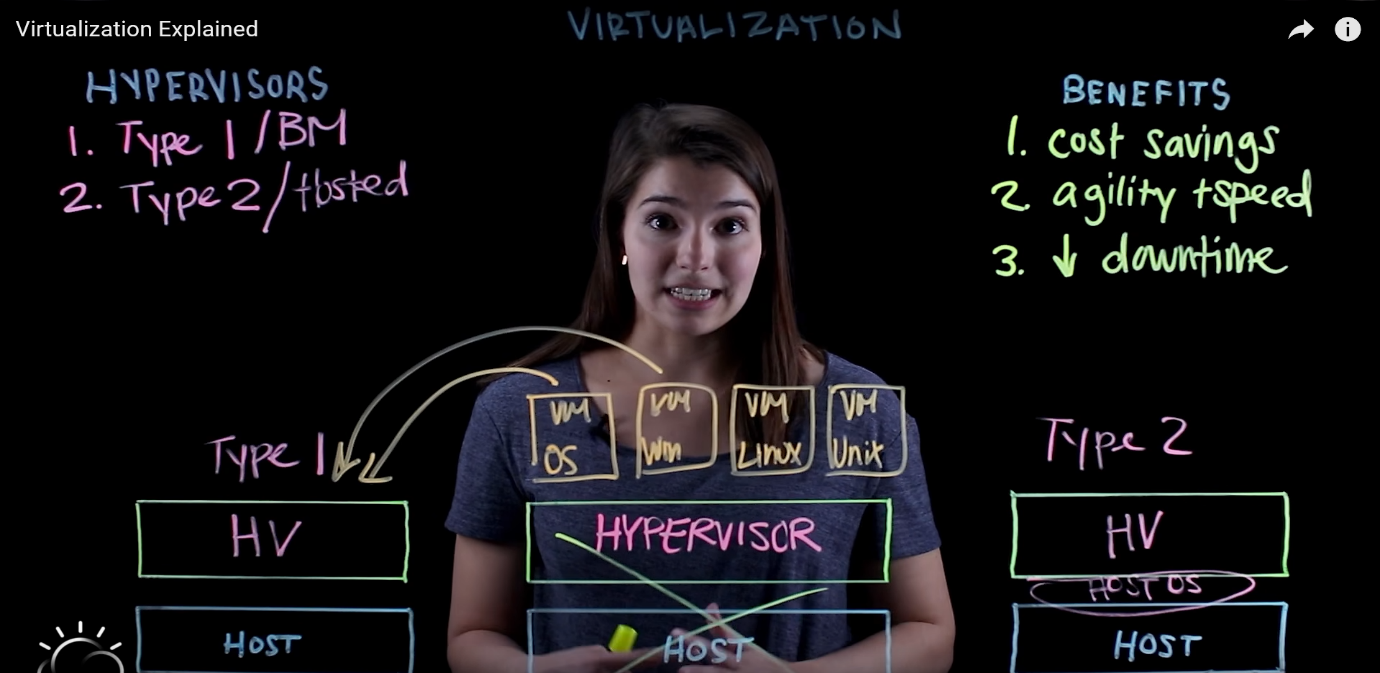
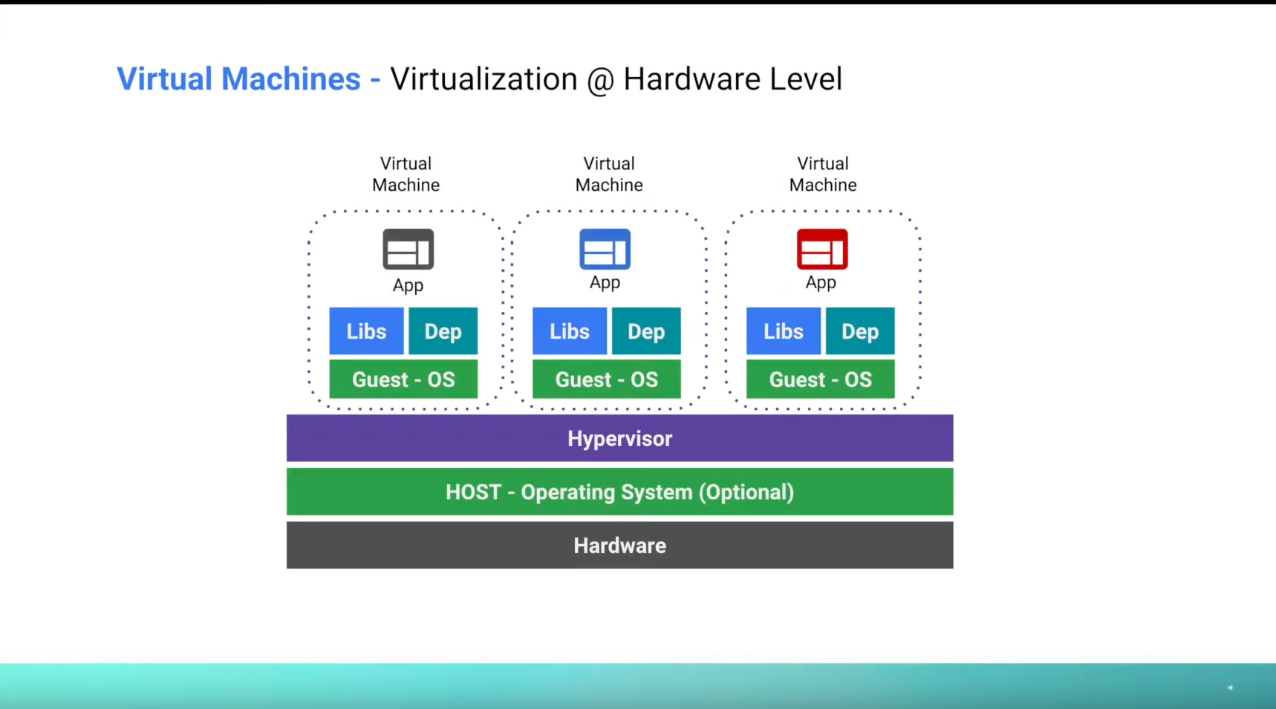
\* Cloud Computing-: Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. The term is generally used to describe data centres available to many users over the Internet.

