EVIDEN

Kubernetes

EVIDEN

01 Container orchestration

Container Orchestration

Container orchestration refers to the automated process of managing, deploying, scaling, and maintaining containerized applications across multiple hosts or environments. It simplifies the complex task of coordinating all the services, infrastructure, and scheduling involved in running containerized applications at scale.

In modern software development, applications are broken down into microservices, each running in its own container. Managing these containers manually becomes difficult as the number of containers increases. This is where container orchestration platforms, such as Kubernetes, come in to handle these complexities.

Container Orchestration

Key Functions of Container Orchestration:

- Automated Deployment:
- Orchestrators ensure that containerized applications are automatically deployed across different nodes or hosts as needed.

- Scaling:
- It allows dynamic scaling of applications, ensuring the right number of containers are running based on the load or traffic.

- Load Balancing:
- It distributes the incoming requests across different containers, ensuring efficient use of resources and preventing any single container from becoming overloaded.

- Self-Healing:
- If a container or a node crashes, the orchestrator automatically replaces or restarts failed containers to maintain availability.

Container Orchestration

Key Features of Container orchestration

- Service Discovery:
- Orchestrators make it easy for containers to find and communicate with each other, ensuring seamless networking between containers.

- Storage Management:
- They manage the storage for containers, ensuring persistence where required, even if containers themselves are ephemeral.

- Monitoring and Logging:
- Orchestrators help monitor the health of containers and collect logs, making troubleshooting easier.

- Security:
- They enforce security policies, manage authentication, and ensure the proper isolation of containers.

Container Orchestration

How Container Orchestration is Accomplished

- Container orchestration is typically achieved using orchestration platforms or tools. The most common platform is Kubernetes, but other tools like Docker Swarm, Apache Mesos, and OpenShift also provide container orchestration capabilities..

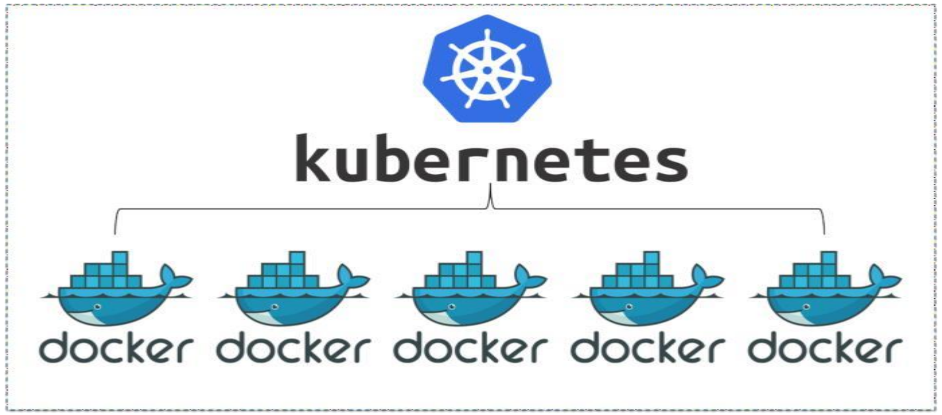
EVIDEN

02 Kubernetes Introduction

Introduction to Kubernetes

What is Kubernetes

- Kubernetes is an open-source Container Management tool which automates container deployment, container (de)scaling & container load balancing.
- Written on Golang, it has a huge community because it was first developed by Google & later donated to CNCF
- Purpose of Kubernetes is to host the applications in the form of containers in automated fashion so that you can deploy as many instances of application and easily enable the communication between different services within the application.



Introduction to Kubernetes

Use cases of Kubernetes

- E-commerce: You deploy and manage the e-commerce websites by autoscaling and load balancing you can manage the millions of users and transactions.
- Media and entertainment: You can store the static and dynamic data can deliver it to the across the world with out any latency to the end users.
- Financial services: kubernetes is well suited for the sinical application because of the level of security it is offering.
- Healthcare: You can store the data of patient and take care the outcomes of the health of patient.

