WOC
WORLD OF CONFERENCES

III international scientific conference

Innovative scientific research

Toronto
16-17.03.2023

Innovative scientific research

Proceedings of the III International Scientific and Practical Conference

16-17 March 2023

Toronto. Canada

2023

UDC 001.1

BBC 1

III International Scientific and Practical Conference «Innovative scientific research», March 16 – 17, 2023, Toronto. Canada.77 p.

ISBN 978-92-44513-41-5

DOI https://doi.org/10.5281/zenodo.7754319

Publisher**: «World of Conferences»**

Main organization: **ESD GROUP**

Editor: Tarmo Vesik

Layout: Asko Laar

The sample of the citation for publication is*: Tsygankov V.D. NEUROCOMPUTER GENETIC "CONTINUUM OF MUTANT NUCLEOTIDES" IN THE FORM OF SILICON NANOELECTRONIC ARTIFICIAL "LIVING MATTER"// Innovative scientific research. Proceedings of the III International Scientific and Practical Conference. Toronto. Canada. 2023. Pp. 5-9. URL: https://conference-w.com/*

Contact information

Website: https://conference-w.com/

E-mail: can@conference-w.com

Content

Technical sciences

ADVANTAGES AND DISADVANTAGES OF MICROSERVICES CONTAINERIZATION TECHNOLOGY

Aralbayev S.U.
Undergraduate, Al-Farabi Kazkh National University, Almaty, Kazakhstan
Ziyatbekova G.Z.
PhD, acting associate professor of Al-Farabi Kazkh National University, Almaty, Kazakhstan; Senior Researcher, RSE Institute of Information and Computational Technologies MSHE RK CS, Almaty, Kazakhstan
ORCID: 0000-0002-9290-6074

Abstract
The article deals with the implementation of containerization technology in the Linux kernel. By studying the isolation of the user space, the file structure in the Linux kernel, we will define the advantages and disadvantages of containerization technology.

Keywords: Linux, Docker, Containerization, Namespaces, Cgroups.

Introduction

Containerization is a technology that allows you to package and run an application Together with its dependencies and configurations in a lightweight and isolated environment called a container. It is very important to understand that the Container runs in the kernel of the operating system and is isolated by operating system tools, not by computer hardware features such as a virtual machine. Today's popular containerization technologies include Docker, Apache Mesos, and Jail. These tools make it easy to manage and organize containers across multiple servers and platforms. This allows organizations to scale their applications quickly and efficiently. Containerization technology has become an important component of modern application development and deployment and is used by many companies and organizations to build and deploy their applications [1].

Experimental

To gain a deeper understanding of modern containerization technology, we first take a brief look at the history of containerization technology. Experts began to think about isolating user space in 1979 when chroot technology appeared in the Unix kernel. A chroot is the replacement of the filesystem root for a group of processes or a temporary change of the root and the context for running selected processes. More precisely: a chroot adds a second root directory "/" to the system which from the user's point of view will not differ from the first one. After using chroot, as shown in Figure 1, the file system began to have the following structure:

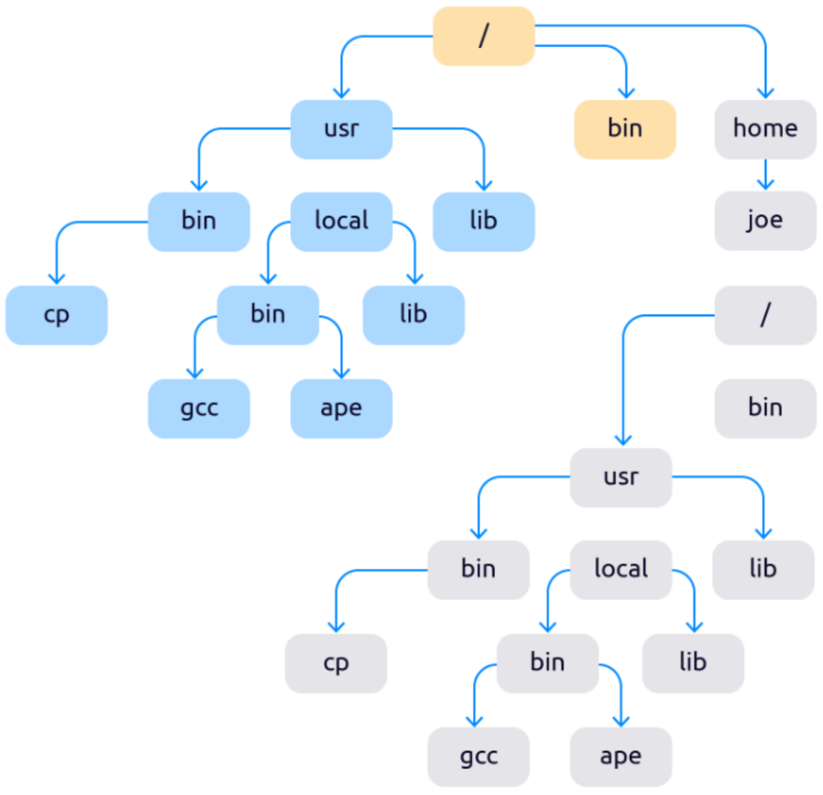

Figure 1 - Structure of the file system

The file system is now split into two parts which do not affect each other. The chroot is now a standard feature for any *Nix OS. This technology was used unchanged until BSD 4.2 in 1982. Therefore, there are many disadvantages:

- Shared process space. A process which is running in chroot sees all other processes;
- Shared network. It is not possible to run servers in isolation in different chroots;
- No possibility to set resource limits. Process in Chroot another in the system all resources, such as processes, can be occupied by one person at a time.

Chroot is still used in the following tasks:

- Restricting the rights of anonymous users connected via FTP;
- Controlling the rights of users connected via SSH.

The Chroot serves to isolate processes. For example, bind in most Linux distributions runs in chroot, many daemons turn chroot into an empty directory before reducing privileges [2].

Jail was introduced in FreeBSD 4.0 in 1999-2000. As we can see in Figure 2, the jail structure is exactly the same as chroot, but more complex. For the first time, it is possible to isolate the network in Jail. We would not go wrong, however, if we say that in FreeBSD this isolation was partial. Because the network interfaces were visible in any environment, but depending on the environment, only certain IP addresses assigned to certain interfaces were visible. Thus, loopback became generic.

Jail was much better than chroot, and even hosts were built, but resources still had no limits. Assigning resource limits became possible only in FreeBSD Version 9.0 with the introduction of the RCTL resource manager. This system allows resources to be limited by individual users and processes as well as by the entire jail.

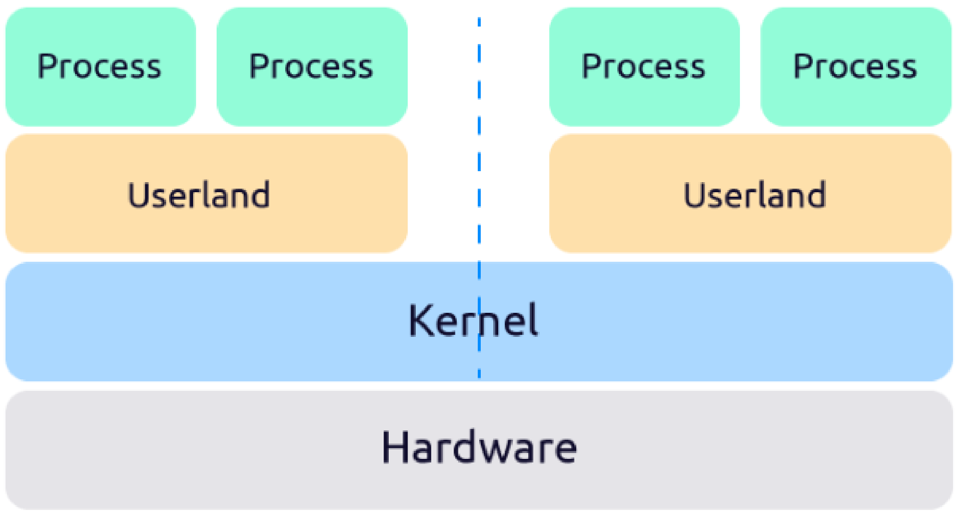

Figure 2 - Jail Containerization Technology in FreeBSD 4.0

Today, Jail technology is mostly used for web hosting, where it provides a secure environment for hosting multiple Web sites on a single server. Each Web site is isolated in its own space, preventing one Web site from affecting the performance or security of the others. It is also used for software development and testing, where it provides a clean environment for testing applications without affecting the host system.

Jail technology is often compared to other containerization technologies such as Docker and Kubernetes. While these technologies provide advanced features and are flexible, they require more resources and are more difficult to install and manage. Jail technology is a lightweight alternative that is easy to use and provides a secure and isolated environment for most use cases [3].

The Namespaces API, which is still used in resource isolation, appeared in 2002 in Linux kernel version 2.4.19. Namespaces is a software layer on top of physical resources. Previously, processes directly accessed resources. Since the introduction of Namespaces, all requests will go through this additional layer of abstraction. The structure of Namespaces is shown in Figure 3.

