Introduction to Kubernetes

The Hard Way

July 1, 2022

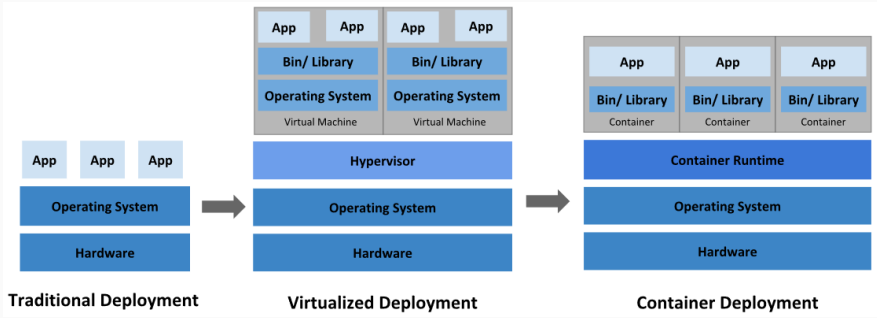
Table of Contents

Backgroud

Why Kubernetes?

Container's **Unique** Advantages

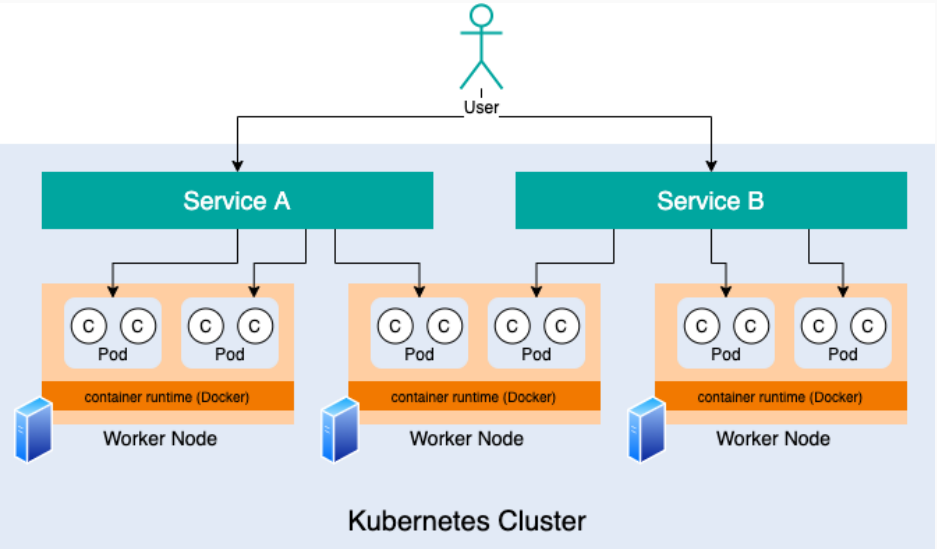
- Lightweight footprint
- Minimal overhead
- Portability across machines
- More consistent operation
- Acceptable Isolation



Why Kubernetes?

DevOps challenges for multiple containers

- Management
  - Metrics
  - Health checks
- Networking
- Scheduling
- Scaling
- Deployment
- Rollbacks
- Zero-downtime



Why Kubernetes?

Meet **Kubernetes**

```
Now the most popular container cluster
Manage applications, not machines
Open source, written in Go
```

Setup

Host

VirtualBox
Oracle Virtual Machine

Vagrant
build and maintain portable virtual machine

through command line

Goal

one master node + 2 worker nodes

=> kubernetes cluster

Host

vagrant up

Vagrantfile

```
MASTER_IP       = "192.168.56.10"
NODE_01_IP      = "192.168.56.11"
NODE_02_IP      = "192.168.56.12"
Vagrant.configure("2") do |config|
  config.vm.box = "geerlingguy/ubuntu2004"
  config.vm.box_version = "1.0.3"
  boxes = [
    { :name => "kube-master",  :ip => MASTER_IP,  :cpus
    => 2, :memory => 4096 },
    { :name => "kube-node-01", :ip => NODE_01_IP, :cpus
    => 2, :memory => 4096 },
    { :name => "kube-node-02", :ip => NODE_02_IP, :cpus
    => 2, :memory => 4096 },
  ]
// ...
```

