2. Evolution of Cloud Computing

We shall look at the different layers of "compute" (yes, cloud compute, is how it is referred to).

Compute has evolved through the following major stages:

1. Dedicated Servers
2. VMs (Virtual Machines)
3. Containers
4. Functions.

// Now we shall look at each of them in some detail.

Dedicated

A physical server WHOLLY utilised by a ***single*** customer.

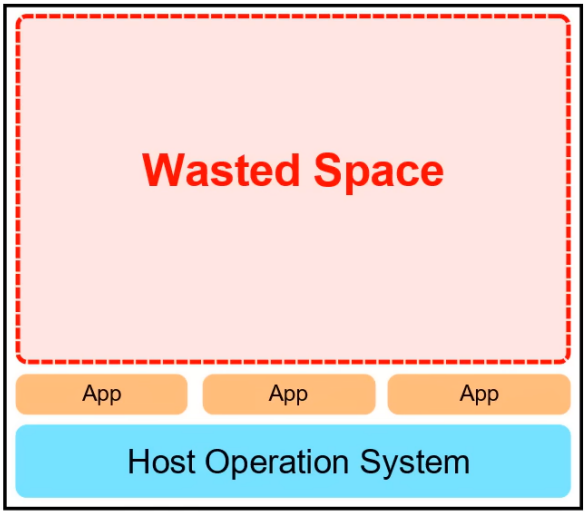
There is a guessing factor to this, as the organisation has to guess their capacity, and then potentially overpay for an underutilised server. (This is also, an example of CAPEX)

Upgrading beyond the present capacity is slow and expensive, since a new machine has to bought and data transferred to it for a Scale Up. (Vertical Scaling)

OS limitations, and resource sharing conflicts, due to multiple apps running on one physical machine.

> *However, a guarantee of security and privacy is given to the organisation.*
> *Also, full control of everything, and customisability.*

(AKA Bare-metal)



physical server

VMs

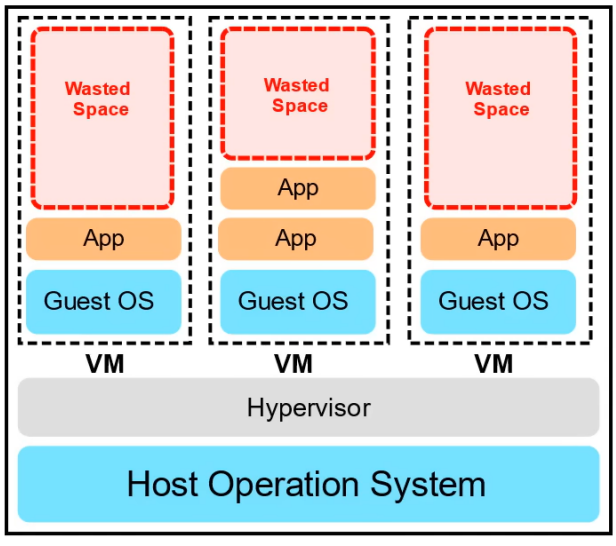
One physical machine (server) can have multiple Virtual Machines running on it.

Hypervisor is the **software layer** that allows a machine to be split into VMs.

Now the physical server is being shared by multiple customers, and this makes the costs be a fraction of dedicated servers.

Still might overpay for an underutilised VM, since here too, you have to guess from a pre-made size offering on the system.
Again, resource conflicts are common here as well.

Limitations by Guest Operating systems.



physical server

Containers

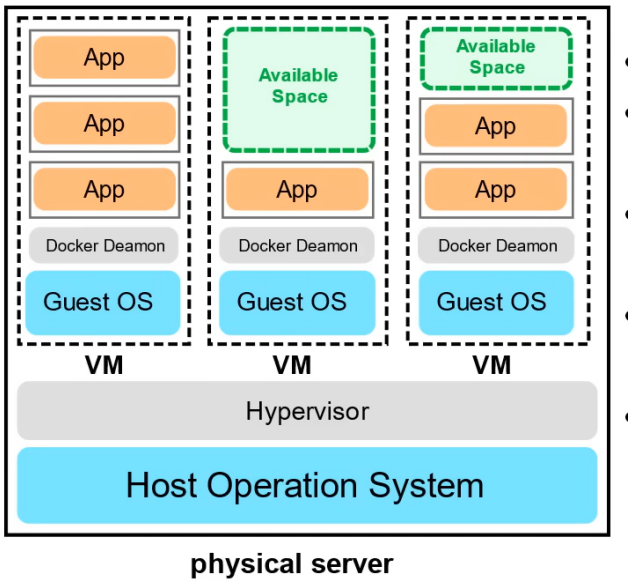
Here, virtual machines run multiple containers inside themselves, and multiple VMs run on the physical machine.

Wasted space is lesser here, and utility of the available capacity can be maximised, making it more cost effective.

An example of a Software that allows for containerisation in VMs is ***Docker.***

Docker Daemon is the name of the ***software layer*** in this case that allows running of multiple containers.

One organisations containers share the same ***underlying*** OS, but at the same time, each container can HAVE A DIFFERENT OS. So, containers are MORE EFFECTIVE than multiple VMs, since multiple apps can be run side by side without being limited by OS requirements, and this significantly reduces resource sharing conflicts.



Functions

Breaks down the idea of utilising space for required functionality further than Containerisation, and provides an even smaller unit for clients to use, i.e, functions.

Basically, the servers , VMs and Containers are all managed by the Compute Provider, and the only thing that the organisation has to do is take the appropriate functions and upload Code and data to it, choosing the memory and duration required, making it extremely cost effective.

Everything else is the Provider's responsibility (which makes management of space even more efficient).

AKA ***Serverless Compute***

> *The downside here is the concept of* `Cold Starts` *.*
>
> *When a function is launched, it takes a while to start because mostly, cloud service providers won't be running servers if there are no processes running on it. So, it takes a while for the server to go active, and then the functions start working as intended.*