

Lab Exercise 8- Creating and Managing a ReplicaSet in Kubernetes

Objective:

A ReplicaSet in Kubernetes ensures a specified number of Pod replicas are running at any given time. This exercise will guide you through creating a ReplicaSet to maintain the desired state of your application.

- Understand the syntax and structure of a Kubernetes ReplicaSet definition file (YAML).
- Learn how to create and manage a ReplicaSet to ensure application availability.
- Understand how a ReplicaSet helps in scaling applications and maintaining desired states.

Prerequisites

- Kubernetes Cluster: Have a running Kubernetes cluster (locally using Minikube or kind, or a cloud-based service).
- kubectl: Install and configure kubectl to interact with your Kubernetes cluster.
- Basic Knowledge of YAML: Familiarity with YAML format will be helpful for understanding Kubernetes resource definitions.

Step-by-Step Guide

Step 1: Understanding ReplicaSet

A ReplicaSet ensures a specified number of Pod replicas are running at any given time. If a Pod crashes or is deleted, the ReplicaSet creates a new one to meet the defined number of replicas. This helps maintain application availability and ensures that your application can handle increased load by distributing traffic among multiple Pods.

Step 2: Create a ReplicaSet

We'll define a ReplicaSet to maintain three replicas of a simple Nginx web server Pod.

Create a YAML file named `nginx-replicaset.yaml` with the following content:

```
apiVersion: apps/v1      # Specifies the API version used.
kind: ReplicaSet          # The type of resource being defined; here, it's a
                           # ReplicaSet.
metadata:
  name: easy-drive-replicaset # The name of the ReplicaSet.
spec:
  replicas: 3              # The desired number of Pod replicas.
  selector:
    matchLabels:           # Criteria to identify Pods managed by this ReplicaSet.
      app: nginx-app       # The label that should match Pods.
  template:                # The Pod template for creating new Pods.
    metadata:
      labels:
        app: nginx-app     # Labels applied to Pods created by this ReplicaSet.
    spec:
      containers:
        - name: easy-drive
          image: booraraman/easy-drive-rentals:1
```

ports:

- containerPort: 5600 # The port the container exposes.

Explanation:

- apiVersion: Defines the API version (apps/v1) used for the ReplicaSet resource.
- kind: Specifies that this resource is a ReplicaSet.
- metadata: Contains metadata about the ReplicaSet, including name.
 - o name: The unique name for the ReplicaSet.
- spec: Provides the specification for the ReplicaSet.
 - o replicas: Defines the desired number of Pod replicas.
 - o selector: Criteria for selecting Pods managed by this ReplicaSet.
 - matchLabels: Labels that Pods must have to be managed by this ReplicaSet.
 - o template: Defines the Pod template used for creating new Pods.
 - metadata: Contains metadata for the Pods, including labels.
 - labels: Labels applied to Pods created by this ReplicaSet.
 - spec: Specification for the Pods.
 - containers: Lists the containers that will run in the Pod.
 - name: The unique name of the container within the Pod.
 - image: The Docker image used for the container.
 - ports: Ports exposed by the container.

Step 3: Apply the YAML to Create the ReplicaSet

Use the `kubectl apply` command to create the ReplicaSet based on the YAML file.

```
kubectl apply -f rs.yaml
```

```
vagrant@controlplane:~$ ls
calico.yaml pod.yaml svc.yaml svcc.yaml
vagrant@controlplane:~$ vi rs.yaml
vagrant@controlplane:~$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
easy-drive-pod 1/1     Running   0           9m41s
vagrant@controlplane:~$ kubectl create -f rs.yaml
error: error parsing rs.yaml: error converting YAML to JSON: yaml: line 18: could not find expected ':'
vagrant@controlplane:~$ vi rs.yaml
vagrant@controlplane:~$ kubectl create -f rs.yaml
replicaset.apps/easy-drive-replicaset created
vagrant@controlplane:~$ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
easy-drive-pod                      1/1     Running             0           11m
easy-drive-replicaset-crngm         0/1     ContainerCreating   0           4s
easy-drive-replicaset-wmxtw         1/1     Running             0           4s
vagrant@controlplane:~$ kubectl stop pods easy-drive-pod
```

Verify the ReplicaSet is running and maintaining the desired number of replicas:

```
kubectl get replicaset
```

```
vagrant@controlplane:~$ kubectl get rs
NAME                DESIRED   CURRENT   READY   AGE
easy-drive-replicaset 3          3         2       2m1s
```

This command lists all ReplicaSets in the current namespace.