Host

Vagrantfile

```
  boxes.each do |opts|
    config.vm.define opts[:name] do |box|
      box.vm.hostname = opts[:name]
      box.vm.network :private_network, ip: opts[:ip]
      box.vm.provider "virtualbox" do |vb|
        vb.cpus = opts[:cpus] vb.memory = opts[:memory]
      end
      box.vm.provision "shell", path:"./install.sh"
      if box.vm.hostname == "kube-master" then
        box.vm.provision "shell", path:"./cfg-master.sh"
      else
        box.vm.provision "shell", path:"./cfg-workers.sh"
      end
    end
  end
end
```

Guest - master node

cfg-master.sh

```
// ...
initialize_master_node () {
sudo systemctl enable kubelet
sudo kubeadm init
    --apiserver-advertise-address=$master_node
    --pod-network-cidr=$pod_network_cidr
    --image-repository
    registry.aliyuncs.com/google_containers
    --ignore-preflight-errors=NumCPU
}

create_join_command () {
kubeadm token create --print-join-command | tee
    /vagrant/join_command.sh
chmod +x /vagrant/join_command.sh
}
```

Guest - worker nodes

cfg-workers.sh

```
#!/bin/bash -e

get_join_command () {
sudo /vagrant/join_command.sh
}

get_join_command
```

Verification

vagrant ssh kube-master

on master node

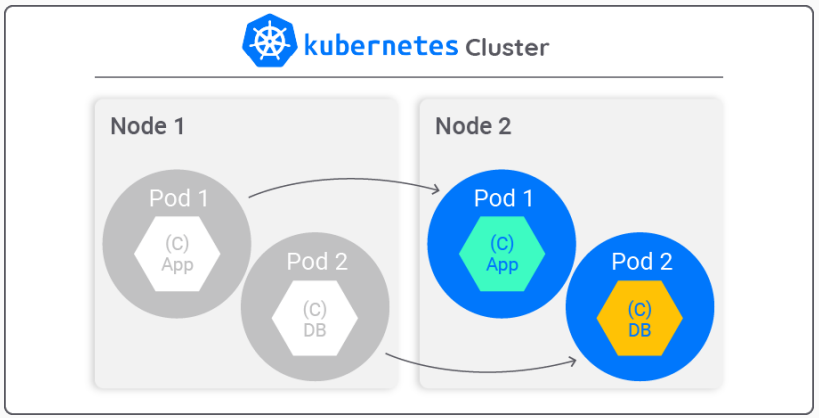
```
vagrant@kube-master:/vagrant$ kubectl get node
NAME           STATUS   ROLES                  AGE  VERSION
kube-master    Ready    control-plane,master   8d   v1.20.11
kube-node-01   Ready    <none>                 8d   v1.20.11
kube-node-02   Ready    <none>                 8d   v1.20.11
```

Concept

Node

Basic Unit that Forms a Cluster

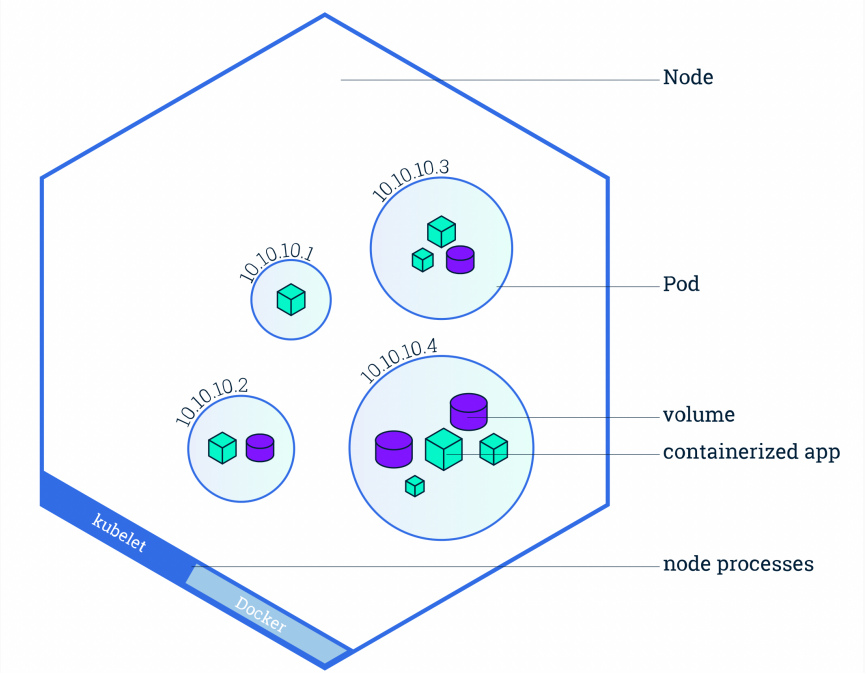
- **Master/Slave** Architecture
- Master node looks after
  - scheduling and scaling applications
  - maintaining the state of the cluster
  - implementing updates
- Worker nodes are responsible for
  - running the containers
  - doing any work assigned by the master node



