CHENNAI INSTITUTE OF TECHNOLOGY
(Autonomous)
(Affiliated to Anna University, Approved by AICTE, Accredited by NAAC & NBA)
Sarathy Nagar, Kundrathur, Chennai – 600069, India.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CCS335-CLOUD COMPUTING

LECTURE NOTES UNIT 3

UNIT III VIRTUALIZATION INFRASTRUCTURE AND DOCKER

UNIT III VIRTUALIZATION INFRASTRUCTURE AND DOCKER

Desktop Virtualization – Network Virtualization – Storage Virtualization – System-level of Operating Virtualization – Application Virtualization – Virtual clusters and Resource Management – Containers vs. Virtual Machines – Introduction to Docker – Docker Components – Docker Container – Docker Images and Repositories.

1.Desktop Virtualization

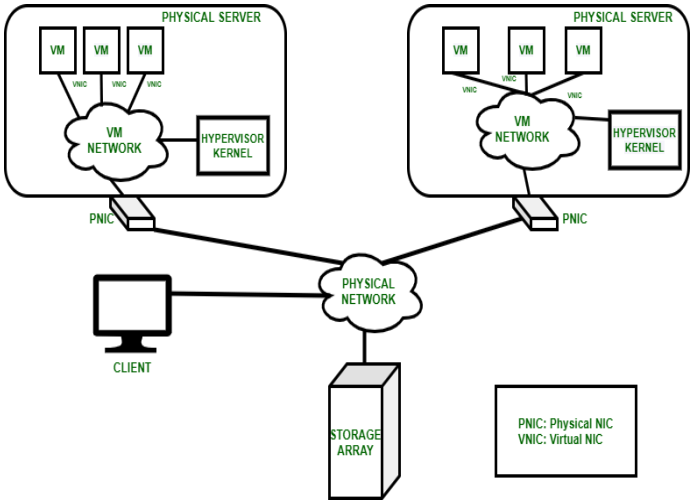
- ➔ Desktop virtualization abstracts the desktop environment available on a personal computer in order to provide access to it using a client/server approach.
- ➔ Desktop virtualization provides the same outcome of hardware virtualization but serves a different purpose. Similarly to hardware virtualization, desktop virtualization makes accessible a different system as though it were natively installed on the host but this system is remotely stored on a different host and accessed through a network connection.
- ➔ Moreover, desktop virtualization addresses the problem of making the same desktop environment accessible from everywhere.
- ➔ Although the term desktop virtualization strictly refers to the ability- environment, generally the desktop Although the term desktop virtualization strictly refers to the ability to remotely access a desktop environment, generally the desktop environment is stored in a remote server or a data center that provides a high availability infrastructure and ensures the accessibility and persistence of the data.
- ➔ In this scenario, an infrastructure supporting hardware virtualization is fundamental to provide access to multiple desktop environments hosted on the same server.A specific desktop environment is stored in a virtual machine image that is loaded and started on demand when a client connects to the desktop environment.This is a typical cloud computing scenario in which the user leverages the virtual infrastructure for performing the daily tasks on his computer. The advantages of desktop virtualization are high availability, persistence, accessibility, and ease of management.

The basic services for remotely accessing a desktop environment are implemented in software components such as Windows Remote Services, VNC, and X Server.

Infrastructures for desktop virtualization based on cloud computing solutions include Sun Virtual Desktop Infrastructure (VDI), Parallels Virtual Desktop Infrastructure (VDI), Citrix XenDesktop, and others.

2. Network Virtualization

Network Virtualization is a process of logically grouping physical networks and making them operate as single or multiple independent networks called Virtual Networks.



Network Virtualization in Virtual Data Center

Physical Network

- Physical components: Network adapters, switches, bridges, repeaters, routers and hubs.
- Grants connectivity among physical servers running a hypervisor, between physical servers and storage systems and between physical servers and clients.

VM Network

- Consists of virtual switches.
- Provides connectivity to the hypervisor kernel.
- Connects to the physical network.
- Resides inside the physical server.

- The result of external network virtualization is generally a virtual LAN (VLAN).
- A VLAN is an aggregation of hosts that communicate with each other as though they were located under the same broadcast domain. Internal network Virtualization is generally applied together with hardware and operating system-level virtualization, in which the guests obtain a virtual network interface to communicate with.
- There are several options for implementing internal network virtualization:

1. The guest can share the same network interface of the host and use Network Address Translation (NAT) to access the network; The virtual machine manager can emulate, and install on the host, an additional network device, together with the driver.
2. The guest can have a private network only with the guest.

Tools for Network Virtualization

1. **Physical switch OS** –

   It is where the OS must have the functionality of network virtualization.

