Virtual machines:

A virtual machine, commonly shortened to just **VM**, is no different than any other physical computer like a laptop, smart phone, or server.

Definition

A virtual machine is a virtual representation, or emulation, of a physical computer.
They are often referred to as a guest while the physical machine they run on is referred to as the host.
Virtualization makes it possible to create multiple virtual machines, each with their own operating system (OS) and applications, on a single physical machine.

Question 1 :- Working mechanism of Virtual Machine.

The fundamental idea behind a virtual machine is to abstract the hardware of single computer (the CPU, memory, disk drives, network interface cards and so forth) into several different execution environments, thereby creating the illusion that each separate execution environment is running its own private computer.

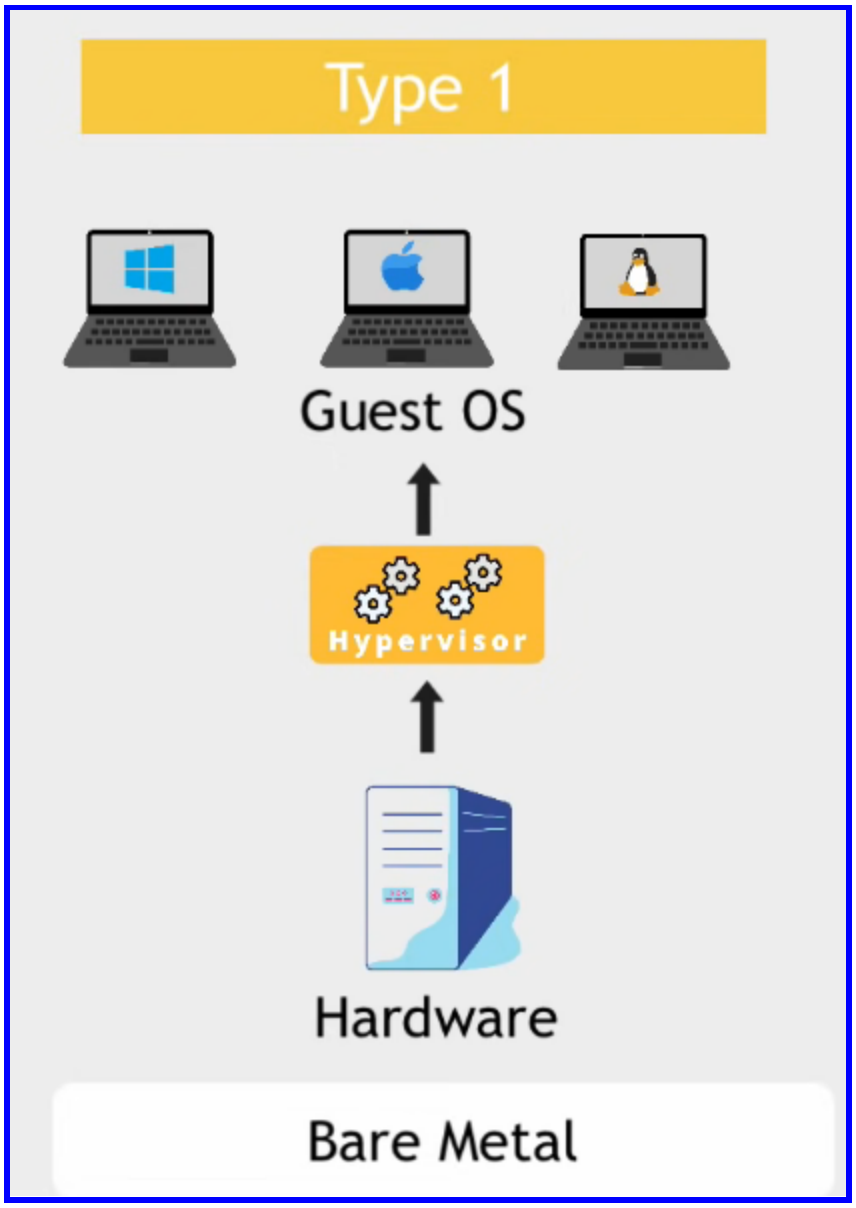
- **Hypervisor** makes virtualization feasible.

