***What is Virtualization?***

Virtualization is the process of creating a software-based, or virtual, representation of something, such as virtual applications, servers, storage and networks. It is the single most effective way to reduce IT expenses while boosting efficiency and agility for all size businesses.

**Virtual Machines**

A VM is a tightly isolated software container with an operating system and application inside. Each self-contained VM is completely independent. Putting multiple VMs on a single computer enables several operating systems and applications to run on just one physical server, or “host.”

A thin layer of software called a “hypervisor” decouples the virtual machines from the host and dynamically allocates computing resources to each virtual machine as needed.

***What is a container?***

Containers use lightweight operating system level virtualization and multiple containers can run on the same host machine. However, containers do not have their own kernel. They share the host machine’s kernel, making them much smaller in size compared to virtual machines. They use process level isolation, allowing processes inside a container to be isolated from other containers.

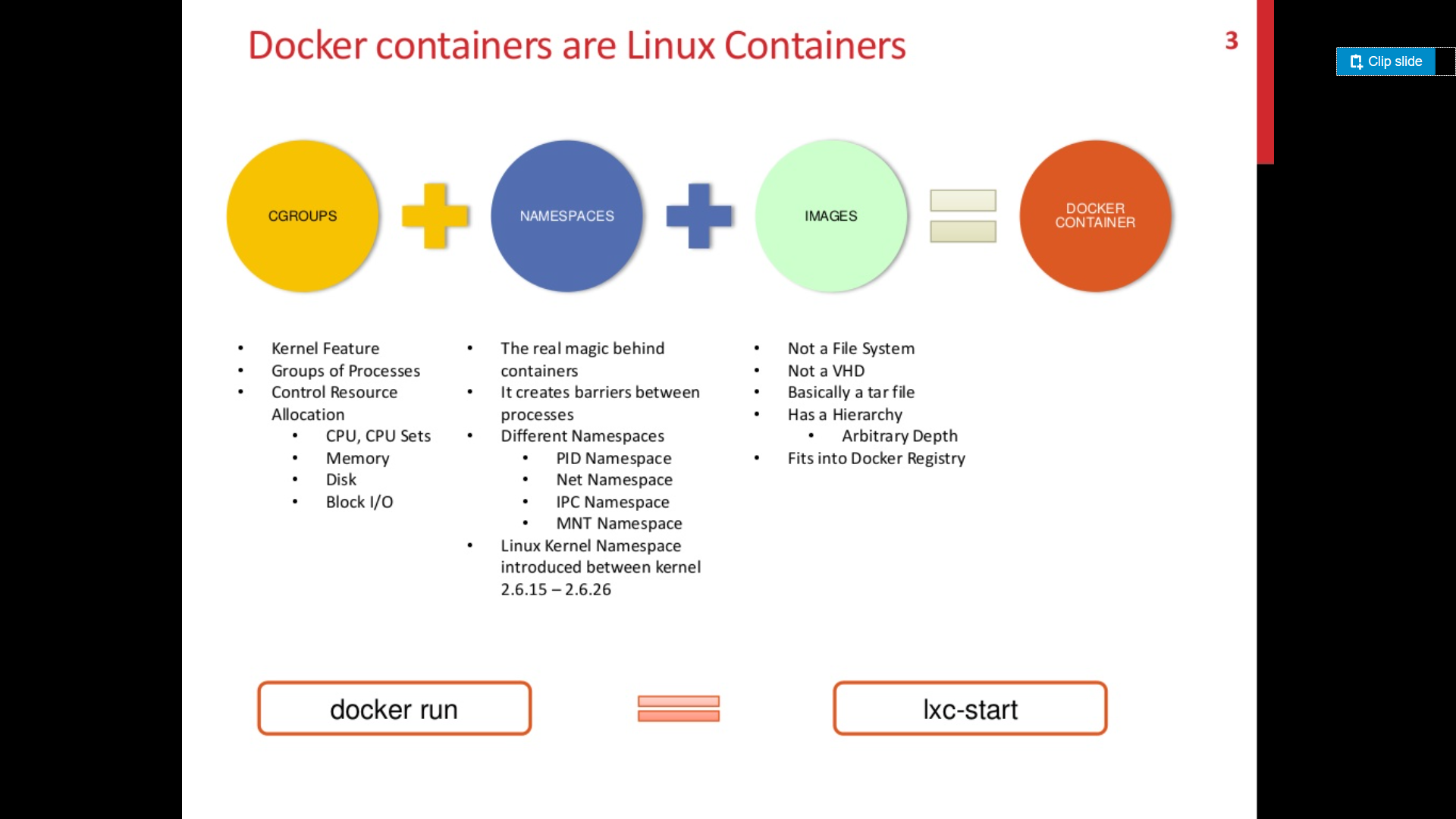
***DOCKER CONTAINERS***

The original Linux container technology is Linux Containers (LXC). LXC is a Linux operating system level virtualization method for running multiple isolated Linux systems on a single host. *Namespaces* and *cgroups* make LXC possible.

