Lesson 03 Demo 09

Deploying and Verifying Kubernetes Objects

Objective: To create a Kubernetes deployment and verify the integrity of its associated pod and deployment objects

Tools required: kubeadm, kubectl, kubelet, and containerd

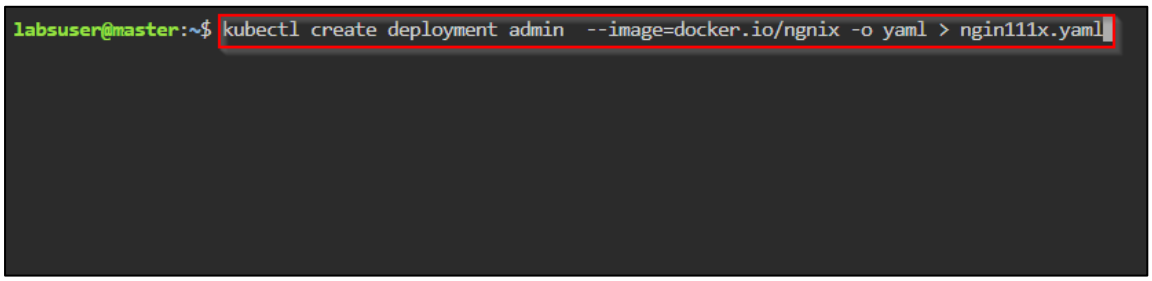
Prerequisites: A Kubernetes cluster should already be set up (refer to the steps in Lesson 02, Demo 01 for guidance).

Steps to be followed:

1. Create and verify the deployment and pod objects

Step 1: Create and verify the deployment and pod objects

1.1 To create a deployment and save its configuration, execute the following command:
kubectl create deployment admin --image=docker.io/nginx -o yaml > ngin111x.yaml



1.2 Enter the following command to get pods list, and use the pod name to get pods details

kubectl get pods

kubectl get pod <pod-name> -o yaml

Ex: **kubectl get pods admin-7bbbcdbbdc-jz9q9 -o yaml**

```
labsuser@master:~$ kubectl get pods
NAME                     READY   STATUS    RESTARTS      AGE
admin-7bbbcdbbdc-jz9q9   1/1     Running   0             6m23s
test-pod                 1/1     Running   3 (13m ago)   8d
labsuser@master:~$ admin-6dcbc5b5f4-9zp8j
admin-6dcbc5b5f4-9zp8j: command not found
labsuser@master:~$ kubectl get pods admin-7bbbcdbbdc-jz9q9 -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    cni.projectcalico.org/containerID: 243ec6658375d37b6b467f3b7525c4e93b2942ef23e782bcb265a801eff69dcc
    cni.projectcalico.org/podIP: 192.168.181.77/32
    cni.projectcalico.org/podIPs: 192.168.181.77/32
  creationTimestamp: "2023-11-04T07:09:30Z"
  generateName: admin-7bbbcdbbdc-
  labels:
    app: admin
    pod-template-hash: 7bbbcdbbdc
  name: admin-7bbbcdbbdc-jz9q9
  namespace: default
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: ReplicaSet
    name: admin-7bbbcdbbdc
    uid: 071e0657-3179-47ae-a22e-c68e95f2eb2a
  resourceVersion: "31158"
  uid: cf08318c-7f1d-41d7-ac3e-bba236eaa7d5
spec:
  containers:
  - image: docker.io/nginx
    imagePullPolicy: Always
    name: nginx
```