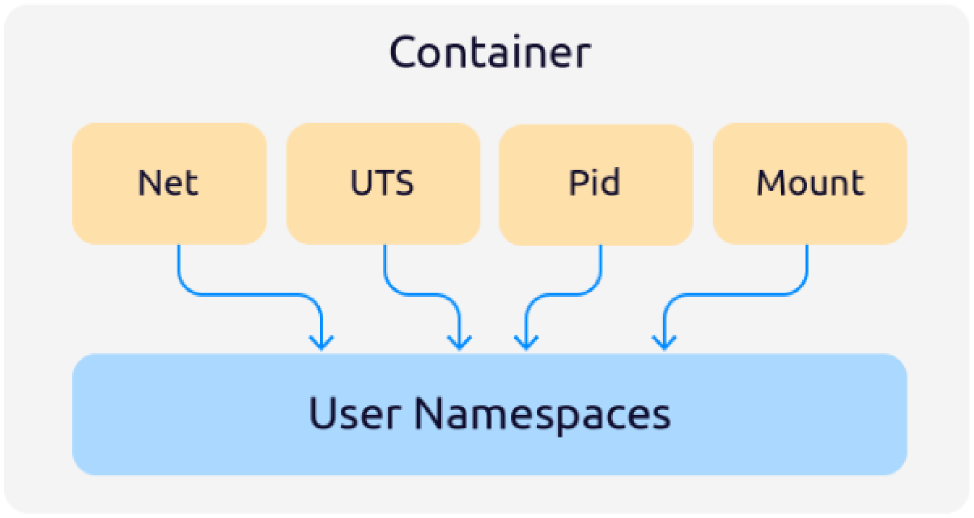

Figure 3 - Structure of Namespaces

The network is an abstraction over the network: interfaces, routing tables, etc. The network namespace basically acts as a tunnel between the various network namespaces.

UTS (Unix time Sharing) is an abstraction on the hostname space and NIS. UTS allows containers to have their own NIS Domain Name and Nodename container names.

PID is an abstraction in the process number space. The PID isolates the process ID space. Processes in different spaces can have the same ID.

Mount is an abstraction over the namespace for file systems. Mount allows a new device to be mounted in multiple file system namespaces at once instead of installing in each namespace.

IPC is an abstraction of interaction between processes. A process in an IPC namespace cannot read or write IPC resources belonging to another namespace. Thus, processes in one container cannot interfere with other containers.

User is an abstraction of the user space. User isolates the user and group identifier, root directory, and keys.

Thus, the Namespaces technology solved the isolation problems, but could not solve the resource limitation problem for isolated processes. A solution to this problem was invented in 2008, thanks to the Cgroups mechanism with the 2.6.20 kernel release [3].

Cgroups (control group) are a group of Linux processes that are isolated and limited to the computing resources of the Linux kernel (CPU, network, memory resources, I/O resources). Cgroups are an excellent solution, which has advanced containerization technology. With this mechanism, as shown in Figure 4, we can not only isolate containers from each other, but also limit the resources they use.

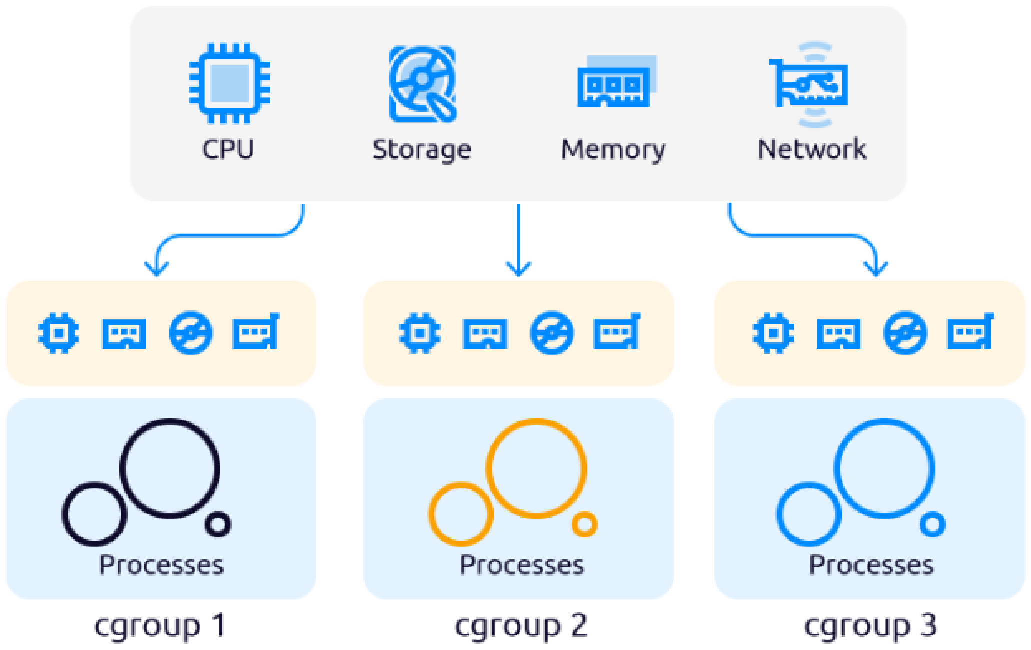

Figure 4 - Resource-limiting cgroups technology

Cgroups are often used in combination with other containerization technologies, such as Namespaces, to create lightweight and isolated environments that can run multiple applications with different resource requirements. For example, a container running an application that requires a CPU can be given a high shared CPU resource in a cgroup, and a container running an application that requires memory can be given a higher memory threshold in another cgroup [4].

