kubernetes

Why kubernetes ?

Container Clusters

- What if we have 10s, 100s, 1000s of running containers on multiple VMs?
- How to deploy, scale, restart, manage all of these containers?
- What problems do they solve?
    - Management
        - Metrics
        - Health checks
        - Security
    - Abstraction of hardware
    - Networking
    - Scheduling
    - Scaling
    - Deployment
        - Rollbacks
        - Zero-downtime / blue-green
    - Service discovery

A Brief Kubernetes History

- “K8s”
- Evolved out of Borg (Google’s internal container cluster)
- Open sourced ~2014
- Grew in popularity, open source velocity increased
- Now the most popular container cluster (most cloud platforms have some sort of managed K8s offering)
- Features added regularly and frequently
- Cloud Native / CNCF - Kubernetes, Prometheus, Fluentd

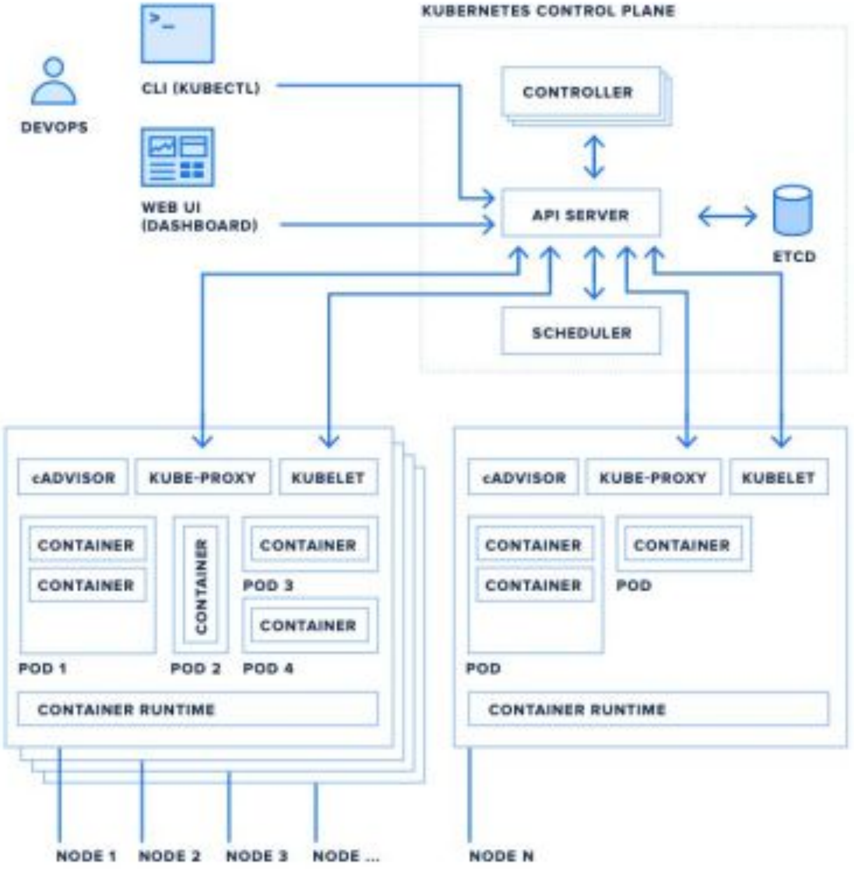
Kubernetes Architecture

- Client side :- CLI(KUBECTL)

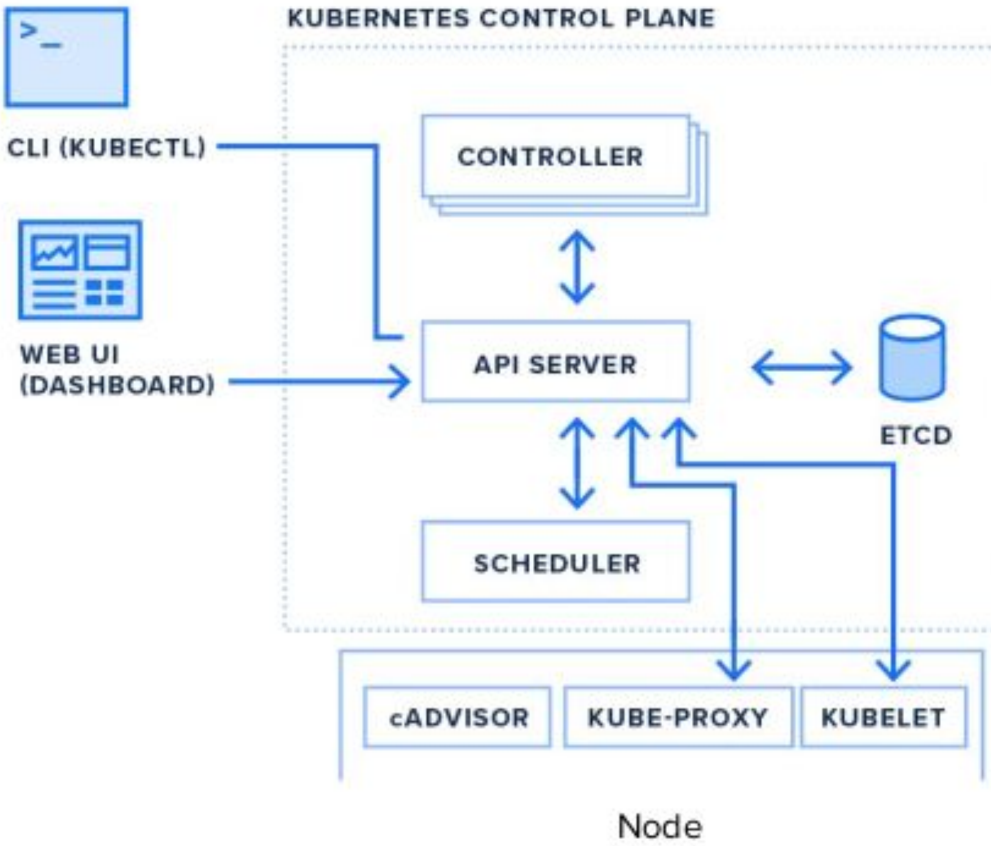
  WEB UI(DASHBOARD)

