Lesson 01 Demo 03

Configuring Pods in the Kubernetes Cluster

Objective: To configure pods in a Kubernetes cluster, encompass pod setup, create service files, and execute Apache services to enhance containerized application management within Kubernetes

Tools required: kubeadm, kubectl, kubelet, and containerd

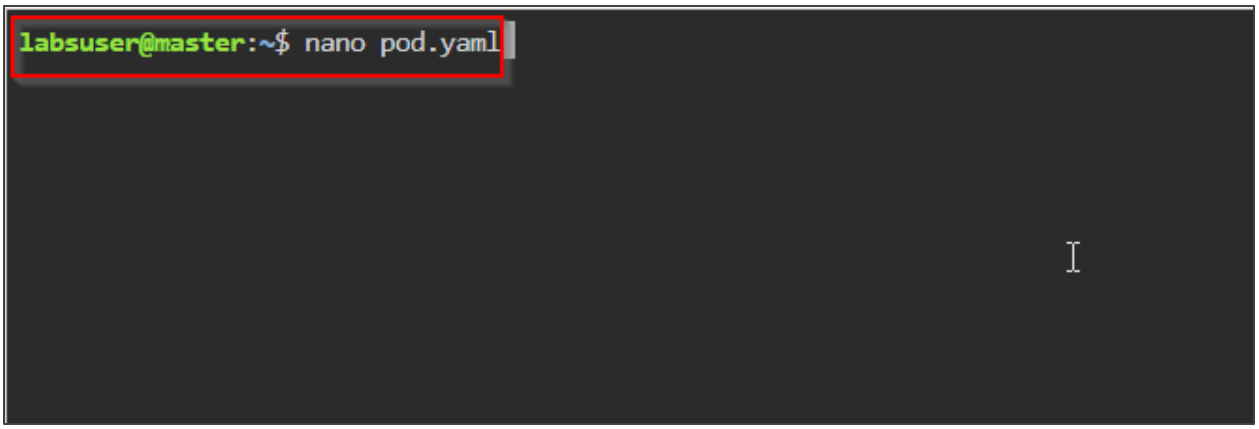
Prerequisites: A Kubernetes cluster (refer to Demo 01 from Lesson 01 for setting up a cluster)

Steps to be followed:

1. Configure and set up the pod files
2. Configure and set up the service file
3. Execute the Apache services

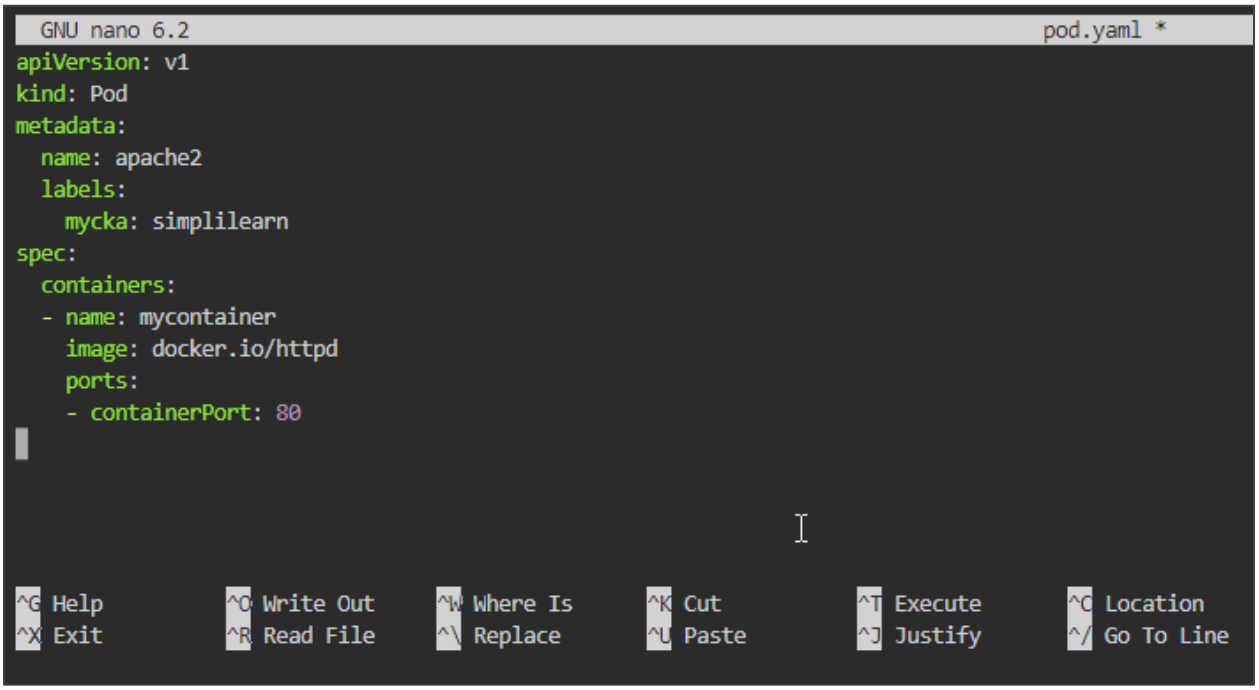
Step 1: Configure and set up the pod files

