Docker &It's Containerization: Popular Evolving Technology And Rise Of Micro Services

[1]Puli ramya,[2]Sarakula devanam priya
[1]M.Tech Student,[2]Associate Professor
Department of computer science and engineering
Kakinada Institute Of Engineering And Technology For WomenA.P, India.

Abstract:

Traditional software development processes usually result in relatively large teams working on a single, monolithic deployment artifact. It is evident that the application is going to grow in size with an increase in the number of services offered. This might become overwhelming for developers to build and maintain the application codebase and there is a problem that sometimes the application works on the developer system and the same does not work on the testing environment so for this, we tried to work with virtual machines before but unless they have a very powerful and expensive infrastructure.VM supports hardware virtualization. That feels like it is a physical machine in which you can boot any OS. In hypervisor-based virtualization, the virtual machine is not a complete operating system instance but its partial instance of the operating system and hypervisor allows multiple operating systems to share a single hardware host. In this virtualization, every virtual machine (VM) needs a complete operating-system installation including a kernel which makes it massive. The proposed system highlights the role of Container-based virtualization and Docker in shaping the future of Microservice Architecture. Docker is an open-source platform that can be used for building, distributing, and running applications in a portable, lightweight runtime and packaging tool, known as Docker Engine. It also provides Docker Hub, which is a cloud service for sharing applications. Costs can be reduced by replacing the traditional virtual machines with docker containers. Microservices and containers are the modern way of building large, independent, and manageable applications. The adoption of containers will continue to grow and the majority of Microservice applications will be built on the containers in the future.

Keywords: Docker, Docker Engine, Docker Hub, Virtualization, Microservices.

I. Introduction

In the history of computing, Containers have unique recognition because of its importance in the virtualization of infrastructure. Unlike traditional Hypervisor virtualization where one or more independent machines run virtually on physical hardware via an intermediate layer, containers run the userspace on top of the operating system kernel. Containers provide isolation between multiple user workspace instances. Because of this unique feature container virtualization is often referred to as operating system-level virtualization. Instead of starting a complete operating system on the host operating system containers shares the kernel with the operating system which eliminates the overheads and it also provides isolation between the applications. These features of the containers make it possible to ship the small container which acts as a complete operating system that encapsulates only those files which are needed to run the desired applications. Docker was born in 2013. It is an open container project based on the Go language. The mainstream Linux operating system all supports Docker. The core idea of Docker technology is to realize "Build, Ship and Run Any App, Anywhere", that is, to manage the application life cycle in stages, to achieve the purpose of once packaging and running everywhere[1]. Build refers to the Docker container technology through instructions to all programs, dependencies, and system environment packaged into a runnable application package. The ship is when packaged applications and environments can be shipped to the host through mirroring; Run Anywhere means that images generate containers on any host to serve users[2]. A container that transports goods at a terminal can load any goods into the container without knowing what goods are inside the container. The container can transport any goods to the corresponding terminal. Different from the traditional system development mode, docker-based container development, test, and operation mode save a lot of repetitive labor in environment configuration and system deployment, which greatly improves the work efficiency at each stage. A private container image library is built through the Docker Registry. The specific deep learning framework is built by using images according to the requirements, and multiple versions of images are provided. All the frameworks can be built automatically using DockerFile[3]. Therefore, through the containerization of micro-software services, application image packaging update, automatic operation, and deployment can reduce the repeated labor in the process of operation and deployment, operation and maintenance services, reduce the cost of the 2 human update, improve the operation and maintenance time efficiency, to achieve the real automation from development test to deployment process. There are a lot of technologies at present in the current era, all those need different programming languages and different environments to run. Some of them are platform-dependent and some are not. But they need a base to run with an environment, This probably leads to occupy more storage area and different software applications need to be installed to deploy the project. To handle all those hurdles by providing consistency while deploying the code into the project we have a solution that Creating Image Files for the required code by using Docker Engine and pushing them into the Docker hub. The idea of Docker is to package up into a box(container) and this would be shipped to somewhere. We need not worry about where and how it has been shipped. This is the technology mostly used for re-using the code where they are needed. It’s a software containerization platform where it provides an abstraction of Operating-System level virtualization. The Container-based virtualization uses a different approach, here standard host operating system is at the base and can be a Windows or Linux host when using Parallels Virtuozzo virtualization. The virtualization layer is on top of it and runs as an application within the operating system. The virtualization layer offers a file system and kernel service abstraction layer which isolates resources among all virtual machines called "Containers" and it ensures that each container appears as a standalone server.

