Containerization in Cloud Deployment- Astudy on How this lightweight approach takesover the traditional Virtualization Technology

Sagar Lulla[1], G. SriPradha[2], Rahul Talreja[3]
[1]*BCA (Mobile Application development & Cloud Computing), Green Valley, Wanwadi, Pune - 411040*
[2]*ResearchScholar, State Resource Centre, Adyar, University of Madras, Chennai –600005*
[3]*Technical Trainer, NIIT China, Qingdao, China.*

Abstract - In today's world, Virtualization technique has enabled us to create useful IT services using resources that are traditionally bound to hardware. Although Virtualization has its advantages, there are few flaws that a concept like Containerization looks to resolve. This article, states Containerization as an alternative solution to Virtualization and not as a replacement since each technology has its own selling points. In this article, we explore the guide to Containerization that focusses on why do need this technology, its features over Virtual Machines, Microservice Architecture using Containerization and also some Use Cases.
Keywords -Virtualization, Virtual Machines, Containerization, Containers, Microservice Architecture, Monolithic Architecture, Kubernetes, Docker Swarm

I. INTRODUCTION

Virtualization has been proven to be an essential tool to sustain the ever-increasing demand for data and services. Although the concept of Virtualization has been around from many decades, it is only now that we are reaping its benefits due to the fact the services and data are evidently managed well in terms of resources and also has proven to be cost effective.
Hardware Virtualization emphasis on virtualizing the underlying hardware resources. Virtualization is a technique that creates virtual resources like network device, storage devices and compute using emulators known as Hypervisors. These Virtual Machines are further used for application hosting, website hosting and also providing various other services for production purposes. In a grand scheme of things, these virtual machines create a lot of overhead and also scalability issues down the line. For managing a few virtual machines in a production environment is sustainable, however when if you have a huge number of virtual machines deployed in your production environment the overhead also increases drastically [1]. Virtual Machines are efficiently inefficient as sometimes in a production environment the resources are allocated surplus to the requirement just to make the managing process simpler. Containers were primarily brought into the picture to nullify the weaknesses in virtual machines namely scalability and better resource management, we will discuss that in detail further. Moreover, these creations of containers are known as Containerization. Containerization focuses on an operating system level by abstracting the user spaces, thus creating isolation between the loosely coupled environment which makes it possible to run packages and applications, also allowing many containers to run simultaneously with having their own set of resources. In simple terms, containers can depict as a process within a sandbox. There are various benefits of a container such as enabling the possibility of implementing Microservice architecture. In this article, we will discuss Use Cases revolving around containers.

II. SYSTEM ARCHITECTURE

2.1 Virtual Machines –
Virtual machine acts like a real computerized environment with all independent resources just like the characteristics of a real computer. Virtual machine concept derives from the use of hypervisors which are responsible for such emulation and isolation. These virtual machines are created and run on top of the physical machine using hypervisors. A hypervisor is a software, firmware that fabricates a platform for guest machines i.e. virtual machines on top of the host machine i.e. a physical server [2]. These hypervisors act an intermediary between them and allocate resources from the host to the guest. These guest machines have their own resources such as operating system, system libraries, network adaptors, storage and more. Furthermore, using the services of guest machines

applications can be hosted. Vendors such as VMware, Oracle and Microsoft have their products for such purposes. This is represented in Fig.1

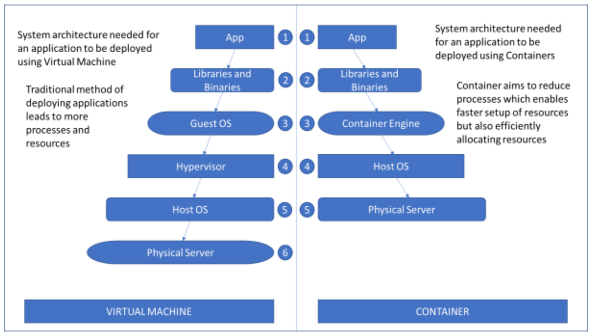
System architecture needed for an application to be deployed using Virtual Machine
Traditional method of deploying applications leads to more processes and resources
App
Libraries and Binaries
Guest OS
Hypervisor
Host OS
Physical Server
VIRTUAL MACHINE
System architecture needed for an application to be deployed using Containers
Container aims to reduce processes which enables faster setup of resources but also efficiently allocating resources
App
Libraries and Binaries
Container Engine
Host OS
Physical Server
CONTAINER

Figure 1 - System Architecture

2.2 Containers -

Unlike Virtual machines, containerization works on operating system level abstraction. Here, the resources are shared by the physical machine, the operating system with some tweaks using the LXC (Linux Containers) [3]. LXC is an Operating system-level virtualization that permits to inculcate the creation and the ability to operate on isolated Linux containers. However, Docker an open source project first started as an extension to LXC capabilities, developed High-level API's which provided a lightweight virtualization solution for running in isolation. Both work in conjunction as a Docker container does not operate on a separate operating system and acts on the host resources [4]. However, docker engine plays an important role in packaging tools like namespaces, cgroups, union file system, various libraries and binaries and any other dependencies that are needed to run the application on any docker container or on any Linux server [5] [6]. This provides a platform for applications to run in containers.

