Current Trends in Software Engineering (CTSE)

Isanka Rathnayaka

Manager – Software Development .NET | Full Stack | Azure

SYDPRO Pvt. Ltd

BSc in IT – Curtin University of Technology, Bentley Western Australia

ishikim86@gmail.com

Agenda

- Why we want to know about current trends.
- Role of cloud computing in current trends.
- Benefits of Cloud Computing
- Who is using Cloud Computing
- Risks of using Cloud Computing
- Types of Clouds
- Cloud Service Types & Which ones are we focusing ?

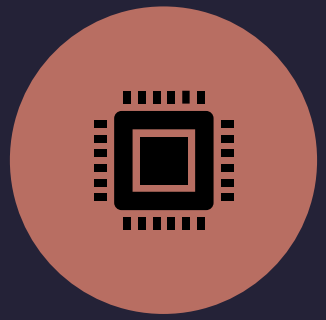



Why we want to know about current Trends..

1. Staying competitive: Being up-to-date with the latest IT trends can help businesses stay competitive in their respective industries. This can help them adapt quickly to changing market conditions and stay ahead of their competitors.

2. Identifying opportunities: Knowing about the latest trends in IT can help individuals and organizations identify new opportunities for growth and innovation. This can lead to the development of new products or services that meet the needs of a changing market.

3. Enhancing efficiency: Many new IT trends are focused on improving efficiency and productivity. By adopting these trends, businesses can streamline their processes and reduce costs, which can ultimately increase their profitability.



4. Improving customer experience: With the increasing use of technology in all aspects of life, customers have come to expect a seamless and personalized experience. Keeping up with the latest IT trends can help businesses meet these expectations and provide a better customer experience.

5. Future-proofing: The world of IT is constantly evolving, and businesses that don't stay up-to-date risk falling behind. By staying current with the latest trends, businesses can future-proof themselves and ensure that they are prepared for whatever changes come their way.

Role of Cloud Computing in Current Trends

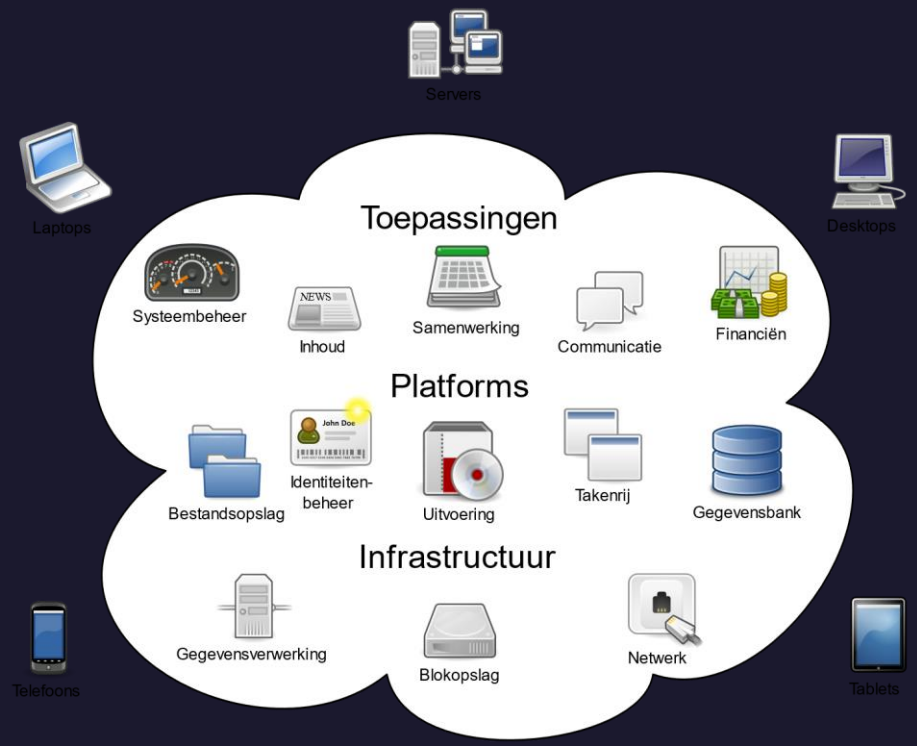
What is Cloud Computing ??

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services over the internet.

Cloud computing is the delivery of computing services

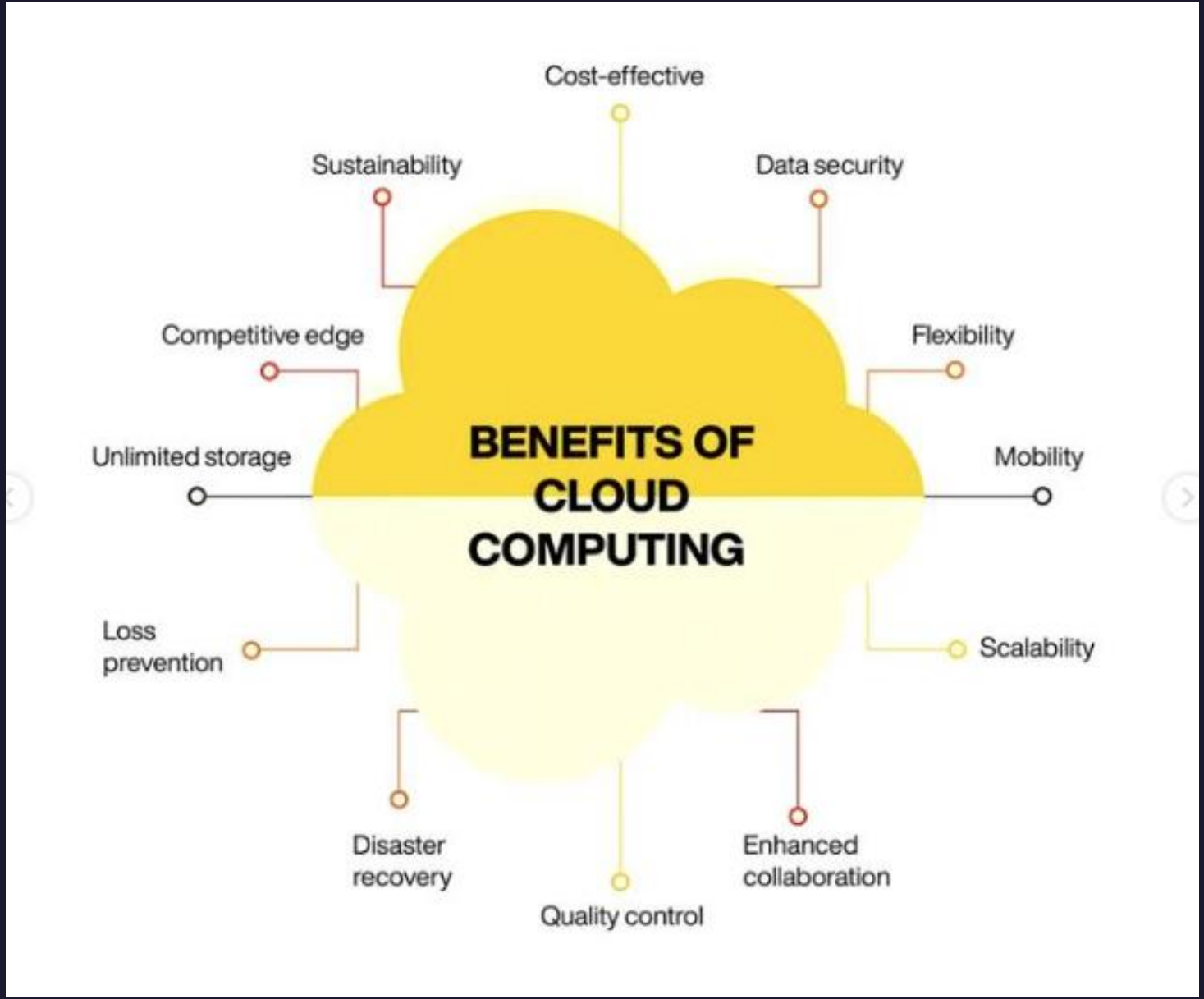
Including

- Servers
- Storage
- Databases
- Networking
- Software
- Analytics



Cloud Computing

Benefits of Cloud Computing...



Who is using Cloud Computing

Organizations of every type, size, and industry are using the cloud for;

- Data backup
- Disaster recovery
- Email
- Virtual desktops
- Software development and testing
- Big data analytics
- Customer-facing web applications.

Risks of cloud computing...

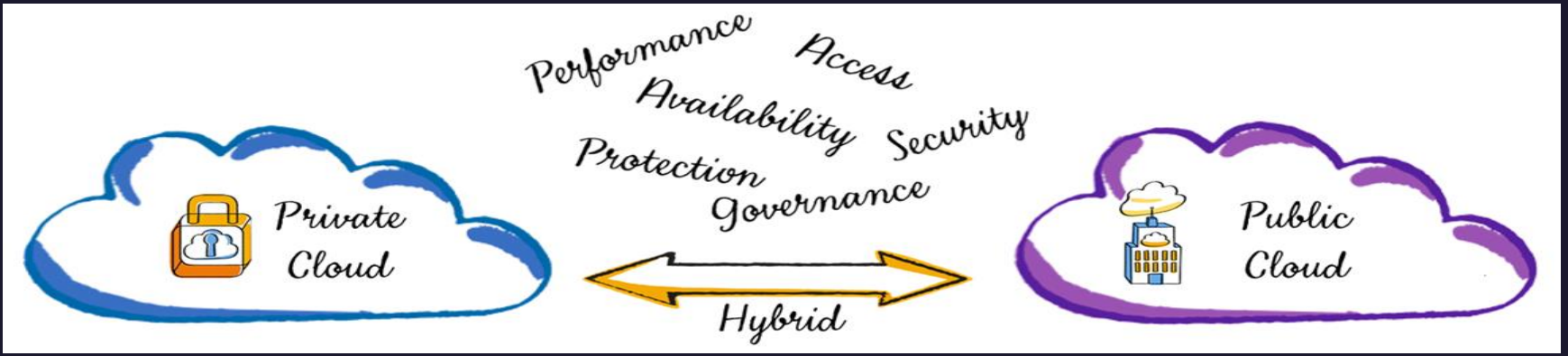
Types of Clouds...

PRIVATE CLOUD

- Inside Organization's corporate network.
- The Organization that owns the private cloud must purchase the cloud hardware, single – tenancy.

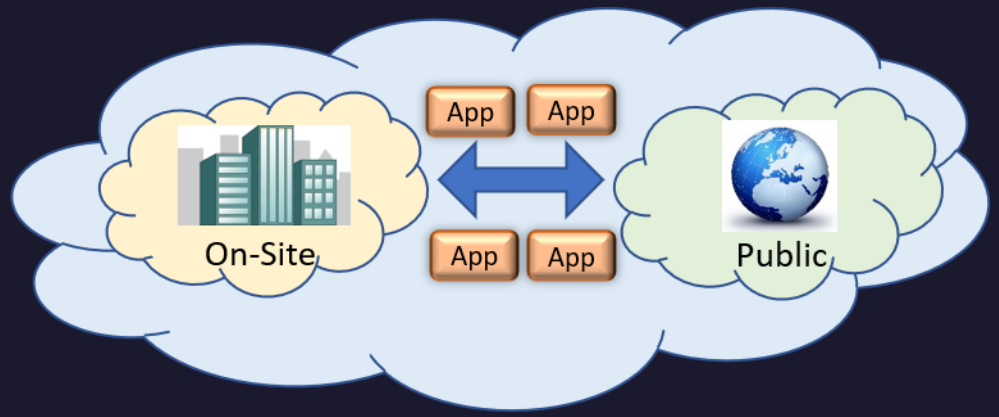
PUBLIC CLOUD

- Anywhere on the internet.
- Cloud Service provider (Microsoft, Amazon, Google..) provides infrastructure, Multi - Tenancy