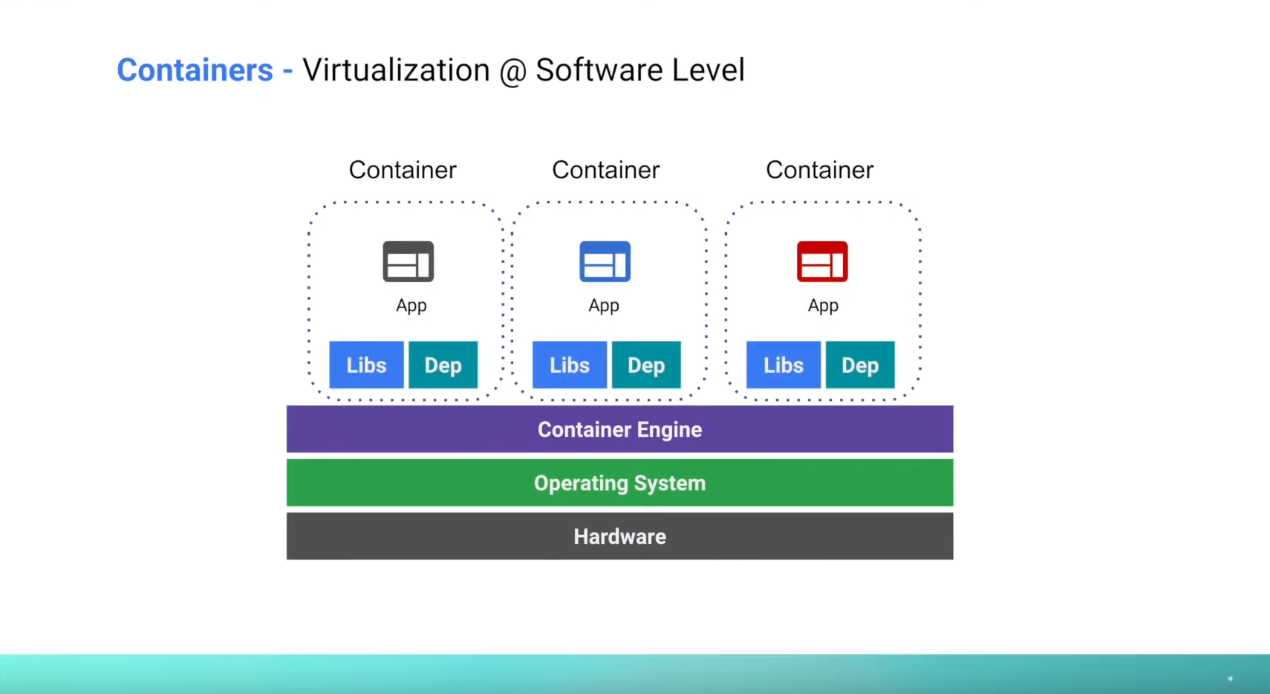
\* Virtualization-: is the process of creating a software-based or virtual, version of something whether that be computing power, storage, networking, servers, or applications. And what makes virtualization feasible is something called a hypervisor. It's simply a piece of software that runs above the physical server or host. They essentially, pool the resources from the physical server and allocate them to your virtual environments.

