
Amrita Vishwa Vidyapeetham
Amritapuri Campus
AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

Welcome All

CLOUD COMPUTING

Week - 6

Dr. Harsha Vasudev
Assistant Professor, CSE
Amrita Vishwa Vidyapeetham

Virtualization

- Virtualization technology is one of the fundamental components of cloud computing, especially in regard to infrastructure-based services.

- Allows the creation of a secure, customizable, and isolated execution environment for running applications, even if they are untrusted, without affecting other users' applications.

- The basis of this technology is the ability of a computer program—or a combination of software and hardware—to emulate an executing environment separate from the one that hosts such programs.

- It provides a great opportunity to build elastically scalable systems that can provision additional capability with minimum costs.

- Therefore, virtualization is widely used to deliver customizable computing environments on demand.

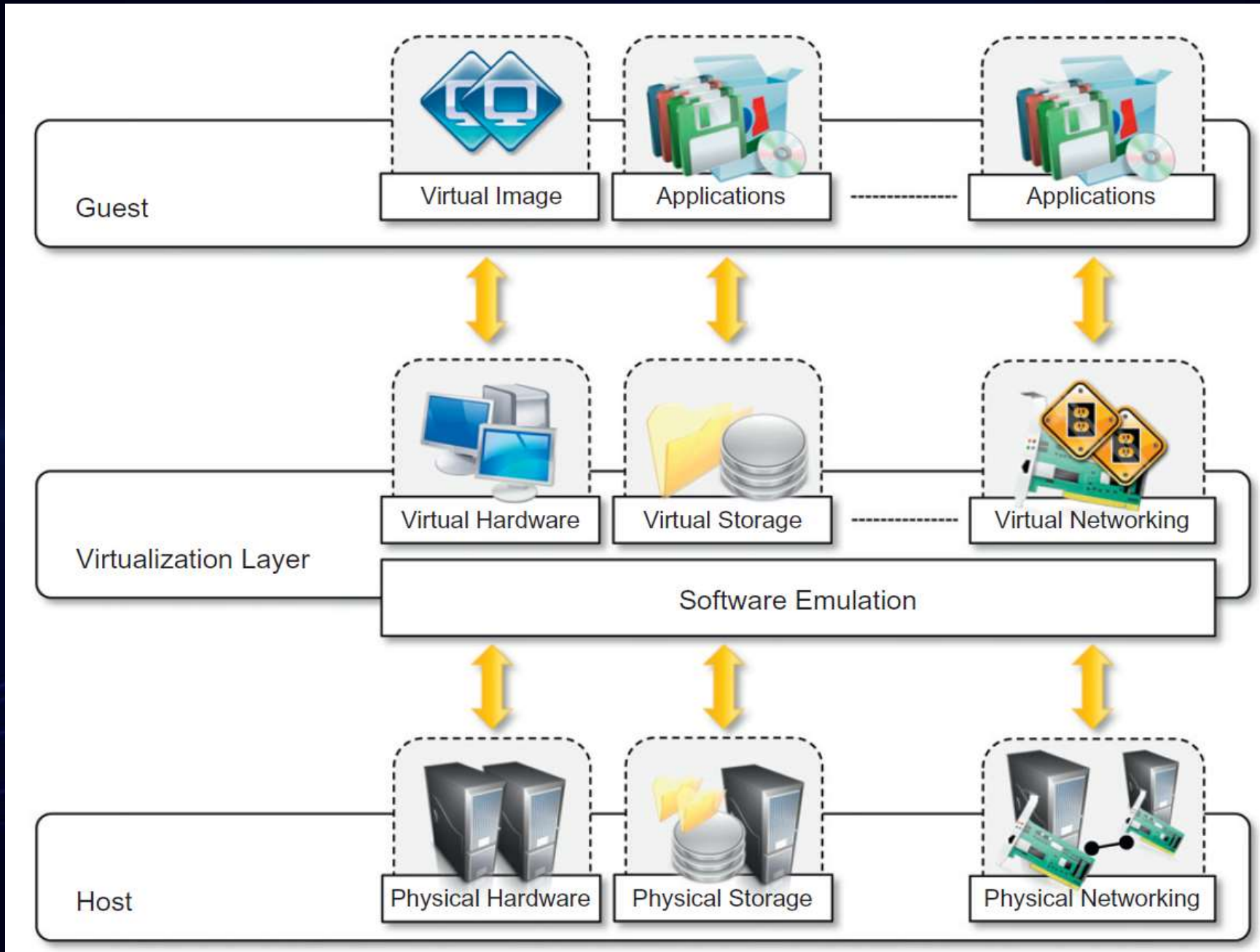
Characteristics of Virtualized Environments

- The main common characteristic of all these different implementations is the fact that the virtual environment is created by means of a software program.

- The ability to use software to emulate such a wide variety of environments creates a lot of opportunities, previously less attractive because of excessive overhead introduced by the virtualization layer.

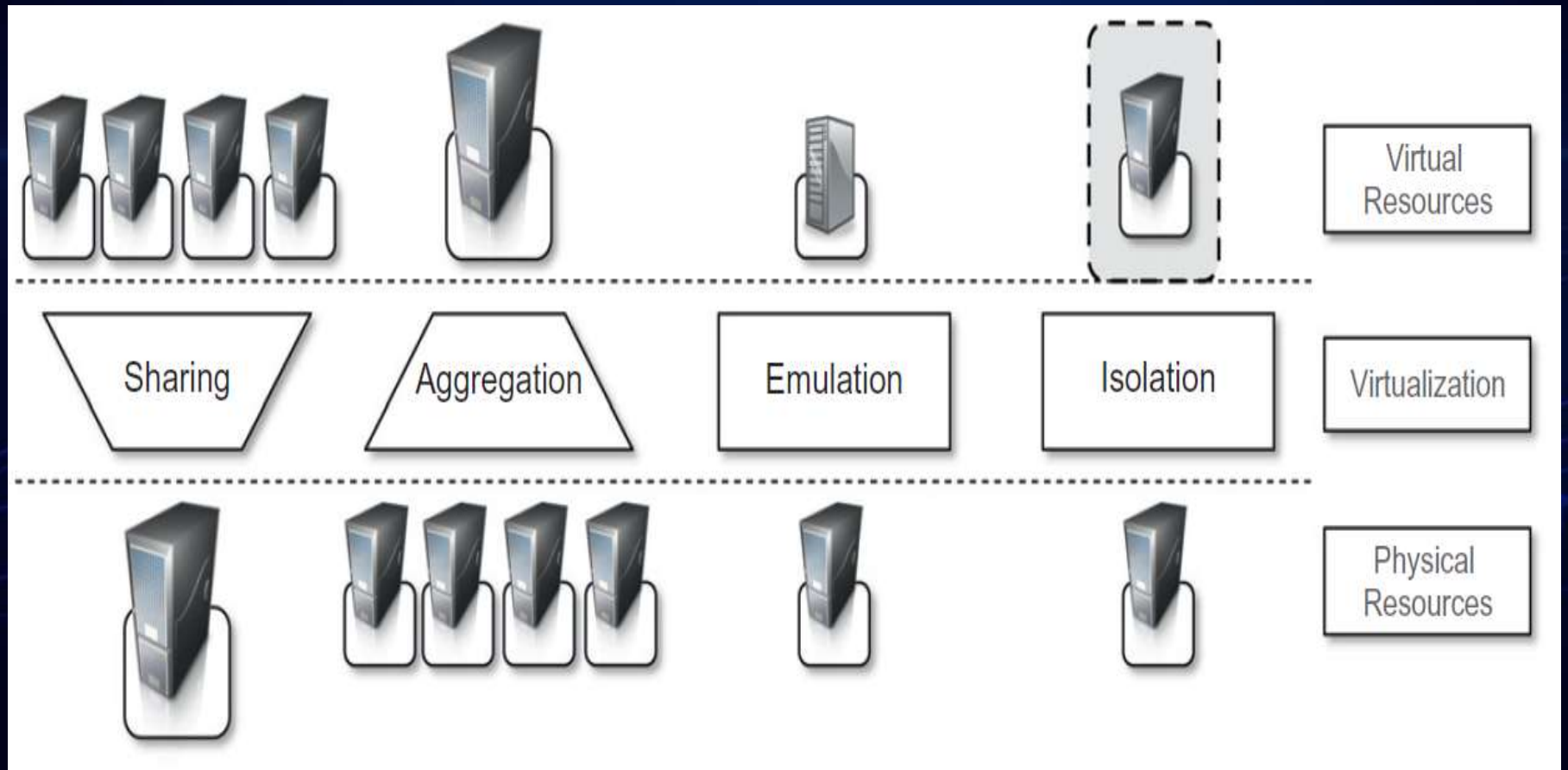
- The technologies of today allow profitable use of virtualization and make it possible to fully exploit the advantages that come with it.

The Virtualization Reference Model



- **Increased Security :** Ability to control the execution of a guest in a completely transparent manner

- **Managed Execution :**



- Sharing : Virtualization allows the creation of a separate computing environments within the same host.
- Aggregation : A group of separate hosts can be tied together and represented to guests as a single virtual host.
- Emulation : Guest programs are executed within an environment that is controlled by the virtualization layer, which ultimately is a program. This allows for controlling and tuning the environment that is exposed to guests.
- Isolation : It allows providing guests—whether they are operating systems, applications, or other entities—with a completely separate environment, in which they are executed

Taxonomy of Virtualization Techniques

- Virtualization is mainly used to emulate execution environments, storage, and networks.
- **Execution Virtualization : Constitutes the oldest, most popular, and most developed area.**
- **Two major categories by considering the type of host they require.**
- **Process-level techniques are implemented on top of an existing operating system, which has full control of the hardware.**
- **System-level techniques are implemented directly on hardware and do not require—or require a minimum of support from—an existing operating system**

Hypervisors

- A fundamental element of hardware virtualization is the hypervisor, or virtual machine manager (VMM).
- It recreates a hardware environment in which guest operating systems are installed.
- There are two major types of hypervisor: Type I and Type II .
- Type I hypervisors : Run directly on top of the hardware.
- This type of hypervisor is also called a native virtual machine since it runs natively on hardware.
- Type II hypervisors : Require the support of an operating system to provide virtualization services.

I. Type I Hypervisors :

- Bare Metal Hypervisor.
- VMware ESX1, Microsoft –Hyper V.

II. Type I Hypervisors :

- Not bare metal
- Oracle Virtual box, VMware workstation

DIFFERENCES ?

Let’s Explore More in the Next Class


Amrita Vishwa Vidyapeetham
Amritapuri Campus
AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

