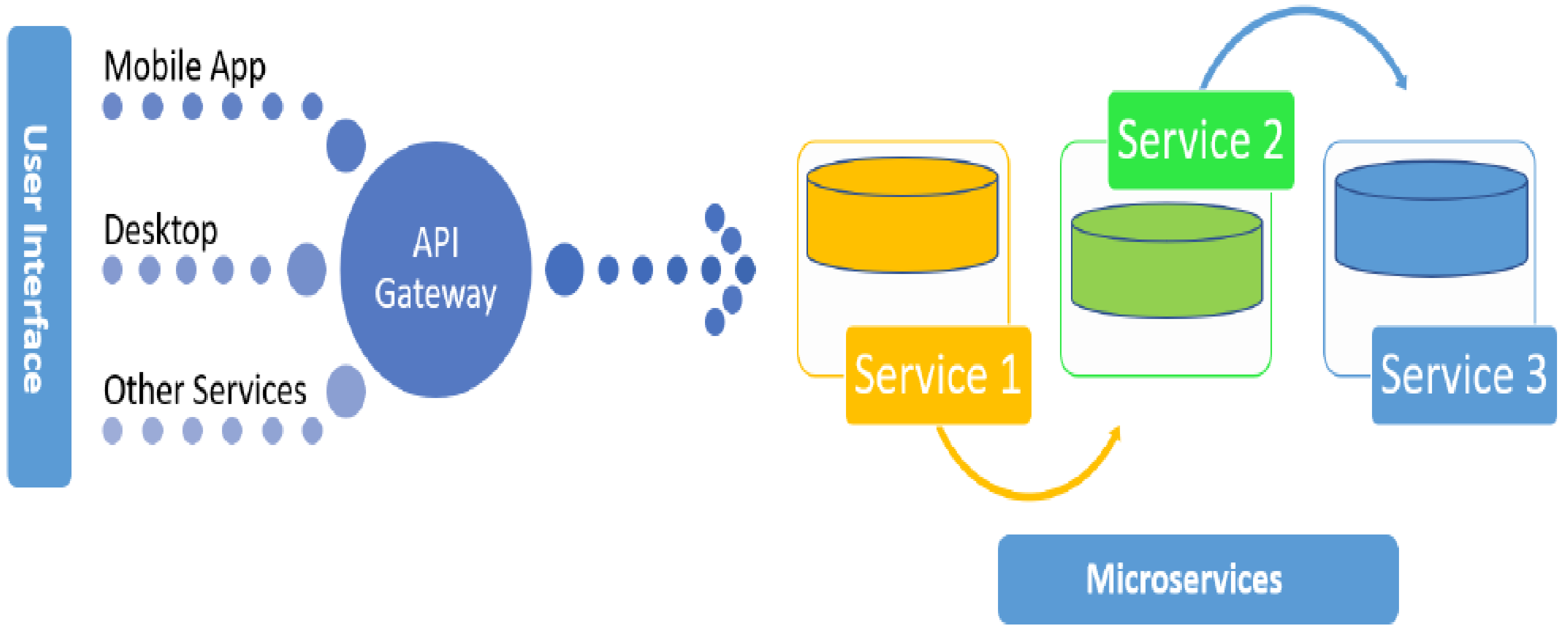
Microservices

Microservice

- Microservice is a service-based application development methodology.
- In a service-oriented architecture, entire software packages will be sub-divided into small, interconnected business units. Each of these small business units will communicate to each other using different protocols to deliver successful business to the client.
- A software development style that has grown from contemporary trends to set up practices those are meant to increase the speed and efficiency of developing and managing software solutions at scale.
- Microservices is more about applying a certain number of principles and architectural patterns.
- Each microservice lives independently, but on the other hand, also all rely on each other.
- All microservices in a project get deployed in production at their own pace, on-premise on the cloud, independently, living side by side.