\* Type 1 hypervisor -: A Type 1 hypervisor runs directly on the host machine's physical hardware, and it's referred to as a bare-metal hypervisor; it doesn't have to load an underlying OS first. With direct access to the underlying hardware and no other software -- such as OSes and device drivers -- to contend with, Type 1 hypervisors are regarded as the most efficient.

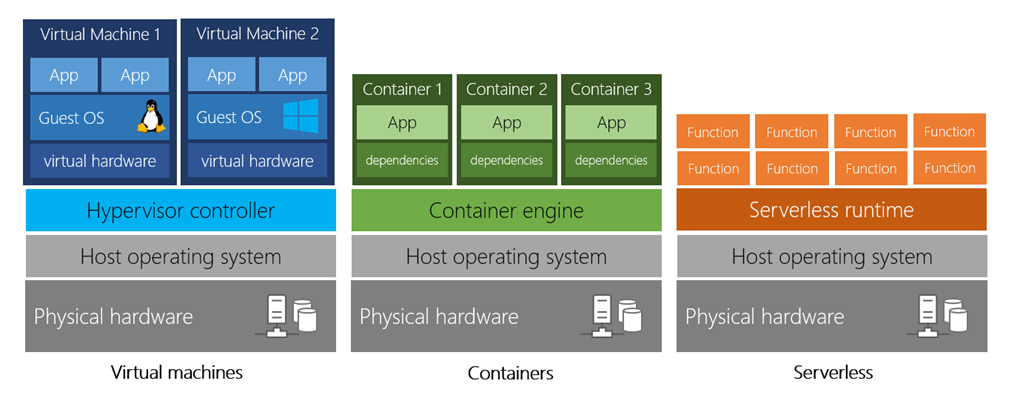
\* Type 2 hypervisor-: A Type 2 hypervisor is typically installed on top of an existing OS, and it's called a hosted hypervisor because it relies on the host machine's pre-existing OS to manage calls to CPU, memory, storage and network resources. 



* Containerization-: Containerization is defined as a form of operating system virtualization, through which applications are run in isolated user spaces called containers. **Containers** provide a consistent, isolated execution environment for applications. They're similar to VMs except they don't require a guest operating system. Instead, the application and all its dependencies is packaged into a "container" and then a standard runtime environment is used to execute the app. This allows the container to start up in just a few seconds, because there's no OS to boot and initialize.



* Serverless Computing-: A serverless architecture allows you to write code (think C#, Java, Node.js, Python, and the likes), set some simple config parameters, and then upload the “package” on to a cloud-based server that’s owned and managed by a third party. Serverless computing is also known as Function as a Service (FaaS), as the company in question is simply requesting the functionality of an external server, leaving them “serverless”, but not functionless.



Differences-:

So, the first thing is the level at which virtualization happens. So, these two technologies are different ways of achieving virtualization, and virtual machines are what's called hardware virtualization because it happens at the hardware level. So, we're going to start with our hardware down at the bottom, because these are computers after all. And what we have on top of our hardware is what's called a hypervisor. And our hypervisor is what's responsible for creating these virtualized instances of each of the components that make up our machines.

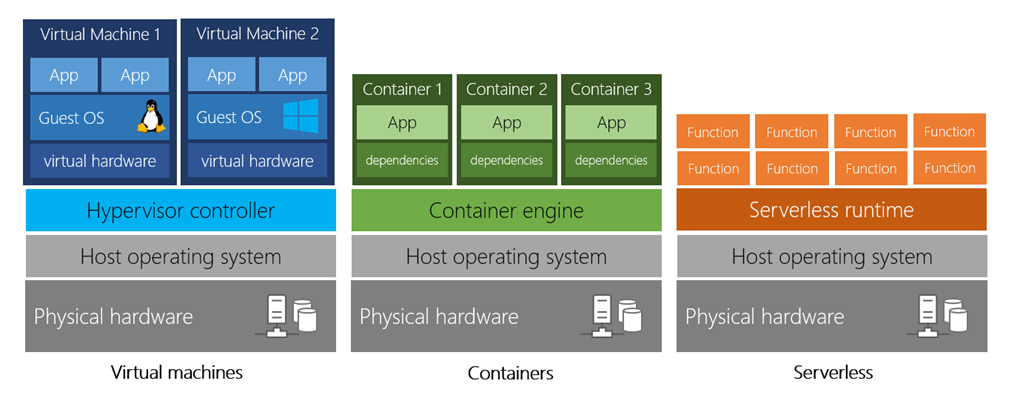
Whereas with containers, we start again with the hardware down at the bottom, but we build them up a little bit differently because we have on top of our hardware our kernel, which is what helps our software and hardware talk to each other. And on top of our kernel we have our operating system and we call it our host OS because it's going to be what's hosting all of our containers. And then on top of the operating system we have each container that's running. And that's why this is called operating system level virtualization because it happens at the operating system level, whereas with our virtual machines we're working at the hardware level.