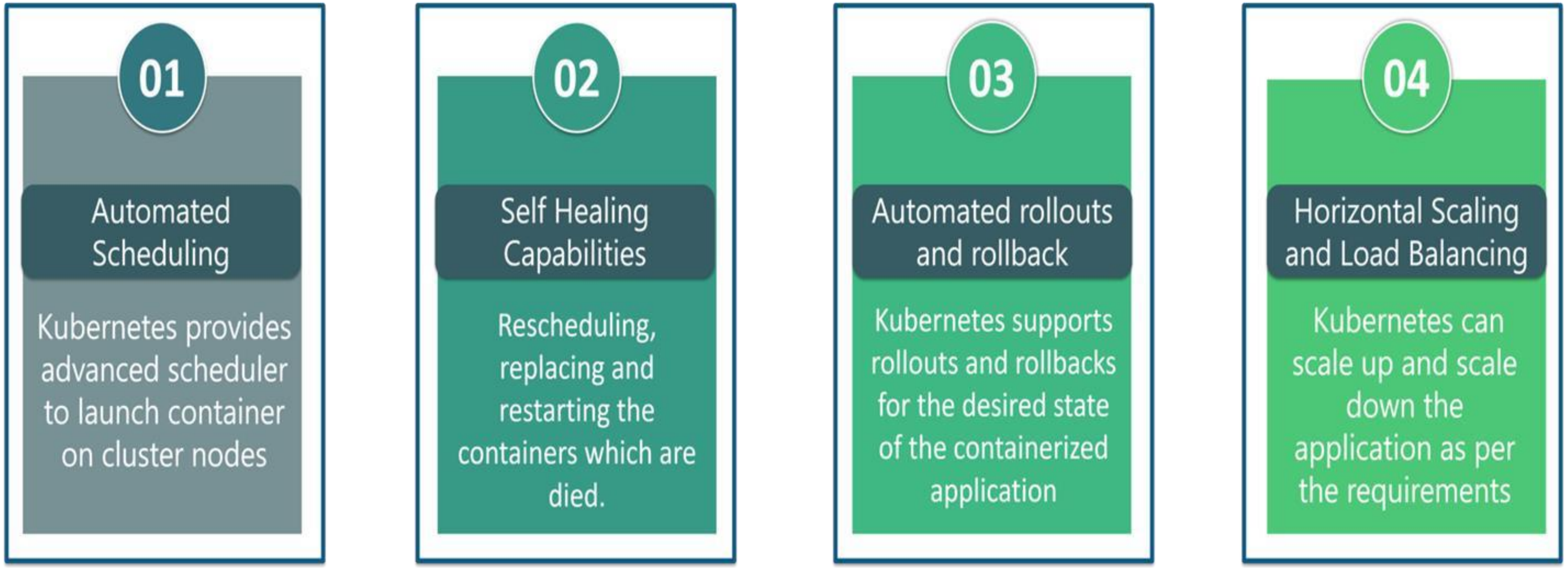
Introduction to Kubernetes

Benefits of Kubernetes

- **Automated deployment and management**
- If you are using Kubernetes for deploying the application then no need for manual intervention kubernetes will take care of everything like automating the deployment, scaling, and containerizing the application.
- **Scalability**
- You can scale the application containers depending on the incoming traffic Kubernetes offers Horizontal pod scaling the pods will be scaled automatically depending on the load.
- **High availability**
- You can achieve high availability for your application with the help of Kubernetes and also it will reduce the latency issues for the end users.
- **Cost-effectiveness**
- If there is unnecessary use of infrastructure the cost will also increase kubernetes will help you to reduce resource utilization and control the overprovisioning of infrastructure.
- **Improved developer productivity**
- Developer can concentrate more on the developing part kubernetes will reduce the efforts of deploying the application.

Introduction to Kubernetes

Features of Kubernetes



01

Automated Scheduling

Kubernetes provides advanced scheduler to launch container on cluster nodes

02

Self Healing Capabilities

Rescheduling, replacing and restarting the containers which are died.

03

Automated rollouts and rollback

Kubernetes supports rollouts and rollbacks for the desired state of the containerized application

