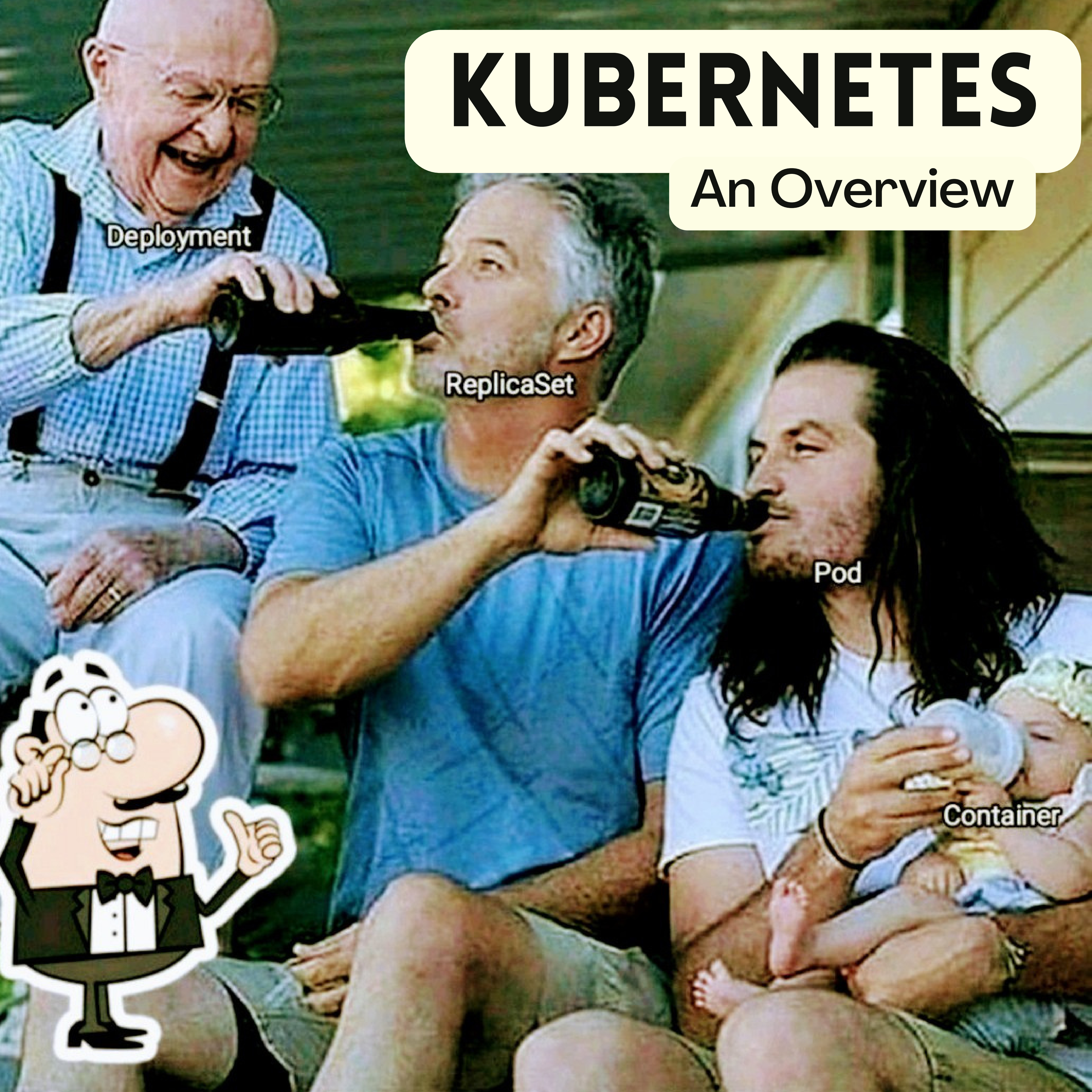
KUBERNETES

An Overview



What is a Container?

- A container is simply like a software unit/wrapper that will package everything – your application code, app related dependencies etc. together.

- You can assume like you get a portable environment to easily run your application. You can easily manage the container on your own (operations like starting, stopping, monitoring etc.).

Why Kubernetes?

- Suppose, you have a requirement for running 10 different applications (microservices) ~ 10 containers.

- And in case you need to scale each application for high availability, you create 2 replicas for each app ~ 2 * 10 = 20 containers.

Now you have to manage 20 containers.

- Would you be able to manage 20 containers on your own? (20 is just an example , there could be more based on the requirement).

It would be difficult , for sure.

Orchestration

- A Container Orchestration tool or framework can help you in such situations. It can help you automate all the deployment/management overhead.
- Once such Container Orchestration tool is Kubernetes.