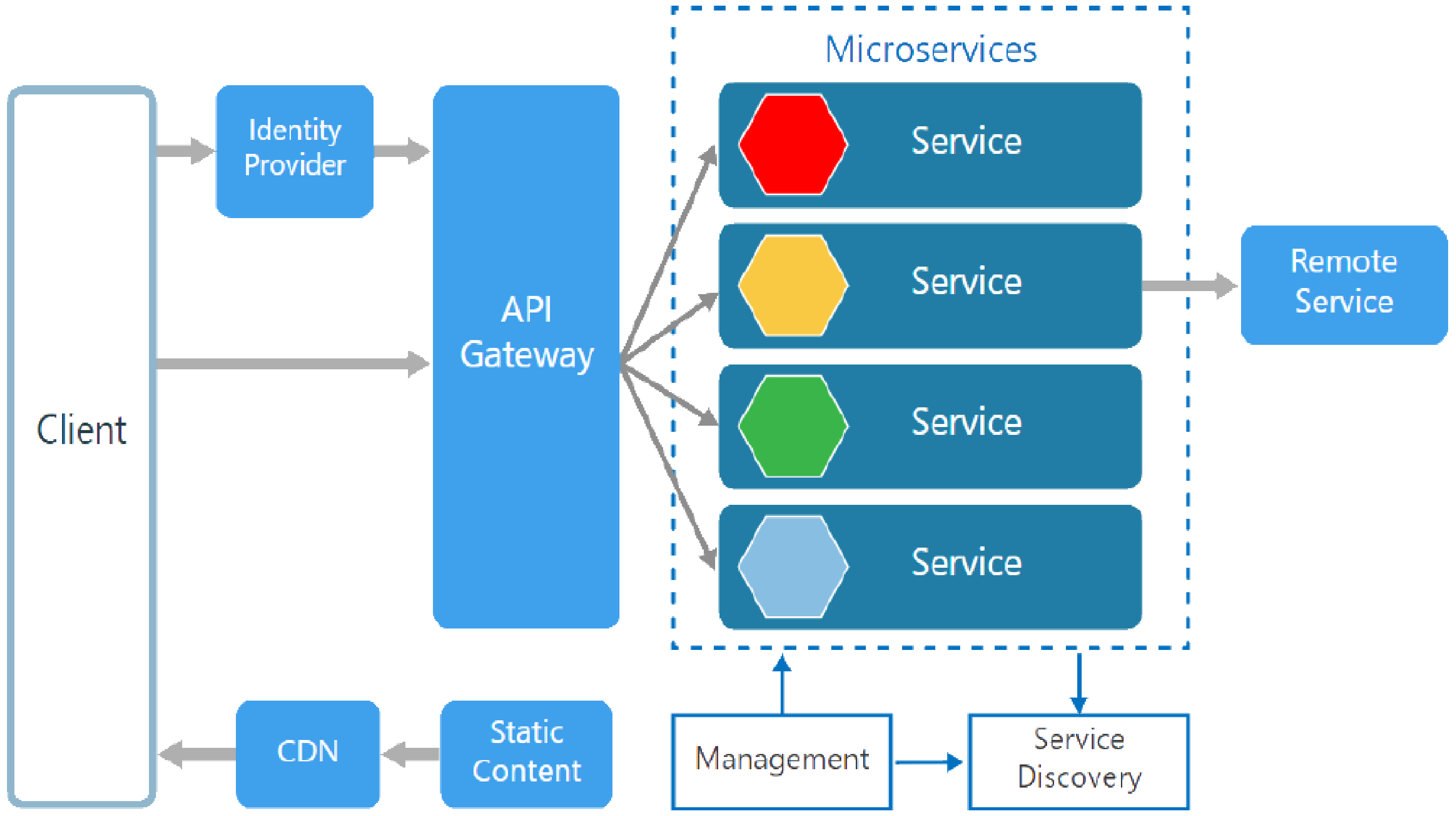
Cont….

- Microservices are deployed independently with their own database per service so the underlying microservices look as shown in the following picture.



Microservices Architecture

The following picture from Microsoft Docs shows the microservices architecture style.



Components in a Microservices Architecture

There are various components in a microservices architecture apart from microservices themselves.