- Kubernetes Control plane or master node

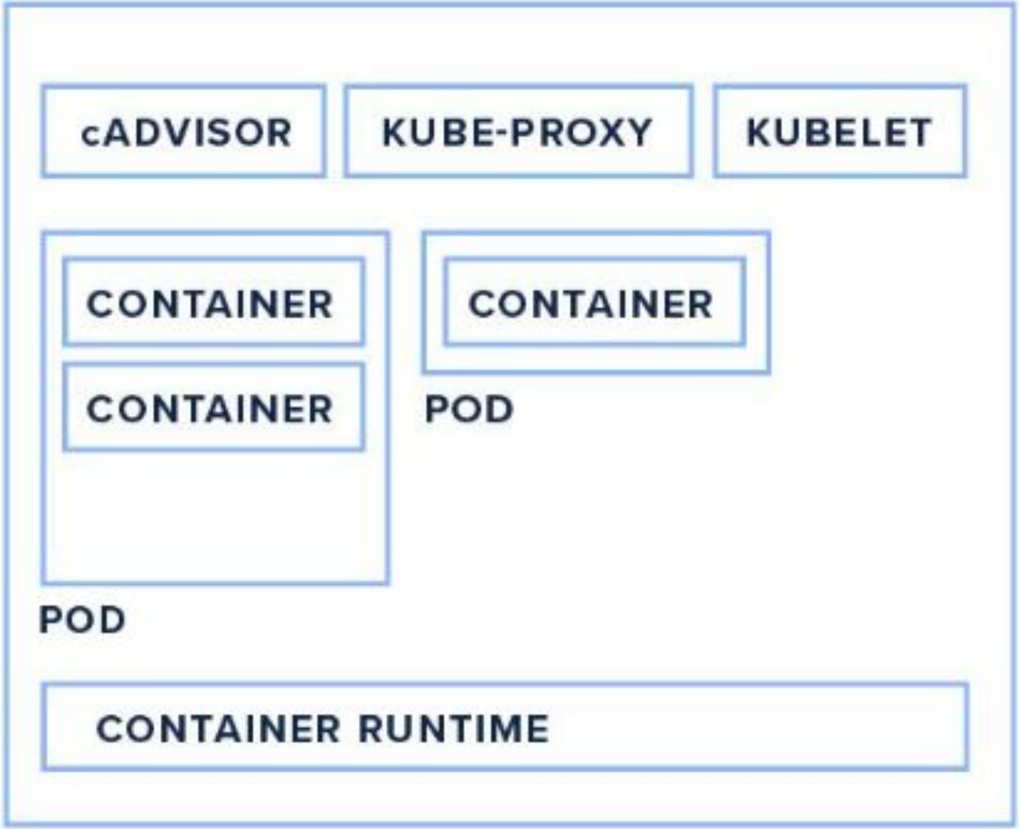
- Minions node or worker node



Kubernetes control plane



Kubernetes worker nodes



Kubernetes installation

Single Node

- Docker desktop
- Minikube

Custom kubernetes

- Kubeadm
- Kubespray

Cloud

- AWS - EKS
- Azure - AKS
- Google - GKE

Minikube installation

What you'll need

- 2 CPUs or more
- 2 GB of free memory
- 20 GB of free disk space
- Internet connection
- Container or virtual machine manager, such as: Docker, QEMU, Hyperkit, Hyper-V, KVM, Parallels, Podman, VirtualBox, or VMware Fusion/Workstation

FOR linux

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

For windows

https://storage.googleapis.com/minikube/releases/latest/minikube-installer.exe

If using powershell

```
New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force
```

```
Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri
'https://github.com/kubernetes/minikube/releases/latest/download/minikube-windows-amd64.exe'
-UseBasicParsing
```

