CONTAINERS AND VIRTUAL MACHINES

Abduaziz Ziyodov
Student at Inha University in Tashkent
abduaziz.ziyodov@mail.ru

https://doi.org/10.5281/zenodo.8213488

Abstract

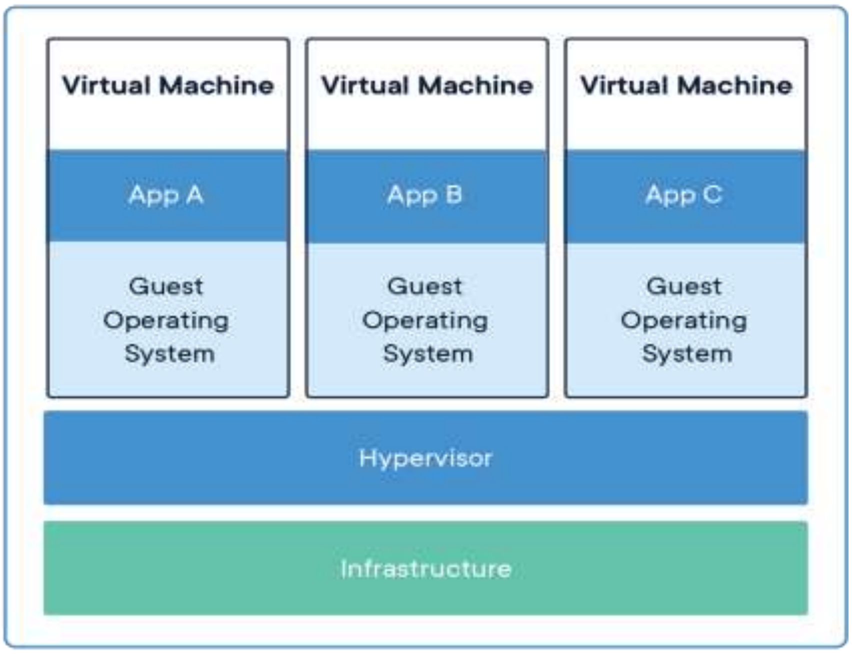
Containers have become dominant in cloud development, and deployment techniques. There are key differences between containers and virtual machines, and their use cases. They both provide complete isolation and make your application independent. This article discusses the common use cases of both technology and key differences and answers the question of how they work.

Keywords
Containers, virtual machines, docker, hypervisor, deployment, system, cloud, DevOps

Introduction

Every software(application) has it is own dependencies, environment, etc. For example, engineers developed a web service that runs on only Linux-based operating systems. It may have something to do with the kernel, or it may have its tuning. Therefore, it should be only deployed/run on Linux-based machines. What if they have to do it with a different type of machine? The answer is there are two kinds of solutions: containers and virtual machines. You don't have to re-write the entire software for another environment.
In other words, those two technologies can package your software into one "box", which is portable, engineers bring or run it everywhere. They solved our traditional problem: "It works only with my machine".

Virtual machines



Virtual machines provide the functionality of a physical computer. There are two types of virtual machines:

- System virtual machines
- Process virtual machines

System virtual machines also known as full-virtualization VMs, act like physical machines. They give full power to execute the entire operating system. It means you can run multiple isolated operating systems in one machine, and we can say that they are duplicates of a real computer machine. Everything is controlled by a hypervisor.

Process virtual machines are designed to execute computer programs in a platform-independent environment. It runs as a normal application inside an operating system and supports a single process. The main aim of process virtual machines is to provide a platform-independent programming environment. That's why we can run Java, .net and Python everywhere. They have their virtual machine, that runs every operating system and interprets your source code. However, the article will focus on full-virtualization/system virtual machines.

Virtual machines run on hypervisors which can be hardware or software-based.

Hypervisor

Hypervisor is a kind of software/hardware that creates, runs and manages virtual machines. The computer that runs the hypervisor is considered as host machine (main computer), and each virtual machine is called a guest machine. There are also two types of hypervisors:

- Native or bare-metal hypervisors
- Hosted hypervisors

Type-1 hypervisors can directly interact with the host machine's hardware, hypervisors can control hardware to manage guest operating systems. The first hypervisor software was developed by IBM in the 1960s. Hypervisors allow guest machines to be created instantly, allowing more efficient utilization.

Nowadays, cloud computing becomes popular, the hypervisors have emerged as an invaluable tool for running virtual machines. Hypervisors are key technology, to make cloud computing possible.