04

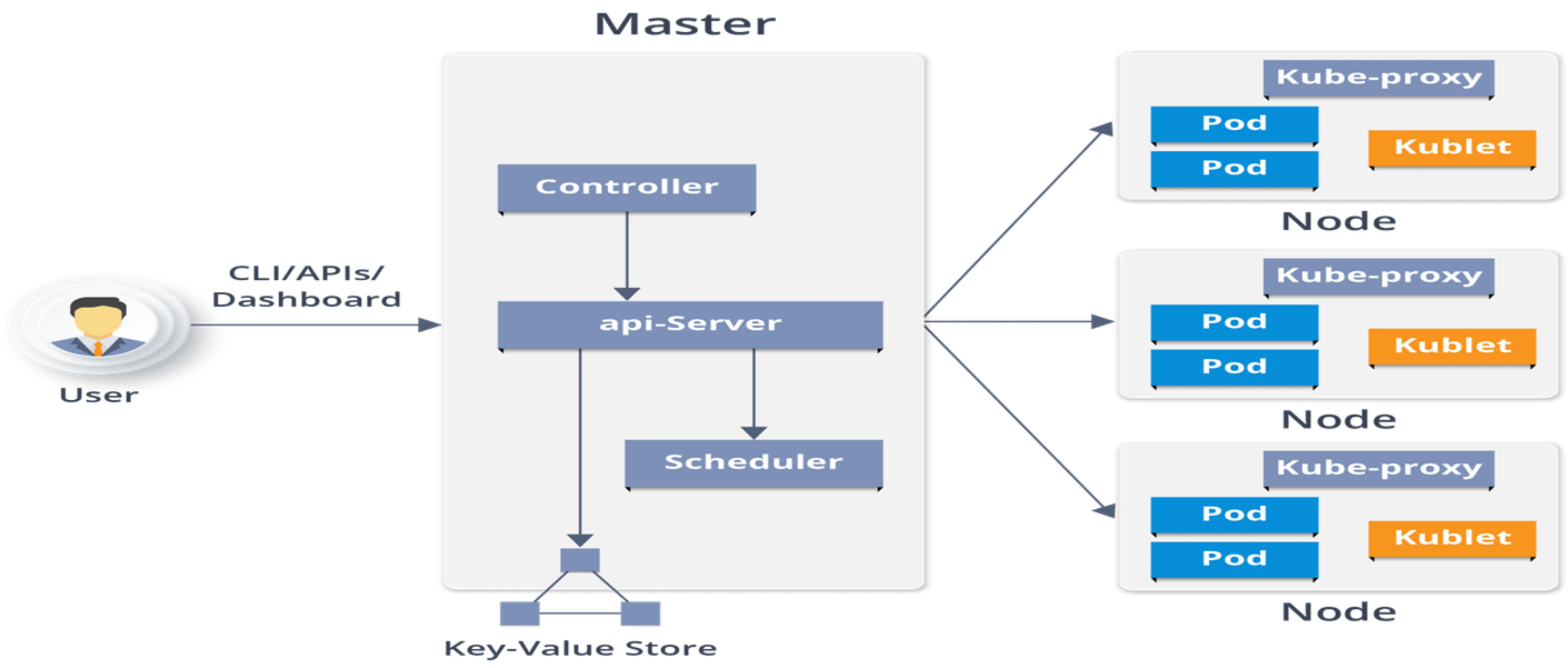
Horizontal Scaling and Load Balancing

Kubernetes can scale up and scale down the application as per the requirements

EVIDEN

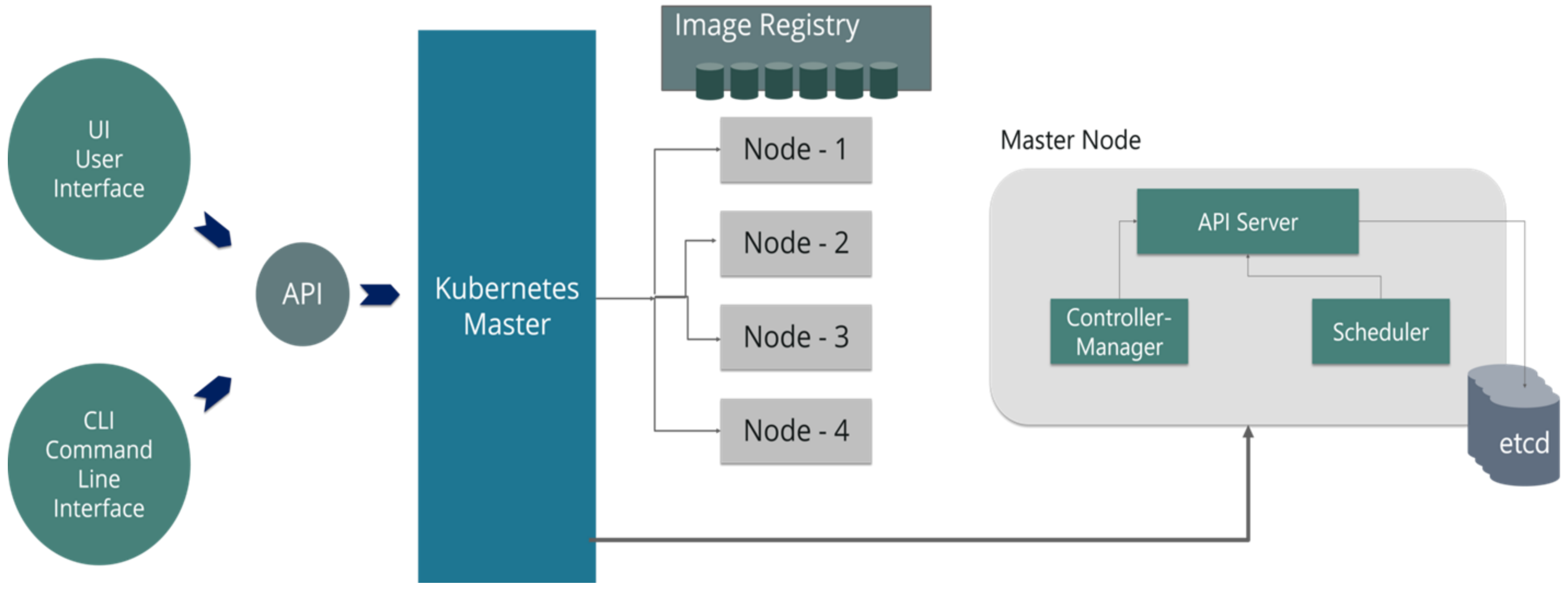
03 Kubernetes architecture

Kubernetes architecture



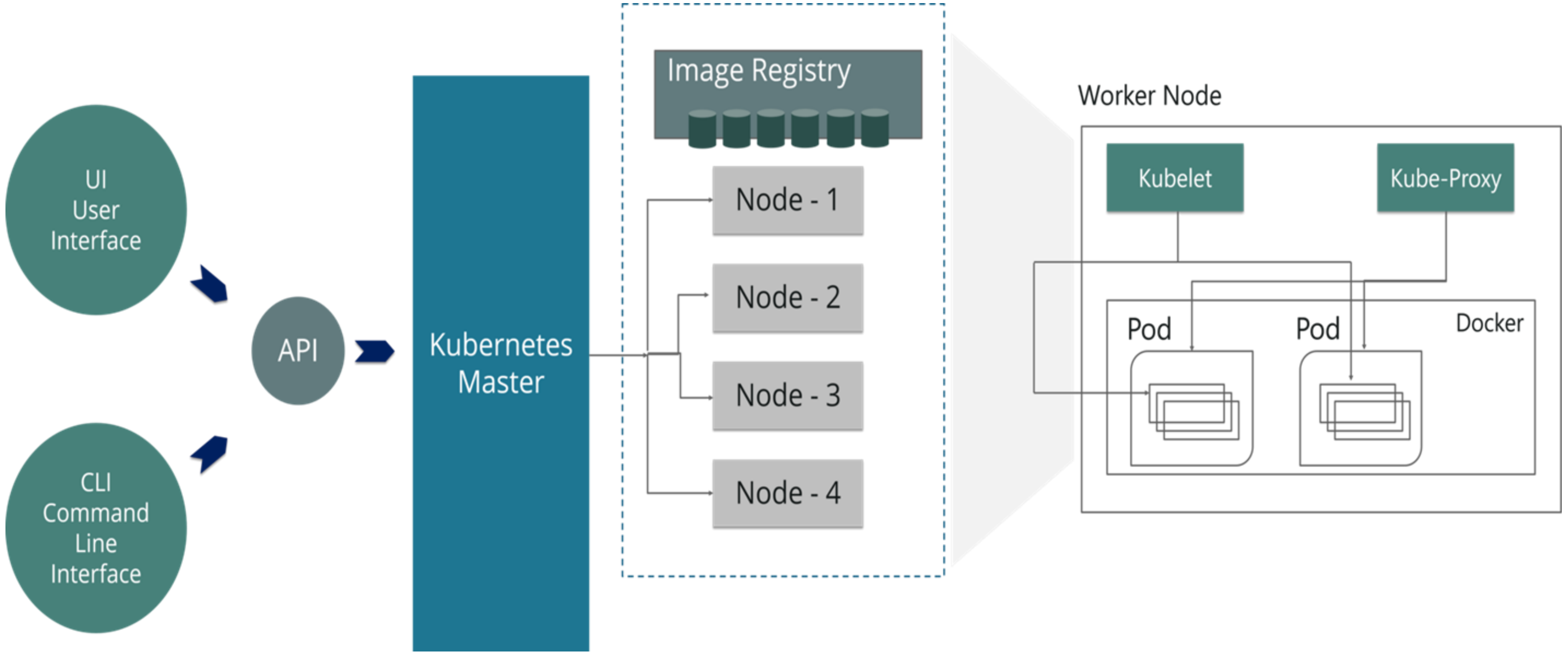
Kubernetes architecture

Master Node



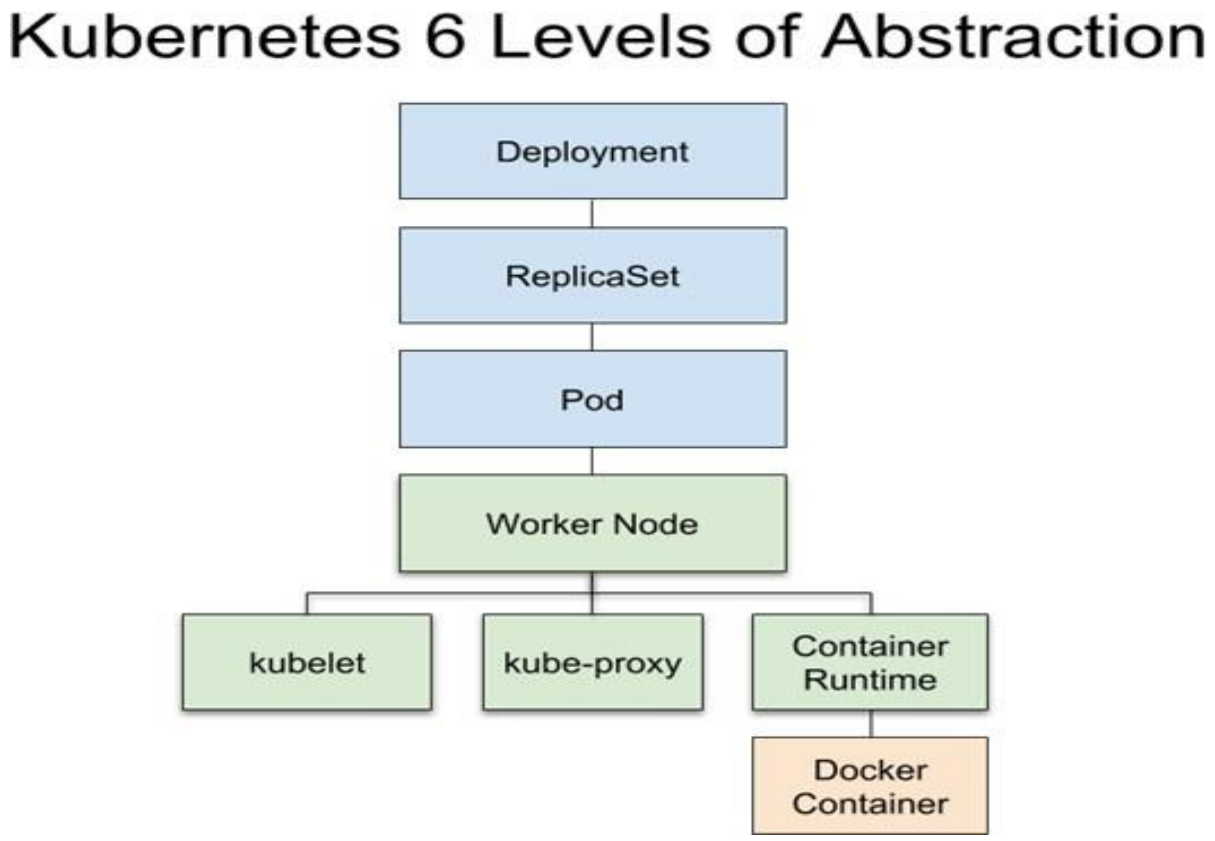
Kubernetes architecture

Worker Node



Kubernetes architecture

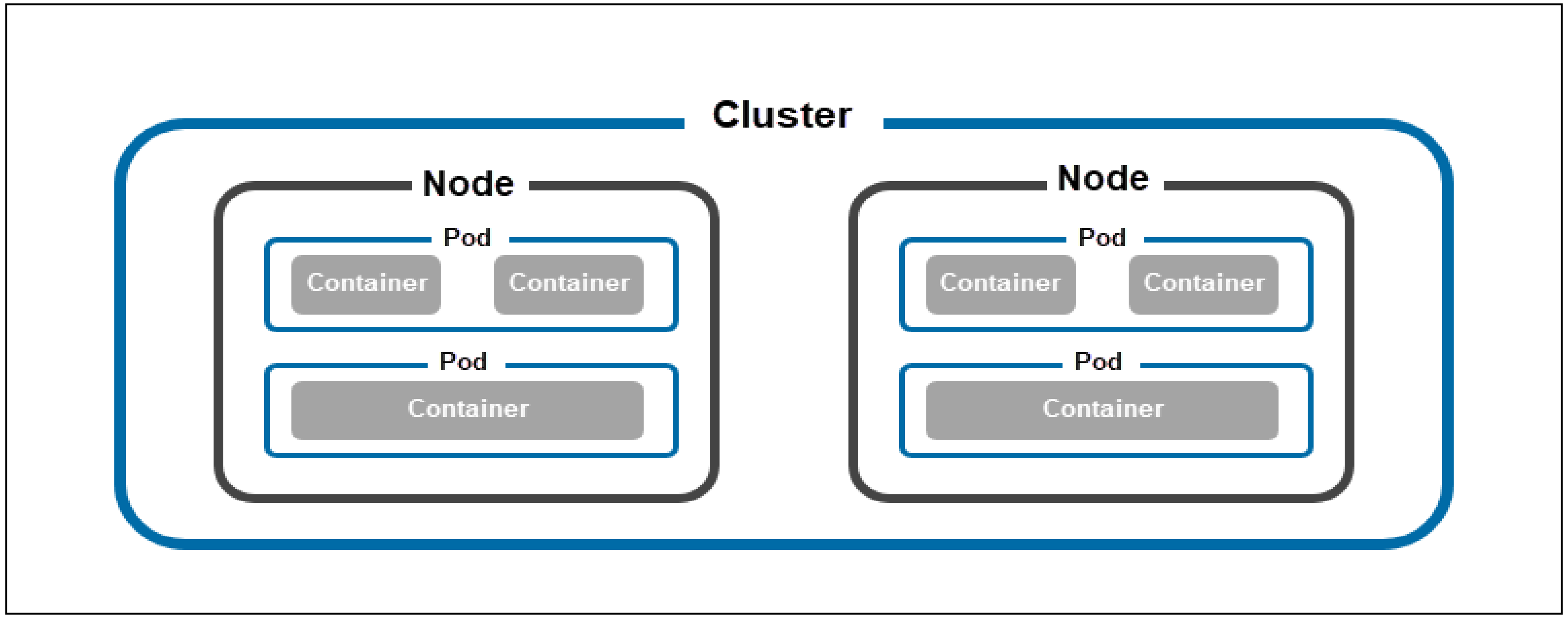
6 levels of abstraction



