KUBERNETES

Kubernetes is a open-source platform for managing containerized workloads and services.

Kubernetes facilitates declarative configuration and automation.

Kubernetes originates from Greek term meaning helmsman or pilot.

Google open-sourced the Kubernetes project in 2014.

Containers are a good way to bundle and run your applications.

In a production environment, containers that run the applications should be always running without downtime.

For example, if a container goes down, another container needs to start.

That is Kubernetes orchestration.

Kubernetes (K8s) is an open source container orchestration tool originally developed and maintained google

Originally developed by google and now maintained by Cloud Native Computing Foundation (CNCF)

Kubernetes orchestration tool can handle:

Automated binpacking

Provision and deployment of containers

Scaling up or Scaling down containers depending on the load.

Movement of containers from one host to another host depending on the resources.

Loadbalancing of containers based on the request by users.

Healing of containers

Rollouts and Rollbacks

Container orchestration solutions

Docker Swarm

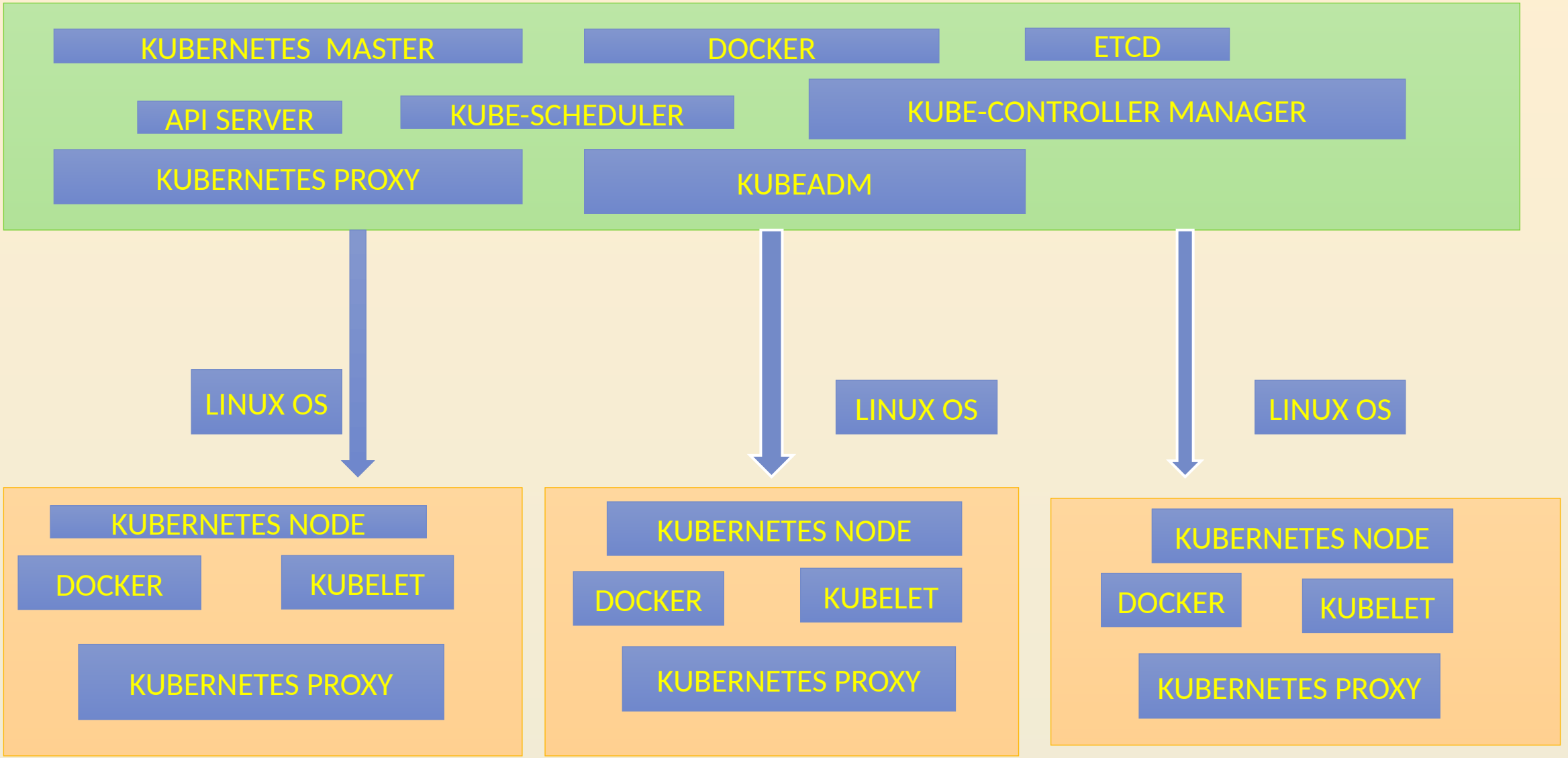
Kubernetes

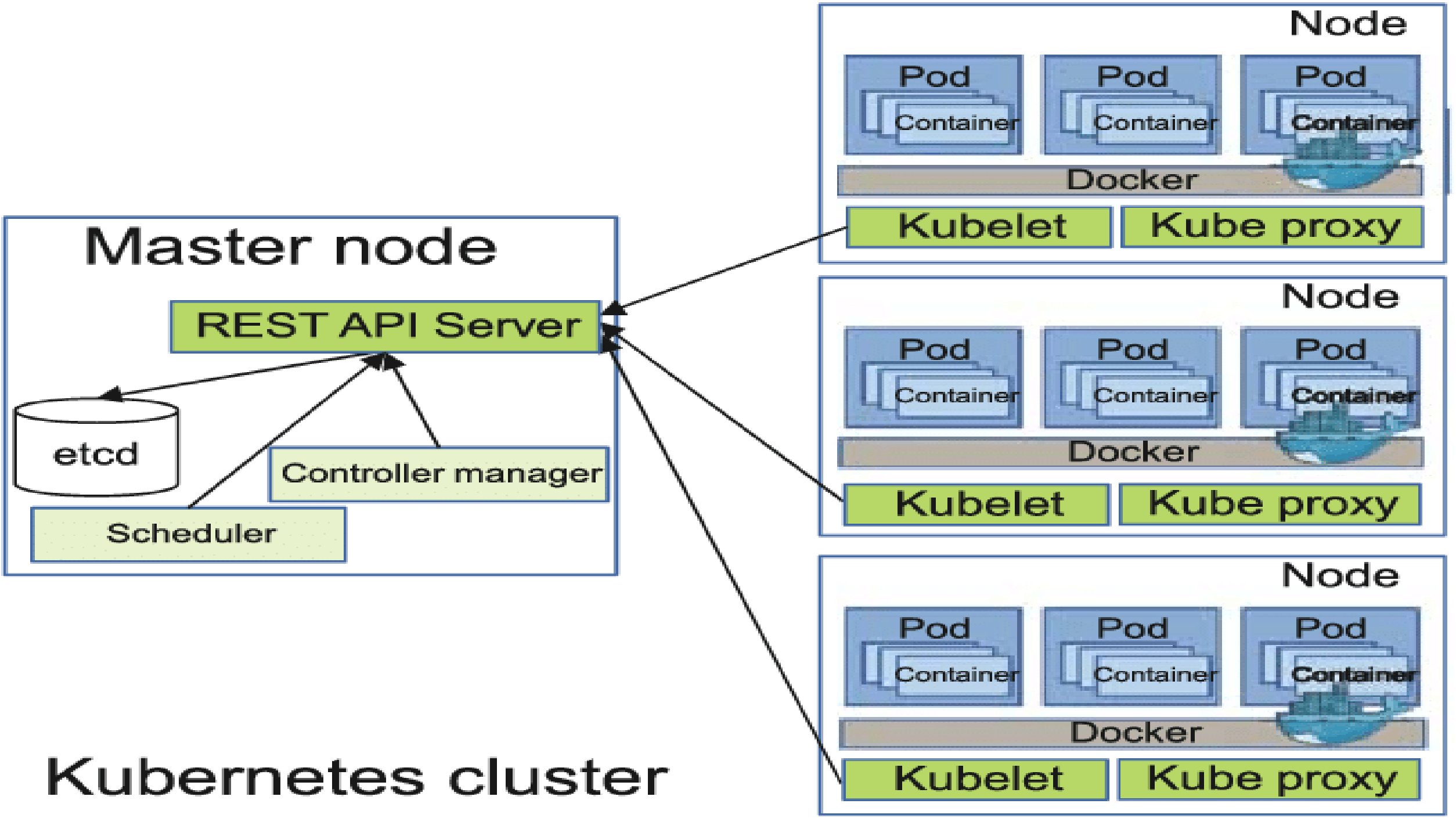
Apache Mesos

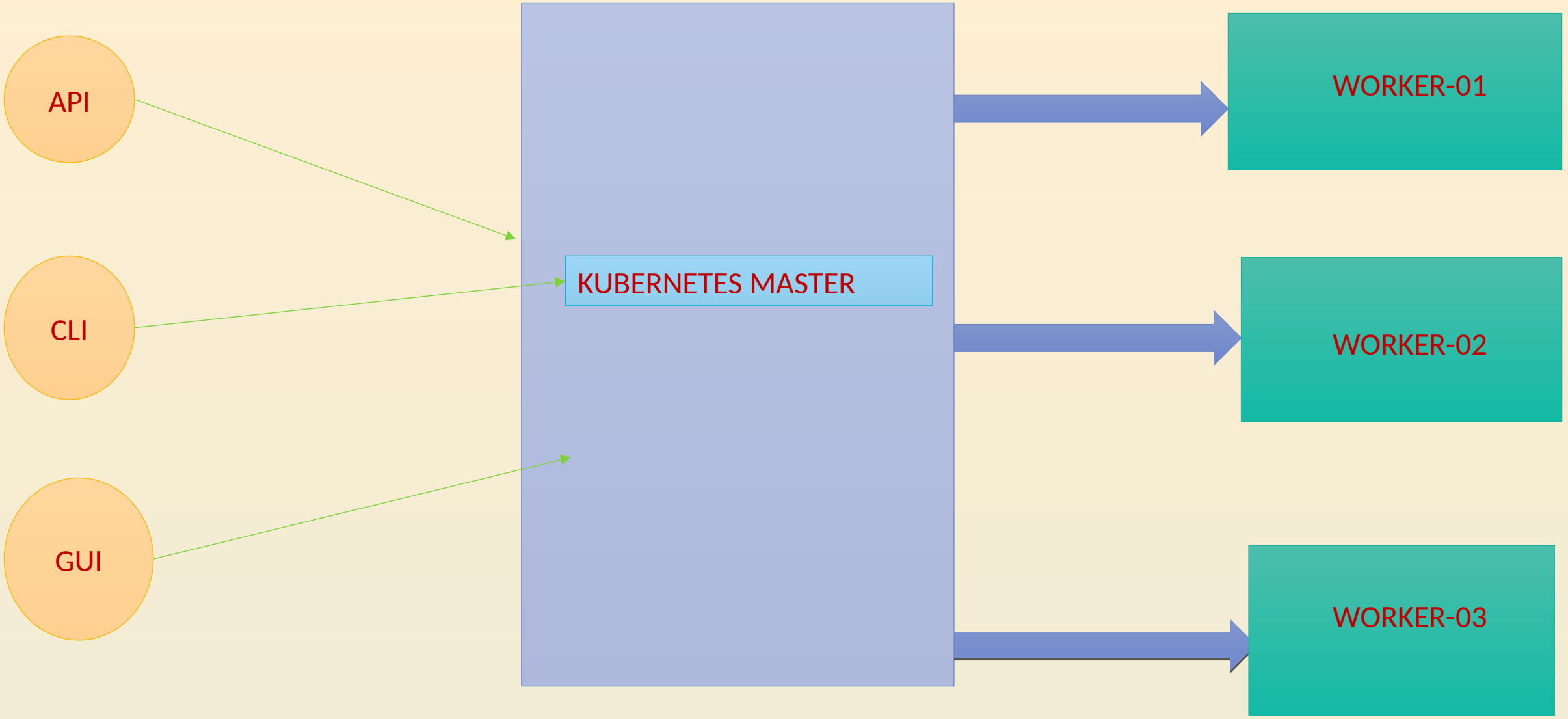
Elastic Kubernetes Service (AWS EKS)

Google Container Engine (GKE)

Kubernetes - Cluster Architecture




Node
Pod
Container
Pod
Container
Pod
Container
Docker
Kubelet
Kube proxy
Node
Pod
Container
Pod
Container
Pod
Container
Docker
Kubelet
Kube proxy
Node
Pod
Container
Pod
Container
Pod
Container
Docker
Kubelet
Kube proxy
Master node
REST API Server
etcd
Controller manager
Scheduler
Kubernetes cluster


