International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Migration from Virtualization to Dockerization in Cloud

Rohit Yadav[1], Rashmi Gaurkar[2]

[1]Student, Institute of Computer Science, Mumbai Educational Trust- MET ICS, Mumbai, India

[2]Professor, Institute of Computer Science, Mumbai Educational Trust- MET ICS, Mumbai, India

Abstract: ***Virtualization is the core of cloud computing for providing services on demand. Distributed systems in cloud computing are mostly developed using virtualization. However, due to issues of significant resources, interpretability and deployment make it less adaptable to distributed systems. Dockerization has been introduced and is gaining popularity in the software development community. Docker has recently introduced its distributed system development tool called Swarm, which extends the Docker Container-based system development process on multiple hosts in multiple clouds. Docker Swarm-based containerized distributed system is a brand-new approach and needs to be compared with the virtualized distributed system. Therefore, this review paper describes the simulation and evaluation of the development of a distributed system using virtualization and dockerization. This simulation is based on Docker Swarm, VirtualBox, Ubuntu, Mac OS X, nginx, and redis. To simulate and evaluate the distributed system in the same environment, all Swarm Nodes and Virtual Machines are created using VirtualBox on the same Mac OS X host. For simulation and evaluation, nginx and redis ware installed different Docker Swarm nodes and Virtual machines. Finally, based on results, it evaluates their required resources and operational overheads.***

Keywords: Virtualization, Dockerization, Distributed Systems, Cloud, Virtual Machine Monitor, Hypervisor, Docker, Virtual Machine, Container

1. Introduction

Today's most people are living in the technology era, where everyone is utilizing computer resources and services. Getting on-demand computing resources and services via the internet is called Cloud Computing. Through Cloud Computing we can access many computer resources and services such as Storage, Networking, Infrastructure, and Security and one of them is Virtualization and Dockerization. Virtualization is a process that allows for more efficient utilization of physical computer hardware and is the foundation of cloud computing [3]. Virtualization is a technology to simulate a distributed environment similar to the real environment. Virtualization made system development more flexible and effective by dynamic allocation of system hardware. Virtualized environments require a long time, tedious installation of software, and configuration process with a highly skilled person. It also demands a wide range of technical knowledge such as operating system, database, application server, and network service. Virtualization manages everything successfully is the biggest issue. Which makes it less adaptable in many distributed systems.

Dockerization or Docker Container-based Virtualization has emerged as an alternative lightweight technology for the system development process and gaining popularity in the software development community [1]. Docker is a software platform that allows us to build, test and deploy applications quickly. In docker we can pack our application with its dependencies and can run any platform or operating system.

Docker packages software is standardized in the unit that is called "containers" [4]. Docker is an operating system for containers similar to virtualize (removes the need to directly manage) server hardware. Containers virtualize the operating system of a server. Docker is installed on each server and provides a simple single set of commands. you can use containers as build, start and stop. Docker ships more software faster, standardizes operation, seamlessly moves, and saves your money.

This review paper has described Virtualization and Dockerization, Virtualization is a well-established technology in the development of cloud computing-based distributed systems. Whereas Dockerization is an effective system development tool [1]. and why people are more preferring Dockerization instead of Virtualization.

2. Virtualization and Dockerization

A. Virtualization

Virtualization is a process that allows a user to create a simulation version of the device or resources, such as desktop, storage, network, application, server, and operating system. Virtualization is a combination of hardware and software engineering that creates Virtual Machines (VMs). It is an abstract computer hardware that provides a single machine that can act as many machines. It enables multiple operating systems to run on a single computer hardware platform. Generally, virtualization architecture has two main components one is Virtual Machine Monitor (VMM)/Hypervisor and another is Virtual Machine (VM) [1]. VMM/Hypervisor recreates a hardware environment in which guest operating systems are installed. Its works as a translation system between Virtual Machines and hardware infrastructures. There are two major types of VMM/Hypervisors: Type 1 and Type 2. Type 1 Hypervisor runs directly on top of the hardware. Therefore, they take the place of operating systems. It is also called a native virtual machine. Whereas, Type 2 hypervisor runs on top of the host operating system. And it is also called a hosted virtual machine. Every VM has its operating system, binaries, libraries, and application as shown in Fig. 1. Virtualization

requires bandwidth, storage, and processing capacities. A large server or desktop is needed if it is going to host multiple running Virtual Machines [1]. There are so many virtualization technologies available such as VMWare, VirtualBox, UNM, and Xen. In the rest of the paper, Virtualization and VM terminology represent the Type 2 Hypervisor category because this architecture is comparable to Dockerization architecture.

