Kubernetes Task2

Date: 27/05/24

Q.1 Make a note on:

a. **Pod**
b. **Replica**
c. **ReplicaSet**
d. **Labels**
e. **Namespace**

1. **Pod:**

- A pod is the smallest deployable unit in Kubernetes.
- It represents a single instance of a running process in your cluster.
- Pods can contain one or more containers that are tightly coupled and share resources, such as networking and storage.
- They are ephemeral by nature, meaning they can be created, destroyed, and replaced dynamically.

2. **Replicas:**

- Replicas refer to the number of identical copies of a pod that should be running at any given time.
- They are used to ensure high availability and scalability of applications.
- Replicas are typically defined in higher-level Kubernetes objects like ReplicaSets or Deployments.

3. **ReplicaSet:**

- A ReplicaSet ensures that a specified number of pod replicas are running at any given time.
- It acts as a higher-level abstraction over pods, managing their lifecycle and ensuring the desired number of replicas is maintained.
- ReplicaSets are generally used to achieve scaling and self-healing capabilities for stateless applications.

4. **Labels:**

- Labels are key-value pairs attached to Kubernetes objects such as pods, services, and deployments.

- They are used to organize and select subsets of objects based on user-defined criteria.
- Labels are highly flexible and can be used for various purposes including grouping, filtering, and identifying related resources within a cluster.

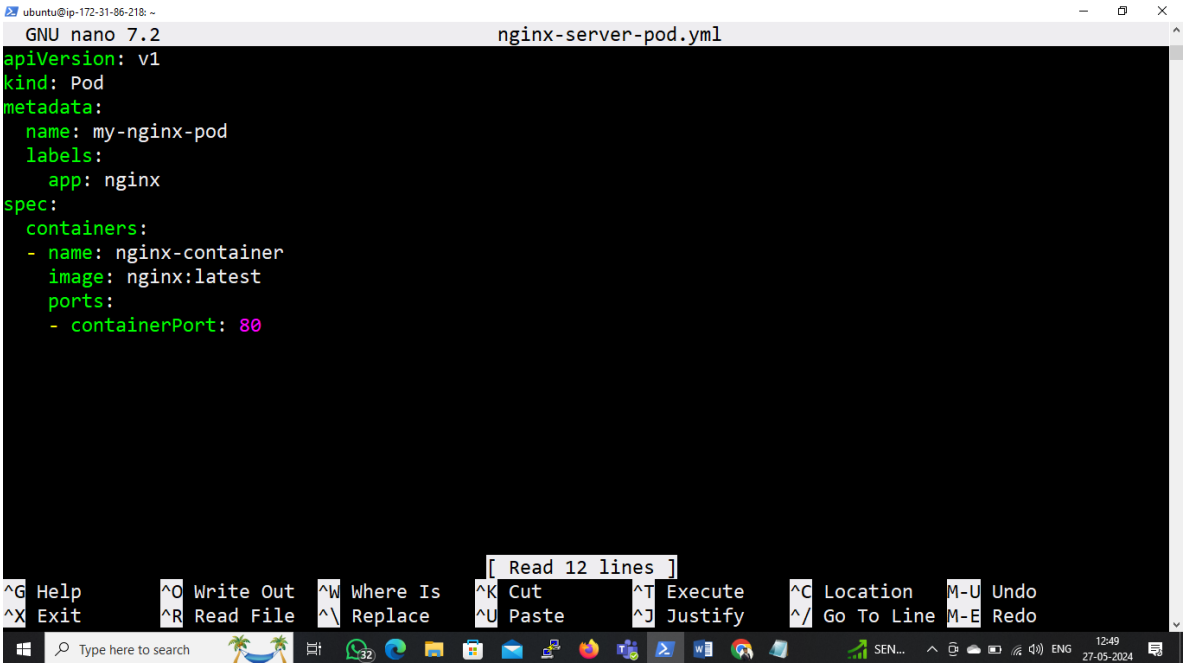
5. **Namespace:**
   - Namespaces provide a way to logically divide cluster resources into virtual clusters within the same physical cluster.
   - They are primarily used to create isolated environments for different teams, projects, or applications, allowing each to have its own scope of resources such as pods, services, and storage volumes.
   - Namespaces also help in resource management, access control, and multi-tenancy scenarios in Kubernetes clusters.

Q.2 Show in practical of RC uses with all types of health probes and their file.

