Presented at the DLSU Research Congress 2019
De La Salle University, Manila, Philippines
June 19 to 21, 2019

Use of Container Technology in an Academic Cloud Computing Environment

Kevin Michael M. Dela Cruz, Carl Anthony P. Genio, Angel Phonzo B. Tan, Miguel L. Uy, Danny C. Cheng

Information Technology Department
College of Computer Studies De La Salle University
**Corresponding Author: danny.cheng@dlsu.edu.ph*

Abstract: Traditional Information Technology Infrastructure is no longer coping up with the changing business needs. Cloud computing allows organizations to have the agility to quickly provision, allocate and deliver on-demand IT resources to meet ever-changing business needs. In the paper, the researchers provided an architecture that incorporates the use of the open-source cloud platform OpenStack and the open-source container platform Docker in the context of usage patterns in an academic institution offering information technology degrees and courses. This research considered academic usage scenarios such as conducting various development classes in computer laboratories as well as offering both students and faculty members their own virtualized workstation in the private cloud. By using a cloud platform, this research addresses issues that include allocation, maintenance and configuration of computer laboratories as workstation instances in the virtualized cloud environment can be easily restored to a working state in case of failures. And by combining containers into the cloud platform, each individual can select tools and services pre-packaged in containers to incorporate and deploy for use in their own virtual workstation without the need for administrator intervention as well the concern of conflicts and configuration problems. Specifically, the research considered basic programming courses, database system courses, and development courses requiring a deployment environment or DevOps like environment. Results of the research shows that by incorporating container platform in a cloud environment would increase flexibility, decrease redundancy, and improve on resource utilization efficiency. Challenges such as user access management in concurrent use of a common workstation instance were also introduced when containers were incorporated as the scope of the virtualization and encapsulation have been reduced to just the containers as oppose to the entire workstation instance.

Key Words: cloud computing, container, OpenStack, Docker, Academic Cloud Environment, virtualization

Presented at the DLSU Research Congress 2019
De La Salle University, Manila, Philippines
June 19 to 21, 2019

1. INTRODUCTION

Organizations are now transitioning from the traditional IT infrastructure to a cloud infrastructure because of its benefits. A cloud infrastructure is a solution to efficiently utilizing the IT infrastructure of an organization. It provides an on-demand self-service platform which allows users to quickly and automatically gain access to the IT resources without requiring any additional human interaction. It also provides a broad network access which users can access a service from any device that is connected to the internet. By allowing pooling of resources and elasticity, users of a cloud environment can scale up or down the resource requirements as the need arises or diminishes (Introduction to OpenStack 2017)(Wood, T. 2011).

Even though the basic cloud environment offers several advantages over the traditional IT infrastructure environment, there is still room for improvement. In our previous work (Añonuevo R. 2017) , it was realized that creation of individual instances from pre-built template images in the OpenStack environment as the need or demand arises is not practical or feasible due to the amount of time, space and resources consumed during this activity which would have to be multiplied to the number of individuals that will perform this action. Long boot times were also encountered when the virtual workstation would have to be booted up from a shutdown state as concurrent bootups would be initiated in the context of starting a class in a computer laboratory. Software licensing for items such as the operating system would also become a concern if individuals would create multiple instances on a per course configuration basis as this would have a multiplier effect on the number of workstations assigned and used by and individual which would eventually lead to the need to acquire more to comply with licensing terms.

Due the issues experienced with a basic cloud environment, this research employed the use of containers to reduce the amount of custom templates and instance that would have to be created thereby reducing the amount of resource and license requirements. By using containers, the template used in creating instances can also be reduced in size as it can now be generalized to a common template that contains minimal tools installed and just allow each user to download containers of tools and services as needed (Shapland, R. 2016). By reducing the size, reduction in boot time is also achieved and can be significant as any amount saved is multiplied in scale to the number of individuals that would use the environment. At the same time, by removing the need to customize the templates and instances, the concern on software licensing is also greatly reduced.