1.1 Create the YAML file by using the following command:
nano pod.yaml



1.2 Add the following code to the **pod.yaml** file to create the pod, save the file by pressing **ctrl+S**, and exit with **ctrl+X**:

```
apiVersion: v1
kind: Pod
metadata:
  name: apache2
  labels:
    mycka: simplilearn
spec:
  containers:
  - name: mycontainer
    image: docker.io/httpd
    ports:
    - containerPort: 80
```



1.3 Use the **cat** command to validate the content of the **pod.yaml** file

```
labsuser@master:~$ nano pod.yaml
labsuser@master:~$ cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: apache2
  labels:
    mycka: simplilearn
spec:
  containers:
  - name: mycontainer
    image: docker.io/httpd
    ports:
    - containerPort: 80
labsuser@master:~$
```

1.4 Create the pod resource using the following command:
kubectl create -f pod.yaml

```
labsuser@master:~$ nano pod.yaml
labsuser@master:~$ cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: apache2
  labels:
    mycka: simplilearn
spec:
  containers:
  - name: mycontainer
    image: docker.io/httpd
    ports:
    - containerPort: 80
labsuser@master:~$ kubectl create -f pod.yaml
pod/apache2 created
labsuser@master:~$
```

1.5 Create another pod file by using the following command:
nano pod1.yaml

```
labsuser@master:~$ nano pod.yaml
labsuser@master:~$ cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: apache2
  labels:
    mycka: simplilearn
spec:
  containers:
  - name: mycontainer
    image: docker.io/httpd
    ports:
    - containerPort: 80
labsuser@master:~$ kubectl create -f pod.yaml
pod/apache2 created
labsuser@master:~$ nano pod1.yaml
```

1.6 Add the following code to the **pod1.yaml** file to create the pod and then save and exit the editor:

```
apiVersion: v1
kind: Pod
metadata:
  name: apache3
  labels:
    mycka: simplilearn
spec:
  containers:
  - name: mycontainer
    image: docker.io/httpd
    ports:
    - containerPort: 80
```

1.7 Use the **cat** command to validate the content of the **pod1.yaml** file

```
    image: docker.io/httpd
    ports:
    - containerPort: 80
labsuser@master:~$ kubectl create -f pod.yaml
pod/apache2 created
labsuser@master:~$ nano pod1.yaml
labsuser@master:~$ cat pod1.yaml
apiVersion: v1
kind: Pod
metadata:
  name: apache3
  labels:
    mycka: simplilearn
spec:
  containers:
  - name: mycontainer
    image: docker.io/httpd
    ports:
    - containerPort: 80
labsuser@master:~$
```

1.8 Create the pod resource by using the following command:
kubectl create -f pod1.yaml

```
    - containerPort: 80
labsuser@master:~$ kubectl create -f pod.yaml
pod/apache2 created
labsuser@master:~$ nano pod1.yaml
labsuser@master:~$ cat pod1.yaml
apiVersion: v1
kind: Pod
metadata:
  name: apache3
  labels:
    mycka: simplilearn
spec:
  containers:
  - name: mycontainer
    image: docker.io/httpd
    ports:
    - containerPort: 80
labsuser@master:~$ kubectl create -f pod1.yaml
pod/apache3 created
labsuser@master:~$
```

1.9 Verify pods creation by using the following command:
kubectl get pods

```
labsuser@master:~$ cat pod1.yaml
apiVersion: v1
kind: Pod
metadata:
  name: apache3
  labels:
    mycka: simplilearn
spec:
  containers:
  - name: mycontainer
    image: docker.io/httpd
    ports:
    - containerPort: 80
labsuser@master:~$ kubectl create -f pod1.yaml
pod/apache3 created
labsuser@master:~$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
apache2   1/1     Running   0          10m
apache3   1/1     Running   0          72s
labsuser@master:~$
```

Step 2: Configure and set up the service file

2.1 Create the YAML file by using the following command:
nano service.yaml

```
labsuser@master:~$ kubectl create -f pod1.yaml
pod/apache3 created
labsuser@master:~$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
apache2   1/1     Running   0          10m
apache3   1/1     Running   0          72s
labsuser@master:~$ nano  service.yaml
labsuser@master:~$
```

2.2 Add the following code to the **service.yaml** file to create the pod and then save and exit the editor:

```
kind: Service
apiVersion: v1
metadata:
  name: myservice
spec:
  selector:
      mycka: simplilearn
  ports:
    - protocol: TCP
      port: 8081
      targetPort: 80
```

2.3 Create the resource for **service.yaml** by using the following command:

kubectl create -f service.yaml

```
metadata:
  name: apache3
  labels:
    mycka: simplilearn
spec:
  containers:
  - name: mycontainer
    image: docker.io/httpd
    ports:
    - containerPort: 80
labsuser@master:~$ kubectl create -f pod1.yaml
pod/apache3 created
labsuser@master:~$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
apache2   1/1     Running   0          10m
apache3   1/1     Running   0          72s
labsuser@master:~$ nano  service.yaml
labsuser@master:~$ kubectl create -f service.yaml
service/myservice created
labsuser@master:~$
```