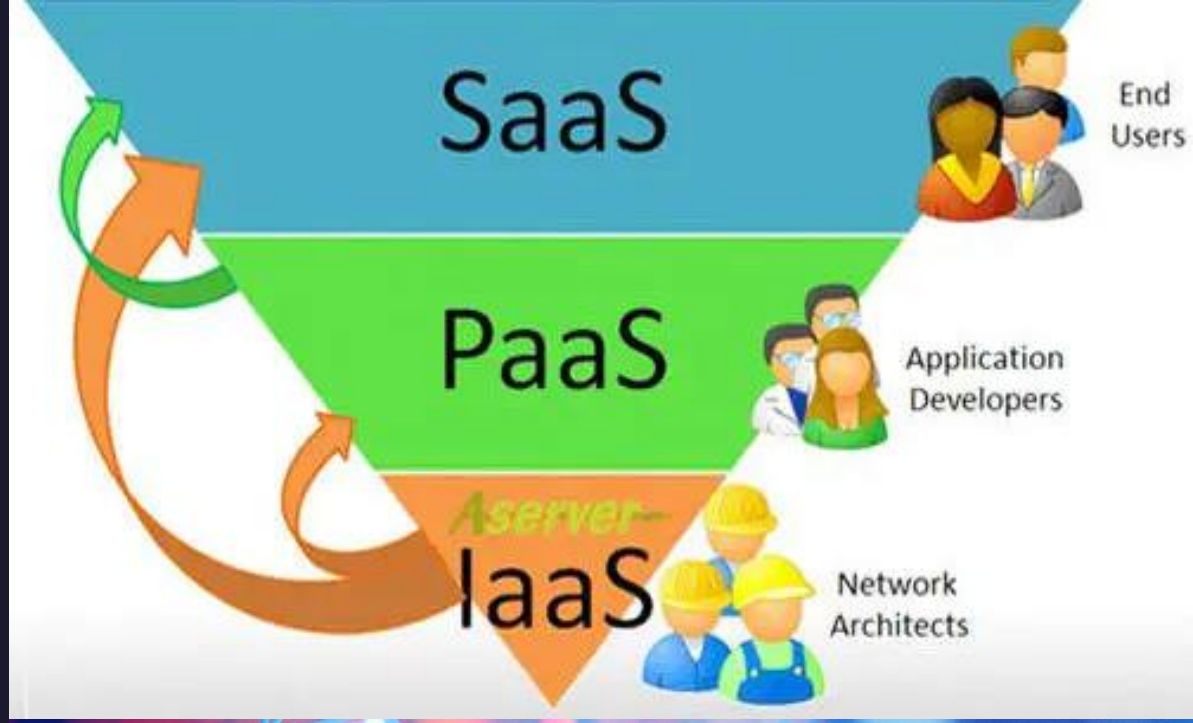


HYBRID CLOUD

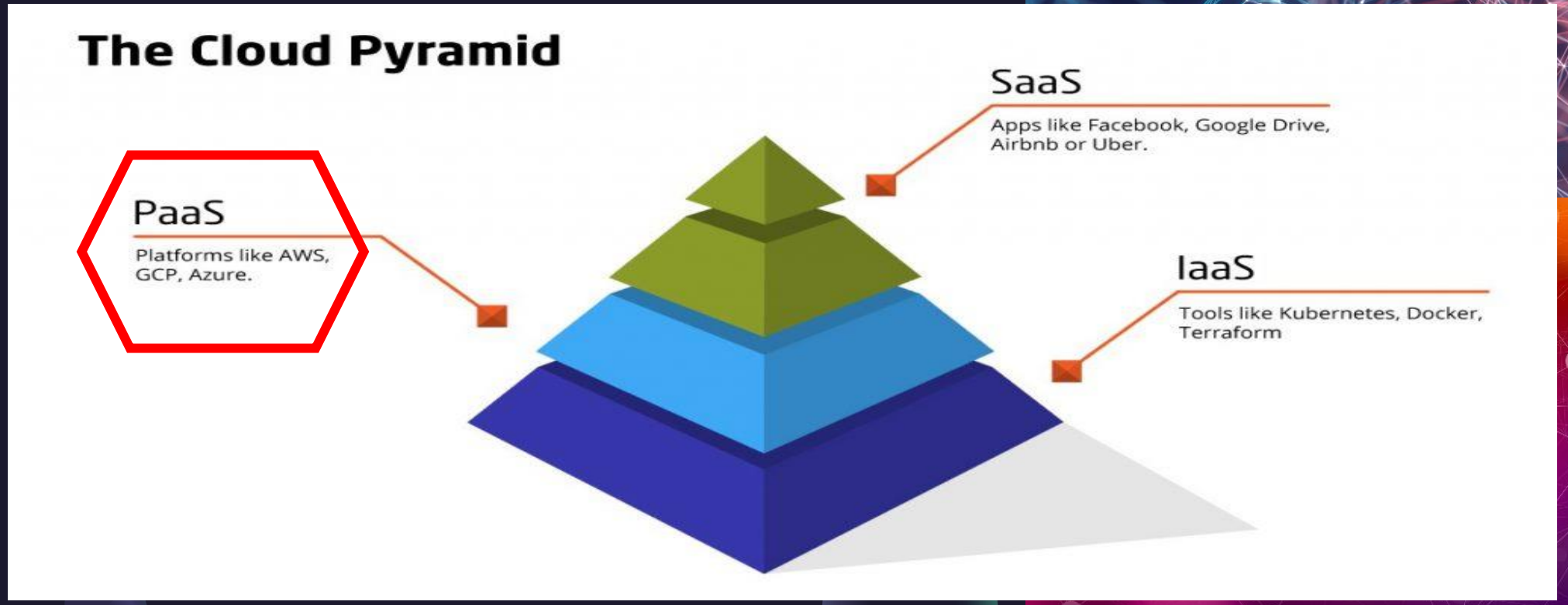
- Inside corporate network or anywhere on the internet.
- Private, your organization provides the hardware, and the cloud service provider provides for the public cloud.
- Single tenancy + Multi Tenancy.



Cloud Service Types



Which one are we focusing....



