Replacing Virtual Machine with Docker to Build or Deploy Application

Roshan[1], Dr. Ashwini K.B[2]
Student[1], Associate Professor[2]
Master of Computer Applications,
RV College of Engineering, Bangalore, India.

Abstract
There are numerous approaches to encourage the creation of large-Scale Projects. One of the most frequently utilized strategies is to make use of virtual machines (VM) that contain the program environment. VM exist as complete standalone environments. VM utilize its own BIOS, CPU, RAM, and a complete OS. The VM regularly need to deal with virtualization problems such as network congestion, VM sprawl and server hardware failures which reduce VM performance. VM are replaced by Docker to make this process considerably simpler. Docker is a tool that uses OS-level virtualization to deliver software in packages. These packages are known as containers. Docker allows creating images that contain source code, libraries, dependencies, tools, and other files needed for an application to run. Instead of abstracting the hardware, Docker containers abstract the OS. Every Docker container shares the exact same OS by reducing the overhead to the host system. In terms of resource Docker are more lightweights than VMs. Containers share operating systems consuming only fraction of the resources as compared to VM. Docker uses the Docker engine which have the ability to sustain server application instances five times the amount of VM.

Keywords Operating System(OS), Central Processing Unit(CPU),Basic Input/output System(BIOS),Network Interface Card(NIC), Random-access memory(RAM), Virtual Machine (VM) , Docker Data Center(DDC),Docker Trusted Registry(DTR) .

I. Introduction

Docker is a platform to develop, run and deploy applications. Docker is a containerization platform that packages applications with all its dependencies needed to run those applications in the form of containers. This ensures that the application will work in any environment. Once the container is deployed it is kept running on top of the operating system's kernel. Docker can also coordinate with outsider instruments, which allows it to easily deploy and manage Docker containers.
Each application will have its own set of libraries and dependencies. This also ensures that there is process level isolation, so that each application is independent of other applications, allowing developers to build applications that will not interfere with one another. The main components of Docker are images, containers and registry. Docker image is a read-only template. For example, an image might contain an operating system with installed applications. Docker containers can be created using these images. Docker makes it easy to create a new image and update existing ones or download ready-only images. Docker-registry is used to store those images. One can upload images to registry or download images from registry. Containers contain everything that is needed to run an application. Each container is created from an image, which forms a secure and isolated platform for the application.

II. Literature Survey

Fog computing platforms are described in [1] which offers virtualized resources located in the vicinity of their end users and how Docker-based systems which locally cache a copy of every container images are executed. Paper [2] describes how Docker containers have become a prominent solution for deploying modern applications. Paper [3] gives a solution to solve the problems of low resource utilization and complex deployment in traditional dispatcher simulation training systems, by using a Docker-based simulation training system.

A description is given in [4] about the status of Distributed computing and its technology which contributes for new stack of computing placed on virtualization of assets and how Dockercan be used in a distributed environment. The importance of Docker, an open source application containerization technology for building, shipping, and running applications and the behavior of Docker swarm under DoS/DDoS is effectively presented in [5]. Cloud computing is a successful and speedy evolving model with new features and capabilities which is being announced regularly. The contribution of Docker to cloud environments is explained in paper[6]. The popularity of Docker because of its user friendly platform for running and managing Linux containers. Is proven by the fact that the vast majority of containerized tools are packaged as Docker images which is highlighted in paper [7].

III. Working of Docker

Docker uses client-server architecture. The Docker client communicates with Docker daemon, which takes care of building, running, and distributing the Docker containers. The Docker client and daemon have the ability to run on the same system, or it can also connect Docker client to a Docker daemon which is running on a remote location. To enable communication between Docker client and daemon it uses REST API on top of UNIX sockets or network interface.

