Pratik Agrawal
500123601
Devops B2

Lab Exercise 10- 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

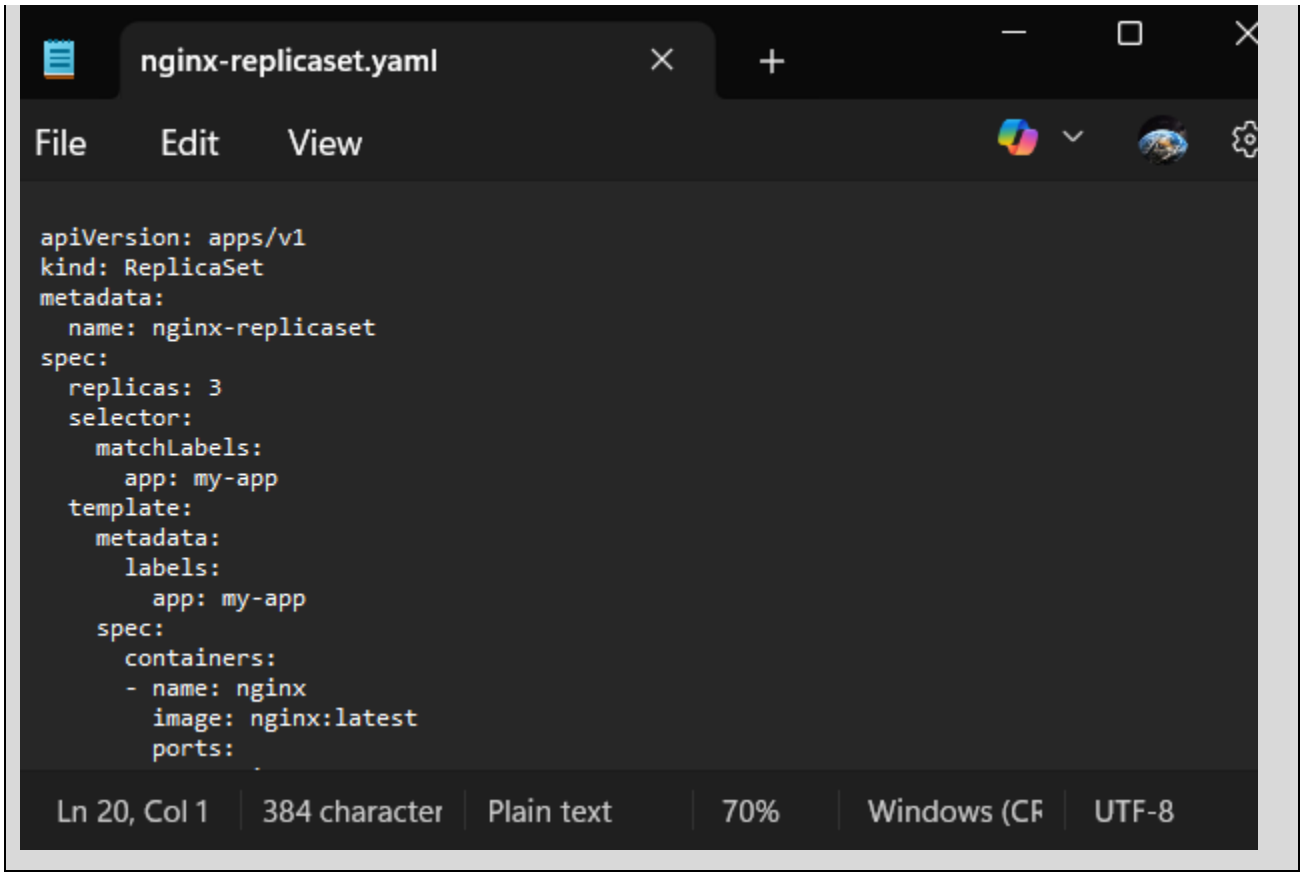
Step 1: Understanding ReplicaSet

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
kind: ReplicaSet
metadata:
  name: nginx-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```



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

```
C:\Users\prati\replicaset>kubectl apply -f nginx-replicaset.yaml
replicaset.apps/nginx-replicaset created
```

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

kubectl get replicaset

```
C:\Users\prati\replicaset>kubectl get replicaset
NAME               DESIRED   CURRENT   READY   AGE
nginx-replicaset   3         3         2       9s
```

This command lists all ReplicaSets in the current namespace.