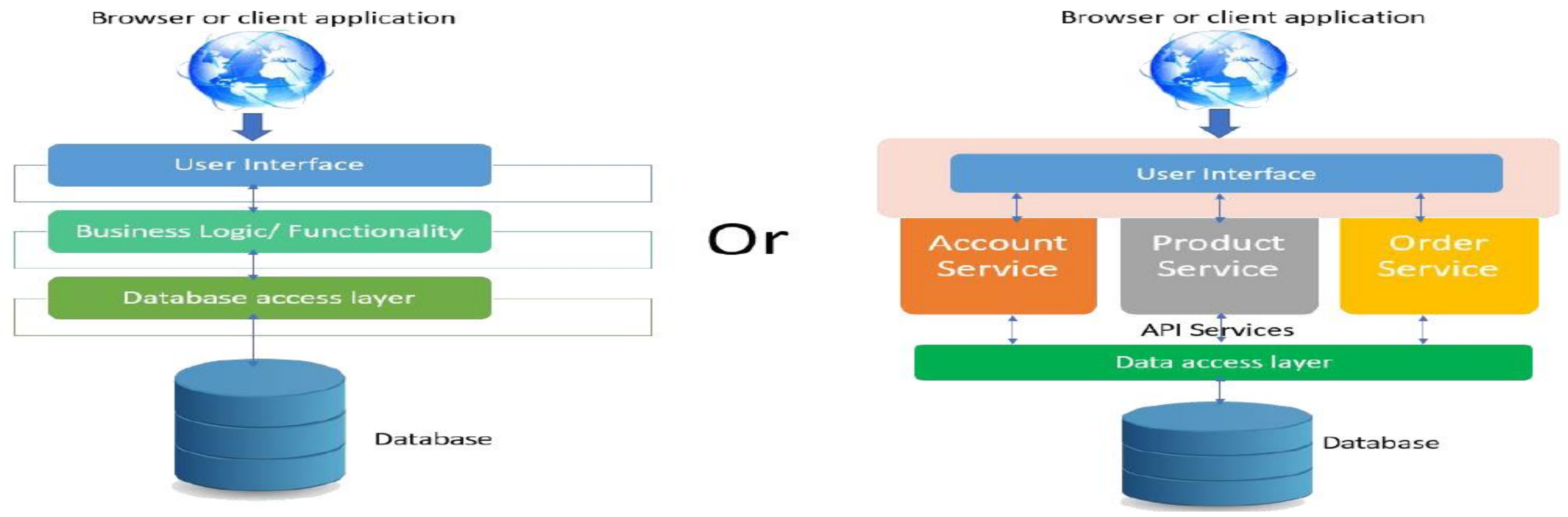
- **Management**. Maintains the nodes for the service.
- **Identity Provider**. Manages the identity information and provides authentication services within a distributed network.
- **Service Discovery**. Keeps track of services and service addresses and endpoints.
- **API Gateway**. Serves as client’s entry point. Single point of contact from the client which in turn returns responses from underlying microservices and sometimes an aggregated response from multiple underlying microservices.

Cont….

- **CDN**. A content delivery network to serve static resources for e.g. pages and web content in a distributed network
- **Static Content** The static resources like pages and web content

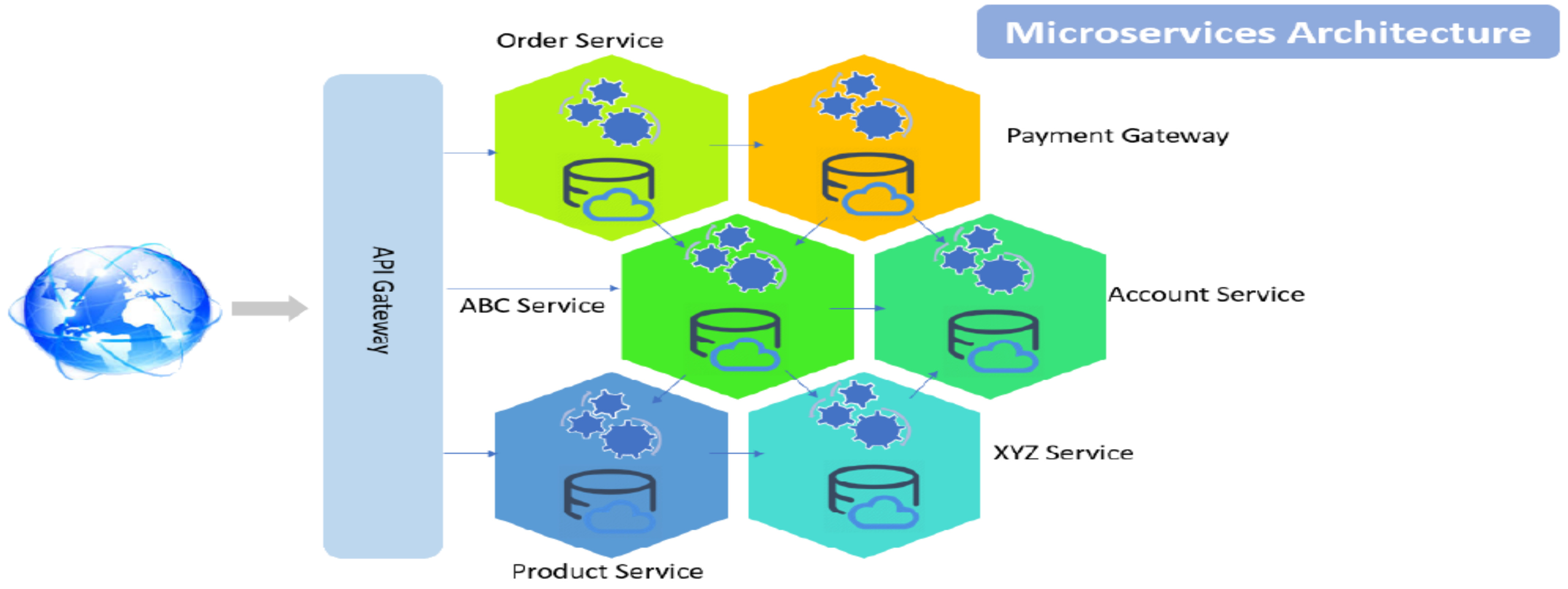
Monolithic vs Microservices Architecture

- Monolithic applications are more of a single complete package having all the related needed components and services encapsulated in one package.
- Following is the diagrammatic representation of monolithic architecture being package completely or being service based.