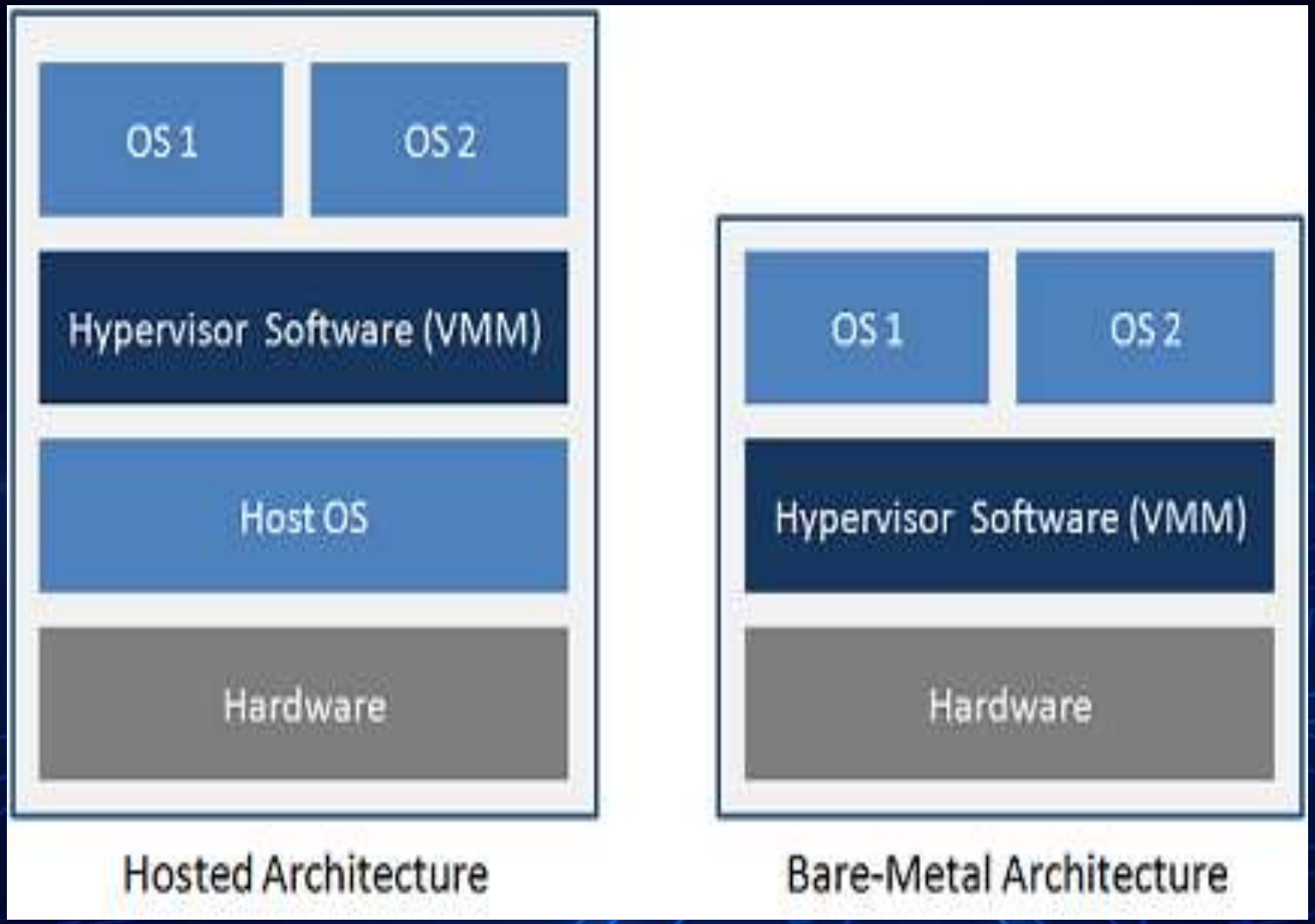
Welcome All

CLOUD COMPUTING

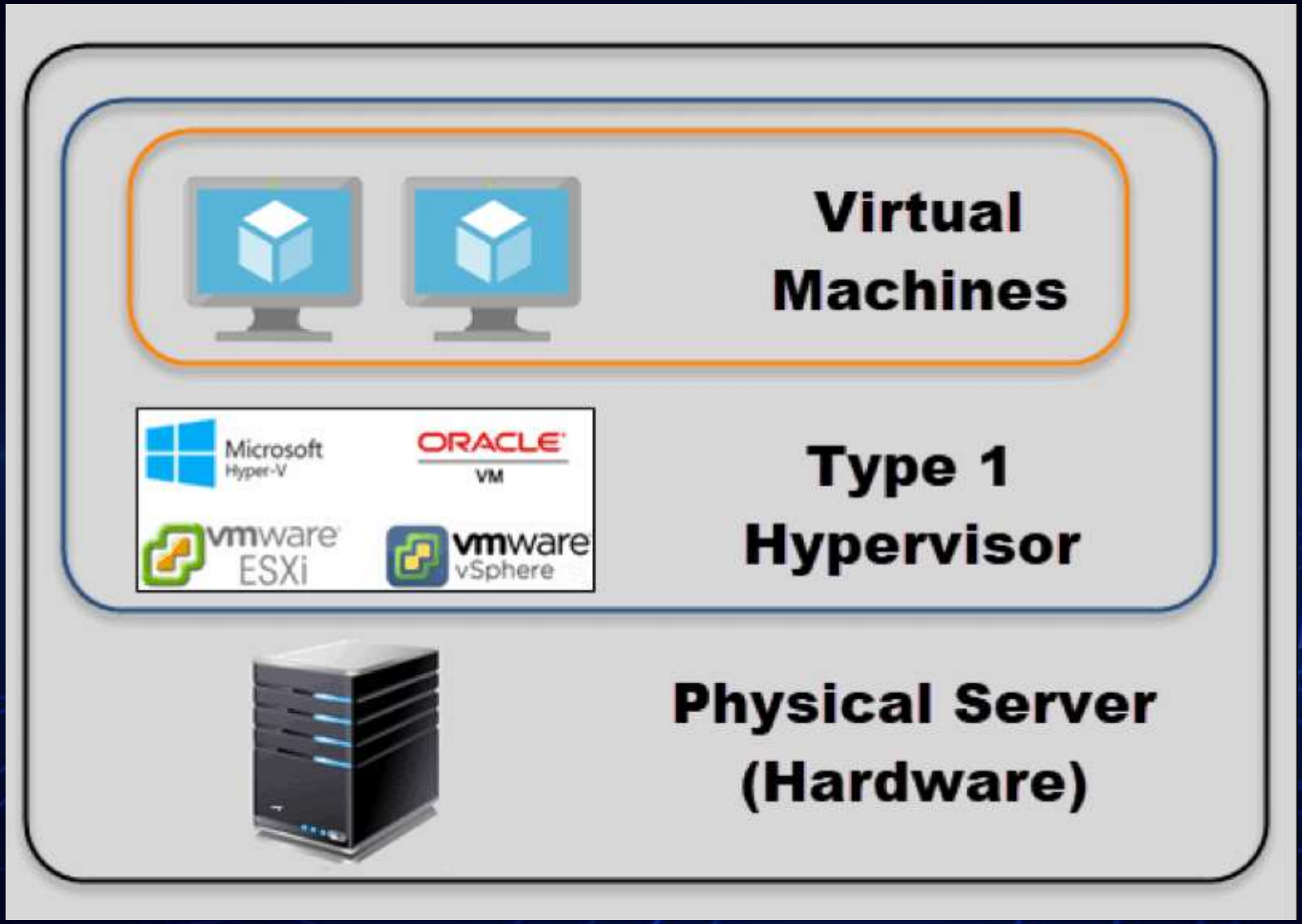
Week - 8

Dr. Harsha Vasudev
Assistant Professor, CSE
Amrita Vishwa Vidyapeetham

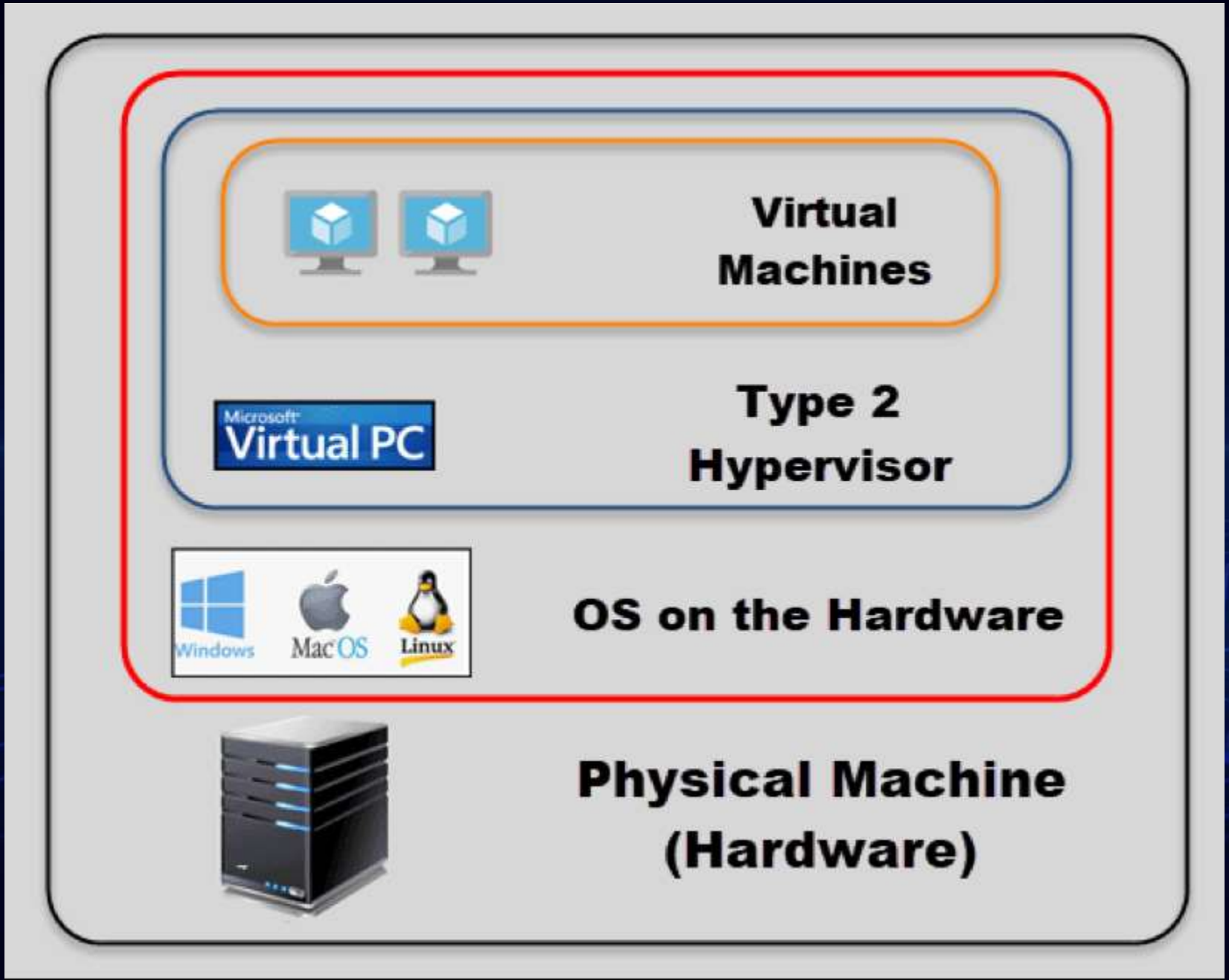
Hypervisors



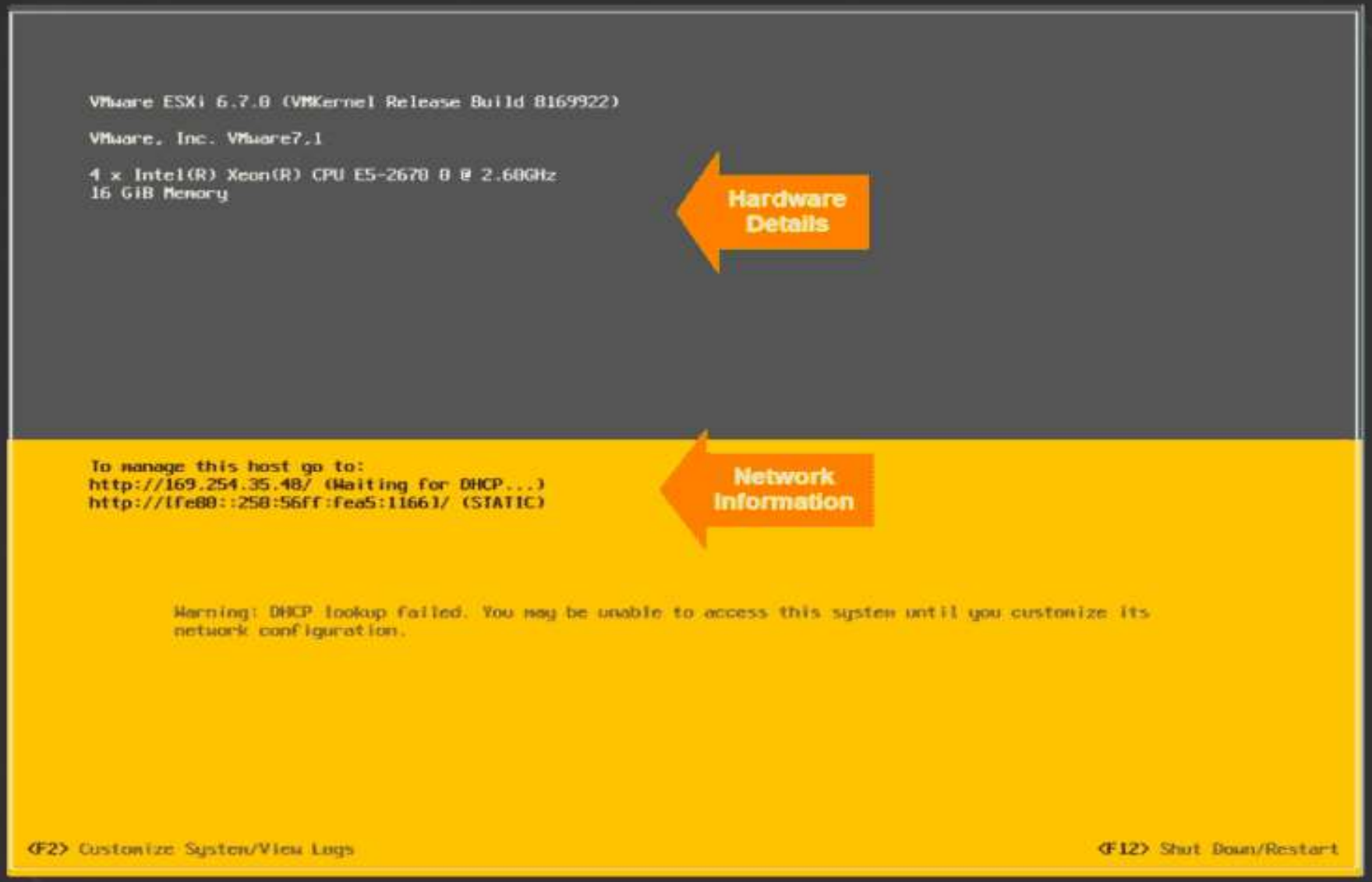
Cont'd


Virtual Machines
Microsoft Hyper-V
ORACLE VM
vmware ESXi
vmware vSphere
Type 1 Hypervisor
Physical Server (Hardware)

