.

Lesson 03 Demo 03

Configuring a Pod Using an Init Container

Objective: To create and configure a pod using an init container to design more complex and flexible workflows for Kubernetes applications

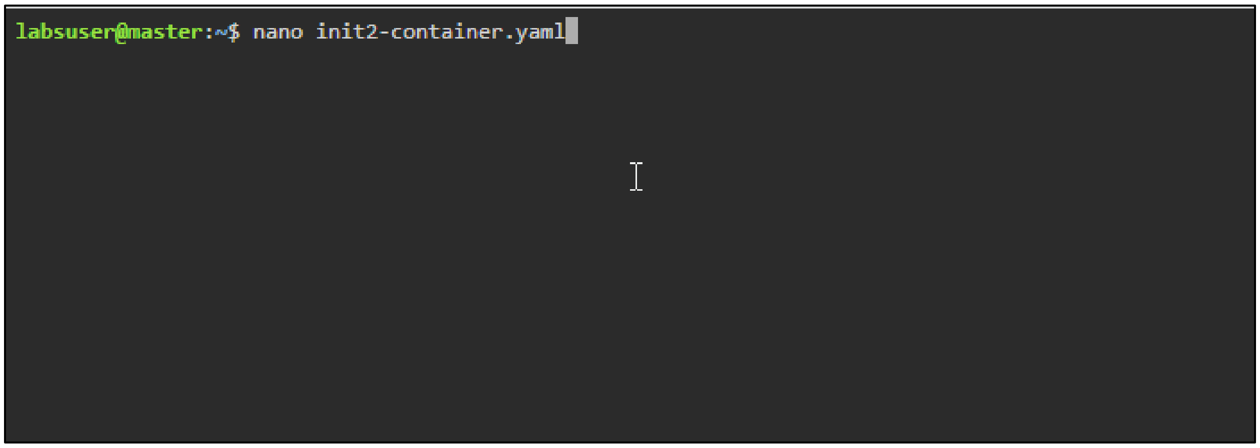
Tools required: kubeadm, kubectl, kubelet, and containerd

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

Steps to be followed:
1. Create a pod
2. Create services
3. Verify the pod's state

Step 1: Create a pod

1.1 On the master node, enter the command **nano init2-container.yaml** to create a YAML file



```
labsuser@master:~$ nano init2-container.yaml
```

.

1.2 Add the following code in the YAML file:

apiVersion: v1

kind: Pod

metadata:

name: myapp-pod

labels:

app: myapp

spec:

containers:

- name: myapp-container

image: registry.access.redhat.com/ubi8/ubi:latest

command: ['sh', '-c', 'echo The app is running! && sleep 3600']

initContainers:

- name: init-myservice

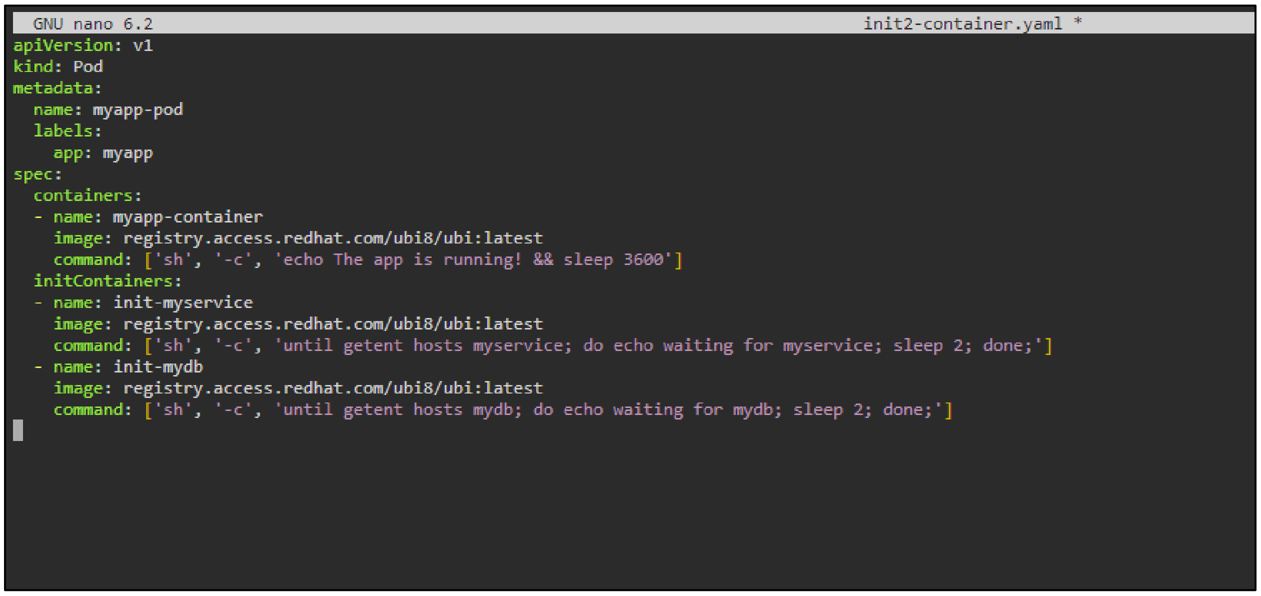
image: registry.access.redhat.com/ubi8/ubi:latest