Microservice is an approach to create small services each running in their own space and can communicate via messaging. These are independent services directly calling their own database.

Following is the diagrammatic representation of microservices architecture.



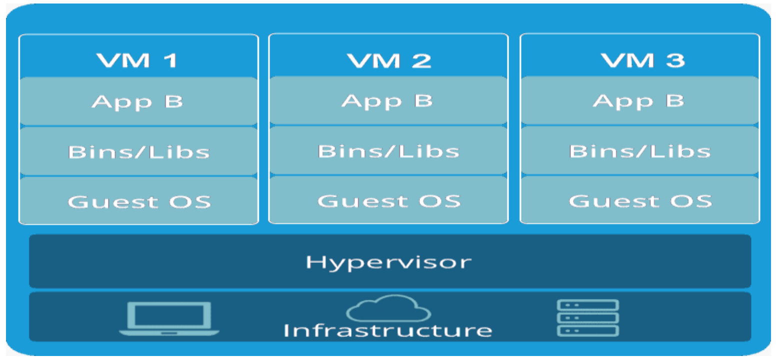
Cont….

- In monolithic architecture, the database remains the same for all the functionalities even if an approach of service-oriented architecture is followed, whereas in microservices each service will have their own database.

Virtual Machine Container & Kubernetes

Virtual Machine

- A Virtual Machine is a software program that emulates the functionality of physical hardware or a computing system. In simple words, VM makes it possible to run what appears to be many separate computers on the hardware of a single computer.
- VMs interact with physical computers by using lightweight software layers called the ***Hypervisors***. These hypervisors can separate VMs from one another and allocate processors, memory, and storage between them. The VM may also contain the necessary system binaries and libraries to run the apps. The host operating system (OS), is managed and executed using the hypervisor.
- Popular VM providers are VMware vSphere, Oracle VM VirtualBox, Microsoft Hyper-V



Pros & Cons of VM

Pros Of Virtual Machine

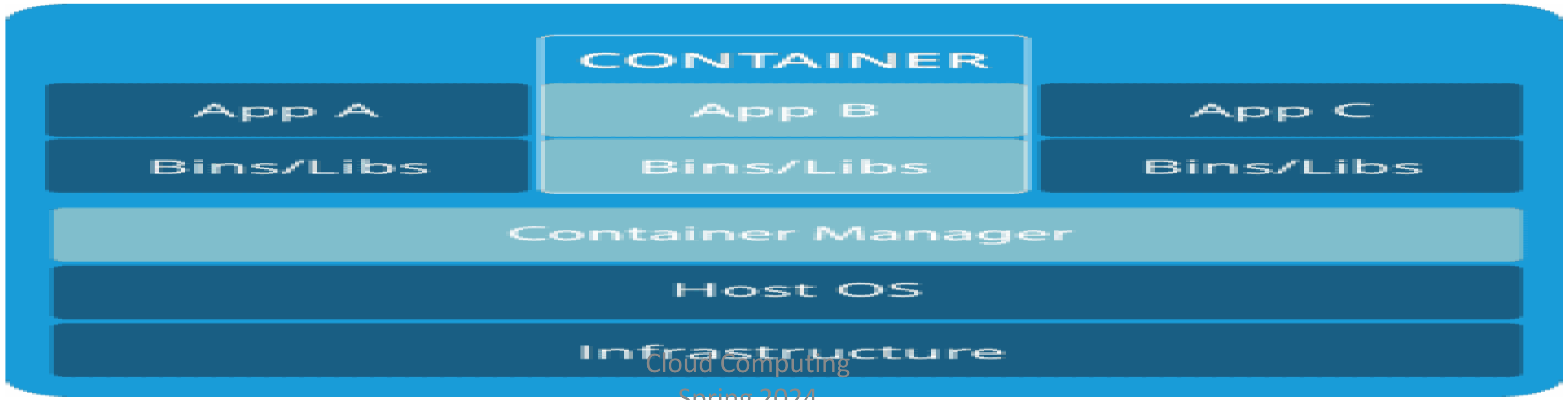
- **Multiple operating system** environments can be used on the same computer.
- VM improves system **reliability** and prevents system crashes. Even if it crashes, the host OS will **not be affected** because of isolation.
- Provides a **layer of security**, if the VM is affected by some malware it will not result in a breach of security in the host OS.