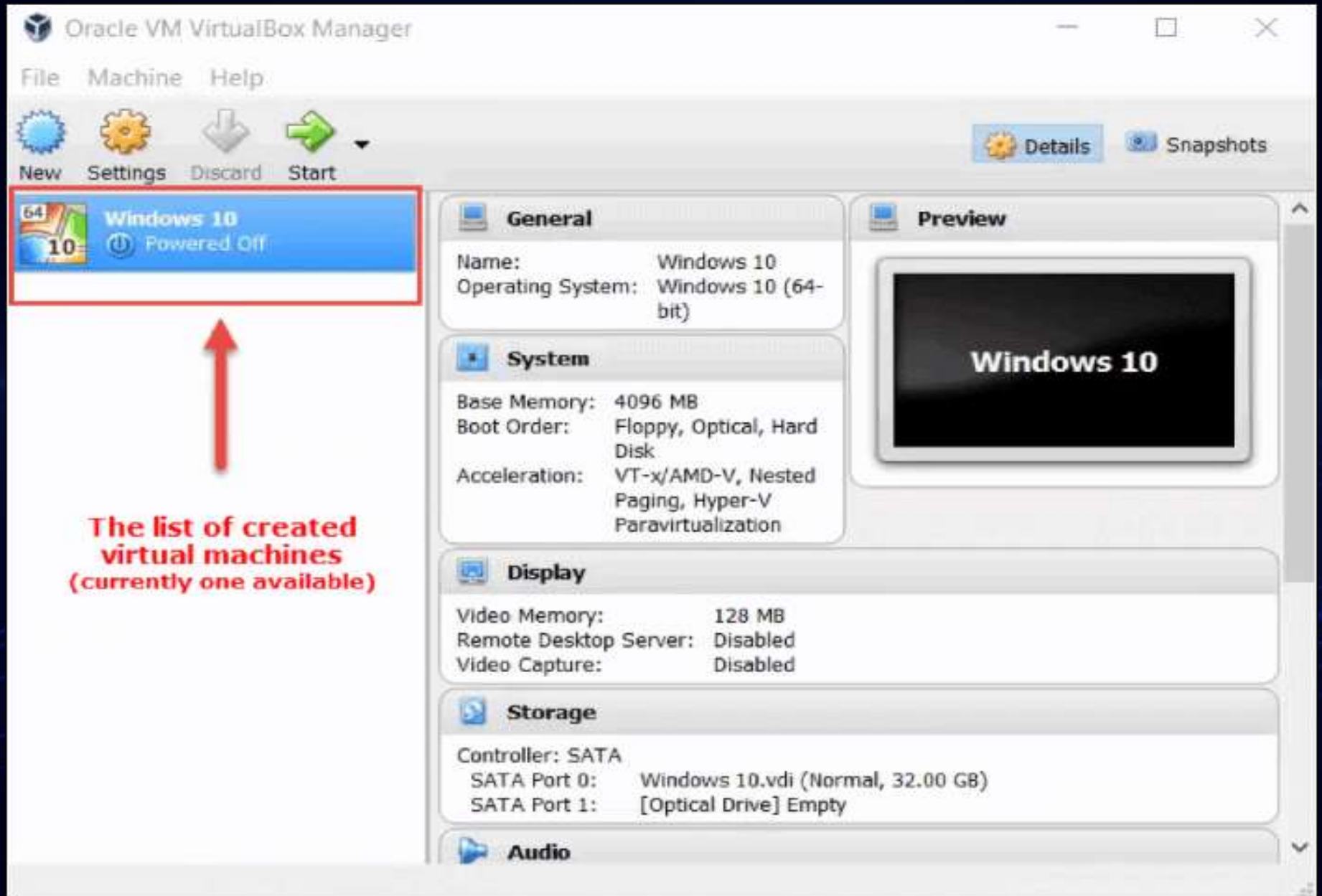
Cont'd


Virtual Machines
Microsoft Virtual PC
Type 2 Hypervisor
Windows
Mac OS
Linux
OS on the Hardware
Physical Machine (Hardware)

Example of a VMware Type 1 Hypervisor's screen after the server boots up.



Example of a Type 2 hypervisor interface (VirtualBox by Oracle):



Challenges

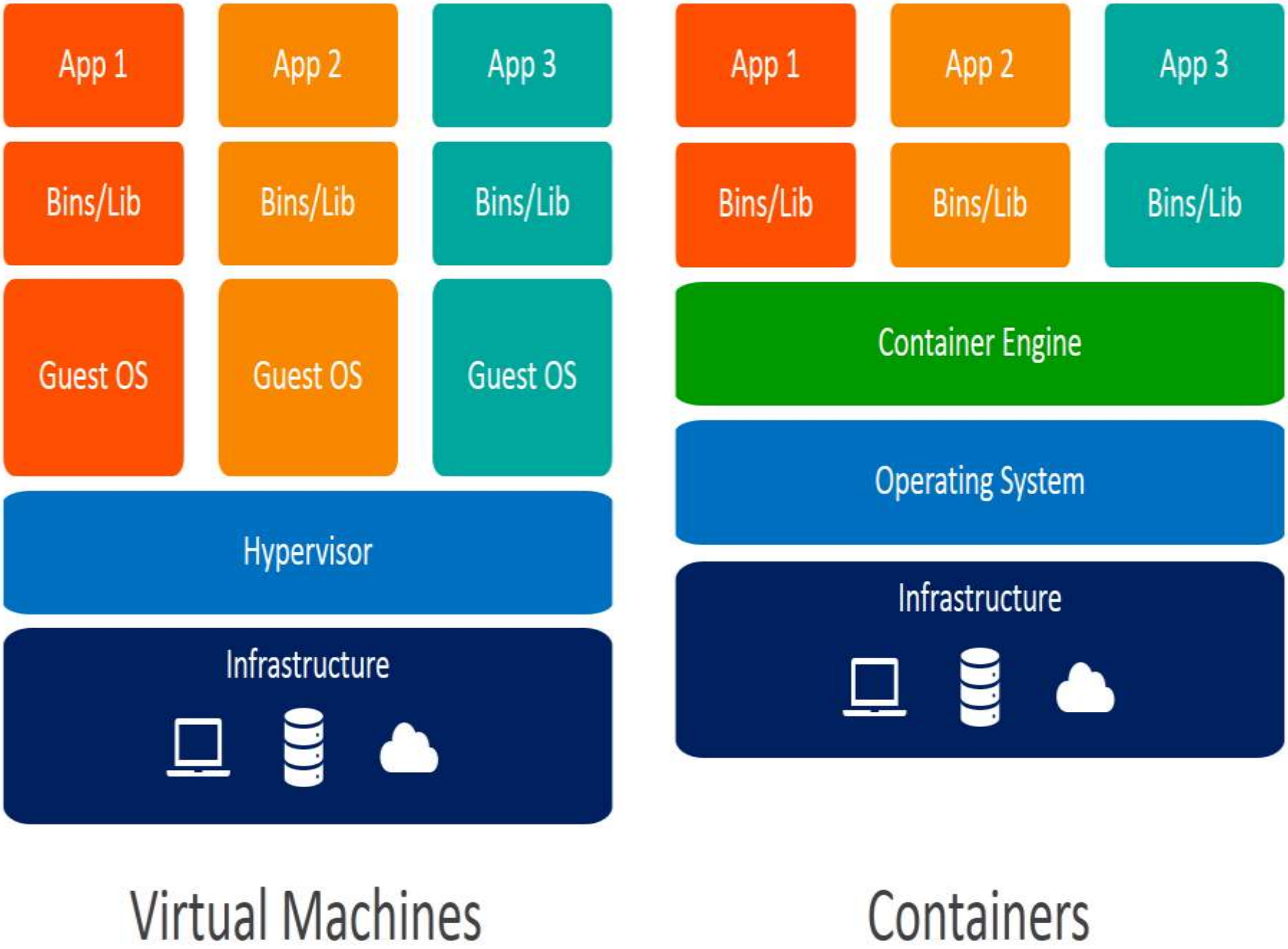
Challenges

- Managing OS
- Instant deployment
- Maintaining multi host applications
- etc.

Containers / Containerization

- ✓ Simply a way to package s/w within a standard environment.
- ✓ Containerization includes all dependencies (frameworks, libraries, etc.) required to run an application in an efficient and bug-free manner.

Virtualization Vs Containerization


App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure
Virtual Machines
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Container Engine
Operating System
Infrastructure
Containers

Virtual Machine	*Container*
• Heavyweight	• Lightweight
• Limited performance	• Native performance
• Each VM runs in its own OS	• All containers share the host OS
• Hardware-level virtualization	• OS virtualization
• Startup time in minutes	• Startup time in milliseconds
• Allocates required memory	• Requires less memory space
• Fully isolated and hence more secure	• Process-level isolation, possibly less secure

Dockers

- Docker is an open source platform that helps a user to package an application and its dependencies into a Docker container for the development and deployment of software .
- Provides suitable frameworks for different applications.

Benefits of Docker Containers

1. Lack of external dependency
2. Host and run applications
3. Isolation

Docker Architecture

❖ The Docker Engine

Docker Engine allows to develop, assemble, ship, and run applications using the following components:

1. Docker Daemon:

- A persistent background process that manages Docker images, containers, networks, and storage volumes.
- The Docker daemon constantly listens for Docker API requests and processes them.

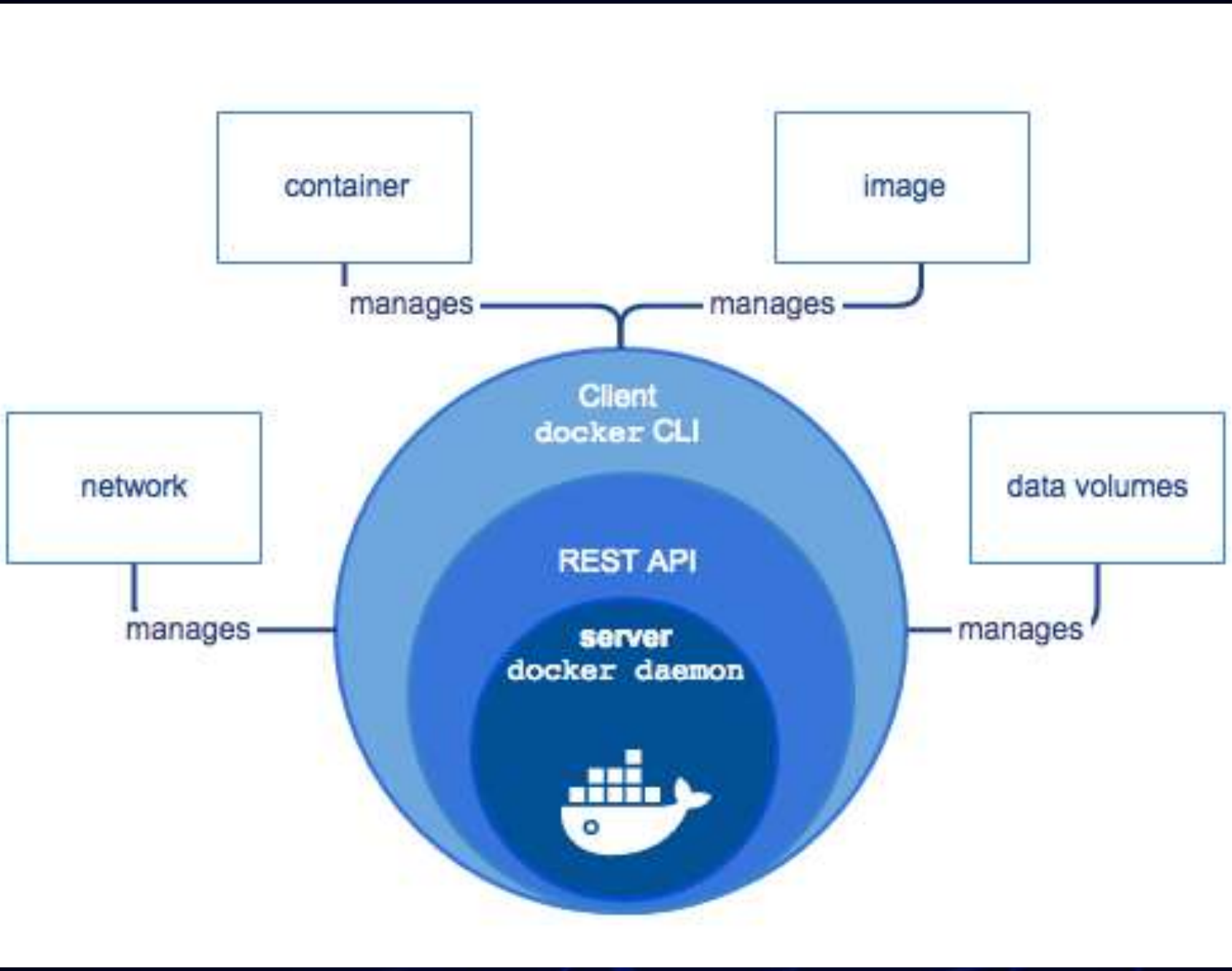
Cont'd

2. Docker Engine REST API:

An API used by applications to interact with the Docker daemon; it can be accessed by an HTTP client.

