International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-2S2, January 2019

A Survey on Paa S Container Platform for Application Deployment

K. Raghava krishna, B. Amutha

Abstract— In today's world, Cloud Computing has become an important base for variety of services like deployment, storage, virtualization etc. So due to this data being generated in the cloud needs to be maintained and secured. In order to maintain we need better ways to manage data and even make resource allocation easier and secure to different clients. Previously and even now Virtual Machines (VM's) are being used by creating different VM's for different clients or for application deployment using VM's. Containers have now replaced these VM's for application deployment and for providing better storage methods. But there is a (Container Sprawl) where security, provisioning, deployment can be extremely challenging for different applications where Openshift platform can be helpful. In order to overcome these problems, a new platform called OPENSHIFT is being used to deploy applications, analysing network and security issues in Containers. Openshift helps containers to overcome these issues and makes organization's adopt container technologies. In this paper, we will use Openshift for Container deployment, analyse various ways and problems for deploying Openshift.

Keywords: - PaaS, Virtual Machines, Containers, Openshift, Openstack

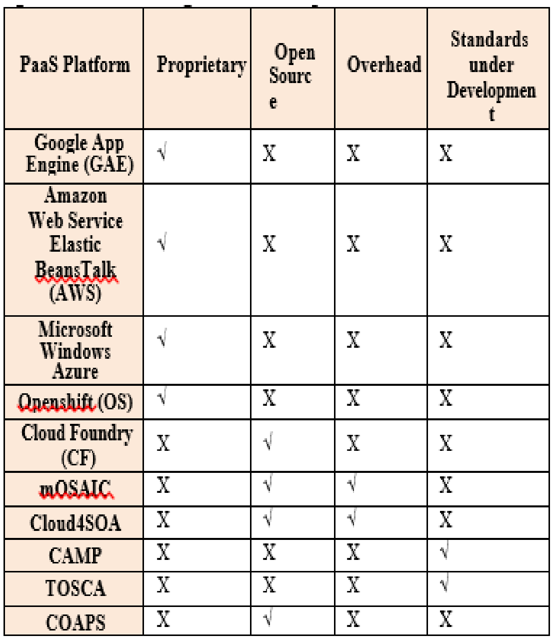
I. INTRODUCTION

Cloud Computing Platform is mainly based on three platforms at different stages, they are IaaS acts as the bottom layer, PaaS acts as the middle layer and SaaS acts as the top layer. In the paper we would completely focus on PaaS platform which is the middle layer between IaaS and SaaS. PaaS is simpler and flexible to work when compared to SaaS. IaaS examples are Openstack, Microsoft Azure, etc. Examples of PaaS platform are Openshift, Google App Engine etc[1]. PaaS platform helps developers design and write their own codes for applications.

Developers normally use programming languages' like Java and Python for writing codes for the applications. PaaS infrastructure contains development tools, database systems and it works on a cycle of building, testing, deploying, managing and updating [2]. Applications such as Apache, Nginx, etc. applications are coded by developers and people directly access the cloud PaaS for using these application for their work.

Manuscript published on 30 January 2019.
* Correspondence Author (s)
K. Raghava Krishna, Department of Computer Science and Engineering, SRM UNIVERSITY, Kattankulathur (k.raghavkrishna@gmail.com)
B. Amutha, Department of Computer Science and Engineering, SRM UNIVERSITY, Kattankulathur (bamutha62@gmail.com)

In order to access these applications or develop them we require different platforms in PaaS to support these operations. Application deployments require specific platforms to run these applications or to even develop them. Nowaday's Virtual Machine's (VM's) are being used for application deployments and the tool used for this application deployment is Openstack. Openstack is a software that supports different services such as compute, storage and networking and is IaaS platform.