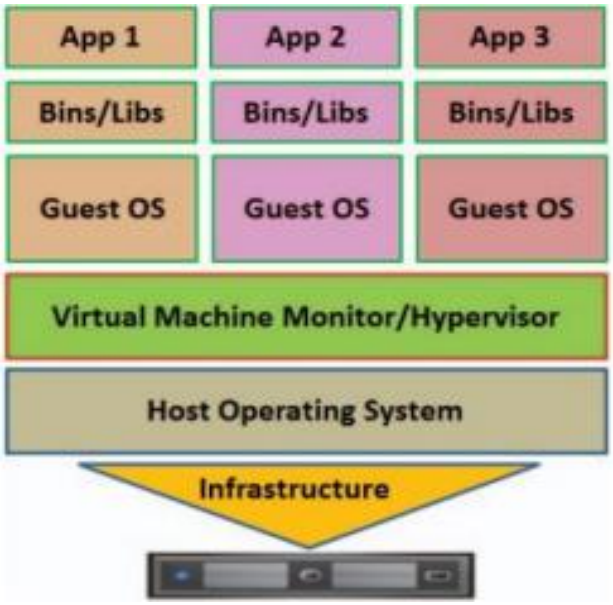

Figure 1: Virtualization and Virtual Machine

B. Dockerization

Containers provide an isolated environment akin to virtualization but without virtual hardware emulation [1]. Containers are executable units of software in which application code is packaged, along with its libraries and dependencies, in common ways so that it can be run anywhere, whether it be on desktop, any operating system, traditional IT, or the cloud [7]. Containers run on top of the user operating system's kernel. Therefore, it is also known as OS
level virtualization. Containers technology allows multiple user-space instances to be run on a single host machine [1]. Containers are classified into two parts: System containers and Application containers. A System Container is similar to a full OS and runs all processes such as init, inetd, sshd, Syslog, and cron. Whereas, Application Container only runs an application. Both types are useful in different circumstances [1]. Docker technology is an example of container virtualization and the process for creating container virtualization is called Dockerization.

Docker is an open-source containerization platform. It enables developers to package applications into container standardized executable components combining application source code with the libraries and dependencies required to run that code in any environment [6]. It is developed by Docker Inc. Docker provides isolated containers (see Fig. 2). On a single Docker, the Engine developer can create multiple containers without a guest operating system.

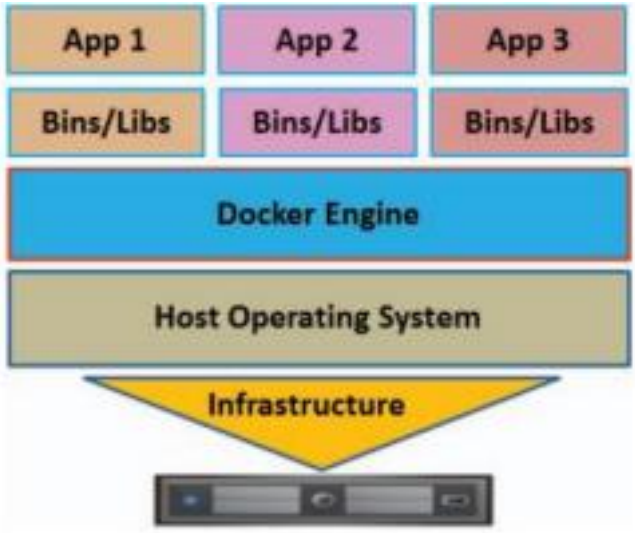

Figure 2: Dockerization in Docker Container

3. Dockerization on Non-Linux Machine

This section reveals the actual implementation of Docker Containers on a non-Linux system (eg: Mac OS X machine). And how it is different from Linux based implementation of Docker Containers. To create Docker containers in a non-Linux-based system requires Docker Host. Docker host is a lightweight Virtual Machine, which require fewer resources and operational overheads. In non-Linux systems, the Docker Engine runs inside a Linux Virtual Machine called "default". The "default" is a lightweight Linux Virtual Machine made specifically to run the Docker engine on a non-Linux machine (e.g., Mac OS X machine). The great success of Docker is this small Virtual Machine, which runs completely in RAM and loads only in a few seconds. This is largely due to its very small size (i.e., 35 MB in this implementation)[1]. the detailed architecture is shown in Fig. 3.