II. Related work

Docker has been one of the hottest buzzwords in the IT community over the last year as the open source project has popularized a standard for Linux containers adopted by the world's largest cloud and development environments. With so much hype in the world of DevOps, it's hard to get a sense of reality as far as adoption of container-tech in the enterprise.StackEngine, a developer of automation and management tools for Docker containers based in Austin, Texas, surveyed 745 IT organizations across a broad range of global businesses in January to figure out who's using Docker, and what they're doing with it. The resulting State of Containers Survey 2015 concluded that "Docker adoption is real and ahead of schedule in the enterprise." More than 20 percent of the respondents are already using Docker, and another half of the 745 polled are currently evaluating the technology. Less than a quarter of the remaining 30 percent who aren't interested in Docker -- only 7 percent of the total pool -- say they've never even heard of it. "Within a virtualization and cloud ecosystem of readers, the fact that only less than 7 percent of those responding selected that they haven’t heard of the technology also speaks volumes as to the marketing efforts and visibility and name recognition gained in such a short amount of time," the StackEngine report concludes. Mostly application development and QA it seems, which were use cases cited by 53 percent and 64 percent of respondents, respectively.Another interesting stat is that 31 percent of the businesses polled are either already using or plan to use Docker for production workloads."Again, this number suggests that Docker is being embraced in a relatively quick order relative to how long the technology has been around."StackEngine asked all 745 respondents what factors they believed would motivate VMware users to adopt Docker.The main one cited was the desire to achieve interoperability between various public and private cloud resources -- an answer given by more than 45 percent of respondents.But more than 40 percent of respondents also thought pressure from test departments would get VMware users to turn to Docker, with 23 percent of the total also citing pressure from development groups.About 44 percent of respondents said VMware's cost would drive users to Docker, and independence from VMware was cited as a motivating factor by 35 percent of the 745 businesses polled.The two greatest challenges to Docker adoption, according to the respondents polled by StackEngine, were the security model and a lack of operational tools for production. Each of those perceived obstacles to Docker adoption were cited by 49 percent of the respondents.Lack of development tools was a concern of 37 percent, and 17 percent found Windows support a challenge. Morethan a quarter of all 745 respondents also saw reliance on open source as a problem. More than 44 percent of all respondents -- those using VMware and those not -- said they believed VMware users wanted to manage their Docker containers using existing VMware tools.About a quarter expect VMware users to prefer a new tool for the Docker ecosystem, and another 23 percent of 4 respondents said they think VMware users would prefer a combination of both.Almost 9 percent of the 745 respondents had no opinion on what kind of tools VMware users would want for container management. Virtualization is the process of migrating physical environment into virtual environment. This virtual environment can include anything from virtual operating systems to virtual servers. Many companies have already adopted virtualization because they reduce the overheads like maintaining the hardware which is included in large rooms or data centers occupied with large number of devices and cables. Although Virtualization did not completely solve the problem of using bulky hardware but it got succeeded in reducing the usage of unnecessary bulky and costly hardware which was a burden to most of the organizations. This chapter focuses more on the existing and newly evolved virtualization technologies.

