A. Okoye, and R. M. Onoshakpor. "Towards Smart Green Energy Metering Design for OpenStack/Amazon Elastic Cloud Integration." In 2019 IEEE PES/IAS PowerAfrica, pp. 328-333. IEEE, 2019.

[23] Pinzaru, Ciprian, Valeriu Vraciu, Paul Gasner, and Octavian Rusu. "Testing of Grid Worker Nodes Integration in OpenStack." In 2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet), pp. 1-4. IEEE, 2021.

[24] Robillard, Simon, and Hélène Coullon. "SMT-Based Planning Synthesis for Distributed System Reconfigurations." In International Conference on Fundamental Approaches to Software Engineering, pp. 268-287. Springer, Cham, 2022.

[25] Rosado, Tiago, and Jorge Bernardino. "An overview of openstack architecture." In Proceedings of the 18th International Database Engineering & Applications Symposium, pp. 366-367. 2014.

[26] Serrano-Iglesias, Sergio, Eduardo Gómez-Sánchez, Miguel L. Bote-Lorenzo, Juan I. Asensio-Pérez, and Manuel Rodríguez-Cayetano. "A self-scalable distributed network simulation environment based on cloud computing." Cluster Computing 21, no. 4 (2018): 1899-1915.

[27] Tabiban, Azadeh, Suryadipta Majumdar, Lingyu Wang, and Mourad Debbabi. "Permon: An openstack middleware for runtime security policy enforcement in clouds." In 2018 IEEE Conference on Communications and Network Security (CNS), pp. 1-7. IEEE, 2018.

[28] Tkachova, Olena, Mohammed Jamal Salim, and Abdulghafoor Raed Yahya. "An analysis of SDN-OpenStack integration." In 2015 Second International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), pp. 60-62. IEEE, 2015.

[29] Wang, Hongbin, Shaoxu Li, Dan Deng, Leixiao Li, Jing Gao, and Jie Li. "A Simple Dashboard for OpenStack Horizon." In 2021 4th International Conference on Computing and Big Data, pp. 135-141. 2021.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 04, April-2022

[30] Wang, Ning, Qianlong Lan, Xuemin Chen, Gangbing Song, and Hamid Parsaei. Development of a Remote Laboratory for Engineering Education. CRC Press, 2020.

[31] Wee, Chee Keong, Xujuan Zhou, Raj Gururajan, Xiaohui Tao, and Nathan Wee. "Adaptive Fault Resolution for Database Replication Systems." In International Conference on Advanced Data Mining and Applications, pp. 368-381. Springer, Cham, 2022.

[32] Yang, Renbo, and Junxing Zhang. "imuLab: Internet of Things Simulation Platform Based on OpenStack and Container Technology." In 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS), pp. 927-932. IEEE, 2021.