

# 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

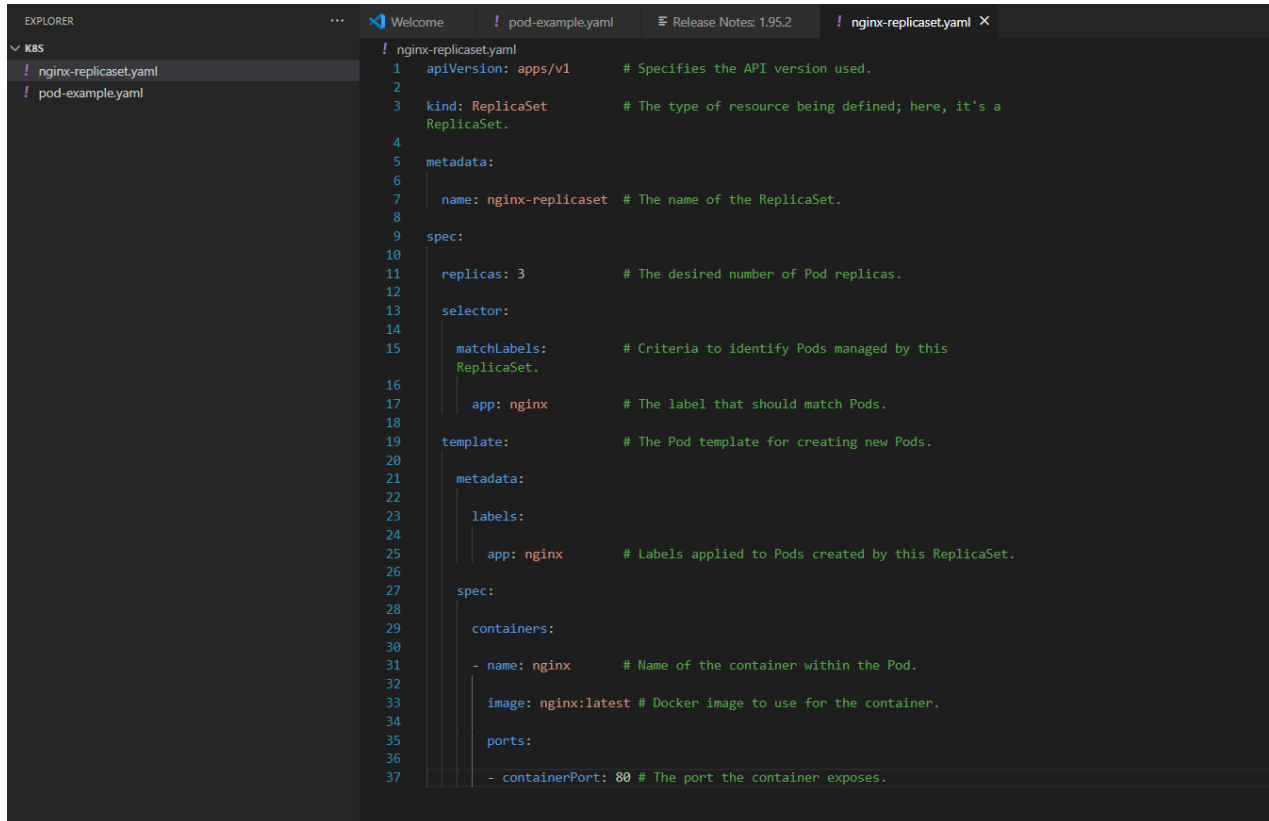
### Step 1: Understanding ReplicaSet

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: nginx-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        # The label that should match Pods.
  template:             # The Pod template for creating new Pods.
    metadata:
      labels:
        app: nginx      # Labels applied to Pods created by this ReplicaSet.
    spec:
      containers:
        - name: nginx    # Name of the container within the Pod.
          image: nginx:latest # Docker image to use for the container.
          ports:
            - containerPort: 80 # The port the container exposes.
```