To check the Pods created by the ReplicaSet:

```
kubectl get pods -l app=nginx-app
```

This command lists all Pods with the label app=nginx.

```
vagrant@controlplane:~$ kubectl get pods -l app=nginx-app
```

NAME	READY	STATUS	RESTARTS	AGE
easy-drive-pod	1/1	Running	0	13m
easy-drive-replicaset-crngm	0/1	ImagePullBackOff	0	2m26s
easy-drive-replicaset-wmxtw	1/1	Running	0	2m26s

Step 4: Managing the ReplicaSet

1. Scaling the ReplicaSet

You can scale the number of replicas managed by the ReplicaSet using the kubectl scale command.

```
kubectl scale --replicas=5 replicaset/easy-drive-replicaset
```

```
vagrant@controlplane:~$ kubectl scale --replicas=5 replicaset/easy-drive-replicaset
replicaset.apps/easy-drive-replicaset scaled
vagrant@controlplane:~$ kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
easy-drive-replicaset	5	5	5	3m54s

This command scales the ReplicaSet to maintain 5 replicas. Verify the scaling operation:

```
kubectl get pods -l app=nginx-app
```

You should see that the number of Pods has increased to 5.

```
vagrant@controlplane:~$ kubectl get pods -l app=nginx-app
NAME                                READY   STATUS    RESTARTS   AGE
easy-drive-pod                     1/1     Running   0           15m
easy-drive-replicaset-crnqm        1/1     Running   0           4m31s
easy-drive-replicaset-l2fkp        1/1     Running   0           43s
easy-drive-replicaset-vs7lk        1/1     Running   0           43s
easy-drive-replicaset-wmxtw        1/1     Running   0           4m31s
vagrant@controlplane:~$
```

2. Updating the ReplicaSet

If you need to update the Pod template (e.g., to use a different Docker image version), modify the YAML file and apply it again. For instance, change the image to a specific version of Nginx:

```
spec:
  template:
    spec:
      containers:
      - name: nginx-app
```

```
image: booraraman/easy-drive-rentals # Change to a specific version
```

Apply the changes:

```
kubectl apply -f replica.yaml
```

```
vagrant@controlplane:~$ kubectl get rs easy-drive-replicaset -o wide
```

NAME	DESIRED	CURRENT	READY	AGE	CONTAINERS	IMAGES	SELECTOR
easy-drive-replicaset	3	3	3	7d18h	easy-drive	booraraman/easy-drive-rentals	app=nginx-app

```
vagrant@controlplane:~$
```

If you need to update the Pod template (e.g., to use a different Docker image version), modify the YAML file and apply it again. For instance, change the image to a specific version of Nginx:

Check the status to ensure the Pods are updated:

```
kubectl get pods -l app=nginx-app
```

```
vagrant@controlplane:~$ kubectl get pods -l app=nginx-app
```

NAME	READY	STATUS	RESTARTS	AGE
easy-drive-pod	1/1	Running	1	7d19h
easy-drive-replicaset-crnmq	1/1	Running	1	7d18h
easy-drive-replicaset-wmxtw	1/1	Running	1	7d18h

Note: Updating a ReplicaSet doesn't automatically replace existing Pods with new ones. In practice, you often create a new ReplicaSet or Deployment for updates.

3. Deleting the ReplicaSet

To clean up the ReplicaSet and its Pods, use the `kubectl delete` command:

```
kubectl delete -f easy-drive-replicaset.yaml
```

```
vagrant@controlplane:~$ kubectl delete rs easy-drive-replicaset  
replicaset.apps "easy-drive-replicaset" deleted
```

This command deletes the ReplicaSet and all the Pods managed by it.