EXPERIMENT 8

AIM: 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.

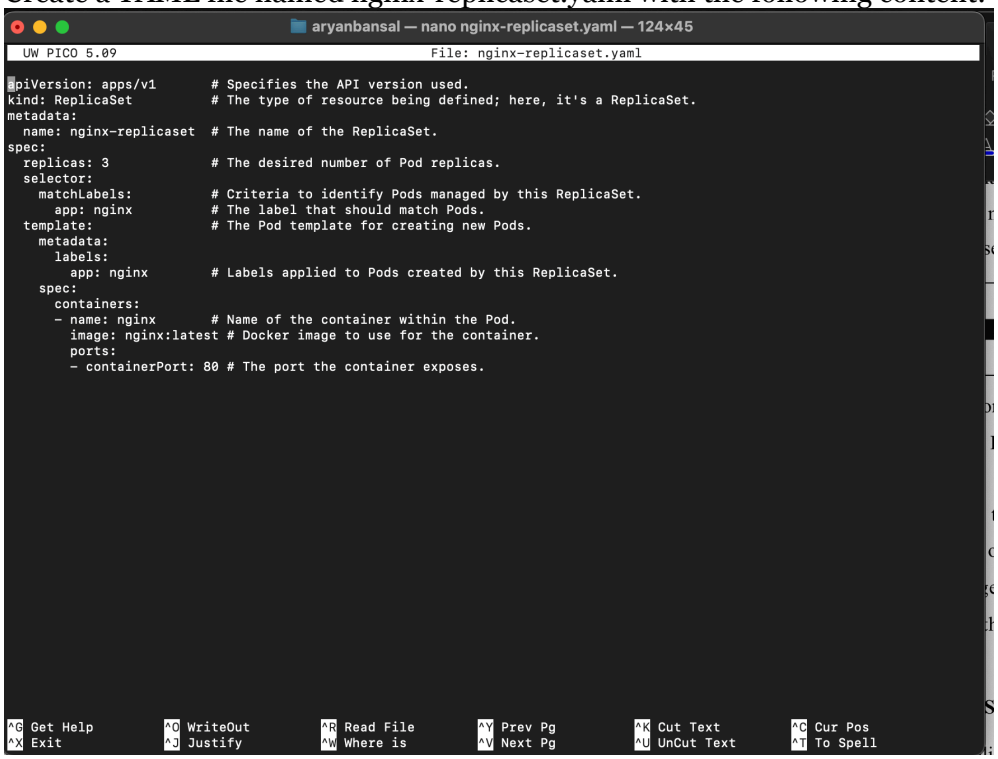
Step-by-Step Guide

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:



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.
- oname: The unique name for the ReplicaSet.
- spec: Provides the specification for the ReplicaSet.
- oreplicas: Defines the desired number of Pod replicas.
- oselector: Criteria for selecting Pods managed by this ReplicaSet.
- matchLabels: Labels that Pods must have to be managed by this ReplicaSet.
- otemplate: 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.
- ospec: 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 nginx-replicaset.yaml

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl apply -f nginx-replicaset.yaml
replicaset.apps/nginx-replicaset created
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

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

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get replicaset
NAME               DESIRED   CURRENT   READY   AGE
nginx-replicaset   3         3         0       41m
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

This command lists all ReplicaSets in the current namespace.
To check the Pods created by the ReplicaSet:
kubectl get pods -l app=nginx
This command lists all Pods with the label app=nginx.

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get pods -l app=nginx
NAME                     READY   STATUS             RESTARTS   AGE
nginx-replicaset-d8l7c   1/1     Running            0          42m
nginx-replicaset-ghrxf   0/1     ImagePullBackOff   0          42m
nginx-replicaset-pfmzp   1/1     Running            0          42m
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

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/nginx-replicaset
This command scales the ReplicaSet to maintain 5 replicas. Verify the scaling operation:

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl scale --replicas=5 replicaset/nginx-replicaset
replicaset.apps/nginx-replicaset scaled
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

kubectl get pods -l app=nginx
You should see that the number of Pods has increased to 5.

```
replicaset.apps/nginx-replicaset scaled
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get pods -l app=nginx
NAME                     READY   STATUS    RESTARTS   AGE
nginx-replicaset-d8l7c   1/1     Running   0          44m
nginx-replicaset-dwrnx   1/1     Running   0          48s
nginx-replicaset-ghrxf   1/1     Running   0          44m
nginx-replicaset-kvpg7   1/1     Running   0          48s
nginx-replicaset-pfmzp   1/1     Running   0          44m
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

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:

Apply the changes:
kubectl apply -f nginx-replicaset.yaml

```
  UW PICO 5.09                                        File: nginx-replicaset.yaml

spec:
  template:
    spec:
      containers:
      - name: nginx
        image: nginx:1.19.3  # Change to a specific version

^G Get Help          ^O WriteOut          ^R Read File         ^Y Prev Pg
^X Exit              ^J Justify           ^W Where is          ^V Next Pg
```

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl apply -f nginx-replicaset.yaml
replicaset.apps/nginx-replicaset configured
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

Check the status to ensure the Pods are updated:
kubectl get pods -l app=nginx

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get pods -l app=nginx
NAME                     READY   STATUS    RESTARTS        AGE
nginx-replicaset-d8l7c   1/1     Running   2 (2m39s ago)   57m
nginx-replicaset-ghrxf   1/1     Running   2 (2m39s ago)   57m
nginx-replicaset-kvpg7   1/1     Running   2 (2m39s ago)   13m
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

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 nginx-replicaset.yaml