04 Kubernetes Object

Kubernetes Object

- Pods
- Kubernetes pods are the smallest unit of deployment in Kubernetes
- They reside on cluster nodes and have their IP addresses, enabling them to communicate with the rest of the cluster.
- A single pod can host one or more containers, providing storage and networking resources.



Kubernetes Object

- Services:
- Services provide a way to expose applications running in pods. Their purpose is to represent a set of pods that perform the same function and set the policy for accessing those pods.
- Volumes
- Volumes are objects whose purpose is to provide storage to pods. There are two basic types of volumes in Kubernetes:

- Ephemeral volumes persist only during the lifetime of the pod they are tied to.
- Persistent volumes, which are not destroyed when the pod crashes. Persistent volumes are created by issuing a request called PersistentVolumeClaim (PVC). Kubernetes uses PVCs to provision volumes, which then act as links between pods and physical storage.

Kubernetes Object

- Namespaces
- The purpose of the Namespace object is to act as a separator of resources in the cluster. A single cluster can contain multiple namespaces, allowing administrators to organize the cluster better and simplify resource allocation.
- Deployments
- Deployments are controller objects that provide instructions on how Kubernetes should manage the pods hosting a containerized application. Using deployments, administrators can:
  - Scale the number of pod replicas.
  - Rollout updated code.
  - Perform rollbacks to older code versions.