Hypervisor is simply a piece of software that runs above the physical server or host.
There are two main types of hypervisor.

  - Type 1 Hypervisor and
  - Type 2 Hypervisor

- Hypervisor pool the resources from the physical server and allocate them to our virtual environments.

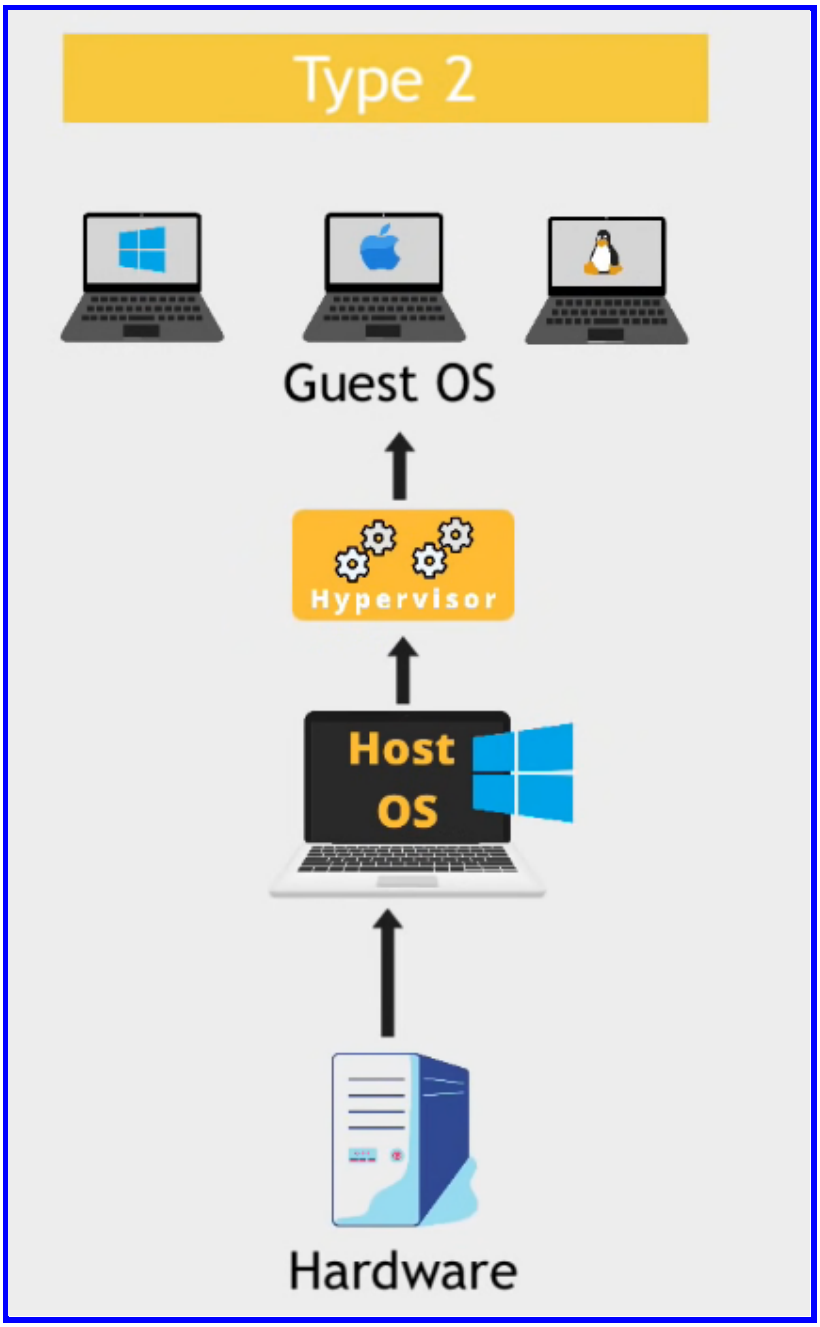
Type 1 hypervisors are hypervisors that are directly installed on the top of any physical server. Also called **bare metal hypervisor**.
Hypervisor controls the hardware resources instead of interacting with the host OS.
One physical server with a bare metal hypervisor installed on hardware and multiple virtual machines running on that hypervisor, all sharing the same hardware resources
Eg: vmware, ESXi, microsoft's hyper-v
Type one Hypervisor is used by big companies and big cloud platforms to create and run their infrastructure. Eg. AWS, Digital ocean, etc



In **Type 2 hypervisors** there is a layer of host OS sits between the physical server and hypervisor. They are also called **Hosted hypervisors.**

Guest OS borrows hardware from the Host OS.