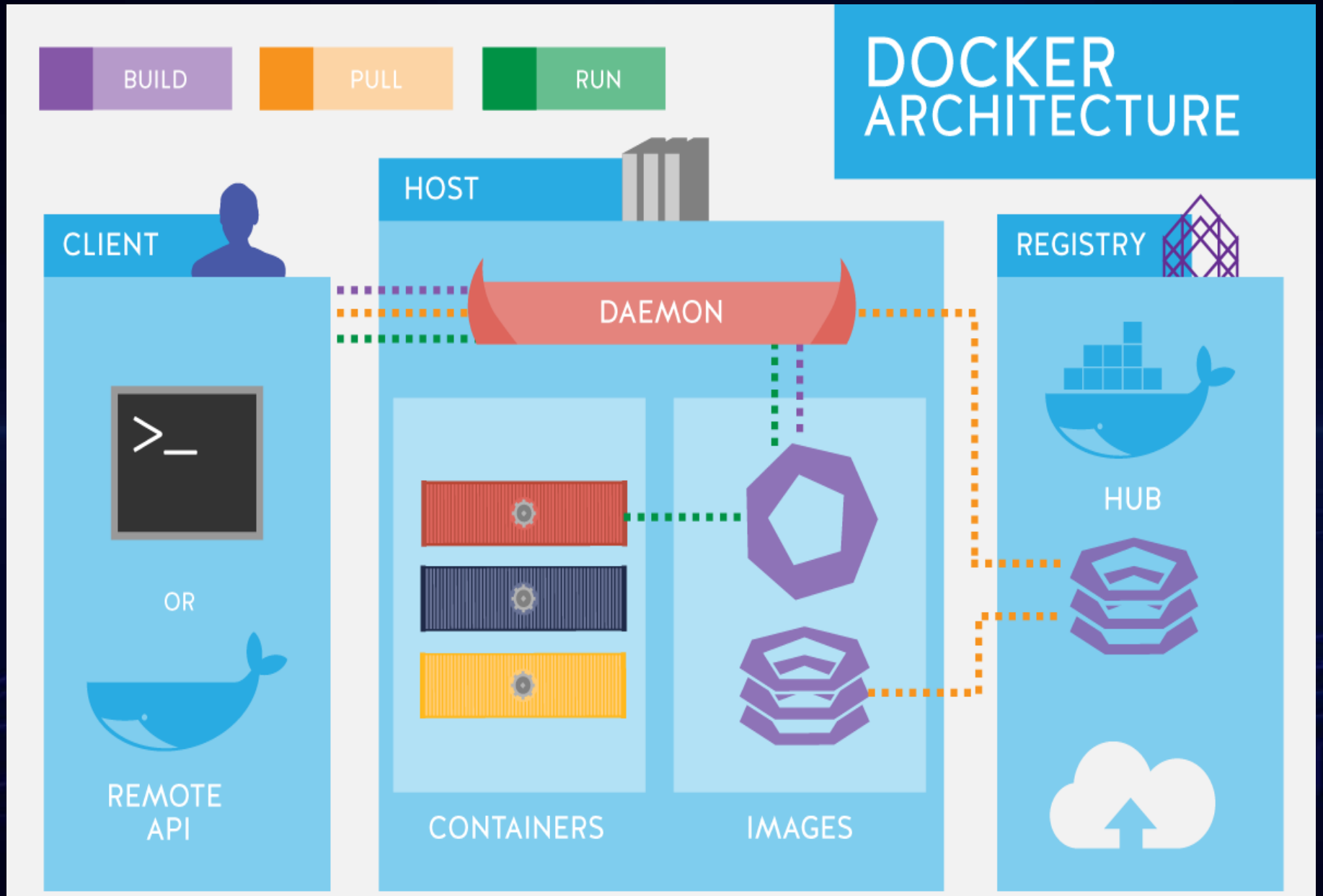
3. Docker CLI:

- A command line interface client for interacting with the Docker daemon.
- It greatly simplifies how you manage container instances and is one of the key reasons why developers love using Docker.


container
image
manages
manages
Client
docker CLI
network
data volumes
REST API
manages
server
docker daemon
manages

Docker Architecture

- The Docker architecture uses a client-server
- Comprises of
  - the Docker Client,
  - Docker Host,
  - Network and Storage components,
  - and the Docker Registry/Hub.


BUILD
PULL
RUN
DOCKER ARCHITECTURE
HOST
CLIENT
REGISTRY
DAEMON
>_
OR
HUB
REMOTE API
CONTAINERS
IMAGES
NORDICAPIS.COM

1. Docker Client :

• The Docker client enables users to interact with Docker.

• The Docker client can reside on the same host as the daemon or connect to a daemon on a remote host.

•A docker client can communicate with more than one daemon.

•The Docker client provides a command line interface (CLI) that allows you to issue build, run, and stop application commands to a Docker daemon.

•The main purpose of the Docker Client is to provide a means to direct the pull of images from a registry and to have it run on a Docker host.

2. Docker Host

• The Docker host provides a complete environment to execute and run applications.

• It comprises of the Docker daemon, Images, Containers, Networks, and Storage.

• The daemon is responsible for all container-related actions and receives commands via the CLI or the REST API.

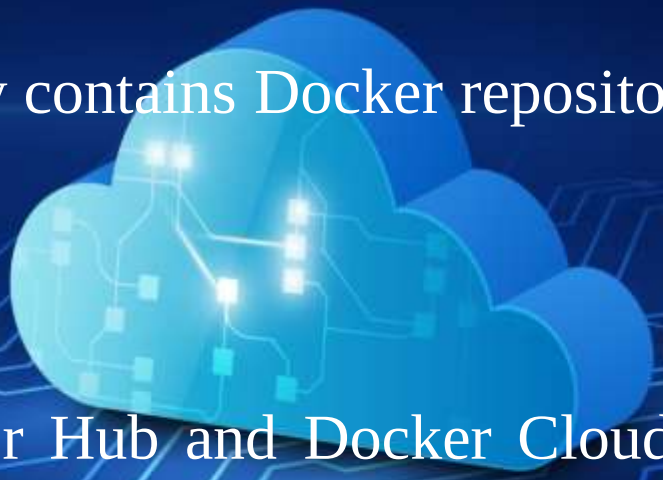
• It can also communicate with other daemons to manage its services.

• The Docker daemon pulls and builds container images as requested by the client.

3. **Docker Registries**

• Docker registries are services that provide locations from where you can store and download images.

• In other words, a Docker registry contains Docker repositories that host one or more Docker Images.



• Public Registries include Docker Hub and Docker Cloud and private Registries can also be used.

Lab Assignment -2

Create a Docker Container on Google Cloud Platform.

Let’s Explore More in the Next Class


Amrita Vishwa Vidyapeetham
Amritapuri Campus
AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY
श्रद्धावान् लभते ज्ञानम्

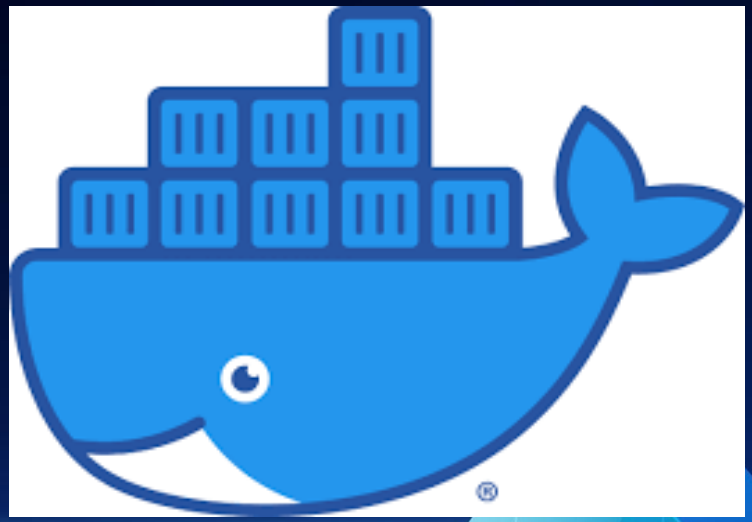
AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

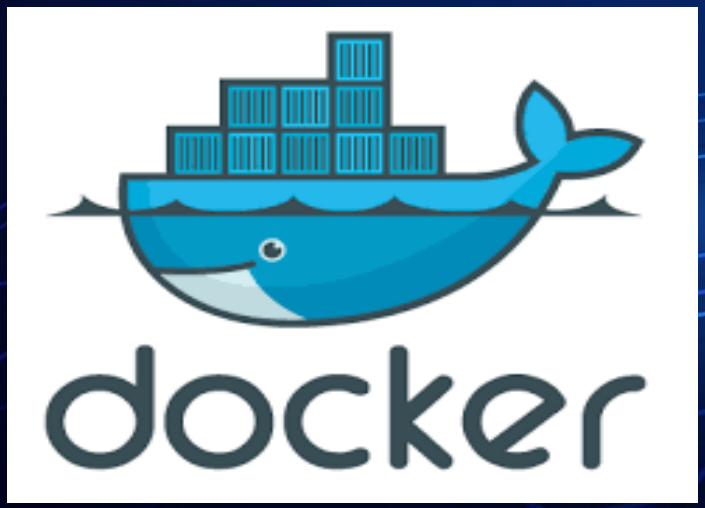
Welcome All

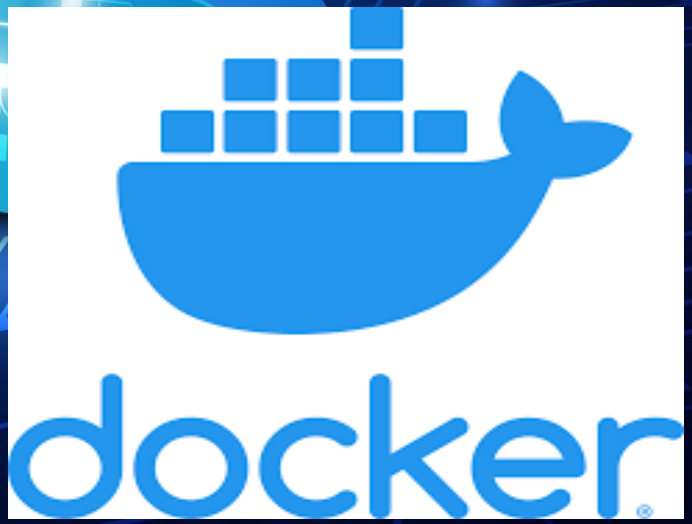
CLOUD COMPUTING

Dr. Harsha Vasudev
Assistant Professor, CSE
Amrita Vishwa Vidyapeetham

Kubernetes




docker


docker

- Container Management Tool
- KUBERNETES – K8s (Internationalization – I 18 n)
- 1980

- DEFINITION : Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.
- It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

A LOOK BACK

- Earlier applications – Monolithic (mono + lithic)
  - Problems ?
- Microservices - API Gateways
  - Problems ?
- Containers
  - Problems ?