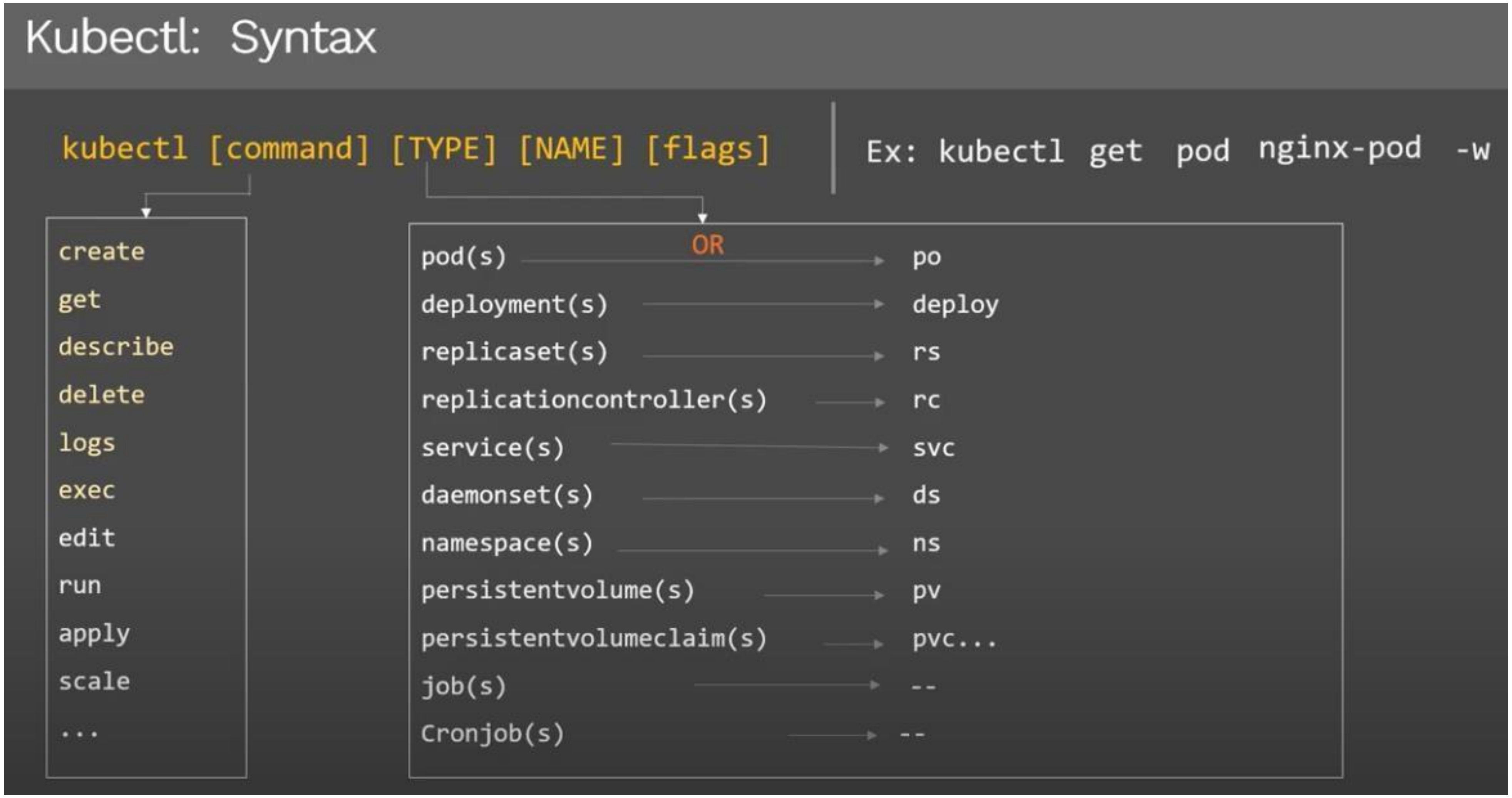
- ReplicationControllers
- ReplicationControllers ensure that the correct number of pod replicas are running on the cluster at all times. When creating a ReplicationController, the administrator specifies the desired number of pods. The controller then maintains their number, creating additional pods and terminating the extra ones when necessary.

Kubernetes Object

- ReplicaSets
- ReplicaSets serve the same purpose as ReplicationControllers, i.e. maintaining the same number of pod replicas on the cluster.
- DaemonSets
- DaemonSets are controller objects whose purpose is to ensure that specific pods run on specific (or all) nodes in the cluster. Kubernetes scheduler ignores the pods created by a DaemonSet
- It is used for cluster-wide services like log collection, monitoring, etc.
- ConfigMap
- Used to store configuration data in key-value pairs.
- ConfigMaps decouple configuration artifacts from image content to keep containerized applications portable.
- Secret
- A Secret is similar to a ConfigMap, but is specifically intended for sensitive data such as passwords, tokens, or keys.
- Secrets can be encrypted and are handled securely

05 Kubectl commands

Kubectl commands



```
kubectl [command] [TYPE] [NAME] [flags]
```

```
Ex: kubectl get pod nginx-pod -w
```

