Kubernetes

HOMESOURCING

Why?

Problems:

- The need to scale up/down quickly in response to the workload
- Combining multiple services into one cluster
- Continuous integration and deployment
- High availability and Fault tolerance

HOMESOURCING

How?

Potential solutions:

- Manually handle everything
- Implement your own system :)
- Use an existing one

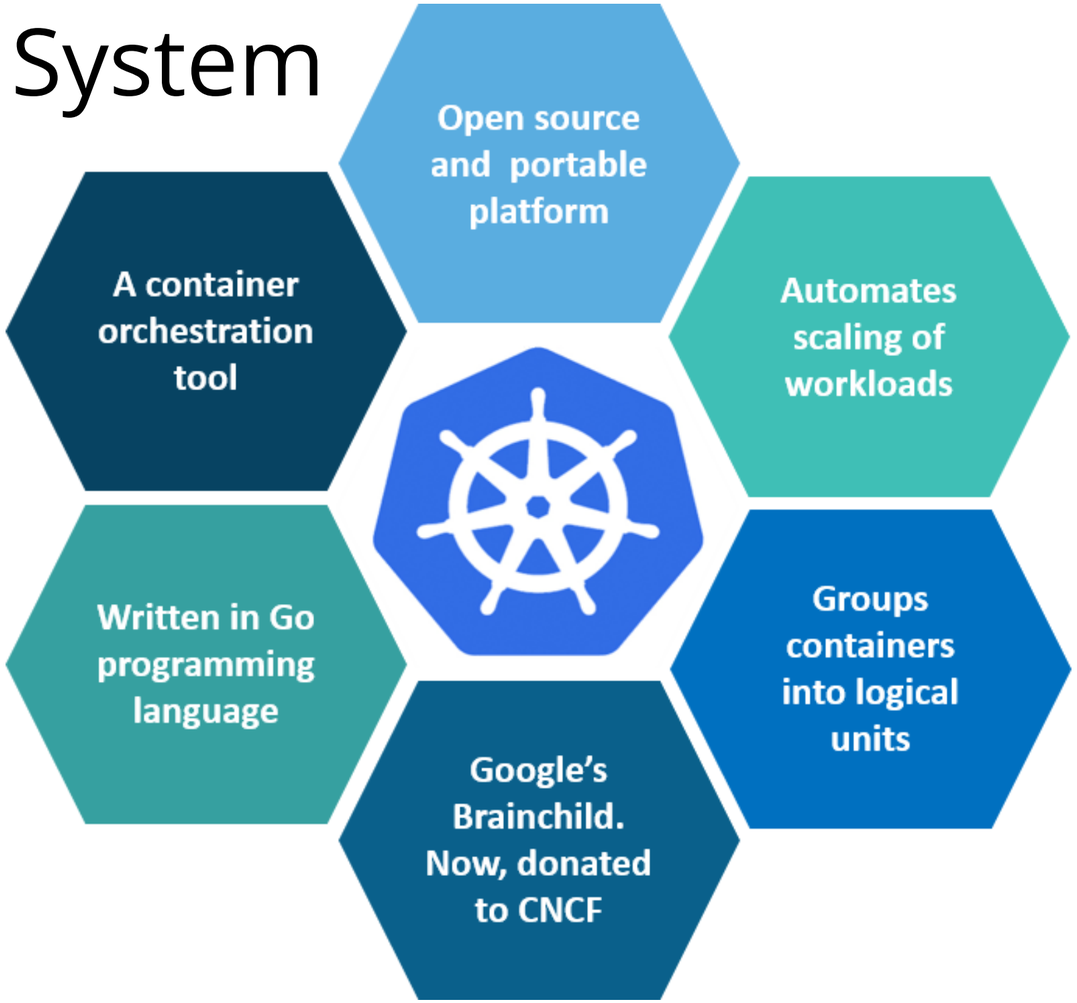
HOMESOURCING

What?

Container Orchestration System

Providing:

- Auto Software Deployment
- Auto Scaling
- System Resilience
- Service Discovery
- System Management



HOMESOURCING

The Premise

HOMESOURCING

The Premise

The ability to enforce the *Desired State.*

"I promise that all the containers running across the cluster are always in the desired state."

