**Step-01: Introduction to ReplicaSets**

* What are ReplicaSets?
* What is the advantage of using ReplicaSets?

**Step-02: Create ReplicaSet**

**Create ReplicaSet**

* Create ReplicaSet

kubectl create -f replicaset-demo.yml

* **replicaset-demo.yml**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: my-helloworld-rs

labels:

app: my-helloworld

spec:

replicas: 3

selector:

matchLabels:

app: my-helloworld

template:

metadata:

labels:

app: my-helloworld

spec:

containers:

- name: my-helloworld-app

image: stacksimplify/kube-helloworld:1.0.0

**List ReplicaSets**

* Get list of ReplicaSets

kubectl get replicaset

kubectl get rs

**Describe ReplicaSet**

* Describe the newly created ReplicaSet

kubectl describe rs/<replicaset-name>

kubectl describe rs/my-helloworld-rs

[or]

kubectl describe rs my-helloworld-rs

**List of Pods**

* Get list of Pods

#Get list of Pods

kubectl get pods

kubectl describe pod <pod-name>

# Get list of Pods with Pod IP and Node in which it is running

kubectl get pods -o wide

**Verify the Owner of the Pod**

* Verify the owner reference of the pod.
* Verify under **"name"** tag under **"ownerReferences"**. We will find the replicaset name to which this pod belongs to.

kubectl get pods <pod-name> -o yaml

kubectl get pods my-helloworld-rs-c8rrj -o yaml

**Step-03: Expose ReplicaSet as a Service**

* Expose ReplicaSet with a service (Load Balancer Service) to access the application externally (from internet)

# Expose ReplicaSet as a Service

kubectl expose rs <ReplicaSet-Name> --type=LoadBalancer --port=80 --target-port=8080 --name=<Service-Name-To-Be-Created>

kubectl expose rs my-helloworld-rs --type=LoadBalancer --port=80 --target-port=8080 --name=my-helloworld-rs-service

# Get Service Info

kubectl get service

kubectl get svc

* **Access the Application using External or Public IP**

http://<External-IP-from-get-service-output>/hello

**Step-04: Test Replicaset Reliability or High Availability**

* Test how the high availability or reliability concept is achieved automatically in Kubernetes
* Whenever a POD is accidentally terminated due to some application issue, ReplicaSet should auto-create that Pod to maintain desired number of Replicas configured to achive High Availability.

# To get Pod Name

kubectl get pods

# Delete the Pod

kubectl delete pod <Pod-Name>

# Verify the new pod got created automatically

kubectl get pods (Verify Age and name of new pod)

**Step-05: Test ReplicaSet Scalability feature**

* Test how scalability is going to seamless & quick
* Update the **replicas** field in **replicaset-demo.yml** from 3 to 6.

# Before change

spec:

replicas: 3

# After change

spec:

replicas: 6

* Update the ReplicaSet

# Apply latest changes to ReplicaSet

kubectl replace -f replicaset-demo.yml

# Verify if new pods got created

kubectl get pods -o wide

**Step-06: Delete ReplicaSet & Service**

**Delete ReplicaSet**

# Delete ReplicaSet

kubectl delete rs <ReplicaSet-Name>

# Sample Commands

kubectl delete rs/my-helloworld-rs

[or]

kubectl delete rs my-helloworld-rs

# Verify if ReplicaSet got deleted

kubectl get rs

**Delete Service created for ReplicaSet**

# Delete Service

kubectl delete svc <service-name>

# Sample Commands

kubectl delete svc my-helloworld-rs-service

[or]

kubectl delete svc/my-helloworld-rs-service

# Verify if Service got deleted

kubectl get svc