Lab1: Creating a pod

Introduction:

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.

A *Pod* is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers.

Objectives:

1. Creating a pod using Imperative command
2. Creating a pod using a YAML file
3. Accessing the pod
4. Deleting the pods

Note: We will be using the commands only on the master node.

1. **Creating a pod using Imperative command:**

   Here we will be creating a simple pod named nginx-pod with nginx image. Use the below command to create a desired pod.

   ```
   Kubectl run nginx-pod –image nginx
   ```

   Now use the below command to check the pod status

   ```
   Kubectl get pods
   ```

   It will show us the pods which are running in the current Namespace. Currently, it will show us the pods in the default namespace.

   ```
   root@master:~# kubectl run nginx-pod --image nginx
   pod/nginx-pod created
   root@master:~# kubectl get pods
   NAME        READY   STATUS    RESTARTS   AGE
   nginx-pod   1/1     Running   0          6s
   ```

   Further to check the more information about the pod we will give us information about the IP address of the pod and where it has been deployed.

   ```
   kubectl get pods -o wide
   ```

   ```
   root@master:~# kubectl get pods -o wide
   NAME        READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
   nginx-pod   1/1     Running   0          4m56s   192.168.235.161   worker1   <none>           <none>
   ```

Above we can see that the pod with IP address 192.168.235.161 has been deployed on worker1 node.
We can describe a particular pod to get the detailed information.

```
kubectl describe pod nginx-pod
```

```
root@master:~# kubectl describe pod nginx-pod
Name:         nginx-pod
Namespace:    default
Priority:     0
Node:         worker1/172.31.0.70
Start Time:   Wed, 11 Jan 2023 08:44:58 +0000
Labels:       run=nginx-pod
Annotations:  cni.projectcalico.org/containerID: d16b82b85e37363406552a8e02b75adeac35773800d554c909790fbe63338be8
              cni.projectcalico.org/podIP: 192.168.235.161/32
              cni.projectcalico.org/podIPs: 192.168.235.161/32
Status:       Running
IP:           192.168.235.161
IPs:
  IP:  192.168.235.161
Containers:
  nginx-pod:
    Container ID:   docker://843c329ef71dd5d730ae55925c7342343998cb2cd50f760fb4b241392cb96cf6
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:4b2e2e4192a2d9fc83c8eb57b070b89307be48a840db6dc50476f852d1768ba5
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Wed, 11 Jan 2023 08:45:00 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-g5vgg (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-g5vgg:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  12m   default-scheduler  Successfully assigned default/nginx-pod to worker1
  Normal  Pulling    12m   kubelet            Pulling image "nginx"
  Normal  Pulled     12m   kubelet            Successfully pulled image "nginx" in 226.792429ms
  Normal  Created    12m   kubelet            Created container nginx-pod
  Normal  Started    12m   kubelet            Started container nginx-pod
```

Above we can get the information which the pod has received by default. We have used a single command but the pod has received the default information like the container name, volume and volume mounts etc.
We can also witness how the pod has been created under the Events field. It shows that default scheduler has selected worker1 node for this pod and further Kubelet will take the help of container runtime engine to pull the image. Kubelet will further create a pod on this node.

2. **Creating a pod using a YAML file:**

We can create a pod using a simple YAML file where we can mention the details as per our requirement. We can use an imperative command to create a template of a pod and can make some changes. Use the below command to create a template of a pod and save it in a file with .yaml or .yml format.

```
kubectl run httpd-pod --image httpd --dry-run=client -o yaml > httpd.yaml
```

The above command will not create a pod instead it will get the output in yaml format and we are further storing it in httpd.yaml file. We can see below the template it has created for us.

```
root@master:~# kubectl run httpd-pod --image httpd --dry-run=client -o yaml > httpd.yaml
root@master:~# cat httpd.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: httpd-pod
  name: httpd-pod
spec:
  containers:
  - image: httpd
    name: httpd-pod
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

Now open the file httpd.yaml in vim editor and make the changes. Below some changes have been done like another label has been added and the name of the container has been changed to container1.

```
apiVersion: v1
kind: Pod
metadata:
  labels:
      run: httpd-pod
      env: prod
  name: httpd-pod
spec:
   containers:
    - image: httpd
      name: container1
```

Save and exit the above file. Now we need to implement the changes. Use the below command to create a pod based on above definition file.

```
kubectl apply -f httpd.yaml
```

Now check the status as shown below.

```
root@master:~# kubectl apply -f httpd.yaml
pod/httpd-pod created
root@master:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
httpd-pod   1/1     Running   0          5s
nginx-pod   1/1     Running   0          48m
```

If we further describe the httpd-pod then we can see that our changes (container name and label) have been implemented.