What We Need ?

Examples

Cont'd

- Orchestration (clustering of any number of containers running on different networks) Tool
- Container Management Tool
- Docker Vs Kubernetes - X
- Docker – Containerization tool that builds Images
- Docker Swarm – All Docker related things managed

Cont'd

- **Kubernetes –**
  - **Doesn't make images**
  - **Makes Cluster of Containers**
  - **Decides when to increase/decrease the number of Containers – Scalability - Auto Scaling**
  - **Management**
  - **High Availability**
  - **Whole scenario changed – How to make Application/how to run/etc. etc.**

Cont'd

- Google Product
- Borg (internal system) – Omega
- In 2014 – Open Source Platform
- CNCF (Cloud Native Computing Foundation)

Features ….. ?

- **Auto Scaling – Vertical and Horizontal**
- **Load balancing**
- **Platform Independence**
- **Fault Tolerance**
- **Roll Back**
- **Batch Processing**

Installation ….. ?

- **Kubeadm – Bare Metal Installation**
- **Minikube – Virtualized Environment for Kubernetes**
- **Kops – Kubernetes on AWS**
- **Kubernetes on GCP – Kubernetes running on Google Cloud Platform**

Let’s Explore More in the Next Class


Amrita Vishwa Vidyapeetham
Amritapuri Campus
AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY
श्रद्धावान् लभते ज्ञानम्

AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

Welcome All

CLOUD COMPUTING

Dr. Harsha Vasudev
Assistant Professor, CSE
Amrita Vishwa Vidyapeetham

Kubernetes

Greek word - helmsman /captain of the ship

- When you deploy Kubernetes - a Cluster
- Cluster is a set of machines called Nodes
- Master and worker nodes
- At least one master and worker node
- It forms a Cluster
- Multiple Clusters are also possible
- Worker nodes – minions
- Multiple Cluster nodes also possible (failure)

Nodes and Pods

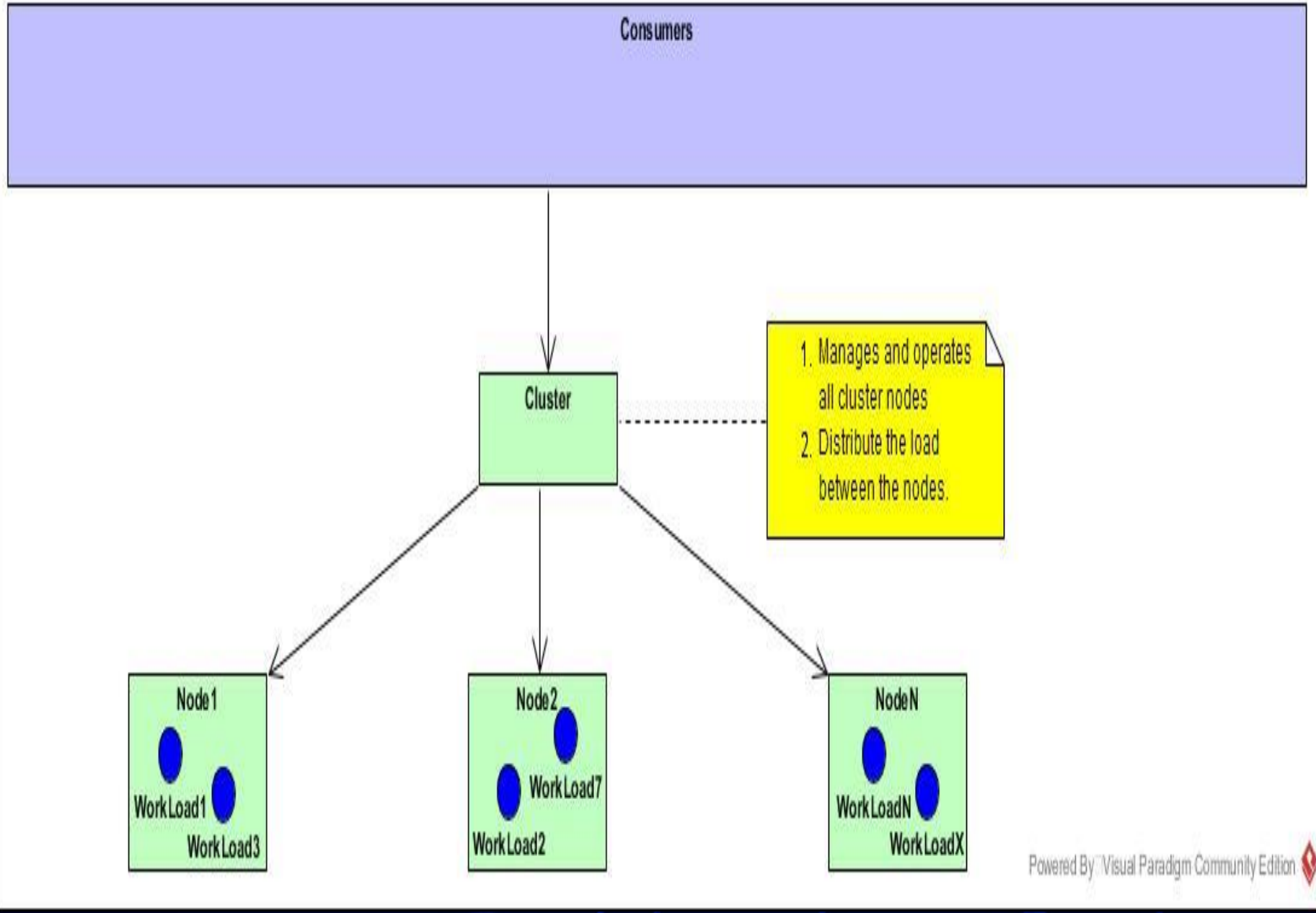
- Master controls Nodes and Pods
- Node – Physical machine, Virtual machine or Virtual machine on cloud
- Pods – Inside every nodes
  - One or more
- Inside Pods - Containers
  - Multiple

Master Nodes - Components

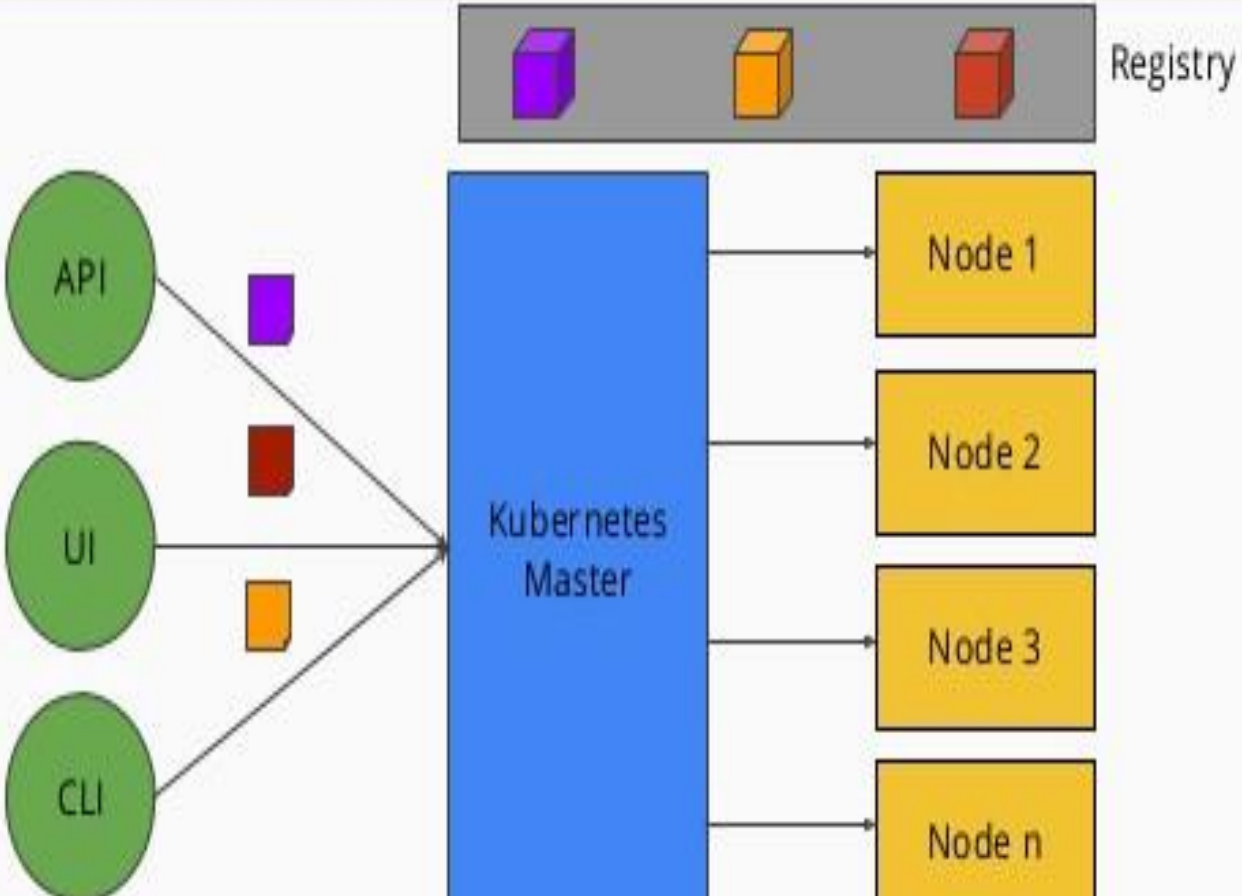
- 4 Components
- API Server – responsible for all communications
- Scheduler – Schedules the pods
- Control Manager
- Etcd – open source – distributed key value database from CoreOS

Worker Nodes

- Kubelet – agent running on each node communicates with components from the master node
- Kube-proxy – A network agent which runs on each node responsible for maintaining network configuration and rules
- Container runtime – The container runtime is the software that is responsible for running containers


Consumers
Cluster
1. Manages and operates all cluster nodes
2. Distribute the load between the nodes.
Node1
WorkLoad1
WorkLoad3
Node2
WorkLoad7
WorkLoad2
NodeN
WorkLoadN
WorkLoadX
Powered By Visual Paradigm Community Edition

Kubernetes Architecture



Let’s Explore More in the Next Class


Amrita Vishwa Vidyapeetham
Amritapuri Campus
AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