command: ['sh', '-c', 'until getent hosts myservice; do echo waiting for myservice; sleep 2; done;']

- name: init-mydb

image: registry.access.redhat.com/ubi8/ubi:latest

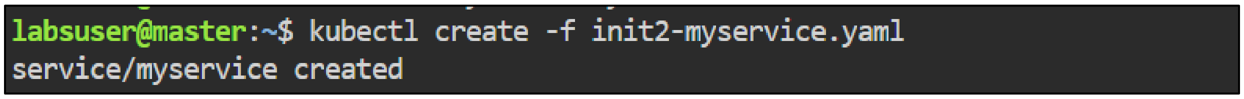
command: ['sh', '-c', 'until getent hosts mydb; do echo waiting for mydb; sleep 2; done;']



```
  GNU nano 6.2                                                                      init2-container.yaml *
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: registry.access.redhat.com/ubi8/ubi:latest
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
  - name: init-myservice
    image: registry.access.redhat.com/ubi8/ubi:latest
    command: ['sh', '-c', 'until getent hosts myservice; do echo waiting for myservice; sleep 2; done;']
  - name: init-mydb
    image: registry.access.redhat.com/ubi8/ubi:latest
    command: ['sh', '-c', 'until getent hosts mydb; do echo waiting for mydb; sleep 2; done;']
```

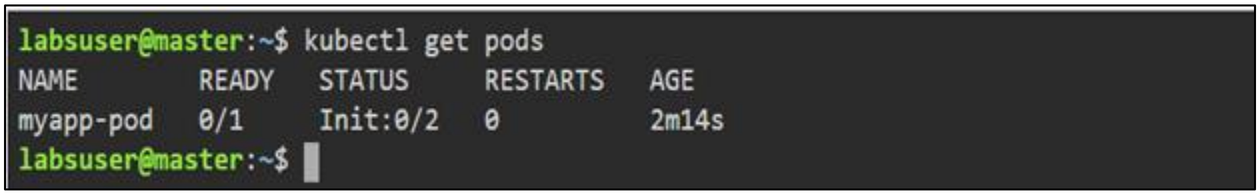
.

1.3 Create a pod using the following command:
kubectl create -f init2-container.yaml



```
labsuser@master:~$ kubectl create -f init2-myservice.yaml
service/myservice created
```

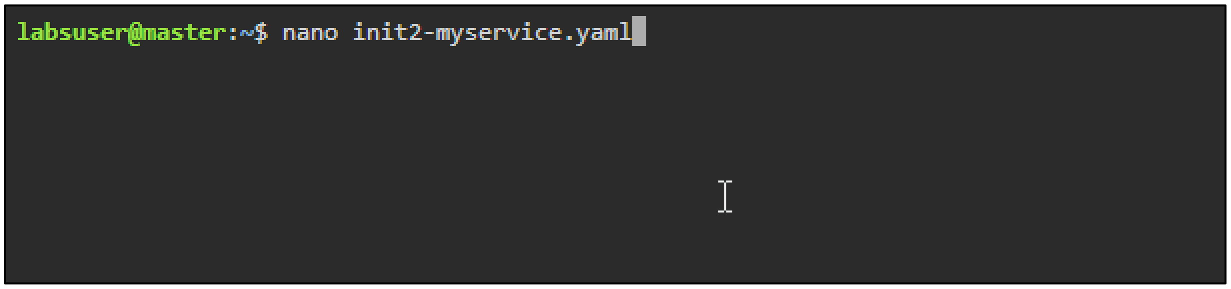
1.4 Verify the pod’s state using the following command:
kubectl get pods



```
labsuser@master:~$ kubectl get pods
NAME        READY   STATUS     RESTARTS   AGE
myapp-pod   0/1     Init:0/2   0          2m14s
labsuser@master:~$
```

Step 2: Create services

2.1 For the first service, create the **init2-myservice.yaml** file using the following command:
nano init2-myservice.yaml



```
labsuser@master:~$ nano init2-myservice.yaml
```

.