API
CLI
GUI
KUBERNETES MASTER
WORKER-01
WORKER-02
WORKER-03

KUBERNETES INSTALLATION METHODS

Use managed Kubernetes Service from various providers like AWS, AZURE, GOOGLE,IBM, DigitalOcean and others.

Minikube (Installation of kubernetes locally a single node cluster)

Install and Configure Kubernetes Manually (Hard Way)

POD

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.

A *Pod* is a group of one or more containers.

Pods that run a single container. The "one-container-per-Pod" model is the most common Kubernetes use case.

in this case, you can think of a Pod as a wrapper around a single container.

Kubernetes manages Pods rather than managing the containers directly.

Pods that run multiple containers that need to work together.

A Pod can run co-located containers form a single cohesive unit of service

For example,

one container serving data stored in a shared volume to the public,

while a separate *sidecar* container refreshes or updates those files.

Each Pod is meant to run a single instance of a given application.

If you want to scale your application horizontally (to provide more overall resources by running more instances),

you should use multiple Pods, one for each instance.

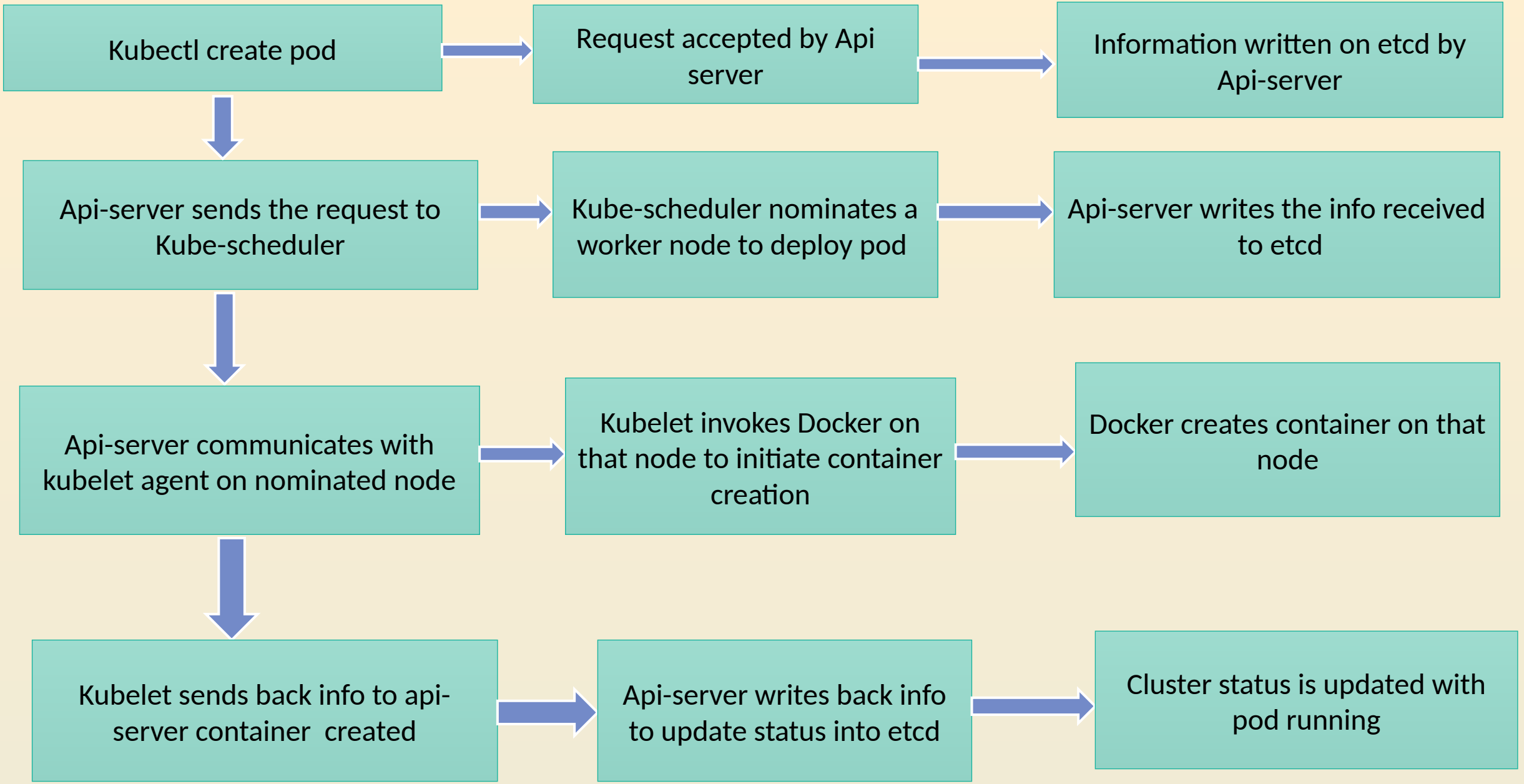
In Kubernetes, this is typically referred to as *replication*.

Replicated Pods are usually created and managed as a group by a controller.

A Pod is not a process, but an environment for running container(s).

A Pod persists until it is deleted.

Flow of Deployment of a Pod in Kubernetes Cluster



Replicaset

A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time.

It is often used to guarantee the availability of a specified number of identical Pods.