PaaS Platform	Proprietary	Open Source	Overhead	Standards under Development
Google App Engine (GAE)	√	X	X	X
Amazon Web Service Elastic BeansTalk (AWS)	√	X	X	X
Microsoft Windows Azure	√	X	X	X
Openshift (OS)	√	X	X	X
Cloud Foundry (CF)	X	√	X	X
mOSAIC	X	√	√	X
Cloud4SOA	X	√	√	X
CAMP	X	X	X	√
TOSCA	X	X	X	√
COAPS	X	√	X	X

Table 1: Available PaaS Platforms [3]

The users do not require to buy any infrastructure as they can access the cloud easily and create a working environment for their work using Openstack. Openstack is made up different components for different purposes like storage, dashboard, authentication, images etc. So these help in deploying applications easily. But there are some drawbacks while deploying applications using VM's like startup time, storage requirements etc. Lets discuss on comparision between Openstack and Openshift and introduce Openshift and its advantages.

This paper has, Section I. Introduction on PaaS platforms and a comparison is shown based on the parameters of different PaaS platforms. Section II. It is about the Literature Survey done on Openshift and Openstack platforms. Then Section III. Explains about the architecture of Openshift, how Docker and Kubernetes help in building up the Openshift platform. Section IV.

It deals with the Openshift deployment, as how the deployment was done, different platforms in Openshift and the issues and solutions for those issues faced during deployment is discussed. Section V. Gives an insight on the Conclusion and Future Scope. Section VI. Gives us the References used in the paper is mentioned.

II. LITERATURE SURVEY

New technologies have developed for application deployment after VM's, a new way of deployment is now becoming popular which is 'Container Deployment'. Container Deployment is better due to easy setup and less resources required to run applications compared to VM's. No we would be looking into the performance analysis and parameters between both the application deployments.

2.1 PERFORMANCE ANALYSIS

Rabindra K. Barik. ETL [3], have done a performance analysis comparison between VM's and Container's to determine the best deployment method for applications. Survey was done on parameters such as Security, Overhead, startup time, etc.

The main aim behind this paper is to check overhead that is introduced by these techniques, when running on a typical edge device. Here the native performance has been taken as the source to do virtualization overhead of both solutions. Different benchmark tests were done to compare the 2 solutions such as CPU/CPU Scheduler, Disk I/O, Network, Memory evaluations. Each benchmark was tested 15 times continuously and the average performance results were taken.

Component	Hardware Configuration	Software Configuration
Host	Intel i5 4GB RAM 500GB 7200RPM	Ubuntu14.04
VM	Intel i5 3GB RAM 100GB 7200RPM	Ubuntu14.04
Container	Intel i5 3GB RAM 500GB 7200RPM	Ubuntu14.10

Table 2.System Specification

Attributes	VM	Container
Guest Operating System	Each VM runs a top of hypervisor and kernel loaded into its own memory region.	All guests share the same kernel. Kernel image is loaded in its physical memory.
Performance/ Efficiency	Suffers light over head as the machine instructions get translated from guest to host OS	Almost native performance as compared to the underlying host OS.
Security	Complete Isolation	Isolation using namespaces
Storage	Takes more storage	Take less storage as the base OS is shared
Isolation	Higher level of Isolation–Need special techniques for file sharing	Subdirectories can be transparently mounted and can be shared
Networking	Can be linked to virtual or physical switches. Hypervisor have their own buffers for IO performance improvement etc.	Leverage standards like IPC mechanisms like signals, pipes, sockets, etc. Advanced features like NIC not available
Bootup Time	Take a few minutes to boot up	Containers boot up in a few seconds as compared to VMs

Table 3. Performance Comparison

From the above survey it was seen that:-

- From the test's conducted between VM's (KVM) and Containers (DOCKER), it was seen that VM's slightly suffered **overhead** compared to Container's.
- Security of Container can be extended in future for more research.
- As it was seen that Security in VM's was better than Container's since VM's had a high isolation.
- In VM's networking was supported by **hypervisor's** whereas in Container's **sockets** were used but required more advance features comparatively.
- Boot up time is better for containers over VM's, since VM's require the whole OS to start whereas Container's just the application.

Fl´avio Ramalho ETL [4], also performed an analysis to analyze virtualization technique comparison for application deployment and management but at the context of IoT devices and network edge.

In this paper, performance result evaluation was done with various benchmark tools between hypervisor-based and container-based solutions on the device that can be used as a gateway in IoT for general usage.