-Kubernetes

HOMESOURCING

The Premise

The Desired State

- represents a state of the system that we aim to achieve.

- should be retained at all times, even in case a member of a cluster fails.

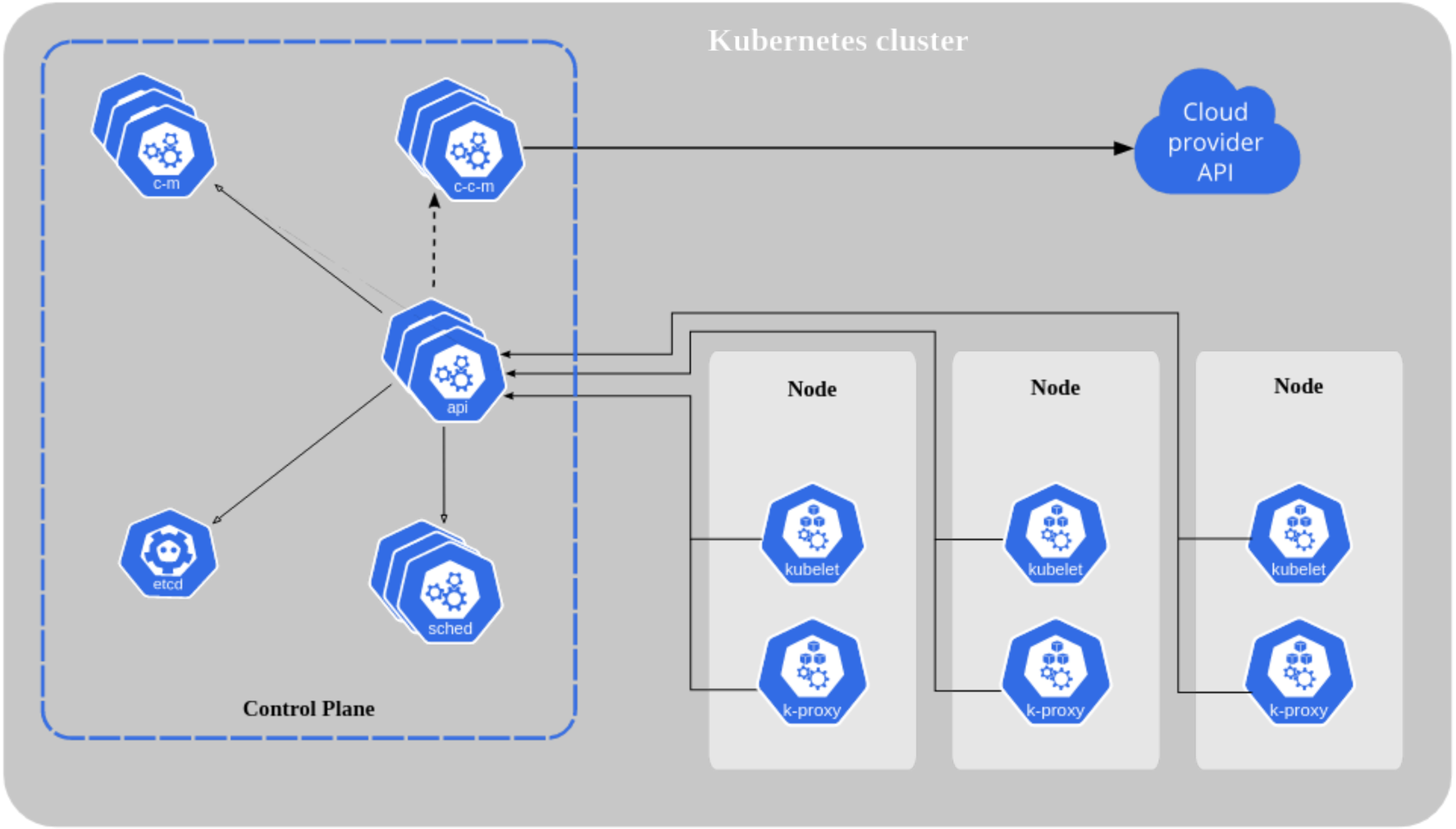
- can be defined through a declarative or imperative API.