With constant evolution in docker technology, docker engine is now is not limited to a Linux based solution, it is also available for mac and windows but has various prerequisites such as toolbox and libraries required to install docker solution on the desired machine. However, Linux based systems are native and recommended.

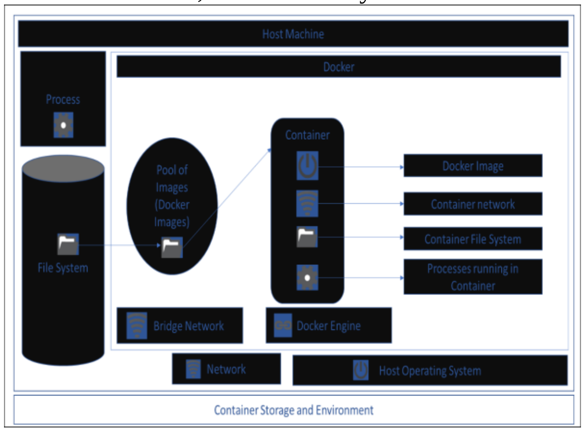

Figure 2 – Container Storage Environment

III. WHY CONTAINERS?

Although virtual machines have proved to be an essential part of cloud computing, there are few limitations that containers overcome.

Scalability- Containers were firstly introduced on the basis that containers scale up and down at ease and at a tremendous rate. In virtual machines, the scalability takes a huge amount of time to deploy the resources, whereas in containers it’s just matter of seconds. Scalability is a core aspect for containers as the workload is distributed into a cluster of containers [7]. Scaling in largely achieved by Cluster Manager tools such as Docker Swarm by Dockers and Kubernetes by Google for containers. This cluster management framework provides not only clustering of containers but also scheduling and integration facilities that help developers build and deploy multi-container environment [8].

Better Utilization of resources - Virtual machines are were created to ensure the total utilization of resources is achieved by providing limited or sufficient amount of resources. Despite of such measures, there is still underutilization of those allocated resources. So here containers come into the picture as deployment of containers, there is no need to establish/allot resources. It is dynamically achieved that is at runtime of a container, the container picks up the required amount of resources not more not less just like a process would function.

Start-up time - Deployment of virtual machines although is now scripted in plenty of scenarios, surely had removed the human efforts, however the actual time to deploy, boot up the operating system, run all the tests and also planned and unplanned events every time faces a lot of downtime. Using containers this aspect can be resolved as containers are very lightweight and act like a process running on the kernel. Containers can be deployed and are up and running within seconds, so that is a dramatic difference.

Integration - Integration in virtual machines can have multiple services such as Jenkins puppet and Ansible Nagios running on multiple virtual machines which brings a lot of scalability issues, infrastructure and more importantly cost goes considerably higher. Using containers these services can run on the same container itself, different containers that have the capability of interacting with one another and also can scale up copies of these services into multiple containers.

Performance - Virtual machines have a big amount of overhead in terms of deployments, operating system tests, and booting up time, duplication platform software for every instance of an application. Wherein, most of that is eliminated as containers share the same operating system, and other resources such as kernel, memory and network capabilities. As mentioned earlier containers can be deployed within seconds. This saves a lot of time and thereby increases the performance and overall throughput.

This has been summarized in the table given below:

Table-1Virtual Machine VS Container

FEATURE	VIRTUAL MACHINE	CONTAINERS
Computing Architecture	Monolithic architecture	Microservice architecture
Virtualization Management	Hypervisor usually hardware virtualization	Container engine works on operating system
Load Balancing	CLOUD SERVICE PROVIDER (CSP) eg: AZURE, AWS, GCP	CLUSTER MANAGEMENT SERVICES eg: KUBERNETES, DOCKER SWARM
Service	Iaas, paas, saas	Iaas, paas, saas
Fault Tolerance	CLOUD SERVICE PROVIDER (CSP) eg: AZURE, AWS, GCP	CLUSTER MANAGEMENT SERVICES eg: KUBERNETES, DOCKER SWARM
Storage	Provided by cloud service provider (csp)	Host machine
Security	Entire os is emulated and thereby offering data security and protection	Sharing os requires root access which may lead to vulnerability
Overhead	Deployment time increases as number of resources increases	Deployment time significantly decreases as commands are executed within seconds
Scalability	OPERATION EXECUTED BY CLOUD SERVICE PROVIDER (CSP) eg: AZURE, AWS, GCP	OPERATION EXECUTED BY CLUSTER MANAGEMENT SERVICES eg: KUBERNETES, DOCKER SWARM