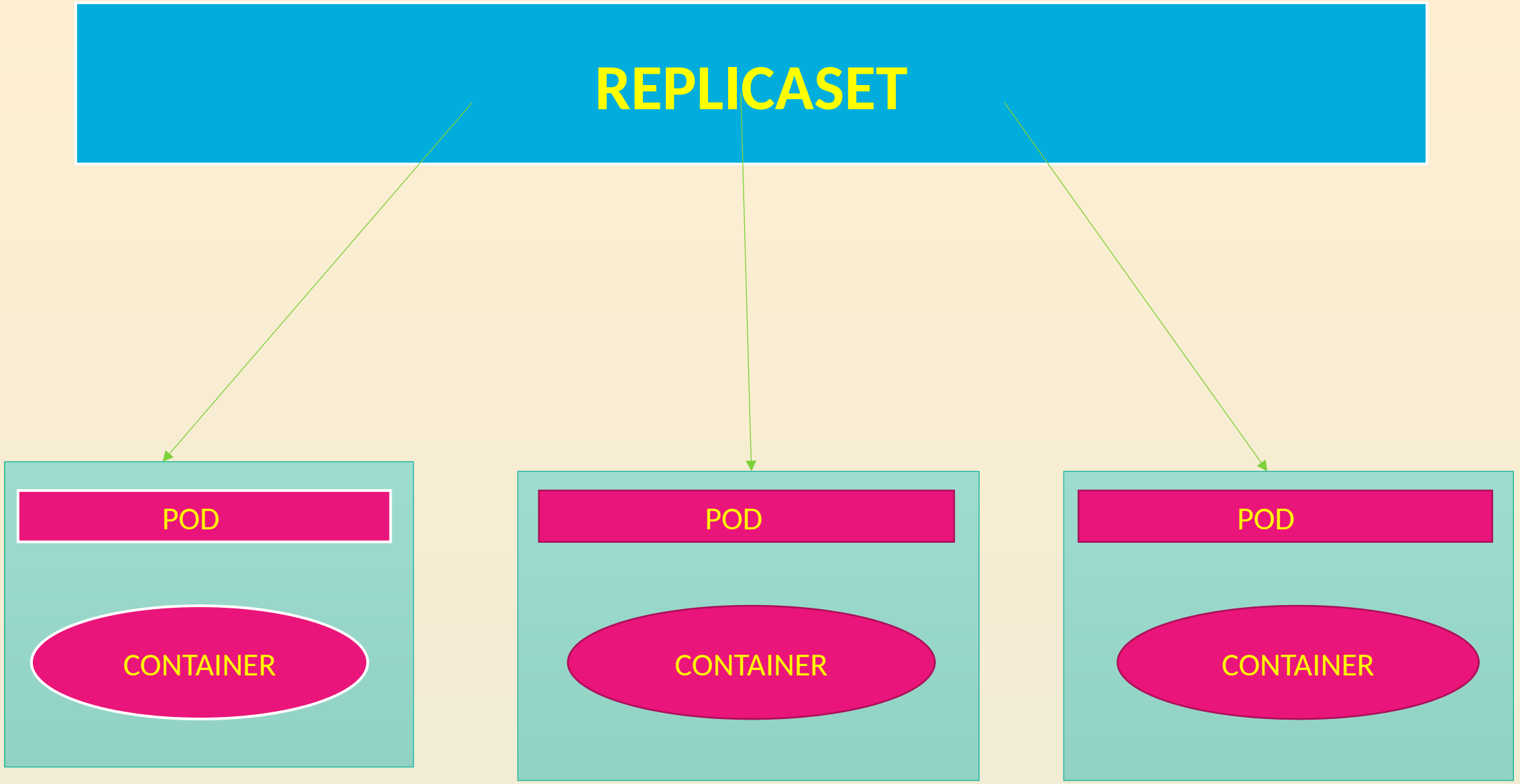
How a ReplicaSet works

Number of replicas indicate how many Pods it should be maintaining,

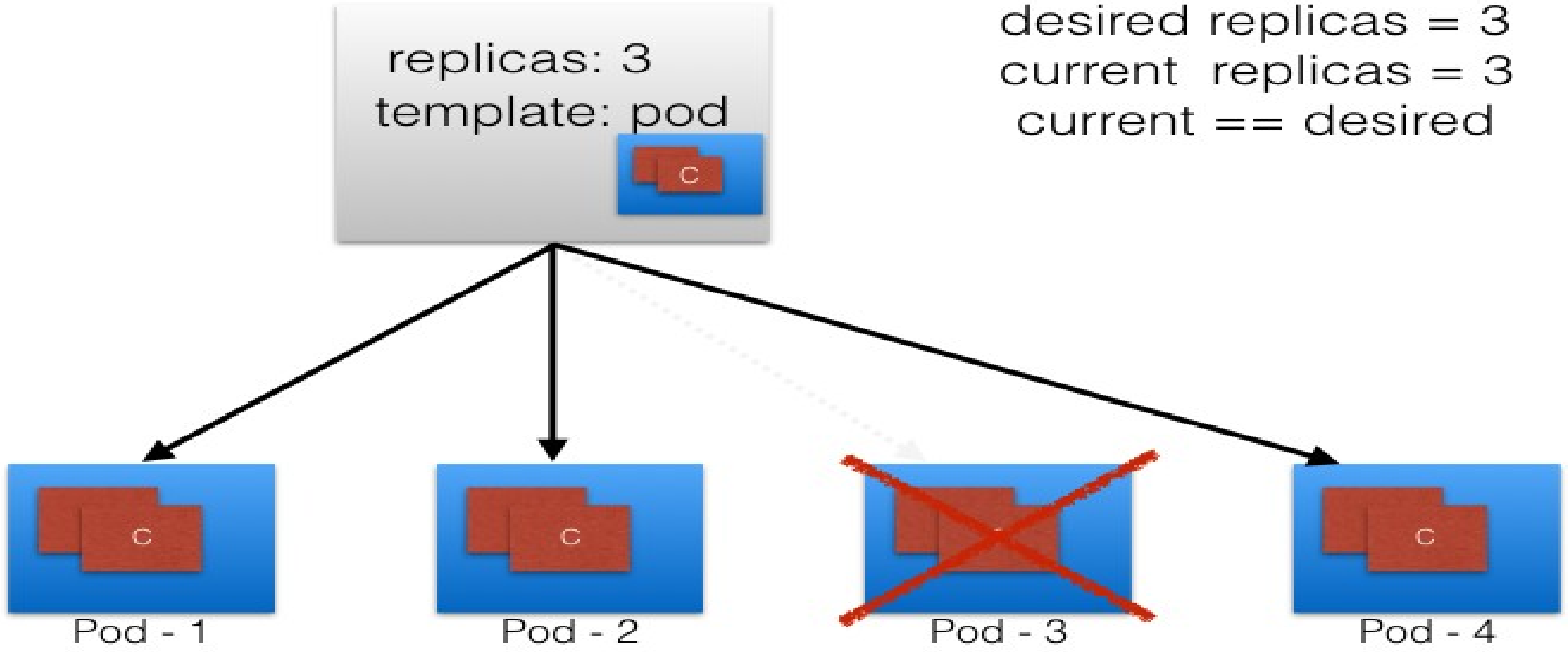
A pod template specifying the data of new Pods it should create to meet the number of replicas criteria.

When a ReplicaSet needs to create new Pods, it uses its Pod template.

A ReplicaSet then fulfills its purpose by creating and deleting Pods as needed to reach the desired number.