PaaS

You Manage,

Application & Data

CSP (Cloud Service Provider) Manages,

Runtime

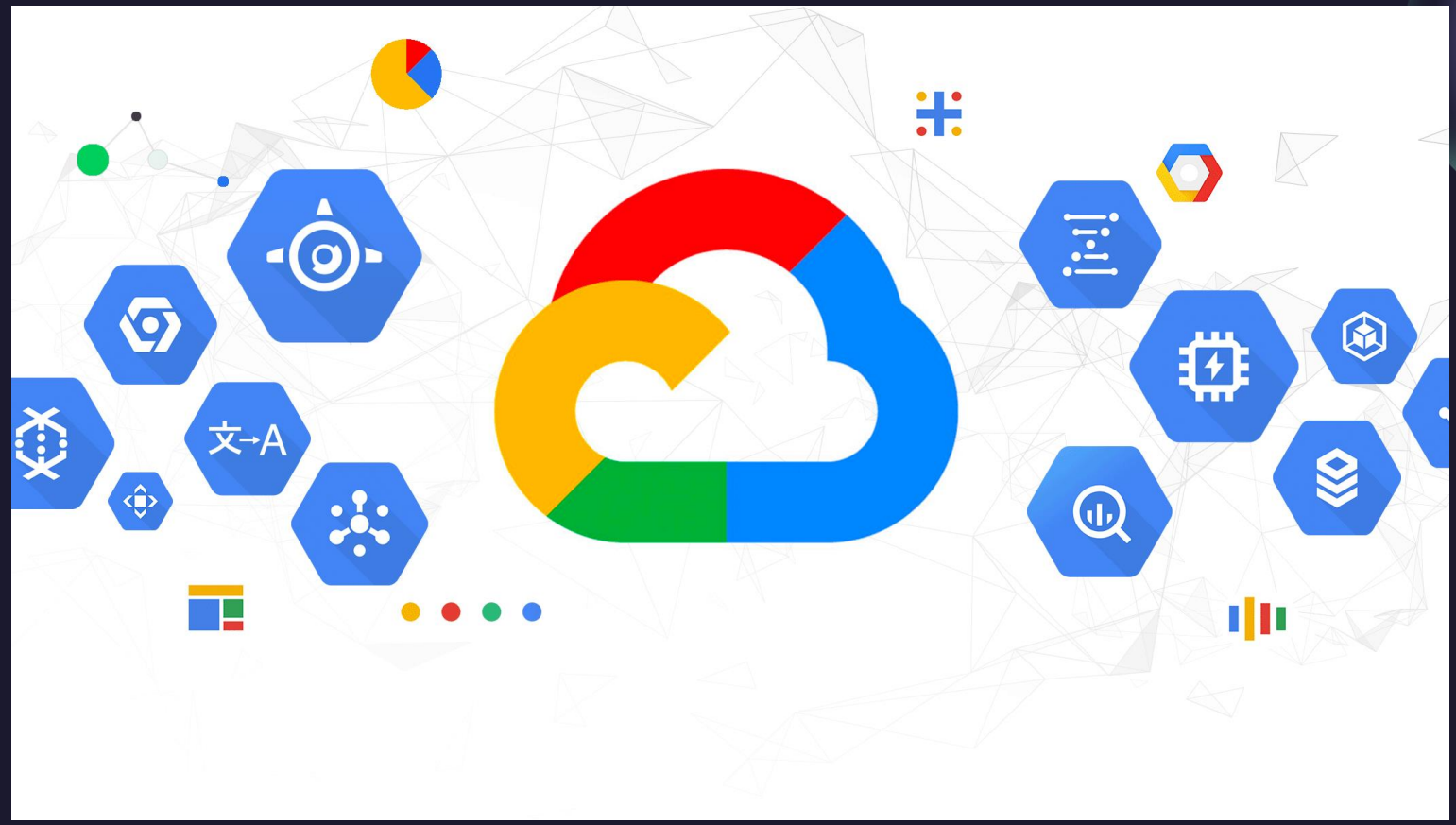
Middleware

OS

Visualization

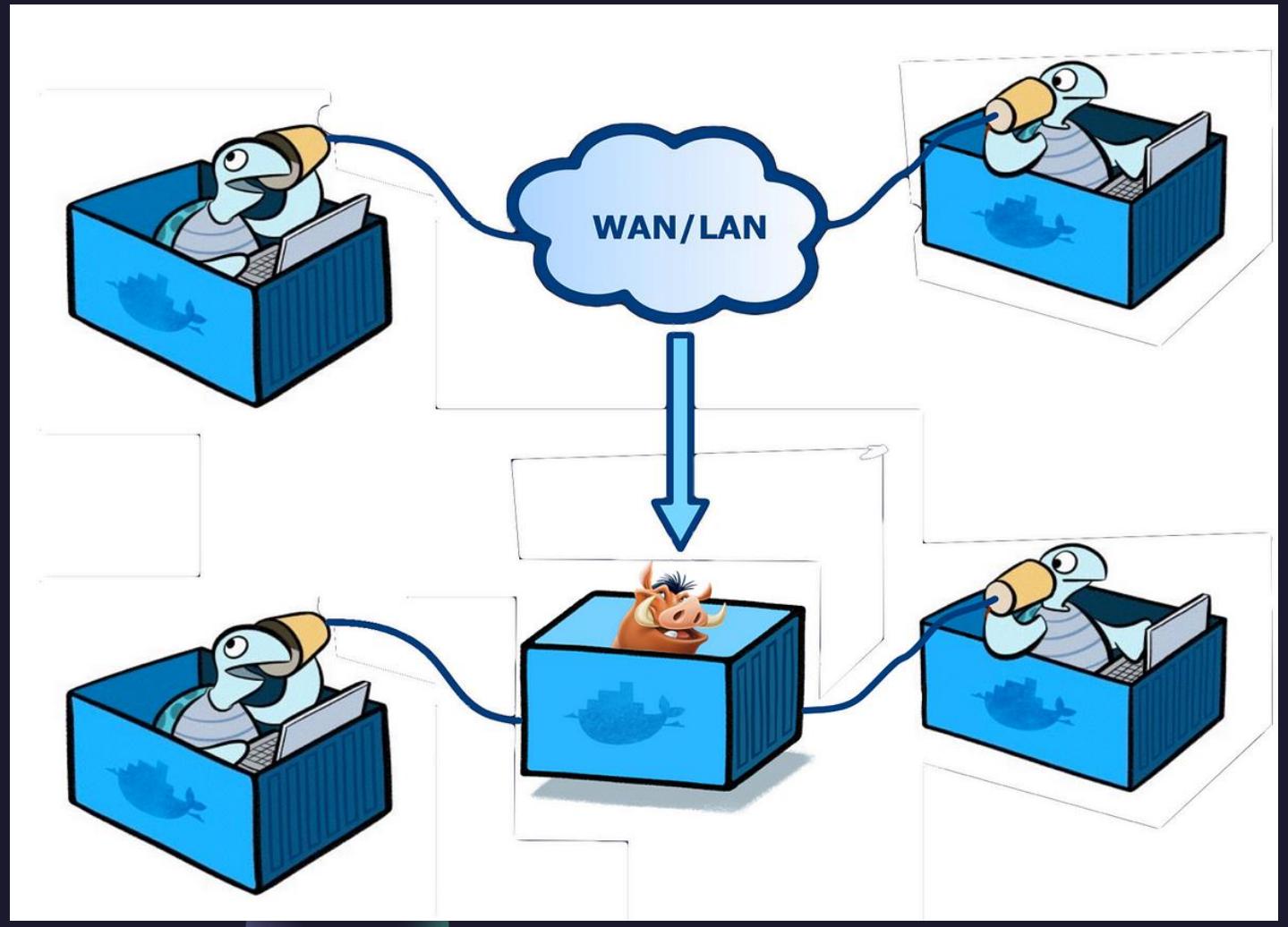
Servers

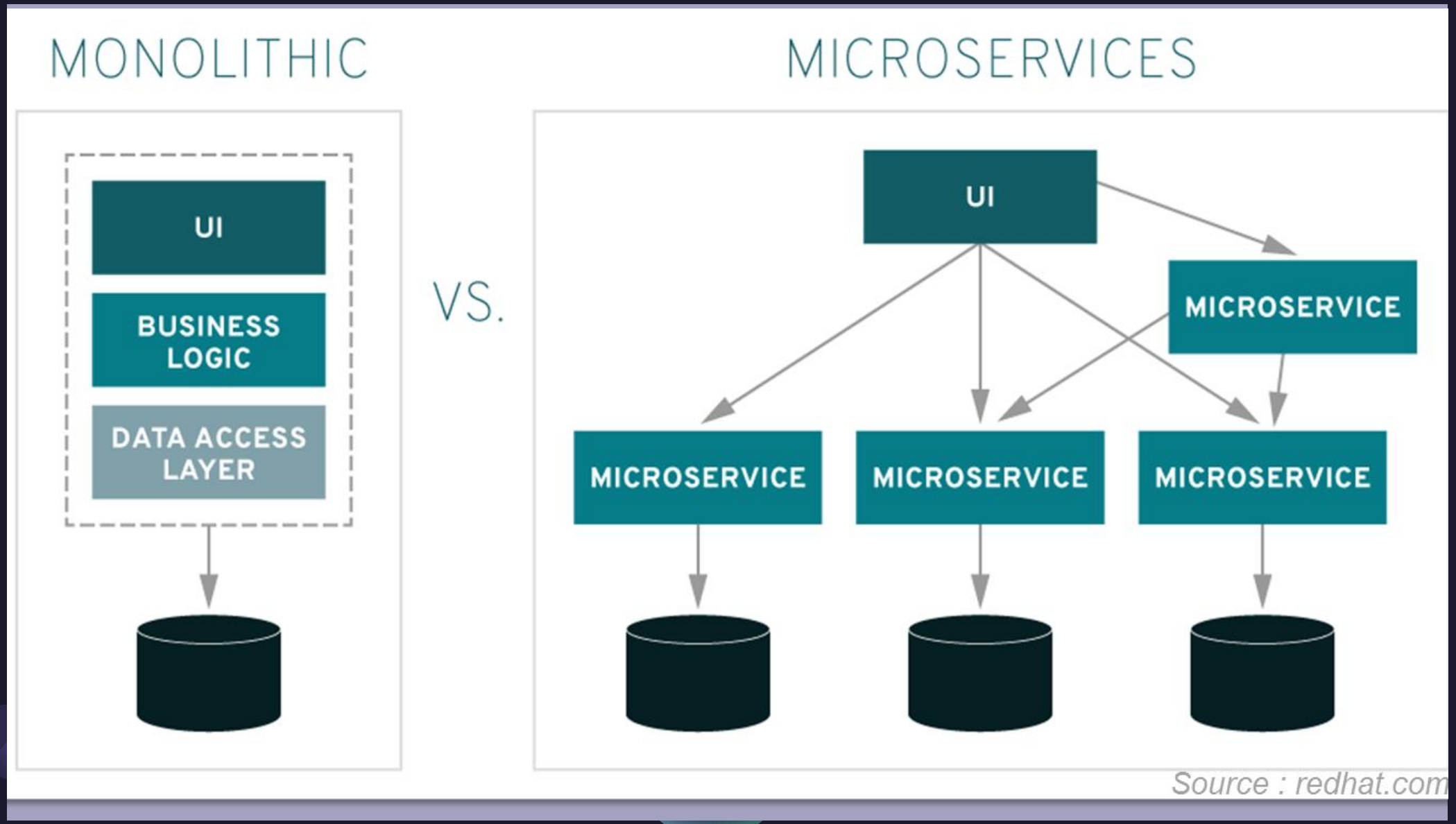
Storage, Networking…

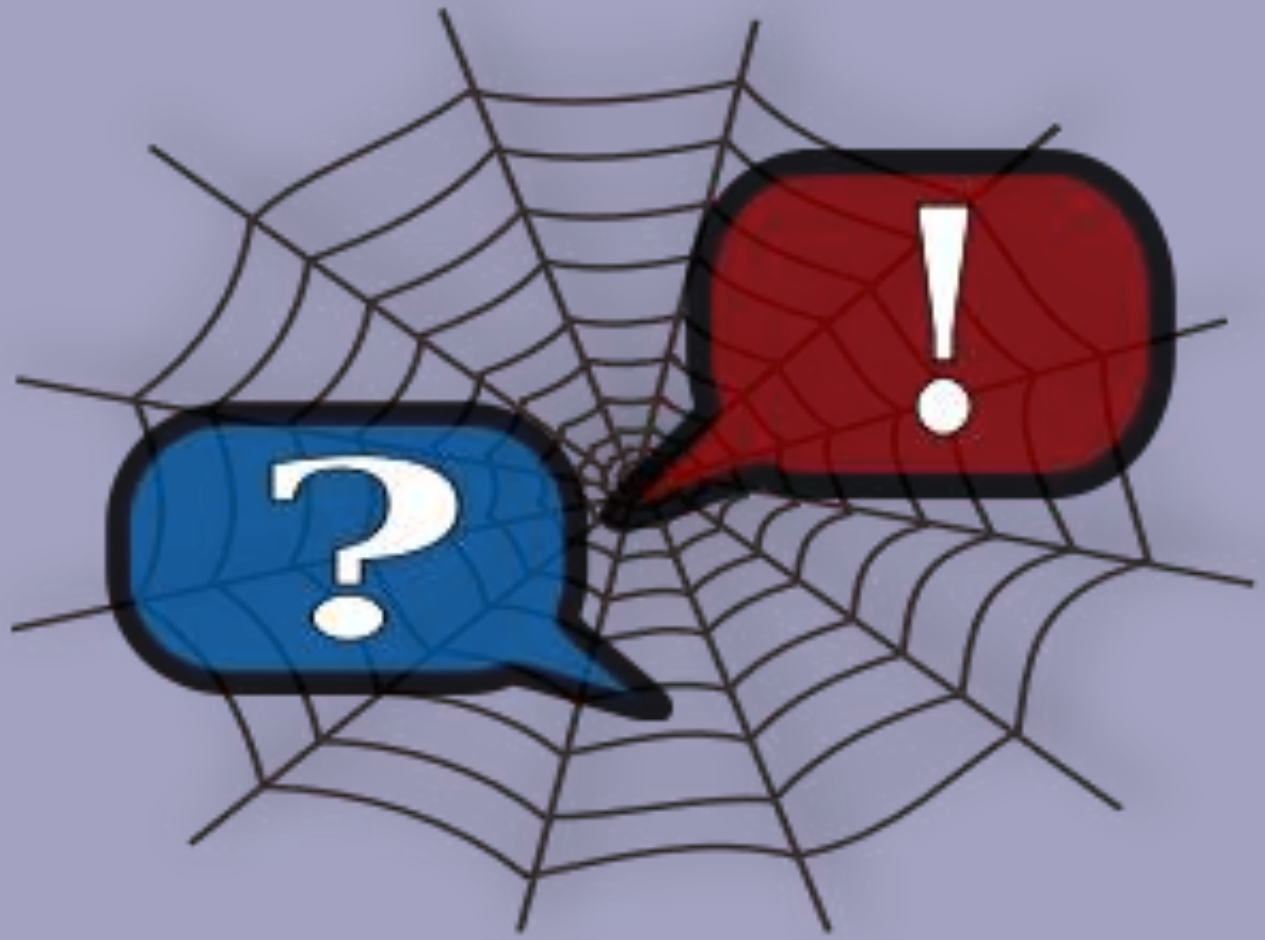


What form are we having our services..??

Microservices….




MONOLITHIC
MICROSERVICES
UI
BUSINESS LOGIC
DATA ACCESS LAYER
VS.
UI
MICROSERVICE
MICROSERVICE
MICROSERVICE
MICROSERVICE
Source : redhat.com



Current Trends in Software Engineering (CTSE

Isanka Rathnayaka

Manager – Software Development .NET | Full Stack | Azure

SYDPRO Pvt. Ltd

BSc in IT – Curtin University of Technology, Bentley Western Australia

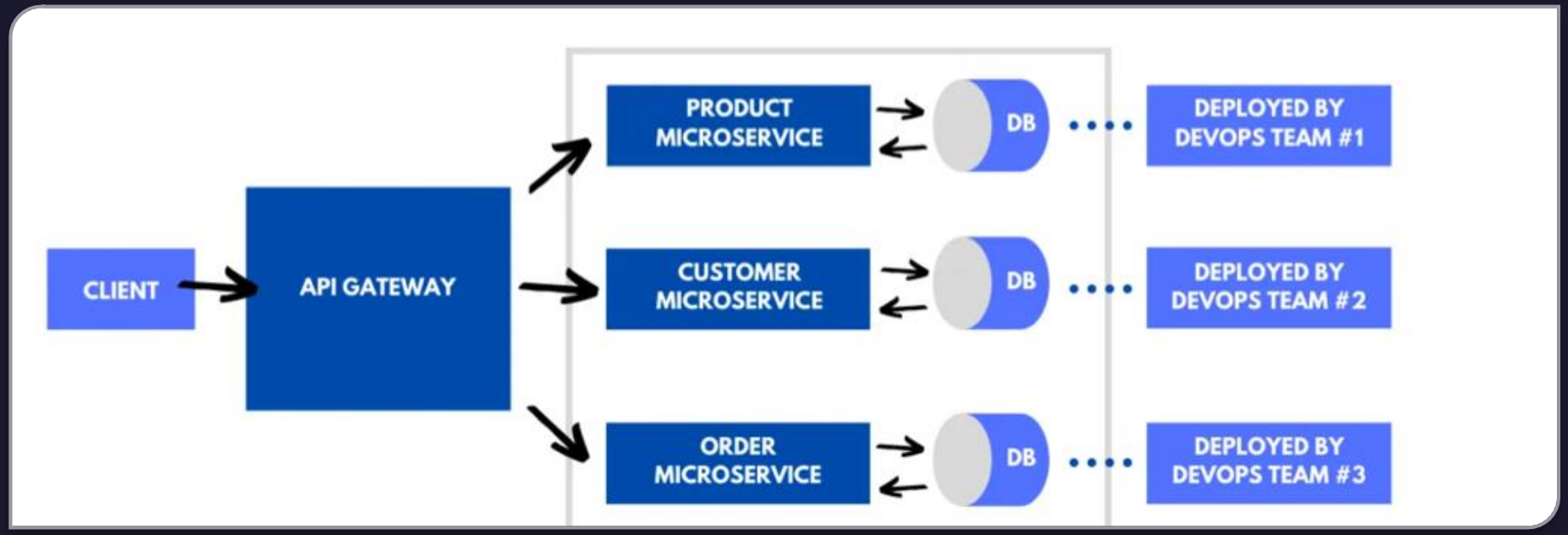
ishikim86@gmail.com

Agenda

- Microservices vs Monolithic Architecture
- Challenges of building microservice architecture
- Five principles to enable your microservice
- Docker to the rescue of microservices
- What is Docker and Docker in Practice ?
- Docker images vs Docker containers
- Advantages & Disadvantages of Docker
- What is Kubernetes ?
- How do Docker and Kubernetes relate to Microservice Architecture
- Q & A

Microservice architecture in ASP.NET CORE ... overview

- Monolithic Architecture - Basics



Microservice Architecture

Microservice vs Monolith Architecture

Monolithic	Microservices
One Solution that contains all the Business Components and Logics.	Multiple Services that contain one single
Single Database	Dedicated Database for each microservi
One Language for the Backend	Can have multiple technologies / languag microservice.
Solution has to be deployed to a single Server. Physical Seperation of concerns can be tricky.	Each of the Microservive can be deploye web.
Services will be tightly coupled to the application itself.	Losely Coupled Architecture.

Challenges of building microservice architecture...

Service tracking - *Services distributed across multiple hosts can be hard to track. Rather than a single stop to tweak monolithic integrations, collaborating microservices scattered throughout your environment need to be inventoried and quickly accessible.*

Rapid resource scaling - *Each microservice consumes far fewer resources than monolithic applications but remember that the number of microservices in production will grow rapidly as your architecture scales. Without proper management, many little hosts can consume as much compute power and storage, or more, as a monolithic application.*

Inefficient minimal resourcing - *If you're using the Amazon Web Services environment, there is a bottom limit to the resources you can assign to any task. Microservices may be so small that they require only a portion of a minimal EC2 instance, resulting in wasted resources and costs that exceed the actual resource demand of the microservice.*

Increased deployment complexity - *Microservices stand alone and can be developed in many programming languages. But every language depends on its own libraries and frameworks, so these multiple programming languages will require a completely different set of libraries and frameworks. This increases resource overhead (and costs) and makes deployment a complex consideration.*

Five principles to enable your architecture

1

Designing an efficient microservice architecture is no accident. Libkage to old Sumo Logic's own Mike Mackrory outlines five principles for staying in control of a complex environment powered by microservices:

2

Cultivate a solid foundation. Everything starts with people, so make sure yours are ready to live and breathe in a microservices world.

3

Begin with the API. Simple math: one microservice starts with one API.

4

Ensure separation of concerns. Each microservice must have a single, defined purpose. If it starts feeling like they should add a responsibility, add a new microservice (and a new API) instead.

5

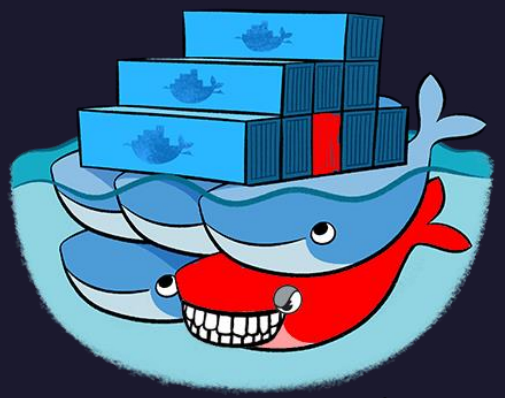
Production approval through testing. Write comprehensive testing parameters for each microservice, then combine them into a full testing suite for use in your continuous delivery pipeline.

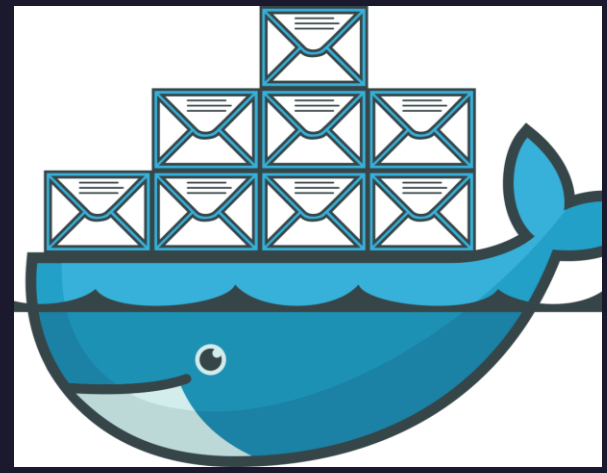
6

Automate deployment. And everything else. Automate code analysis, container security scans, pass/fail testing, and every other possible process in your microservice environment.

DOCKER TO THE RESCUE OF MICROSERVICES....

- The Docker technology of the container, now emulated by other container services, helps address the biggest challenges to building a microservice architecture in the following ways.





- **Task isolation** - Create a Docker container for each individual microservice. This solves the problem of resource bloat from over-provisioned instances idling under the almost non-existent strain of a lone service, and multiple containers can be run per instance.

- **Support multiple coding languages** - Divvy all the services required to run a language, including libraries and framework information, into linked containers to simplify and manage multiple platforms.

- **Database separation** - Use containers to host one or more data volumes, then reference them from other microservices and containers. Chris Evans at Computer Weekly explains the concept:

"The benefit of this method of access is that it abstracts the location of the original data, making the data container a logical mount point. It also allows 'application' containers accessing the data container volumes to be created and destroyed, while keeping the data persistent in a dedicated container."

What is docker and how does docker help ?

"Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, data center VMs, and any cloud."

- a container manager
  - lightweight virtualisation
    (host and guest systems share the same kernel)
  - based on linux namespaces and cgroups

- massively copy-on-write
  - immutable images
  - instant deployment
  - suitable for micro-services (one process, one container)

→ immutable architecture

- a build system
  - images may be built from sources
  - using a simple DSL (`Dockerfile`)
- a set of REST APIs
  - Engine API (control the docker engine)
  - Plugin API (extend the engine $\rightarrow$ network, storage, authorisation)
  - Registry API (publish/download images)
  - Swarm API (manage a clusted of docker machines)

In practice

A docker image is an immutable snapshot of the filesystem

A docker container is

- a temporary file system
  - layered over an immutable fs (docker image)
  - fully writable (copy-on-write[1])
  - dropped at container's end of life (unless a `commit` is made)
- a network stack
  - with its own private address *(by defaut in 172.17.x.x)*
- a process group
  - one main process launched inside the container
  - all sub-processes SIGKILLed when the main process exits

Docker images & docker containers

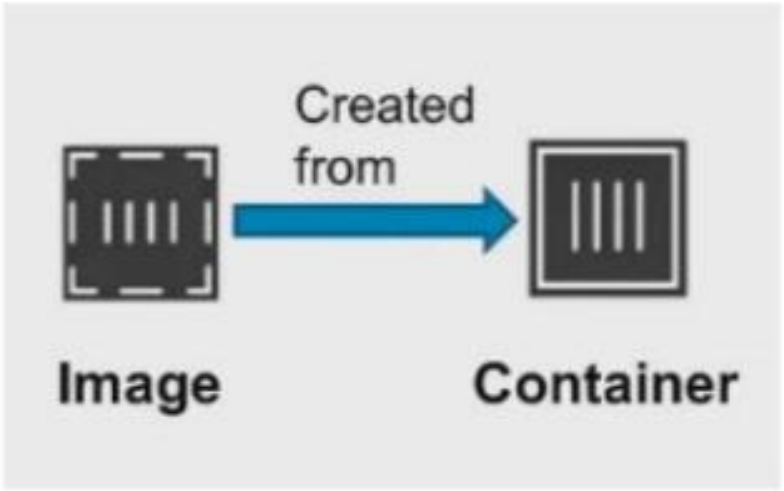
Docker Images

- Read only template used to create containers
- Stored in the Docker Hub or in your local registry
- Image is a Read Only Template and is use to create container
- You can't Edit , But u can delete
- 2 Method to create Image (Interactive Method ,Dockerfile Method)

A Docker image is made up of a collection of files that bundle together all the essentials, such as installations, application code and dependencies, required to configure a fully operational container environment.