Cons Of VM

- Running multiple virtual machines can lead to an unstable output.
- Virtual machines are less efficient and slow compared to a physical machine.
- A virtual machine can be infected with the weaknesses of the host machine.

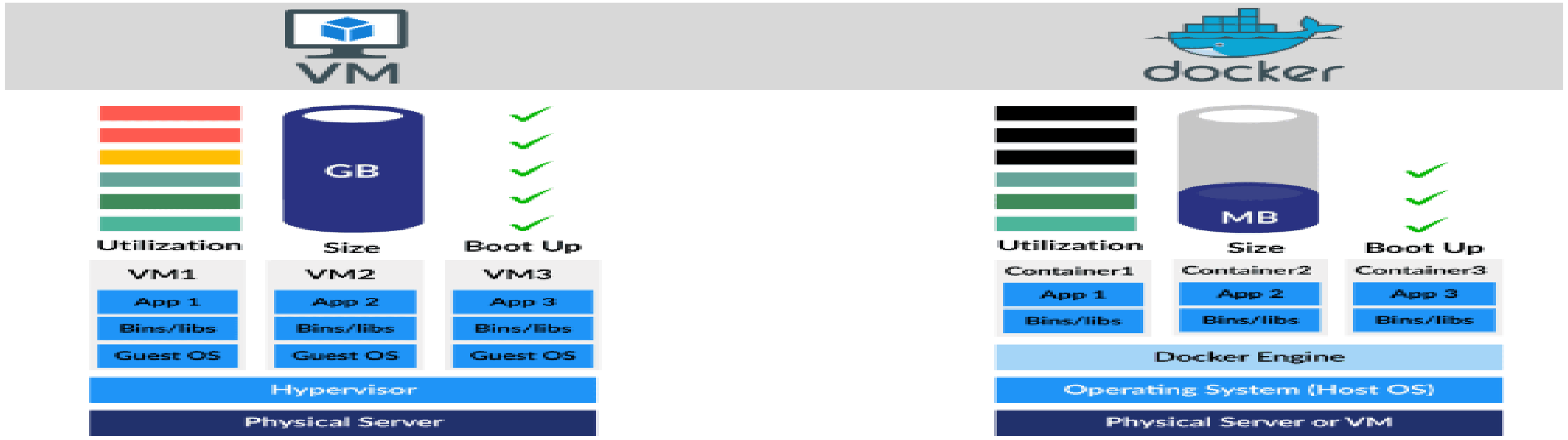
Containers

- **Containers** are a form of operating system virtualization. A single container might be used to run anything from a small microservice or software process to a larger application.
- They are an abstraction at the app layer that package code and dependencies together. The containers share the host OS kernel and, usually, the binaries and libraries, too. Containers are exceptionally **lightweight** — they are only **megabytes in size** and take just seconds to boot.



Difference Between Containers And Virtualization

- The major difference between the Docker vs. VM is that in VMs a hypervisor is used to virtualize physical hardware. Each VM contains a guest OS, a virtual copy of the hardware that the OS requires to run, while in Containers instead of virtualizing the underlying hardware, they virtualize the operating system so each container contains only the application and its libraries.



Container vs Virtual Machines (VMs)