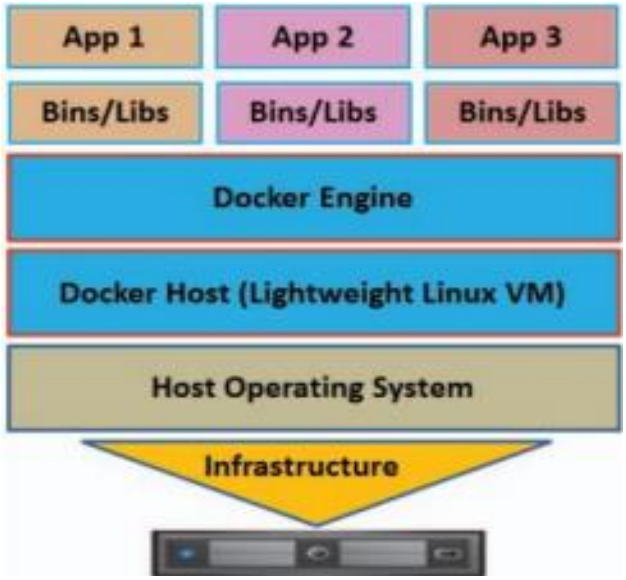

Figure 3: Dockerization for a non-Linux machine (Mac OS X Machine)

4. Experimental Simulation: Development of a Distributed System Using Virtualization and Dockerization

A. Experimental Simulation: Development of a distributed system using Dockerization

This simulation demonstrates the development of a distributed system using dockerization. In this, a simple distributed system is developed by Docker Swarm. nginx and redis. This distributed system is implemented as a cluster of four Docker Swarm nodes. These Swarm nodes are created on the same host computer to measure the performance in the same environment. However, this simulation can be implemented and extended to develop the distributed system on Docker-supported multiple clouds such as Amazon Web Services, Microsoft Azure, Digital Ocean, Google Compute Platform [1]. The complete system development process is carried out on Mac OS X, therefore, it uses the non-Linux architecture of dockerization technology. Initially, Docker is running in the lightweight Virtual Machine called "default".

For creating a Docker Swarm cluster of nodes, steps are mentioned. In this simulation of the distributed system using dockerization, Oracle VirtualBox driver is used. Any other Docker-supported driver can be used. Such as Amazon Web Services, Microsoft Azure, Google Compute Engine, Digital Ocean, OpenStack, IBM Softlayer, VMware vCloud, VMware Fusion, VMware vSphere [1].

- First step is creating a Docker Swarm image. Which can be used as the discovery token to connect all nodes within Swarm clusters.
- Use Swarm cluster image token to create Swarm nodes including Swarm master node in VirtualBox.
- After creating Swarm nodes, these nodes can be managed by any cloud just changing the driver name to desired cloud name.
- Once all Swarm nodes are created and running, they can be seen in VirtualBox Manager.
- Next step is to set the environment variable for the Swarm cluster up and running.

Making Swarm cluster as a working distributed system, required certain servers to be installed such as *nginx* and *redis*. In this implementation. Here, two servers nginx and redis are started as two separate Containers on each Swarm node. This typical configuration is just to demonstrate the scheduling and scaling features of Swarm, however, in a real system, only one server can be used. Firstly, nginx servers are started on all the Swarm nodes. All these nginx Containers are automatically allocated by Swarm-agent to different Swarm nodes. Finally, redis servers are started on all the Swarm nodes. Again, all these redis Containers are automatically allocated by Swarm-agent to different Swarm nodes. Now the list of all the running Containers with nginx and redis servers on different Swarm nodes. This Docker Swarm-based distributed system can be used to develop the distributed application in the cloud environment.

B. Experimental Simulation: Development of a Distributed System using Virtualization

This simulation is similar to the Dockerization based simulation with an almost similar environment for making a comparison between the two technologies. Here, a similar distributed system is developed using VirtualBox, Ubuntu, Nginx, and Redis. This distributed system is implemented as a cluster of four Virtual Machines on the same host computer to measure the performance of virtualization and Dockerization in the same environment. However, this simulation can be implemented and extended to develop the distributed system on multiple clouds in the same way. The complete system development process is carried out on Mac OS X, and its experimental architecture uses virtualization technology [1].

Creating Virtual Machines in VirtualBox is very easy and well known, once Virtual Machines is created it can be seen in the VirtualBox manager panel.

Making these virtual Machines work distributed systems requires certain servers to be installed such as nginx and redis in this implementation.