Docker Containers

- Running State of Image
- It is Like a Virtual Machine
- It Works on Layered File System
- Runnable instance of a docker image
- Isolated application platform
- Contains everything needed to run your application
- Based on one or more images
- Each container has its own Root file system , Processes ,Memory , Devices , Network ports



Advantages & disadvantages of docker

Docker Advantages

- Rapid Deployment
- No pre-allocation of RAM
- CI Efficiency , Build App only once
- Less Cost and light weight
- It can run on the Physical H/W ,VM
- You can reuse the image
- Less time to create container (VM)
- Version Controlling
- Portability
- Isolation

Docker Disadvantages

- Not a Good solution for Rich GUI
- Difficult to Manage Large Amount (Containers)
- Cross platform compatibility issue
- Only suitable when team OS is same
- No solution For data recovery & Backup

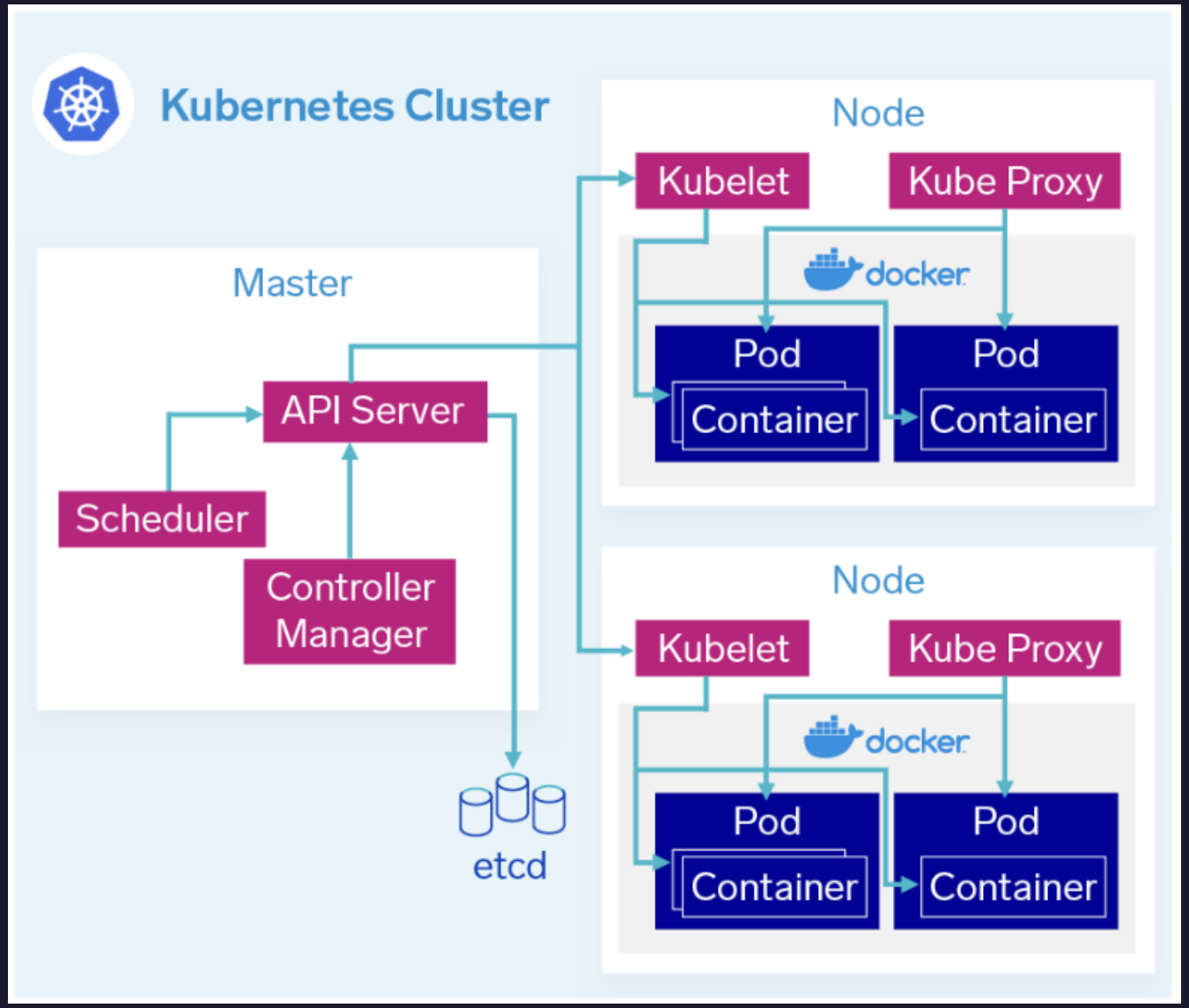
What is Kubernetes ?

Kubernetes is an open-source container management system developed by Google and made available to the public in June 2014.

A container is a virtualized environment that consists of an application and all the configuration files, libraries, binaries and dependencies needed to execute that application.

Kubernetes aims to make deploying and managing complex distributed systems easier for DevOps engineers and developers that want to break up an application monolith into microservices.

A cluster is typically composed of a parent machine (called a node) with multiple child nodes that run the applications in a container.


Kubernetes Cluster
Node
Kubelet
Kube Proxy
docker
Pod
Container
Pod
Container
Master
API Server
Scheduler
Controller Manager
etcd
Node
Kubelet
Kube Proxy
docker
Pod
Container
Pod
Container

How do Docker and Kubernetes relate to Microservice Architecture

- Docker and Kubernetes are two important technologies in the microservices architecture stack. Docker provides containerization, while Kubernetes provides orchestration, making it easier to deploy, manage, and scale microservices.

Q & A

Current Trends in Software Engineering (CTSE)

Isanka Rathnayaka

Manager – Software Development .NET | Full stack | Azure

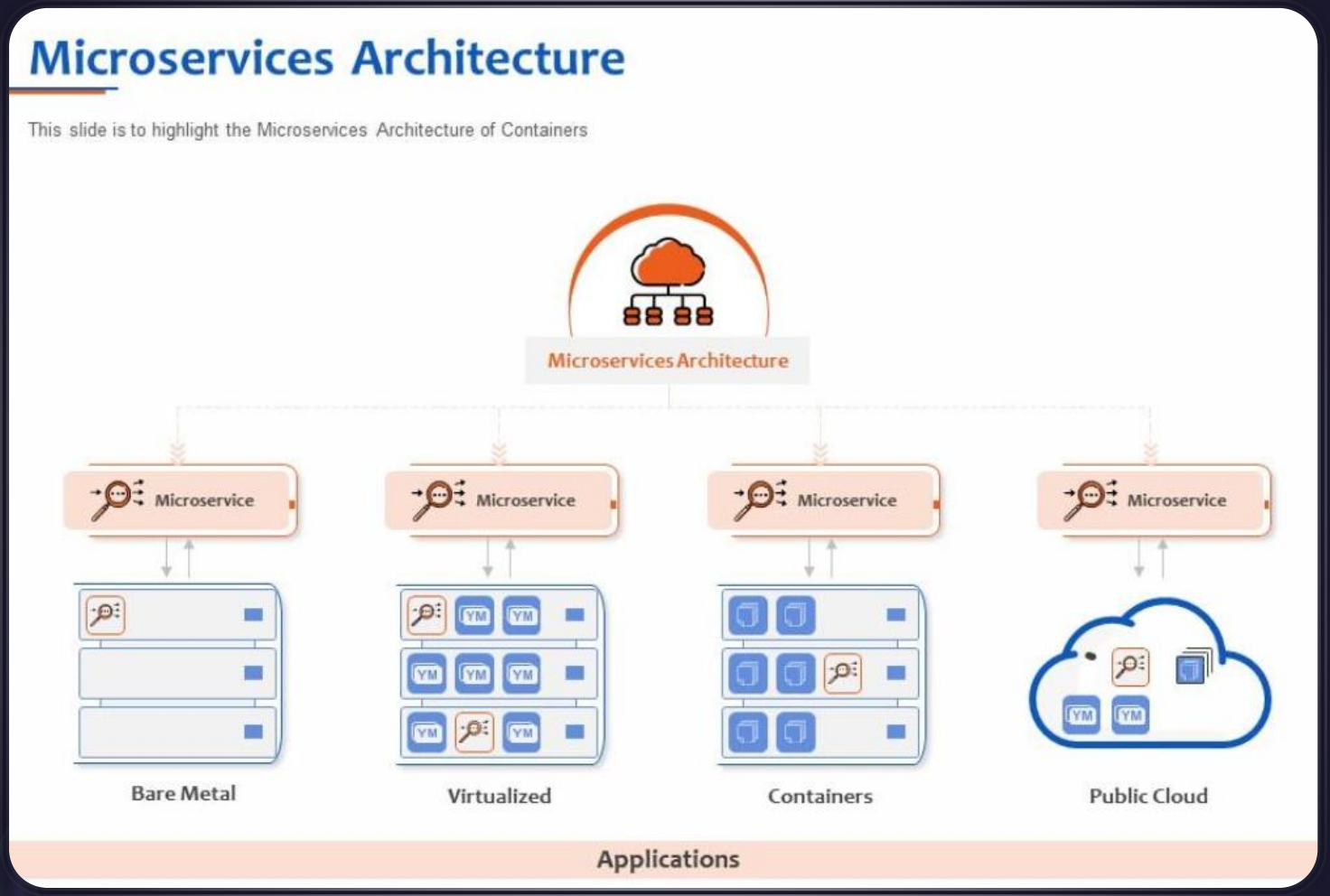
SYDPRO PVT. LTD

BSC. It – CURTIN UNIVERSITY OF TECHNOLOGY , BENTLEY, western Australia

ishikim86@gmail.com

AGENDA

- Are containers the same thing as microservices?
- More into microservice architecture
- Container architecture & Microservice use cases
- Why your organization need Kubernetes?

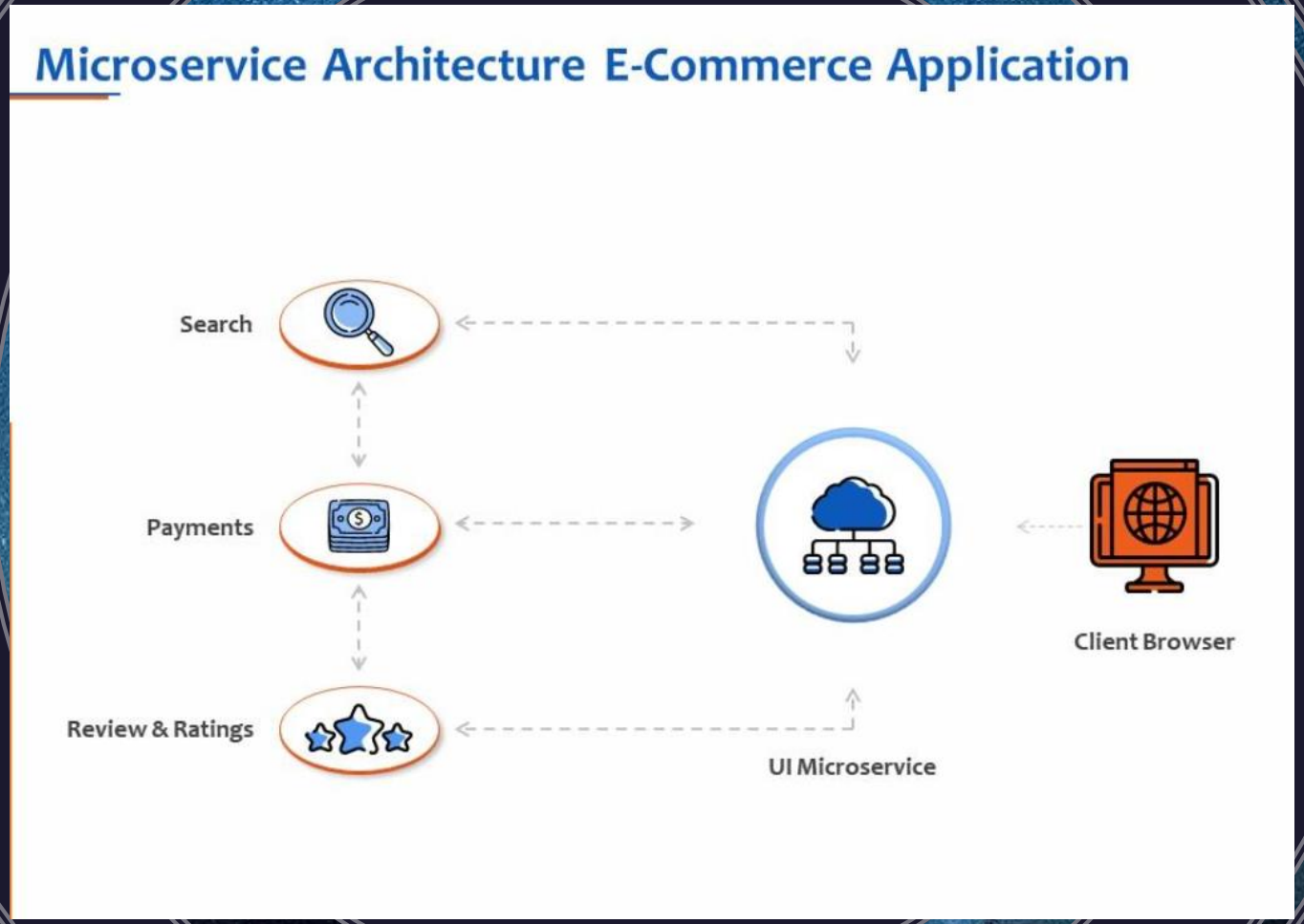


Are containers the same thing as microservices?

- The main difference between microservices and containers is that microservices are an architectural paradigm, while containers are a means to implement that paradigm.