Mostly used for end-user virtualization(PC)



- Once a hypervisor is installed we can build virtual environments or virtual machines. We can run multiple VMs on a hypervisor.
- Hypervisor manages the resources that are allocated to these virtual environments from the physical server.

We can run different OS(Windows, Linux, MAC OS) on different VMs.

- Hardware resources are shared, you can only give resources you actually have.

This is the core virtualization as a process.

Containers

Definition

Containers are considered as a box of ships, which makes it easier to ship a particular code from one machine to another machine.

A container is an isolated, lightweight for running an application on the host operating system.

Question 2 :- Working mechanism of Containers

Containers are built on top of the host operating system's kernel and contain only apps and some lightweight operating system APIs and services that run in user mode.

It is three step process

- It needs some kind of **manifest** that describes the container itself.
  Like **Dockerfile** in Docker World. **YAML manifest** in Cloud Foundry
- **Creating Actual Image.**
  Like Docker Images in Docker World, ACI in Rocket
- **Running Actual Container.**
  Which contains all the runtimes and libraries, binaries needed to run an application

When any instruction is given, runtime

- Daemon builds/pulls the image from Container Registry
- Creates a container from that image which runs the executable that produces output and
- The Daemon again streams that output to the client.

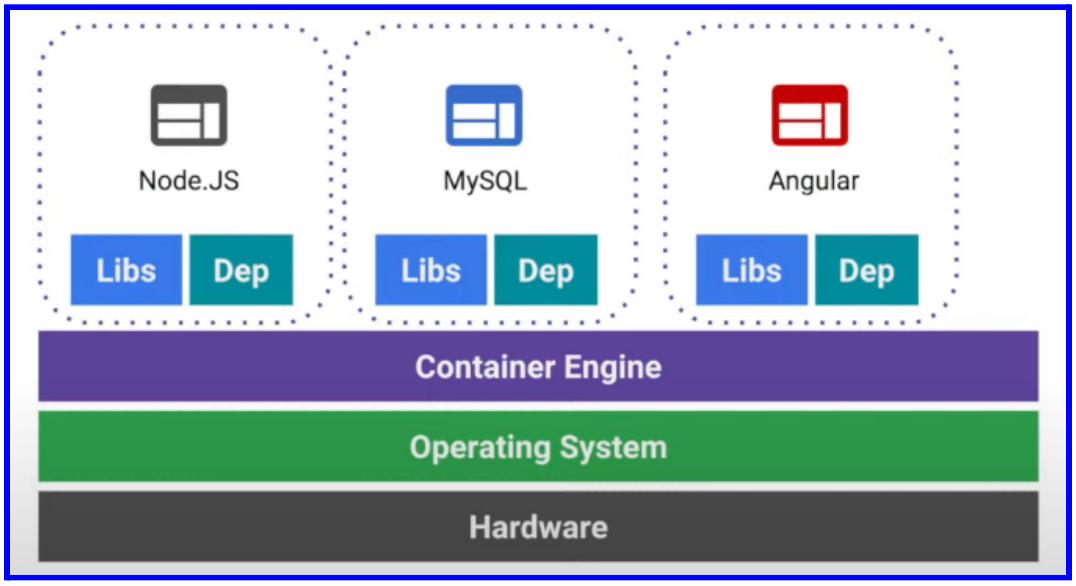


Figure: Architecture of Containerization

3. What problem does Virtualization solve and what is its drawback in context to modern application deployments?

Before virtualization when a company had a server like their own database server or a server where Jenkins were running, Company manage their own server

When OS is directly installed on the hardware then the OS is tightly coupled to the hardware that means

- If the hardware component of that computer failed (hard disk, motherboard failed,etc) and is not repairable any more,
- Then your whole computer will be useless, and the OS, applications installed and all the data will be gone.