FOR mac

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-darwin-amd64
```

```
sudo install minikube-darwin-amd64 /usr/local/bin/minikube
```

Some K8s commands

Minikube start

Minikube stop

Kubectl version

Kubectl get

Kubectl apply

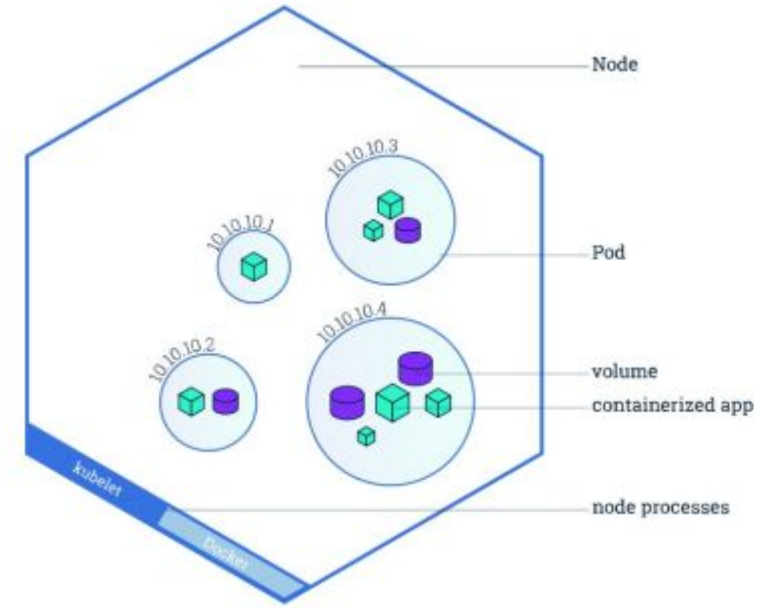
Kubectl create

Kubectl delete

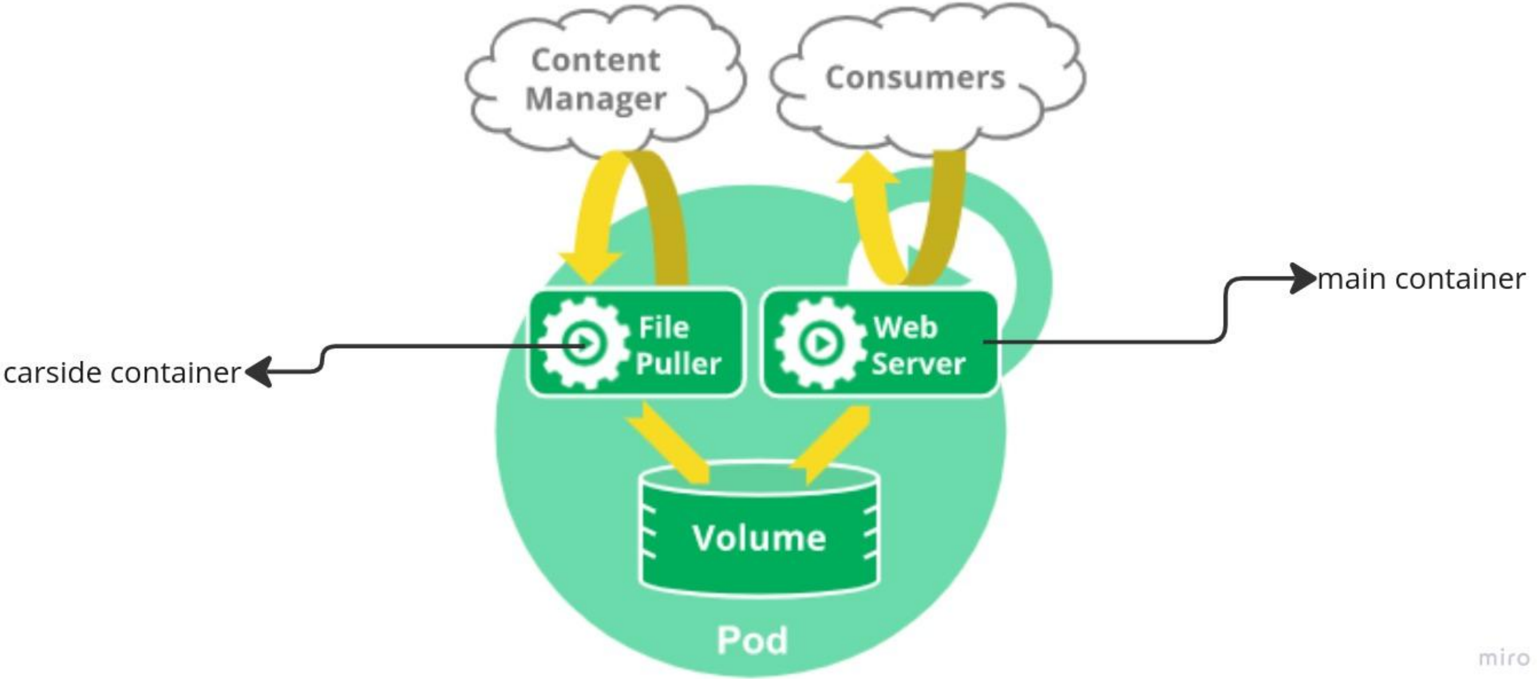
K8s components

PODS

- **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.
- **Pods** that run multiple containers that need to work together.
- **Pod** containers share resources
    - Storage
    - Network (localhost)
    - Always run on the same Node

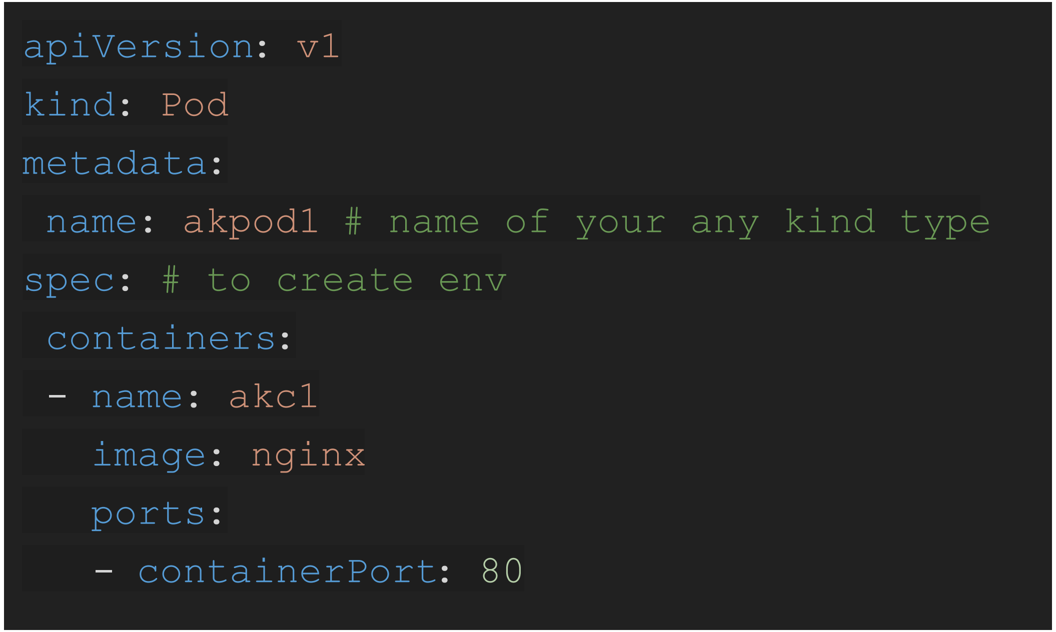


Multiple containers in single pod



Create a container

Create a yaml file eg :- ak.yaml