To check the Pods created by the ReplicaSet:

kubectl get pods -l app=nginx

```
C:\Users\prati\replicaset>kubectl get pods -l app=nginx
No resources found in default namespace.
```

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

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

```
C:\Users\prati\replicaset>kubectl scale --replicas=5 replicaset/nginx-replic
aset
replicaset.apps/nginx-replicaset scaled
```

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

kubectl get pods -l app=nginx

```
C:\Users\prati\replicaset>kubectl get pods -l app=nginx
No resources found in default namespace.
```

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

2. Updating the ReplicaSet

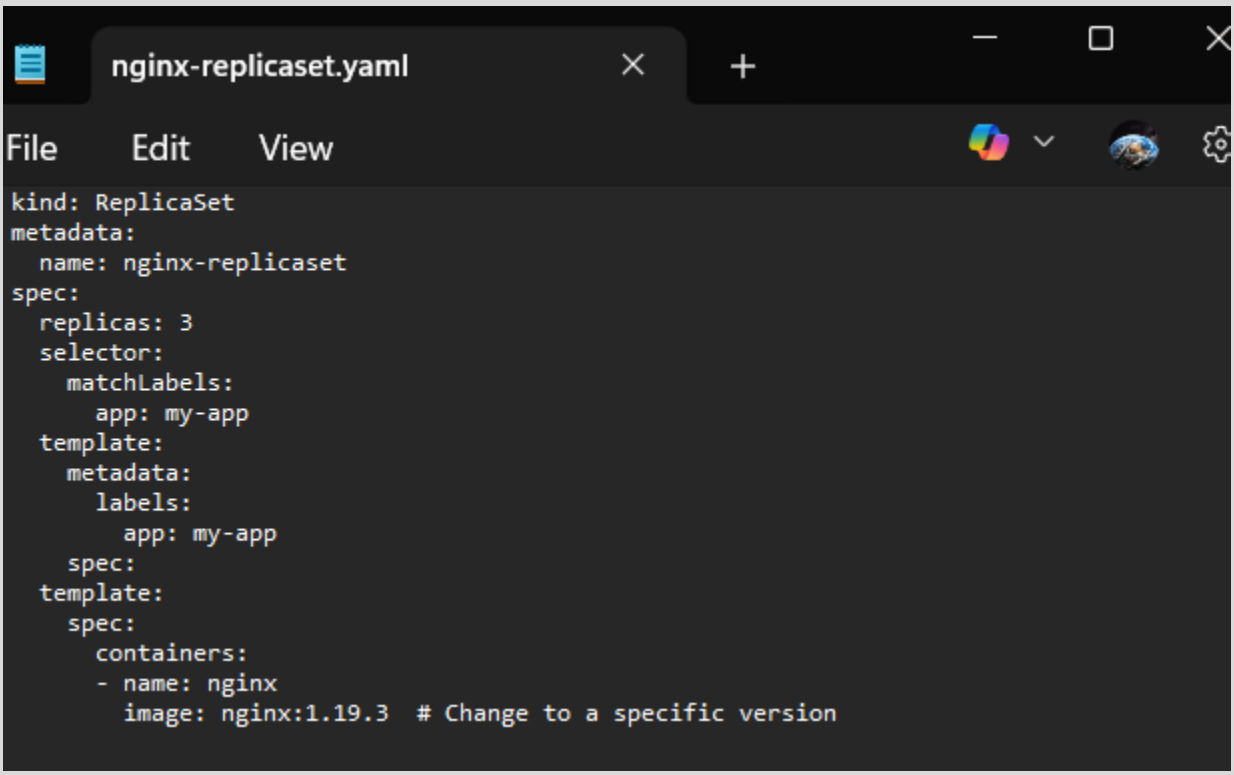
If you need to update the Pod template (e.g., to use a different Docker image version), modify the YAML file and apply it again. For instance, change the image to a specific version of Nginx:

```
spec:
  template:
    spec:
```

containers:

- name: nginx

  image: nginx:1.19.3 # Change to a specific version



Apply the changes:

kubectl apply -f nginx-replicaset.yaml

```
C:\Users\prati\replicaset>kubectl apply -f nginx-replicaset.yaml
The ReplicaSet "nginx-replicaset" is invalid:
* spec.template.metadata.labels: Invalid value: null: `selector` does not ma
tch template `labels`
* spec.selector: Invalid value: {"matchLabels":{"app":"my-app"}}: field is i
mmutable
```

Check the status to ensure the Pods are updated:

kubectl get pods -l app=nginx

```
C:\Users\prati\replicaset>kubectl get pods -l app=nginx
NAME                      READY   STATUS    RESTARTS   AGE
nginx-replicaset-8sqnh    1/1     Running   0          5m12s
nginx-replicaset-txkfg    1/1     Running   0          5m12s
nginx-replicaset-zv9rz    1/1     Running   0          5m12s
```

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

```
C:\Users\prati\replicaset>kubectl delete -f nginx-replicaset.yaml
replicaset.apps "nginx-replicaset" deleted from default namespace
```

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