Pod

Fundamental Kubernetes Work Unit

- Group of one or more containers
- Live and die together
- Support co-located processes
- Share
  - IP
  - Labels
  - Volumes



Pod

kubectl apply -f flask-pod.yaml

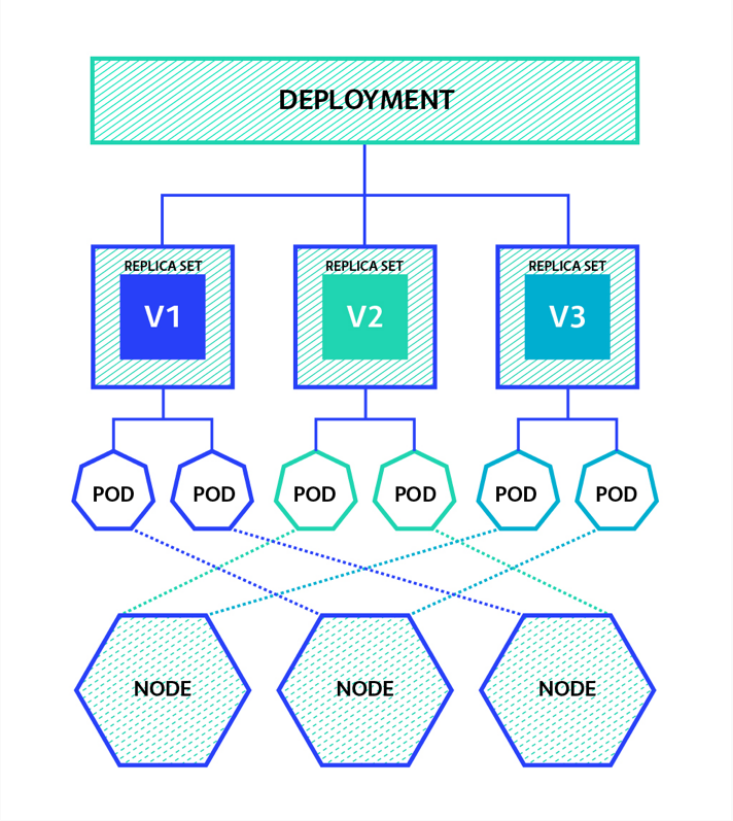
flask-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: flask-pod
  labels:
    app: flask-helloworld
spec:
  containers:
  - name: flask
    image: digitalocean/flask-helloworld:latest
    ports:
    - containerPort: 5000
```

Deployment

Manage Multiple Pods

- Try to maintain the desired state
- Creates new resources or replaces the existing resources when necessary
  - Life Cycle
  - Scheduler
  - Scaling



Deployment

kubectl apply -f flask-deployment.yaml

flask-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-dep
  labels:
    app: flask-helloworld
spec:
  replicas: 2
  selector:
    matchLabels:
      app: flask-helloworld
// ...
```