2. **Hypervisor –**
   - ➔ It uses third-party software or built-in networking and the functionalities of network virtualization.
   - ➔ The basic functionality of the OS is to give the application or the executing process a simple set of instructions. System calls that are generated by the OS and executed through the library are comparable to the service primitives given at the interface between the application and the network through the SAP (Service Access Point).
   - ➔ The hypervisor is used to create a virtual switch and configure virtual networks on it. The third-party software is installed onto the hypervisor and it replaces the native networking functionality of the hypervisor. A hypervisor allows us to have various VMs all working optimally on a single piece of computer hardware.

Functions of Network Virtualization:

- It enables the functional grouping of nodes in a virtual network.
- It enables the virtual network to share network resources.
- It allows communication between nodes in a virtual network without routing of frames.
- It restricts management traffic.
- It enforces routing for communication between virtual networks.

Network Overlays –

- A framework is provided by an encapsulation protocol called VXLAN for overlaying virtualized layer 2 networks over layer 3 networks.
- The Generic Network Virtualization Encapsulation protocol (GENEVE) provides a new way of encapsulation designed to provide control-plane independence between the endpoints of the tunnel.

Network Virtualization Platform: VMware NSX –

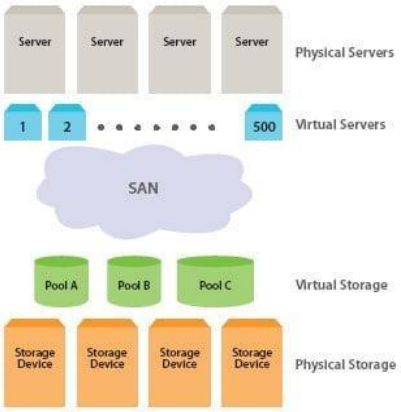
- VMware NSX Data Center transports the components of networking and security such as switching, firewalling and routing that are defined and consumed in software.
- It transports the operational model of a virtual machine (VM) for the network.

3.Storage Virtualization

Storage virtualization is a system administration practice that allows decoupling the physical organization of the hardware from its logical representation

- ➔ Using this technique, users do not have to be worried about the specific location of their data, which can be identified using a logical path.
- ➔ Storage virtualization allows us to harness a wide range of storage facilities and represent them under a single logical file system.
- ➔ There are different techniques for storage virtualization, one of the most popular being network based virtualization by means of storage

area networks (SANS).



- ➔ SANS uses a network accessible device through a large bandwidth connection to provide storage facilities.

Advantages of Storage Virtualization

1. Data is stored in the more convenient locations away from the specific host. In the case of a host failure, the data is not compromised necessarily.
2. The storage devices can perform advanced functions like replication, reduplication, and disaster recovery functionality.
3. By doing abstraction of the storage level, IT operations become more flexible in how storage is provided, partitioned, and protected.

Storage virtualization is a major component for storage servers, in the form of functional RAID levels and controllers. Operating systems and applications with devices can access the disks directly by themselves for writing. The controllers configure the local storage in RAID groups and present the storage to the operating system depending upon the configuration. However, the storage is abstracted and the controller is determining how to write the data or retrieve the requested data for the operating system.

★ **File servers:** The operating system writes the data to a remote location with no need to understand how to write to the physical media.

★ **SAN Accelerators:** Instead of sending multiple copies of the same data over the WAN environment, WAN accelerators will cache the data locally and present the re-requested blocks at LAN speed, while not impacting the WAN performance.

★ **SAN and NAS:** Storage is presented over the Ethernet network of the operating system. NAS presents the storage as file operations (like NFS). SAN technologies present the storage as block level storage (like Fibre Channel). SAN technologies receive the operating instructions only when the storage is a locally attached device.

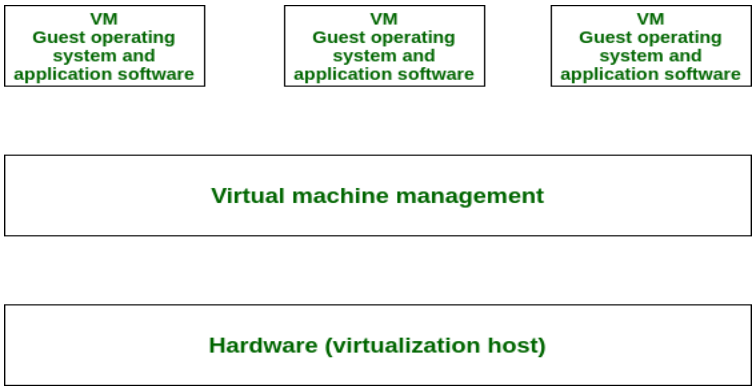
★ **Storage Tiering:** Utilizing the storage pool concept as a stepping stone, storage tiering analyze the most commonly used data and places it on the highest performing storage pool. The lowest used data is placed on the weakest performing storage pool.

