IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.105122

An Insight into Containerization

T Terry Newton[1], Nagaraj G Cholli[2]

Student, Information Science, R.V College of Engineering, Bangalore, India[1]

Professor, Information Science, R.V College of Engineering, Bangalore, India[2]

Abstract: Containerization has become a major trend in the IT industry. It is starting to pose stiff competition to VMs. It is believed that in the near future that containerization would dominate the IT industry. This paper will give insights into the pros and cons of containerization. It will also provide some knowledge on the road ahead for containerization and end with an example of one significant container implementation.

Keywords: containers, container engine, microservices, containerization, virtual machines

I. INTRODUCTION

Containerization in simple terms can be described as wrapping a software system (code) along with its dependencies enabling it to run independently of the infrastructure. This technology is gaining significant recognition from developers and operations groups' due to the benefits it can provide. Containerization is a relatively new trend in software system development even though it has been present for a few years now. It is often considered as an alternative or even as a replacement for virtualization.

A container is a software entity that encapsulates the code along with each of its dependencies which allows the application or the software system to run quickly and reliably on a variety of computing environments. The main principle behind containerization is "*Write Once and Run Anywhere*". It offers a solution to one of the major problems in the IT industry i.e. namely portability. It also addresses many other problems such as fault isolation, ease of management, security, efficiency to mention a few. Containerized applications are 'isolated' i.e. they are not associated with a copy of the OS. In place of the association, they have a container runtime engine that runs on the host's OS and it provides a pathway for containers to share the OS.

The most common quality that is attributed to a container is 'lightweight', the reason that they are considered lightweight is that containers share the machine's OS kernel and don't have the overhead which comes when we associate an OS with each application. The absence of this association in containers allow them to be smaller in capacity than Virtual Machines and the time required to get them up and running are less. It allows many containers to run on an equivalent computing capability as that of a single VM. This has a positive effect on efficiency and pricing. All in all, we get the same or in some cases an even better service for a lower cost.

The performance of containers and VMs are usually measured with respect to each other since both of these technologies enable a considerable amount of computing efficiencies by allowing various categories of software systems to be run in a single environment. But due to the crucial benefits it provides over VM containers are becoming more favourable to use. Figure 1 & 2 gives us a better idea of the structure of containers vs structure of VMs.

One of the IT trends that software firms are embracing is microservices. It is considered a better way of application development and management compared to the traditional monolithic model which packages a software system, UI and database into a single unit. Microservices breaks a complex application into smaller atomic specialized services each having its database and logic.

Microservices and containers form a very good pair when combined. The backbone idea behind both of these is the same i.e. to break an application into modular components so that they are portable, scalable and efficient to manage. Lightweight encapsulation of microservices is possible using containers. Microservice once encapsulated then gains all the inherent perks of containers.

Enterprises are embracing containerization as it gives the developer the ability to build and deploy quickly and securely. There are various reasons behind why enterprises are rapidly switching to containerization and why they consider it as a better approach for the development and management of applications. In this paper, we discuss those

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.105122

reasons. We also have a look at the road ahead for containerization, the challenges it faces and may face and the scope for its development.

