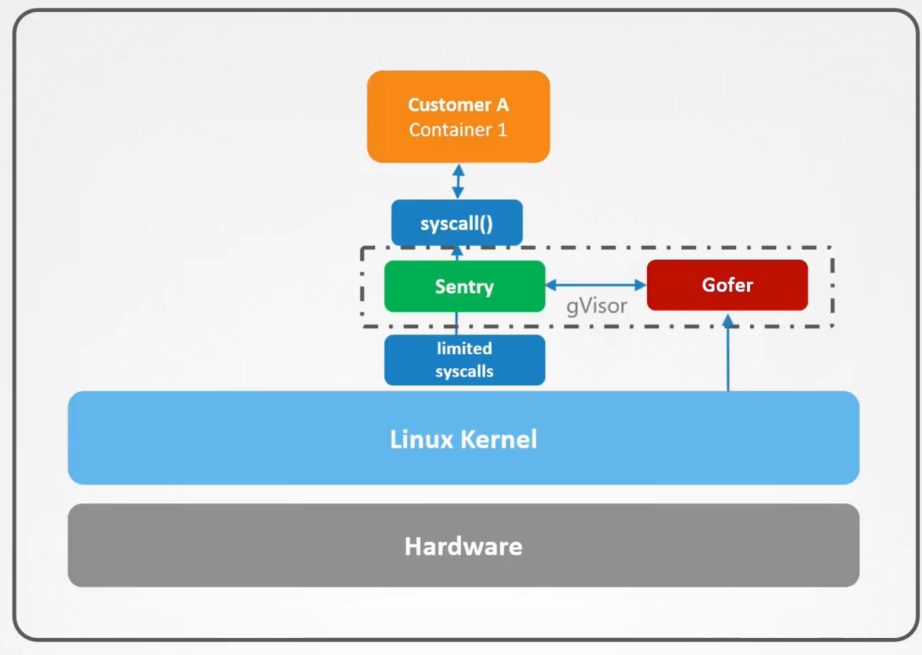
# Container Sandbox

### gVisor

In previous lecture, we can make use of blacklist & whitelist rules tools such as seccomp & AppArmor to allow or deny certain actions that can be done by containers. However, when we have more different type of containers and need to make everything still work correctly, the work loading will be huge.

The Core problem is that the containers interact pretty much directly with the same operating system and especially the same kernel.

So what we actually want is a way to improve the level of isolation between container and container and between container and the operating system or the kernel.