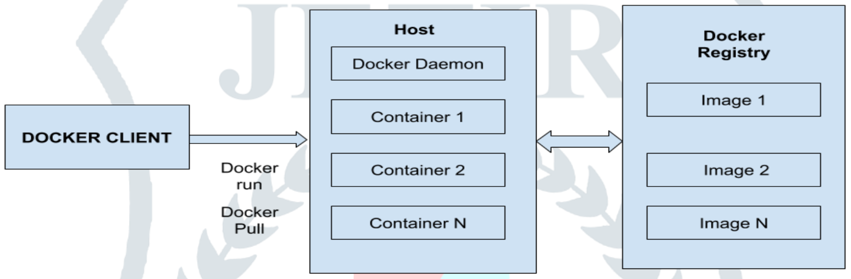

Fig 1: Working of Docker

a) Docker Demon
The Docker daemon uses Docker API requests to manage Docker objects. Docker objects may be containers, images, networks, and volumes. To manage Docker services the Docker daemon communicates with other Docker daemon.
b) Docker Client
Users interact with Docker using Docker client. Docker commands such as dockerrun, are sent to dockerd. The Docker command makes use of Docker API. The Docker client can also communicate with more than one daemon.
c) Docker registry
Docker registry is used to store Docker images. Docker Hub is a public registry that a can be used to store the images. Docker is configured in such a way that by default it looks for images on Docker Hub. Private registry is also available making use of DDC and DTR.
d) Docker image
An image is a template for creating a Docker container. An image is a read-only template. Docker has the ability to create its own images or make use of images that are created by others. To build an image, create a Dockerfile .When we compare to other virtualization technologies the images are so lightweight, small, and fast.
e) Docker Container
An instance of an image is known as Docker Container. Using Docker API or CLI users have the ability to create, start, stop, move, or delete a container. It is also possible to connect a container to one or more networks or attach storage. A new image can be created based on its current state. By default, a container is isolated from other containers.

IV. Docker vs Virtual Machine

Virtual machines are software computers that provide the same functionality as physical computers. They are multiple guest operating systems based on hardware, like physical computers, they also run applications and an operating system. In simple terms, virtual machines are nothing but computer files that run on a physical computer and provide the same functionality as a physical computer.

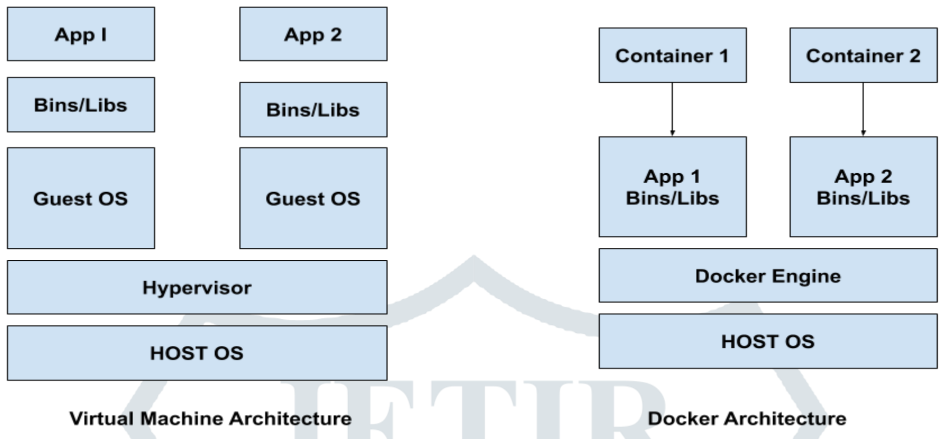

Fig 2: Virtual machine Architecture vs Docker Architecture

Virtual Machine	Docker
Heavyweight	Lightweight
Limited performance	Native performance
Each VM runs in its own OS	All containers share the host OS
Hardware-level virtualization	OS virtualization
Startup time in minutes	Startup time in milliseconds
Fully isolated and hence more secure	Process-level isolation, possibly less secure
Allocates required memory	Requires less memory space
Security is high as VM does not share operating System	Security is low as Docker shares resources, an attacker can exploit all the containers if the attacker gets access to even one container.

V. Process Vs Container Vs VM

Instance of a program running in a computer is known as process. A process is started when a program is either by a user entering a shell command or by another program like a task .A process is a running program with a set of data. A process can also initiate a sub-process, which is a called a child process.