REPLICASET
POD
CONTAINER
POD
CONTAINER
POD
CONTAINER

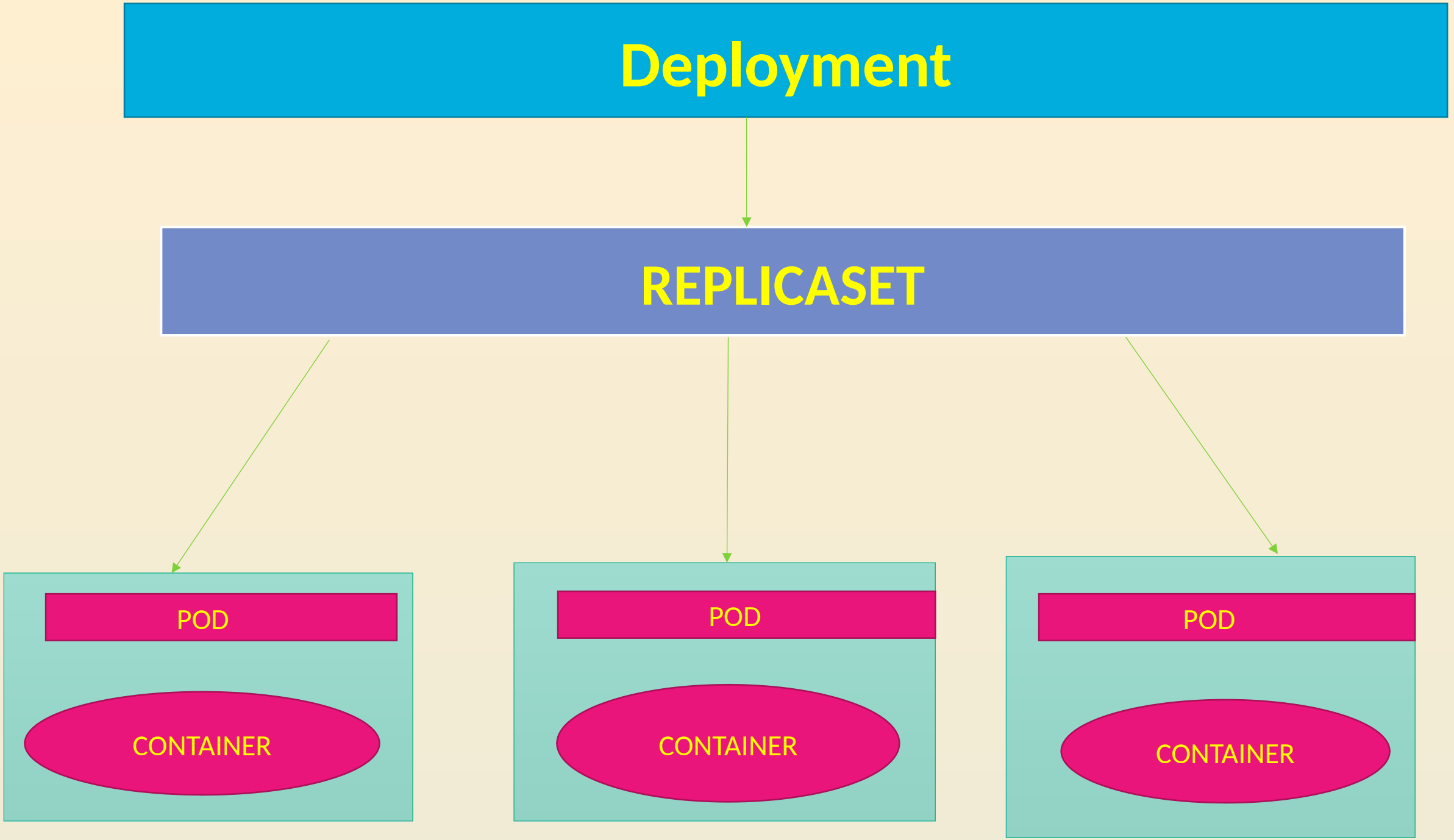
Replica Set

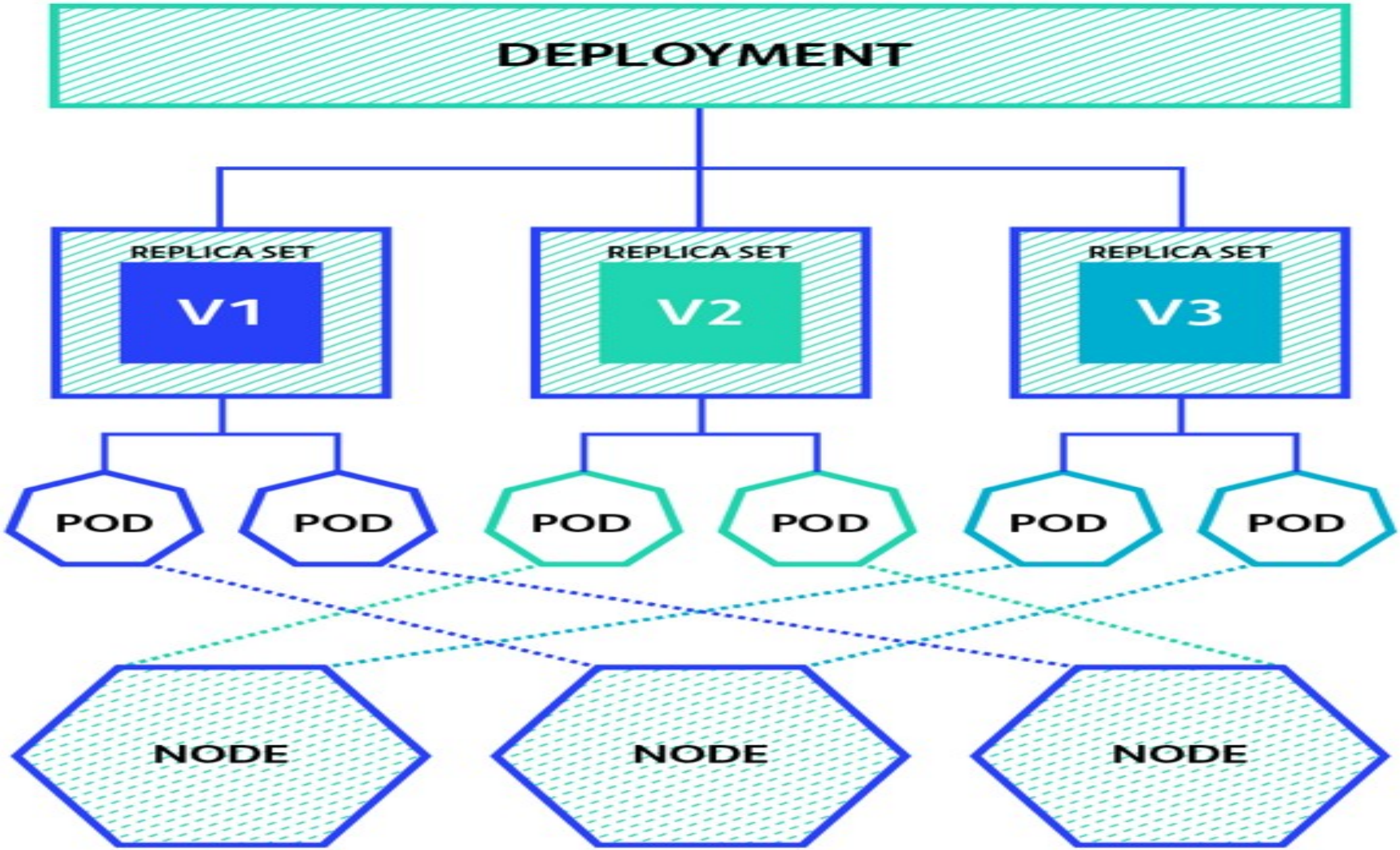

replicas: 3
template: pod
C
desired replicas = 3
current replicas = 3
current == desired
C
C
C
C
Pod - 1
Pod - 2
Pod - 3
Pod - 4

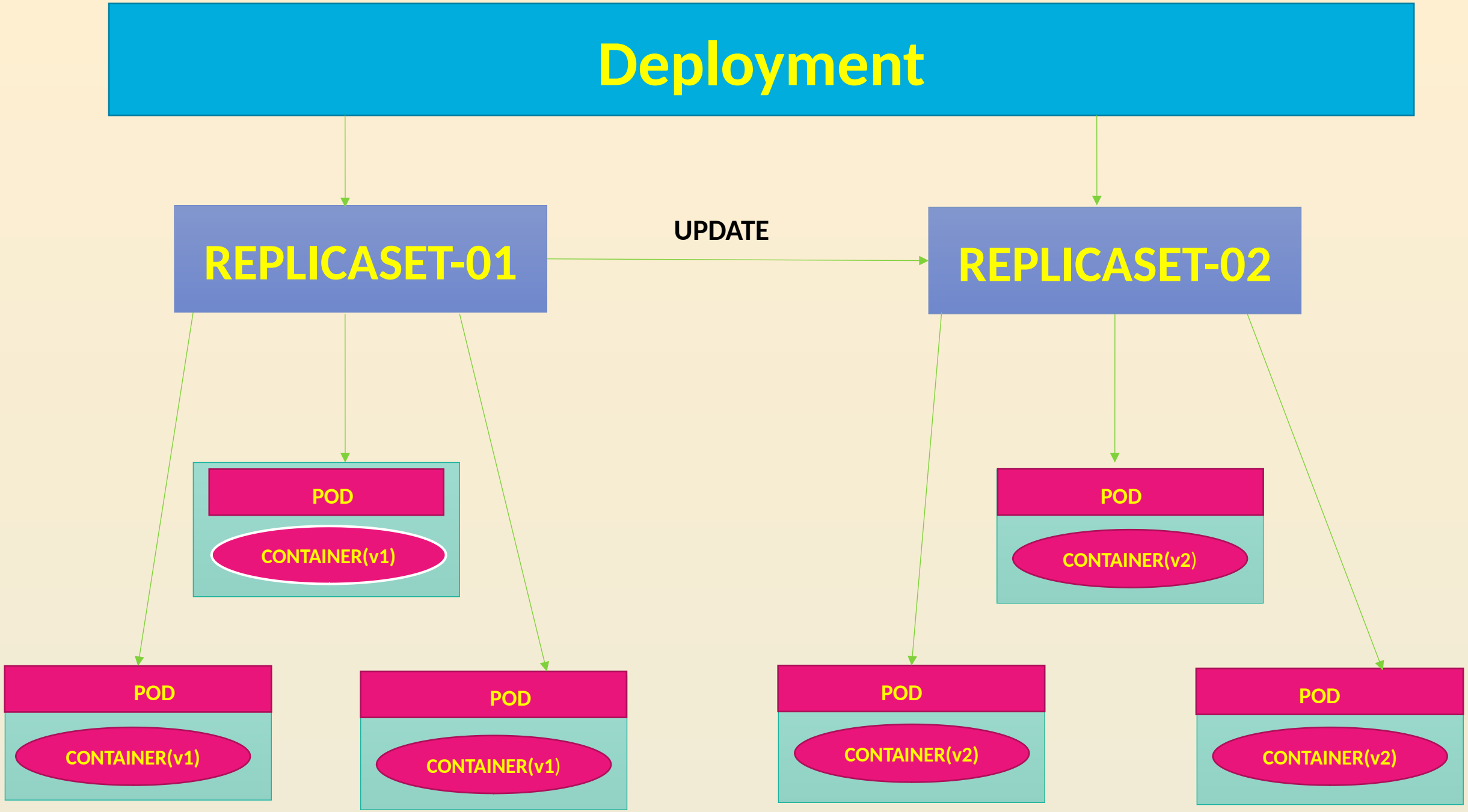
DEPLOYMENT

Deployment manages replica sets which is also an upgraded version of the replication controller.

Deployment has capability to update the replica set and are also capable of rolling back to the previous version.