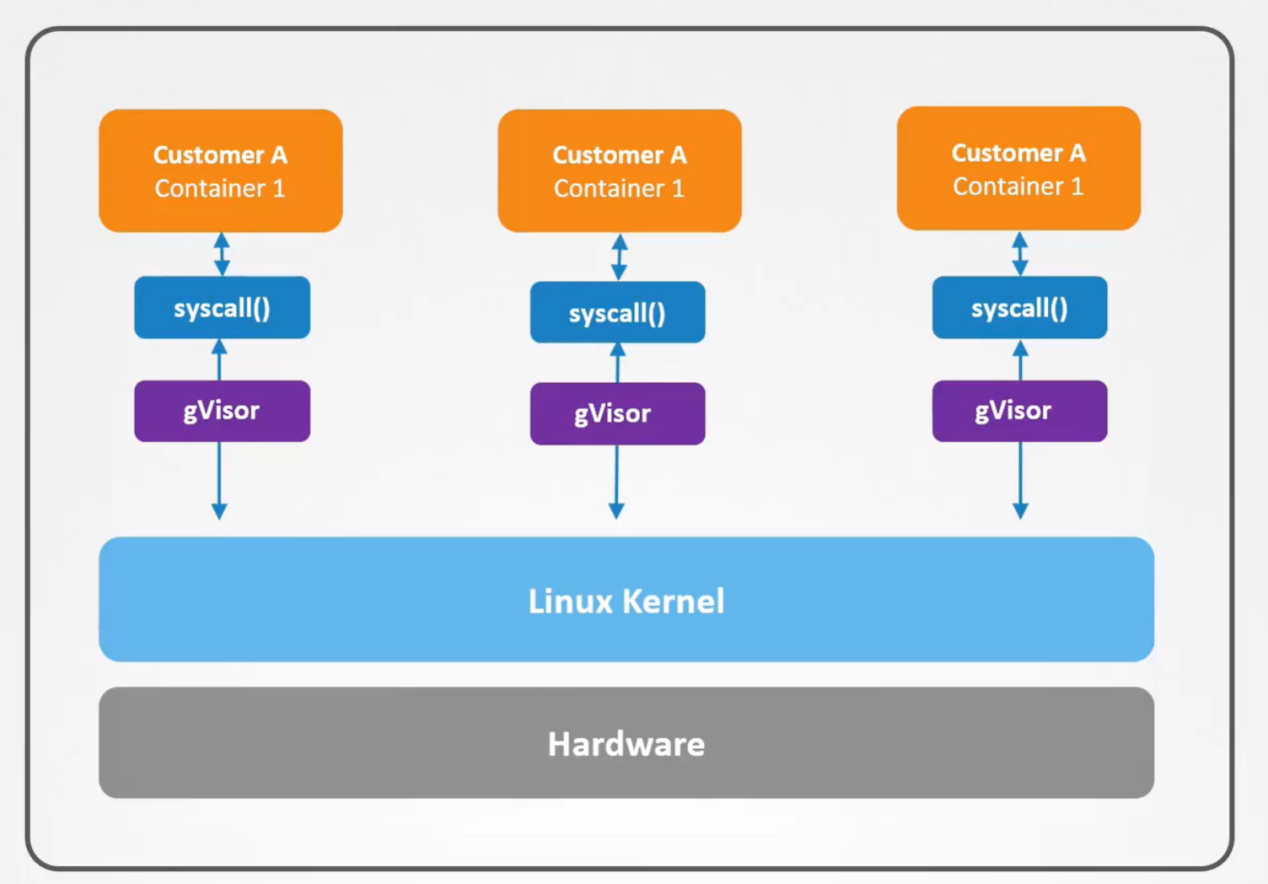


gVisor is a tool from google that allows an additional layer of isolation between the container and host kernel

### **Sentry**

The main purpose of Sentry is to intercept and respond to system calls which are made by containerized application. It supports far fewer functionality than the actual linux kernel.

Gofer is a file proxy that implements the logic that is needed for containerized apps to access the system files.