4.System-level of Operating Virtualization:

Operating system-based Virtualization refers to an operating system feature in which the kernel enables the existence of various isolated user-space instances. The installation of virtualization software also refers to Operating system-based virtualization. It is installed over a pre-existing operating system and that operating system is called the host operating system.

In this virtualization, a user installs the virtualization software in the operating system of his system like any other program and utilizes this application to operate and generate various virtual machines. Here, the virtualization software allows direct access to any of the created virtual machines to the user. As the host OS can provide hardware devices with the mandatory support, operating system virtualization may affect compatibility issues of hardware even when the hardware driver is not allocated to the virtualization software.

Virtualization software is able to convert hardware IT resources that require unique software for operation into virtualized IT resources. As the host OS is a complete operating system in itself, many OS-based services are available as organizational management and administration tools can be utilized for the virtualization host management.



Some major operating system-based services are mentioned below:

1. Backup and Recovery.
2. Security Management.
3. Integration to Directory Services.

features of operating system-based virtualization are:

- **Resource isolation:** Operating system-based virtualization provides a high level of resource isolation, which allows each container to have its own set of resources, including CPU, memory, and I/O bandwidth.
- **Lightweight:** Containers are lightweight compared to traditional virtual machines as they share the same host operating system, resulting in faster startup and lower resource usage.
- **Portability:** Containers are highly portable, making it easy to move them from one environment to another without needing to modify the underlying application.
- **Scalability:** Containers can be easily scaled up or down based on the application requirements, allowing applications to be highly responsive to changes in demand.
- **Security:** Containers provide a high level of security by isolating the containerized application from the host operating system and other containers running on the same system.
- **Reduced Overhead:** Containers incur less overhead than traditional virtual machines, as they do not need to emulate a full hardware environment.
- **Easy Management:** Containers are easy to manage, as they can be started, stopped, and monitored using simple commands.

Operating system-based virtualization can raise demands and problems related to performance overhead, such as:

1. The host operating system employs CPU, memory, and other hardware IT resources.
2. Hardware-related calls from guest operating systems need to navigate numerous layers to and from the hardware, which shrinkage overall performance.
3. Licenses are frequently essential for host operating systems, in addition to individual licenses for each of their guest operating systems.

Advantages of Operating System-Based Virtualization:

- **Resource Efficiency:** Operating system-based virtualization allows for greater resource efficiency as containers do not need to emulate a complete hardware environment, which reduces resource overhead.
- **High Scalability:** Containers can be quickly and easily scaled up or down depending on the demand, which makes it easy to respond to changes in the workload.**Easy Management:** Containers are easy to manage as they can be managed through simple

commands, which makes it easy to deploy and maintain large numbers of containers.

Reduced Costs: Operating system-based virtualization can significantly reduce costs, as it requires fewer resources and infrastructure than traditional virtual machines.

Disadvantages of Operating System-Based Virtualization:

- **Security:** Operating system-based virtualization may pose security risks as containers share the same host operating system, which means that a security breach in one container could potentially affect all other containers running on the same system.
- **Limited Isolation:** Containers may not provide complete isolation between applications, which can lead to performance degradation or resource contention.

5.Application Virtualization

Application virtualization separates the OS, and the hardware that it runs on, from the app. This separation makes it possible to run applications from data center hardware, rather than having to have individual instances stored on and run from individual devices.

Application server virtualization abstracts a collection of application servers that provide the same services as a single virtual application server by using load balancing strategies and providing a high availability infrastructure for the services
hosted in the application Server

Virtualization of servers on a shared cluster can consolidate web services.

In cloud computing, virtualization also means the resources and fundamental infrastructure are virtualized.

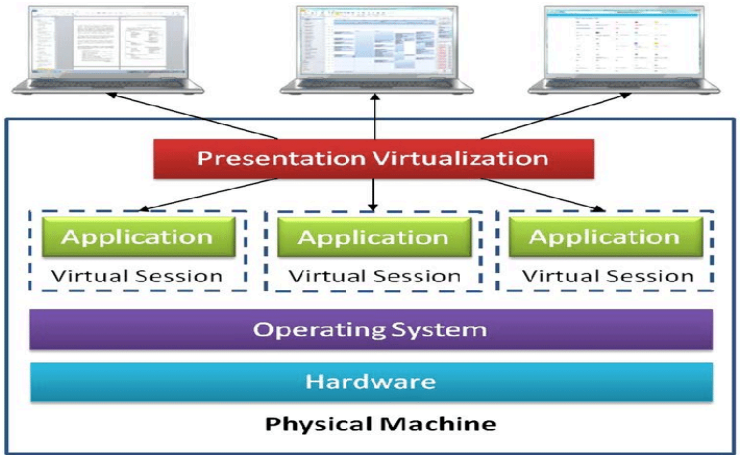
The user will not care about the computing resources that are used for providing the services