Deployment
REPLICASET
POD
CONTAINER
POD
CONTAINER
POD
CONTAINER


DEPLOYMENT
REPLICA SET
V1
REPLICA SET
V2
REPLICA SET
V3
POD
POD
POD
POD
POD
POD
NODE
NODE
NODE


Deployment
REPLICASET-01
UPDATE
REPLICASET-02
POD
CONTAINER(v1)
POD
CONTAINER(v1)
POD
CONTAINER(v1)
POD
CONTAINER(v2)
POD
CONTAINER(v2)
POD
CONTAINER(v2)

Kubernetes service

Kubernetes Pods are created and destroyed to match the state of your cluster.

Pods are nonpermanent resources.

If you use a Deployment to run your app, it can create and destroy Pods dynamically.

Each Pod gets its own IP address.

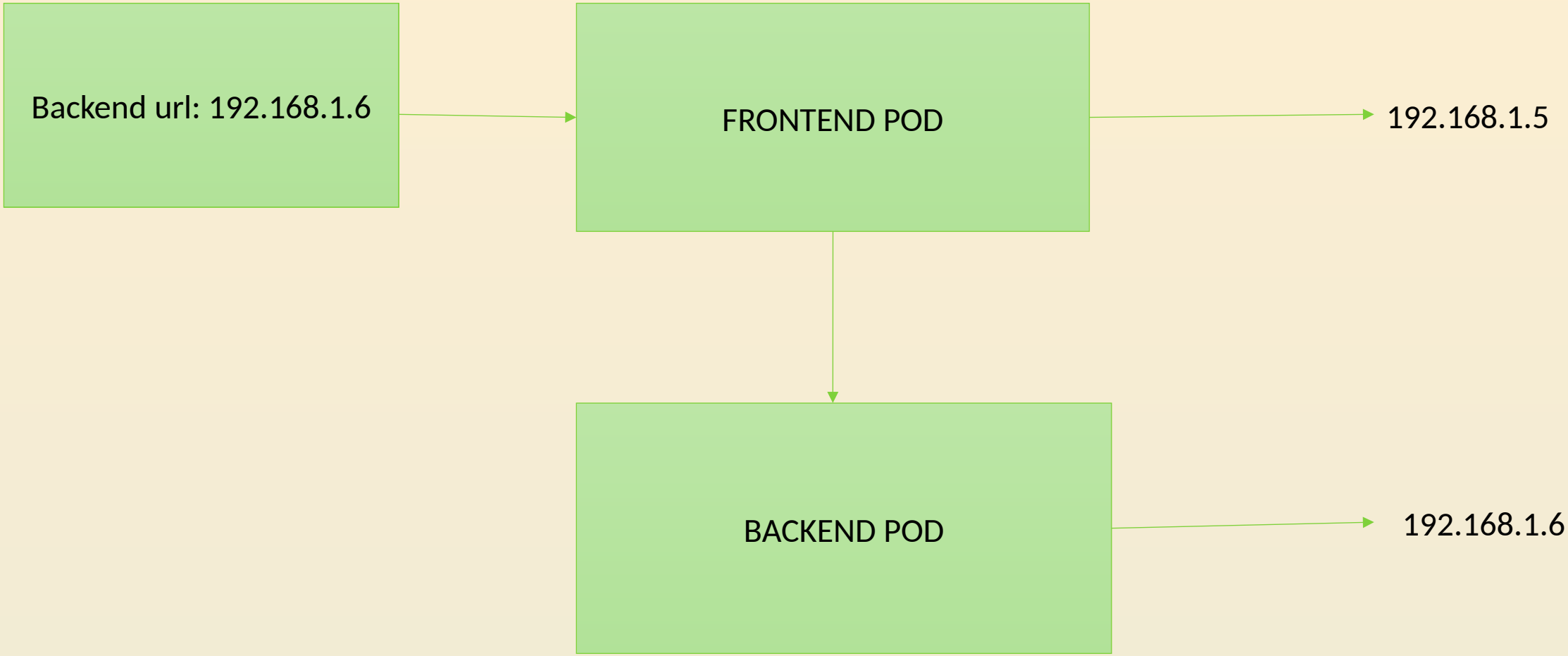
This leads to a problem:

If some set of Pods (call them "backends") provides functionality to other Pods (call them "frontends") inside your cluster,

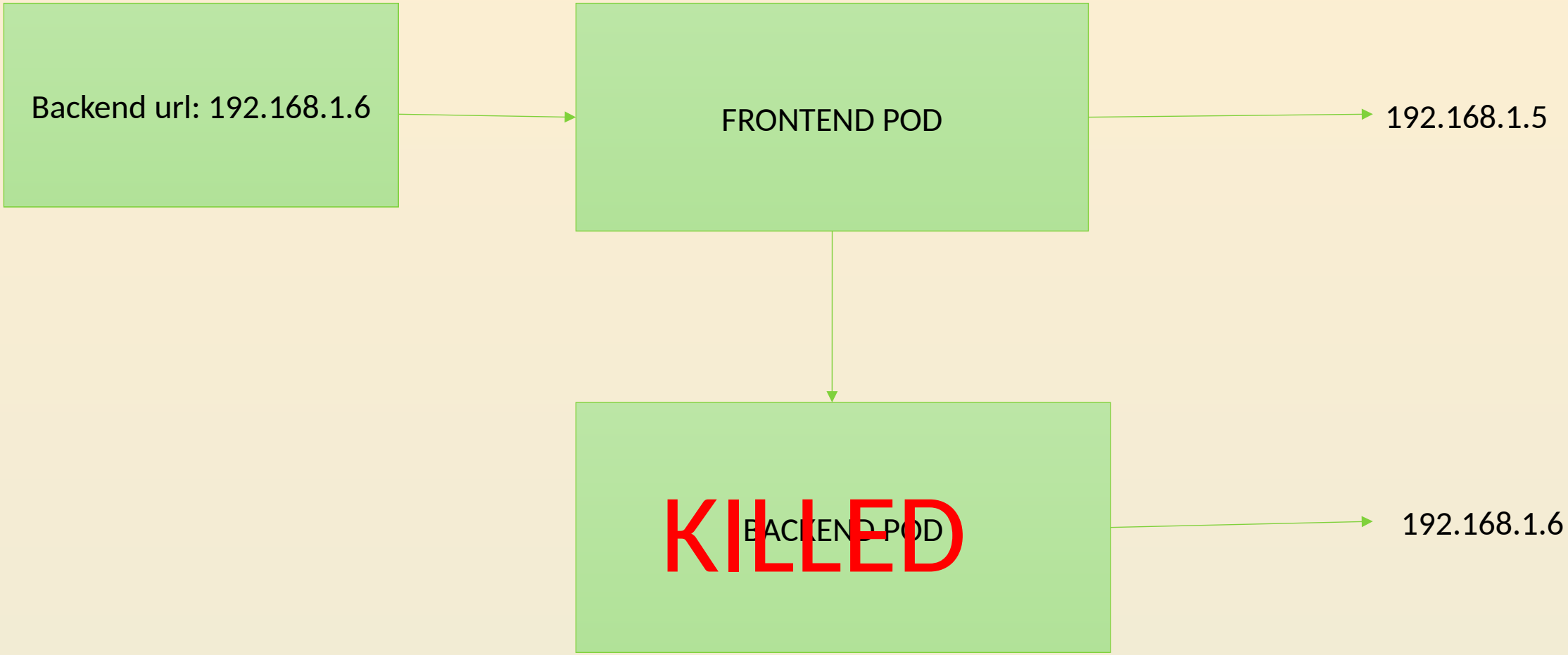
How do the frontends find out and keep track of which IP address to connect to?

KUBERNETES SERVICE

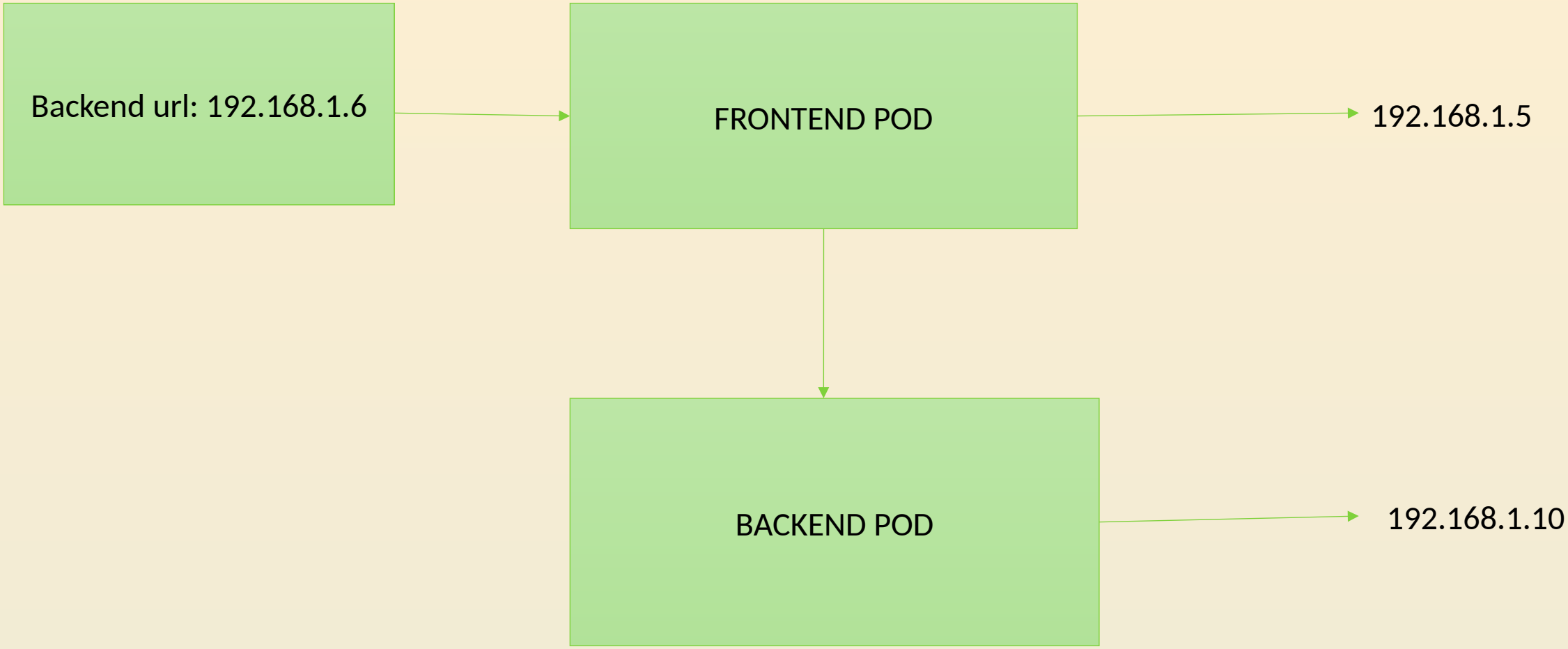
When you create a pod each pod has an unique private ip address



