A Comparative Study of using Virtual Machine and Docker Container for deploying a ReactJS web application

Rohith Raj S[1], Dr. Badari Nath K[2]

[1]Student, Dept. of Computer Science and Engineering, RV College of Engineering, Karnataka, India
[2]Assistant Professor, Dept. of Computer Science and Engineering, RV College of Engineering, Karnataka, India

***

Abstract – *In the recent past, there has been a surge in the usage of container technology compared to that of virtual machines. This is primarily due to the fact that container methodology enables quicker and easier deployment of applications. However, there still lacks an in-depth and systematic study in the effectiveness of Docker container over virtual machine to deploy a ReactJS web application. In this paper, a detailed comparative analysis is carried out between virtual machine and Docker container in deploying a ReactJS web application, by considering a wide range of parameters. The results show that Docker container is the best practice compared to that of virtual machine. The work in this paper can help developers to make informed decisions in deploying their ReactJS web application, so as to achieve better performance.*

Key Words: Containerization, Docker, Docker Containers, ReactJS, Node.js, Virtual Machines, Virtualization

1. INTRODUCTION

With the development of inumerous applications, there comes a need to deploy multiple, isolated applications on a single platform. The two primary ways of achieving this are: Virtualization and Containerization.

Virtualization allows running multiple heterogeneous operating systems on the hardware of a single physical server by easily sharing the CPU and memory resources [1]. The widely used software for system virtualization is called the Hypervisor aka Virtual Machine Manager (VMM). Hypervisor is responsible for allocating resources to each Virtual Machine (VM) created and also handling the deletion of a VM. Some of the well known categories of virtualization are [2]:

- Desktop Virtualization
- Network Virtualization
- Server Virtualization
- Software Virtualization
- Storage Virtualization

Whereas, Containerization makes it possible to deploy multiple heterogeneous applications using the same operating system on a single server. This is achieved by encapsulating an application and all its dependencies within its own environment thus, allowing them to run in isolation while using the same system resources. Containerization also allows for a much quicker and lightweight deployment of applications due to the fact that the resources are not being wasted on running separate operating systems [5].

In this paper, the trade-offs of using virtual machine and Docker container for the deployment of a ReactJS Web application have been discussed. And also, an attempt has been made to understand the logic as to, how the deployment time is reduced from days to minutes with Docker containers as opposed to virtual machines.

2. BACKGROUND

This section compares virtual machine and container technology along with its system architecture. It also throws light on how container technology is becoming increasingly popular.

Virtual machine is an application environment which emulates a particular computer hardware. Whereas, a container is similar to an application running as an isolated process on top of an operating system (OS) in a designated address space. Docker is the most popular platform for containerization and hence, container and Docker container are used interchangeably.

Running different containers on one physical machine is enabled by Cgroups and Namespaces feature. Cgroups aka Control groups, is a Linux kernel mechanism primarily used by system administrators for controlling allocation of resources, such as CPU, memory, network, or any other combination of them, for the running containers. And, none of the containers can use more resources, than what's being specified in the Cgroups. Namespace is used to provide abstraction of the Linux kernel resources, such as host names, network interfaces, PIDs, so that it appears like each container has its own isolated resources [3].

Fig. 1 shows how the Virtual machine architecture is different from that of a Container. Due to the notable differences between VM and Container as shown in Table I, researchers and DevOps developers are paying special attention to the container technology especially Docker.

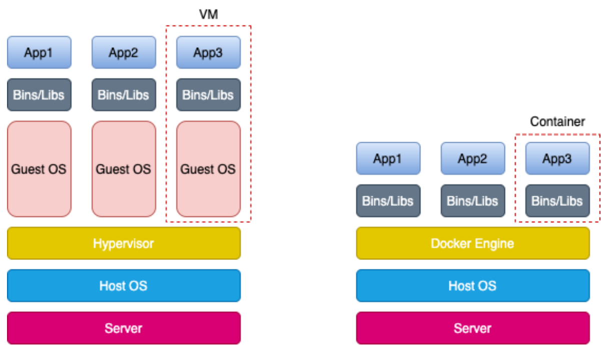


Fig -1: Virtual machine vs. Container architecture comparison

Table -1: Virtual Machine vs. Container

	Virtual Machine	*Container*
1	Hardware level virtualization	OS level virtualization
2	Heavyweight	Lightweight
3	Contains full set of OS, executables and dependencies	Contains only executables and dependencies
4	Each runs its own OS	All containers share the host OS
5	Requires more memory space	Requires less memory space
6	Size is in several GBs	Size is within tens of MBs
7	Fully isolated hence more secure	Only process level isolation hence less secure
8	Not cost-effective	Cost effective