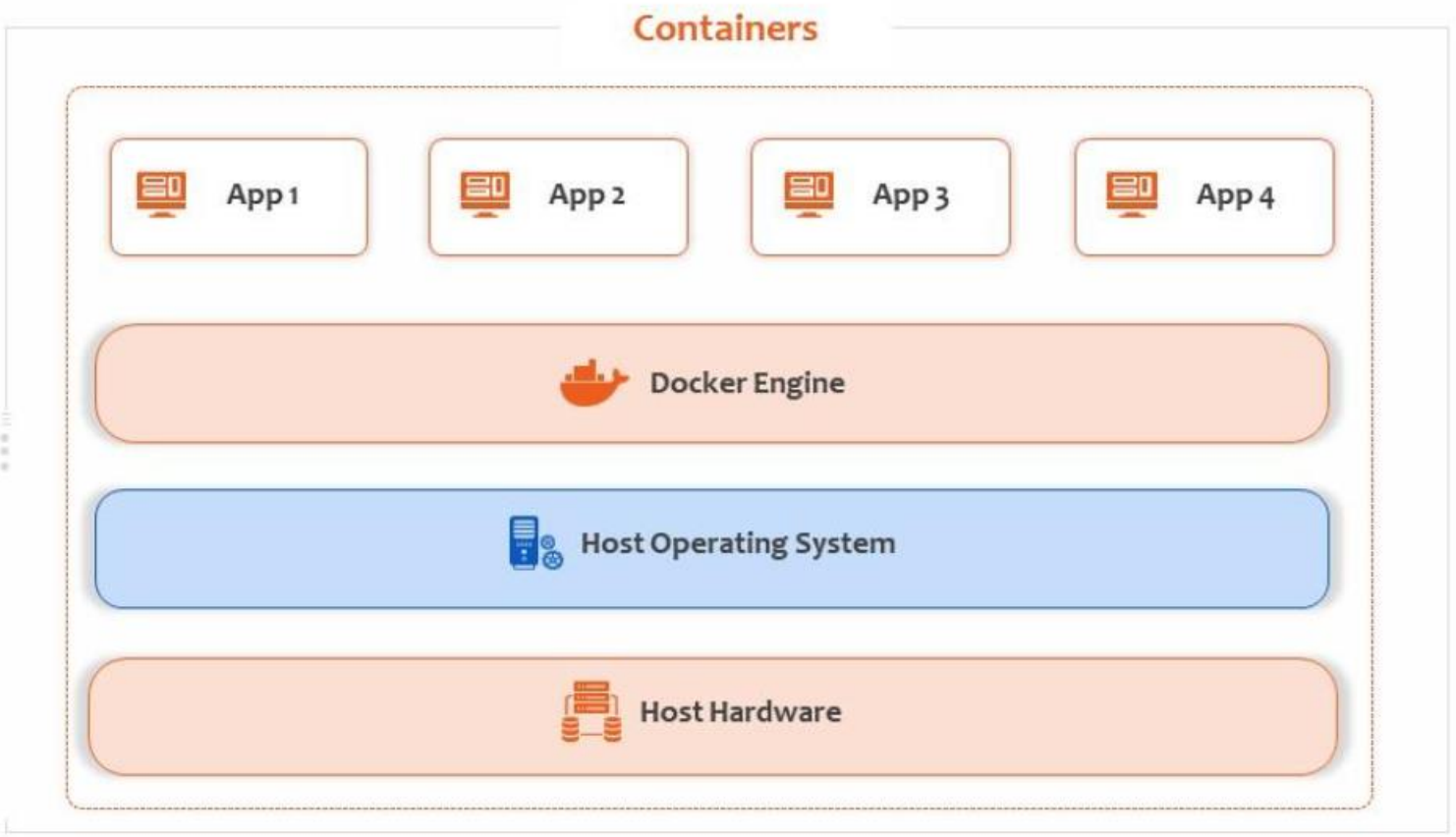
- Containers host the individual microservices that form a microservices application.

Real world

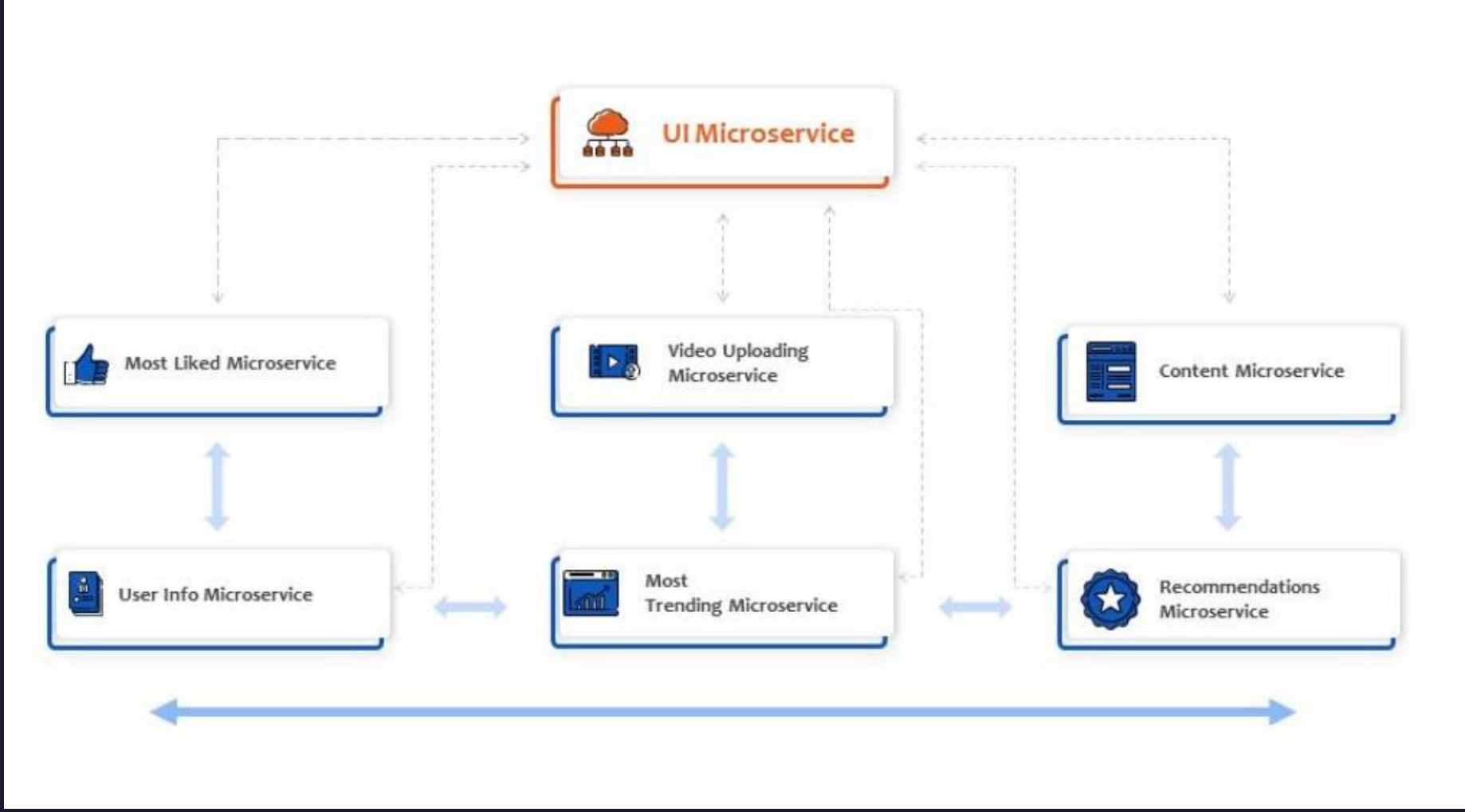

Microservice Architecture E-Commerce Application
Search
Payments
Review & Ratings
Client Browser
UI Microservice

Container Architecture

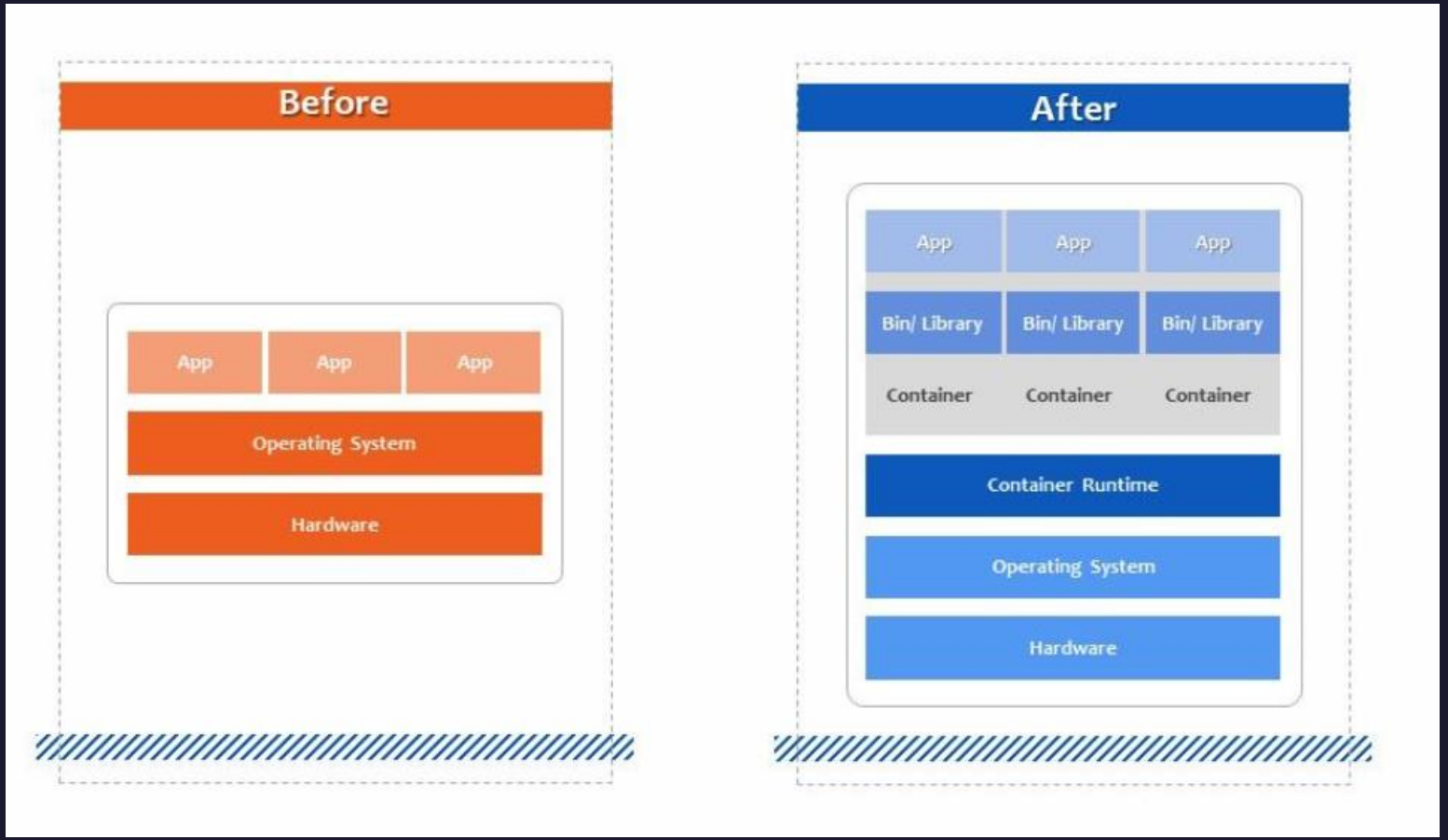
This slide highlights the core Architecture of Containers and Applications hosted to the Docker Engine.



Microservices use cases



Before and after Kubernetes



Why organizations should user Kubernetes?

What can Kubernetes do?

Instrumentality Deployments and Rollout Management

Describe your containers and the way several you wish with a "Deployment." Kubernetes can keep those containers running and handle deploying changes (such as change the image or dynamical atmosphere variables) with a "rollout." you'll pause, resume, and rollback changes as you prefer.

Resource Bin Packing

You'll declare minimum and most figure resources (CPU and memory) for your containers. Kubernetes can slot your containers into wherever ever they match. This will increase your figure potency and ultimately lowers prices.

Inbuilt Service Discovery and Autoscaling

Kubernetes will mechanically expose your containers to the web or alternative containers within the cluster. It mechanically load balances traffic across matching containers. Kubernetes supports service discovery via atmosphere variables and DNS, out of the box. you'll conjointly tack together CPU-based autoscaling for containers for inflated resource utilization.

Heterogeneous Clusters

Kubernetes runs anyplace. you'll build your Kubernetes cluster for a combination of virtual machines (VMs) running the cloud, on-premises, or blank metal in your datacenter. merely opt for the composition in keeping with your needs.

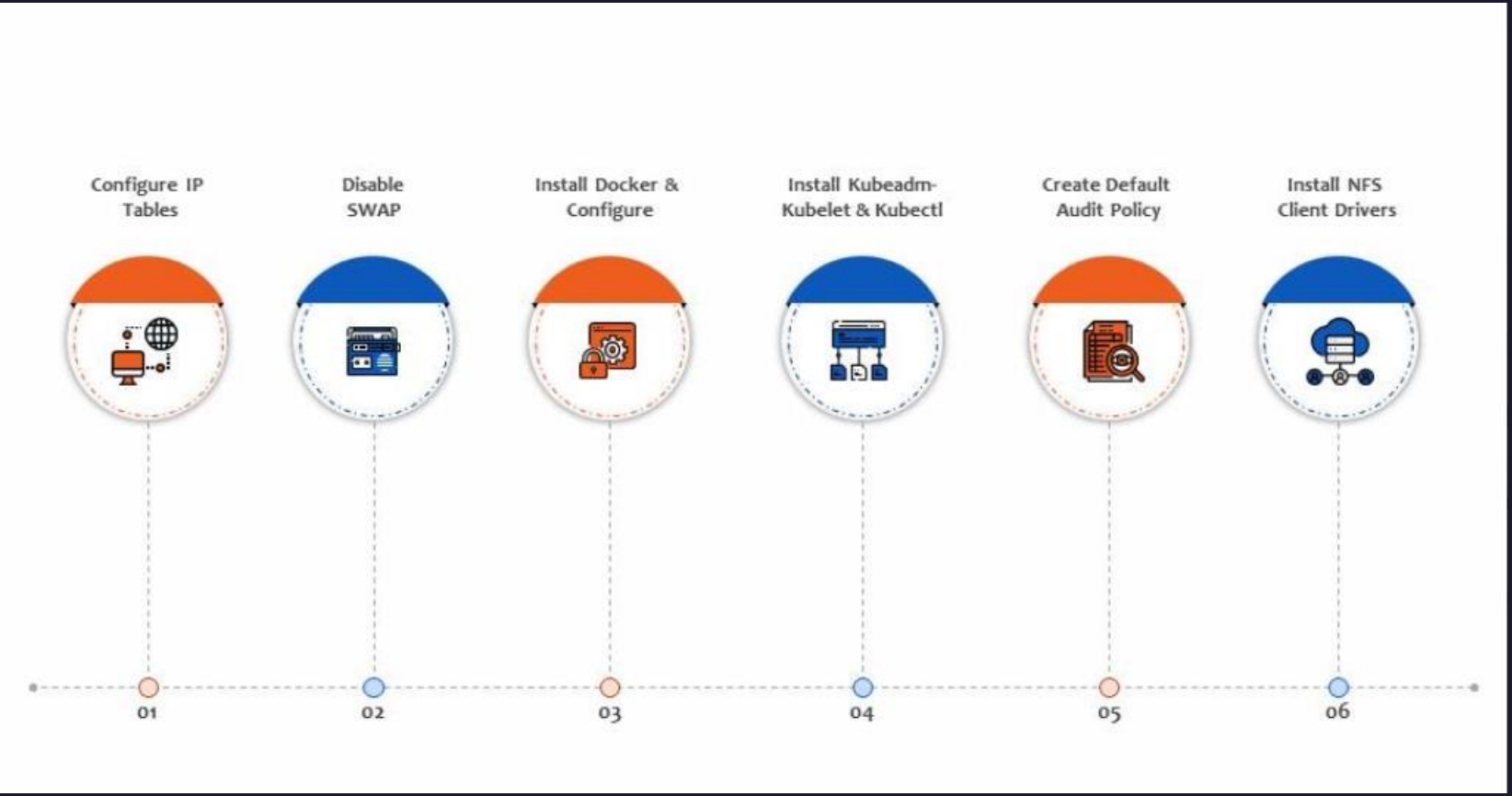
Persistent Storage

Kubernetes includes support for persistent storage connected to homeless application containers. there's support for Amazon net Services east by south, Google Cloud Platform persistent disks, and many, many more.