Cloud users do not need to know and have no way to discover physical resources that are involved while processing a service request. In addition, application developers do not care about some infrastructure issues such as scalability and fault tolerance. Application developers focus on service logic. In many cloud computing systems, virtualization

software is used to virtualize the hardware.System virtualization software is a special kind of software which simulates the execution of hardware and runs even unmodified operating systems.

Cloud computing systems use virtualization so well as the running environment for legacy software such as old operating systems and unusual applications.

- Application Virtualization Structure:



- Types of application virtualization:
- There are two types of app virtualization: remote and streaming.
- **Application Virtualization Example:**
- Virtualizing apps means that they run without any dependencies through another operating system or browser. An example would be virtualizing Microsoft PowerPoint to run on Ubuntu over an Opera browser. The implementation of both environments differs, as well.

6.Virtual clusters and Resource Management

Definition: Virtualization is a computer architecture technology where multiple virtual machines are multiplexed in the same hardware machine.

- **Types:**

➔ **Physical Cluster**:Collection of servers interconnected by a physical network.

➔ **Virtual Cluster**:virtual cluster built with virtual machine.

➔ **physical versus virtual cluster:**

➢ **Properties** :

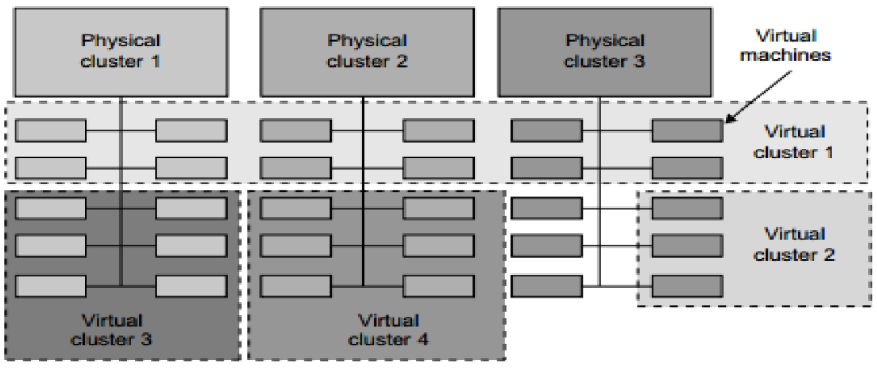
- ❖ multiple VMs running different Os developed same physical machine
- ❖ A VM runs with a guest OS.
- ❖ VM consolidate multiple functions on the same server.
- ❖ Physical machines are host systems,VMs are guest systems.

VM migration steps:

1. step 0 and 1:Start migration->preparations for migrations
2. step 2:Transfer Memory->sending VM memory to destination node
3. step 3:suspend VM copy the last portion of the data.
4. step 4 and 5:commit and activate the new data.

Migration of memory, files and network Resources:

1. **Memory Migration**->The internet suspend-Resumes
2. **File system migration**->file system is mapped, transporting the content of the virtual disk.
3. **Network migration-**>source and destination machine VM connected single switched LAN
4. **Live migration**->moving VM from one physical node to another.



A cloud platform with four virtual clusters over three physical clusters shaded differently.

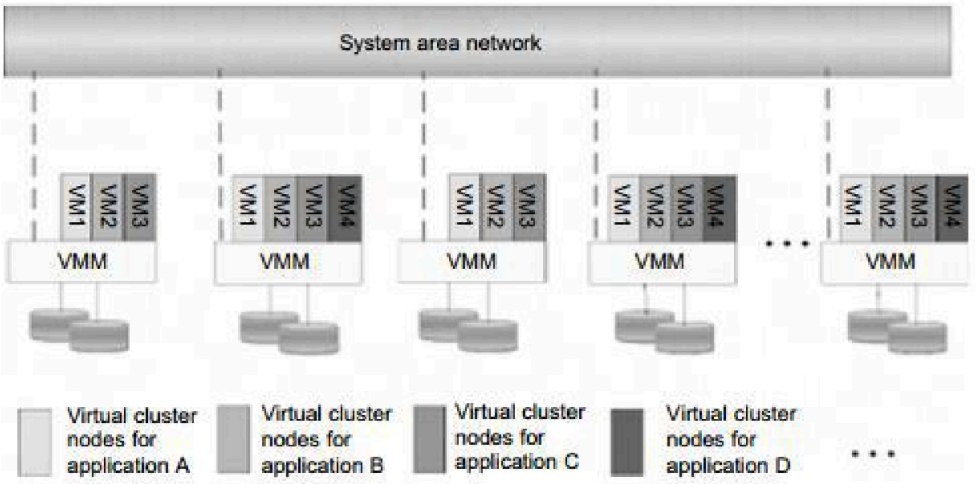
Virtual clustering provides a flexible solution for building clusters consisting of both physical and virtual machines.

