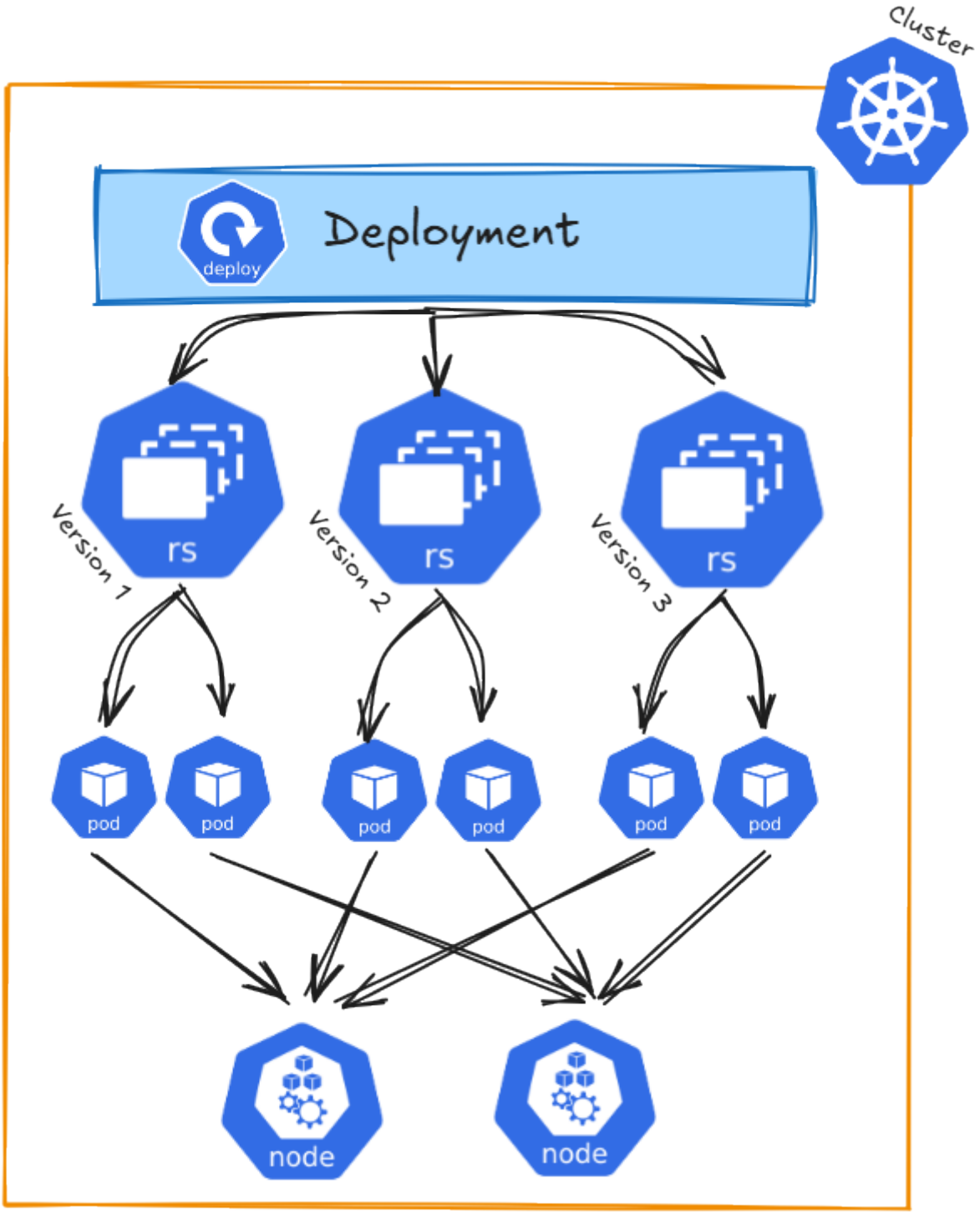
StatefulSets / DaemonSet

Useful Links

- Kubernetes Official Documentation (StatefulSets)
- Kubernetes Official Documentation (DaemonSet)

Architecture



Detailed Description

Next to deployments, there are also `StatefulSets` and `DaemonSets` in Kubernetes, designed for specific types of workloads.