3. ADVANTAGES OF DOCKER CONTAINERS

In the recent past, Docker has become quite popular due to the amazing benefits which it provides to DevOps developers. Some of the main advantages of Docker are as follows:

A. Easy Deployment

Docker containers can run on any Linux machine and also be deployed on desktops, physical servers, on premise datacenters, cloud environments and so on. Thus, Docker containers can be quickly and seamlessly moved from a cloud environment to desktop and back to physical server.

B. Density

As containers don't use hypervisor and a full operating system, resources are judiciously utilized with low requirements. Thus, many Docker containers can be deployed on a single host.

C. Portability

Containers are extremely portable, as the applications can be encapsulated into a single unit and deployed in various production environments without any changes to the container.

D. Rapid build times

As Docker containers are extremely small in size; they have short build times. So, this reduces the development, testing and deployment time.

E. Scalability

Docker containers can be scaled up from one to thousand's and scaled down when not needed. Thus, according to the user needs, the scale can be adjusted.

4. EXPERIMENT

This section provides the steps that we followed to measure the effectiveness of Docker containers over Virtual machines to deploy a ReactJS application. The application source code is housed in a remote Git repository. And, the test environment considered is a Node.js server for serving a simple ReactJS application with the system configuration as follows:

A. Virtual Machine

First, the application is being tested with a virtual machine using Ubuntu Linux distribution. The system specification that we used for studying the deployment of the ReactJS application through virtual machine is:

- Operating System: Ubuntu 16.04.6 LTS
- Node.js Server: v12.14.1
- Hypervisor: Oracle VM VirtualBox Type-2 Hypervisor

B. Container

Having deployed the ReactJS application through virtual machine, it is now tested by deploying it using Docker container. The system specification that we used for studying the deployment of the ReactJS application through Docker container is:

- Base Docker Image: node:12.14.1
- Docker: 19.03.8

Fig. 2 shows the steps to be followed to deploy a ReactJS application through virtual machine. Firstly, we need to install Node.js and npm on the virtual machine. This is primarily done because, Node.js provides us with thousands of npm modules which can be directly used in our application without reinventing the wheel. Now, the next step is to clone the remote Git repository onto our virtual machine. Having done that, the dependencies mentioned in package.json file is installed with the npm install command. Finally, we can run an instance of the application with npm start command.

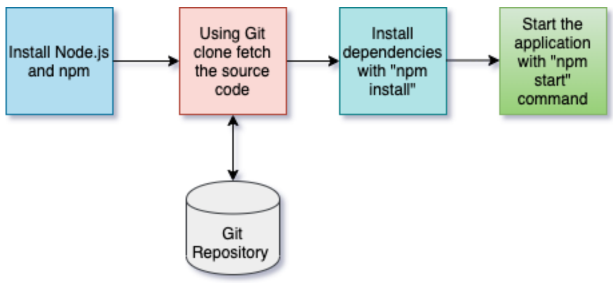

Install Node.js and npm
Using Git clone fetch the source code
Install dependencies with "npm install"
Start the application with "npm start" command
Git Repository

Fig -2: Process flow to deploy ReactJS application through VM

As shown in Fig. 3, to containerize an application we have to first install Docker on our machine. The next step is to clone the remote Git repository onto our machine running the Docker daemon. Finally, we define a Dockerfile and build it, to generate a Docker Image. Fig. 4 shows the Dockerfile for the ReactJS web application under consideration. Now, we can run a Docker Container which is a running instance of the built Docker Image [6].