III. Docker:

Docker is an open-source platform that run applications and makes the process easier to develop, distribute. The applications that are built in the docker are packaged with all the supporting dependencies into a standard form called a container. These containers keep running in an isolated way on top of the operating system's kernel. The extra layer of abstraction might effect in terms of performance. Even thou, the technologies of the container have been around for over 10 years, but docker, a generally new hopeful is right now a standout amongst the best innovations since it accompanies new capacities that prior technologies did not have. Initially, it gives the facility to create and control containers. Besides that, applications can easily be packed into lightweight docker containers by the developer. These virtualized applications can easily be worked anywhere without any alteration. Moreover, docker can convey more virtual situations than different innovations, on the same equipment. To wrap things up, docker can easily coordinate with third-party instruments, which help to easily deploy and manage docker containers. Docker containers can easily be deployed into the cloud-based environment. This paper is a review of the technology of docker and will analyze its performance by a systematic literature review. The article is organized as follows. The next section will introduce the technology of docker. A more detailed description of the docker and its components will be presented. Briefly compare the technology of Virtual Machine and Docker. When they are deployed into Containers. In a Container environment where the applications are virtualized and executed, docker adds up an extra layer of deployment engine on top of it. The way that docker is designed is to give a quick and lightweight environment where code can be run efficiently and moreover it provides an extra facility of the proficient work process to take the code from the computer for testing before production [9]. Russell (2015) confirms that as quick as it is possible docker allows you to test your code and deploy it into the production environment [6]. Turnbull (2014) concludes by saying that docker is amazingly simple. Certainly, you can begin with a docker with a simple configuration system, a docker binarywith Linux kernel.

Docker Inside

There are four main internal components of docker, including Docker Client and Server, Docker Images, Docker Registries, and Docker Containers. These components will be explained in detail in the following sections.

Docker Client and Server

Docker can be explained as a client and server-based applicationThedocker server gets the request from the docker client and then process it accordingly. The complete RESTful (Representational state transfer) API and a command-line client binary are shipped by docker. Docker daemon/server and docker client can be run on the same machine or a local docker client can be connected with a remote server or daemon, which is running on another machine

Docker Images

There are two methods to build an image. The first one is to build an image by using a read-only template. The foundation of every image is a base image. Operating system images are basically the base images, such as Ubuntu 14.04 LTS, or Fedora 20. The images of the operating system create a container with an ability to complete running OS. The base image can also be created from scratch. Required applications can be added to the base image by modifying it, but it is necessary to build a new image. The process of building a new image is called "committing a change". The second method is to create a docker file. The docker file contains a list of instructions when the "Docker build" command is run from the bash terminal it follows all the instructions given in the docker file and builds an image. This is an automated way of building an image.

Docker Registries

Docker images are placed in docker registries. It works correspondingly to source code repositories where images can be pushed or pulled from a single source. There are two types of registries, public and private. Docker Hub is called a public registry where everyone can pull available images and push their own images without creating an image from scratch. Images can be distributed to a particular area (public or private) by using the docker hub feature.

Docker Containers

Docker image creates a docker container. Containers hold the whole kit required for an application, so the application can be run in an isolated way. For example, suppose there is an image of Ubuntu OS with SQL SERVER, when this image is run with docker run command, then a container will be created and SQL SERVER will be running on Ubuntu OS.