Deployment

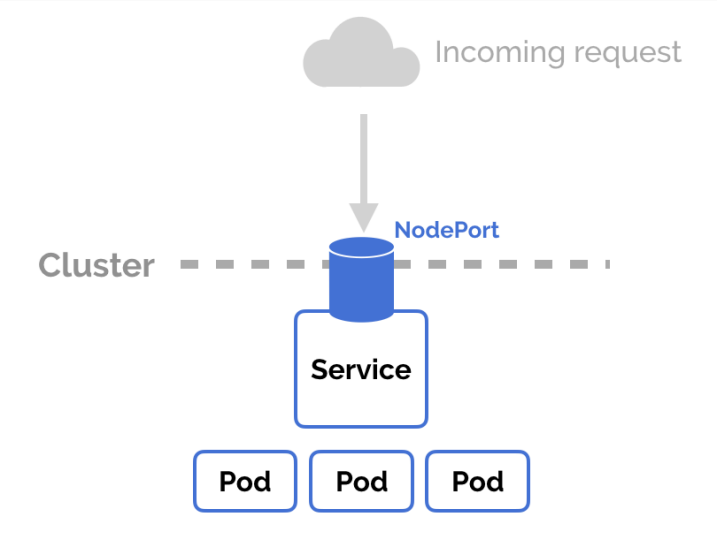
flask-deployment.yaml

```
// ...
template:
  metadata:
    labels:
      app: flask-helloworld
  spec:
    containers:
    - name: flask
      image: digitalocean/flask-helloworld
      ports:
      - containerPort: 5000
```

Service

Abstraction of Logical Pods

- Routes traffic across a set of Pods
- Allow pods to die and replicate without impacting applications
  - ClusterIP
  - NodePort
  - LoadBalancer