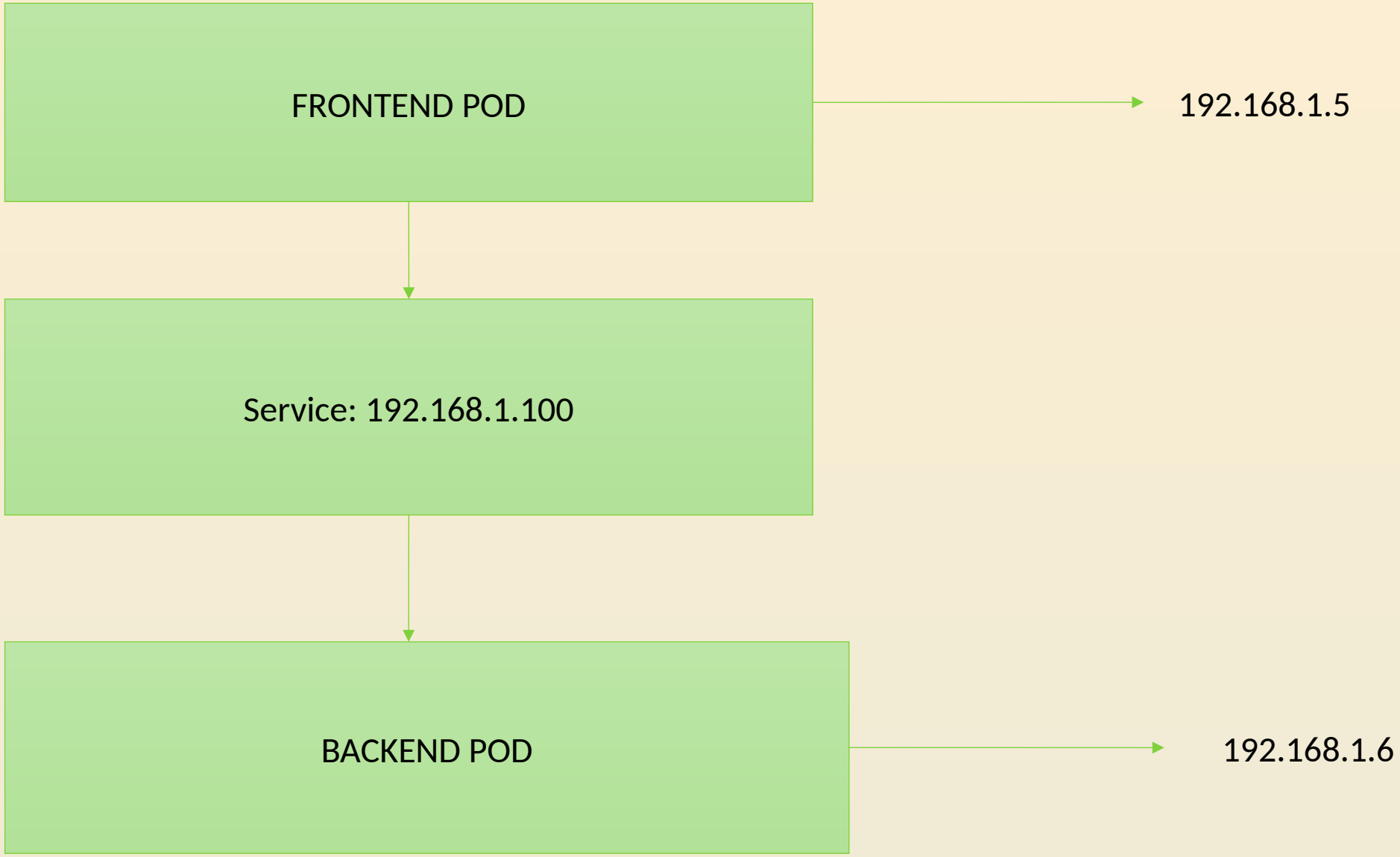
Pods are Ephermel, pods are recreated



New created pod has an different private ip address



You can create a service which has static ip and act as gateway for your frontend to talk to backend pod



Types of Services in Kubernetes

ClusterIP

NodePort

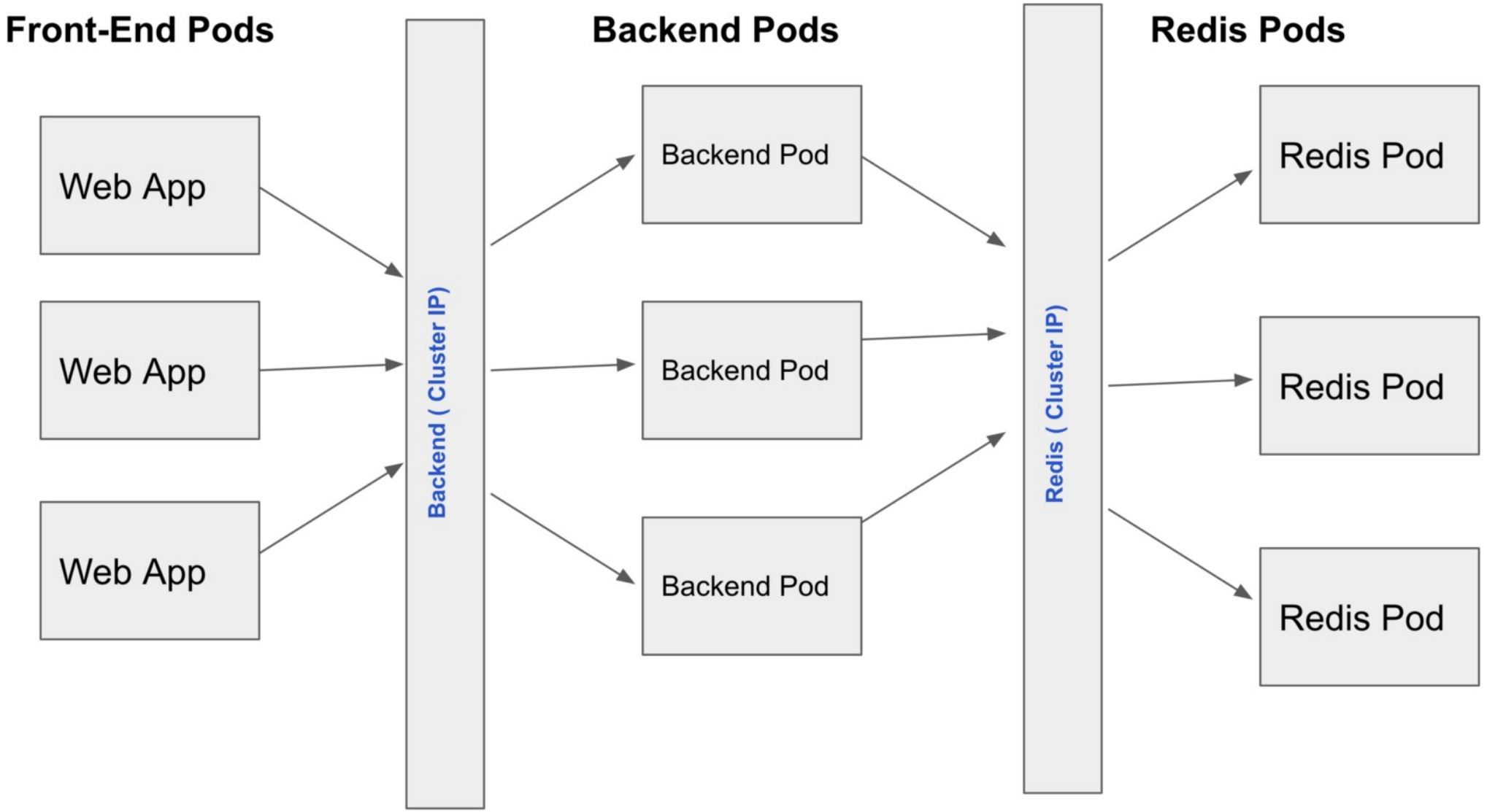
LoadBalancer

ClusterIP

Whenever the service type is ClusterIP, an internal ip of the cluster is assigned to the service.