Platform	Memory Index		Integer Index		Floating-Pt Index	
Native	4.328	0	4.500	0	0.420	0
Docker	4.100	-0.12%	4.209	-0.80%	0.420	0.00%
KVM	3.900	-4.30%	4.121	-2.88%	0.420	-1.50%

Table 4. CPU Benchmark

Platform	CPU (s)		Threads (s)	
Native	145.547	%	7.194	%
Docker	145.918	+0.25%	9.695	+34.77%
KVM	147.250	+1.17%	78.529	+991.65%

Tablee 5. CPU Scheduler Benchmark

Platform	Copy		Scale		Add		Triad	
Native	1759.4	%	806.8	%	654.6	%	534.7	%
Docker	1754.5	-0.28%	804.8	-0.25%	652.8	-0.27%	533.2	-0.28%
KVM	1723.4	-2.05%	786.3	-2.54%	641.2	-2.05%	523.7	-1.78%

Table 6. Memory Benchmark

From the survey the result was:-

- In some cases VM's didn't perform so well. As VM's took minutes to boot up, this time was enough for hacker's to attack the VM's and exploit its vulnerabilities. Whereas Container's required just seconds to startup since they required less resources [5].
- VM's required more resources like memory, storage etc. since they had to load the host and guest OS, which even made mobility of VM's difficult due to large images. Containers occupy less space compared to VM's so are easy to move.
- VM's introduced more overhead compared to Container's.
- VM's offer better **networking** and **security** as they have advanced features compared to Container's and provide good **isolation**.

From the above benchmark tests Containers performed better compared to VM's. While VM's gave same performance results compared with base (non-virtualized) case.

So from the above two surveys that was done for the performance analysis between VM's and Container's, we infer that:-

- **Containers** have proven better by showing better performance results than **VM (Virtual Machines)** in terms of CPU, Memory, I/O evaluations etc.
- Since VM's introduce more **overhead** it has affected its performance mainly compared to Container's.
- Comparing these results its evident that we can move to Container's from VM's in cloud for application deployments.
- Container's too face some issues like **networking** and **security** which has to be taken care of and requires future research[6]. There is a **(Container Sprawl)** where security, provisioning, deployment can be extremely challenging.
- We need a better platform to support Containerized deployment so that issues like **networking**, **security** etc. can be overcome.

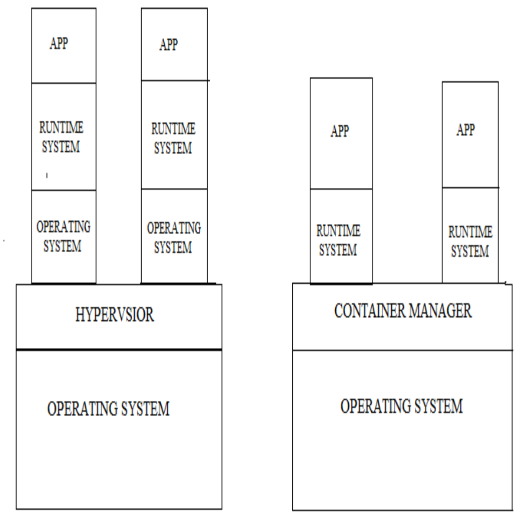


Fig 1. CONTAINER VS VM

In order to overcome these drawbacks faced in Container's such as networking, security a new application container platform called **Openshift** is introduced. Let see how Openshift helps in Container deployment and overcome drawbacks faced in Container deployment.

III. ARCHITECTURE OF OPENSHIFT

Openshift is an open source application container platform built by Red Hat, it has pods deployed on nodes, and these pods consists of multiple container's for application deployment. Openshift architecture is built upon Docker and Kubernetes. Docker is an open platform for developers to build and run applications on different devices such as laptops, cloud, VM's. Kubernetes is an open source platform for deploying and managing containerized applications and providing security for the containers.