It is widely used in various computing systems such as cloud platforms, high- performance computing systems, and computational grids.

Virtual clustering enables the rapid deployment of resources upon user demand or in response to node failures.There are four different ways to manage virtual clusters, including having the cluster manager reside on the guest or host systems, using independent cluster managers, or an integrated cluster manager designed to distinguish between virtualized and physical resources.

A VM can be in one of four states, including an inactive state, an active state, a paused state, and a suspended state.

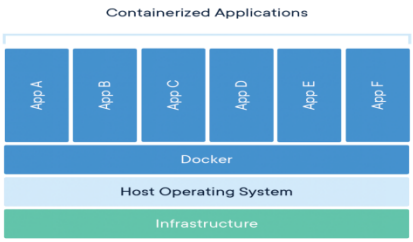
The live migration of VMs allows for VMs to be moved from one physical machine to another.



The concept of a virtual cluster based on application partitioning.

7.Introduction to Docker

Docker is a freemium platform as a service (PaaS) solution that aids in creating, operating, and maintaining containers for isolated software development and testing by creating a virtualized operating system (OS) environment. This article explains the working of Docker, its components, and top use cases.



How Does Docker Work?

Docker employs a client-server design. A Docker client communicates with the daemon (that

develops, operates, and distributes Docker containers). REST application programming interfaces (APIs) are used for network communication and UNIX sockets between the client and daemon.

The client may be connected to a daemon remotely, or the developer can run the daemon and client of Docker on the same computer. You may also interact with applications made up of a set of containers by using Docker Compose, which is another client for Docker.

The container's connection to the default network will be made possible through a network interface. After that, Docker will launch the container and run /bin/bash. You can type input while the result is logged to the terminal because the container is connected to the terminal and operating interactively. You can type exit to end the /bin/bash command, and the container will shut down.

Regarding container-based implementation, Docker has taken over as the de facto standard. Developers utilize Docker, an open-source platform created by DotCloud and implemented in the GO programming language, to create, distribute, and operate programs. With Docker, it is simple to segregate apps from the surrounding IT infrastructure for a project's quick delivery. It allows programmers to containerize and run the application in loosely remote environments.

Docker provides a level of separation that allows developers to run numerous containers concurrently on a single host. This allows for more flexibility. Originally based on Linux Containers, Docker currently uses the container runtime libcontainer function (a part of runc). The Linux kernel's built-in virtualization and process isolation features enable containers. Similar

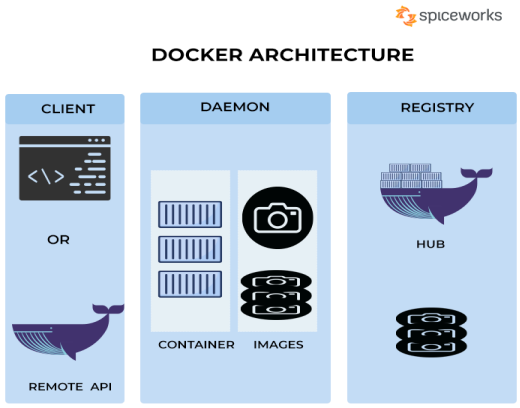
to how a hypervisor allows multiple virtual machines (VMs) to share the CPU, memory, and other resources of a single hardware server.

These capabilities, such as control groups (cGroups) for allocating resources among processes and namespaces for restricting a process's access or visibility into other resources or sections of the system, allow the different parts of an application to share the resources of a single instance of the host operating system.

Although Docker is not required to create containers, the platform's set of tools, dependencies, and libraries needed to run an application make container creation simpler, safer, and more manageable.

One can build modern platforms and apps with the help of Docker as a fundamental building piece. Undoubtedly, Docker makes constructing and running distributed microservices architecture, publishing code into the continuous integration and delivery (CI/CD) pipeline, and developing scalable data processing systems simple. A constant and quick distribution of an application is made possible by Docker. A developer's laptop, a cloud provider's virtual machine, a data center, or a mixed environment can execute Docker.