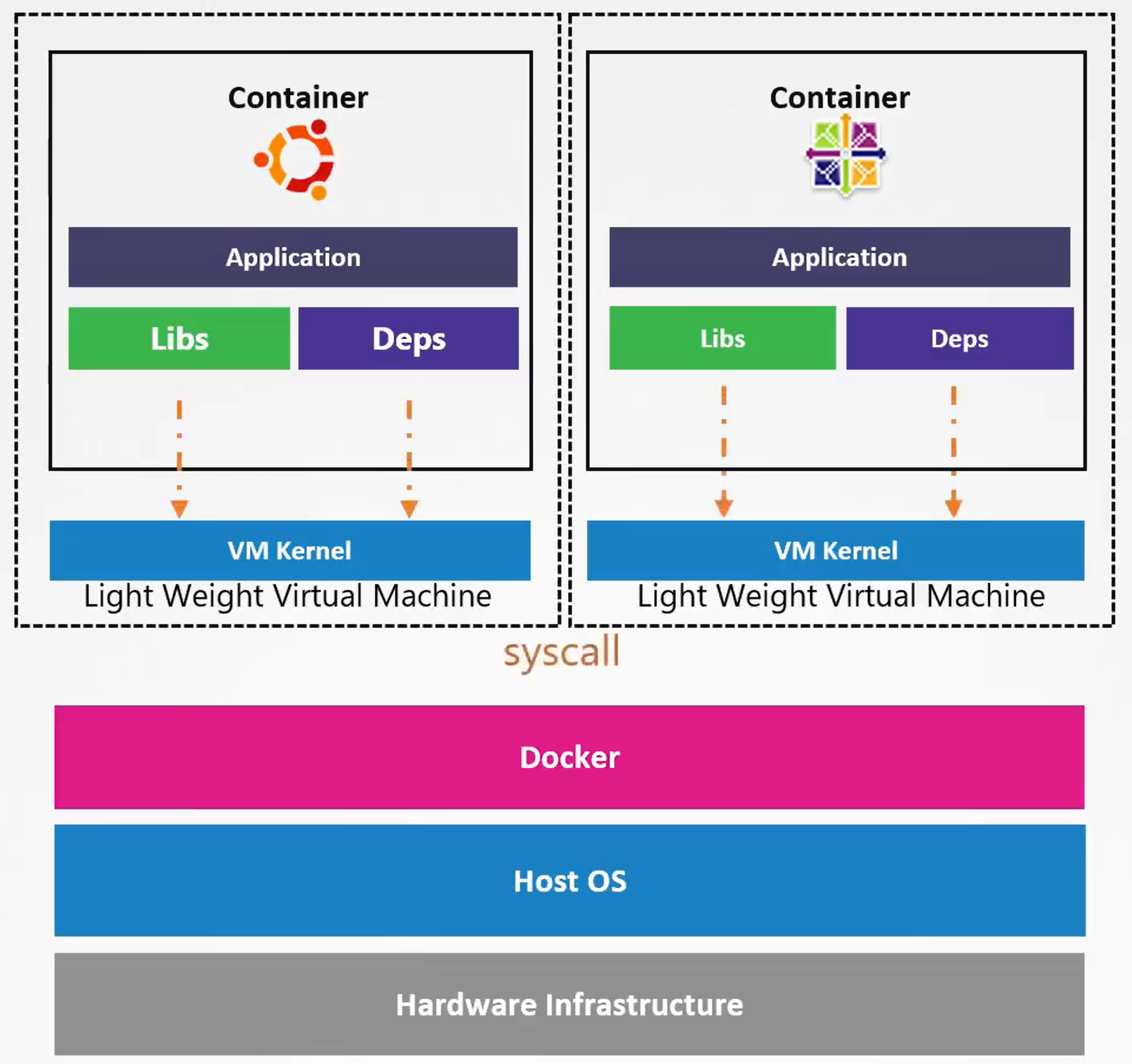


Not one gVisor kernel serves all containers. Each containerized environment has its own dedicated gVisor kernel acting as a middleman between the application and the linux kernel, so that each container would be isolated in its own virtualized sandbox. It will affect all container if one fail.

Disadvantage:

1. Not all apps can work with it, need test each one of application
2. More CPU usage, and lower performance compared with container without gVisor

