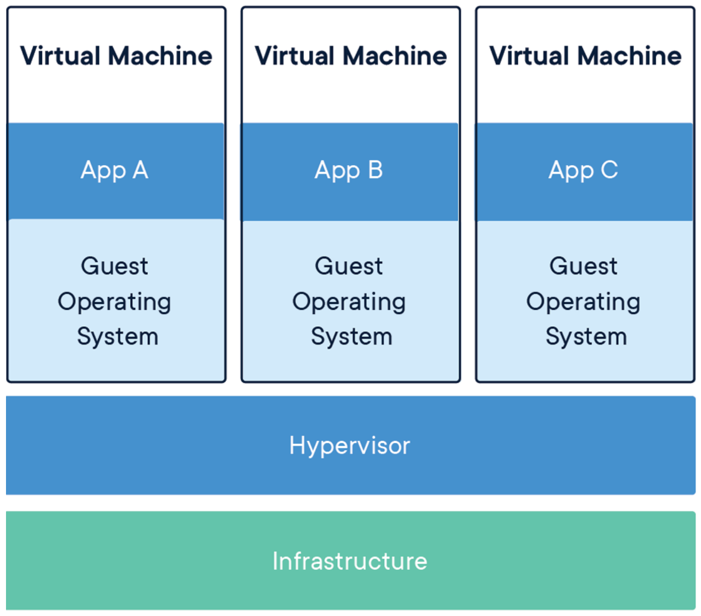
Working mechanism of Virtual Machine

A virtual machine is a platform or software environment that allows us to run one or more computers within a host computer. Each Operating System running inside a VM acts the same as the Operating system of a real computer running in an isolated device. The process of running a virtual instance of a computer system in a layer abstracted from the actual hardware is called virtualization.
The VM utilizes existing resources (CPU, memory, storage) using software to simulate virtual hardware that allows multiple VMs to run on a single machine. The software is called Hypervisor. It also schedules operations in VMs so they don’t overrun each other when using resources. VMs only work if there is a hypervisor to virtualize and distribute host resources. The physical machine is known as the host while the VMs running on it are called guests.



The hypervisor utilizes underlying resources and distributes the resources to each virtual machine it hosts separately. The hosts utilize the resources so as to avoid interference and deadlock, hence each virtual machine is given separate resources (fraction of available resource) for itself. The virtual machine is partitioned from the rest of the system, meaning that the software inside a VM can't interfere with the host computer's primary operating system.

Working mechanism of 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 executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. It is a ‘Develop locally and deploy anywhere’ mechanism, a smart way of managing resources.
Containers are an abstraction in the application layer, whereby code and dependencies are compiled or packaged together. It is possible to run multiple containers on one machine. Each container instance shares the OS kernel with other containers, each running as an isolated process. A sample application, or a microservice, is packaged into a container image and deployed for use through the container platform. The container platform is a client-server software facilitating the execution of the container by providing three key operational components:

1. A daemon is a process that runs in the background. It manages objects like images, containers, and other communication (network), and storage (data volume) objects needed by the microservice encapsulated within the container.
2. An application programming interface (API) allows programs to interact with and direct the daemon process.
3. A command line interface (CLI) client issues commands, like "pull" and "run", and is used to access container images from a configured registry. The command line uses the API to control or interact with the daemon through direct commands, or scripts containing commands. Then the daemon delivers the results through the Host OS for final output.

What problem does Virtualization solve and what is its drawback in context to modern application deployments?

Virtualization helps to run different virtual machines like individual computers with their own operating systems remaining completely independent of one another and isolated from the physical host machine. It helps to run different OS on different VMs at the same time. So it provides a better opportunity for testing. Like testing newly released OS, testing Network procedures and also to test Network Applications.

It helps to support cross platform applications, like running an application on an OS even if the app was unintended for that OS. Overall hardware requirements are also reduced which lessens the total cost. Also, the core technology implemented by cloud computing is virtualization. So we can say that, without virtualization, the cloud computing services would have been costly and almost impossible to serve as per demand.

However there are certain drawbacks of virtualization. The guest OS are isolated from the host OS but their performance might depend upon the host OS. Slight disturbance in performance of host OS might vastly affect the performance of every guest OS hosted by the host device. It needs a host OS compulsorily, which increases storage size and memory requirements.

There may be security issues and probability of risk of data breach while using storage resources with virtualization. Also virtualization is a complex concept. So finding bugs in the system might be a tedious task.

What are the problems Container solves in regard to app deployment and how it solves?

Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another. It could be from a developer's desktop to a client’s desktop, from a staging environment into production, and perhaps from a physical machine in a data center to a virtual machine in a private or public cloud. Since the operating system is shared by all containers, they are much more lightweight than traditional virtual machines. It’s possible to host far more containers on a single host than fully-fledged virtual machines. Another advantage is that containers share a single operating system kernel start-up in a few seconds, instead of the minutes required to start-up a virtual machine. Containers are also very easy to share. Containers also allow developers to control the display of the module or package. The packages and libraries can easily be added or removed.

Rather than run an entire complex application inside a single container, the application can be split into modules (like database, API ,front-end). This is the so-called microservices approach. Applications built in this way are easier to manage because each module is relatively simple, and changes can be made to modules without having to rebuild the entire application. Because containers are so lightweight, individual modules (or microservices) can be instantiated only when they are needed and are available almost immediately. It enhances code reusability too. Once a module is implemented for a container, if we want to create another container then we can recreate the same module with specific changes required.