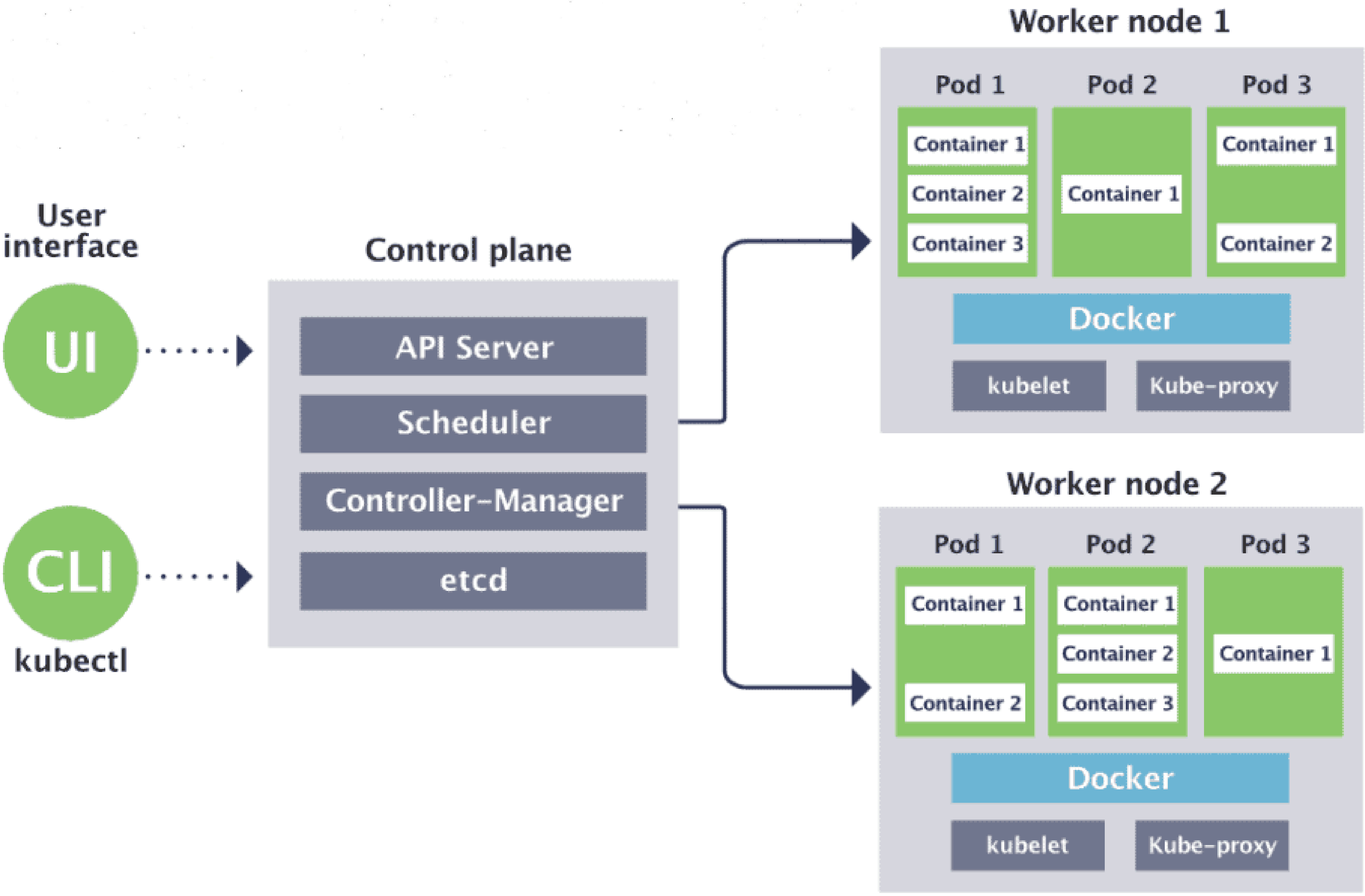
What is Kubernetes?

- Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications.
- It provides a set of abstractions and APIs for managing containers, so you can focus on building your application and not worry about the underlying infrastructure.

- With Kubernetes, you can manage multiple containers across multiple machines, making it easier to streamline and automate the deployment and management of your application infrastructure.

- Kubernetes is fast becoming the de facto standard for container orchestration in the cloud-native ecosystem.

Kubernetes Architecture



Control Plane : This is the brain of the Kubernetes cluster and manages the overall state of the system. It includes -

- API Server: Provides a REST API for the Kubernetes control plane and handles requests from various Kubernetes components and external clients.
- etcd: This is a distributed key-value store that stores the configuration data of the entire Kubernetes cluster.
- Controller Manager: This components ensures that the desired state of the cluster is maintained by monitoring the state of various Kubernetes objects (e.g., ReplicaSets, Deployments, Services) and reconciling any differences.
- Scheduler: This component assigns Pods to worker nodes based on resource availability and other scheduling policies.

Worker Nodes : These are the machines that run the application containers. Each worker node includes the following components:

- Kubelet: This component communicates with the API server to receive instructions and ensures that the containers are running correctly.

- Container Runtime: This is the software that runs the containerized applications (e.g., Docker, containerd).

- kube-proxy: This component handles network routing for services in the cluster.

Other Key Components –

- Pod: A pod is the smallest deployable unit in Kubernetes and represents a single instance of a running process in the cluster. A pod can contain one or more containers.

- Container: A container is a lightweight, standalone executable package that contains everything needed to run an application, including code, runtime, system tools, and libraries.

- Service: A service is an abstraction that defines a set of pods and a policy for how to access them. Services provide a stable IP address and DNS name for a set of pods, allowing other parts of the application to access them.

- ReplicaSet: It ensures that a specified number of replicas of a pod are running at all times. It takes care of auto scaling of the replicas based on demand.

- Deployment: A higher-level object that manages ReplicaSets and provides declarative updates to the pods and ReplicaSets in the cluster.

- ConfigMap: A configuration store that holds configuration data in key-value pairs.

- Secret: A secure way to store and manage sensitive information such as passwords, API keys, and certificates.

- Volume: A directory that is accessible to the containers running in a pod. Volumes can be used to store data or share files between containers.

Summary -

- You can imagine Kubernetes as a classical 'Master - Worker' cluster setup. Master node has responsibilities to perform absolutely necessary processes to run/manage the cluster and the Worker nodes would actually run your applications.

- So you basically instruct Kubernetes about the application's desired state and then it is responsibility of Kubernetes to achieve and maintain the state.

- You need to use YAML or JSON manifest/config files to give the instruction. (for example, I want to run 3 different springboot applications each having 2 replicas on some specified ports. I would prepare the manifest files and give it to kubernetes and rest would be taken care. 😊)

That's It.

Keep Learning !