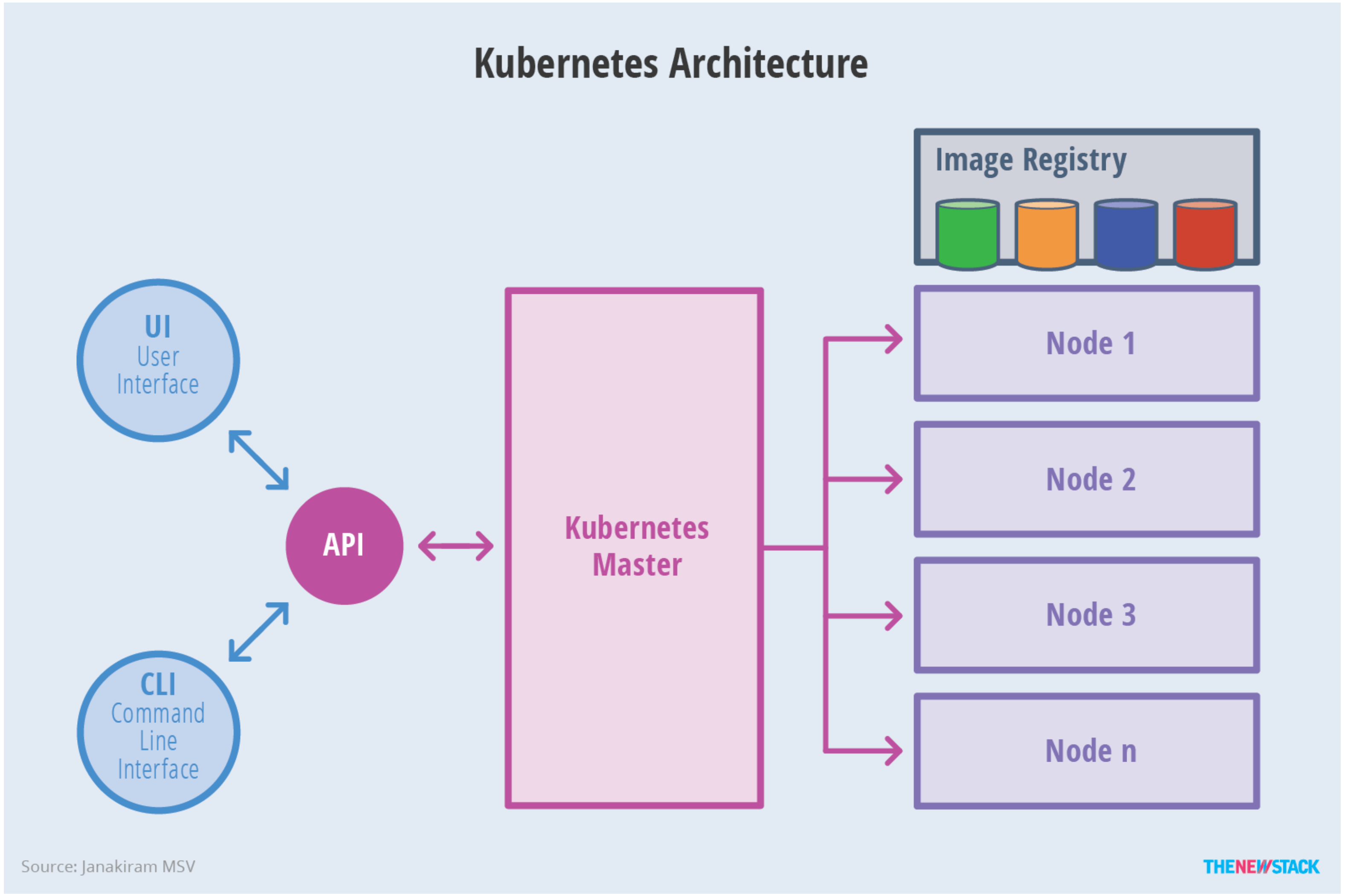
Kubernetes

- In organizations, multiple numbers of containers running at a time so it is very hard to manage all the containers together we use Kubernetes.
- Kubernetes is an open-source platform for managing containerized workloads and services. Kubernetes takes care of scaling and failover for your application running on container.