To demonstrate the use of ReplicaController (RC) with various types of health probes in Kubernetes.

Create a simple example with a Pod running a my-nginx-pod.

Below is the YAML manifest file for the Pod configuration:(nginx-server-pod.yml)



- kubectl apply -f nginx-server-pod.yml
- kubectl get pods

```
ubuntu@ip-172-31-86-218: ~
mypod-cg99x   1/1     Terminating   0          2d20h
mypod-fs7zc   1/1     Terminating   0          2m39s
mypod-j6lwr   1/1     Terminating   0          2d20h
mypod-jq489   1/1     Terminating   0          2d20h
mypod-sz86l   1/1     Terminating   0          2d20h
mypod-th7vr   1/1     Terminating   0          2d20h
ubuntu@ip-172-31-86-218:~$ kubectl delete all --all  --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been terminate
d. The resource may continue to run on the cluster indefinitely.
pod "mypod-75dtm" force deleted
pod "mypod-cg99x" force deleted
pod "mypod-fs7zc" force deleted
pod "mypod-j6lwr" force deleted
pod "mypod-jq489" force deleted
pod "mypod-sz86l" force deleted
pod "mypod-th7vr" force deleted
service "kubernetes" force deleted
ubuntu@ip-172-31-86-218:~$ kubectl get pods
No resources found in default namespace.
ubuntu@ip-172-31-86-218:~$ kubectl apply -f nginx-server-pod.yml
pod/my-nginx-pod created
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
my-nginx-pod   1/1     Running   0          5s
ubuntu@ip-172-31-86-218:~$
```

We'll use different types of health probes:

livenessProbe, readinessProbe, and startupProbe.

First create a ReplicaController file;

eg nginx-server-pod-rc.yml.

```
ubuntu@ip-172-31-86-218: ~
  GNU nano 7.2                    nginx-server-pod-rc.yml
apiVersion: v1
kind: ReplicationController
metadata:
  name: mypod
spec:
  replicas: 3
  template:
    metadata:
      labels:
        env: test
    spec:
      containers:
        - image: nginx
          name: mynginx
          ports:
            - containerPort: 80

                              [ Read 16 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line M-E Redo
```

Apply this file;

- kubectl apply -f nginx-server-pod-rc.yml
- kubectl get pods

```
service "kubernetes" force deleted
ubuntu@ip-172-31-86-218:~$ kubectl get pods
No resources found in default namespace.
ubuntu@ip-172-31-86-218:~$ kubectl apply -f nginx-server-pod.yml
pod/my-nginx-pod created
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
my-nginx-pod   1/1     Running   0          5s
ubuntu@ip-172-31-86-218:~$ nano nginx-server-pod-rc.yml
ubuntu@ip-172-31-86-218:~$ nano nginx-server-pod-rc.yml
ubuntu@ip-172-31-86-218:~$ kubectl apply -f nginx-server-pod-rc.yml
replicationcontroller/mypod created
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
my-nginx-pod   1/1     Running   0          11m
mypod-2ktz5    1/1     Running   0          7s
mypod-g9rfp    1/1     Running   0          7s
mypod-h2jv7    1/1     Running   0          7s
ubuntu@ip-172-31-86-218:~$
```

Use of LivenessProbe:

- **Liveness Probe:**

➢ **Purpose:** Determines whether the container in a Pod is running properly.
➢ **Functionality:** Periodically checks if the container is responsive and restarts it if it's not.
➢ **Typical Use Case:** Used to detect and recover from application-specific issues such as deadlocks or resource exhaustion that cause the container to become unresponsive.
➢ **Action on Failure:** If the liveness probe fails, Kubernetes restarts the container.

➢ Create a manifestfile of nginx-server-pod-rc-liveness.yml

```
  GNU nano 7.2                    nginx-server-pod-rc-liveness.yml
apiVersion: v1
kind: ReplicationController
metadata:
  name: liveness-pod
spec:
  replicas: 5
  template:
    metadata:
      labels:
        env: test
    spec:
      containers:
        - image: nginx
          name: mynginx
          ports:
            - containerPort: 80
          livenessProbe:
            httpGet:
              path: index.html
              port: 80
            intitialDelaySeconds: 10
                                  [ Read 22 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line M-E Redo
```