HOMESOURCING

The Architecture

HOMESOURCING

The Architecture



API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

kubelet

kube-proxy

Scheduler

Control plane

Node

HOMESOURCING

Distributions

HOMESOURCING

Distributions

- *Vanilla Kubernetes* - configure everything on your own
- *Managed Kubernetes* - comes pre-compiled and pre-configured

- Managed Kubernetes versions are also known as Kubernetes distributions.

HOMESOURCING

Distributions

Key Elements of a Kubernetes Distribution

- *Container Runtime*

  - helps create and manage containers on the physical/virtual machine in which they are hosted.

- *Storage*

  - offers a way to persist data, dynamically provision volumes, and manage them.

- *Networking*

  - allows for seamless communication and interaction between different containerized components.

HOMESOURCING

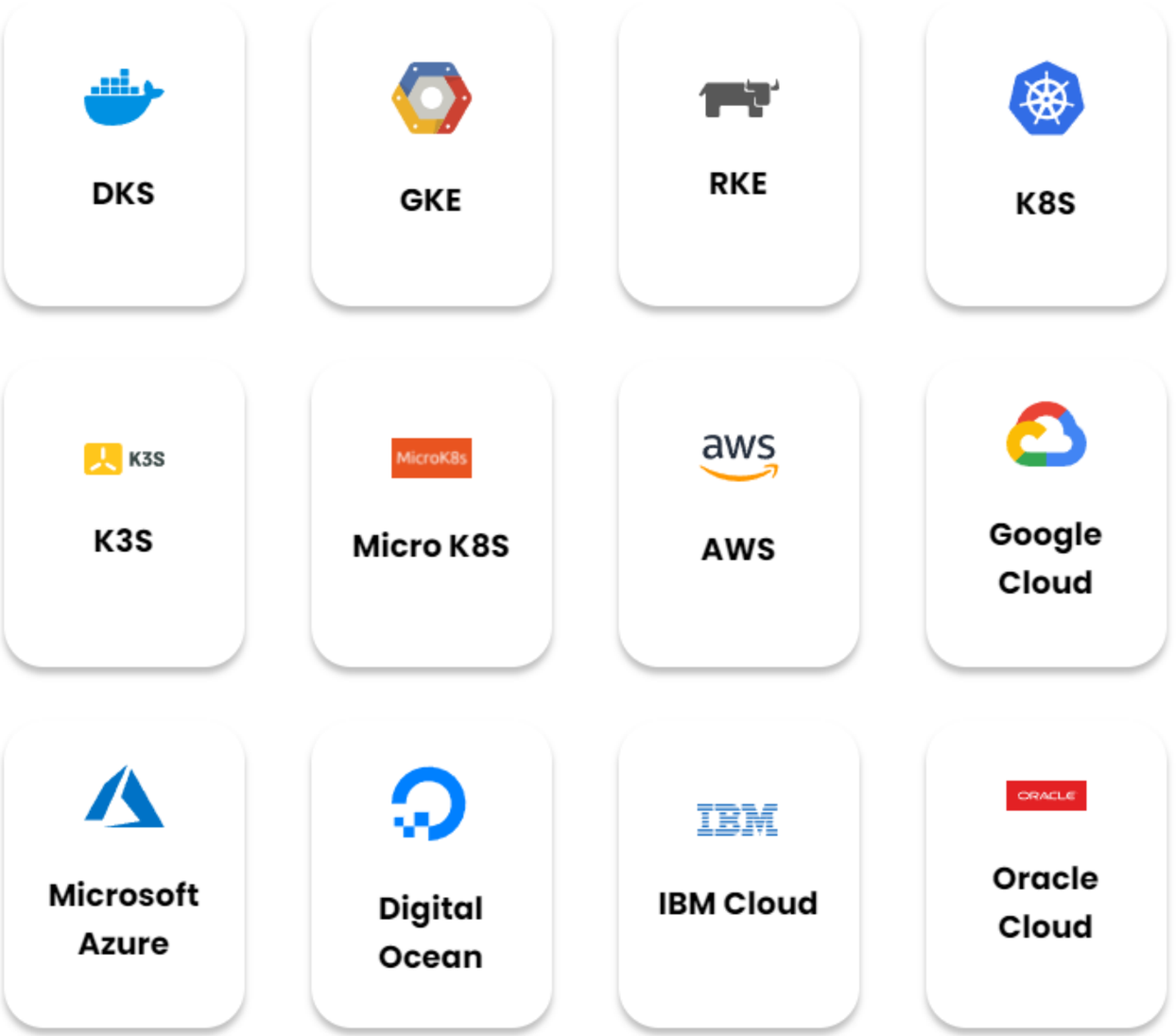
Distributions

Key Elements of a Kubernetes Distribution

- *Container Runtime*
  - Docker, Apache Mesos, CoreOS, rkt, Canonical LXC, containerd...
- *Storage*
  - EBS, GlusterFS, Portworx, Rook, OpenEBS, Longhorn...
- *Networking*
  - Flannel, Weave Net, Calico, Canal...