8. Key Components of Docker



Rapid application development, testing, and deployment are made possible by the solution known as Docker. Docker packages software into standardized units called containers

containing all the necessary code, libraries, system tools, and runtime. With Docker, you can quickly scale and deploy apps into any environment, confident that your code will work.

Docker accelerates, simplifies, and secures containerization. When you wanted to operate a web application in the past, you would purchase a server, install Linux, build up a LAMP stack, and then launch the application. To prevent the application from crashing due to excessive traffic, you should practice proper load balancing by creating a second server if your app becomes popular.

However, as technology has advanced, the internet is based on clusters of redundant, interconnected servers, referred to as “the cloud.” Docker uses this system to free application development for hardware infrastructure lock-in.

1. The Docker Engine

The Docker Engine, also known as simply DE, is the essential central component of the Docker system. It must be downloaded and installed on the host computer. Next, a lightweight runtime system and the client-server technology that lies on top of the DE are used in creating and managing containers. There are three parts to the Docker Engine:

- **Server:** The Docker daemon (docker d) is in charge of creating and managing containers.
- **Rest API:** The Rest API enables the communication between applications and Docker and gives Dockerd instructions.
- **Command Line Interface (CLI):** Docker commands are executed using the CLI.

2. Docker images

The building blocks for containers are called Docker images. Like snapshots for virtual machines, Docker images are immutable, read-only files that include the source code. It also contains libraries, tools, dependencies, and additional files to run an application. Images are essential for reducing disk utilization, enhancing reusability, and accelerating Docker build. The importance of maintaining compact images cannot be overstated because you want your containers to be quick and light. A Dockerfile, which contains detailed instructions for constructing a specific Docker image, is used to create each image. You can develop images and customized containers more quickly and easily if you’ve mastered producing Docker images from Dockerfiles.

3. Dockerfile

A Dockerfile is a script with instructions on how to make a Docker image. In these instructions, you can find information about the operating system, languages, environment variables, file locations, network ports, and other details needed to run the image. The commands in the file are placed in groups, and those groups are automatically executed.

An image contains several layers. A new read-write layer is introduced once a Docker image has been launched to construct a container. The container layer is another name for this. With the additional layer, you can modify the base image and then commit your modifications to produce a fresh Docker image for usage in the future.

4. The Docker Hub

Docker Hub is the most extensive cloud-based archive of Docker's container images. More than 100,000 images produced by open-source initiatives, software companies, and the Docker community are made accessible for use. The platform enables you to swiftly integrate your applications into a development pipeline, engage with team members, and ship your applications anywhere.

Like GitHub, developers can choose whether to maintain their private or public container images on Docker Hub by pushing and pulling them. Users of Docker Hub can freely distribute their images. To use the Docker filesystem as a starting point for any containerization project, they can also download prepared base images from it.

5. Docker volumes

When preserving data generated by a container that is already operating, using Docker volumes is a superior choice to adding extra layers to an image. With this feature's assistance, users can store data, transfer it across containers, and mount it to new ones.

Because they are kept on the host, Docker volumes are immune to any changes that may occur throughout the container's life cycle. When starting a container, you may generate and mount a Docker volume in several ways. These methods are all available.

6. Docker Compose

The process of creating and testing multi-container applications is made more accessible with the help of Docker Compose. It chooses the services to include in the application and generates a YAML Ain't Markup Language (YAML) file. One can use a single command to deploy and run containers. Docker Compose is a helpful tool to streamline the process of running and managing numerous containers simultaneously.

It connects the containers that must cooperate and manages them with a single coordinated command. You can also define nodes, storage, and service dependencies using Docker Compose. Other technologies like Kubernetes and Docker Swarm allow more complex variations of similar tasks, known as container orchestration.

7. Docker Desktop

Docker Desktop was known initially as Docker for Windows and Docker for Mac. When using Docker Desktop, it takes a few minutes to start creating and operating containers on either Windows or Mac. The entire Docker development environment may be installed and set up quickly. The environment may include components like Kubernetes and Credential Helper, and Docker features such as the DE, Docker Compose, Docker CLI client, and Docker Content Trust. On any cloud platform, the tool is used to create and share containerized apps and microservices in various languages and frameworks.

8. Docker containers

The active, operational instances of Docker images are known as containers. Docker images are

read-only files, whereas containers contain executable, transient content. Users can interact with them, and administrators can use Docker commands to change their parameters.