Virtual machines: use cases

1. Build a malware lab

Virtual machines can act like sandboxes. It refers to an isolated space, where any software can run without impacting the host machine. It means you can test malicious software, viruses, etc. It is widely used by cybersecurity analysts.

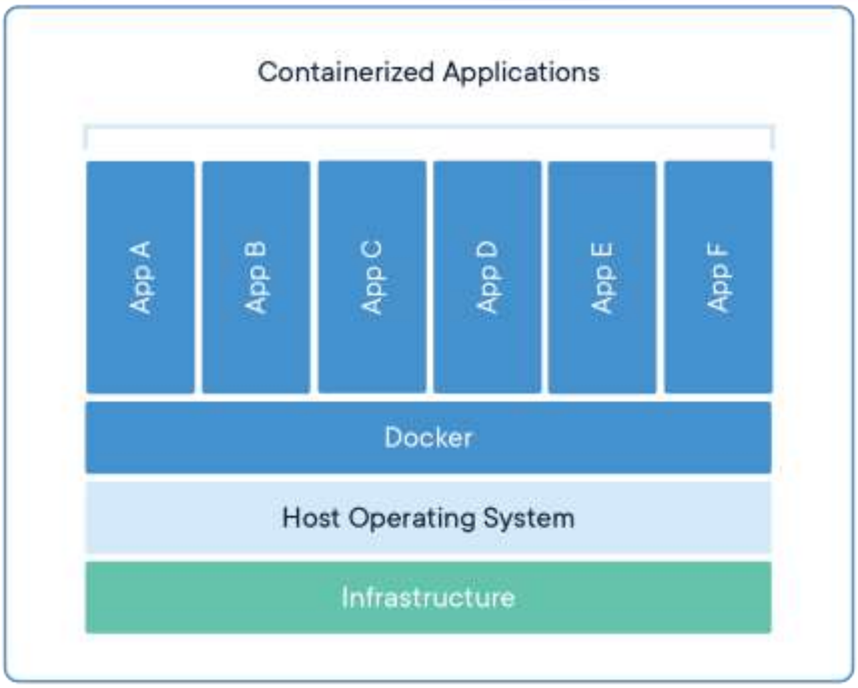
2. Cross-platform development

Imagine you are developing Windows software on your Windows computer. You may want to release the Linux version of this software, but then you need to install the entire Linux on your machine. But, by using virtual machines you can create a Linux environment inside of your host operating system, and then you can test/develop your software.

3. Run outdated software

You can create a virtual version of an old OS, like Windows XP and install your favourite old software to this VM.

Containers



A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings[1].

They first appeared decades ago, but most modern developers remember 2013 as the start of the modern container era with the introduction of Docker.

Containers are more lightweight than compared to virtual machines. They share the host machine's operating system kernel and container files are small. We can spin up quickly, and better support cloud-native applications that scale horizontally. They have similarities with VMs, such as they are also portable and platform-independent. Containers carry all their dependencies, and you can run them everywhere by using a container engine. Also, they have better utilization than VMs.

The main use case of containers is deployment. When you need ultimate portability across multiple environments, using containers might be the easiest decision ever.

Containers: use cases

1. Microservices: containers are small and lightweight, which makes sense. Traditionally, microservices are quite many, and by the small size of containers, we can scale many microservices in an isolated environment.
2. DevOps: by combining microservices and containers, many teams embrace DevOps as the way they build, ship and run the software.
3. Application modernizing and migration.

Virtual machines versus Containers

In virtual machines, hypervisor virtualizes physical hardware where containers instead of virtualising the underlying hardware virtualize the operating system. Every application contains only the application source code and its dependencies. Containers are

[1] https://www.docker.com/resources/what-container, Development, Shipment and Deployment

IBET | Volume 3, Issue 7, July

flexible, and they are perfect for a multi-cloud world.

Containers are also ideal for automation and DevOps pipelines, including continuous integration and continuous deployment implementation. Virtual machines are a popular choice for cloud computing because they can be easily scaled.

Conclusion

Each technology has it is own use cases, problems, and advantages. Engineers should know their architecture, and analyze their needs. Then pick the right choice.

References:

1.https://www.ibm.com/blog/containers-vs-vms/
2.https://www.docker.com/resources/what-container/
3.https://en.wikipedia.org/wiki/Container
4.https://www.vmware.com/topics/glossary/content/hypervisor.html
5.https://en.wikipedia.org/wiki/Hypervisor
6.https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor
7.https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-virtual-machine