HOMESOURCING

Distributions

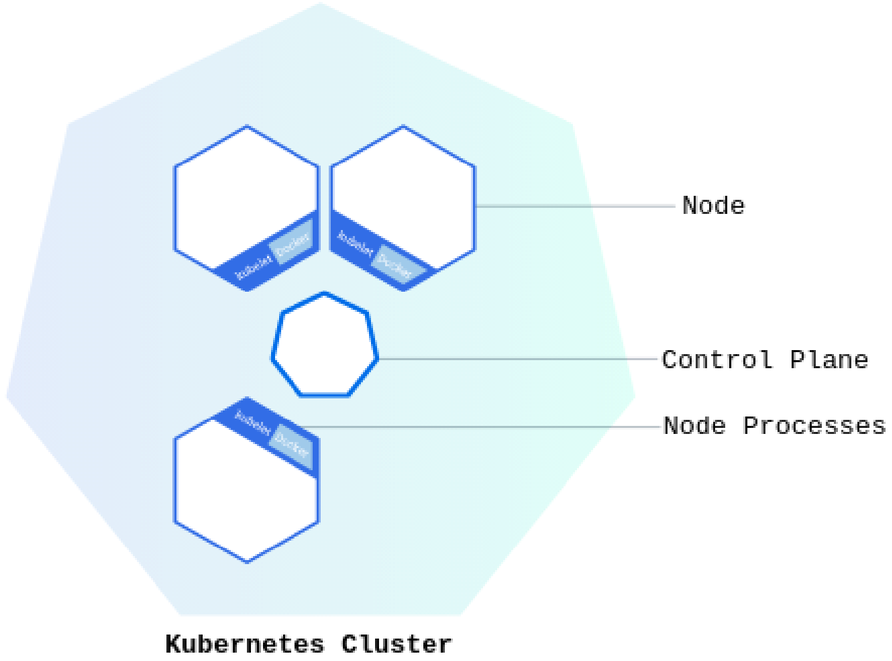


HOMESOURCING

Core Concepts

HOMESOURCING

Core Concepts

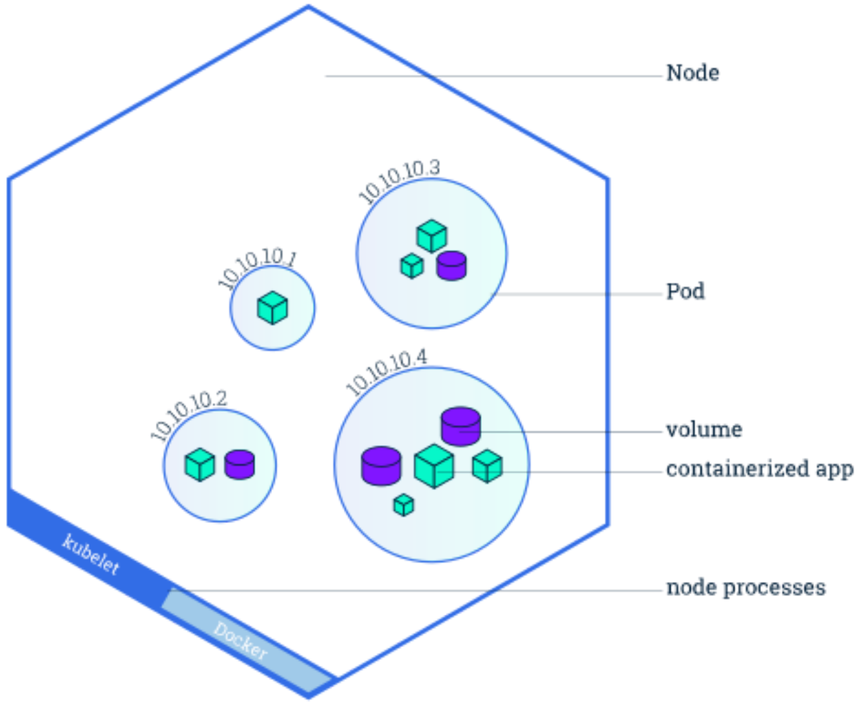


Control Plane - is responsible for managing the cluster.

Node - a VM or a physical computer that serves as a worker machine in a Kubernetes cluster.