1.3 To expose the deployment and create an associated service, run the following:

kubectl expose deployment admin --port=80

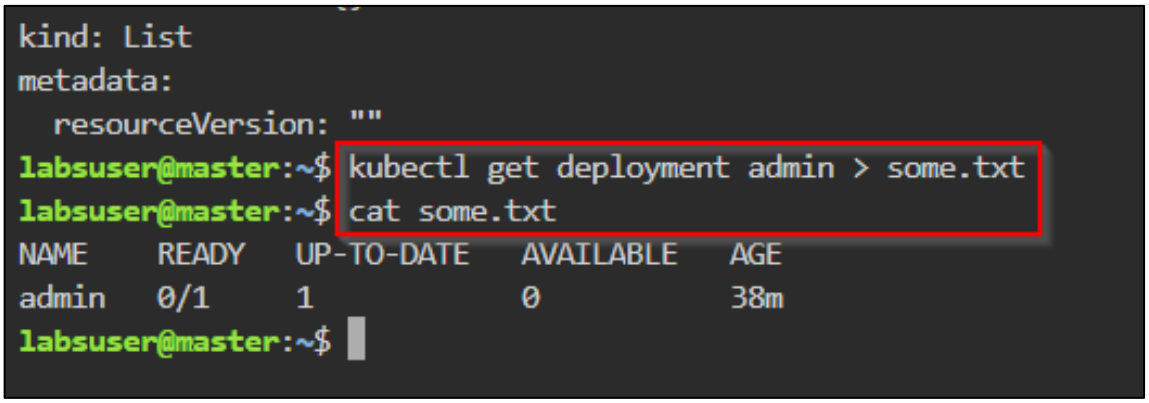
```
labsuser@master:~$ kubectl expose deployment admin --port=80
service/admin exposed
labsuser@master:~$
```

1.4 To view the details of the service object you have created, run the following command:

kubectl get svc admin -o yaml

```
labsuser@master:~$ kubectl expose deployment admin --port=80
service/admin exposed
labsuser@master:~$ kubectl get svc admin -o yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2023-11-04T07:21:56Z"
  labels:
    app: admin
  name: admin
  namespace: default
  resourceVersion: "32199"
  uid: 727e3c75-bb15-4dfe-bd0c-ef3d42967e36
spec:
  clusterIP: 10.100.55.7
  clusterIPs:
  - 10.100.55.7
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: admin
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
labsuser@master:~$
```

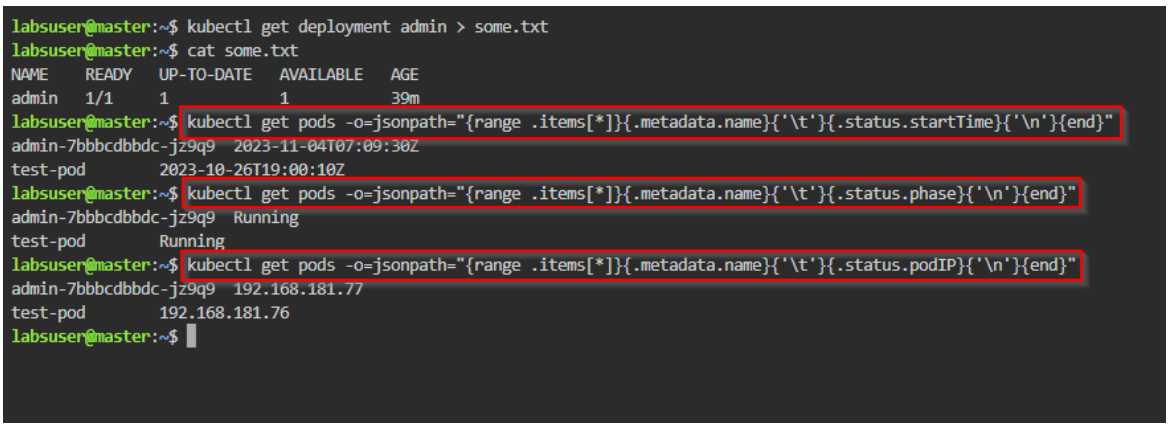
1.5 To retrieve details about the **admin** deployment and store them in a text file, execute the following commands:
kubectl get deployment admin > some.txt
cat some.txt



1.6 JSONPath queries are useful for extracting specific fields from a JSON object. Here are the correct commands to use for:
Start time: **kubectl get pods -o=jsonpath="{range .items[*]}{.metadata.name}{'\t'}{.status.startTime}{'\n'}{end}"**

Pod status: **kubectl get pods -o=jsonpath="{range .items[*]}{.metadata.name}{'\t'}{.status.phase}{'\n'}{end}"**

Pod IP: **kubectl get pods -o=jsonpath="{range .items[*]}{.metadata.name}{'\t'}{.status.podIP}{'\n'}{end}"**



By following these steps, you have successfully managed and verified Kubernetes deployments and their related objects.