IV. MICROSERVICE ARCHITECTURE

Microservice Architecture framework was adopted due to scenarios such as the inability to scale, unreliable, inflexible, blocks continuous development, the complexity of existing service model, comparatively high scalable costs, slow development, risk of any failure in any of tiers of the application may lead to failure of the total system. Such are the drawbacks of Monolithic Architecture framework. Microservice architecture overcomes these limitations. Microservice architecture is a cloud-native based framework which has gained popularity and widely accepted in the DevOps field. In Microservice architecture the application's system is divided into small and lightweight services that are independent and can communicate with each other using RESTful or RPC-based API's [9]. This not only helps in terms of flexibility and scalability but also reduces the risk of failure of a single point source. If one tier fails it is sure that it would not affect the functionality of other tiers and application as a whole in an unplanned scenario.

For docker containers, the Microservice service model can be implemented by the use of Cluster Management tools such as Kubernetes by Google and Docker Swarm by Docker. These two are perhaps the most popular ones and rightly so as they manage the containers and their connections in a cluster. In a production environment, either of those tools is implemented to achieve reusable images and data.

V. USE CASE

Uber is a well-known transportation network company based in the United States of America. They provide their services worldwide and are currently one of the best in their field of business.

However, they too started their journey with a monolithic based architecture. Having one system to provide the capability for drivers, customers, billing and payment seemed satisfactory that point of time. The system was rudimentary in terms of functionality. With the use of REST API, Adapters the communication was taken place with their servers and database. Their functionality such as driver management, customer database, notification management, billing and payment and more were all dependant on each other immensely.

Furthermore, with the rapid expansion of the company in different geographical regions and problems of driver management and customer searches connectivity, the company continuously faced problems of connectivity, integration, and scalability. Not only that but also the application upgrades and fixing bugs were slow and difficult handle.

Later Uber decided to follow the footsteps of Amazon, Netflix, SoundCloud and various other companies who implemented Microservice architecture as their service model. An API gateway was established which solved the problem for customer searches and driver connectivity. Also, this was applicable for their entire system, where each and every functionality of application was broken down to a microservice and the connectivity between each microservice remained intact with the help of REST API and the API gateway. The API gateway gets automatically connected to each and every microservice. This enabled developers to deploy quick upgrades and patches and this also enabled developers to deploy any changes to the desired microservice without affecting the other running microservices [10]. This led to rapid scalability and efficient management of each resource.

VI. CONCLUSION

The purpose of this article is to view Containers as an alternative solution to virtual machines as the advantages are comparatively greater than the drawbacks. Not only that but also it acts a reliable option and has nullified the drawbacks of a virtual machine. Containers in recent years have surpassed its expectations and have already solved so many problems that an organization faces. That being said, there are some limitations to this technology such as security, having a common Operating System used throughout the deployment and maintaining phase, having to face a challenge of running an embedded based application where virtual machine clearly is very capable at and lastly the requirement to run various different applications on various software platform where virtual machines have a clear edge over containers. Both technologies have their own specialty and excel on various fronts. Also, Microservice Architecture is the way to go into the future depending on the situation of an organization. Microservice Architecture helps DevOps to fully showcase their skillsets by providing a constant improvement to the system.

VII. REFERENCES

[1] Comparative Study of Virtual Machines and Containers for DevOps Developers Instructor: Prof. Richard Martin Sumit Maheshwari, Saurabh Deochake, Ridip De, Anish Grover, 2015.

[2] Performance comparison of a WebRTC server on Docker versus Virtual Machine: Cristian Constantin Spoiala, Alin Calinciuc, Corneliu Octavian Turcu, Constantin Filote, 2016.

[3] Docker vs LXC by Mike Baukes, 2017: https://www.upguard.com/articles/docker-vs-lxc

[4] Containers and Cloud: From LXC to Docker to Kubernetes by CLOUD TIDBITS, 2017.

[5] Comparing Containers versus Virtual Machines for Achieving High Availability Wubin Li and Ali Kanso, 2016.

[6] Medium source blog by Preethi Kasireddy: https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b, 2016.

[7] An Updated Performance Comparison of Virtual Machines and Linux Containers Wes Felter Alexandre Ferreira Ram Rajamony Juan Rubio, 2014.

[8] Scaling containers by David Linthicum Chief Cloud Strategy Officer, Deloitte Consulting, 2018. https://techbeacon.com/scaling-containers-essential-guide-container-clusters.

[9] A Systematic Mapping Study in Microservice Architecture by Nuha Alshuqayran, Nour Ali and Roger Evans, 2016.

[10] Microservices Architecture Enables DevOps: An Experience Report on Migration to a Cloud-Native Architecture by Armin Balalaie Abbas Heydarnoori, Pooyan Jamshidi, 2017.