HOMESOURCING

Core Concepts

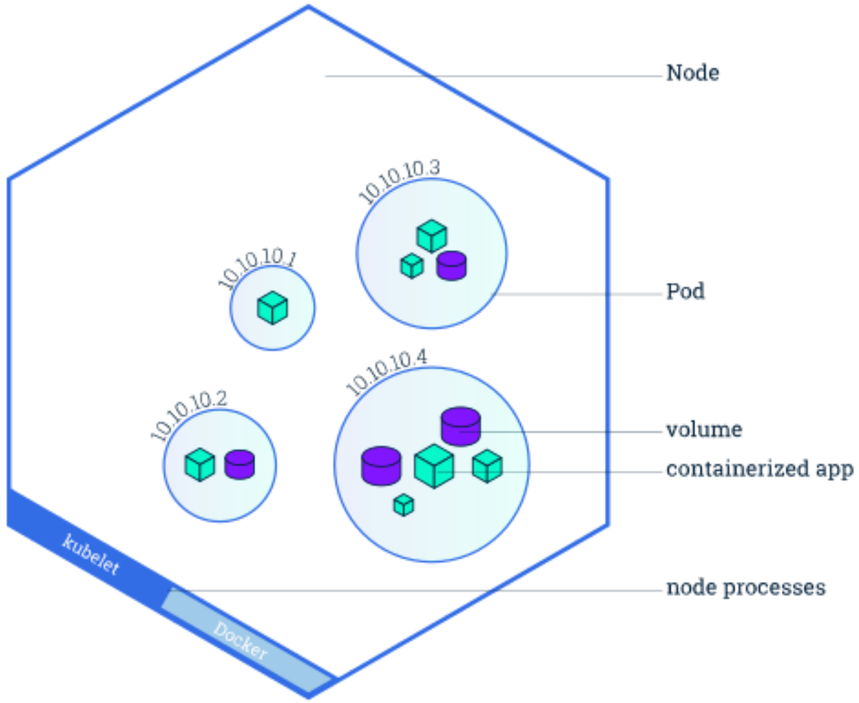


Pod - models an application-specific "logical host" and can contain different application containers which are relatively tightly coupled.

Volume - a directory, possibly with some data in it, which is accessible to the containers in a pod.