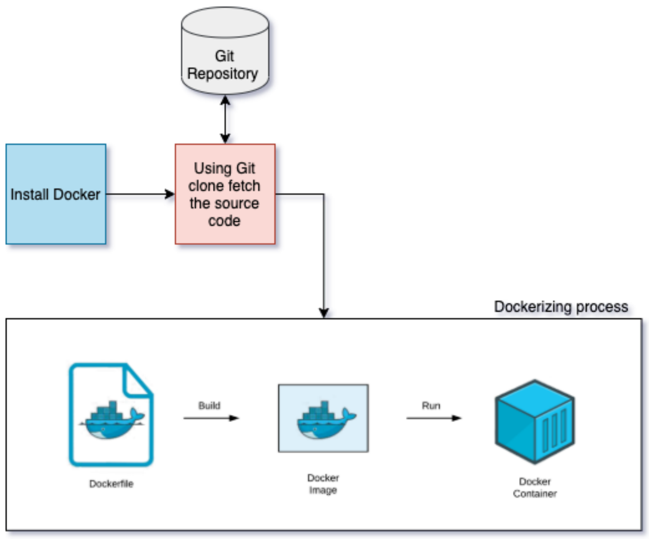
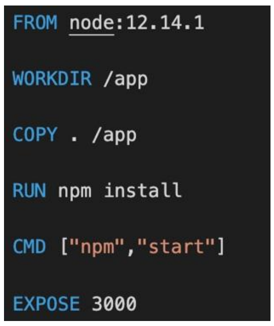


Fig -3: Process flow to deploy ReactJS application through Docker container



```
FROM node:12.14.1

WORKDIR /app

COPY . /app

RUN npm install

CMD ["npm","start"]

EXPOSE 3000
```

Fig -4: Dockerfile

We have tested virtual machine and Docker containers with wide range of parameters like time required to start/stop a particular service, image size, CPU performance and memory utilization. Finally, we throw light on the best practice to be used for deploying, between Virtual machine and Docker container.

5. RESULTS

We have tested a virtual machine and a Docker container, serving a ReactJS web application, with wide range of parameters as follows:

A. Start and Stop time

In this, the time required to start and stop a virtual machine and Docker container is analyzed. To carry out the comparison test, it's assumed that we have Ubuntu operating system already installed inside the virtual machine and a prebuilt Docker Image with Node.js installed. As shown in test result Fig. 5, the time taken by virtual machine is around 55 sec to start and 30 sec to stop as it runs as a standalone operating system and has to undergo all the generic operating system boot process. Whereas, we know that Docker Image is a file, made up of multiple layers, and used for executing the code in a Docker container [4]. Hence, the time taken by Docker container is very less and around 45 ms to start and 32 ms to stop on an average, as compared to that of virtual machine. Therefore, we observed that Docker container technology is a better choice instead of virtual machines, primarily because it saves a lot of human effort in bringing up the system or stopping it.

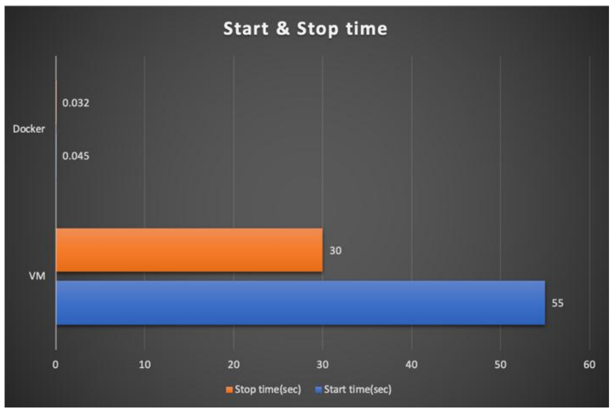


Fig -5: Benchmarking of Start and Stop time (lower the better)

B. Image Size

We know that, virtual machine images are stored as vmdk files. These files are generally large in size, as they contain the entire information of the virtual machine. Whereas, Docker images use Union file system (aka UnionFS) as the building blocks for containers which operate by creating layers. Thus, Docker images are lightweight and consume less storage space. As shown in Fig. 6, Docker image for the ReactJS web application under consideration is 85% lighter than virtual machine image. Hence, it can be concluded that

Docker container technology is the best choice in terms of Image size.

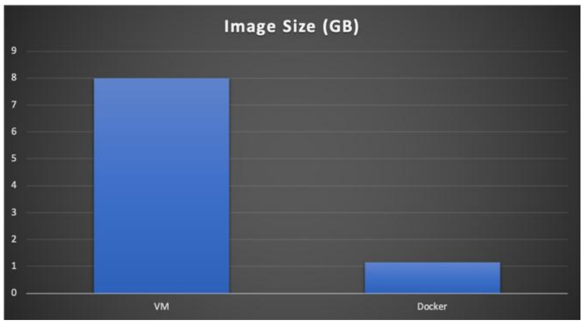


Fig -6: Benchmarking of Image size (lower the better)

C. CPU Performance

In this, the CPU performance of virtual machine and Docker container is analyzed in terms of Floating Point Operations Per Second (FLOPS). We achieved this using Intel's Linpack tool. With our tests, we found out that Docker container again outperforms virtual machine as shown in Fig. 7. This is primarily because, Docker container uses native system calls of host operating system, whereas, virtual machine uses hypervisor which translates the system call to hardware, and hence, performs poorly.

