International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 3 | ISSN : 2456-3307

# Providing Deployment Services Using Container Technologies

**Shanmugapriya R[1], Nissha Lingesh S[2], Devi K[3]**
[1,2] UG Scholars, Department of CSE, Velammal Engineering College, Chennai, Tamil Nadu, India
[3]Assistant Professor, Department of CSE, Velammal Engineering College, Chennai, Tamil Nadu, India

## ABSTRACT

Utilizing cloud services is not economic support in the recent technology revolution which is main road block of common users to gain knowledge in cloud technologies. This paper explains about providing a highly efficient and performant cloud environment using container technology such that it can run on low end system like a home computer. We also discuss the process of container creation, benefits and limitations of containers.
**Keywords:** cloud, container, container Image, Deployment, MicroServices

## I. INTRODUCTION

Containers are generally lightweight virtual machine but not a virtual machine. Consider an Operating System(eg.Linux) we can run as many processes we want but if we want to isolate a process Containers are used. Containers are basically sandbox for a process. Sandbox means that the process for start has its own namespace and cgroups which allows us to restrict what the process is able to do certain capabilities and resource limits that we can apply to the container. Containers and container process are tightly coupled, if we start a container its process will also get started and if we stop the container process container will also be stopped.

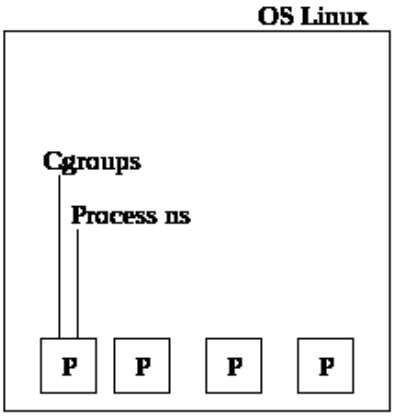


**Figure 1.** Isolating process using cgroups and namespaces

## II. CONTAINER IMAGE

Container Images are binary representation of a container, just a bunch of bits on a file system.Container Image is a notion of parent-child relationship and image layering.

Consider If we start at the root of the parent-child tree we have an image called scratch and it is an empty formatted file system on the top of scratch we have the bare bones of an operating system(eg.debian) then that image can be the parent of another image and that image may be sshd and on top of that may be we can have a small application. The images are arranged in this image hierarchy. The advantage of this image hierarchy is we can share images. If we want to run the app the application it will pull the branch of this tree and we not stack the entire application in single file. It also allows us to concentrate specific things in specific places and know where they are.

For example consider a particular vulnerability has been identified in the user libraries in debian and we need to replace it. All the applications that are running is going to be affected by the same

vulnerability. We can easily identify this because of the tree structure, every single child node of debian is going to be affected by the vulnerability. So by rebuilding debian and by rebuilding all child node of debian and by redeploying everything there is high degree of certainty that the thing patched is now inherited by everything else.

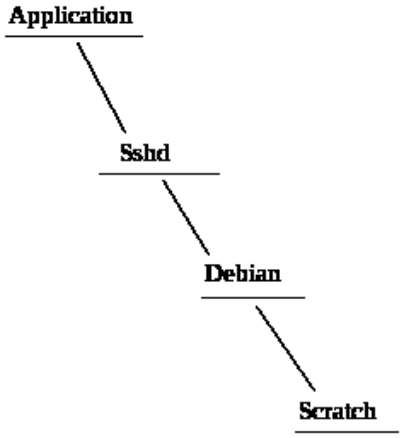


**Figure 2.** Image Tree showing parent-child Relationship

Considering Image as a template and creating any number of instances of that template clearly explains the runtime representation of container.

## III. CONTAINER CREATION

Containers are created by creating an instance of root file system and the created instance is wrapped using namespaces(host system,pid,network namespace). Now the created instance will act as a separate system this is established using chroot.Finally 'proc' and other necessary library files are mounted from the host system to the cloned system.

### A. Algorithm

Input : rtfs (of host system)
set x= create() → instance of rtfs created
rtfs as root file system
Input : hs,pid,nn
x = instance(hs,pid,nn) → wrapped instance created
hs as host system, pid as process id, nn as network namespace
Input : i
x = chroot(i) i→ created instance
Input : proc
x = mount(proc) proc → file used to run all processes
Input : lib
x = mount(lib) lib → library files

### B. cgroups

cgroups is a feature of linux kernel.Cgroups limits, accounts for, and isolates the resource usage of a collection of process.Some of the features of cgroups are resource limiting, prioritization, accounting and control.