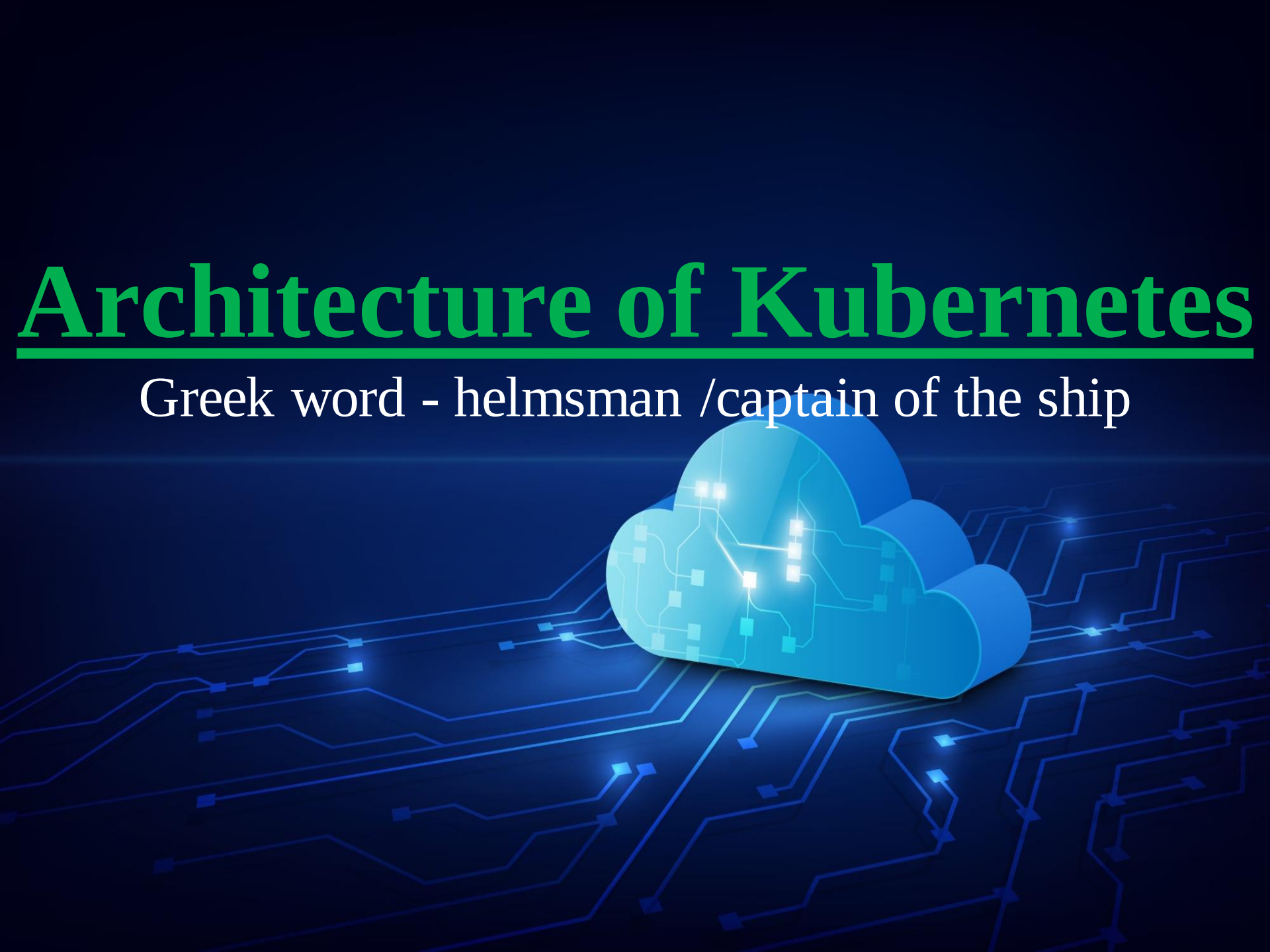
Welcome All

CLOUD COMPUTING

Dr. Harsha Vasudev
Assistant Professor, CSE
Amrita Vishwa Vidyapeetham

Architecture of Kubernetes

Greek word - helmsman /captain of the ship



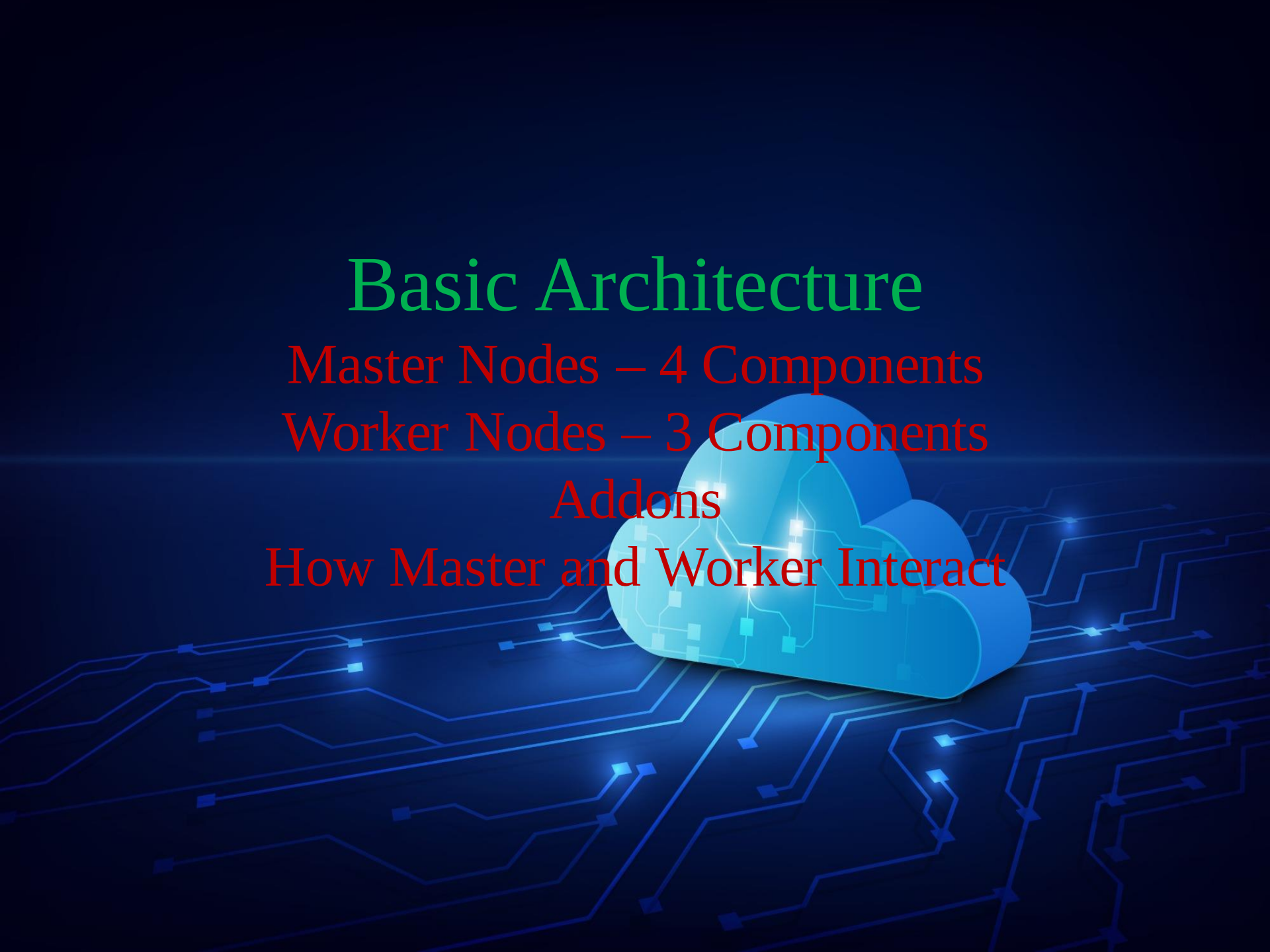
Basic Architecture

Master Nodes – 4 Components

Worker Nodes – 3 Components

Addons

How Master and Worker Interact



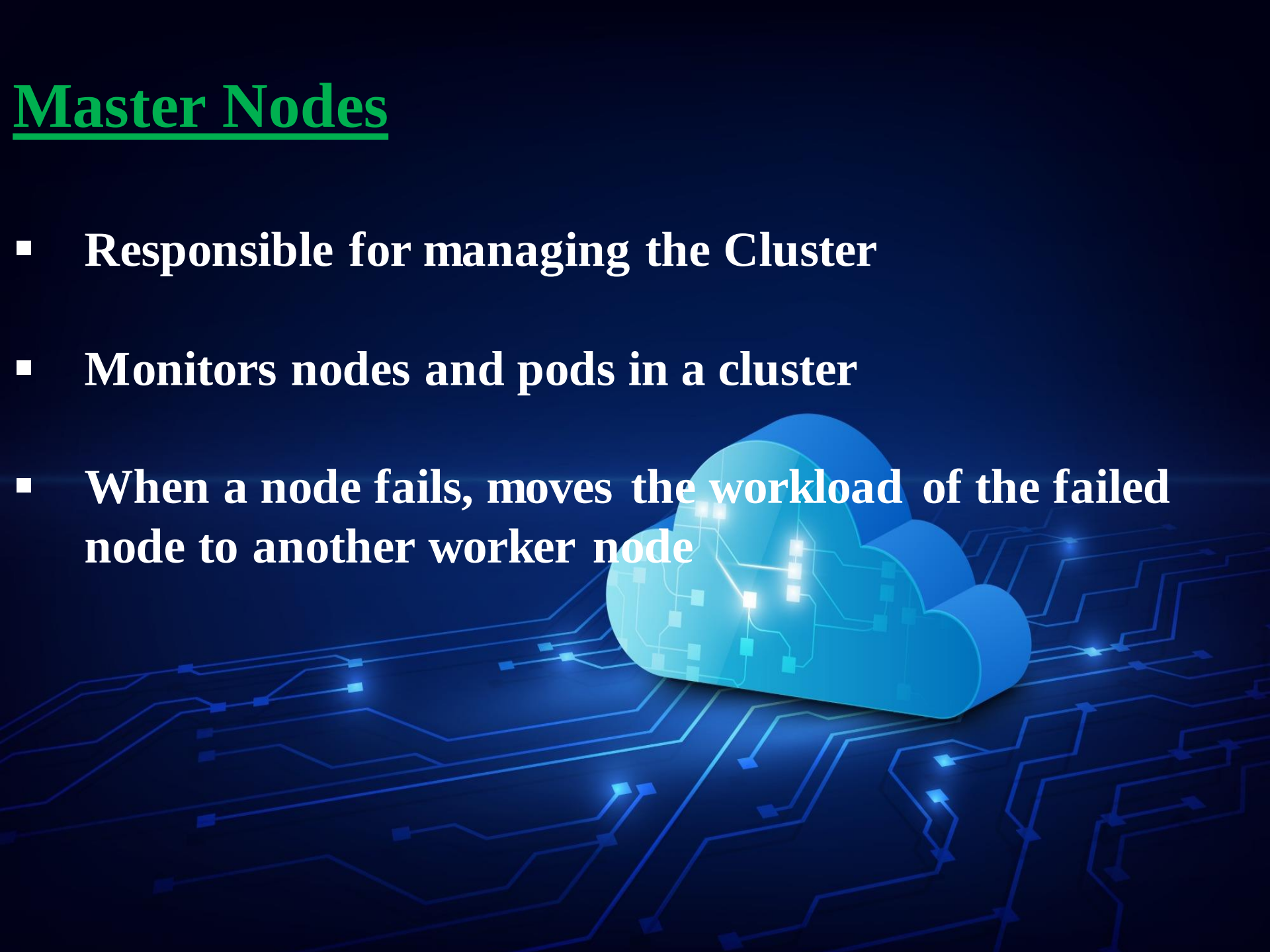
- When we deploy Kubernetes – a Cluster
- Cluster is a set of machines called Nodes
- Master and worker nodes = CLUSTER
- At least one master and worker node
- It forms a Cluster
- Multiple Clusters are also possible
- Worker nodes – minions
- Multiple Master nodes also possible (failure) – in Kubernetes Architecture

Nodes and Pods

- Master controls Nodes and Pods
- Node – Physical machine, Virtual machine or Virtual machine on cloud
- Pods – Inside every nodes
  - One or more
- Inside Pods - Containers
  - Multiple

Master Nodes

- Responsible for managing the Cluster
- Monitors nodes and pods in a cluster
- When a node fails, moves the workload of the failed node to another worker node



Master Nodes - Components

- 4 Components

- API Server – responsible for all communications

- Scheduler – Schedules the pods on nodes. It can read the configuration from the configuration files and accordingly selects the nodes where the pods can be scheduled.

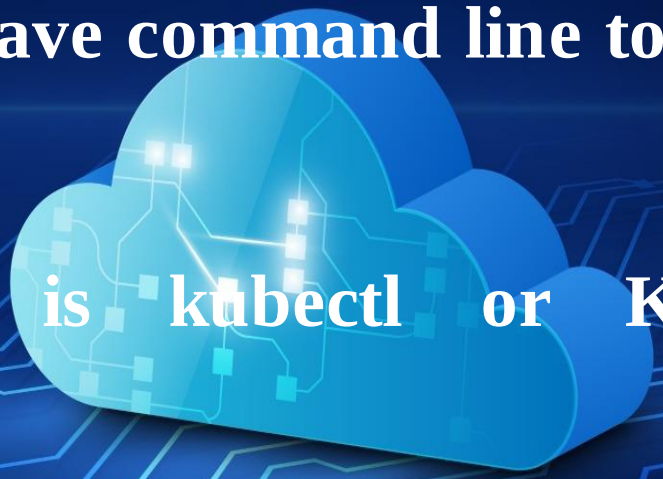
- Control Manager - Runs controllers and manages all activities.

- Etcd – open source database – distributed key value database from CoreOS

API Server

- Gives some API's where users can interact.

- To call the API's – we have command line tool or from a UI.
- Command line tool is kubectl or Kubernetes dashboard –UI
- Written in Go language



Scheduler

- Schedules pods across multiple nodes.
- Monitors newly created pods that have no node assigned, and selects a node for them to run on .
- It gets information for hardware configuration from configuration files and schedule pods on nodes accordingly

Control Manager

- Runs Controllers .
- Two types –
  - Kube-controller-manager and
  - Cloud-controller-manager .