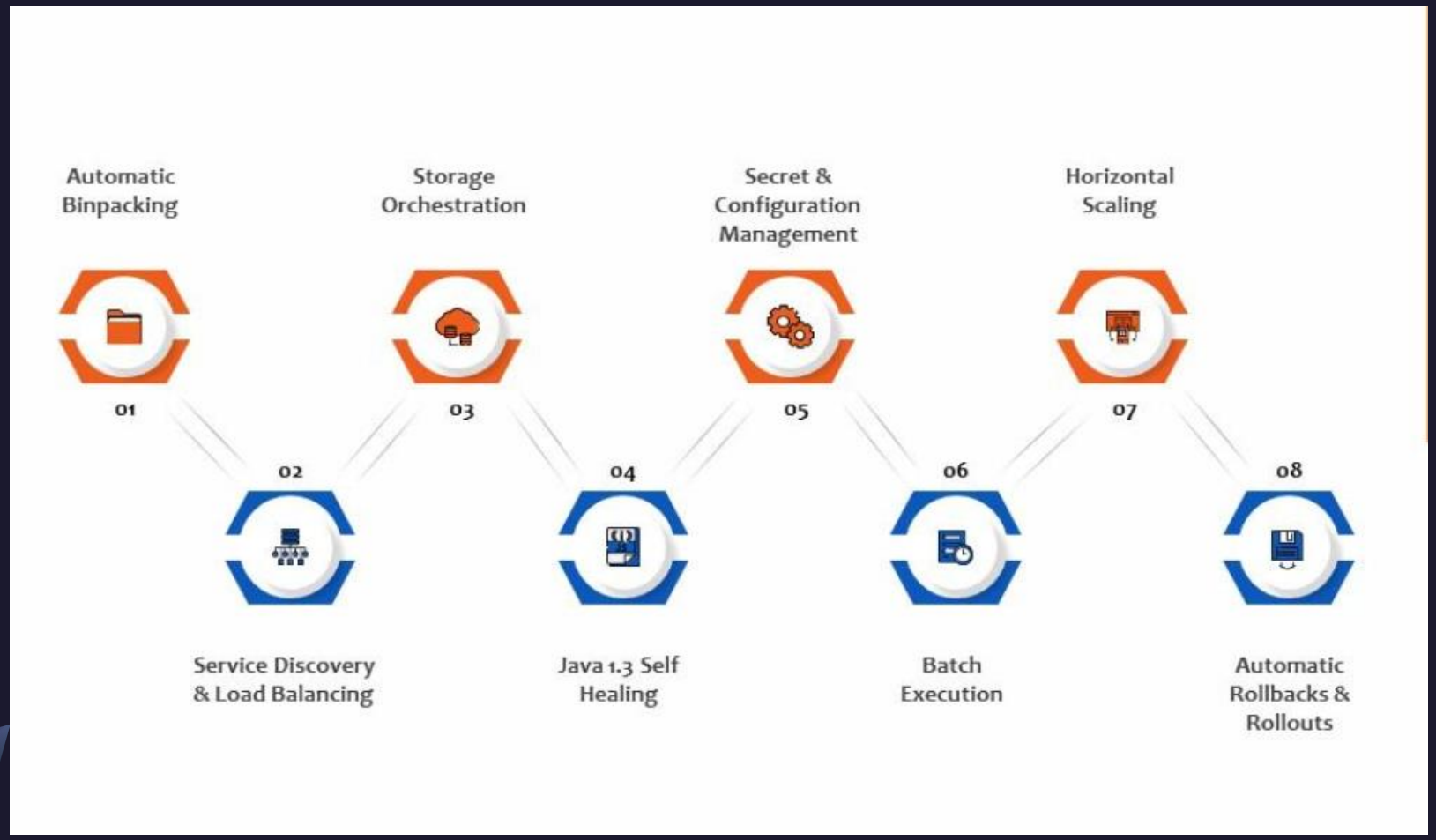
High Convenience Options

Kubernetes is planet scale. this needs special attention to high-availability options like multi-master or cluster federation. Cluster federation permits linking clusters along in order that if one cluster goes down, containers will mechanically move to a different cluster. Grammar Check Re-write Again Next Demo Video (Paraphrasing

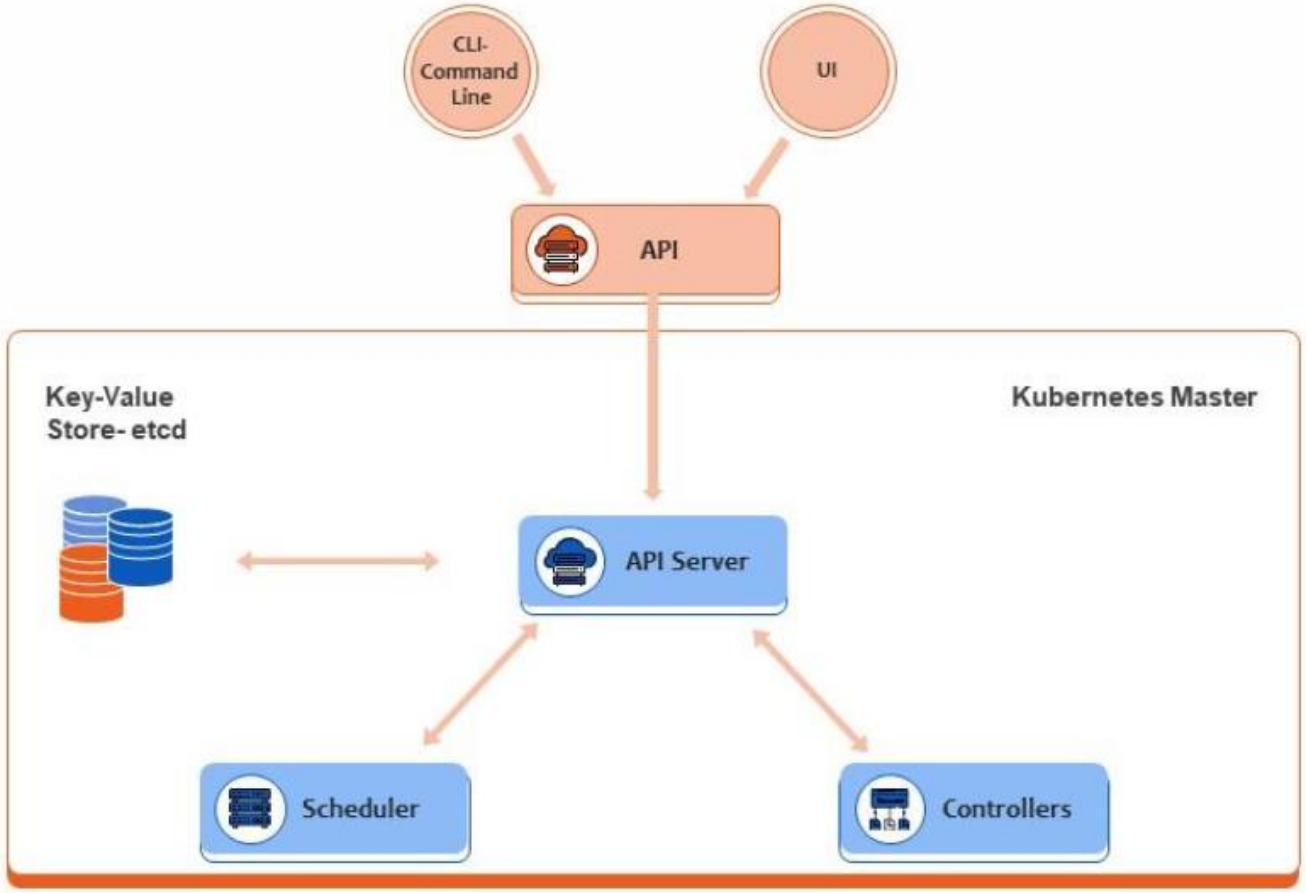
Road map to install Kubernetes in your organization..



Features of Kubernetes



Kubernetes Architecture



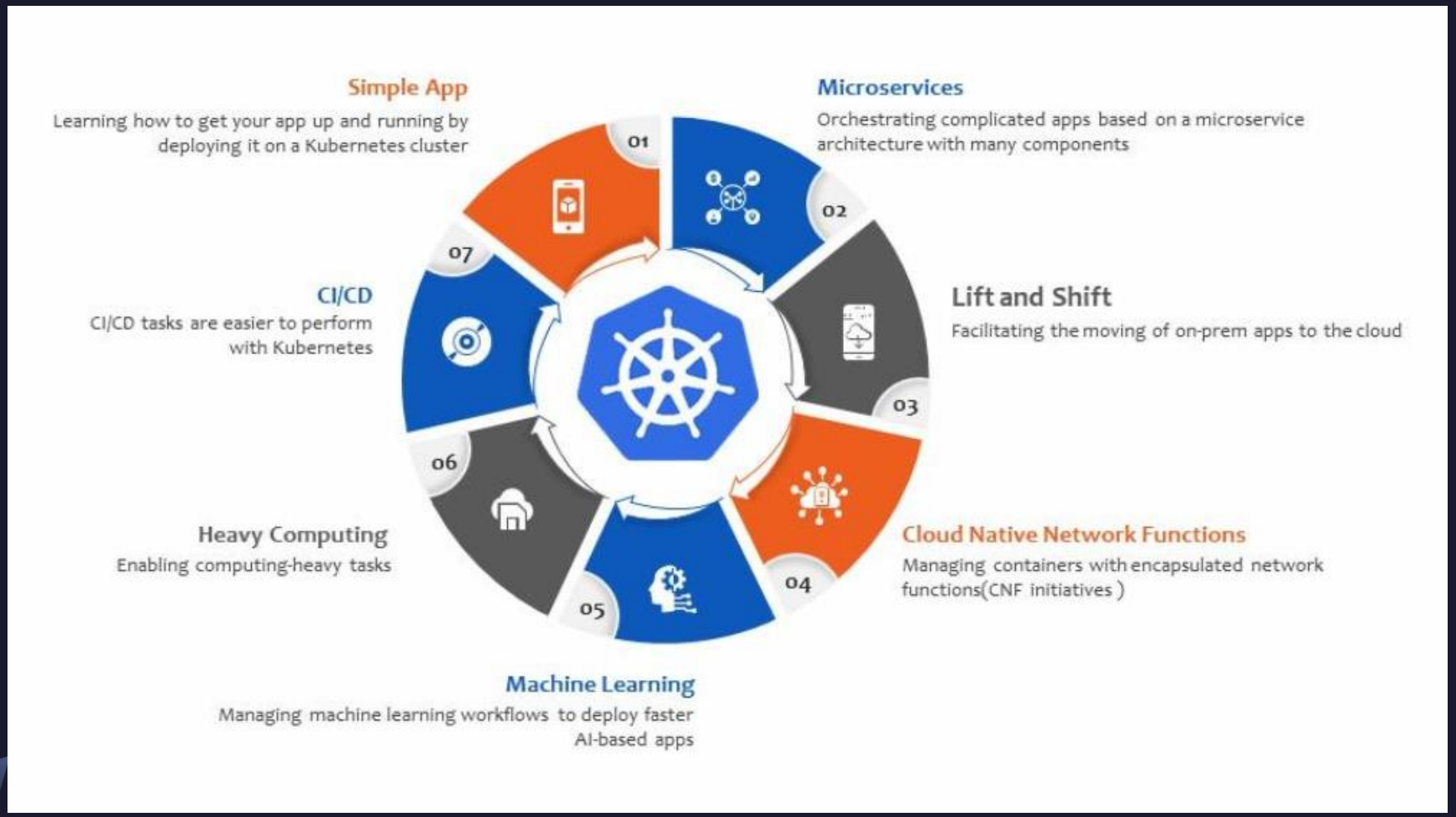
Kubernetes vs Docker Swarm

The slide provides the key difference between Kubernetes and Docker Swarm

Docker Swarm	Vs	Kubernetes
No Auto Scaling	01	Auto Scaling
Good Community	02	Great Active Community
Easy to Start a Cluster	03	Difficult to Start a Cluster
Limited to the Docker Api's Capabilities	04	Can Overcome Constraints of Docker and Docker API
Does not have as Much Experience with Production Deployments at Scale	05	Deployed at Scale more often among Organisations