3.1 DOCKER

Docker Containers are gaining importance and traction in IT due to its simple and easy deployment in different environments, which allow users to deploy applications faster and more efficiently [7]. Docker is an open source software that allows to use the same kernel unlike VM's which require separate individual kernels to operate. Sharing a single kernel multiple containers can be deployed, so in case of isolation VM's provide better isolation compared to containers. Docker containers just require some libraries to run containers whereas VM's require a separate OS for each machine, due to which start up time of VM's is more compared to containers[8]. Docker images can be pulled directly from Docker. Docker files can be created according to our requirements a container can be built.

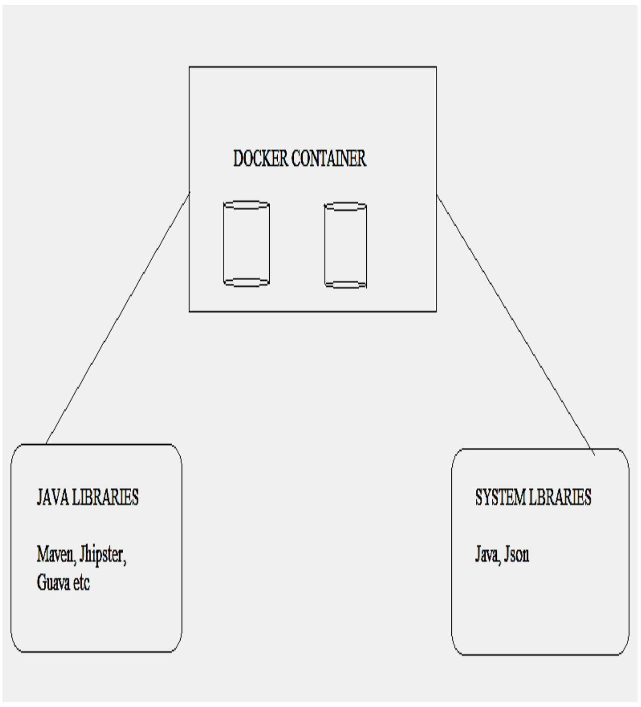


Fig 2. Docker Container Image

3.2 KUBERNETES

Kubernetes has been an open source system for automation, scaling and management of containerized applications, it also a helps to provide various workloads, containers can easily be deployed using kubernetes[9]. Kubernetes helps to maintain containers providing security, maintenance, load balancing, logging and monitoring. It maintains the life cycle of containerized applications, with kubernetes we scale applications up and down, roll updates and work with different versions of applications. Kubernetes assigns a **master** which serves as the brain and gateway for assigning work, health checking and orchestrating communications between devices [10].

Other machines act as nodes which do the assigned workloads using the required resources. Nodes perform functions such as building or destroying the container, forwarding traffic, adjusting networking parameters. These instructions are received from the master node and the work updates are provided back to the master node. Etcd is a storage configuration data that can be accessed across all nodes, it helps to maintain cluster state, leader selection details and various other configuration data.

Etcd maintenance is also maintained by kubernetes[10]. API server is required for users to change workloads in kubernetes. API acts as an interface and it contains libraries and various tools for communication between user and system.

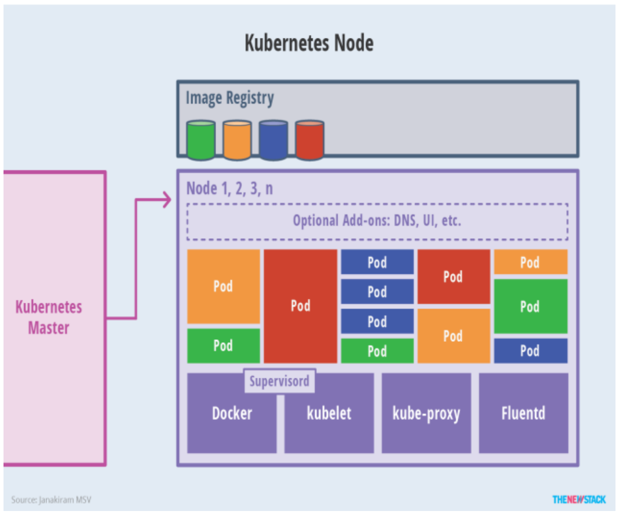


FIG 3. Kubernetes Architecture [11]

Kubelet is a service for relaying information to and from the control plane and helps to authenticate master components and workload. **Kube-proxy** is a proxy for forwarding work and requests to correct containers [10]. **FluentD** is an open source collector for data which maintains logs for developers to work on it. It acts as a unified data layer for better and faster access by systems.