Since an internal ip is assigned, it can only be reachable within the cluster.

This is the default service type.

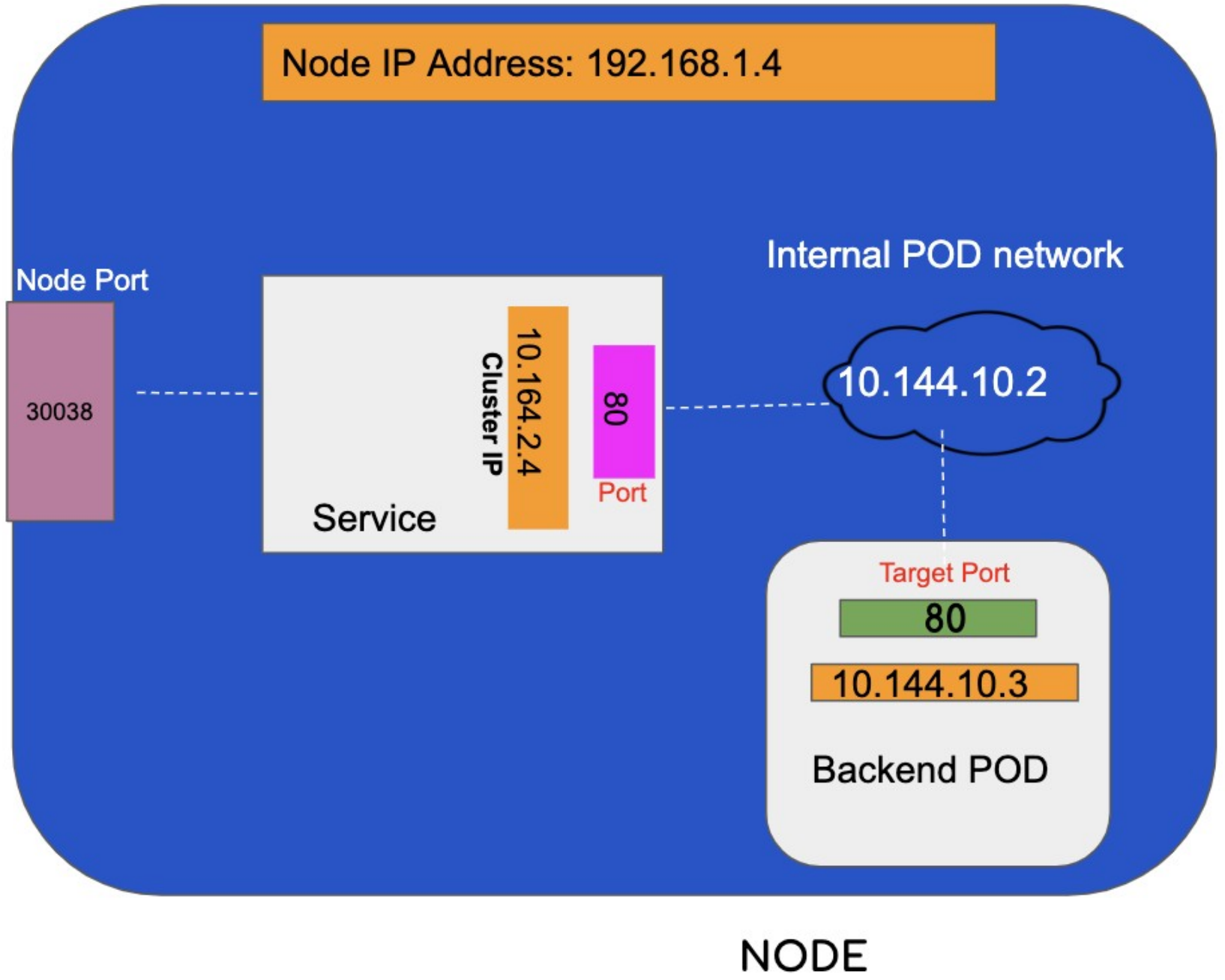

Front-End Pods
Backend Pods
Redis Pods
Web App
Web App
Web App
Backend (Cluster IP)
Backend Pod
Backend Pod
Backend Pod
Redis (Cluster IP)
Redis Pod
Redis Pod
Redis Pod

NODEPORT

Nodeport service opens the port on the nodes.

If service type is Nodeport, then kubernetes will allocate a port (default- 30000-32767) on all the nodes in the cluster.

Each node will proxy that port into your service.


Node IP Address: 192.168.1.4
Node Port
30038
Service
Cluster IP
10.164.2.4
80
Port
Internal POD network
10.144.10.2
Target Port
80
10.144.10.3
Backend POD
NODE