HOMESOURCING

Core Concepts



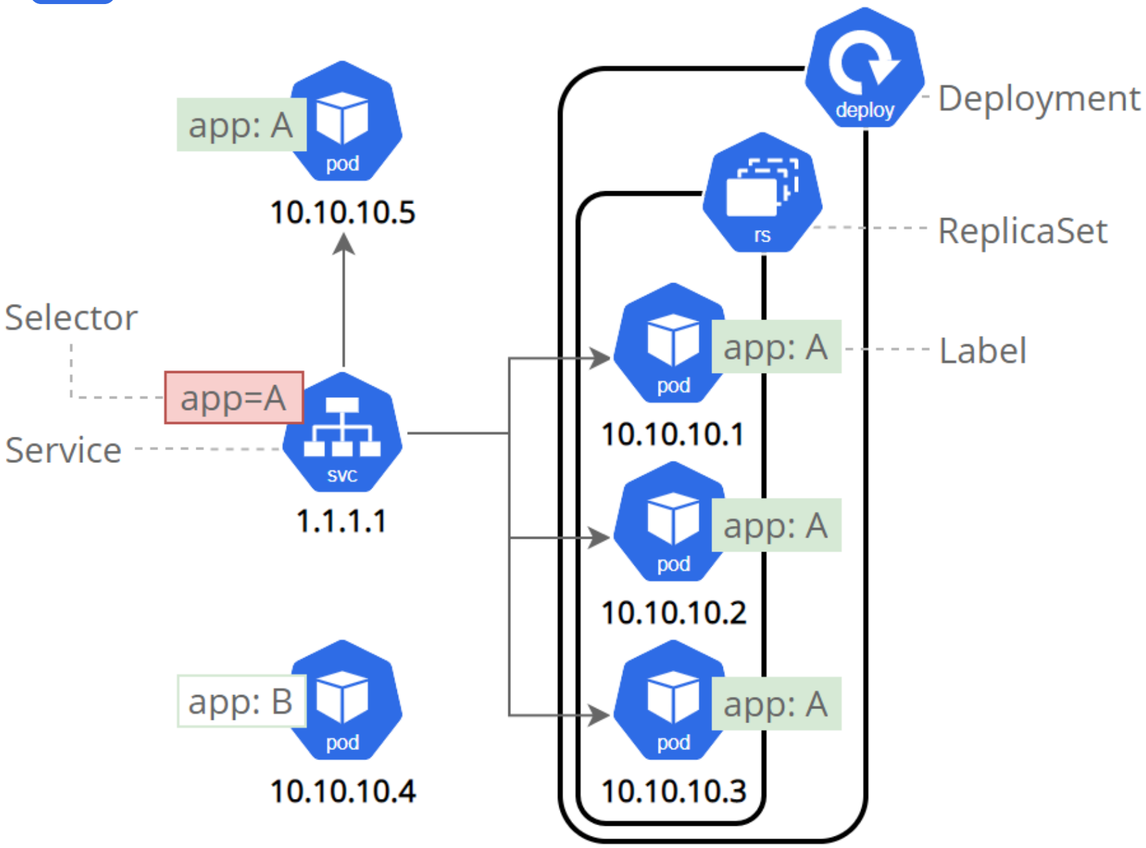
Ephemeral Volume - a volume that has the lifetime of a pod.

Persistent Volume - a volume that exists beyond the lifetime of a pod.
It can be provisioned *statically* or *dynamically* (using the *PVC*).

Persistent Volume Claim - a request for the resources that *PV* needs, and also acts as a claim check to the resource.

HOMESOURCING

Core Concepts



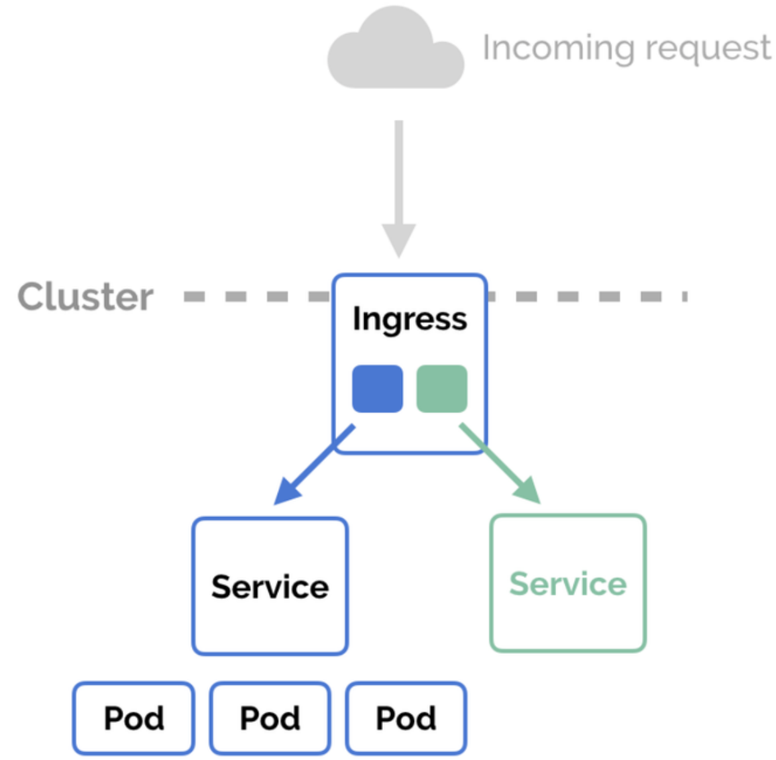
Service - an abstraction that defines a logical set of Pods and a policy by which to access them.
It can be of type *ClusterIP* (default), *LoadBalancer*, *NodePort* and *ExternalName*.