2. METHODOLOGY

2.1 Scenarios Considered

Three types of courses considered in the research, Basic Programming courses that uses OpenJDK, Database courses where the language used is focused on SQL and the resource requirements consumes resources when executing a query within a database where the consumption of resources depends on how efficient the query is and on how large the size of the database is, Development courses where the student would be required to simulate a production environment where they would need to use a Web Server and a Database Server. The course would also need graphical tools such as an IDE, a Browser, Database Management Tool, and software emulators for mobile device development.

2.2 Architecture and Implementation

The architecture of the private cloud infrastructure is based on the design of the general architecture of OpenStack (Fig. 1).

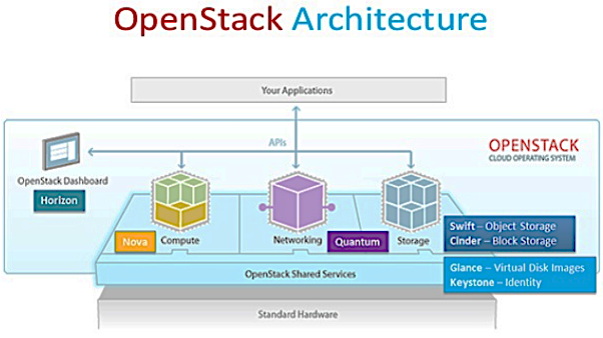

Fig. 1. OpenStack cloud platform architecture

The architecture of OpenStack focuses mainly on the controller node where most of the services of OpenStack are located such as the Horizon (Dashboard) which is the user interface for OpenStack, Glance (Image Service) where the images of OpenStack is managed and provided, Neutron (Network Service) which provides network connectivity for the instances and containers and where the network inside OpenStack is managed, Keystone (Identity Service) which is responsible for authentication and authorization services inside OpenStack, and Nova-API service which is responsible for scheduling the guest instances to the compute node. Also, located inside the controller node is the MariaDB(Database) where metadata of each services and nodes is stored, RabbitMQ (Advanced Messaging Queuing Protocol) a message broker to be used to communicate to other services and nodes inside OpenStack, Nova Compute where it is responsible for running instances and provisioning resources such as memory, CPU cores, and storage to the instances. A Block Storage node is also installed in the OpenStack setup which is responsible for creating additional Volume Storages for the guest instances.

Each student account in OpenStack and is given their own virtualized workstation instance within OpenStack and inside each of the instances a Docker service runs the Docker engine responsible for creating the Docker Container instances, and a service for storing each Docker Images created based on scenarios considered in Section 2.1 (Fig. 2). Given that each workstation is virtual and individualized, the student is also given administrative access to their own workstation.

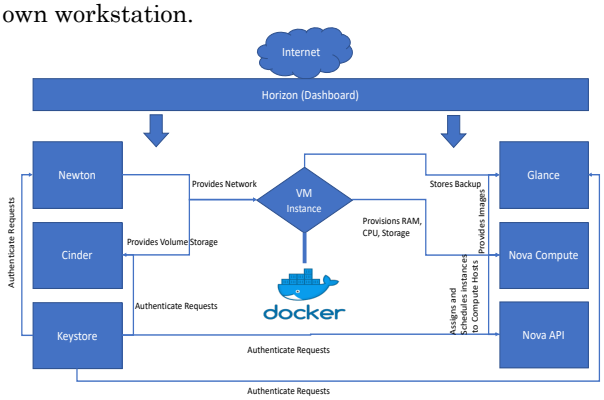

Fig. 2. An OpenStack cloud environment implementation with Docker container technology in the compute node virtual instance.

Each student can access Docker directly on their own virtual workstation, they are given the ability to download container images directly for execution depending on their needs and requirements. Container images can be downloaded from the official Docker Hub or the academic institution can develop and customize their own images and upload them directly to the private cloud for use of the students.