2.2 Add the following code in the YAML file:

kind: Service

apiVersion: v1

metadata:

name: myservice

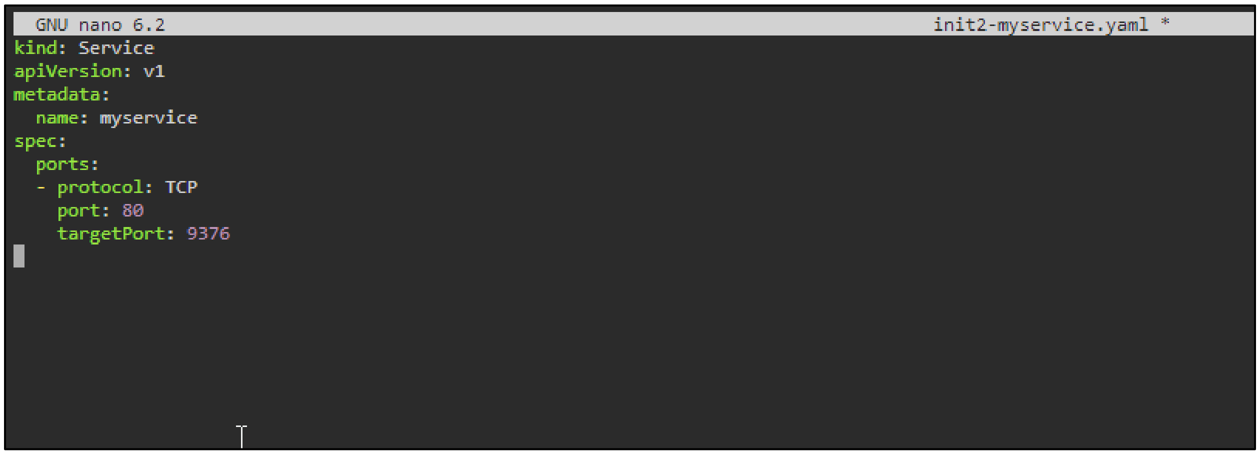
spec:

ports:

- protocol: TCP

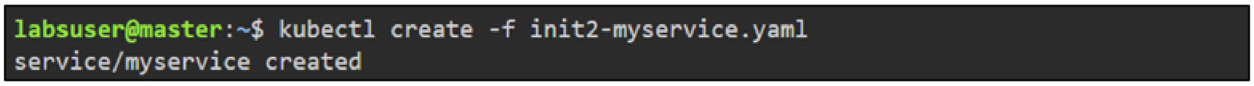
port: 80

targetPort: 9376



```
  GNU nano 6.2                                                          init2-myservice.yaml *
kind: Service
apiVersion: v1
metadata:
  name: myservice
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

2.3 Run the following command to create the first service named **myservice**:
kubectl create -f init2-myservice.yaml

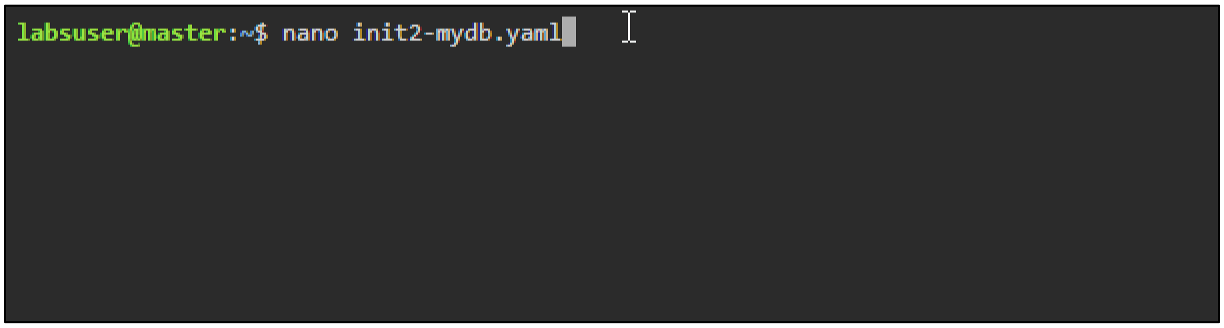