The next step is to install the nginx server with "sudo apt-get install nginx" and Redis with "sudo apt-get install redis" on Linux-based systems on virtual machines.

Within this Virtual Machine-based distributed system, cooperation and application development would be similar to a stand-alone machine-based activity and one Virtual Machine can be promoted to the master or domain controller to coordinate all other machines.

5. Experimental Results and Evaluation of Virtualization and Dockerization Technologies for the Development of a Distributed System

Designing distributed systems in the cloud using virtualization and dockerization is a wide topic and includes several parameters such as resource and operational overheads, scalability, portability, interoperability, isolation, inter-communication, and security [1]. This review paper mainly focuses on the explanation of system performance and operational overheads of Docker Swarm Node-based and Virtual Machine-based distributed systems. After implementation of a distributed system by virtualization and dockerization, overall results illustrate that dockerization consumes fewer resources and operational overheads as compared to virtualization. The dockerization based distributed system utilizes less memory, CPU usage, boot time, and energy within the same environment.

Memory is one of the most important metrics for the measurement of the performance of virtualization and dockerization. Docker Swarm Node has better memory usages as compared to Virtual Machine. Storage is another crucial metric to determine the performance of virtualization and dockerization. A similar result is obtained for this metric, and the Docker Swarm Node is very compact requiring less storage. The main reason for higher storage overheads in the virtualization-based development process is due to the large size of the base image because it is a full OS image. Whereas, Docker Swarm Node needs a very small size of the base image as it comes with minimum features. This large image size in virtualization causes the problem of large system/application

images which is difficult to implement or deploy [1]. This is one of the reasons for the popularity of the compact Docker Swarm Node among the software community.

Another decisive performance metric is CPU usage and time; again, the Docker Swarm Node has achieved better performance results over the Virtual Machine. Therefore, Docker Container is beneficial for CPU-intensive applications. However, Docker Container is a locked-down environment with no direct access to many kernel resources; and it contains multiple abstraction layers. Consequently, large I/O and network usage may consume more resources because of Docker-related abstraction layers demanding significant context switching between kernel and userspace [1]. Similarly, depending on the nature of an application, virtualization can be beneficial or expensive. Initial boot time of virtualization can also be reduced automatically if the lightweight version of Linux is used in the distributed system.

This particular evaluation outlines the requirement of fewer resources and overheads for the Docker Swarm-based distributed system as compared to Virtual Machine-based distributed system. However, it cannot be taken as general guidance for all kinds of systems. Therefore, the use of virtualization and dockerization can be determined precisely depending on the need of a specific system. Despite Docker Swarm having demonstrated better performance, its Containers and images can be run only under dockerized GNU/Linux, and an application has to be Docker locked-in. Additionally, it is supported by a few cloud computing corporations. Finally, dockerization technology is mainly focused on PaaS (Platform-as-a-Service), and its prime objective is the deployment of distributed systems (software/applications) with portability and interoperability while utilizing operating systems (OS) virtualization principles [1]. Virtualization technology is mainly focused on IaaS (Infrastructure-as-a-Service), and its prime objective is the deployment of distributed systems (hardware provisioning/allocation and management) while utilizing hardware virtualization principles [1].

6. Conclusion

This review paper describes the simulation and evaluation of the development of a distributed system using dockerization and virtualization. It was based on Docker Swarm, VirtualBox, Ubuntu, Mac OS X, nginx, and redis. To simulate and evaluate the distributed system all the Swarm nodes and Virtual Machines were built using VirtualBox on the same host system. And install the nginx and redis server in Docker Swarm nodes and Virtual Machines.

The overall result shows that the Docker Swarm-based distributed system consumed fewer resources and operational overheads as compared to Virtual Machine based distributed systems.

References

[1] Naik, Nitin. “Migrating from Virtualization to Dockerization in the Cloud: Simulation and Evaluation of Distributed Systems.” *2016 IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA)* (2016): 1-8.
[2] N. Naik, "Performance Evaluation of Distributed Systems in Multiple Clouds using Docker Swarm," 2021 IEEE International Systems Conference (SysCon), 2021, pp. 1-6
[3] Virtualization https://www.ibm.com/cloud/learn/virtualization-a complete-guide
[4] Docker https://aws.amazon.com/docker/
[5] Virtualization Solution https://www.vmware.com/in/solutions/virtualization.html
[6] Docker https://www.ibm.com/cloud/learn/docker
[7] Containers https://www.ibm.com/cloud/learn/containers