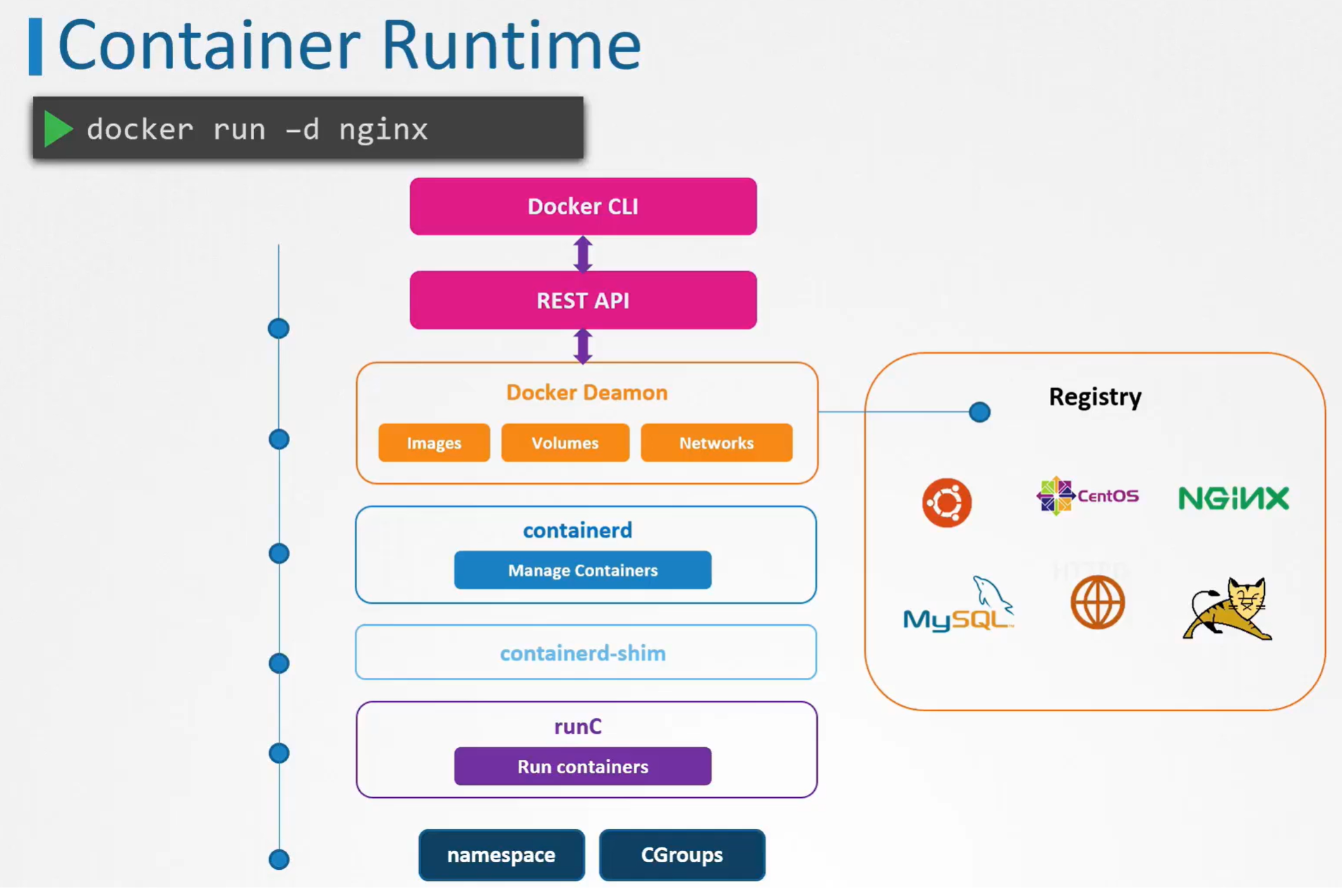
### Kata Containers



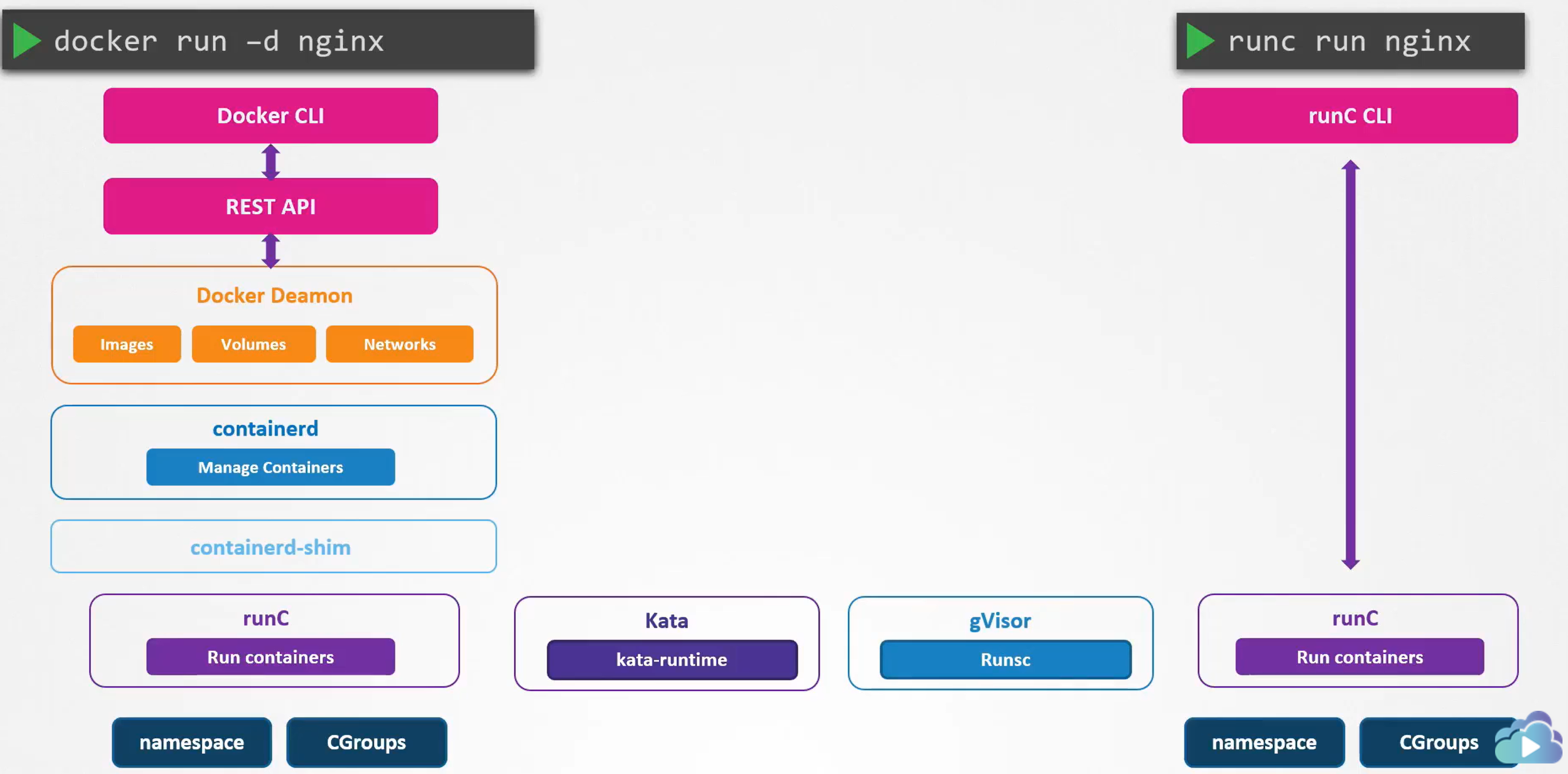
The VMs created by kata are lightweight and more focus on performance, although it’s not as good as bare metal server. Because each one will need slightly more memory and compute resources.

Kata need to run a virtual machine inside another virtual machine. This is called nested virtualization, which is not supported by many cloud providers. Even google cloud is supported, the virtualization performance is usually very poor.