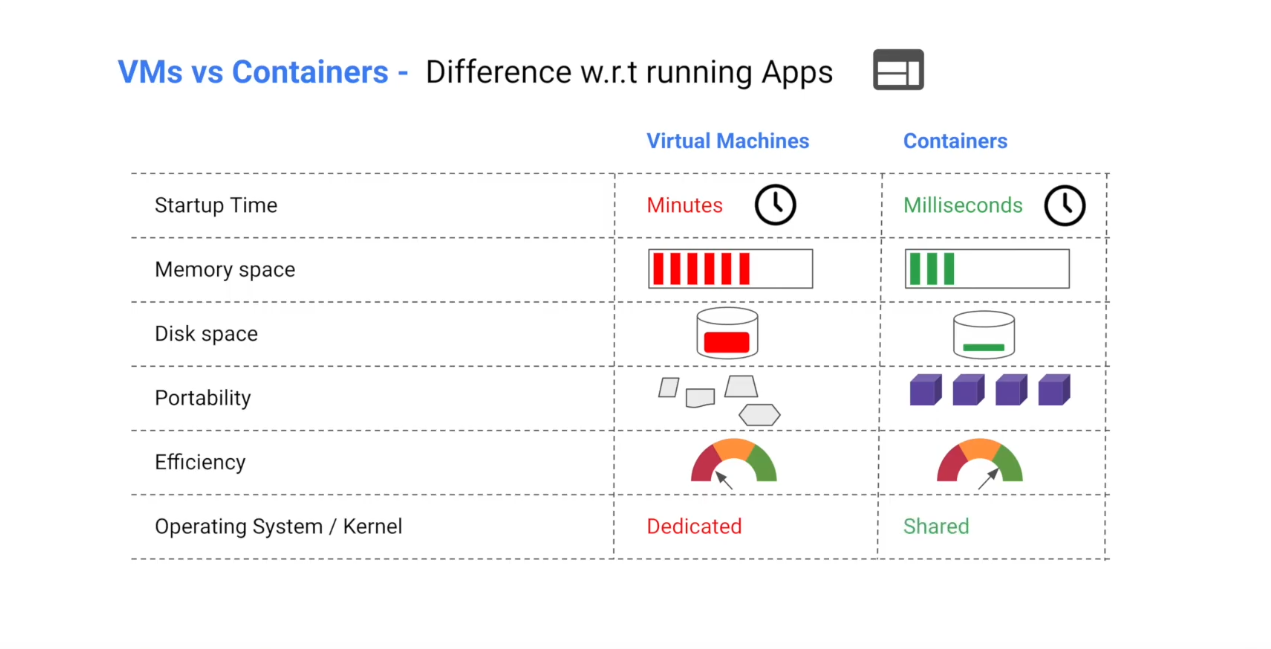
The second thing is about the type of isolation that we're achieving. With our virtual machines we're achieving isolation of machines. So, if we can imagine at our base layer we have one server that's in a rack somewhere, but we want to take our resources and split them up so that we're getting much more use out of what we have. So, we take our hypervisor and we make a machine one, and we make a machine two, and we make a machine three. We're creating what looks like separate workstations, separate servers out of one, we're making our one server look like it's many different machines. And each machine is relatively independent of each other.

Whereas with our containers we're dealing with process isolation. When we're dealing with containers, for security we want to make sure that our applications can only see what's absolutely necessary for them to run and nothing else. And that's what containers allow us to do, where they're sharing the same operating system, they're sharing the same kernel, but it's appearing to each container as if they have their own operating system and only what's installed in them is the libraries that are needed, as well as the scripts, the code, everything that we need to run our applications and that's it.

And the last thing that I want to talk about is this difference in like portability and flexibility.

So with our virtual machines we have what I like to think of as infinite flexibility of our hardware, because we're making a different machine out of our server, we're saying this is how many processors that I want this machine to have, this is how much RAM I would like it to have. And we're able to be flexible about the kind of system that we're building. Whereas with containers we have infinite portability is how I like to think of it, because we have like our container that's being defined in a single file. But we have essentially a few lines of text that are saying exactly how to build our container, how to run our container, what libraries are necessary, what steps to take to build our container up. And we take this one file, we run it on our machine, we can spin up our application. We store it in, we store it in a repository somewhere else and then we're able to run that on different machines. We can take our one file and be able to make it run pretty much wherever, there's no hardware limitations anything like that.





Hybrid Container Architecture-: 