Control Manager (Cont'd)

- **Kube-controller-manager : Node controller, Replication controller, endpoints controller, service account and token controllers.**
- **Cloud-controller-manager :Node controller, Route controller, Service controller, Volume controller**
- **Ensures : Nodes are running all the time, correct number of pods are running as per the configuration files.**
- **IMP : These controllers are separate processes, but to reduce complexity they all are compiled into a single binary and run in a single process**

Control Manager (Cont'd)

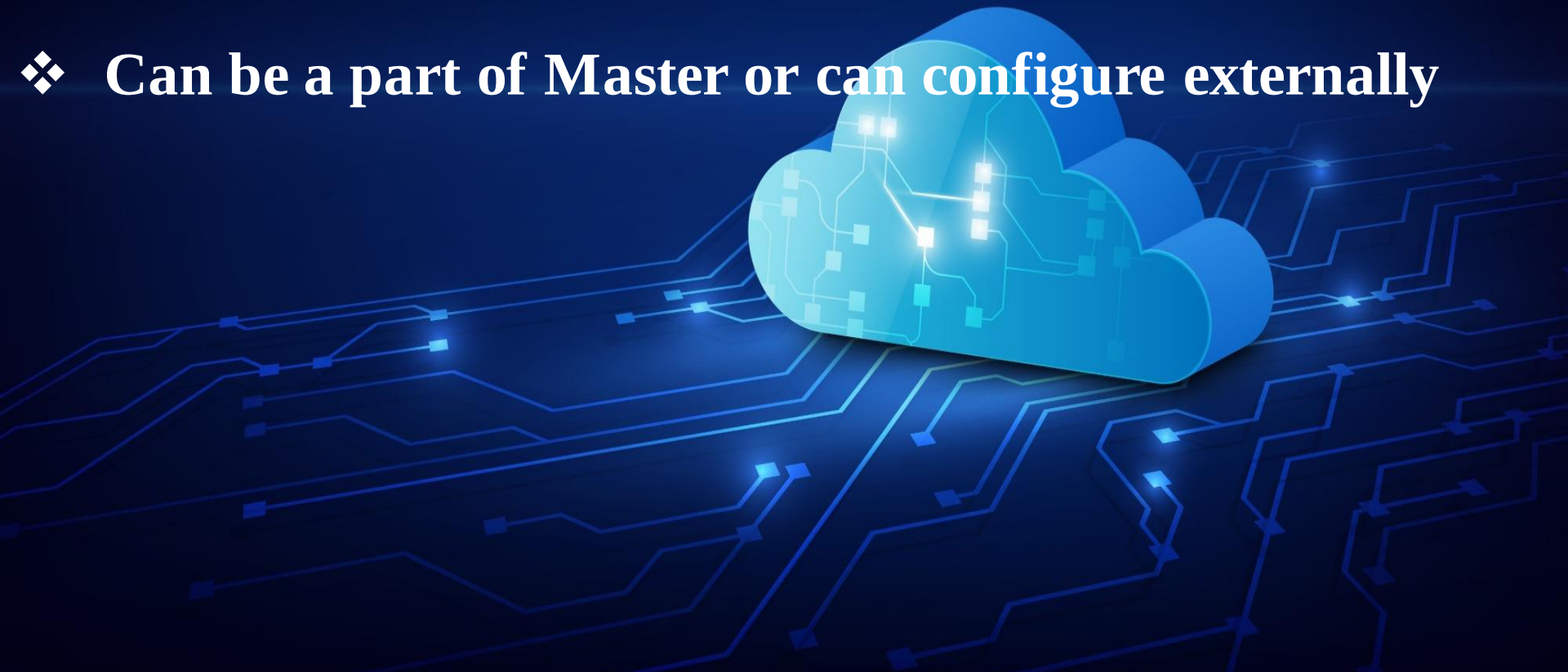
- Kube-controller-manager : runs some monitors, it is responsible to act when nodes become unavailable, ensures pod counts are as expected, to create endpoints, service accounts, and API access tokens

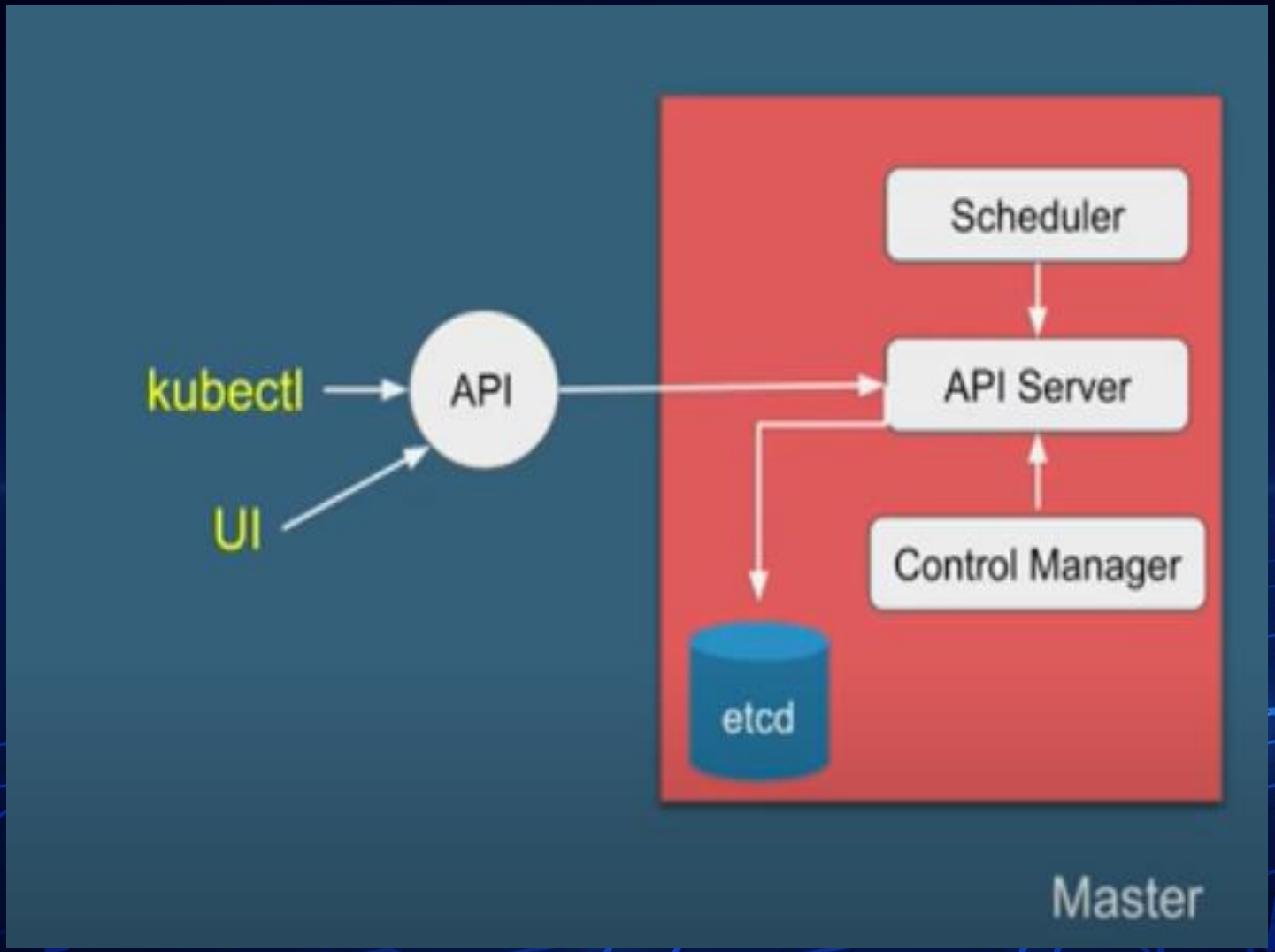
- Cloud-controller-manager - runs controllers responsible to interact with the underlying infrastructure of a cloud provider when nodes become unavailable, to manage storage volumes when provided

How you can disable the controller loops…..?

etcd

- Consistent and highly available
- Backing store for all cluster data
- Only API server can interact directly
- Can be a part of Master or can configure externally




Scheduler
kubectl
API
API Server
UI
Control Manager
etcd
Master

Worker Nodes

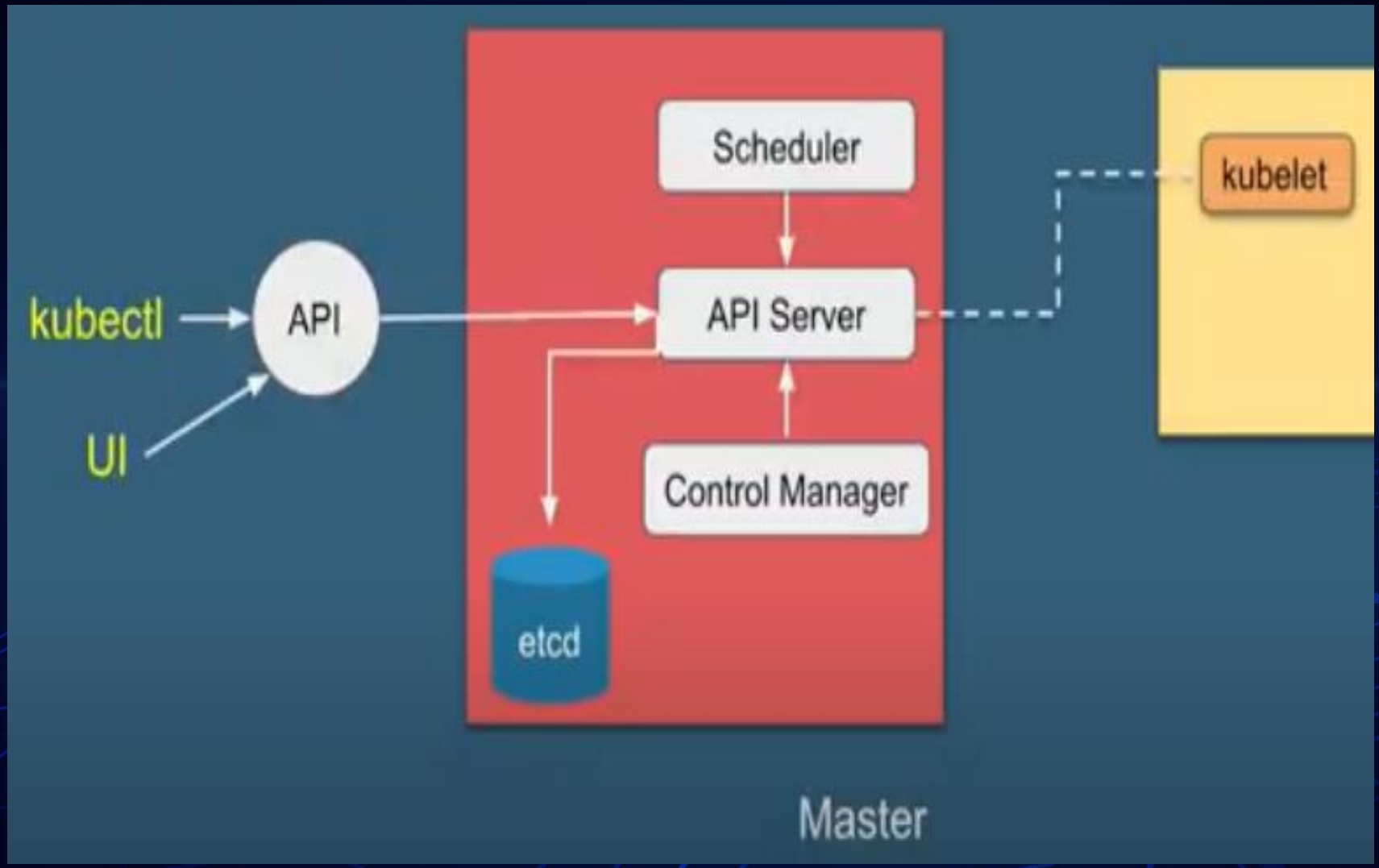
- Kubelet, kube-proxy, Container Runtime
- Kubelet – agent running on each node communicates with components from the master node using API server
- Kubelet takes information from a set of PodSpecs that are provided through various mechanisms and ensures that all the containers are running properly
- In case any pod has any issues, kubelet tries to restart the pods on the same node or a different node.
- Only manage the containers which are created by Kubernetes