Service

kubectl apply -f flask-service.yaml

flask-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: flask-svc
  labels:
    app: flask-helloworld
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 5000
    protocol: TCP
  selector:
    app: flask-helloworld
```

Operator

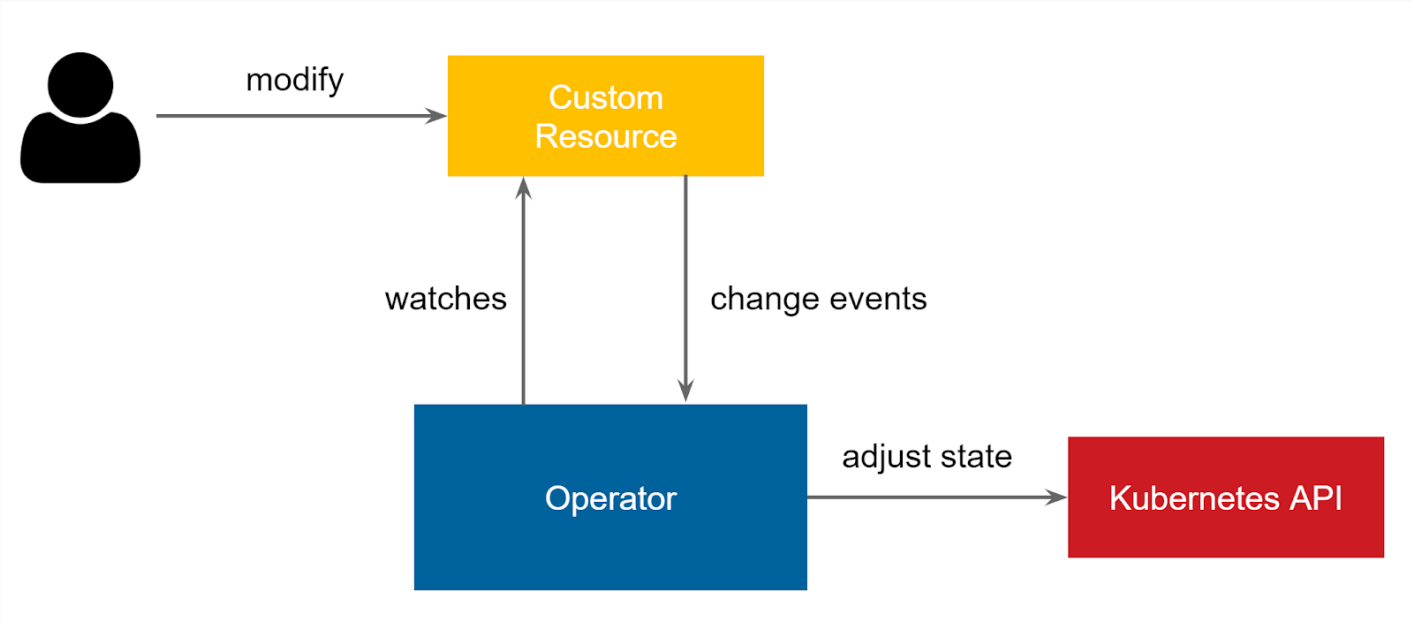
Resource Object

A resource is essentially **a collection of similar objects**.

- Built-in resources we already know
  - Pod
  - Deployment
  - Service
- Custom resources
  - A way to extend it and customize kubernetes resources.

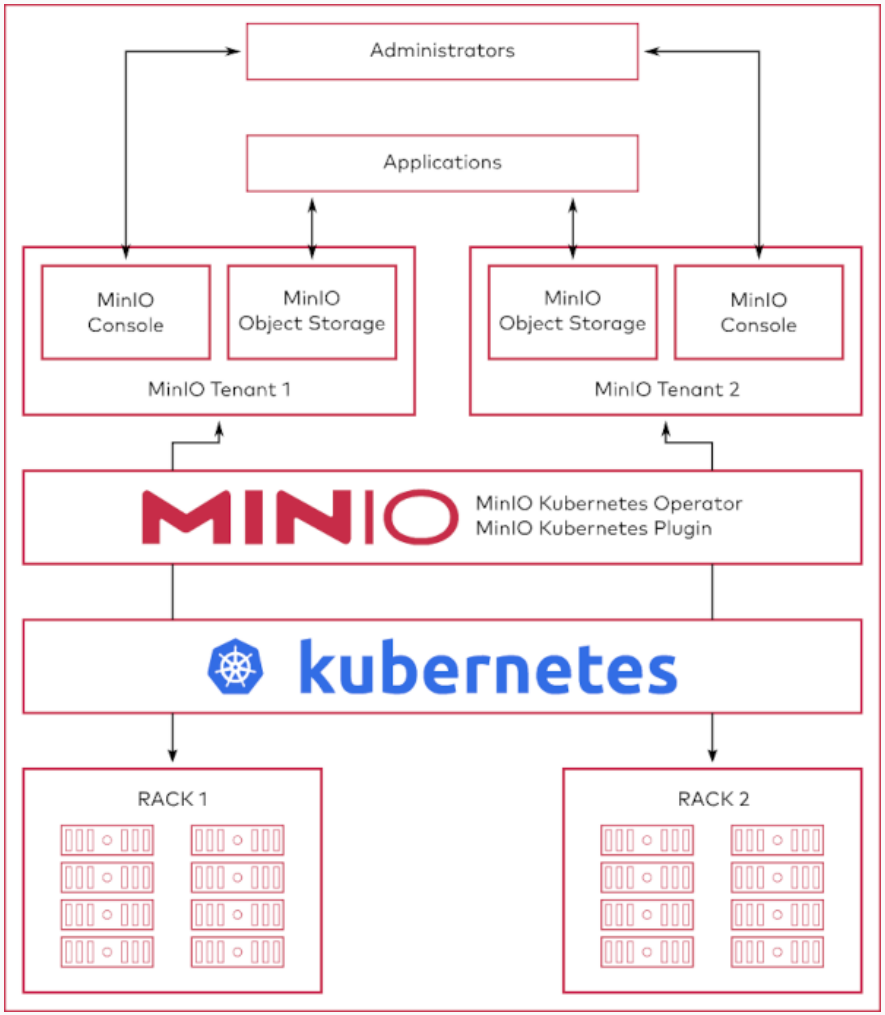
Controller

Each controller tries to move the current cluster state closer to the desired state.



What's Operator?

operator = custom resource definition + controller



Deploy MinIO on Kubernetes

Without Specific Operator

minio-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    app: minio
    release: minio
  name: minio
  namespace: default
spec:
  accessModes: - ReadWriteOnce
  capacity: storage: 10Gi
  volumeMode: Filesystem
  hostPath:
    path: /mnt/minio
```

Deploy MinIO on Kubernetes - Without Specific Operator

minio-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: minio-pv-claim
  labels:
    app: minio-storage-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Deploy MinIO on Kubernetes - Without Specific Operator

minio-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: minio-deployment
spec:
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: minio
  template:
    metadata:
      labels:
        app: minio
```

Deploy MinIO on Kubernetes - Without Specific Operator

minio-deployment.yaml

```
spec:
  volumes:
  - name: storage
    persistentVolumeClaim:
      claimName: minio-pv-claim
  containers:
  - name: minio
    # Pulls the default MinIO image from Docker Hub
    image: minio/minio
    args:
    - server
    - /storage
```

Deploy MinIO on Kubernetes - Without Specific Operator

minio-deployment.yaml