StatefulSets:

- **Purpose**: Used for applications that require **stable, persistent storage** and **consistent network identities** (like databases or messaging systems).
- **Key Features**:
  - Pods are created **sequentially** with unique, predictable names (e.g., `my-app-0`, `my-app-1`).
  - Each Pod gets its **own persistent storage** (via Persistent Volume Claims) that remains even if the Pod is deleted.
  - Useful for applications that need to keep track of their state or require **ordered scaling**.

DaemonSets:

- **Purpose**: Ensure that a **copy of a Pod runs on every node** (or specific nodes) in the cluster.
- **Key Features**:
  - Pods are automatically **added or removed** when nodes are added or removed.
  - Commonly used for **system-level services** like logging, monitoring, or networking agents (e.g., Fluentd, Prometheus Node Exporter).
  - Each node gets exactly **one Pod**.

Command Reference Guide

> Remeber to use dry-run and tee to check the configuration of each command first.
> `--dry-run=client -o yaml | tee nginx-deployment.yaml`

Create a StatefulSets using a YAML file (declarative method)

nginx-statefulset.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: stateful-namespace
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
```

```
  name: nginx-statefulset
  namespace: stateful-namespace
spec:
  serviceName: "nginx"
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: kubernetes.io/hostname
                    operator: In
                    values:
                      - minikube-m02
      containers:
        - name: nginx
          image: nginx
          volumeMounts:
            - name: nginx-storage
              mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
    - metadata:
        name: nginx-storage
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 1Gi
  persistentVolumeClaimRetentionPolicy:
    whenScaled: Delete
```

```
    whenDeleted: Retain
```

```
# Apply the StatefulSet
kubectl apply -f nginx-statefulset.yaml

# Write data to pod
kubectl exec -it -n stateful-namespace nginx-statefulset-0 -- /bin/bash
echo "Hello from StatefulSet" > /usr/share/nginx/html/index.html
exit

# Delete pod
kubectl delete pod nginx-statefulset-0 -n stateful-namespace --now

# Check file since pod is recreated on same node (check nodeAffinity in yaml)
kubectl exec -it -n stateful-namespace nginx-statefulset-0 -- /bin/bash
cat /usr/share/nginx/html/index.html

#Delete the stateful set
kubectl delete statefulset nginx-statefulset -n stateful-namespace