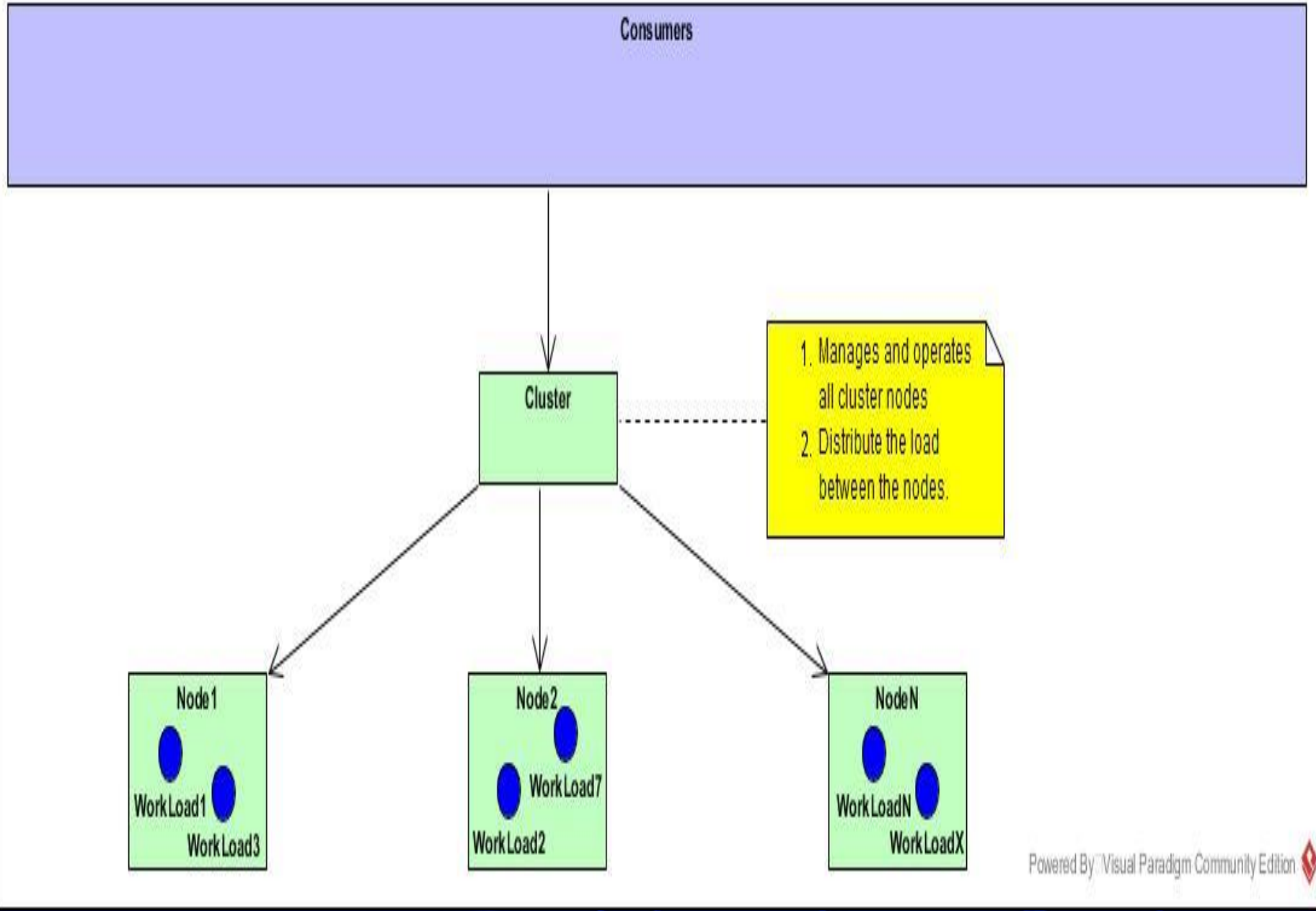
Worker Nodes (Cont'd)

- **Kube-proxy – A network agent which runs on each node responsible for maintaining network configuration and rules**
- **Exposes services to the outside world**
- **Core networking components in Kubernetes**
- **All worker nodes run a Daemon called "kube-proxy", which watches the API server on the master node for the addition and removal of services and endpoints**

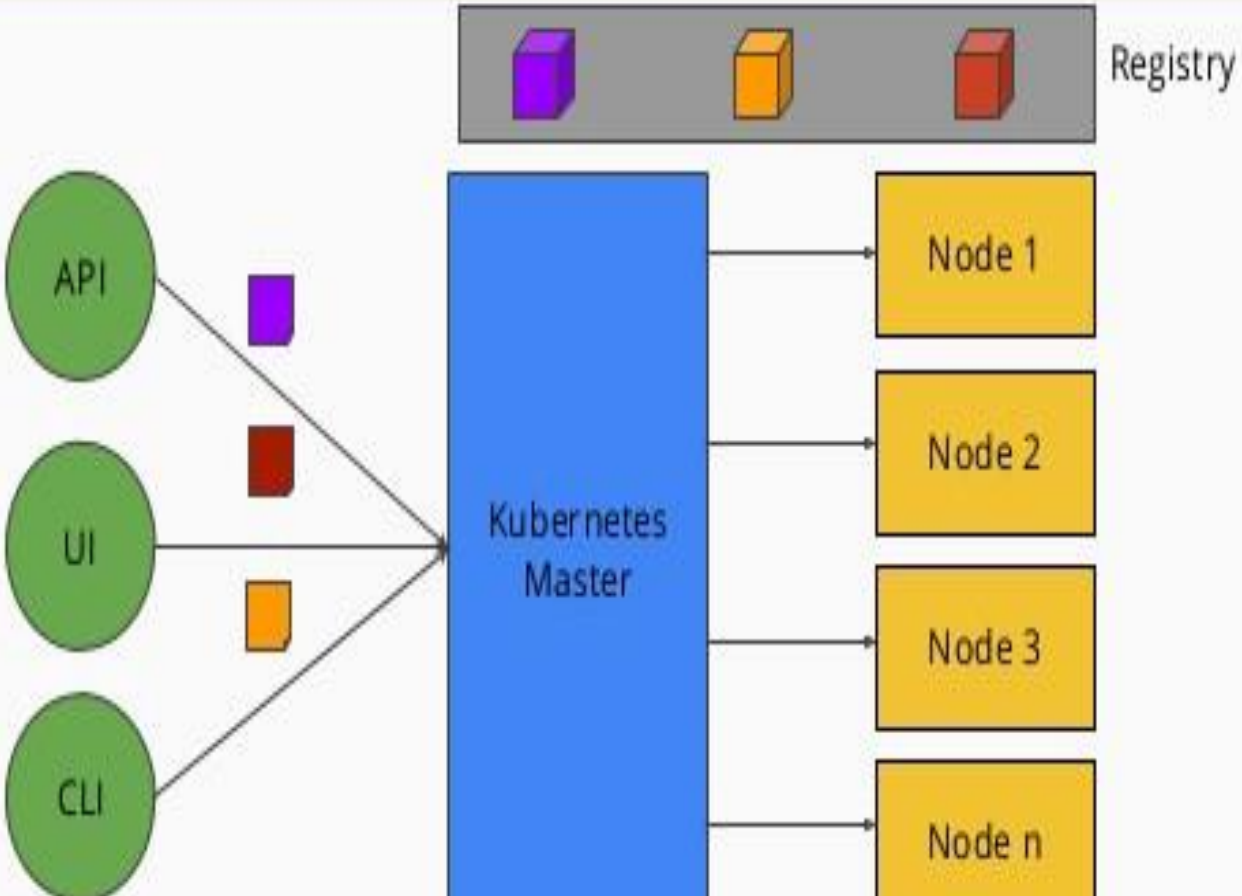
Worker Nodes (Cont'd)

- **Container Runtime –** Is a software that is responsible for running containers
- Kubernetes supports several container runtimes :
  - Docker
  - Containerd
  - Cri-o
  - Rktlet
  - Kubernetes CRI (Container Runtime Interface)


Scheduler
kubelet
kubectl
API
API Server
UI
Control Manager
etcd
Master


Consumers
Cluster
1. Manages and operates all cluster nodes
2. Distribute the load between the nodes.
Node1
WorkLoad1
WorkLoad3
Node2
WorkLoad7
WorkLoad2
NodeN
WorkLoadN
WorkLoadX
Powered By Visual Paradigm Community Edition

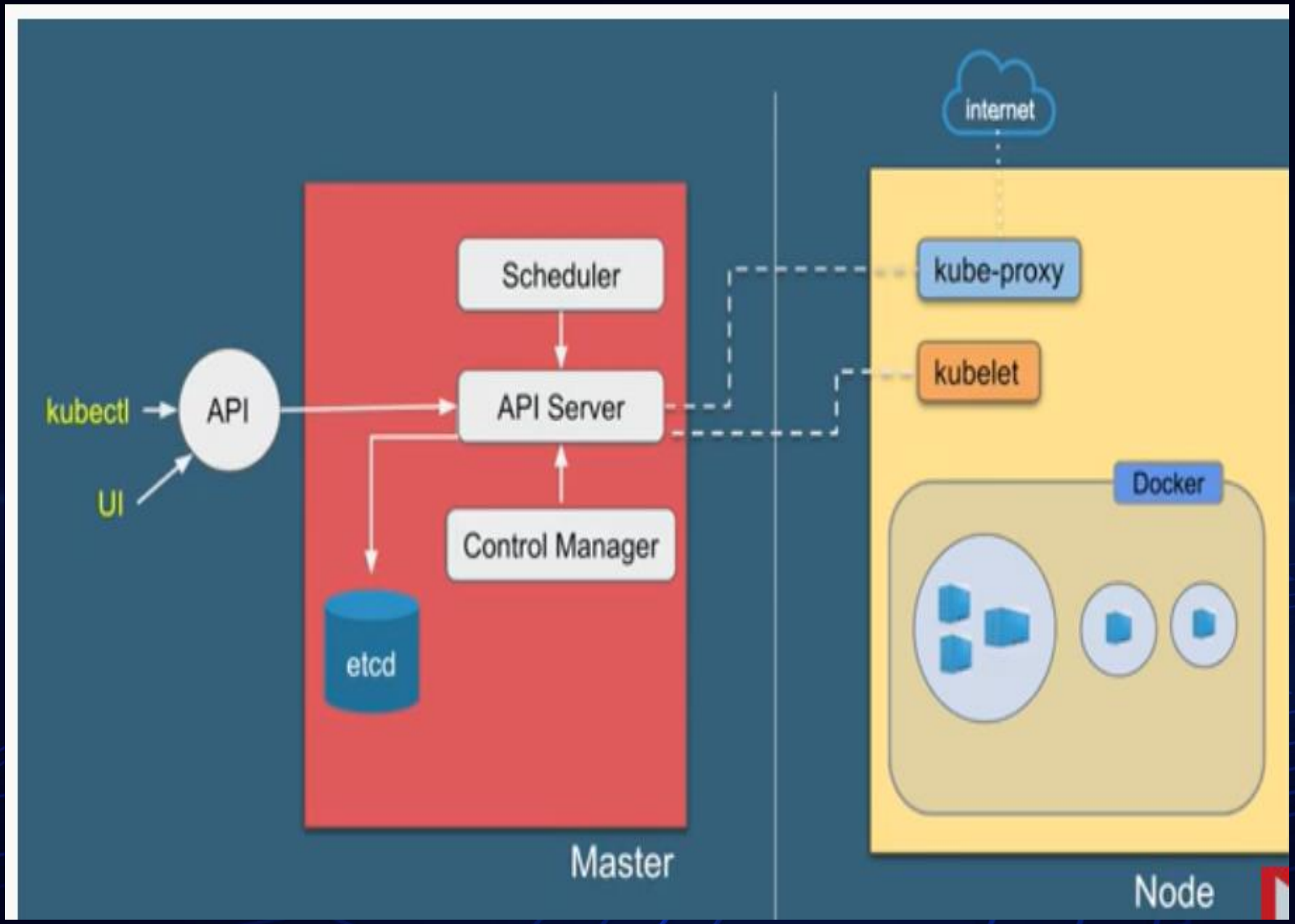
Kubernetes Architecture



Addons

- For DNS, Dashboard, Monitoring, Logging, etc.
- Extends the functionality of Kubernetes
- Dashboard – A general purpose web-based user interface for cluster management
- Monitoring – Collects cluster-level container metrics and saves them to a central data store
- Logging – Collects cluster-level container logs and saves them to a central log store for analysis
- Cluster DNS is a DNS server required to assign DNS records to Kubernetes objects and resources

How Master and Worker Interact ….. ?


internet
Scheduler
kube-proxy
kubectl
API
API Server
kubelet
UI
Docker
Control Manager
etcd
Master
Node

Assignment III

Install Minikube with Kubernetes in Windows and deploy an application

Let’s Explore More in the Next Class