```
1 apiVersion: apps/v1      # Specifies the API version used.
2
3 kind: ReplicaSet         # The type of resource being defined; here, it's a
4                           ReplicaSet.
5
6 metadata:
7   name: nginx-replicaset # The name of the ReplicaSet.
8
9 spec:
10   replicas: 3             # The desired number of Pod replicas.
11
12   selector:
13     matchLabels:          # Criteria to identify Pods managed by this
14                           ReplicaSet.
15     app: nginx            # The label that should match Pods.
16
17   template:               # The Pod template for creating new Pods.
18     metadata:
19       labels:
20         app: nginx        # Labels applied to Pods created by this ReplicaSet.
21
22     spec:
23       containers:
24         - name: nginx     # Name of the container within the Pod.
25           image: nginx:latest # Docker image to use for the container.
26           ports:
27             - containerPort: 80 # 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.
  - **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\Vibhav Khaneja\OneDrive\Desktop\K8S>kubectl apply -f nginx-replicaset.yaml
replicaset.apps/nginx-replicaset created

C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>
```

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

```
kubectl get replicaset
```

```
C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>kubectl get replicaset
NAME           DESIRED   CURRENT   READY   AGE
nginx-replicaset 3          3         0       69s

C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>
```

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\Vibhav Khaneja\OneDrive\Desktop\K8S>kubectl get pods -l app=nginx
NAME                READY   STATUS    RESTARTS   AGE
nginx-replicaset-jpvnd 1/1     Running   0          102s
nginx-replicaset-vll5g 1/1     Running   0          102s
nginx-replicaset-x494l 1/1     Running   0          102s

C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>|
```

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\Vibhav Khaneja\OneDrive\Desktop\K8S>kubectl scale --replicas=5 replicaset/nginx-replicaset
replicaset.apps/nginx-replicaset scaled

C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>|
```

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

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

```
C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>kubectl get pods -l app=nginx
NAME                READY   STATUS    RESTARTS   AGE
nginx-replicaset-9lkz1 1/1     Running   0          28s
nginx-replicaset-jpvnd 1/1     Running   0          3m22s
nginx-replicaset-pl6jz 1/1     Running   0          28s
nginx-replicaset-vll5g 1/1     Running   0          3m22s
nginx-replicaset-x494l 1/1     Running   0          3m22s

C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>
```

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
```

```

! nginx-replicaset.yaml
1  apiVersion: apps/v1      # Specifies the API version used.
2
3  kind: ReplicaSet         # The type of resource being defined; here, it's a
  ReplicaSet.
4
5  metadata:
6
7    name: nginx-replicaset # The name of the ReplicaSet.
8
9  spec:
10
11    replicas: 3             # The desired number of Pod replicas.
12
13    selector:
14
15      matchLabels:         # Criteria to identify Pods managed by this
  ReplicaSet.
16
17      app: nginx           # The label that should match Pods.
18
19    template:              # The Pod template for creating new Pods.
20
21      metadata:
22
23        labels:
24
25          app: nginx       # Labels applied to Pods created by this ReplicaSet.
26
27      spec:
28
29        containers:
30
31          - name: nginx     # Name of the container within the Pod.
32
33            image: nginx:1.19.3 # Docker image to use for the container.
34
35            ports:
36
37              - containerPort: 80 # The port the container exposes.

```

**Apply the changes:**

```
kubectl apply -f nginx-replicaset.yaml
```

```

C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>kubectl apply -f nginx-replicaset.yaml
replicaset.apps/nginx-replicaset configured

```

```
C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>
```

**Check the status to ensure the Pods are updated:**

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

```
C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>kubectl get pods -l app=nginx
NAME                                READY   STATUS    RESTARTS   AGE
nginx-replicaset-jpvnd             1/1     Running   0           5m43s
nginx-replicaset-vll5g             1/1     Running   0           5m43s
nginx-replicaset-x494l             1/1     Running   0           5m43s
C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>|
```

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\Vibhav Khaneja\OneDrive\Desktop\K8S>kubectl delete -f nginx-replicaset.yaml
replicaset.apps "nginx-replicaset" deleted
C:\Users\Vibhav Khaneja\OneDrive\Desktop\K8S>|
```

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