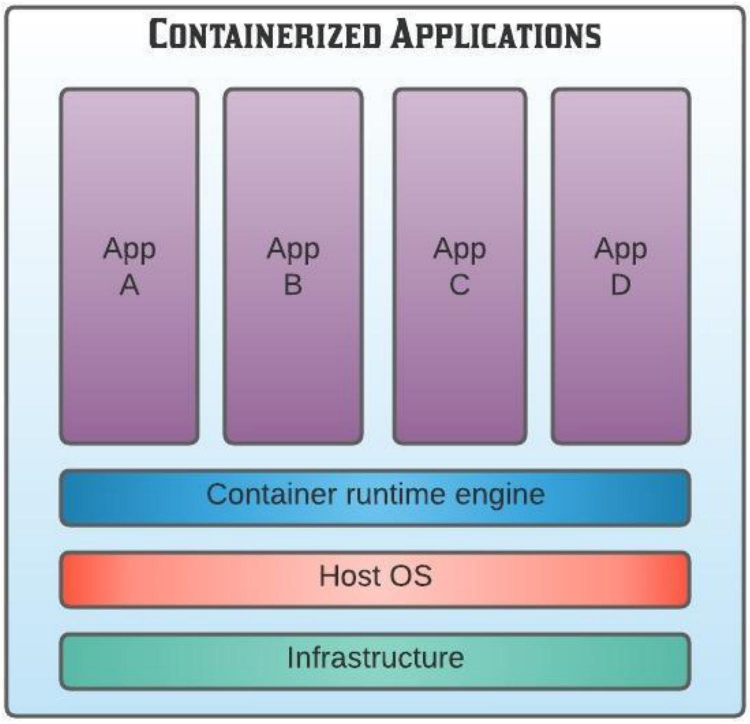

Figure 1: Containerized Applications

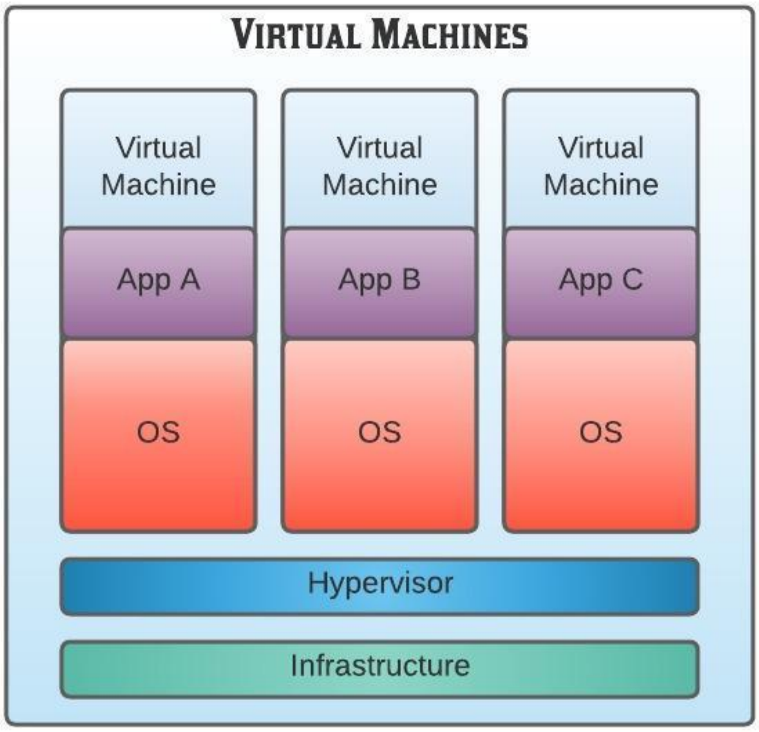

Figure 2: Virtual Machines

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.105122

II. Benefits of Containerization

In this section, we try to list the various benefits or advantages that containerization provides.

A. *Portability*

Containers have this property of portability due to the fact that it creates an executable package of code that is independent of the host OS, which enables it to run on various platforms or even the cloud. The concept of creating containers was developed keeping in mind the portability that the traditional software system development lacked which often lead to companies or individuals developing the same application multiple times for different platforms., which in turn reduced the efficiency of the development process.

B. *Better Application Development*

Containers are development-friendly. It supports many techniques which makes the development a lot easier. To mention a couple, it supports agile and DevOps to accelerate development, test and production cycles. All in all, it is not tough to develop an application in containers despite its comparative newness in the IT industry.

C. *Isolation*

Isolation in this context means the ability of the software system to run independently of the host OS. Containers don't link a copy of the OS with each containerized application. Hence isolation is attained. Even though a copy of the host OS is not linked to the containerized application other container layers, like bins and libraries may be shared among multiple containers.