Docker architecture

The Docker architecture currently leverages Linux Containers (LXC) which have features like cgroups and namespaces for resource control and strong process isolation etc. In addition, Docker architecture could leverage the Kernel-based Virtual Machine(KVM) to do the same things. Linux Containers(LXC) is OS-level virtualization to run multiple isolated Linux systems on a single Linux control host and it works as a userspace interface for the Linux kernel containment features. Linux Containers(LXC) are based on chroot which contains binaries, libraries, and configuration files, therefore, called chroot jail and this approach allows for an isolated environment on top of the kernel. Cgroups are groups of resources that can be created at the Linux kernel level and can be assigned priorities therefore it can be ensured that each virtual machine has exactly only those resources which are actually required. It makes container-based virtualization an efficient environment. Container platforms from Cloud Foundry, Kubernetes, and CoreOS offer feasible virtualization alternatives but Docker containers have gained a lot of momentum and achieved hype status. Companies like VMware, IBM, and Microsoft are working on developing their container strategy on the cloud. Docker provides a feasible and cheaper alternative to hypervisor-based virtual machines. It has two major components- the open-source containerization platform called Docker and Docker Hub which is a Software-as-a-Service(SaaS) platform to share and manage Docker containers. Docker uses a client-server architecture model. The Docker client can talk to the Docker daemon which creates, run, and distribute Docker containers. 18 The Docker client and daemon can run on the same system or a Docker client can ccommunicate through sockets or RESTful API to a remote Docker daemon.

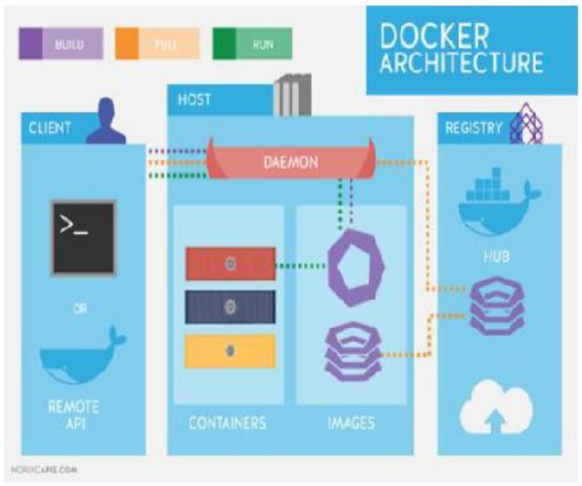

Figure: Docker Architecture

Benefits of Docker Containers:

Docker is a container technology that makes it easy to package and distribute software along with its other dependencies. It makes shipping of software code easy to staging or production or any other environment. Docker is written in Go, an open-source programming language created in 2007 at Google by Robert Griesemer, Rob Pike, and Ken Thompson. The developer community is working aggressively on Docker API which has 15 revisions made so far in the past 1.5 years. IBM is a founding member of the Open Container Platform (OCP) formed by partners and users to create industry standards around container formats and runtime. IBM Containers are built on Bluemix which is a Platform-as-a-Service(PaaS) that provides an efficient environment to enable faster integration and access to big data, analytics, and security services.

Application Portability:

Docker puts all application dependencies in a container that is portable on different platforms. The distributed applications can be built, move, and run using containers. The application developers and administrators can run the same application on laptops, VMs, or cloud by automating deployment inside containers.

Docker is lightweight and fast: Containers are lightweight and fast compared to Virtual Machine(VM) since VMs boot an entire operating system to start and consume resources as each VM has to run a full OS instance. However, starting a container is just like starting a process.

Optimal resource utilization:

Docker allows allocating and limit CPU, memory, network, and disk resources to all the processes using Linux's Control Groups. It ensures that one process is not taking over all of the computer resources and starving the other processes.

Docker is growing:

Docker depends on Linux Containers (LXC), groups and namespaces capabilities that don't exist in Windows. Microsoft has its own container technology on Windows but they are working on hooks to enable Docker containers to run on Windows Server.

Best fit for Microservices architecture:

The Microservice architecture is supported by containers as each microservice can be deployed without interfering with other microservices. Containers provide a suitable environment for service deployment in terms of speed, isolation, and ease of deployment of new versions.

Rapid Deployment:

Docker manages to reduce deployment to seconds. This is due to the fact that it creates a container for every process and does not boot an OS. Data can be created and destroyed without worry that the cost to bring it up again would be higher than what is affordable.