With Virtualization we can have our OS as a portable file that we can move around and these files (Virtual Machine Images)

So our OS, application installed on it, all the configurations and all the data will be inside that portable file. And we can even make copies of Image files as a purpose of backup.

More benefits can illustrated as:

Cost savings:

We can run multiple virtual machines on one piece of infrastructure. We don't have to maintain nearly as many servers.

Learn and Experiment

- You don't need to buy a new computer
- You don’t endanger your main os

Test your app on different OS

Example: if we are developing a website and we want to see how that works and how it looks like in different OS in different browsers i.e how your application performs in linux machine in a firefox browser or on windows in internet explorer browser.

And when you are done we can easily delete them.

Efficient uses of Hardware resources

- Use all resources of a performant big server.
- Users can choose any resource combinations.

Each VMs runs in its own isolated environment i.e. completely independent of each other.

Issues and incompatibilities

If something breaks inside VM, it does not affect the host machine

Agility and speed

Spinning a virtual machine is relatively easy and quick. And helps developers for dev-test scenarios.

Lowers the down time

If the host goes down we can simply migrate VMs from one host to another.

Abstraction of OS from the hardware

VMs' drawback in context to modern application deployments

In context of Modern Application Deployments, VMs are losing popularity when IT services needs :-

- **Faster development and release**

Since for installing any application we need multiple commands, checks for multiple dependencies, resolve the conflict of libraries if any and we have to repeat the same process if we are installing the same application in a different environment(development, QA, Production).

- **High Scalability**

Workload operation rarely consumes all the resources made available to associated VM. As a result, the remaining unused resources may not be incorporated in capacity planning abd distribution across all VMs and workloads. i.e. significant resource wastage

- **Flexibility to evolve application development in response to changing business and market needs.**

Monolithic app development practices are losing popularity and organizations are pursuing infrastructure architecture solutions to further optimize hardware utilization. This is precisely why containerization was invented and gained popularity as a viable alternative.

4. What are the problems Container solves in regard to app deployment and how it solves?

In traditional classical deployment scenarios, we have hardware with an OS where we run multiple applications.

Eg: node.js that has certain dependencies, needs certain libraries and specific kinds of operating system. Each software application will have its own library and dependencies.

Sometime there might be a conflict between libraries of different application, such as version issues of same library for two applications

Containers help to solve this problem as it provides an independent isolated environment which has its own libraries, dependencies to run that application.

i.e. if we want to run MySQL applications with different versions, we can easily pull MySQL 5.7 version image and run a container and we can run another container of MySQL with latest version 8 with another image.

Containers help build our application, ship the application, deploy and scale the application with ease and independently.

Container helps to solve other problems too as illustrated :-

Time consuming and multiple commands:

Containers solve this problem since everything is packaged and we can just move it and the run a single command which automatically installs all the dependencies and libraries without conflicting with others

Portability

Although we have images of Virtual Machines (**VMI**), it is not easy to ship the VMIs to another machine as compared to container shipping.

We can easily push our container images to a container registry which is light weighted then VMI(which contains the entire OS). and the Images can be pulled and run containers by one command.

High Scalability

If container processes aren't utilizing the whole CPU or memory, all of the shared resources become accessible for the other containers running within that hardware. So with container-based technology, we can truly take advantage of cloud-native-based architectures.

Operating System Updates and Upgrades

We can easily update and upgrade OS files within a container. Need to edit container image's files (DockerFile) to point to the latest version of the windows base image. Rebuild the container Image, push the container image to the container registry and redeploy the containers.

Dependency conflict

Due to its own isolated environment and its own libraries and dependencies, it sweep away the dependency conflict during running a container

Packaging not easy

Since Image files are the package of libraries and dependencies needed to run an application, so containers makes packaging easy,

Duplicate efforts and compatibility issues

Since these image files are easy to ship through container registry, duplicating efforts become easy and compatibility issues are solved because containers accept cross platform execution. All you need is Containerization runtime Daemon

Containers offer a lot of flexibility and ease in how we can manage our application.