APRAMEYA L
4NI19IS016
'A' SECTION

Assignment 2

1. Hypervisor vs. Docker

A **hypervisor** is a piece of software that allows multiple operating systems to run on a single physical machine. It acts as a "virtual machine" (VM), creating a layer of abstraction between the hardware and the operating system, allowing multiple VMs to run on the same physical hardware.

Here is an example of a unique use case for a hypervisor:

- A company has a large number of legacy applications that run on outdated versions of Windows. They want to migrate these applications to newer hardware, but they can't update the operating system on the legacy applications because they are critical to the business and can't be taken offline.

In this scenario, the company could use a hypervisor to run the legacy applications on the new hardware. They could create a VM for each legacy application and install the appropriate version of Windows on each VM. The hypervisor would allow the VMs to run on the new hardware, while isolating each VM from the others and from the underlying hardware.

On the other hand, **Docker** is a tool for packaging and deploying applications in containers. Containers are a way to package an application and its dependencies together, so that the application can run in any environment that supports the container runtime.

Here is an example of a unique use case for Docker:

- A company has developed a microservices-based application that consists of many small, independently-deployable services. They want to deploy the application to multiple cloud environments, but they don't want to have to worry about the specifics of each environment.

In this scenario, the company could use Docker to package each microservice into a container. The containers would include everything the microservices need to run, including the code, libraries, and runtime environment. This would allow the company to deploy the containers to any environment that supports the Docker runtime, without having to worry about the specific details of each environment. The containers would run the same way in any environment, making it easy to deploy the application to multiple cloud environments.

APRAMEYA L
4NI19IS016
‘A’ SECTION

2. Containers vs. Virtual machines

Containers and virtual machines are both technologies that allow multiple operating systems to run on a single physical machine. However, they work in different ways and are designed for different use cases.

Here are some key differences between containers and virtual machines:

- Resource isolation: Virtual machines provide strong resource isolation, as each VM runs on its own operating system and is isolated from the other VMs on the physical machine. Containers provide less resource isolation, as they share the operating system of the host and rely on the host's kernel to manage resources.

- Overhead: Virtual machines require more resources to run, as they each need their own operating system and hardware virtualization layer. Containers are more lightweight, as they share the operating system of the host and do not require hardware virtualization.

- Deployment: Virtual machines are typically deployed as a complete package, including the operating system, application, and dependencies. This makes it easy to deploy a VM to any environment that supports the hypervisor, but it also means that VMs can be large and take up a lot of storage space. Containers are deployed as a lightweight package that includes only the application and its dependencies. This makes it easy to deploy containers to any environment that supports the container runtime, but it also means that containers can be more complex to set up and manage.

- Use cases: Virtual machines are well-suited for applications that require strong resource isolation or that need to run on an operating system that is different from the host. Containers are well-suited for applications that are designed to be broken down into microservices, or that need to be quickly and easily deployed to multiple environments.

Taking the same use case as in the first question, both containers and Virtual Machine examples can be provided:
Here is an example of a unique use case for containers:

APRAMEYA L
4NI19IS016
'A' SECTION

- A company has developed a microservices-based application that consists of many small, independently-deployable services. They want to deploy the application to multiple cloud environments, but they don't want to have to worry about the specifics of each environment.

In this scenario, the company could use containers to package each microservice into a container. The containers would include everything the microservices need to run, including the code, libraries, and runtime environment. This would allow the company to deploy the containers to any environment that supports the container runtime, without having to worry about the specific details of each environment. The containers would run the same way in any environment, making it easy to deploy the application to multiple cloud environments.

On the other hand, here is an example of a unique use case for virtual machines:

- A company has a large number of legacy applications that run on outdated versions of Windows. They want to migrate these applications to newer hardware, but they can't update the operating system on the legacy applications because they are critical to the business and can't be taken offline.

In this scenario, the company could use virtual machines (VMs) to run the legacy applications on the new hardware. They could create a VM for each legacy application and install the appropriate version of Windows on each VM. The VMs would allow the legacy applications to run on the new hardware, while isolating each application from the others and from the underlying hardware. This would allow the company to take advantage of newer hardware without having to update the operating system on the legacy applications.