	Process	Container	VM
Definition	A representation of a running program.	Isolated group of processes managed by a shared kernel	A full OS that shares host hardware via hypervisor
Use Case	Abstraction to store sate about a running process	Creates isolated environments to run many apps	Creates isolated environments to run many apps
Types of OS	Same OS and distro as host	Same Kernel, but different distribution	Multiple independent operating System
OS isolation	Memory space and user privileges	Namespaces and cgroups.	Full OS isolation
Size	Whatever users application uses	Images measured in MB +users application	Images measured in GB+users application
Lifecycle	Created by forking, can be long or short lived, more often short	Runs directly on kernel with no boot process, often is short lived.	Has a boot process and is typically long lived

VI. Advantages of using Docker to create or deploy application than VM

In virtual machines Hypervisor lies between host and guest operating systems. It is a virtual platform that handles more than one operating system on the server. It works between the operating system and CPU. Whereas Docker uses Docker Engine that adds up an extra layer between host operating systems where the applications are virtualized and executed.
In the case of virtual machines, each virtual machine has its own full operating system, so when running applications built into VM, memory usage can be higher than necessary and VM can make use of resources needed by the host. But containers share an operating system environment known as kernel so they use fewer resources than full VMs and reduce pressure on the host's memory.
VMs utilize a lot of disk space as they contain a full operating system and associated application the VM is hosting. Containers are light weight as they contain only those libraries that needed to run the containerized application so they are more compact than VMs and can start more quickly than virtual machines.
When it comes to updating or patching the operating system, VMs must be updated one-by-one each guest OS must be patched separately. With containers, only the operating system of the machine hosting the containers must be updated. This simplifies maintenance significantly.

VII. Docker benefit in IT businesses.

i) Continuous Integration, Deployment and Testing
In DevOps-driven organizations, containers simplify the processes of CI/CD pipeline. Containers operate as reliable infrastructure environment such that developers don't need to perform complicated configuration tasks.
ii) Software Quality and Compliance
Collaboration between Devs and Ops becomes more transparent in delivering operating chunks of the application that leads to better software quality and allows application to develop faster and deploy by improving compliance.
iii)Workload Portability
IT workloads can shift between different infrastructure instances and virtual environments without major configuration changes or rework on the application code.
iv)Cost Optimization
Containers maximize resource utilization as it has its own isolated virtualized environments. Allowing organizations to plan for infrastructure capacity.
v) Infrastructure Agnostic
Containers make the app components infrastructure skeptical, allowing organizations to move workloads between simple metal servers to virtualized environments to cloud infrastructure as per as changes of business needs.

VIII. Conclusion

The paper gives a clear comparison between VM and Docker. The ability of Docker to solve the IT personnel and businesses problems. It identified the challenges faced to deploy server environments using VMs and how Docker can solve the problems. The usage of Docker to create images that contain source code, libraries, dependencies, tools, and other files needed for an application to run and finally deploy server environments to improve the performance is also mentioned.

References

[1] Arif Ahmed ,Guillaume Pierre: Docker Image Sharing in Distributed Fog Infrastructures,2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)
[2] Nannan Zhao , Vasily Tarasov , Ali Anwar , Lukas Rupprecht, DimitriosSkourtis,Amit Warke Slimmer :Weight Loss Secrets for Docker Registries,2019 IEEE 12th International Conference on Cloud Computing (CLOUD)
[3] Qun Ma , Jiabing Han , Zhengqing Xu , Xuanhuai Yang ,Qunshan Li :Docker-based Simulation Training System on Dispatching and Control Cloud 2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)
[4] Nikhil Marathe,Ankita Gandhi , Jaimeel M Shah : Docker Swarm and Kubernetes in Cloud Computing Environment 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)
[5] Gaurav Bhatia , Arjun Choudhary, KratiDadheech :Behavioral Analysis of Docker Swarm Under DoS/ DDoS Attack 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)
[6] ArshModak ,S. D. Chaudhary , P. S. Paygude, S. R. Ldate :Techniques to Secure Data on Cloud: Docker Swarm or Kubernetes? 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)
[7] Abdulrahman Azab: Enabling Docker Containers for High-Performance and Many-Task Computing 2017 IEEE International Conference on Cloud Engineering (IC2E)
[8] Docker: Enterprise Container Platform — URL: www.docker.com
[9] Docker Hub — URL: https://hub.docker.com/