```yaml
apiVersion: v1
kind: Pod
metadata:
 name: akpod1 # name of your any kind type
spec: # to create env
 containers:
 - name: akc1
   image: nginx
   ports:
   - containerPort: 80
```

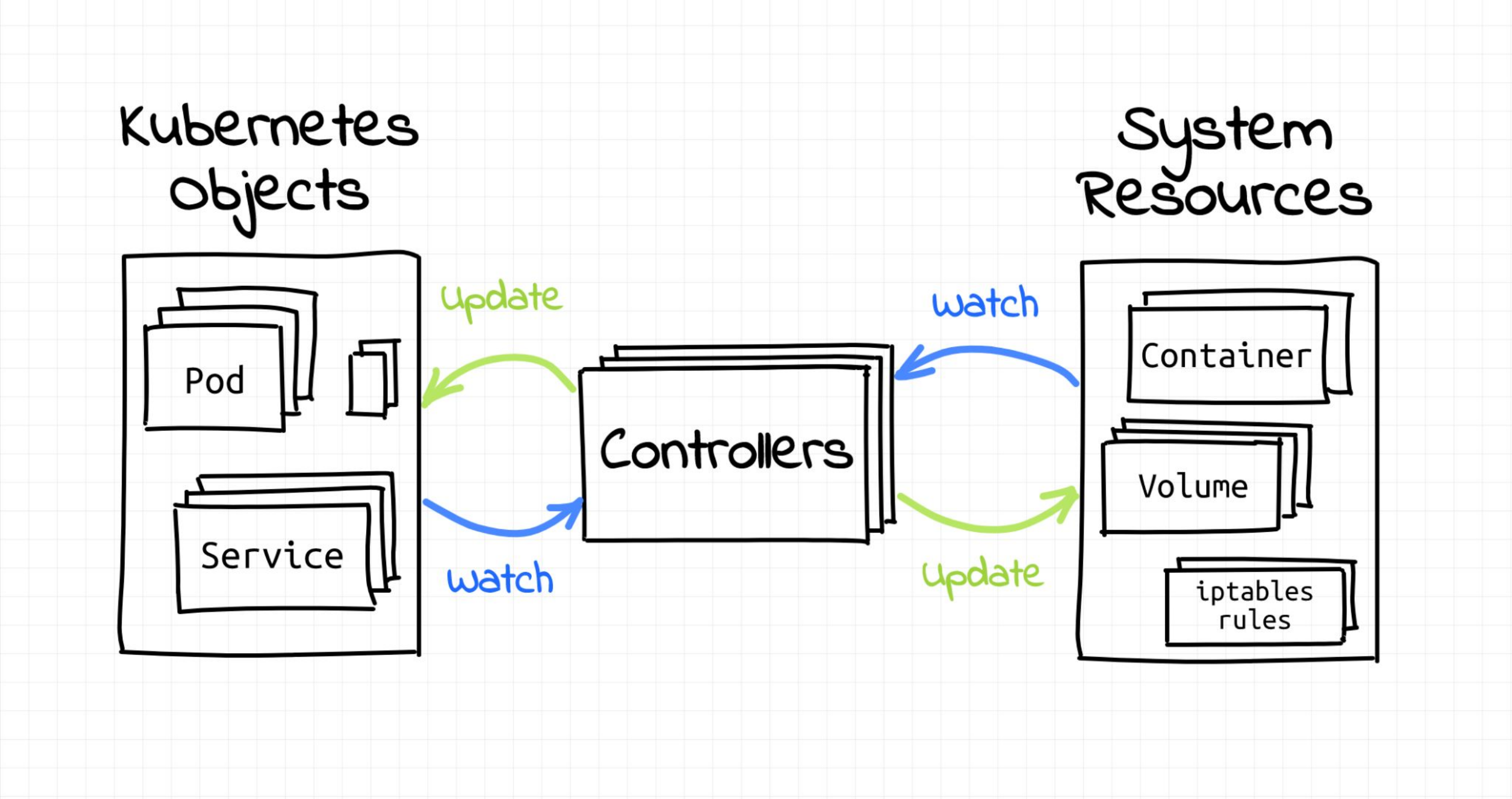
POD problems

1) Recrete [auto]

2) Scale(pod)

3) perhaps several Pods, to carry out a task and then stop.

Controller


Kubernetes objects
Pod
Service
Update
watch
Controllers
watch
Update
System Resources
Container
Volume
iptables rules

K8s native Controller

Replication controller[RC]:-A ReplicationController ensures that a specified number of pod replicas are running at any one time. In other words, a ReplicationController makes sure that a pod or a homogeneous set of pods is always up and available.

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

ReplicationController

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
 name: ashu-rc1
spec:
 replicas: 1 # number of pods
 template: # pod yaml info
   metadata:
     labels:
       x1: akash
   spec: # to create env
     containers:
     - name: ashuc1
       image: nginx
       ports:
       - containerPort: 80
```

Kubernetes workload according to apps

For Stateless app:- (eg :- Webapp)

- Deployments
    - ReplicaSets
        - Pods
            - Container

For stateful app:- (eg : Databases)