```
labsuser@master:~$ kubectl create -f init2-myservice.yaml
service/myservice created
```

The first service is created successfully.

.

2.4 For the second service, create the **init2-mydb.yaml** file using the following command:
nano init2-mydb.yaml



```
labsuser@master:~$ nano init2-mydb.yaml
```

2.5 Enter the following code in the YAML file:

kind: Service

apiVersion: v1

metadata:

name: mydb

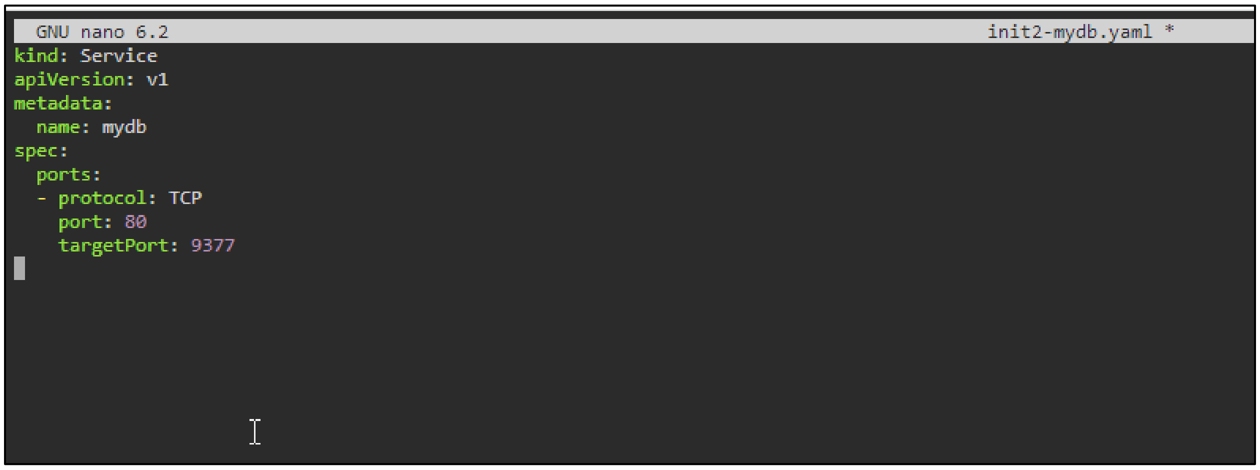
spec:

ports:

- protocol: TCP

port: 80

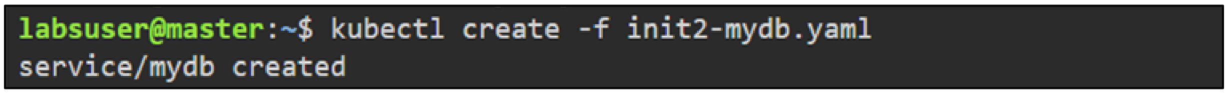
targetPort: 9377



```
  GNU nano 6.2                                                        init2-mydb.yaml *
kind: Service
apiVersion: v1
metadata:
  name: mydb
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9377
```

.

2.6 Run the following command to create the second service named **mydb**:
kubectl create -f init2-mydb.yaml

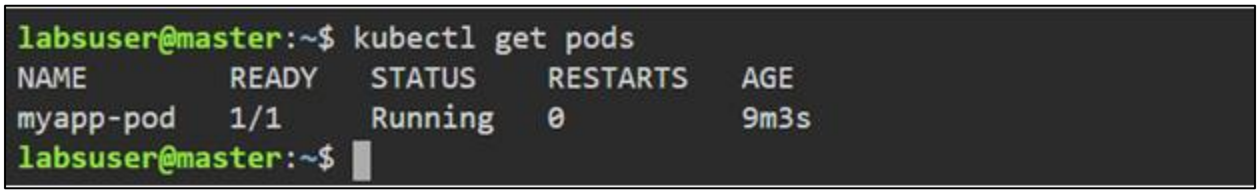


```
labsuser@master:~$ kubectl create -f init2-mydb.yaml
service/mydb created
```

The second service is created successfully.

Step 3: Verify the pod's state

3.1 Run the following command to verify the state of the pod:
kubectl get pods



```
labsuser@master:~$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
myapp-pod    1/1     Running   0          9m3s
labsuser@master:~$
```

You can see that the pod is running.

By following these steps, you have successfully configured a pod using an init container.