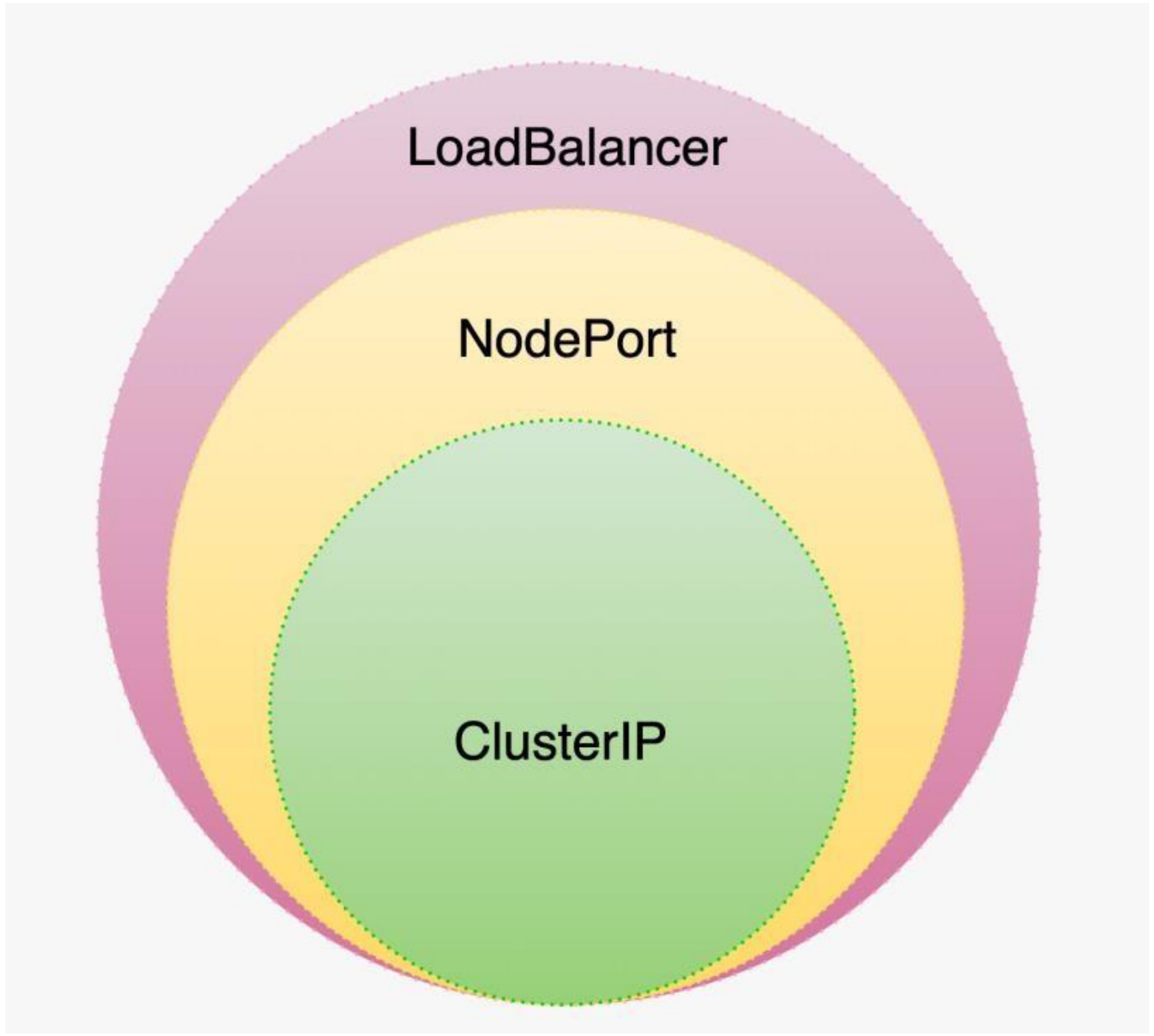
```
create
get
describe
delete
logs
exec
edit
run
apply
scale
...
```

```
pod(s)                      -> po
deployment(s)               -> deploy
replicaset(s)               -> rs
replicationcontroller(s)    -> rc
service(s)                  -> svc
daemonset(s)                -> ds
namespace(s)                -> ns
persistentvolume(s)         -> pv
persistentvolumeclaim(s)    -> pvc...
job(s)                      -> --
Cronjob(s)                  -> --
```

EVIDEN

06 Services

Services



Services

- Cluster IP Service:
- Cluster IP – Exposes the service on an internal IP in the cluster. This type makes the service only reachable from within the cluster.
- It is the default service
- Node Port – Exposes the service on the same port of each selected Node in the cluster.
- Makes a service accessible from outside the cluster <NodeIP>:<NodePort>
- Superset of ClusterIP.
- Load Balancer– creates an external load balancer in the current cloud and assigns a external IP to the service.
- Superset of NodePort