- StatefulSets

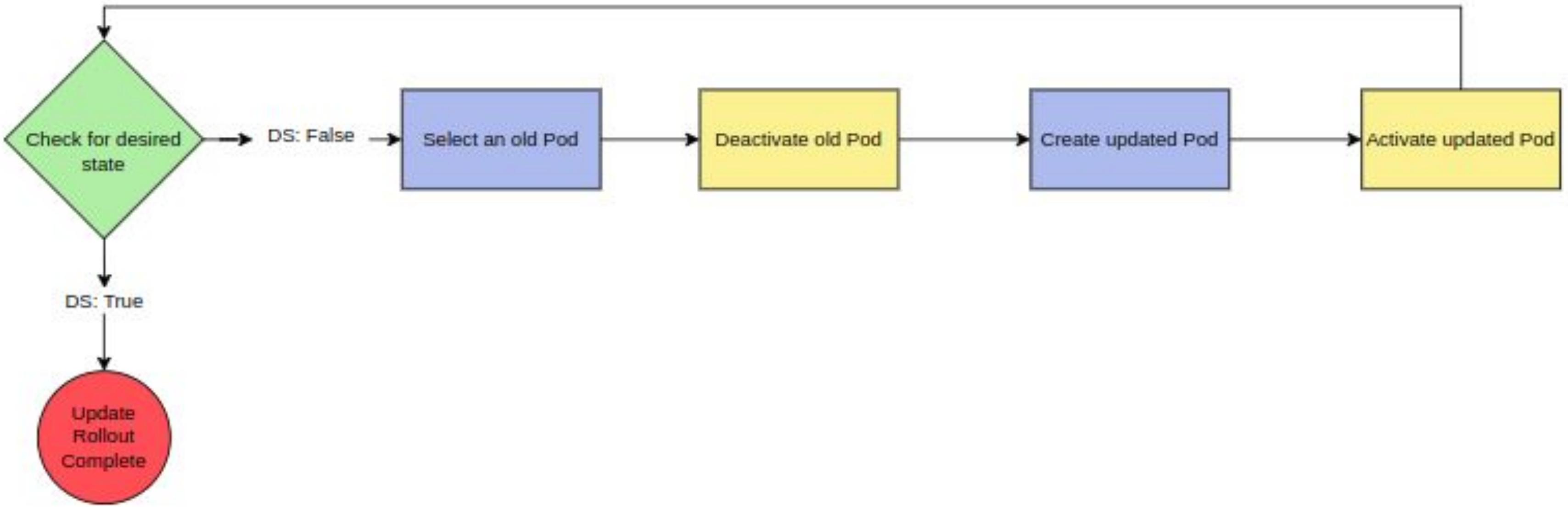
Deployments

A Deployment provides declarative updates for Pods and ReplicaSets.

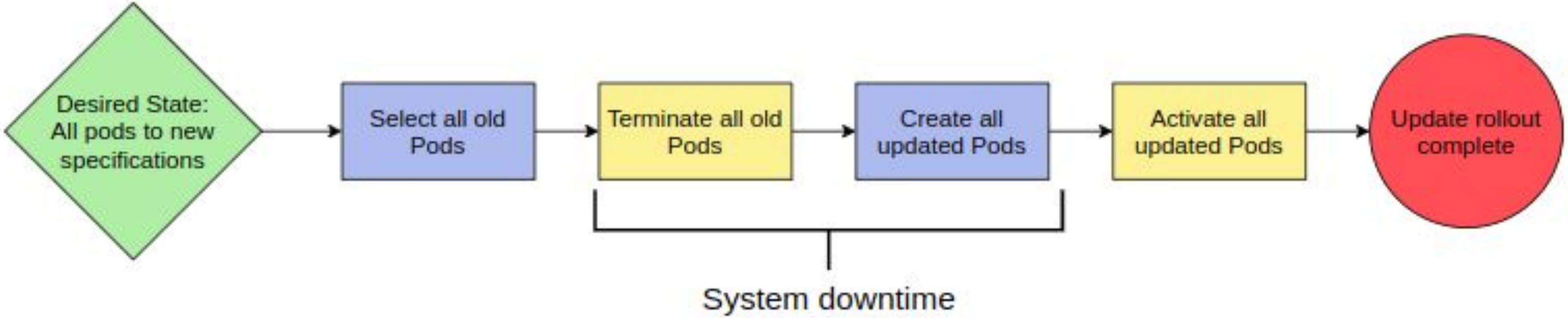
Update Deployment Strategies

- **Rolling update strategy**: Minimizes downtime at the cost of update speed.
- **Recreation Strategy**: Causes downtime but updates quickly.
- **Canary Strategy**: Quickly updates for a select few users with a full rollout later.