2.4 Verify the service by using the following command:

kubectl get svc

```
spec:
  containers:
  - name: mycontainer
    image: docker.io/httpd
    ports:
    - containerPort: 80
labsuser@master:~$ kubectl create -f pod1.yaml
pod/apache3 created
labsuser@master:~$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
apache2   1/1     Running   0          10m
apache3   1/1     Running   0          72s
labsuser@master:~$ nano  service.yaml
labsuser@master:~$ kubectl create -f service.yaml
service/myservice created
labsuser@master:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP    79m
myservice    ClusterIP   10.101.183.1   <none>        8081/TCP   75s
labsuser@master:~$
```

Step 3: Execute the Apache services

3.1 Access the container in pod **apache2** and change the content in **htdocs/index.html** by using the following commands:
kubectl exec -it apache2 bash
echo “Hello from pod1 ” > htdocs/index.html
cat htdocs/index.html
exit

```
labsuser@master:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP    79m
myservice    ClusterIP   10.101.183.1   <none>        8081/TCP   75s
labsuser@master:~$ kubectl exec -it apache2 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@apache2:/usr/local/apache2# echo \342\200\234Hello from pod1 \342\200\235 > htdocs/index.html
root@apache2:/usr/local/apache2# cat htdocs/index.html
“Hello from pod1 ”
root@apache2:/usr/local/apache2# exit
exit
labsuser@master:~$
```

3.2 Access the container in pod **apache3** and change the content in **htdocs/index.html** by using the following commands:
kubectl exec -it apache3 bash
echo “Hello from pod2 ” > htdocs/index.html
cat htdocs/index.html
exit

```
labsuser@master:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP    79m
myservice    ClusterIP   10.101.183.1   <none>        8081/TCP   75s
labsuser@master:~$ kubectl exec -it apache2 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@apache2:/usr/local/apache2# echo \342\200\234Hello from pod1 \342\200\235 > htdocs/index.html
root@apache2:/usr/local/apache2# cat htdocs/index.html
“Hello from pod1 ”
root@apache2:/usr/local/apache2# exit
exit
labsuser@master:~$ kubectl exec -it apache3 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@apache3:/usr/local/apache2# echo \342\200\234Hello from pod2 \342\200\235 > htdocs/index.html
root@apache3:/usr/local/apache2# cat htdocs/index.html
“Hello from pod2 ”
root@apache3:/usr/local/apache2# exit
exit
labsuser@master:~$
```

3.3 Validate if the **myservice** service is connected to **apache2** and **apache3** by using the following command:
kubectl get svc -o wide

```
myservice    ClusterIP   10.101.183.1   <none>        8081/TCP   75s
labsuser@master:~$ kubectl exec -it apache2 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@apache2:/usr/local/apache2# echo \342\200\234Hello from pod1 \342\200\235 > htdocs/index.html
root@apache2:/usr/local/apache2# cat htdocs/index.html
“Hello from pod1 ”
root@apache2:/usr/local/apache2# exit
exit
labsuser@master:~$ kubectl exec -it apache3 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@apache3:/usr/local/apache2# echo \342\200\234Hello from pod2 \342\200\235 > htdocs/index.html
root@apache3:/usr/local/apache2# cat htdocs/index.html
“Hello from pod2 ”
root@apache3:/usr/local/apache2# exit
exit
labsuser@master:~$ kubectl get svc -o wide
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE     SELECTOR
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP    84m     <none>
myservice    ClusterIP   10.101.183.1   <none>        8081/TCP   6m28s   mycka=simplilearn
labsuser@master:~$
```

3.4 Copy the IP and port number and write them in the following format:
curl <ClusterIP:PortNumber>

```
“Hello from pod1 ”
root@apache2:/usr/local/apache2# exit
exit
labsuser@master:~$ kubectl exec -it apache3 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@apache3:/usr/local/apache2# echo \342\200\234Hello from pod2 \342\200\235 > htdocs/index.html
root@apache3:/usr/local/apache2# cat htdocs/index.html
“Hello from pod2 ”
root@apache3:/usr/local/apache2# exit
exit
labsuser@master:~$ kubectl get svc -o wide
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE     SELECTOR
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP    84m     <none>
myservice    ClusterIP   10.101.183.1   <none>        8081/TCP   6m28s   mycka=simplilearn
```

Note: Initially, execute the **curl** command only for the service named **myservice** using the **curl 10.101.183.1:8081** command. However, if you do not see both the services running and messages displayed for pods **apache2** and **apache3**, execute **Kubernetes** and **myservice**, as shown in the next step.

3.5 Execute the **curl** command to complete the task, as shown in the screenshot below:

```
labsuser@master:~$ curl 10.96.0.1:443
Client sent an HTTP request to an HTTPS server.
labsuser@master:~$ curl 10.101.183.1:8081
"Hello from pod1 "
labsuser@master:~$ curl 10.101.183.1:8081
"Hello from pod2 "
labsuser@master:~$
```

By following these steps, you have successfully completed the configuration of pods in a cluster, service file creation, and Apache service execution to enhance the management of containerized applications within Kubernetes.