```
    env:
      # MinIO access key and secret key
      - name: MINIO_ACCESS_KEY
        value: "admin123"
      - name: MINIO_SECRET_KEY
        value: "admin123"
      ports:
      - containerPort: 9000
      # Mount the volume into the pod
      volumeMounts:
      - name: storage # must match the volume name,
  above
        mountPath: "/storage"
```

Deploy MinIO on Kubernetes - Without Specific Operator

minio-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: minio-service
spec:
  type: NodePort
  ports:
    - port: 9000
      targetPort: 9000
      protocol: TCP
  selector:
    app: minio
```

Deploy MinIO on Kubernetes

With Specific Operator

minio-usage.sh

```
# install operator
chmod +x kubectl-minio_4.0.3_linux_amd64
cp kubectl-minio_4.0.3_linux_amd64
    /usr/local/bin/kubectl-minio

# init
kubectl minio init

# deployment
kubectl minio tenant create tenant1 --namespace
    minio-tenant-1 --storage-class longhorn --servers 2
    --volumes 2 --capacity 40Gi
```

Deploy MinIO on Kubernetes

With Specific Operator

minio-usage.sh

```
# install operator
chmod +x kubectl-minio_4.0.3_linux_amd64
cp kubectl-minio_4.0.3_linux_amd64
    /usr/local/bin/kubectl-minio

# init
kubectl minio init

# deployment
kubectl minio tenant create tenant1 --namespace
    minio-tenant-1 --storage-class longhorn --servers 2
    --volumes 2 --capacity 40Gi
```

Deploy MinIO on Kubernetes - With Specific Operator

minio-deployment.yaml

```
apiVersion: miniocontroller.min.io/v1beta1
kind: MinIOInstance
metadata:
  name: minio
spec:
  metadata:
    labels:
      app: minio
    annotations:
      prometheus.io/path: /minio/prometheus/metrics
      prometheus.io/port: "9000"
      prometheus.io/scrape: "true"
  image: minio/minio:RELEASE.2019-07-24T02-02-23Z
  credsSecret:
    name: minio-creds-secret
  replicas: 2
```

Deploy MinIO on Kubernetes - With Specific Operator

minio-deployment.yaml

```
requestAutoCert: false
 certConfig:
   commonName: ""
   organizationName: []
   dnsNames: []
env:
   - name: MINIO_BROWSER
     value: "on"
 resources:
   requests:
     memory: 512Mi
     cpu: 250m
 liveness:
   httpGet:
     path: /minio/health/live
     port: 9000
```

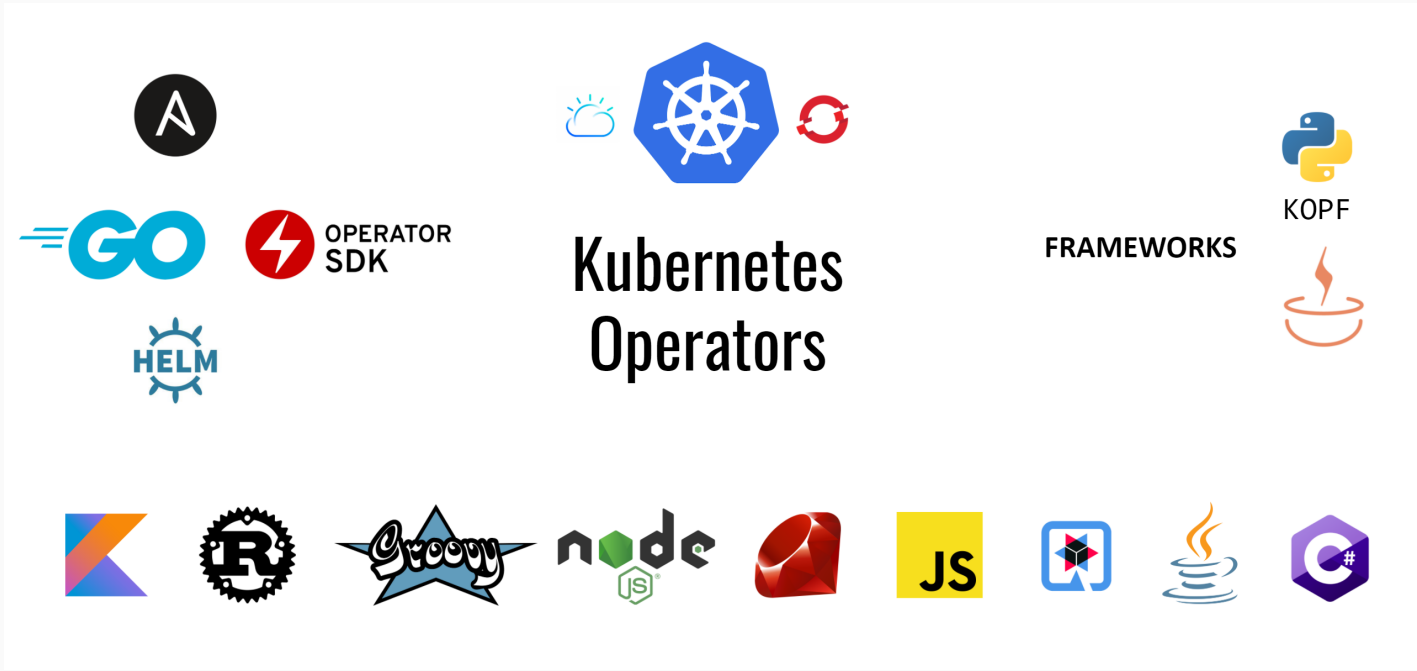
Deploy MinIO on Kubernetes - With Specific Operator