Rolling update strategy

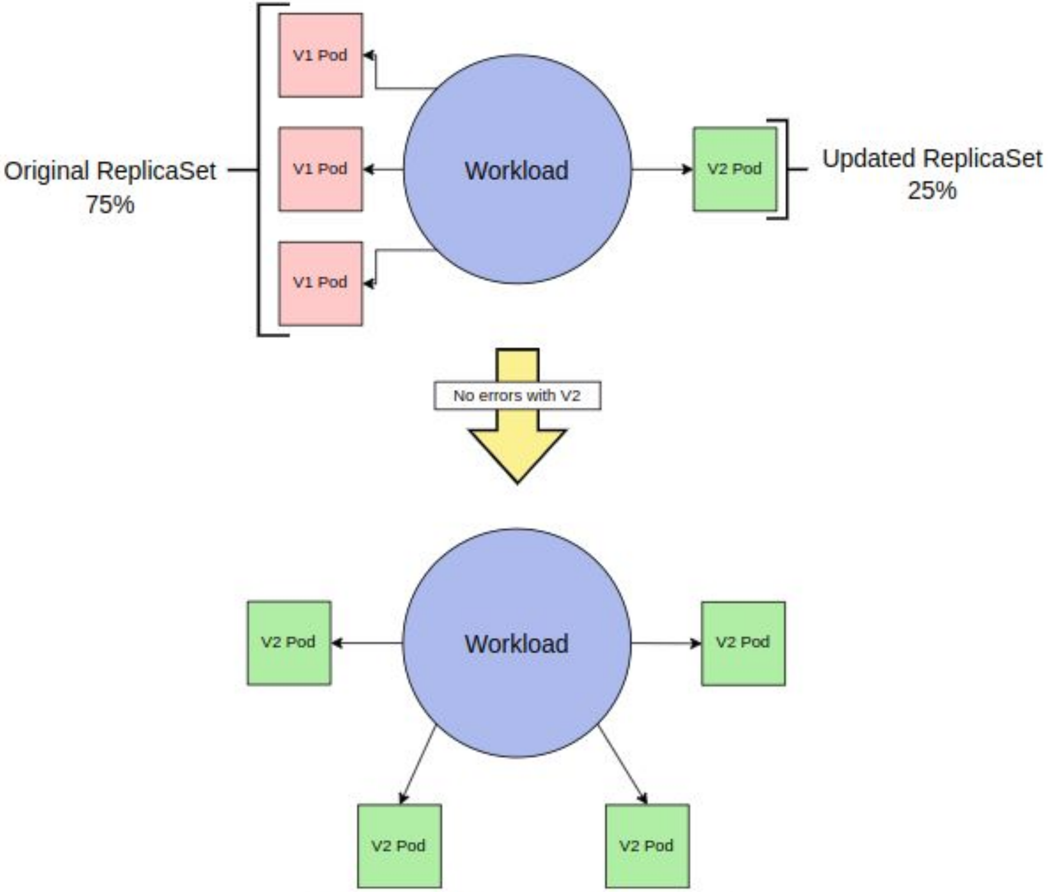


Recreation Strategy



Recreate update strategy flowchart

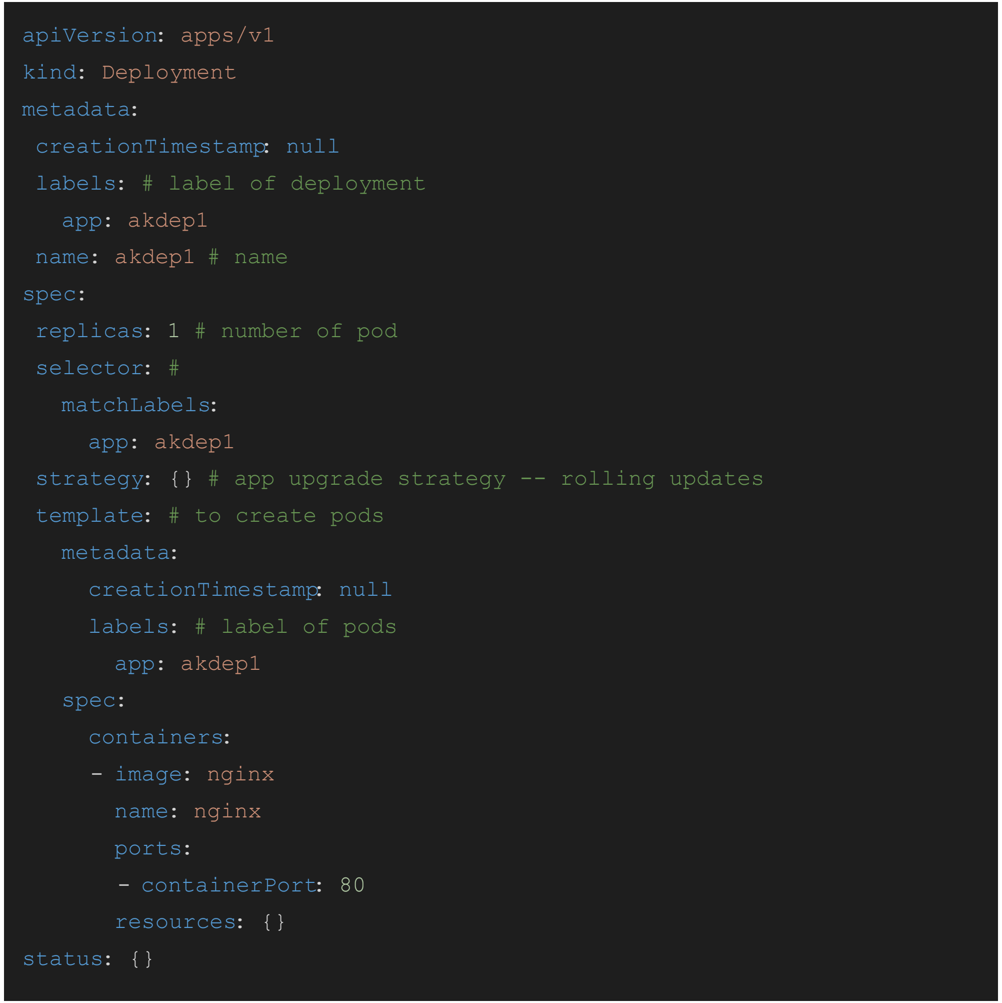
Canary Strategy



Both steps of the canary update strategy

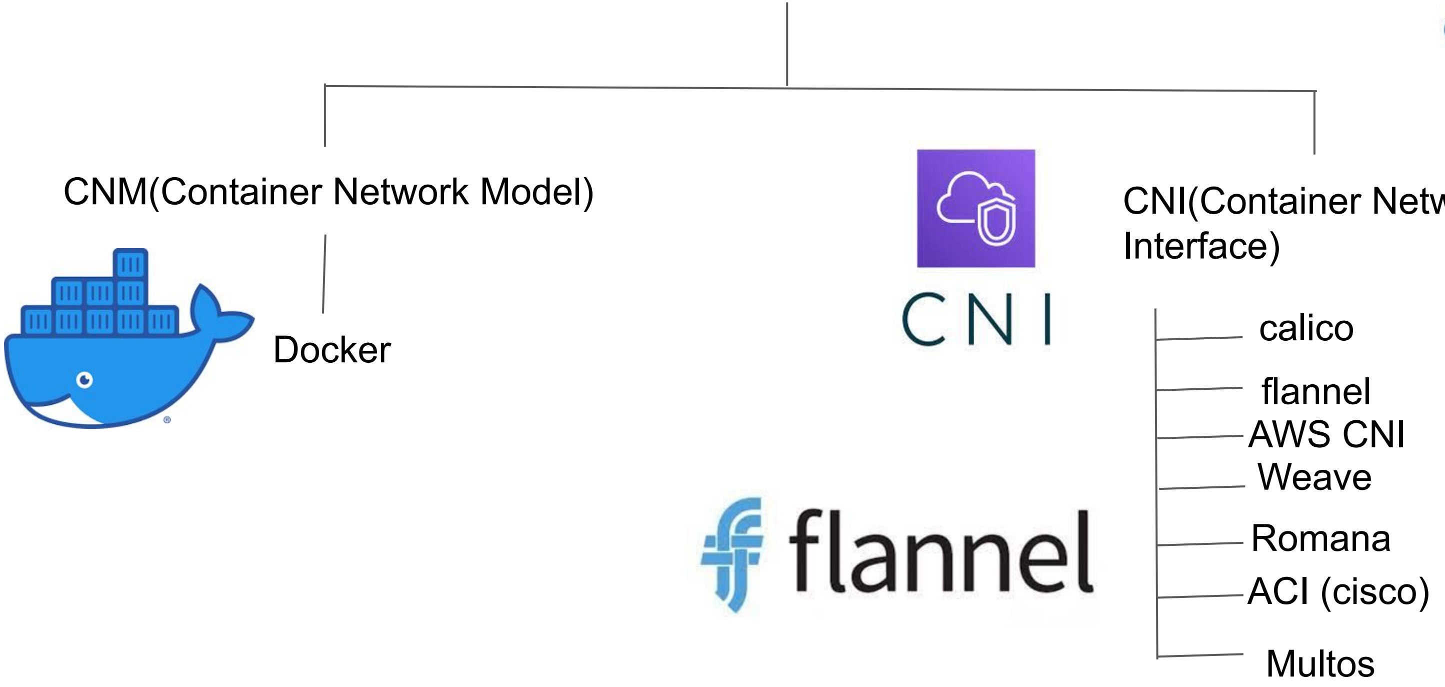
Deployments

kubectl create deployment akdep1 --image=nginx --port 80 --dry-run=client -o yaml >deployment.yaml