IV. OPENSHIFT DEPLOYMENT

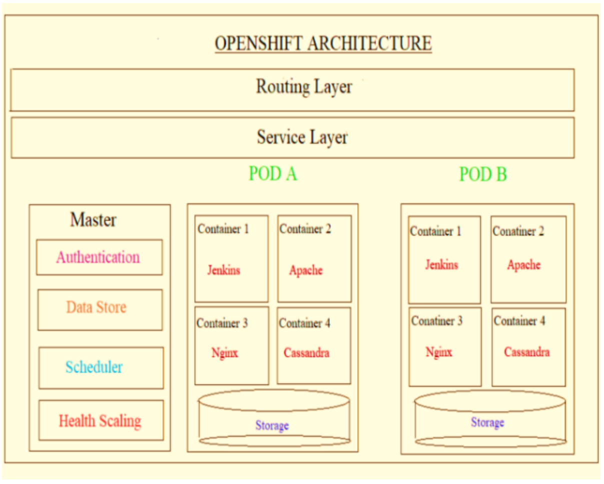


Fig 4. Openshift architecture

In this section an Openshift deployment is done and researched on the issues and solutions found for the issues faced. Here a simple deployment is being shown. Openshift deployment can be done in many ways and on different platforms [12]. Openshift supports Linux, CentOS, etc. Openshift requires various requirements for setting up a platform for Openshift deployment. Requirements for a simple deployment of Openshift using minimum number of resources:-

Platforms Supported: - Linux, CentOS

Systems Required: - 1 Master and 1Node

RAM: - Minimum 8GB for Master and 4GB for Node

Storage: - 50 GB for Master and 40GB for Node

We require minimum 20GB for Docker storage for both master and node.

4.1 Process

This Openshift Deployment was done using CentOS as the platform using VM's for both Master and Node. First a static IP has to be assigned to the VM's and then the Docker storages have to be added to the VM's storage for each master and node. Once the VM's are up, Dnsmasq has to be setup on both the Master and Node and connectivity should be tested between master and node by adding their IP addresses in their host folder. Once the dnsmasq is set a private key should be shared between the two for sharing folders [13]. Then we need to install the latest docker release and later setup docker storages by specifying the docker storage name that we have setup during installation. The main step comes now that is to install Openshift in our system, for this we have to install atomic Openshift packages like atomic Openshift installer, bridge-utils, atomic excluder, bind utils, ip_tables etc. This process should be done on both the master and node. Then we would find all the openshift tools downloaded to our VM's in the conf file. The final command would be atomic Opeshift install which will start the Openshift installation and it would ask many questions for setting up your platform like ssh key, nodes, etc. and then a message would appear saying the Openshift installation is complete.

This is one of the processes for deploying Openshift, we can deploy Openshift on different platforms and in different ways, but we require at least the minimum resources that have been mentioned above.

4.2 Openshift Platforms

Openshift has 3 types of platform

- **Openshift Online:-** In this platform[13], Redhat community provides an online cloud deployment of Openshift where we can create an account and start deploying containers and applications as per requirements all resources are provided in the cloud. Only 1 container, application, pod are given for free accounts but for more requirements, an amount has to be paid.
- **Openshift Enterprise:-** In this platform, Enterprise platforms are used by companies, businesses etc. To use this platform also an amount has to be paid according to usage.
- **Openshift Origin:-** This is the platform we have used for deployment, this platform is free open source available for all users and developers without any cost[14].

4.2.1 Deployment Issues and Solutions

1. **Faced problems while setting up the network between VM's for communication between them.**

Sol. Network was setup using dnsmasq for both the master and node by setting up the local config file for both.