minio-deployment.yaml

```
    initialDelaySeconds: 120
    periodSeconds: 20
volumeClaimTemplate:
    metadata:
      name: data
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 10Gi
```

Operator Implementation

From Scratch or With Framework



Operator Implementation - minio operator v1.0.0

pkg/apis/miniocontroller/v1beta1/types.go

```
type MinIOInstanceSpec struct {
    Image string `json:"image,omitempty"`
    Replicas int32 `json:"replicas,omitempty"`
    // ...
    VolumeClaimTemplate *corev1.PersistentVolumeClaim
    `json:"volumeClaimTemplate,omitempty"`
    // ...
    Mountpath string `json:"mountPath,omitempty"`
    Subpath string `json:"subPath,omitempty"`
    Liveness *corev1.Probe `json:"liveness,omitempty"`
    Readiness *corev1.Probe `json:"readiness,omitempty"`
    // ...
}
```

Operator Implementation - minio operator v1.0.0

pkg/controller/cluster/controller.go

```
func (c *Controller) syncHandler(key string) error {
    // ...
    mi.EnsureDefaults()

    // ...
    if mi.Spec.Replicas != *ss.Spec.Replicas {
        glog.V(4).Infof("MinIOInstance %s replicas: %d,
    StatefulSet replicas: %d", name, mi.Spec.Replicas,
    *ss.Spec.Replicas)
        ss = statefulsets.NewForCluster(mi, svc.Name)
        _, err =
    c.kubeClientSet.AppsV1().StatefulSets(mi.Namespace).Update(ss)
    }
    // ...
}
```

Operator Implementation - minio operator v1.0.0

pkg/apis/miniocontroller/v1beta1/helper.go

```
func (mi *MinIOInstance) EnsureDefaults() *MinIOInstance {
    if mi.Spec.Replicas == 0 {
        mi.Spec.Replicas = constants.DefaultReplicas
    }
    if mi.Spec.Image == "" {
        mi.Spec.Image = constants.DefaultMinIOImage
    }
    if mi.Spec.Mountpath == "" {
        mi.Spec.Mountpath = constants.MinIOVolumeMountPath
    } else {
        mi.Spec.Mountpath = path.Clean(mi.Spec.Mountpath)
    }
    // ...
    return mi
}
```

Operator Implementation - minio operator v1.0.0

pkg/resources/statefulsets/statefulset.go

```
func NewForCluster(mi *miniov1beta1.MinIOInstance,
    serviceName string) {
    // ...
    if mi.Spec.VolumeClaimTemplate == nil {
        podVolumes = append(podVolumes,
    corev1.Volume{Name: constants.MinIOVolumeName,
                VolumeSource:
    corev1.VolumeSource{EmptyDir:
    &corev1.EmptyDirVolumeSource{Medium: ""}}})
    }
    // ...
    containers := []corev1.Container
    {minioServerContainer(mi, serviceName)}
    // ...
    return ss
}
```

Summay

Summary

So far, we know...

- the basic concept of kubernetes
- the interface and usage
- the principle of operator

Not yet introduced...

- kubernetes storage
- minio with kubernetes

Thanks for your listening!