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 creationTimestamp: null
 labels: # label of deployment
   app: akdep1
 name: akdep1 # name
spec:
 replicas: 1 # number of pod
 selector: #
   matchLabels:
     app: akdep1
 strategy: {} # app upgrade strategy -- rolling updates
 template: # to create pods
   metadata:
     creationTimestamp: null
     labels: # label of pods
       app: akdep1
   spec:
     containers:
     - image: nginx
       name: nginx
       ports:
       - containerPort: 80
       resources: {}
status: {}
```

Kubernetes Networking



CNM(Container Network Model)

Docker

CNI(Container Network Interface)

- calico
- flannel
- AWS CNI
- Weave
- Romana
- ACI (cisco)
- Multos

CNM (container network model)

Company - docker

Runtime engine - docker

CNI(container networking model interface)

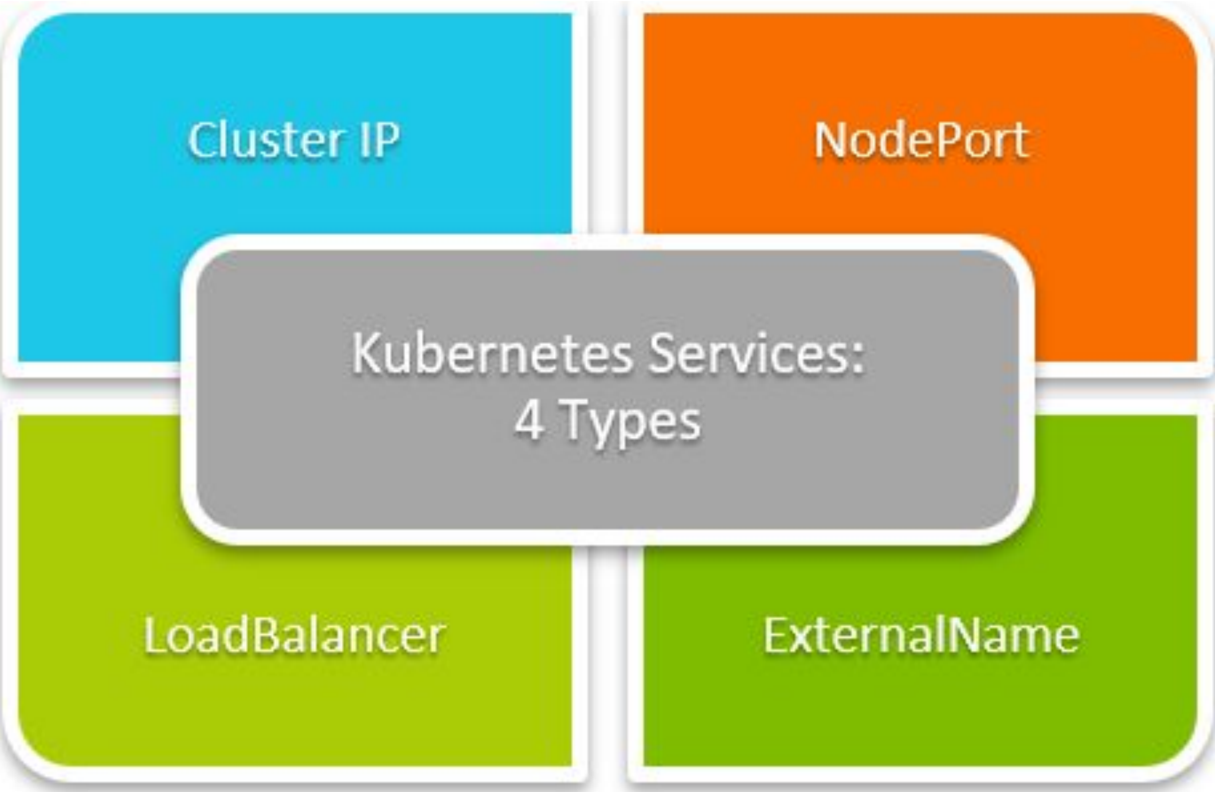
Company - CoreOS

Runtime engine - RKT

Service in k8s

Service is a method for exposing a network application that is running as one or more Pods in your cluster.

Each Service object defines a logical set of endpoints (usually these endpoints are Pods) .



Cluster IP

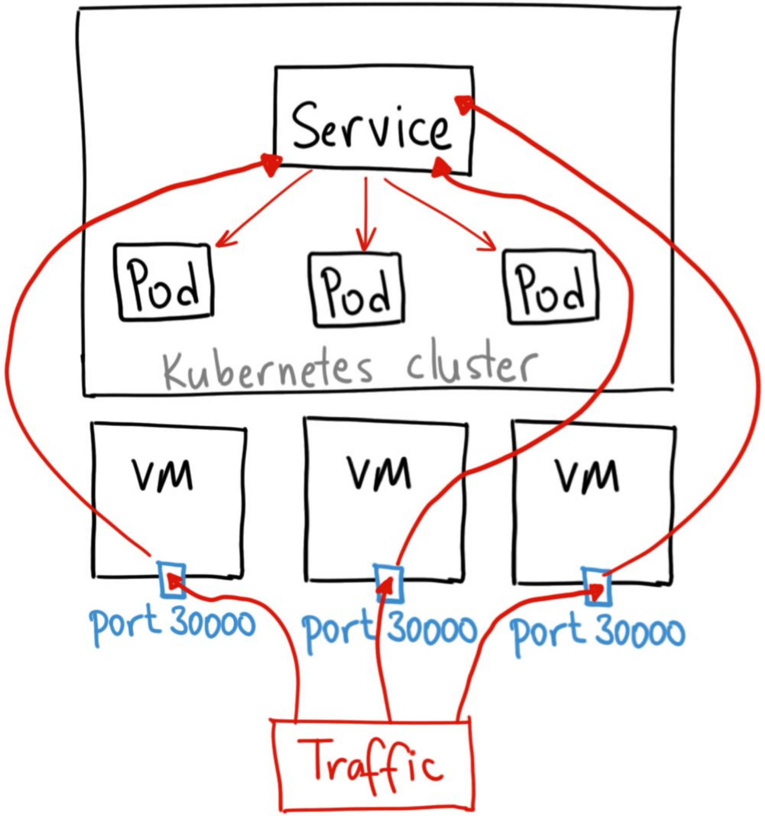
NodePort

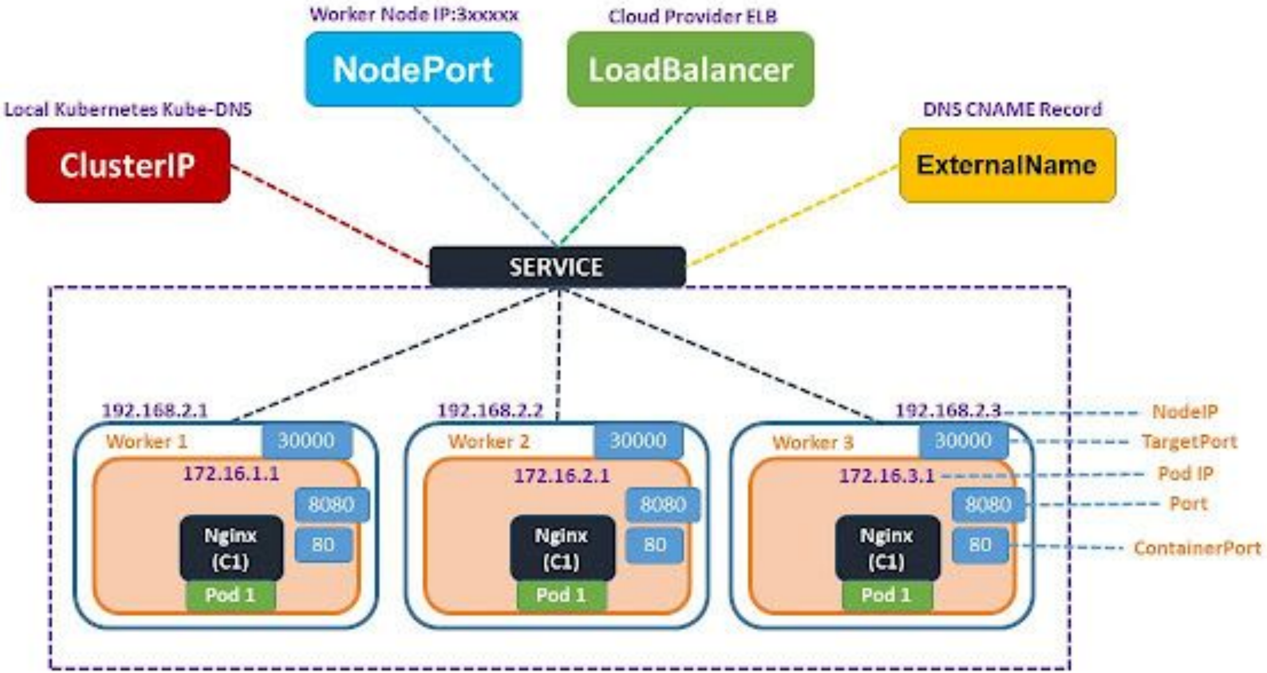
Kubernetes Services:
4 Types

LoadBalancer

ExternalName

NodePort


Service
Pod
Pod
Pod
Kubernetes cluster
VM
VM
VM
Port 30000
Port 30000
Port 30000
Traffic


Worker Node IP:3xxxxx