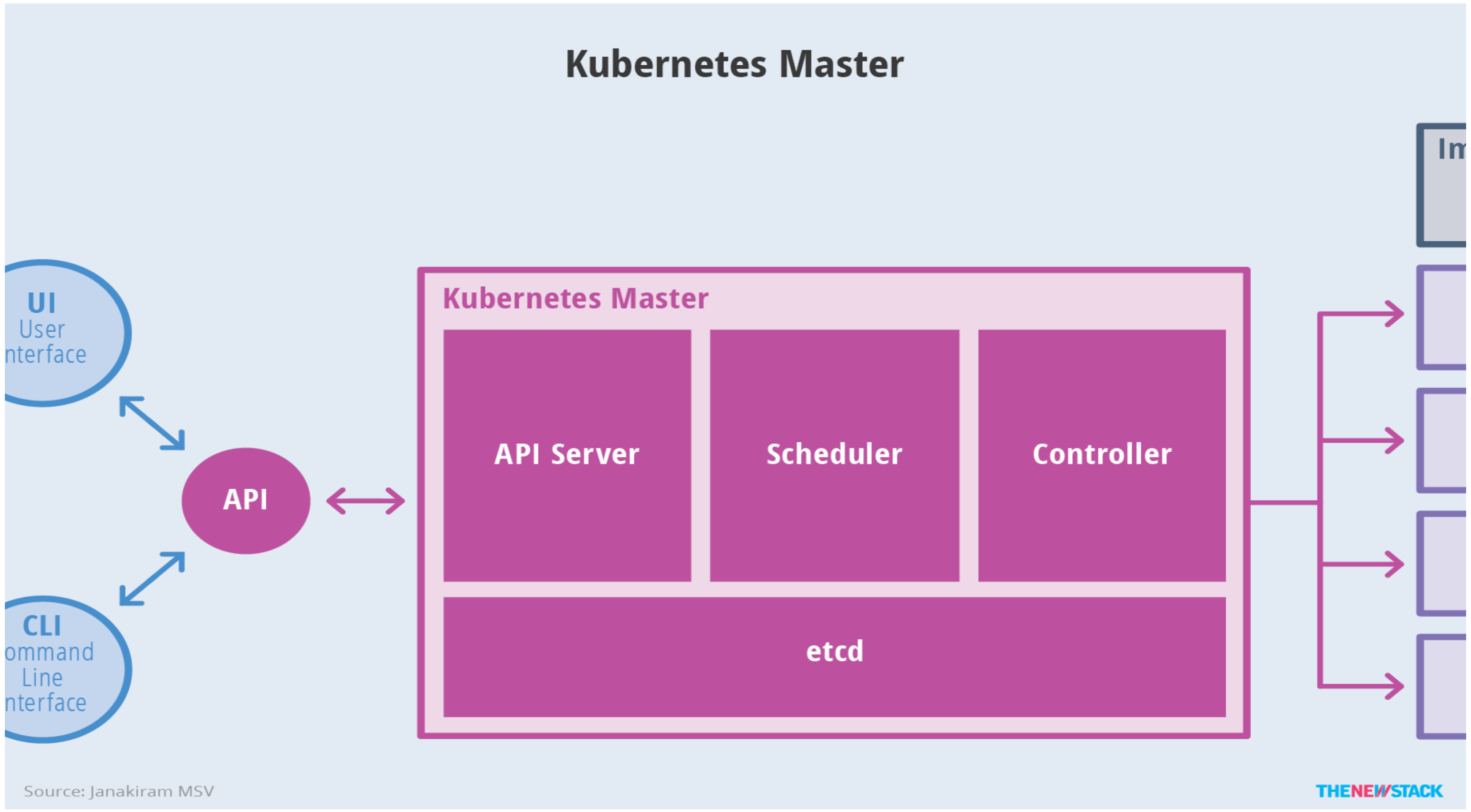
Kubernetes Architecture

- This architecture of Kubernetes provides a flexible, loosely-coupled mechanism for service discovery.
- Like most distributed computing platforms, a Kubernetes cluster consists of at least one master and multiple compute nodes.
- The master is responsible for exposing the application program interface (API), scheduling the deployments and managing the overall cluster.
- Each node runs a container runtime, such as Docker or rkt, along with an agent that communicates with the master.
- The node also runs additional components for logging, monitoring, service discovery and optional add-ons. Nodes are the workhorses of a Kubernetes cluster. They expose compute, networking and storage resources to applications. Nodes can be virtual machines (VMs) running in a cloud or bare metal servers running within the data center.

Cont….