The main benefit which is obtained due to isolation is that the overhead of running an OS inside every container is eliminated and also renders the container smaller in capacity and in turn driving up the start-up times. Not only that since containers are isolated the possibility of malicious code present in one system is drastically reduced.

D. *Speed*

Containers share the machine's operating system (OS) kernel and they aren't slowed down due to the added overhead of accessing the same OS. In fact, they have the same speed or even higher speed with respect to certain aspects of the software. One notable difference is the boost in start times because when a containerized application is loaded only the application has to be booted up and not the OS.

E. *Abstraction*

Abstraction in simple terms could be described as hiding the internal details of a system. For example, a coffee vending machine makes coffee but we don't see how the coffee is made or the ingredients that are added.

Similarly, containerization abstracts containerized application from the host OS. The OS doesn't know the internal details of how the application is set up, how it has been developed etc. The only thing that the host OS does is to provide resources for the containerized application to use and run.

This attribute of abstraction which containerized application possess makes them portable and gives them the ability to run without much problems across any platform or cloud. Containers can be moved from a PC to a virtual machine or vice versa or hosted on a different OS, they can run on 'bare metal' hypervisors as well as hosted hypervisors. Abstraction pretty much multiplies the range of platforms on which a containerized application can run. The developer can develop an application using native tools which might be specific to a particular platform but have it run on many other platforms.

F. *Isolation of Faults*

Containerized applications are isolated that is they operate independent of each other, because of this independence fault isolation is a lot easier. Fault in one of the containers doesn't affect the operations of other containers. Other containers can continue to remain operational as development teams determine and rectify the technical problems with the faulty container. For further isolation of faults within containers, the container engine can take the aid of OS security isolation techniques like SE Linux access control.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.105122

G. Improved Efficiency

Efficiency is the one thing that every company or individual always tried to achieve, is trying to achieve and will try to achieve. Efficiency can include many things such as cost efficiency, resource efficiency etc. Containerization enhances efficiency by reducing the capacity compared to a VM (which stores the OS as well). In other words, the capacity required by a VM is always larger than the capacity required by a container. Due to their smaller capacity, it allows many containers to run on equivalent computing ability as a single VM. This leads to higher server efficiency, time efficiency as well as cost efficiency.

H. Faster Deployment

Deployment has always been a platform specific process. It makes a software system available to the intended users. The reason it has been platform specific is that applications developed based on traditional application development methods and processes cannot be deployed on a variety of platforms without difficulty.

Containers inherently being isolated from platforms makes it much easier to deploy. Hence deployment is faster compared to the traditional application deployment.

I. Easy Management

Automation of installation, scaling and management of containerized workloads and services can be handled by a container orchestration platform. They can reduce human intervention by a large margin and can handle managements task such as the scaling of a containerized application, version control, rectify errors just to mention a few of them. Google Kubernetes is one of the most commonly known containers orchestration platforms. It works with container engines and makes management a lot easier.

J. Lower Prices

Containerizing an application costs less than having it run on an individual virtual machine. The main factor causing a difference in prices is that containerized applications operate on a virtualized OS whereas each VM runs its OS. Virtualizing OS is cheaper than having a separate OS for each VM.

K. Higher Productivity

Containers support a rapid development environment to create more applications. Since containers are portable and portable applications use the platform’s source code to run, it provides the developers with the ability to change and track changes in the platform’s source code which results in higher productivity.

L. Simplicity of Development

Development in containerization is a lot simpler than traditional approaches as scaling is a lot easier and cost efficient as well. Handling failures are relatively simplified and are graceful and upgrading is simpler as well.

M. Improved Security

Security is a broad concept and it can be affected in many different ways. Containerized applications are isolated from each other because of which the presence of malicious code can’t affect other containers. To complement it, security permissions are often defined to automatically filter out components entering the container or limit unnecessary communication with resources.

III. CONS OF CONTAINERIZATION

A. Major Support for Linux Systems

At present containerization works well and has major support for Linux. While container environments for other OS like Windows exist, it’s not as well supported as for Linux

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.105122

B. Exposition due to shared OS kernel

Containers share a single OS kernel because of which risks could occur. If the OS kernel becomes vulnerable, the vulnerability is passed on to the containers. It implies that all of the containers which share the affected vulnerable OS become vulnerable.