- kubectl apply -f nginx-server-pod-rc-liveness.yml
- kubectl get pods.

```
ubuntu@ip-172-31-86-218: ~
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
my-nginx-pod    1/1     Running   0          11m
mypod-2ktz5     1/1     Running   0          7s
mypod-g9rfp     1/1     Running   0          7s
mypod-h2jv7     1/1     Running   0          7s
ubuntu@ip-172-31-86-218:~$ nano nginx-server-pod-rc-liveness.yml
ubuntu@ip-172-31-86-218:~$ ubuntu@ip-172-31-86-218:~$
ubuntu@ip-172-31-86-218:~$ kubectl apply -f nginx-server-pod-rc.yml
replicationcontroller/mypod unchanged
ubuntu@ip-172-31-86-218:~$ nano nginx-server-pod-rc-liveness.yml
ubuntu@ip-172-31-86-218:~$ kubectl apply -f nginx-server-pod-rc-liveness.yml
replicationcontroller/liveness-pod created
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME                 READY   STATUS    RESTARTS   AGE
liveness-pod-5bnfq   1/1     Running   0          6s
liveness-pod-ds2tf   1/1     Running   0          6s
liveness-pod-jms6s   1/1     Running   0          6s
liveness-pod-q2pr5   1/1     Running   0          6s
liveness-pod-rjgf2   1/1     Running   0          6s
my-nginx-pod         1/1     Running   0          19m
mypod-2ktz5          1/1     Running   0          8m8s
mypod-g9rfp          1/1     Running   0          8m8s
mypod-h2jv7          1/1     Running   0          8m8s
ubuntu@ip-172-31-86-218:~$
```

Use of ReadinessProbe:

- **Readiness Probe:**

- **Purpose:** Determines whether the container in a Pod is ready to serve traffic.
- **Functionality:** Periodically checks if the container is ready to receive requests and tells Kubernetes whether the Pod should receive traffic.
- **Typical Use Case:** Used to ensure that only healthy Pods receive traffic from services or load balancers. It helps avoid sending requests to Pods that are still initializing or experiencing issues.
- **Action on Failure:** If the readiness probe fails, the Pod is removed from service endpoints until it becomes ready again.

Create a manifestfile **of** nginx-server-pod-rc-readiness.yml

```
ubuntu@ip-172-31-86-218: ~
  GNU nano 7.2                    nginx-server-pod-rc-readiness.yml
apiVersion: v1
kind: ReplicationController
metadata:
  name: readiness-pod
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
        readinessProbe:
          httpGet:
            path: index.html
                                        [ Read 24 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line M-E Redo
```

- kubectl apply -f nginx-server-pod-rc-readiness.yml
- kubectl get pods

```
ubuntu@ip-172-31-86-218: ~
liveness-pod-7grns    0/1     Terminating   0          15s
liveness-pod-kg8zb    0/1     Terminating   0          15s
liveness-pod-vg8b7    0/1     Terminating   0          15s
liveness-pod-vw449    0/1     Terminating   0          15s
mypod-rn9xv           0/1     Terminating   0          15s
mypod-snw2f           0/1     Terminating   0          15s
readiness-pod-ft6v9   0/1     Terminating   0          15s
ubuntu@ip-172-31-86-218:~$ kubectl get pods
No resources found in default namespace.
ubuntu@ip-172-31-86-218:~$ kubectl apply -f nginx-server-pod-rc-readiness.yml
replicationcontroller/readiness-pod created
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME                  READY   STATUS              RESTARTS   AGE
readiness-pod-d97jz   0/1     ContainerCreating   0          3s
readiness-pod-sn2nw   0/1     ContainerCreating   0          3s
readiness-pod-t7gvj   0/1     ContainerCreating   0          3s
ubuntu@ip-172-31-86-218:~$ kubectl apply -f nginx-server-pod-rc-readiness.yml
replicationcontroller/readiness-pod unchanged
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
readiness-pod-d97jz   1/1     Running   0          28s
readiness-pod-sn2nw   1/1     Running   0          28s
readiness-pod-t7gvj   1/1     Running   0          28s
ubuntu@ip-172-31-86-218:~$ nano nginx-server-pod-rc-readiness.yml
ubuntu@ip-172-31-86-218:~$
```