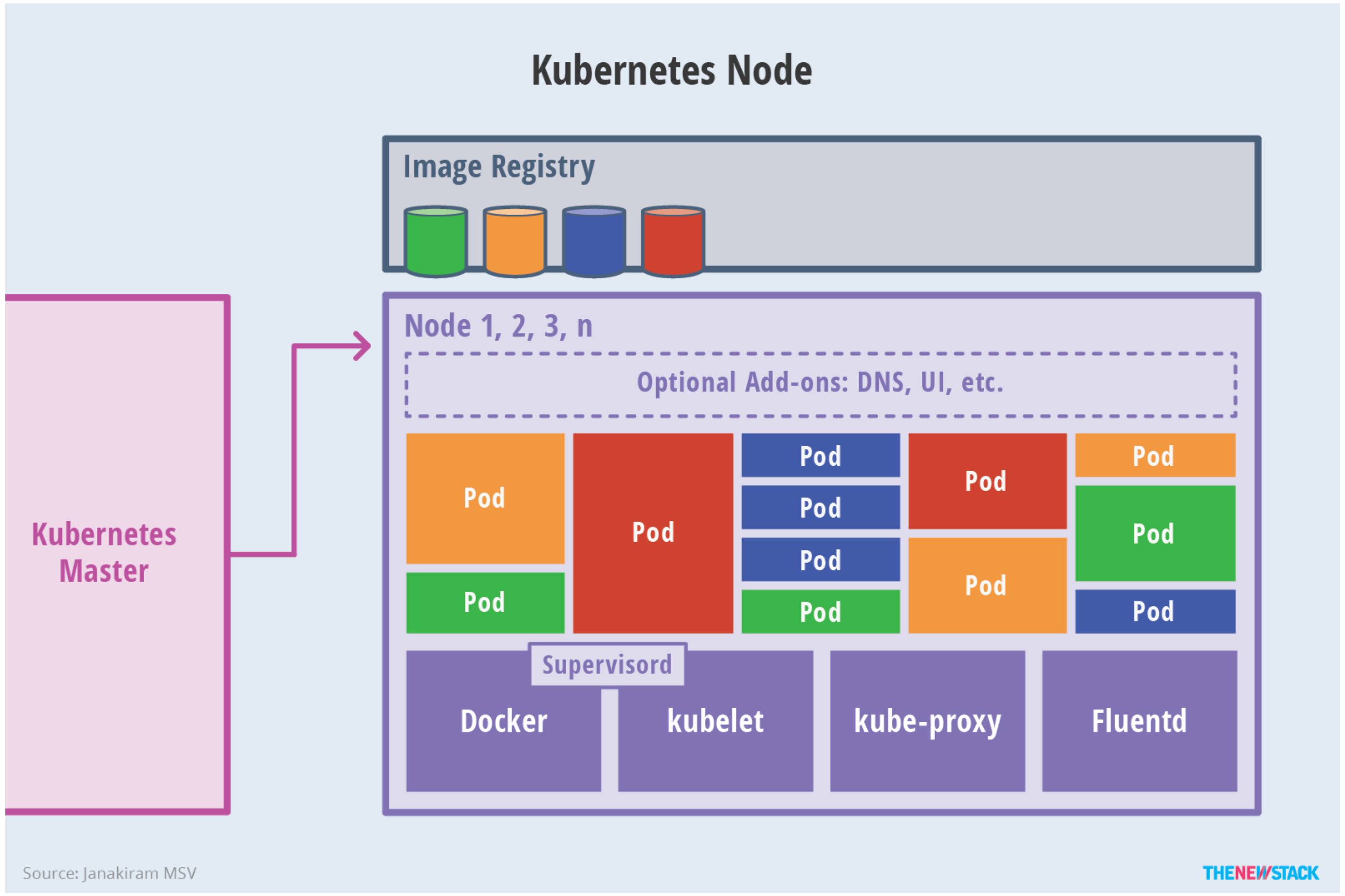

Kubernetes Architecture
Image Registry
UI User Interface
CLI Command Line Interface
API
Kubernetes Master
Node 1
Node 2
Node 3
Node n
Source: Janakiram MSV
THENEWSTACK

Kubernetes Master



- **API server** is the gateway to the Kubernetes cluster. The API server implements a RESTful API over HTTP, performs all API operations, and is responsible for storing API objects into a persistent storage backend.
- The **Kubernetes Scheduler** is a core component of **Kubernetes**: After a user or a controller creates a Pod, the **Kubernetes Scheduler**, monitoring the Object Store for unassigned Pods, will assign the Pod to a Node. Then, the Kubelet, monitoring the Object Store for assigned Pods, will execute the Pod.
- In **Kubernetes**, **controllers** are control loops that watch the state of your cluster, then make or request changes where needed. Each **controller** tries to move the current cluster state closer to the desired state.
- **etcd** is an open source, distributed key-value database from CoreOS, which acts as the single source of truth (SSOT) for all components of the Kubernetes cluster. The master queries etcd to retrieve various parameters of the state of the nodes, pods and containers.

Cont….


Kubernetes Node
Image Registry
Node 1, 2, 3, n
Optional Add-ons: DNS, UI, etc.
Pod
Supervisord
Docker
kubelet
kube-proxy
Fluentd
Kubernetes Master
Source: Janakiram MSV
THENEWSTACK

Cont....

- A **pod** is the smallest execution unit in Kubernetes. A pod encapsulates one or more applications. Pods are ephemeral (Short-live) by nature, if a pod (or the node it executes on) fails, Kubernetes can automatically create a new replica of that pod to continue operations. Pods include one or more containers (such as Docker containers).
- **Replica sets** deliver the required scale and availability by maintaining a pre-defined set of pods at all times. A single pod or a replica set can be exposed to the internal or external consumers via services. Services enable the discovery of pods by associating a set of pods to a specific criterion. Pods are associated to services through key-value pairs called labels and selectors. Any new pod with labels that match the selector will automatically be discovered by the service. This architecture provides a flexible, loosely-coupled mechanism for service discovery.

Cont….

- The definition of Kubernetes objects, such as pods, replica sets and services, are submitted to the master. Based on the defined requirements and availability of resources, the master schedules the pod on a specific node. The node pulls the images from the container image registry and coordinates with the local container runtime to launch the container.
- This architecture of Kubernetes makes it modular and scalable by creating an abstraction between the applications and the underlying infrastructure.

Cont.....

- **kube-proxy** is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept. kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.
- The **kubelet** is the primary "node agent" that runs on each node. It can register the node with the apiserver using one of: the hostname; a flag to override the hostname; or specific logic for a cloud provider. The kubelet works in terms of a PodSpec. A PodSpec is a YAML or JSON object that describes a pod.
- **Fluentd** is a cross platform open-source data collection software project originally developed at Treasure Data. It is written primarily in the Ruby programming language.