NodePort
Cloud Provider ELB
LoadBalancer
Local Kubernetes Kube-DNS
ClusterIP
DNS CNAME Record
ExternalName
SERVICE
192.168.2.1
Worker 1
30000
172.16.1.1
8080
80
Nginx
(C1)
Pod 1
192.168.2.2
Worker 2
30000
172.16.2.1
8080
80
Nginx
(C1)
Pod 1
192.168.2.3
Worker 3
30000
172.16.3.1
8080
80
Nginx
(C1)
Pod 1
NodeIP
TargetPort
Pod IP
Port
ContainerPort

K8s secrets

A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key. Such information might otherwise be put in a Pod specification or in a container image.

K8s Persistent volume

Reserving a PersistentVolume

The control plane can bind PersistentVolumeClaims to matching PersistentVolumes in the cluster. However, if you want a PVC to bind to a specific PV, you need to pre-bind them.

By specifying a PersistentVolume in a PersistentVolumeClaim, you declare a binding between that specific PV and PVC. If the PersistentVolume exists and has not reserved PersistentVolumeClaims through its claimRef field, then the PersistentVolume and PersistentVolumeClaim will be bound.

Pvc yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: foo-pvc
  namespace: foo
spec:
  storageClassName: "" # Empty string must be explicitly set otherwise default StorageClass will be set
  volumeName: foo-pv
  ...
```

Pv yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: foo-pv
spec:
  storageClassName: ""
  claimRef:
    name: foo-pvc
    namespace: foo
```