```
root@master:~# kubectl describe pod httpd-pod
Name:         httpd-pod
Namespace:    default
Priority:     0
Node:         worker2/172.31.0.82
Start Time:   Wed, 11 Jan 2023 09:33:07 +0000
Labels:       env=prod
              run=httpd-pod
Annotations:  cni.projectcalico.org/containerID: 5fd0100044833ee6976267906253864546ea97eb0a6f980a0aabd7b8c60afb41
              cni.projectcalico.org/podIP: 192.168.189.105/32
              cni.projectcalico.org/podIPs: 192.168.189.105/32
Status:       Running
IP:           192.168.189.105
IPs:
  IP:  192.168.189.105
Containers:
  container1:
    Container ID:   docker://50e96f025a2cad0b1b12e503527f7a28d35592473e9c9646f60df91e6acafba3
    Image:          httpd
    Image ID:       docker-pullable://httpd@sha256:eb44faad041d2cde46389a286a4dd11e42d99f5e874eb554a24c87fd8f1cce0b
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Wed, 11 Jan 2023 09:33:09 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-zc2g7 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-zc2g7:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  114s  default-scheduler  Successfully assigned default/httpd-pod to worker2
  Normal  Pulling    113s  kubelet            Pulling image "httpd"
  Normal  Pulled     112s  kubelet            Successfully pulled image "httpd" in 347.423558ms
  Normal  Created    112s  kubelet            Created container container1
  Normal  Started    112s  kubelet            Started container container1
```

3. Accessing the pod

We can access the pod using the exec command and install the packages inside the pod. Currently we are having only one container so by default exec will make us go inside the first container. Use the below command to go inside the pod.

```
kubectl exec -it nginx-pod -- /bin/bash
```

We can see below we are inside the pod and further updated the **apt package** and installed the **iproute2** package. We have also checked the IP address of the container (IP address of the pod as container shares Pod's IP address).

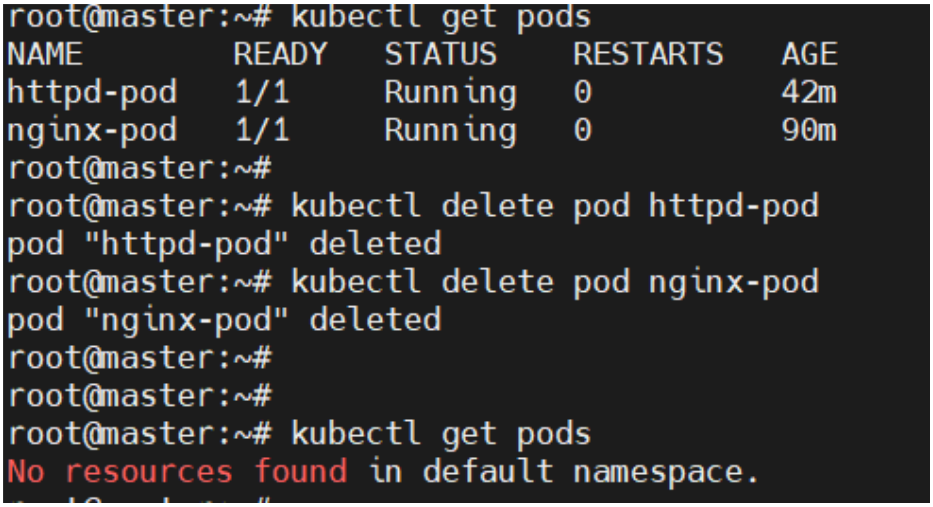
```
root@master:~# kubectl exec -it nginx-pod -- /bin/bash
root@nginx-pod:/#
root@nginx-pod:/# apt update -y && apt install iproute2
Hit:1 http://deb.debian.org/debian bullseye InRelease
Hit:2 http://deb.debian.org/debian-security bullseye-security InRelease
Hit:3 http://deb.debian.org/debian bullseye-updates InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
iproute2 is already the newest version (5.10.0-4).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@nginx-pod:/#
root@nginx-pod:/#
root@nginx-pod:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
4: eth0@if14: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8981 qdisc noqueue state UP group default
    link/ether 26:21:dd:73:4f:af brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.235.161/32 scope global eth0
       valid_lft forever preferred_lft forever
```

4. Deleting the pods:

Use the below command to delete the pods.

```
kubectl delete pod httpd-pod
kubectl delete pod nginx-pod
```

The above commands will delete the specified pod from our cluster and we can see the result below.


root@master:~# kubectl get pods
NAME READY STATUS RESTARTS AGE
httpd-pod 1/1 Running 0 42m
nginx-pod 1/1 Running 0 90m
root@master:~#
root@master:~# kubectl delete pod httpd-pod
pod "httpd-pod" deleted
root@master:~# kubectl delete pod nginx-pod
pod "nginx-pod" deleted
root@master:~#
root@master:~#
root@master:~# kubectl get pods
No resources found in default namespace.