3. RESULTS AND DISCUSSION

To evaluate and validate the implementation, the research implemented performance and resource utilization tests to determine if containers do reduce consumption and improve on performance, A CentOS with Docker image was used in these tests. The image contains various applications that are needed in different classes such as PHP, MySQL, Tomcat, Python and OpenJDK. The flavor used in this test: 1GB RAM, 25GB total disk and 25GB root disk. OpenStack allows to overcommit the CPU and RAM on compute node. The CPU allocation ratio is 16:1, this means that the scheduler allocates up to 16 virtual cores per physical core. Since there are 4 physical cores, there are 64 vCPUs. The RAM allocation ratio is 2.5:1, this means that the scheduler allocates instances to a physical node as long as it doesn't exceed the 2.5 times the total amount of RAM available on the physical node. There is 20GB physical RAM therefore there is a total of 50GB virtual RAM.

In evaluating performance, the load time and amount of instances that are actually loaded on the same resources were tested. As seen on Table 1, the Docker container implementation was able to start 22% more instances and was able to reduce the number of erroneous loading to just 28% of the basic OpenStack implementation. Loading time for concurrent use also show more than 2x the improvement in speed ups. Before all instances were launched, the group started one instance to determine the time it would take to startup. It took 9.48 seconds to run and 40.59 seconds for the login page to show in the instance. After this, all 22 instances were launched at the same time. All instances were able to run. The first instance took 16.8 seconds to run and 14mins and 55 seconds to display the login page. All 22 instances booted up and 3 of them prompted a

different startup screen. It still ran the CentOS Docker image, but it showed a different start up screen (emergency mode) from other instances

Table 1. Performance Comparison

Criteria	OpenStack time	OpenStack w/ Docker Time
Instances that started up	18	22
Instances that prompted login screen	7	19
Instances that prompted different startup screen	11	3
Duration of one instance (one-by-one)	42.76 s	9.48s
Duration of first instance (multiple)	37.42 s	16.80s

On Table 2, the research evaluated the behavioral differences for with and without the use of containers. It shows that maintenance is reduced as the number of templates needed is also reduced. It also shows that when using shared containers, management may have problems due to uneven use of resources as the quota system is not as developed compared to the basic OpenStack.

Table 2. Behavioral Comparison

Criteria	OpenStack	OpenStack w/ Docker
Image Template	Several images templates with varying specifications for each course	One image template and use of docker images as needed
Quota Distribution	The quota can be divided unevenly to more Virtual Machine.	The quota can be shared if on a shared instance
Boot	Virtual Machine Instance takes longer time to boot because it has a larger size.	Virtual Machine Instance that has docker takes a shorter time to boot.
Capacity	RAM, vCPU, and storage consumption is higher	Lower RAM, vCPU, and storage consumption via docker instances

4. CONCLUSIONS

The research was able to apply a customizable deployment of applications on the private cloud infrastructure through the container technology. Furthermore, tests were done to assess the capabilities of Docker inside OpenStack. For OpenStack without Docker it consumes more resources at it grows than OpenStack with Docker. It is possible to use Docker as a standalone service within OpenStack but not inside any virtual workstation instance running via KVM. This would consume less resources than what has been currently implemented. However, the service needed to execute this approach was still under development during the time of the research. However, once the service has been published and stable, the research highly recommends it instead to fully utilize the advantages of using Docker with OpenStack. The research also recommends the use of a docker cluster manager such as docker swarm or Kubernetes, through the use of an OpenStack service called Magnum, as it will help deploy docker containers in a flexible and convenient method.

5. REFERENCES

Introduction to OpenStack. (n.d.). Retrieved March 17, 2017, from https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html.

Añonuevo R. , Ferrer D., Llanita F., Mercado M., Cheng D. (2017) Towards a Private Cloud Infrastructure Implementation for a University delivering IT curriculum using OpenStack

Shapland, R. (2016). Cloud containers - - what they are and how they work. Retrieved from http://searchcloudsecurity.techtarget.com/feature/Cloud-containers-what-they-are-andhow-they-work

Wood, T. (2011). Improving Data Center Resource Management, Deployment, and Availability with Virtualization (Doctoral dissertation). Retrieved from http://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1495&context=open_access_dissertations.