Cgroups are also used in container orchestration platforms such as Kubernetes, which provide a high-level abstraction for managing containers and their associated resources. Kubernetes uses cgroups to manage resource allocation and scaling of container applications, providing more efficient and automated container management [5].

Result and discussion

As a result of the above research, we can draw a number of conclusions about containerization technology. Thus, containerization technology has the following advantages, which we have identified:

- Portability: containers are portable, meaning that containerization can be run on any platform or infrastructure that supports the technology. This makes it easy to move applications between development, test and production environments, as well as between different cloud providers or local data centers.

- Consistency: consistency regardless of where the containers provides a runtime environment.

This ensures that applications behave the same across platforms and environments, reducing the risk of compatibility issues and errors.

- Scalability: Containers are designed to be scalable, which means that they can be modified you

can easily scale up or down as required. This makes it easier to scale applications to handle traffic spikes or support a growing user base.

- Efficiency: Containers are lighter and better than traditional virtual machines Uses fewer

resources. This makes them more efficient and cost-effective because they require less hardware and can run more applications on the same infrastructure.

- Security: Containers provide a more secure working environment than traditional applications

provides because they are isolated from the host operating system and other containers. This reduces the risk of vulnerabilities and attacks, making applications more secure and easier to manage.

- Flexibility: Containers allow developers to build complex infrastructure or allows applications

to be built, tested and deployed quickly and easily without the need for customization. This allows organizations to respond more quickly and effectively to market demands and changing consumer needs.

While containerization technology has many advantages, we've also identified some disadvantages that we need to be aware of:

- Security risks: containers entirely from the host operating system non-isolated, which means

that if one container breaks, other containers or the host system could be affected. Organizations must take steps to ensure that adequate security measures are in place to mitigate this risk.

- Limited isolation: containers have the same core as the operating system host, which means

they are not as isolated as virtual machines. This can lead to potential compatibility, performance and security issues.

- Complexity: Installing and managing containerization can be a challenge, especially in large

environments with a large number of containers. Organizations may need to invest in specialized skills and tools to manage container organization, networking and security.

- Resource constraints: containers are more efficient than virtual machines even though they

still require resources such as CPU and memory. If an organization runs too many containers on a single host, it may encounter resource constraints that affect application performance.

- Compatibility issues: containers are operating systems that only support containerization are

compatible with systems, which means they may not work on older or legacy systems.

- Data management: containers are ephemeral, which means they are usually stable Not designed

to store data. Organizations need to think about how to manage and store data created by containerized applications.

Conclusions

In conclusion, while containerization has many benefits, it can be said that organizations must be aware of its potential drawbacks and take steps to mitigate them. Proper planning and management are critical to ensuring that containerization delivers the desired benefits while minimizing any potential risks.

References

1. Jung, Kwang wook, Chao, Yang-Ki, Tak, Yong-Jin. “Containers and orchestration of numerical ocean model for computational reproducibility and portability in public and private clouds: Application of ROMS 3.6”, (2021) Simulation Modelling Practice and Theory, 109 p.

2. Rodriguez-del-Pino, Rubio-Royo, Hernandez-Figueroa. Edited by: Chova, Belenguer, Martinez. “Scalable architecture for secure execution and test of students' assignments in a virtual programming lab”, (2011) Edulearn Proceedings, pp. 4315-4322.
3. Stan, Rosner, Ciocirlan. Edited by: Gasner. “Enforce a global security policy for user access to clustered container systems via user namespace”, (2020) RoEduNet International Conference, pp. 3560-3568.
4. Sveshnikova, Gankevich. Edited by: Smari. “Using virtualization for reproducible research and code portability”, (2017) 2017 International conference on high performance computing & simulation (HPCS), pp. 891-892.
5. Eric Hitimana, Gaurav Bajpai, Richard Musabe, Louis Sibomana and Kayalvizhi Jayavel. Containerized Architecture Performance Analysis for IoT Framework Based on Enhanced Fire Prevention Case Study: Rwanda. (2022) Sensors, 22(17).

https://conference-w.com/canada
can@conference-w.com

ISBN

9 789244 513415