1. **Resource Limiting:** Groups are configured with a memory limit
2. **Prioritization:** Each group may contain different share of CPU utilization or disk I/O throughput.
3. **Accounting:** Measures the resource usage of usage of groups which may be used for billing purposes.
4. **Control:** control over groups of processes.

### C. chroot

chroot changes the apparent root directory for the current running process and its children when we change the root to another directory we can not access files and commands outside that directory.

### D. Namespaces

Each process in a container are associated with namespace and the process can see or see the resources only with the same namespace. Namespaces are usually classified into two PID namespace and Network namespace .PID namespace provides process isolation and Network namespace provides network isolation.

## IV. CONTAINER DEPLOYMENT

### A. Monolithic Architecture Versus Micro Service Architecture

In monolithic architecture applications were contained in one large blob of binaries and libraries.Since they were built on a single stack such

as .NET/Java , patches and new functionalities usually took a long time to develop.Applications which are developed based on monolithic architecture are usually deployed on a single server. Even though they are load balanced they still monolithic architecture.

In microservice architecture applications are constantly developed and will not be able to see patches and updates to a huge blob of a specific application usually but would see updates for each part of the app.Applications which are developed based on microservice architecture are usually deployed to a multitude of servers and the pieces of the app can communicate with each other.

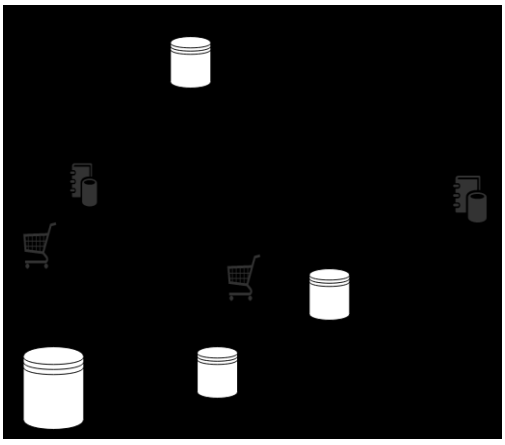

**Figure 3.** Monolithic versus Micro Service Architecture

## V. ADVANTAGES OF CONTAINERS

**A. Portability:** If we build a container in a laptop it can run on another laptop which has compatible operating system and a compatible control plane.

**B. Image Format:** The fact that every image is just a snapshot so it allows us to be much more efficient with how we move binaries around.We are only moving around the snapshots with necessarily moving around the entire stack.

Images have a parent-child relationship and we have effectively a tree of images that is really helpful in identifying a particular image which has a problem and it has children which are going to impacted by that problem.

**C. Performance:** Containers are created much faster than virtual machine instances.

**D. Productivity:** Each container can be seen a process and so it can be independently operated without any synchronization problems.

**E. Standardization:** Containers are mostly developed based on open standards and it can run on all major linux distributions.

**F. Secure:** Container isolates the process of one container with another container so any upgradation or changes in one container do not affect another container.

## VI. DISADVANTAGES OF CONTAINERS

**A. Complexity:** Increase in the complexity of the containers with the increase in n of containers running an application.

**B. Linux Support:** Most containers are based out of linux running these containers in microsoft environment will result in complications.

## VII. CONCLUSION

To conclude containers are more flexible and powerful even though there is a security risk of placing containers on the top of host system we can create a virtual machine on the top of host and place the containers in virtual machine.Containers made easy the process of deployment and it is faster because containers are just isolated processes.

## VIII. REFERENCES

[1]. Jamshidi, P., Ahmad, A., and Pahl, C. (2013a). Cloud migration research: a systematic review. IEEE Transactions on Cloud Computing, 1(2):142-157.

[2]. Jamshidi, P., Ghafari, M.,Ahmad, A., and Pahl, C. (2013b). A framework for classifying and comparing architecture-centric software evolution research. In 7th European Conference

on Software Maintenance and Reengineering,pages 305-314.

[3]. Kecskemeti, G., Marosi, A. C.,and Kertesz, A.(2016). The entice approach to decompose monolithic services into microservices. In Intl Conf on High Performance Computing Simulation (HPCS), pages 591-596.

[4]. Kratzke, N. (2015). About microservices,containers and their underestimated impact on network performance. Conf on Cloud Computing, 2015.

[5]. S. Yang, P. Lai, and J. Lin,"Design role-based multi-tenancy access control scheme for cloud services," in Proc. Int. Symp. Biometrics Security Technol., 2013, pp. 273-279.