<u>Qno1</u>
Explain the working mechanism of a Virtual Machine.

<u>Answer:</u>

Virtualization is the process of creating a software-based, or "virtual" version of a computer, with dedicated amounts of CPU, memory, and storage that are "borrowed" from a physical host computer—such as our personal computer— and/or a remote server—such as a server in a cloud provider's datacenter. A virtual machine is a computer file that behaves like an actual computer, typically called an image. It can run in a window as a separate computing environment, often to run a different operating system—or even to function as the user's entire computer experience—as is common on many people's work computers. 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.

<u>Qno2</u>
Explain the working mechanism of Containers.

<u>Answer:</u>

Containerization is defined as a form of operating system virtualization, through which applications are run in isolated user spaces called containers, all using the same shared operating system (OS).

Each container is an executable package of software, running on top of a host OS. A host(s) may support many containers (tens, hundreds, or even thousands) concurrently, such as in the case of a complex microservices architecture that uses numerous containerized ADCs. This setup works because all containers run minimal, resource-isolated processes that others cannot access.

- At the bottom, there is the hardware of the infrastructure in question, including its CPU(s), disk storage and network interfaces.
- Above that, there is the host OS and its kernel – the latter serves as a bridge between the software of the OS and the hardware of the underlying system.
- The container engine and its minimal guest OS, which are particular to the containerization technology being used, sit atop the host OS.
- At the very top are the binaries and libraries (bins/libs) for each application and the apps themselves, running in their isolated user spaces (containers).

<u>Qno3</u>
What problem does Virtualization solves and what is its drawback in context to modern application deployments?

<u>Answer:</u>

The three big problems that virtualization solves are:

Common platform for servers: If all your virtual hosts are running the same virtualization software, all your virtual machines are portable between hosts. If a host goes down, you can run those VMs on other hosts.

Another part of this is supported. If you only have to make sure your apps are supported on your virtual platform, you save a lot of support effort compared to older days where you may have driver conflicts.

Talking of which, you are much less likely to run into platform issues on your VMs since all the hardware is abstracted and virtualized. Running servers bare bones can introduce problems where your OS needs to support all the hardware on the physical server. If you take into account how many possible combinations of hardware there are, you can see why this might eventually lead to a problem. First-hand experience - I see so few server issues running virtualized compared to bare metal servers.

Easy backup: Using native virtual backup software like Veeam makes backing up servers incredibly easy. Restores are also easy as you don't worry about bare metal restores onto dissimilar hardware. You can also leverage virtual-only features like instant recovery.

Consolidation of server hardware: Compute hardware is so powerful these days it's actually a waste of processor and memory resources to run servers bare metal. Even running two or three servers may as well be done on a single virtual host. If you imagine a 1U server can support upwards of 20 good-sized VMs, you are saving as much as 19U rack space. Even taking storage into account, with various high-density options you can still fit more processing and storage resources into less rack space.

The drawbacks of virtualization in context to modern application deployments:

1. It can have a high cost of implementation

   The cost for the average individual or business when virtualization is being considered will be quite low. For the providers of a virtualization environment, however, the implementation costs can be quite high. Hardware and software are required at some point and that means devices must either be developed, manufactured, or purchased for implementation.

2. It still has limitations

   Not every application or server is going to work within an environment of virtualization. That means an individual or corporation may require a hybrid system to function properly. This still saves time and money in the long run, but since not every vendor supports virtualization and some may stop supporting it after initially starting it, there is always a level of uncertainty when fully implementing this type of system.

3. It creates security risks
4. It creates an availability issue
5. It creates a scalability issue
6. It requires several links in a chain that must work together cohesively
7. It takes time

Qno4

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

Answer:

A container consists of an entire runtime environment: an application, plus all its dependencies, libraries and other binaries, and configuration files needed to run it, bundled into one package. By containerizing the application platform and its dependencies, differences in OS distributions and underlying infrastructure are abstracted away.

A container may be only tens of megabytes in size, whereas a virtual machine with its own entire operating system may be several gigabytes in size. Because of this, a single server can host far more containers than virtual machines.

Another major problem solved by Container is that virtual machines may take several minutes to boot up their operating systems and begin running the applications they host, while containerized applications can be started almost instantly. That means containers can be instantiated in a "just in time" fashion when they are needed and can disappear when they are no longer required, freeing up resources on their hosts.

A third solved problem is that containerization allows for greater modularity. Rather than run an entire complex application inside a single container, the application can be split into modules (such as the database, the application front end, and so on). 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.