C. Lower performance of graphics intense applications

Graphics intense, GUI based applications often require a high amount of resources for them to run optimally but at present containers struggle at this and VMs continue to be the better option.

D. Complexions due to multi-layered architecture

Containerization generally adopts multi-layered infrastructure. It follows the principle of one application per container. It can lead to additional overhead which wouldn't be there if all the applications are run on a single VM.

IV. FUTURE CHALLENGES

Containerization is a technology that provides many benefits but as with any technology, there is always a scope for improvement and has its own set of challenges.

A. Susceptibility due to Meltdown And Spectre Attacks

Meltdown and Spectre Vulnerabilities exist in nearly all modern processors. Modern processors perform what is known as speculative execution to speed up execution times. Meltdown and Spectre make use of speculative execution to access memory locations which they aren't allowed to. Nearly all systems in the world can be susceptible to this. But containers can be even more susceptible since they share the same kernel.

If either meltdown or spectre attack occurs in a system, the kernel is affected and since the kernel is shared among the containers it makes all of the containers running on the OS susceptible as well. To cloud providers that offer Caas (Container as a Service), it can be a major threat.

B. Standardization of Container Security

Containerization being a fairly new technology lacks the standardization of security protocols. It is necessary to establish standards for container security in order for containers to continue to grow in the market.

C. Usage of Insecure Images

Containers are built using images, it may either be a parent image or base image. The usage of images to build containers gives the ability to reuse various components of the image rather than building a container from scratch. But on the flip side of the coin, they are prone to be vulnerable, either the image or their dependencies may contain malicious bugs.

It is necessary to curtail these vulnerabilities. Image authenticity can be verified using a notary but it ends up beings a centralized solution. For a decentralized one usage of blockchain appears as a viable solution and in fact, a solution was proposed known as Decentralised Docker Trust (DDT) for Docker for image verification.

D. Vulnerability due to Privileged Containers

Certain containers may have a privileged flag, it enables those containers to do nearly anything that a host can do. It has increased capabilities and gains access to the host's device. Hackers can use this to their advantage if can hack into a container running on privileged flags. The one straightforward way to combat this is to remove such privileges and maybe implement fine-grained permissions.

E. *Digital Forensics*

Digital forensics involves analysing security incidents. The current trend in the IT industry is to use microservices that are run using containers. Digital forensic techniques for containers are still at a dormant stage. Hence performing digital forensics for containers is not easy at present.

F. *Problems due to excessive inter-container communication*

Containers by default communicate with each other in order to function. Due to the number of containers and microservices involved it may be a challenging task to implement networking and firewall protocols. The main aim is to minimize communication as much as possible and only allow mandatory communication. Container orchestration platforms such as Kubernetes are a good option to handle these communication problems.

G. *Optimizing Infrastructure for Containers*

The mechanism of containerization has a lot of benefits but containers don't run independently on their own meaning they have an underlying architecture on which they run. The current traditional architecture is not as supportive to containers as it is to VMs and traditional machines. Hence to implement containers efficiently, it is necessary to optimize the infrastructure on which the container will be running.

H. *Choosing the right container technology*

In a fast-paced world, the rapid evolution of container platform components such as orchestration, storage, load balancing, networking etc makes it harder to choose. Companies want to invest in technologies that they can use for a long period of time but due to the rapid evolution of container technology, it is tough to choose one.

I. *Lack of Uniformity in Performance*

Containerization promises performances benefits but it is not uniform. The ones mainly able to use this advantage are microservices as they are small and have lesser interdependencies. But large monolithic applications aren't able to use this performance advantage and are less ideal for deployment and execution.

J. *The Skills required are different*

The skillset required for developers who work on containerization is different. Containerization is a new technology it's not even a decade old hence many of the current developers don't have the expertise as well as the experience of working with containers. Not only that the skill-sets required are nearly completely different than those required for building traditional applications.