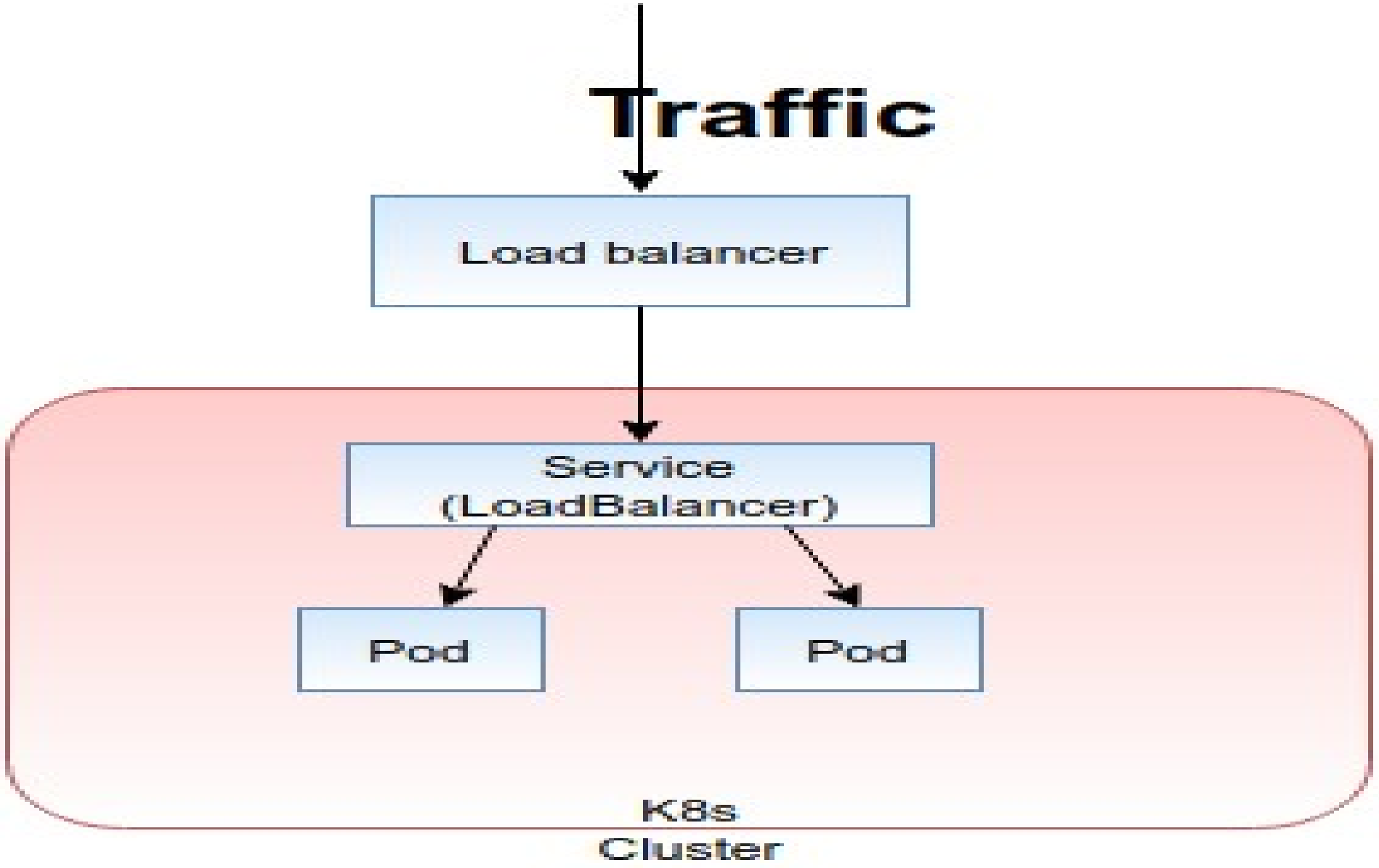
LOADBALANCER

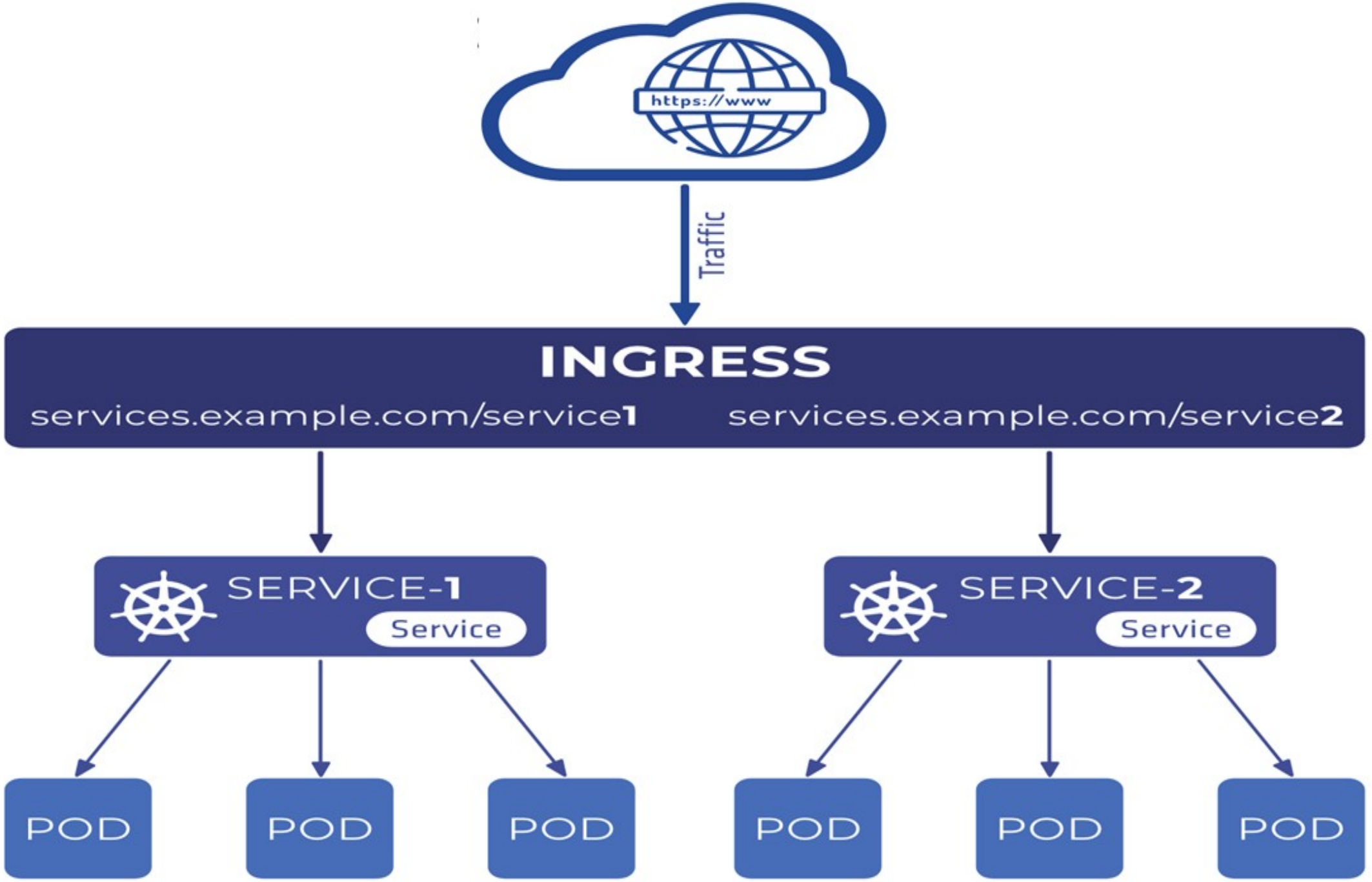
Loadbalancer service type automatically deploys an external loadbalancer for your service.

This loadbalancer service takes care of routing the requests to the underlying NodePort exposed.

Implementation of Load balancer depends upon your cloud service provider.

If using loadbalancer as a service on bare-metal, you have to provide own implementation of loadbalancer.


Traffic
Load balancer
Service
(LoadBalancer)
Pod
Pod
K8s
Cluster


https://www
Traffic
INGRESS
services.example.com/service1
services.example.com/service2
SERVICE-1
Service
SERVICE-2
Service
POD
POD
POD
POD
POD
POD

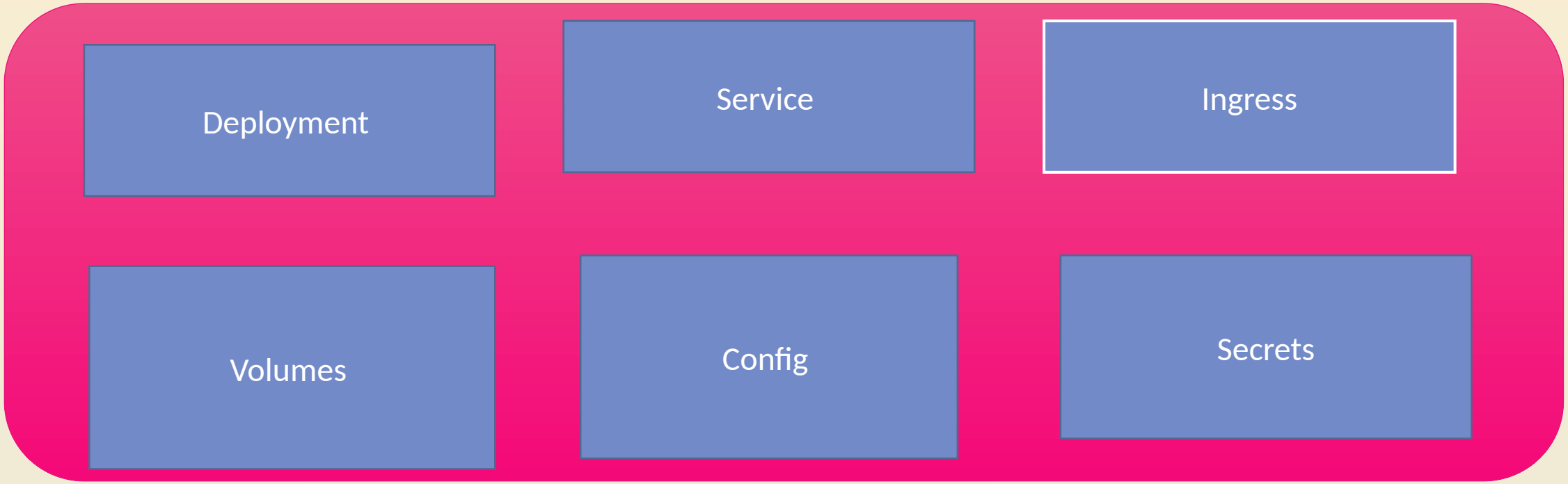
HELM

Helm is a package manager for Kubernetes.

Helm is the K8s equivalent of yum or apt.

Helm deploys charts, which you can think of as a packaged application.

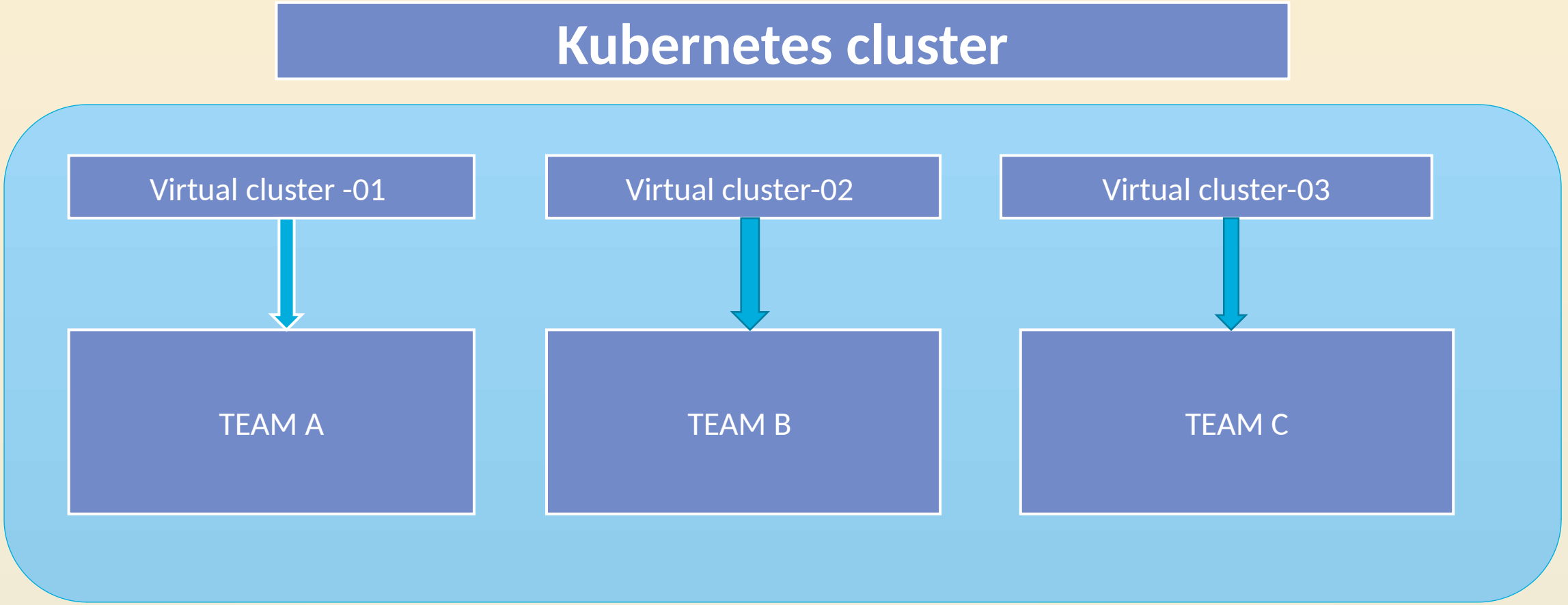
It is a collection of all your versioned, pre-configured application resources which can be deployed as one unit.



KUBERNETES NAMESPACE

Kubernetes supports multiple virtual clusters for the same physical cluster.

These virtual clusters are called as Namespace.



THANK YOU