Resource Utilization:

The containers are lightweight, portable, efficient, and can run on physical servers. We can run more containers on physical servers than virtual machines which result in higher resource utilization. Containers consist of application code along with its dependencies and runs as an isolated process sharing the kernel with other containers in the user space on the host operating system

IV. Proposed Methodology:

Now we are introducing the container platform called Docker.in this we have docker-engine it is the service that runs on background and interacts with containers and we have binaries and libraries instead of being run on the guest OS they are built into special packages called docker images. Then docker engines run those images. Containers share the kernel so they are much efficient than virtual machines. As in the example discussed above in existing system and it can be solved using docker because the container has several layers all the changes you have made to the os is saved into one or more layers and those layers are part of images, So where ever the images go the dependencies would be present as well.

● Here the containers simply share the host operating system, including kernels and libraries so they don't need to have a private memory space.

● the containerized application starts in the second.

● So the application works same in the both the developed environment and testing environment.

V. Conclusion

Containers provide a mechanism to improve efficiency through the sharing of operating system binaries. This approach helps to improve host capacity and makes patch management easier since there are fewer operating systems to patch. A container keeps the application binaries and configuration files and stores the OS components which application modifies. Containers as a service (CaaS) are a form of container-based virtualization in which container engines, orchestration & underlying computing resources are provided to users as a service. Most of the public cloud providers like Amazon Web Services (AWS), IBM, Google, Rackspace, and Joyent have some type of CaaS offering. The adoption of containers is expected to grow and the majority of Microservice applications will be built on the containers in the future. And existing cloud platforms will either switch to a new container stack or at least start supporting containers.

• It is, easy to use since it uses the GUI provided in the user dialog.

• User-friendly screens are provided.

• The usage of software increases efficiency decreases the effort.

• It has been efficiently employed as a Site management mechanism.

• It has been thoroughly tested and implemented.

Future Enhancements

I have primarily focused on Docker containers and it is out of box features used for faster delivery and containerization of our applications. In the future, I would like to work on automating the creation of containers using scripts and several other automation tools such as Jenkins where we can achieve Continuous Integration and Continuous Delivery while building an application. Apart from this, I would also like to work on several other third-party cluster management tools such as Apache Mesos and Kubernetes which are much more sophisticated and advanced than the Docker's own swarm mode.

VI. References

[1]container-based virtualization (operating system-level virtualization) - http://searchservervirtualization.techtarget.com/definition/container-based-virtualization-operating-system-level-virtualization

[2]Making the case for container-based virtualization over hypervisorshttp://searchservervirtualization.techtarget.com/tip/Making-the-case-for-container-based-virtuali zation-over-hypervisors

[3]Virtualization performance and container-based virtualizationhttp://searchservervirtualization.techtarget.com/tip/Virtualizationperformance-and-container-based-virtualization

[4]Virtualization withoutthe Hypervisorhttp://docs.media.bitpipe.com/io_12x/io_128710/item_1261181/Virtualization%20Without%20T he%20Hypervisor_hb_final.pdf

[5]Containers vs. VMs: What's the difference? http://searchservervirtualization.techtarget.com/answer/Containers-vs-VMs-Whats-the-difference

[6]Why system Linux containers make sensehttp://searchservervirtualization.techtarget.com/tip/Why-systemd-Linux- containers-make-sense

[7]Docker leads the container technology charge in cloudhttp://searchcloudcomputing.techtarget.com/feature/Docker-leads-the-container-technology-char ge-in-cloud

[8]VMware container platforms offer increased flexibilityhttp://searchvmware.techtarget.com/tip/VMware-container-platforms- offer-increased-flexibility 69

[9]IBM hitches a ride on the Docker bandwagonhttp://searchcloudcomputing.techtarget.com/news/2240236168/IBM-hitches-a-ride-on-the-Dock er-bandwagon

[10]Containers: Fundamental to the cloud's evolutionhttp://www.zdnet.com/article/containers-fundamental-to-the-evolution- of-the-cloud/