# Reaply the statefulset
kubectl apply -f nginx-statefulset.yaml

# Check file since pod is recreated on same node (check nodeAffinity in yaml)
kubectl exec -it -n stateful-namespace nginx-statefulset-0 -- /bin/bash
cat /usr/share/nginx/html/index.html
```

Change `whenDeleted: Retain` to `whenDeleted: Delete` in yaml file and retry full szenario.

Create a Deplyoment (imperative method)

nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
```

```
  metadata:
    labels:
      app: nginx-deployment
    name: nginx-deployment
  spec:
    replicas: 25
    selector:
      matchLabels:
        app: nginx-deployment
    strategy:
      rollingUpdate:
        maxSurge: 25%
        maxUnavailable: 25%
      type: RollingUpdate
    template:
      metadata:
        creationTimestamp: null
        labels:
          app: nginx-deployment
      spec:
        containers:
        - image: nginxdemos/hello:0.4
          imagePullPolicy: Always
          name: hello
          ports:
          - containerPort: 80
            protocol: TCP
          resources: {}
  status: {}
```

```
# Create nginx deployment with the default of one replica
kubectl create deployment nginx-deployment --image=nginxdemos/hello --port=80

# Create nginx deployment with three replicas
kubectl create deployment nginx-deployment --image=nginxdemos/hello --port=80  --replicas=3

# Check deployment
kubectl get deployment -o wide
```

```
# Get detailed deployment information
kubectl describe deployment

# Get ReplicaSet information created by deployment
kubectl get replicasets -o wide

# Get History for deployment
kubectl rollout history deployment/nginx-deployment

# Annotate inital history entry
kubectl annotate deployment/nginx-deployment kubernetes.io/change-cause="init nginx deployment"
kubectl rollout history deployment/nginx-deployment

# Scale up/down deployment (scale is not changing history)
kubectl scale deployment/nginx-deployment --replicas=2; watch kubectl get pods -o wide
kubectl scale deployment/nginx-deployment --replicas=20; watch kubectl get pods -o wide



# Run update and rollback
FIRST_POD=$(kubectl get pods -l app=nginx-deployment -o jsonpath='{.items[0].metadata.name}')
# Check image name
kubectl get pod $FIRST_POD -o jsonpath='{.spec.containers[0]}'}
kubectl get deployment/nginx-deployment -o yaml > nginx-deployment.yaml
# check Update yaml modifications under code section
kubectl apply -f nginx-deployment.yaml && kubectl rollout status deployment/nginx-deployment
kubectl annotate deployment/nginx-deployment kubernetes.io/change-cause="update new version"
kubectl rollout history deployment/nginx-deployment
# Recheck image name after update
kubectl get pod $FIRST_POD -o jsonpath='{.spec.containers[0]}'}
# Check replicaset
kubectl get replicaset

# ROllback to previouse revision
kubectl rollout undo deployment/nginx-deployment --to-revision=1 && kubectl rollout status deployment/nginx-
deployment
kubectl rollout history deployment/nginx-deployment
# check pod image, as it was reverted to old revision
FIRST_POD=$(kubectl get pods -l app=nginx-deployment -o jsonpath='{.items[0].metadata.name}')
kubectl get pod $FIRST_POD -o jsonpath='{.spec.containers[0]}'}
```

```
# Run into failed state (change image in yaml to  nginxdemos/hello:5.1)
kubectl apply -f nginx-deployment.yaml && kubectl rollout status deployment/nginx-deployment
kubectl annotate deployment/nginx-deployment kubernetes.io/change-cause="failed version"
kubectl rollout history deployment/nginx-deployment
kubectl rollout undo deployment/nginx-deployment --to-revision=3 && kubectl rollout status deployment/nginx-deployment
# check pod image, as it was reverted to healthy revision
FIRST_POD=$(kubectl get pods -l app=nginx-deployment -o jsonpath='{.items[0].metadata.name}')
kubectl get pod $FIRST_POD -o jsonpath='{.spec.containers[0]}'}
kubectl rollout history deployment/nginx-deployment

# Pause from deplyoment
kubectl rollout pause deployment nginx-deployment
kubectl set image deployment/nginx-deployment nginx=nginx:1.22
kubectl get deployment nginx-deployment -o yaml | grep paused
kubectl rollout resume deployment nginx-deployment
kubectl rollout status deployment nginx-deployment



# Delete deployment
kubectl delete deployment/nginx-deployment
```

Hints

Deployments manage **ReplicaSets**, primarily due to historical reasons. There is no practical need to manually create ReplicaSets (or previously, ReplicationControllers), as Deployments, built on top of ReplicaSets, offer a more user-friendly and feature-rich abstraction for managing the application lifecycle, including replication, updates, and rollbacks.

ReplicaSets do not support auto updates. As long as required number of pods exist matching the selector labels, replicaset's jobs is done.

When a **rollback** is applied to a Deployment, Kubernetes creates a new history revision for the rollback. It doesn’t simply go back to an old revision but treats the rollback as a new change. This means the rollback gets its own revision number, while the previous revisions remain saved. This helps you keep track of all changes, including rollbacks.

Revision #5
Created 28 November 2024 17:02:24 by Admin
Updated 29 November 2024 16:22:19 by Admin