Deployment - instructs Kubernetes on how to create and update instances of your application.

ReplicaSet - maintains a stable set of replica Pods running at any given time.

HOMESOURCING

Core Concepts



Ingress - an API object that manages external access to the services in a cluster and exposes HTTP and HTTPS routes from outside the cluster to services within the cluster.

HOMESOURCING

Creating a Cluster

HOMESOURCING

Creating a Cluster

K8S

- *Install kubeadm*
- *Init control plane node* - **kubeadm init <args>**
- *Join a cluster* -

kubeadm join <control-plane-host> --token <token> --discovery-token-ca-cert-hash sha256:<hash>

HOMESOURCING

Creating a Cluster

micro8s

- *Install microk8s* - **snap install microk8s --classic**
- *Optionally check status* - **microk8s status --wait-ready**
- *Join a cluster* - **microk8s join <cluster-host>/<hash> --worker**

HOMESOURCING

Creating a Cluster

minikube

- *Install minikube*
- *Start the cluster* - **minikube start**

HOMESOURCING

Creating a Cluster

K3S

- *Run -* **curl -sfL https://get.k3s.io | sh -**
- **:)**
- *Join a cluster -*

k3s agent --server <server-url> --token ${NODE_TOKEN}

HOMESOURCING

Interacting with a Cluster

HOMESOURCING

Interacting with a Cluster

List nodes in a cluster:

- **kubectl get node**

List resources in a cluster:

- **kubectl get <type>**
- **kubectl get pod**
- **kubectl get service -n custom-namespace**
- **kubectl get deployment -A**

Get a specific resource:

- **kubectl get <type>/<resouce-name>**
- **kubectl get pod/test-01hss -o yaml**

Edit a specific resource:

- **kubectl edit <type>/<resouce-name>**

HOMESOURCING

Interacting with a Cluster

Describe a specific resource:

- **kubectl describe <type>/<resouce-name>**
- **kubectl describe service/nginx-svc**

Get logs of a pod:

- **kubectl logs pod test-01hss**
- **kubectl logs deployment test**

Forward a port to the cluster:

- **kubectl port-forward <type>/<r-name> <lport>:<tport>**

Scale deployment:

- **kubectl autoscale deployment <r-name> --min=2 --max=10**
- **kubectl scale --replicas=3 deployment <r-name>**

HOMESOURCING

Interacting with a Cluster

Get resource consumption:

- **kubectl top node**
- **kubectl top pod**

Execute a command in a pod:

- **kubectl exec -it <pod-name> -- <command>**

Apply yaml file:

- **kubectl apply -f <file-path>**

CRUD:

- **kubectl create <type> <args>**
- **kubectl get <type> <args>**
- **kubectl patch <type> <args>**
- **kubectl delete <type> <args>**

HOMESOURCING

Deploying an App

HOMESOURCING

Deploying an App

Imperative

Start a pod:

- **kubectl run nginx --image=nginx -n test**

Expose a pod:

- **kubectl expose deployment nginx --port=80 --target-port=80**

Declarative

Define desired state in yaml files

Apply:

- **kubectl apply -f <file-path>**

HOMESOURCING

What's next?

HOMESOURCING

What's next?

- Explore Kubernetes resources
  (ConfigMaps, Secrets, Certificates, StatefulSets, Jobs...)
- Use Helm - Kubernetes package manager
- Learn Kubernetes YAML file syntax
- Advanced zero-downtime deployments
- Play around :)

HOMESOURCING

Thank you!

HOMESOURCING