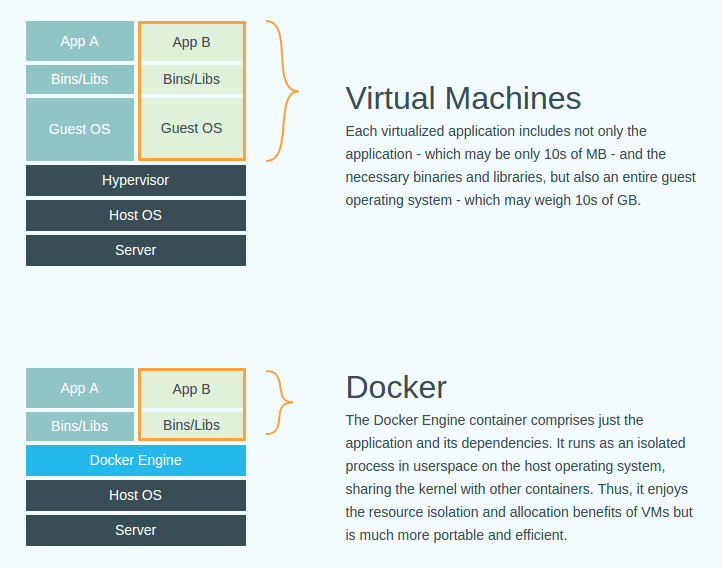
Docker, which started as a project to build single-application LXC containers, introduced several significant changes to LXC that make containers more portable and flexible to use.

Docker is a Linux utility that can efficiently *create, ship,* and *run* containers.

Both Docker and LXC containers *are user-space lightweight* virtualization mechanisms that implement cgroups and namespaces to manage resource isolation.



Docker is an open source tool that lets you to incorporate and store your code and its dependencies into a handy package called an **image**.

[](https://www.google.be/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwiEu63F7rvbAhUBLqwKHd_3BI0QjRx6BAgBEAU&url=http%3A%2F%2Fwww.thevirtualist.org%2Fdocker-questions-answered%2F&psig=AOvVaw1T3TgH9yHUIHazmEFzD3IH&ust=1528265167351482)

**Virtual Machines V/S Containers**

|  |  |
| --- | --- |
| **VM** | **CONTAINERS** |
| Have their own operating system, with kernel, binaries, libraries etc. | Share the host operating system kernel with other containers. |
| OS sizes are in GBs so take a lot of space and are resource intensive. | Lightweight thus smaller in size and less resource intensive. |
| Each VM includes a separate OS image, which adds overhead in memory and storage footprint | Less management overhead as they share a common OS. Only a single OS needs care and feeding for bug fixes, patches, and so on. |
| Less fast and quicker when compared to containers. | Faster, quicker to provision and easy to scale. |