2. **Allotting Dynamic IP for both master and node created connectivity problems between the both faced problems like permission denied and sometimes it used to connect.**

Sol. To solve this issue a static IP was setup during a static IP was setup during the installation of OS in the system that solved the Permission denied problem.

3. **Tried to allot docker storage in between the openshift deployment, but docker storage could not get created.**

Sol. Docker storage was allotted before while creation of VM's itself, then the docker storage could be synced easily.

4. **Openhsift version 3.10(latest version) was failing due to some missing packages and other issues**

Sol. The deployment worked well with Openshift 3.9 version, it was successful without any issues.

5. **To deploy Openshift atomic packages are required for Openshift installation, sometimes the packages couldn't get downloaded and created many problems**

Sol. This issue was resolved by reinstalling previous packages and setting up dnsmasq again.

6. **After deploying Openshift, faced problems while trying to display the web console for Openshift, as it worked only on some systems properly, when connected to a particular network additional packages were getting update and some due to this web console was failing**

Sol. This issue was solved by removing those extra packages that were added when connected to a particular network.

V. CONCLUSION

First a survey was done on the PaaS platforms to have a study about current ongoing deployments, as Openstack was being used and many applications were deployed using Openstack. Since a new platform was required for easy and fast deployment, we have gone for Openshift platform deployment since it uses containers for application deployment which is an efficient way. Container technology was brought before Openshift. But with containers using Openshift platform for application deployment was a very good and efficient thing compared to Openstack.

The main purpose for writing this paper is to do a survey on Openshift and compare it with Openstack platform and explain its benefits to the future PaaS deployments. Here also a normal deployment is shown to have an idea on how Openshift is deployed. The future scope for this would be that more features can be added to Openshift platform and can support more and more applications in future with containers, docker, kubernetes being the backbone for Openshift deployment. Improving security, ease, les overhead and many more functions would make Openshift platform a better deloyment for applications in PaaS platform.

ACKNOWLEDGEMENT

I sincerely thank my guide and staffs who contributed to and helped in this research work. This would not have been here without them.

REFERENCES

1 ZENG Shu-Qing, Xu Jei-Bin," The Improvement of PaaS Platform," in 2010 First International Conference in Networking and Distributed Computing, IEEE. pp 1

2 Eman Hossny, Sherif Khattab,Fatma Omara, Hesham Hassan, "A Case Study for Deploying Applications on Heterogenous PaaS Platforms", 2013 International Conference on Cloud Computing and Big Data, IEEE, pp 1

3 Rabindra K. Barik. Rakesh K. Lenka, K. Rahul Rao, Devam Ghose, "Performance Analysis of Virtual Machines and Containers In Cloud Computing", International Conference on Computing, Communication and Automation (ICCCA2016), IEEE, pp 2

4 Fl´avio Ramalho and Augusto Neto," Virtualization at the Network Edge:A Performance Comparison",IEEE 2016 , pp 2

5 Alok Shrivastwa and Sunil Sarat, "Openstack: A Building Environment, 19 September 2016, pp 3

6 James Turnbull, "The Docker Book: Containerization is the new virtualization", 12 July 2014, pp 3

7 Andy Hayes, "A Quick-Start Beginner's Guide", 23 January 2017, pp 3-4

8 Nigel Poulton, "Docker Deep Dive, 19 September 2016, pp - 4

9 George Sammons, "Basics of Kubernetes", May 2017, pp - 4

10 Nigel Poulton, "The Kubernetes Book: Version 2.2.", June 2017, pp 4-5

11 Janakiram MSV," Kubernetes an Overview: Version 2.2", June 2017, pp 4

12 Graham Dumpleton(June 2018), "Deploying to Openshift: A Guide for Busy Developers", pp 4-5

13 Red Hat Inc., " Openshift Container Platform 3.5 Installation and Configuration", 2018[E-Book], pp

14 Denis Zuev, Artemii Kropochav, Aleksey Usov, Learn OpenShift: Deploy, build, manage, and migrate applications with OpenShift Origin 3.9, July 2018, pp 5