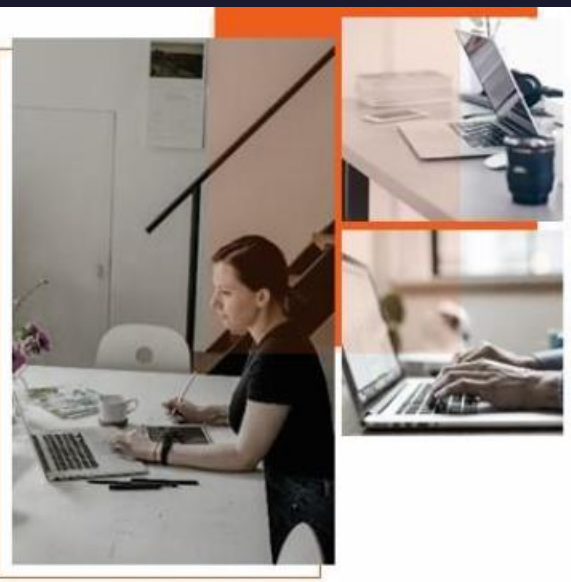
Kubernetes vs docker swarm

Common user cases for Kubernetes



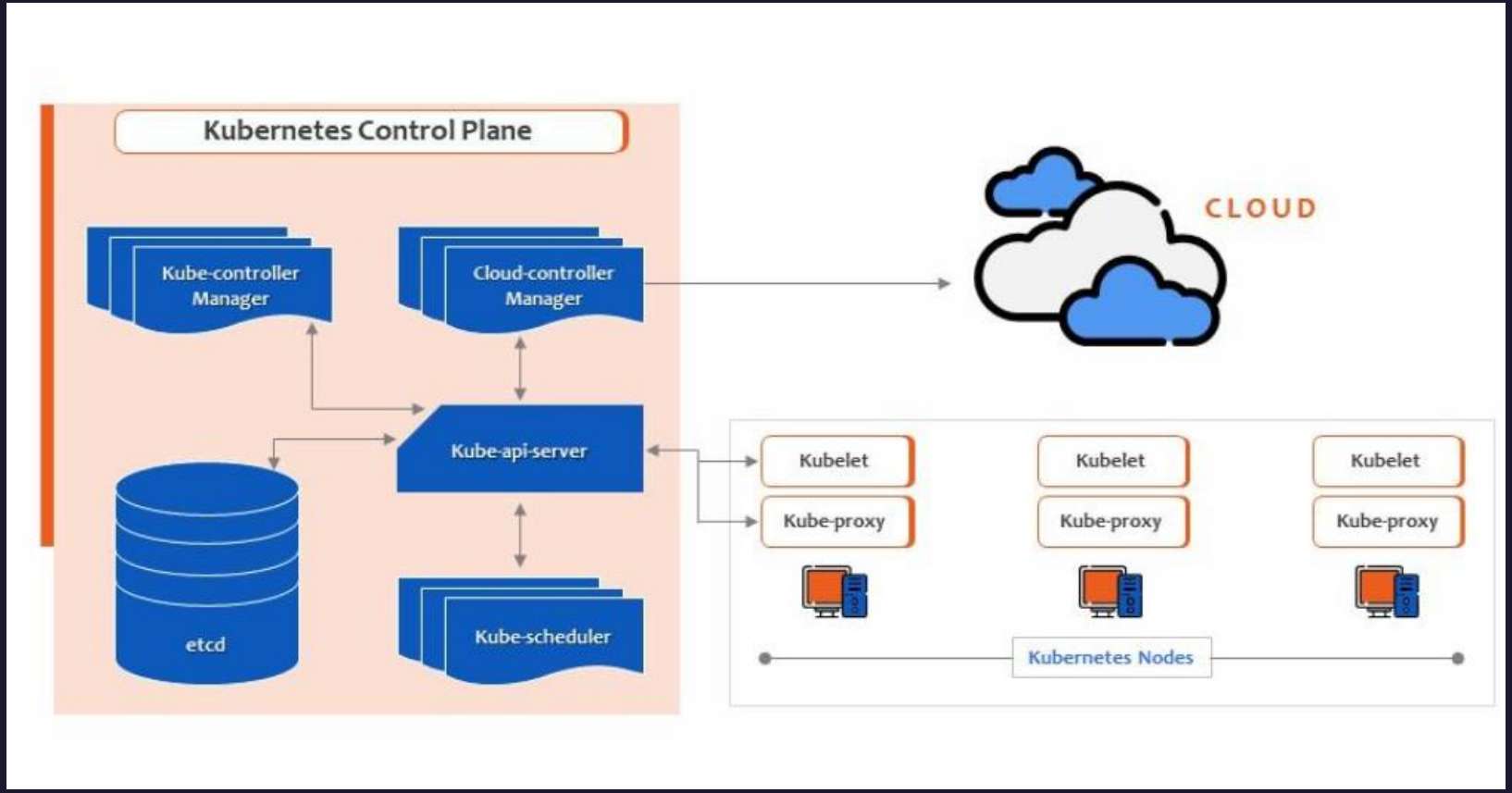
Advantages of Kubernetes

- Using Kubernetes and its Brobdingnagian Scheme will improve your productivity
- Kubernetes and a cloud-native technical school stack attracts talent
- Kubernetes could be a future proof answer
- Kubernetes helps to form your application run additional stable
- Kubernetes is cheaper than its alternatives
- Orchestrate containers on multiple hosts

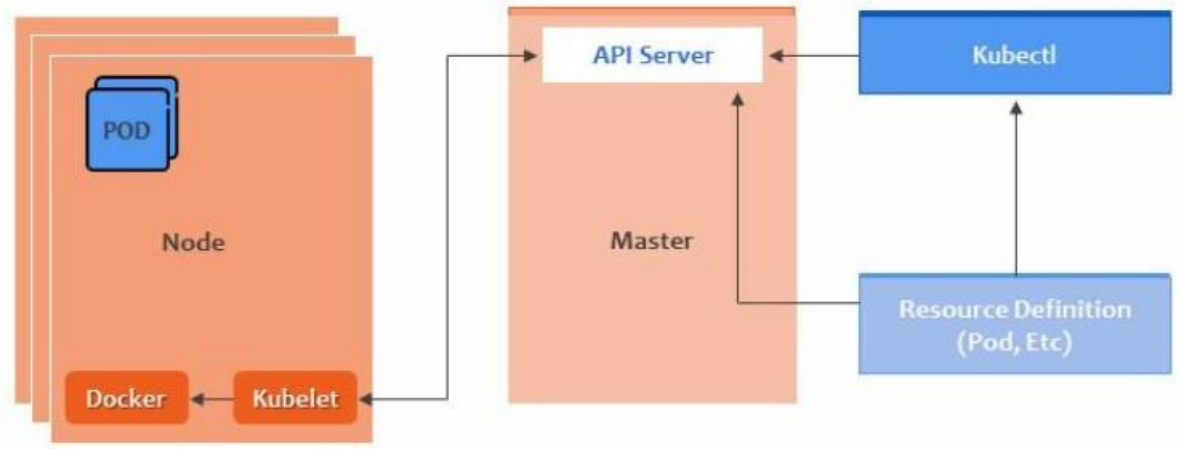


KUBERNETES COMPONENTS ..

Sky view..

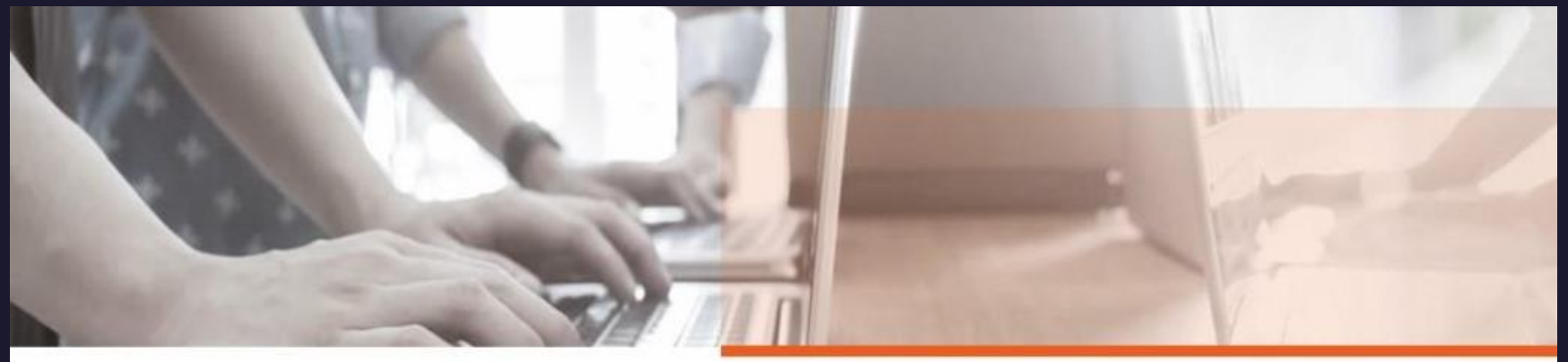

Kubernetes Control Plane
Kube-controller Manager
Cloud-controller Manager
Kube-api-server
etcd
Kube-scheduler
CLOUD
Kubelet
Kube-proxy
Kubelet
Kube-proxy
Kubelet
Kube-proxy
Kubernetes Nodes

What is kubelet?



The kubelet is that the primary "node agent" that runs on every node.

The kubelet works in terms of a PodSpec. A PodSpec may be a YAML or JSON object that describes a pod. The kubelet takes a group of PodSpecs that ar provided through numerous mechanisms (primarily through the apiserver) and ensures that the containers delineate in those PodSpecs ar running and healthy. The kubelet doesn't manage containers that weren't created by Kubernetes



What is Kubectl?

Kubectl is a instruction tool for controlling Kubernetes clusters. Kubectl looks for a file named config within the $HOME/. ... For details regarding every command, toget her with all the supported flags and subcommands, see the kubectl reference documentation.

KUBECTL

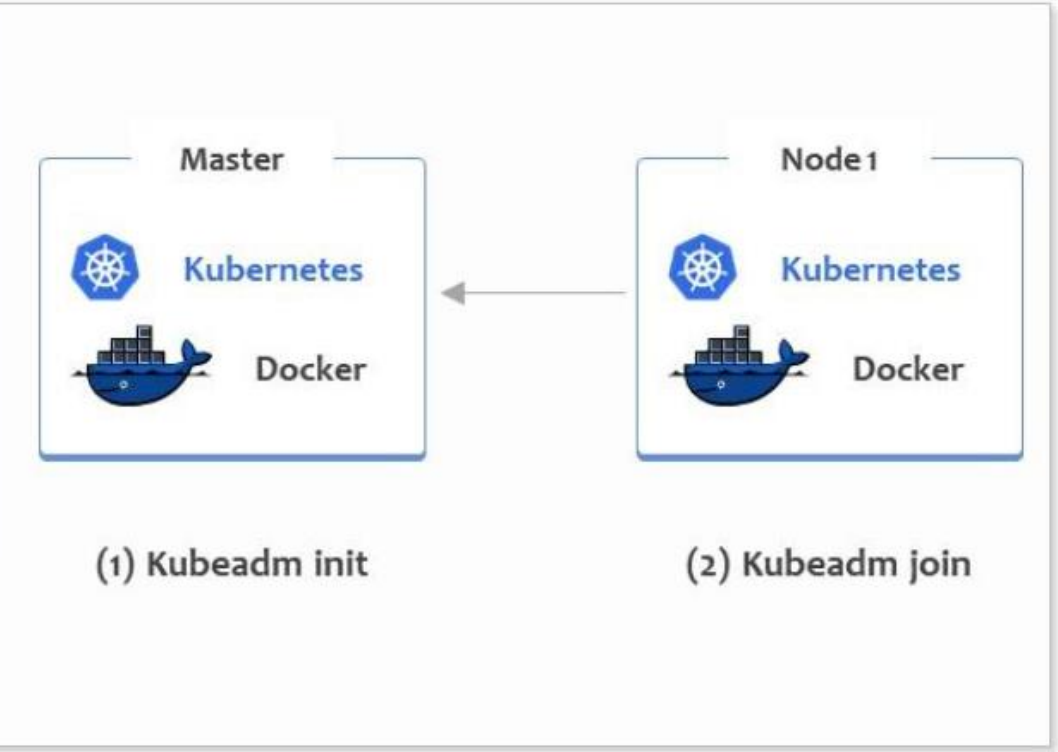
HTTP

KUBERNETES API

Kubernetes

What is Kubeadm?

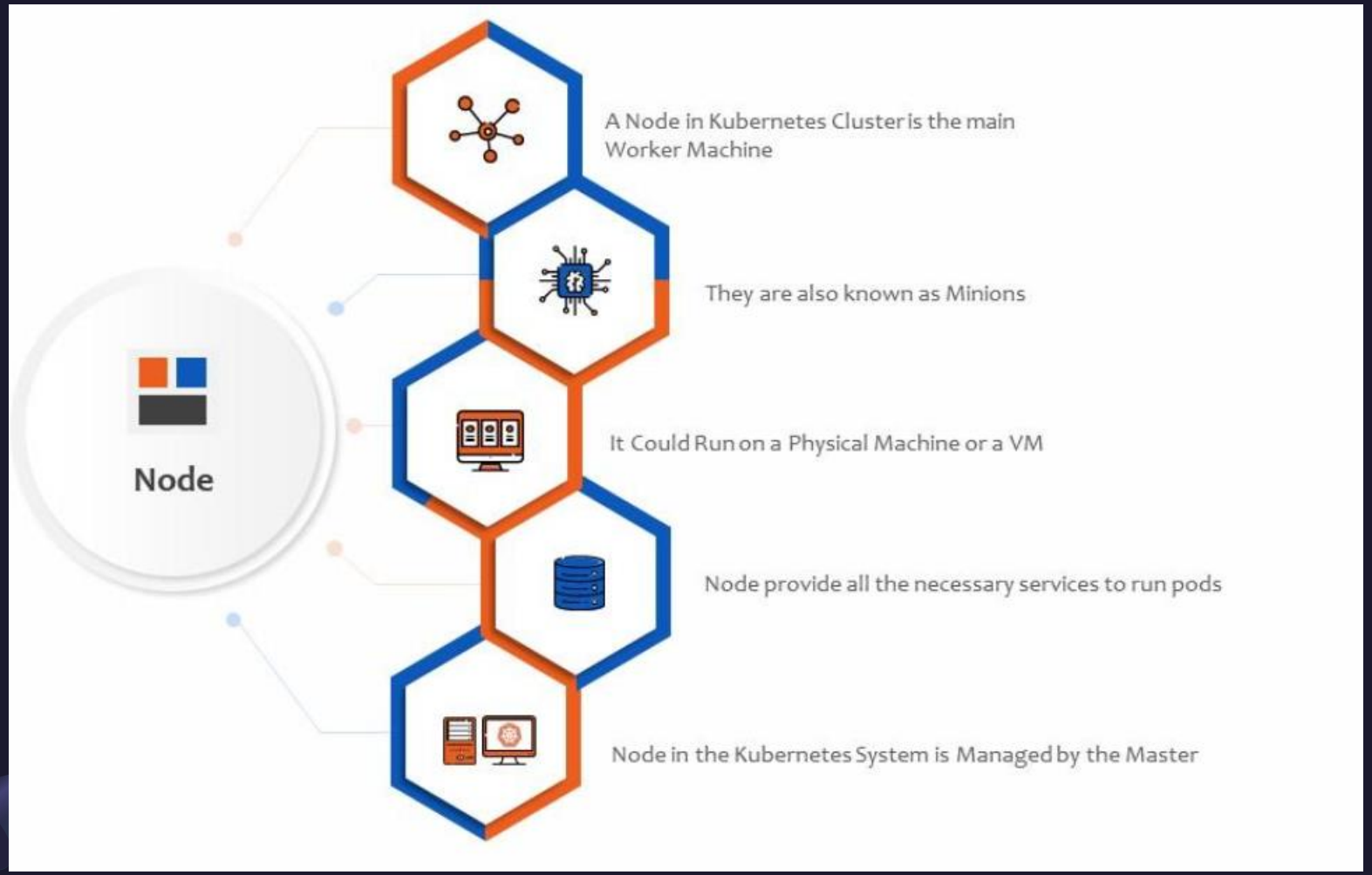
Kubeadm could be a tool designed to produce kubeadm init and kubeadm be part of as best-practice "fast paths" for making Kubernetes clusters. Kubeadm performs the actions necessary to urge a minimum viable cluster up and running.



