Name: Simranjeet kour

Sap Id: 500110713

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**

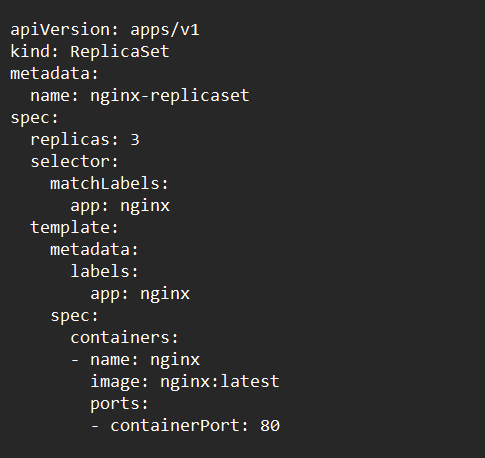
A ReplicaSet ensures a specified number of Pod replicas are running at any given time. If a Pod crashes or is deleted, the ReplicaSet creates a new one to meet the defined number of replicas. This helps maintain application availability and ensures that your application can handle increased load by distributing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







## 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.

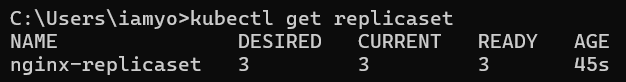
## Step 3: Apply the YAML to Create the ReplicaSet

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



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



This command lists all ReplicaSets in the current namespace.

## To check the Pods created by the ReplicaSet:



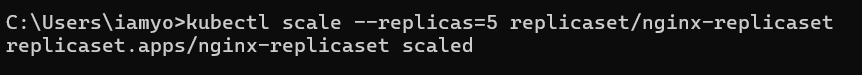
This command lists all Pods with the label app=nginx.

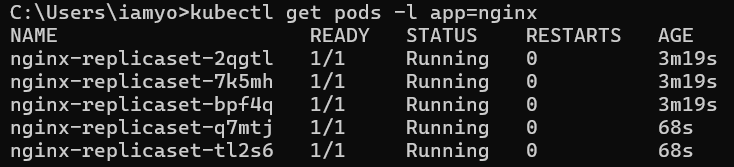
## Step 4: Managing the ReplicaSet

1. **Scaling the ReplicaSet**

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



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



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

## 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:



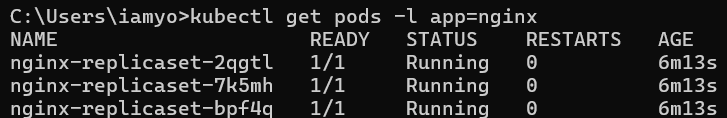


## Apply the changes:



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



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.

## Deleting the ReplicaSet

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



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