9. Docker Container

A Docker container

- is open source
- is an easy way to build a development environment

- can hold applications “inside the container”
- is portable across Linux, Mac, and Windows machines
- is much smaller than a virtual machine
- sets up a user-defined partition between the host machine and “container land”
- allows “root” inside the container, but does not alter permissions on the host machine
- requires root access to install Docker
- runs on a single node

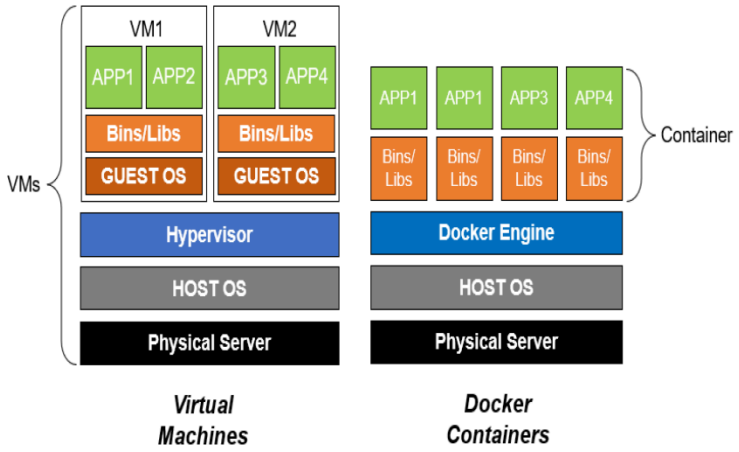
A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Why use containers?

- Software systems require substantial set-up to get all the necessary code, including external libraries, compiled on a specific platform à Containers help solve this problem!
- Efficient, lightweight, secure, and self-contained (including operating system, libraries, code, and executables) systems
- Everything required is packaged into isolated components, ready for development, shipment, and deployment directly to users
- Software should always run the same, regardless of where it is deployed
- Eliminates possible frustrations with up-front system setup

Virtual machine vs. containers

Containers vs. virtual machines: VMs bundle a full operating system, whereas containers only contain necessary libraries and dependencies



Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime.

Possible States of a Docker Container:

- Created. Docker assigns the created state to the containers that were never started ever since they were created. ...
- Running. ...
- Restarting. ...
- .Excited. ...
- Paused. ...
- Dead.

Principle of Docker container:

Docker uses a technology called namespaces to provide the isolated workspace called the container. When you run a container, Docker creates a set of namespaces for that container. These namespaces provide a layer of isolation.

USE of Docker Container:

Docker allows you to ship, test, and deploy your applications in any environment without worrying about incompatible issues regardless of the machine's configuration settings. Thus, no less work when developing applications coupled with a standardized way to deploy them.

What is the difference between Docker and container?

The key difference between a Docker image Vs a container is that a Docker image is a read-only immutable template that defines how a container will be realized. A Docker container is a runtime instance of a Docker image that gets created when the $ docker run command is implemented.

10.Docker image and Repositories:

Docker Image

In order to run these applications, we first need to create an image of the current application state. An image can be sometimes referred to as a snapshot of our project. Images are read-only in nature and consist of a file that contains the source code, libraries, dependencies, tools, and other files needed for an application to run.

A Docker image is a read-only template that contains a set of instructions for creating a container which can run on the Docker platform

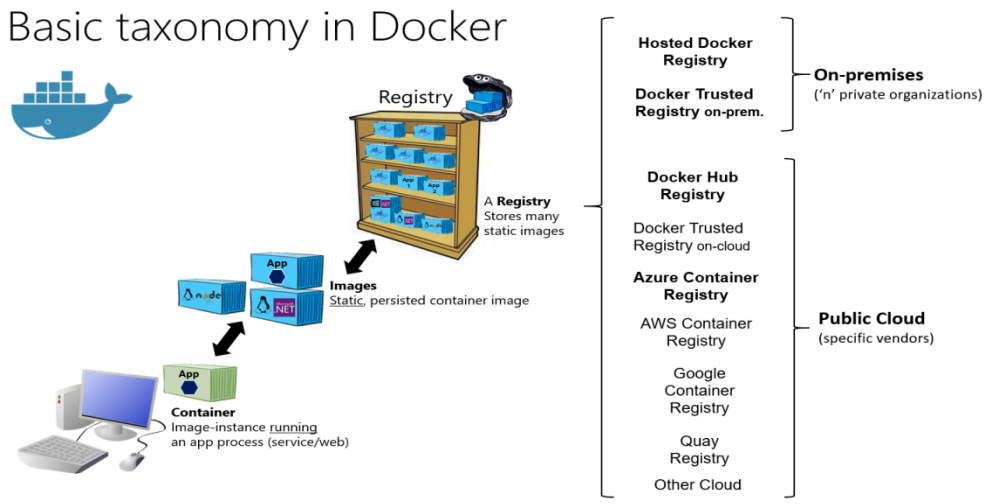
Docker Repository

A Docker repository is a collection of different Docker images with the same name, that have different tags. Tags basically are identifiers of the image within a repository.