### Runtime Classes

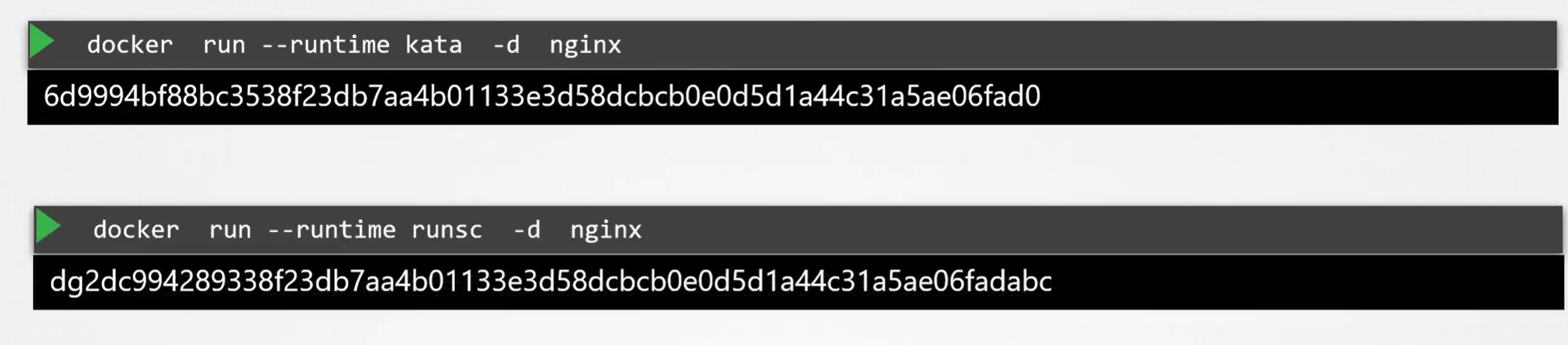


Container runtime need to follow the standard of OCI, which specifies standards around container formats and runtime



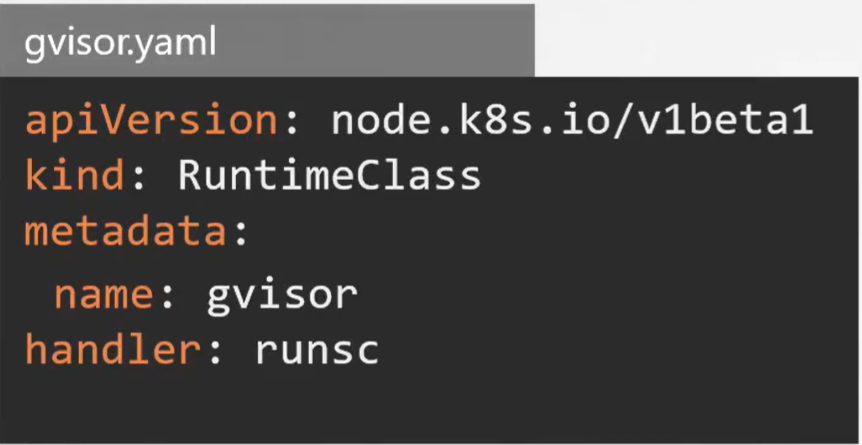
With runc installed on the server, we can now run containers without using docker by making use of runC CLI. However, without the features provided by Docker such as image, volume and network management, it would be difficult to manage the lifecycle of the container.

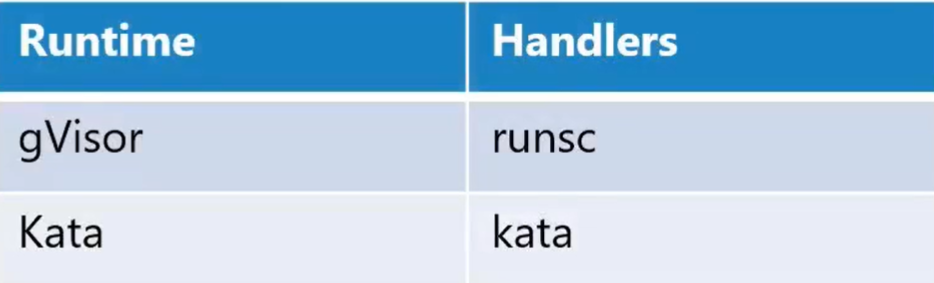
Because Kata-runtime & gVisor-runtime they follow OCI Standard(are OCI-compatible), it is possible to run kata containers and gVisor containers using Docker CLI.



### Using Runtimes in Kubernetes

1. Create RuntimeClass Object in k8s





1. Use RuntimeClass in Pod