Enter in pod **readiness-pod-d97jz**

- kubectl exec -it readiness-pod-d97jz /bin/bash

 # Remove index.html page and when probe check health of pods they will get un-healthy, and mark as not ready which shows in below screenshot;

```
ubuntu@ip-172-31-86-218: ~
replicationcontroller/readiness-pod unchanged
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
readiness-pod-d97jz   1/1     Running   0          28s
readiness-pod-sn2nw   1/1     Running   0          28s
readiness-pod-t7gvj   1/1     Running   0          28s
ubuntu@ip-172-31-86-218:~$ nano nginx-server-pod-rc-readiness.yml
ubuntu@ip-172-31-86-218:~$ kubectl exec -it readiness-pod-d97jz /bin/bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [P
OD] -- [COMMAND] instead.
root@readiness-pod-d97jz:/# cd /usr/share/nginx/html/
root@readiness-pod-d97jz:/usr/share/nginx/html# ls
50x.html  index.html
root@readiness-pod-d97jz:/usr/share/nginx/html# rm index.html
root@readiness-pod-d97jz:/usr/share/nginx/html# exit
exit
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
readiness-pod-d97jz   0/1     Running   0          9m4s
readiness-pod-sn2nw   1/1     Running   0          9m4s
readiness-pod-t7gvj   1/1     Running   0          9m4s
ubuntu@ip-172-31-86-218:~$ kubectl get rc
NAME            DESIRED   CURRENT   READY   AGE
readiness-pod   3         3         2       9m43s
ubuntu@ip-172-31-86-218:~$
```

Use of StartupProbe:

- **Startup Probe:**

➢ **Purpose:** Determines whether the container in a Pod has started successfully.
➢ **Functionality:** Runs probes during the initial startup of the container, delaying the readiness check until the application inside the container has started.
➢ **Typical Use Case:** Used for applications with a long startup time or complex initialization process. It allows Kubernetes to wait until the application is fully up and running before sending traffic to the Pod.
➢ **Action on Failure:** If the startup probe fails, the Pod is treated as failed, similar to how a liveness probe failure is handled.

Create a manifestfile of nginx-server-pod-rc-startup.yml

```
ubuntu@ip-172-31-86-218: ~
  GNU nano 7.2                    nginx-server-pod-rc-startup.yml
apiVersion: v1
kind: ReplicationController
metadata:
  name: startup-pod
spec:
  replicas: 5
  template:
    metadata:
      labels:
        env: test
    spec:
      containers:
        - image: nginx
          name: mynginx
          ports:
          - containerPort: 80
          startupProbe:
            httpGet:
              path: index.html
              port: 80

                                [ Read 20 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line M-E Redo
```

- kubectl apply -f nginx-server-pod-rc-startup.yml
- kubectl get pods

```
ubuntu@ip-172-31-86-218: ~
ubuntu@ip-172-31-86-218:~$ nano nginx-server-pod-rc-startup.yml
ubuntu@ip-172-31-86-218:~$ kubectl apply -f nginx-server-pod-rc-startup.yml
replicationcontroller/startup-pod created
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME                  READY   STATUS              RESTARTS   AGE
readiness-pod-d97jz   0/1     Running             0          44m
readiness-pod-sn2nw   1/1     Running             0          44m
readiness-pod-t7gvj   1/1     Running             0          44m
startup-pod-8pqkk     0/1     ContainerCreating   0          5s
startup-pod-ktbsr     0/1     ContainerCreating   0          5s
startup-pod-wbshd     0/1     ContainerCreating   0          5s
startup-pod-wqm46     0/1     ContainerCreating   0          5s
startup-pod-xjt8q     0/1     ContainerCreating   0          5s
ubuntu@ip-172-31-86-218:~$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
readiness-pod-d97jz   0/1     Running   0          44m
readiness-pod-sn2nw   1/1     Running   0          44m
readiness-pod-t7gvj   1/1     Running   0          44m
startup-pod-8pqkk     1/1     Running   0          33s
startup-pod-ktbsr     1/1     Running   0          33s
startup-pod-wbshd     1/1     Running   0          33s
startup-pod-wqm46     1/1     Running   0          33s
startup-pod-xjt8q     1/1     Running   0          33s
ubuntu@ip-172-31-86-218:~$ nano nginx-server-pod-rc-startup.yml
ubuntu@ip-172-31-86-218:~$
```

- In summary, liveness probes ensure that the **container is running correctly**, readiness probes ensure that the **container is ready to serve traffic**, and startup probes **delay the readiness check until the container has started successfully**.
- Each probe type serves a specific purpose in managing the lifecycle and health of applications running in Kubernetes Pods.