The Docker Hub is a cloud-based repository service where users can push their Docker Container Images and access them from anywhere via the internet.It offers the option to push images as private or public and is primarily used by DevOps teams.

The Docker Hub is an open-source tool that is available for all operating systems. It functions as a storage system for Docker images and allows users to pull the required images when needed.However, it is necessary to have a basic knowledge of Docker to push or pull images from the Docker Hub. If a developer team wants to share a project along with its dependencies for testing, they can push the code to Docker Hub. To do this, the developer must create images and push them to Docker Hub. The testing team can then pull the same image from Docker Hub without needing any files, software, or plugins,

as the developer has already shared the image with all dependencies.



Features of Docker Hub:

- Docker Hub simplifies the storage, management, and sharing of images with others. It provides security checks for images and generates comprehensive reports on any security issues.
- Additionally, Docker Hub can automate processes like Continuous Deployment and Continuous Testing by triggering webhooks when a new image is uploaded.
- Through Docker Hub, users can manage permissions for teams, users, and organizations.
- Moreover, Docker Hub can be integrated with tools like GitHub and Jenkins, streamlining workflows.

Advantages of Docker Hub

- Docker Container Images have a lightweight design, which enables us to push images in a matter of minutes using a simple command.
- This method is secure and offers the option of pushing private or public images.
- Docker Hub is a critical component of industry workflows as its popularity grows, serving as a bridge between developer and testing teams.
- Making code, software or any type of file available to the public can be done easily by publishing the images on the Docker:Public

Docker Registry

A Docker registry is a service that stores and manages Docker images. Docker registry could be hosted by a third party, as a public or private registry.

Some examples of Docker registries are

- Docker Hub
- GitLab
- AWS Container Registry
- Google Container Registry
- Docker – Private Registries.

11. Containers vs. Virtual Machines:

- Virtual machines provide an abstracted version of the entire hardware of a physical machine, including the CPU, memory, and storage. Containers are portable instances of software with its dependencies that run on a physical or virtual machine.
- Virtual machines (VM) provide stronger isolation compared to containers. Because each VM runs on its own dedicated OS, it creates a complete virtualized environment. This isolation ensures that applications and processes within one VM are isolated from others, providing enhanced security.
- Containers and virtual machines are very similar resource virtualization technologies. Virtualization is the process in which a system singular resource like RAM, CPU, Disk, or Networking can be 'virtualized' and represented as multiple resources. The key differentiator between containers and virtual machines is that virtual machines virtualize an entire machine down to the hardware layers and containers only virtualize software layers above the operating system level.
-

Container providers

- **Docker**

  Docker is the most popular and widely used container runtime. Docker Hub is a giant public repository of popular containerized software applications. Containers on Docker Hub can be instantly downloaded and deployed to a local Docker runtime.

- **RKT**

  Pronounced "Rocket", RKT is a security-first focused container system. RKT containers do not allow insecure container functionality unless the user explicitly enables insecure features. RKT containers aim to address the underlying cross contamination exploitative security issues that other container runtime systems suffer from.
- **Linux Containers (LXC)**

  The Linux Containers project is an open-source Linux container runtime system. LXC is used to isolate operating, system-level processes from each other. Docker actually uses LXC behind the scenes. Linux Containers aim to offer a vendor neutral open-source container runtime.
- **CRI-O**

  CRI-O is an implementation of the Kubernetes Container Runtime Interface (CRI) that allows the use of Open Container Initiative (OCI) compatible runtimes. It is a lightweight alternative to using Docker as the runtime for Kubernetes.

  Virtual machine providers

- **Virtualbox**

  Virtualbox is a free and open source x86 architecture emulation system owned by Oracle. Virtualbox is one of the most popular and established virtual machine platforms with an ecosystem of supplementary tools to help develop and distribute virtual machine images.
- **VMware**

  VMware is a publicly traded company that has built its business on one of the first x86 hardware virtualization technologies. VMware comes included with a hypervisor which is a utility that will deploy and manage multiple virtual machines. VMware has a robust UI for managing virtual machines. VMware is a great enterprise virtual machine option offering support.
- **QEMU**

  QEUM is the most robust hardware emulation virtual machine option. It has support for any generic hardware architecture. QEMU is a command line only utility and does not offer a graphical user interface for configuration or execution. This trade-off makes QEMU one of the fastest virtual machine options.

Difference between Virtual machine and containers