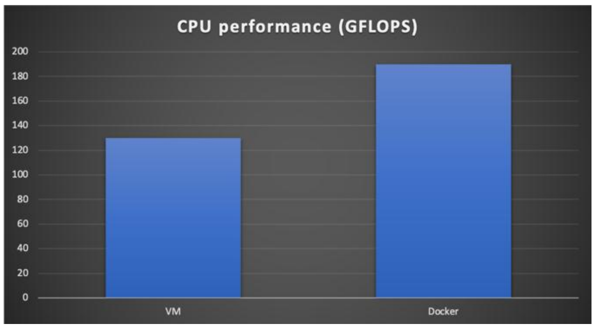


Fig -7: Benchmarking of CPU performance (higher the better)

D. Memory Utilization

Monitoring memory usage is crucial for any application running on a machine. We have monitored the memory utilization of both the Docker container and virtual machine serving the same ReactJS web application using Glances tool. Glances is a cross-platform monitoring tool which aims to present a large amount of monitoring information through a console or Web based interface. The information dynamically adapts depending on the size of the user interface [7]. As shown in our study in Fig. 8, virtual machine has high memory requirements compared to that of Docker container. This is mainly because, virtual machine runs on top of a guest operating system, which consumes a reasonable amount of memory to keep itself running. Hence, it can be concluded that Docker container technology is the best choice in terms of Memory utilization.

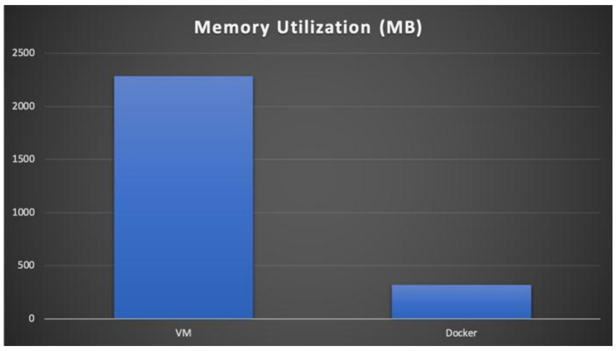


Fig -8: Benchmarking of Memory utilization (lower the better)

6. DRAWBACKS OF DOCKER CONTAINERS

Some of the major drawbacks of docker containers are as follows:

- Only 64-bit machines support Docker.
- As all containers share the host operating system, a different host machine is required for Docker containers based on a different operating system.
- A security vulnerability in the operating system kernel poses a threat to all Docker containers running on the host machine.

7. CONCLUSION

The concepts of virtualization and containerization is explained in this paper, using virtual machines and containers. And, the comparative study shows that Docker container uses lesser system resources and provide just enough isolation. Therefore, overall performance is much better compared to that of virtual machine. However, if the web application needs to be highly secure and isolated, virtual machine would be a better choice. Hence, it can be concluded that Docker container is the best practice to deploy a ReactJS web application if and only if, the application requires just enough isolation with lesser system resource utilization.

REFERENCES

[1] Kumar, A., Sathasivam, C., and Periyasamy, P. (2016). "Virtual Machine Placement in Cloud Computing" in Indian Journal Of Science And Technology, 9(29). doi:10.17485/ijst/2016/v9i29/79768

[2] Prakash and Raghavi Suresh, "Comparative Analysis on Docker and Virtual Machine in Cloud Computing" in International Journal of Pure and Applied Mathematics, vol. 117, no. 7, pp.175–184, 2017.

[3] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu and W. Zhou, "A Comparative Study of Containers and Virtual Machines in Big Data Environment," 2018 IEEE 11th International Conference on Cloud Computing

(CLOUD), San Francisco, CA, 2018, pp. 178-185.

[4] Babak Bashari Rad, Harrison John Batti, and Mohammad Ahmadi, "An Introduction to Docker and Analysis of its Performance" in IJCSNS International Journal of Computer Science and Network Security, vol. 17, no. 3, March 2017.

[5] Containerization vs. Virtualization: What's the Difference?, www.burwood.com/blog-archive/containerization-vs-virtualization

[6] Docker Documentation, https://docs.docker.com/

[7] Glances – An eye on your system, https://glances.readthedocs.io/en/stable/