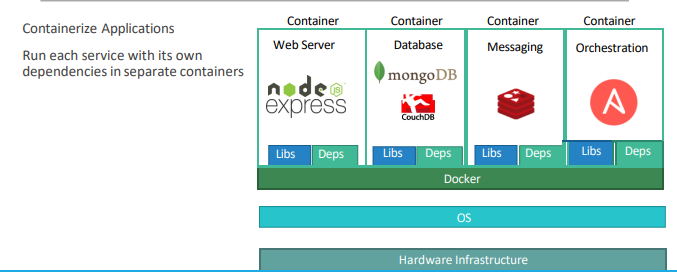
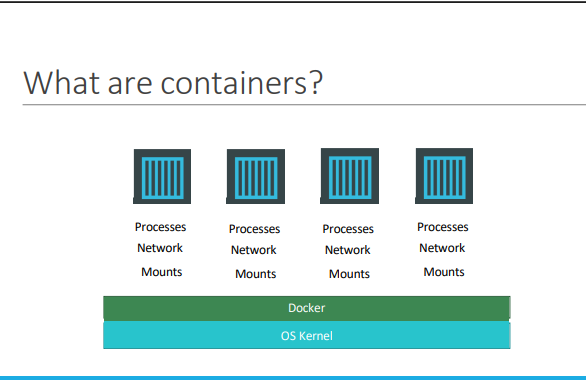
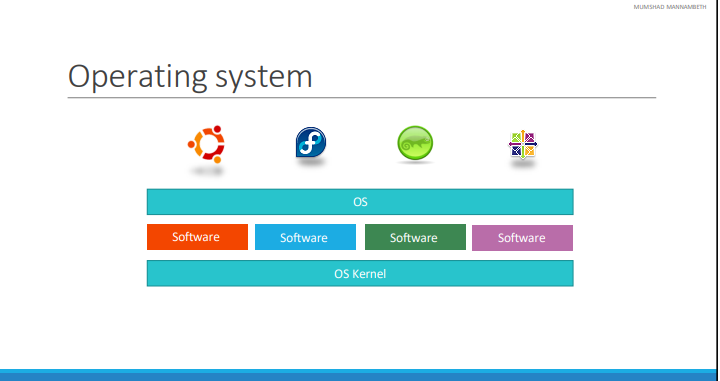
**Container Overview**



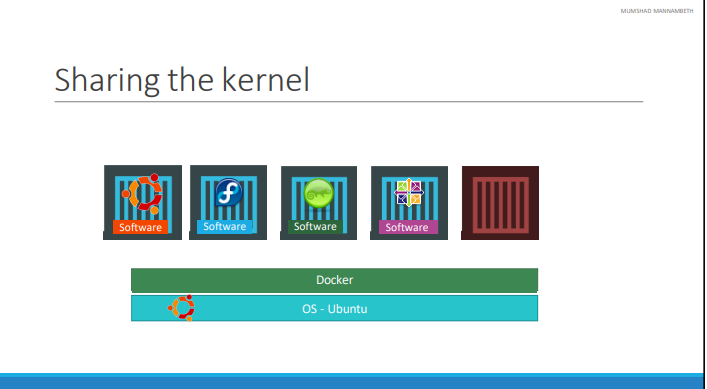
With Docker we are able to run each component in a separate container – with its own libraries and its own dependencies. All on the same VM and the OS, but within separate environments or containers. We just had to build the docker configuration once, and all our developers could now get started with a simple “docker run” command. Irrespective of what underlying OS they run, all they needed to do was to make sure they had Docker installed on their system



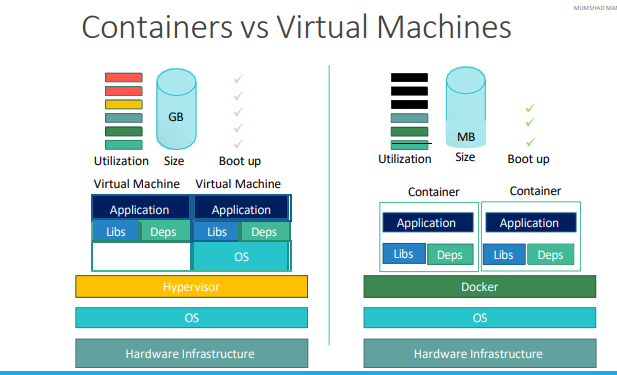
Containers are completely isolated environments, as in they can have their own processes or services, their own network interfaces, their own mounts, just like Virtual machines, except that they all share the same OS kernel



To understand how Docker works let us revisit some basics concepts of Operating Systems first. If you look at operating systems like Ubuntu, Fedora, Suse or Centos – they all consist of two things. An OS Kernel and a set of software. The OS Kernel is responsible for interacting with the underlying hardware. While the OS kernel remains the same– which is Linux in this case, it’s the software above it that make these Operating Systems different. This software may consist of a different User Interface, drivers, compilers, File managers, developer tools etc. SO you have a common Linux Kernel shared across all Oses and some custom softwares that differentiate Operating systems from each other.



We said earlier that Docker containers share the underlying kernel. What does that actually mean – sharing the kernel? Let’s say we have a system with an Ubuntu OS with Docker installed on it. Docker can run any flavor of OS on top of it as long as they are all based on the same kernel – in this case Linux. If the underlying OS is Ubuntu, docker can run a container based on another distribution like debian, fedora, suse or centos. Each docker container only has the additional software ,that makes these operating systems different and docker utilizes the underlying kernel of the Docker host which works with all Oses above. So what is an OS that do not share the same kernel as these? Windows ! And so you wont be able to run a windows based container on a Docker host with Linux OS on it. For that you would require docker on a windows server. You might ask isn’t that a disadvantage then? Not being able to run another kernel on the OS? The answer is No! Because unlike hypervisors, Docker is not meant to virtualize and run different Operating systems and kernels on the same hardware. The main purpose of Docker is to containerize applications and to ship them and run



This overhead causes higher utilization of underlying resources as there are multiple virtual operating systems and kernel running. The virtual machines also consume higher disk space as each VM is heavy and is usually in Giga Bytes in size, wereas docker containers are lightweight and are usually in Mega Bytes in size. This allows docker containers to boot up faster, usually in a matter of seconds whereas VMs we know takes minutes to boot up as it needs to bootup the entire OS

It is also important to note that, Docker has less isolation as more resources are shared between containers like the kernel etc. Whereas VMs have complete isolation from each other. Since VMs don’t rely on the underlying OS or kernel, you can run different types of OS such as linux based or windows based on the same hypervisor.