03 Nodes in Kubernetes

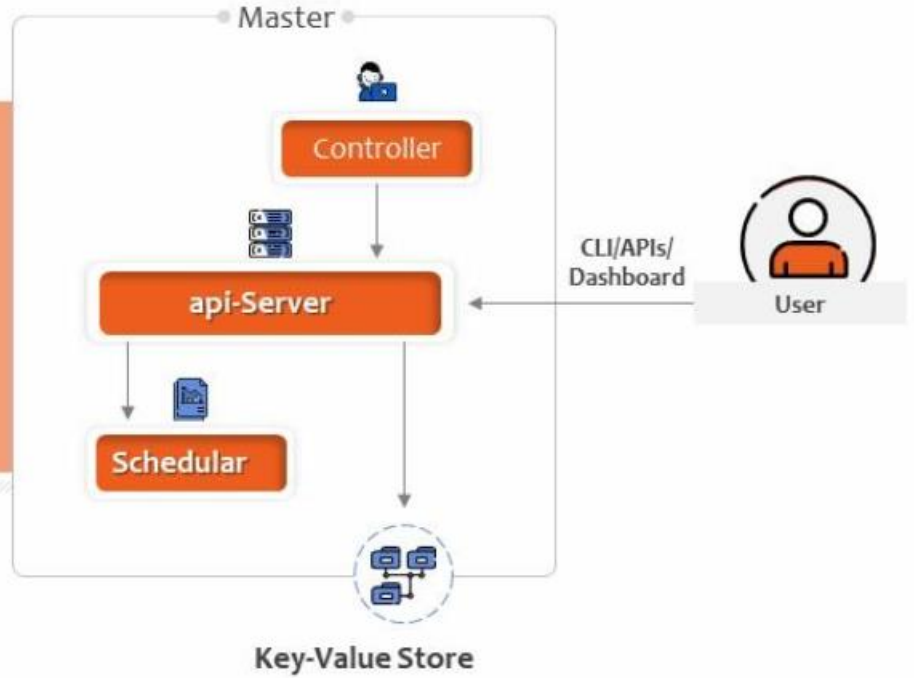
- What is Node in Kubernetes?
- Master Node
- Worker Node

What is NODES IN KUBERNETES ?



MASTER NODES IN KUBERNETES..

A master node
is a node which controls and manages a group of worker nodes (workloads runtime) and resembles a cluster in Kubernetes. ... All external communication to the cluster is via the API-Server. Kube-Controller-Manager, that runs a group of controllers for the running cluster.

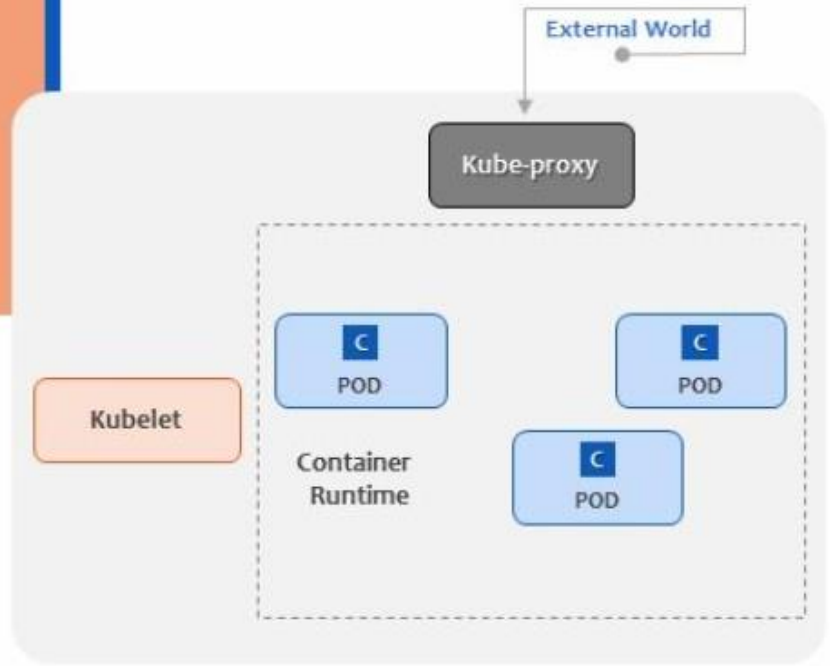


WORKER/ SLAVE NODES...

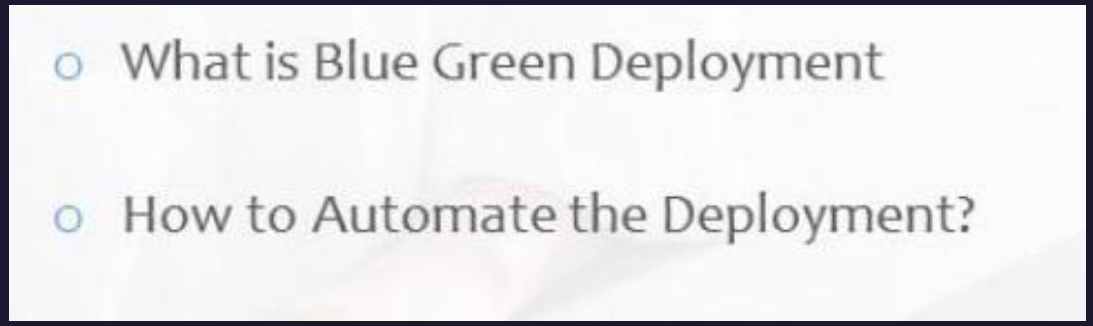
It is a physical server otherwise you will say a VM that runs the applications victimization Pods (a pod programing unit) that is controlled by the master node.

On a physical server (worker/slave node), pods area unit scheduled.

For accessing the applications from the external world, we have a tendency to connect with nodes.



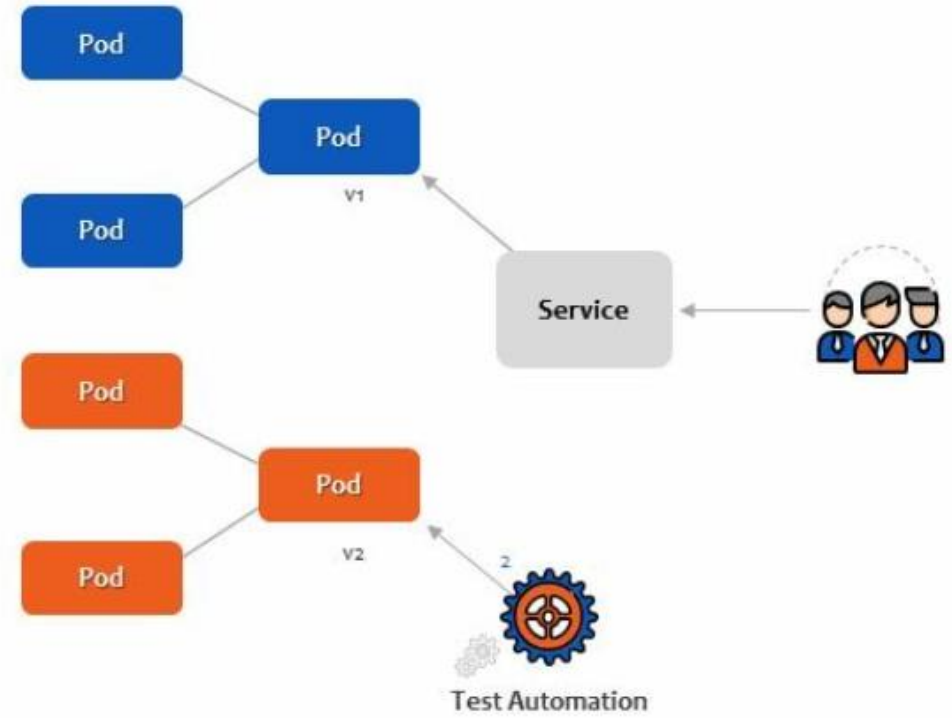
Kubernetes deployment process



What is blue green deployment

Blue-green deployment is a method that reduces period and risk by running 2 identical production environments called Blue and Green.

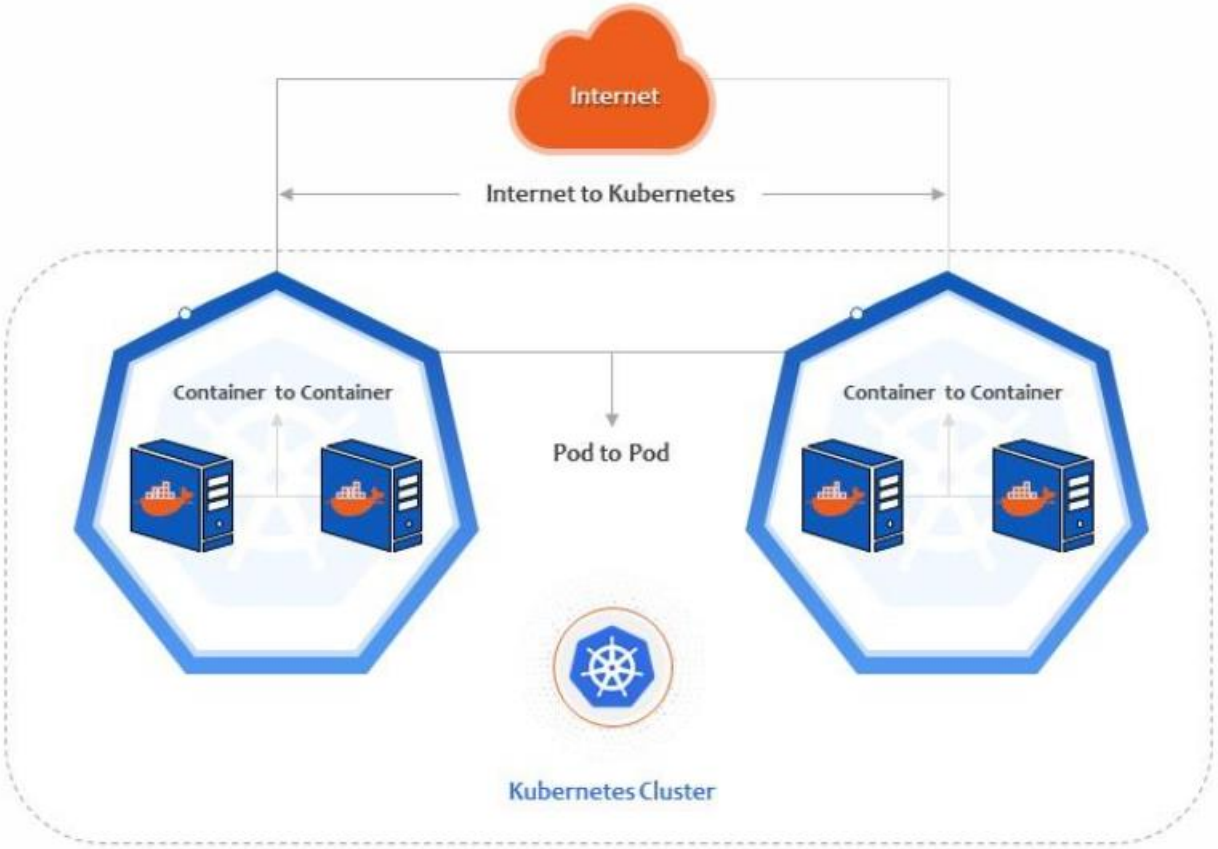
- At any time, just one of the environments is live, with the live atmosphere serving all production traffic.
- For this instance, Blue is presently live and Green is idle.



Networking in Kubernetes

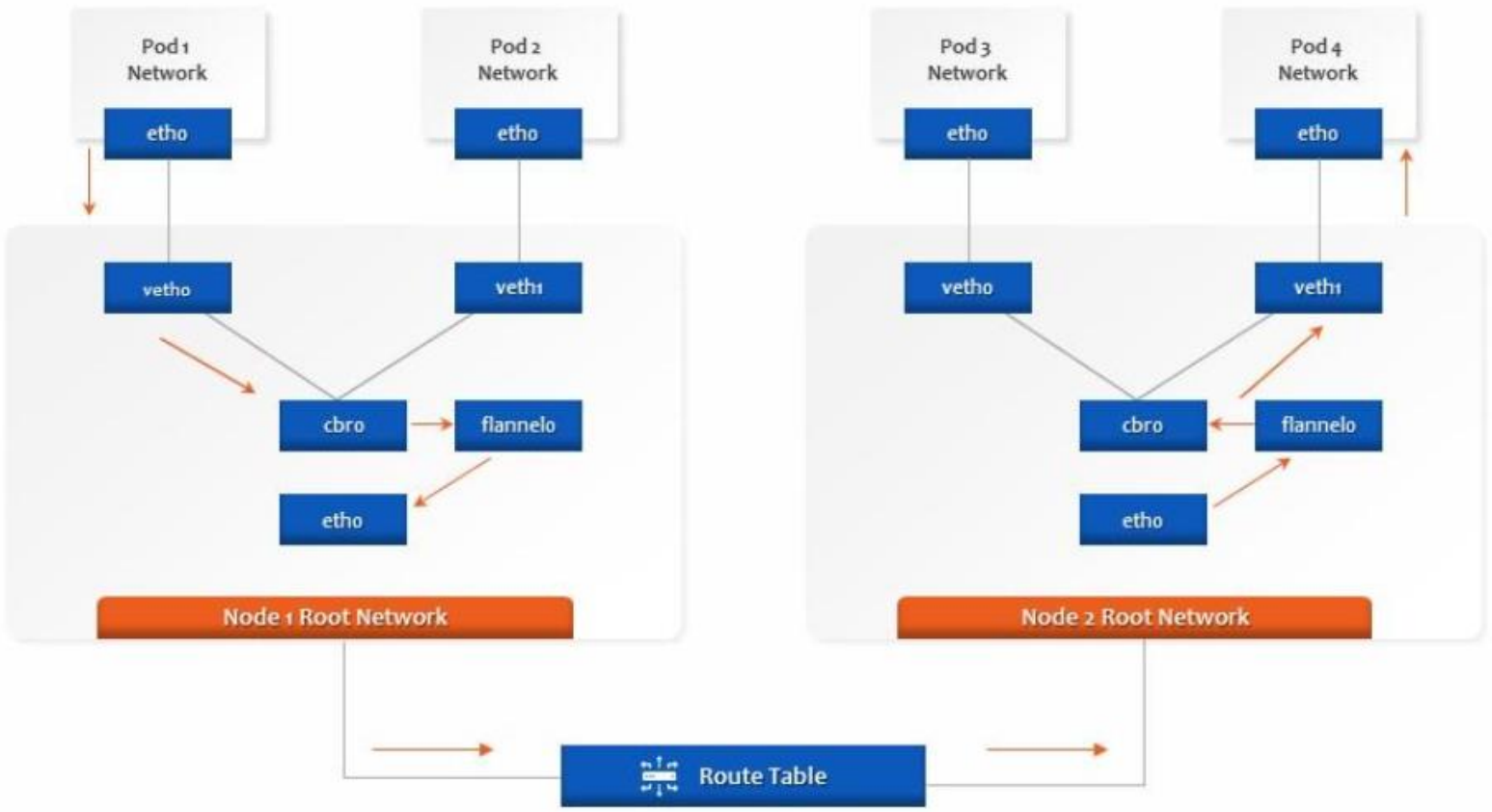
- Kubernetes Networking Model
- Ingress Networking in Kubernetes

Kubernetes Networking Model



Ingress Networking in Kubernetes

The slide explains the Ingress Networking in Kubernetes and its working



Thank You!