It requires them to know architecture, data flow and application logging along with good coding & scripting skills. And due to the rapid evolution of containers, it is tough to find a developer who has detailed knowledge.

K. *Container Monitoring*

Container monitoring is not easy as it requires us to keep pace with ephemeral, super-fast and lightweight containers. We also have to ensure that we don't put any unwanted load on the containers. We have to deal with huge complexities and is not as simple as monitoring regular conditions. Very few companies have been able to do this successfully.

L. *Vulnerability Assessment Tools' Dependability*

Surveys done by researchers have shown that docker images may have high-risk vulnerabilities that vary from 30% – 90%. To detect these, we require vulnerability assessment tools. The market has many such tools designed for this purpose. But the only hitch is to ensure that if whether the tools are usable in the production environment and do not hinder the deployment process in any way.

V. APPLICATIONS OF CONTAINERS

Containerization can be used in different fields, a few of them are listed below:

- Healthcare
- Network-Function Virtualization
- Microservices
- Scientific Applications
- It can be used in the aid of Edge Computing (EC)
- IOT
- Gaming

Now we have a look at one such use of containerization that has been having tremendous success

Pokémon GO a game that became a big-time success was built based on container based development. Niantic i.e. the company used Google's Google Container Engine (GKE) is based on Google's Kubernetes Project to run the application logic for the game.

Due to GKE's ability to orchestrate container cluster at a huge scale Niantic's team could focus on updates to the game rather than focussing on management tasks. An interesting thing to note is that the teams targeted 1x traffic and expected it to be 5x at the worst case scenario but the game's popularity made the user traffic surge 50x of the expected traffic. But due to the scalable nature of containers, Google was able to provide scalability ensuring that the players had no problems. Pokémon GO was the largest deployment of Kubernetes on Google Container Engine.

Containers were able to provide scalability without the disruption of services to mention one among the many advantages it has. Hence it proves that the range of container applications can only become diverse.

VI. CONCLUSION

Containerization is a relatively new technology that is beginning to make its impact on the market due to the advantages and benefits it has to offer. As with any technology, containerization has its pros and cons. But due to its higher share of benefits, it can make a huge impact on the IT industry. It is also necessary to keep in mind that the road ahead is not smooth as there are challenges to overcome. All in all, containerization has the potential to simplify a lot of things.

REFERENCES

[1] C. Pahl, A. Brogi, J. Soldani and P. Jamshidi, "Cloud Container Technologies: A State-of-the-Art Review," in IEEE Transactions on Cloud Computing, vol. 7, no. 3, pp. 677-692, 1 July-Sept. 2019, doi: 10.1109/TCC.2017.2702586.

[2] J. Watada, A. Roy, R. Kadikar, H. Pham and B. Xu, "Emerging Trends, Techniques and Open Issues of Containerization: A Review," in IEEE Access, vol. 7, pp. 152443-152472, 2019, doi: 10.1109/ACCESS.2019.2945930.

[3] S. Sultan, I. Ahmad and T. Dimitriou, "Container Security: Issues, Challenges, and the Road Ahead," in IEEE Access, vol. 7, pp. 52976-52996, 2019, doi: 10.1109/ACCESS.2019.2911732.

[4] R. Dua, A. R. Raja and D. Kakadia, "Virtualization vs Containerization to Support PaaS," 2014 IEEE International Conference on Cloud Engineering, 2014, pp. 610-614, doi: 10.1109/IC2E.2014.41.

[5] S. Singh and N. Singh, "Containers & Docker: Emerging roles & future of Cloud technology," 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016, pp. 804-807, doi: 10.1109/ICATCCT.2016.7912109.

[6] 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), 2018, pp. 178-185, doi: 10.1109/CLOUD.2018.00030.

[7] Sharma, Prateek et al. "Containers and Virtual Machines at Scale: A Comparative Study." *Proceedings of the 17th International Middleware Conference* (2016): n. pag.