EXERCISES ON KUBERNETES

1. Create a simple deployment of the given app with name of your choice and 3 replicas of pods. Check the status of pod by sending request. App should be accessed from outside the cluster.

dep.yaml

Note: Replace usn with your USN starting as "ms" i.e. exclude "1" from your USN

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: usn-nginx-deployment
  namespace: usn
  labels:
    app: usn-nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: usn-nginx
  template:
    metadata:
      labels:
        app: usn-nginx
    spec:
      containers:
      - name: nginx
        image: 172.1.14.168:5001/nginx
        ports:
        - containerPort: 80
```

Command to create name space:
kubectl create namespace ms99cs001

Command to deploy:
kubectl apply -f dep.yaml

Command to check pods:
kubectl get pods --namespace=ms99cs001

Command to expose
kubectl expose deployment usn-nginx-deployment --type=NodePort --name=usn-nginx-service --namespace=ms99cs001

To get exposed port:
kubectl get svc --namespace=ms99cs001
Open the browser and type :
http://172.1.14.168:<NodePort>

2. Demonstrate the updation of image in live container in a pod using command line.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: usn-nginx-deployment
  namespace: usn
  labels:
    app: usn-nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: usn-nginx
  template:
    metadata:
      labels:
        app: usn-nginx
    spec:
      containers:
      - name: nginx
        image: 172.1.14.168:5001/nginx
        ports:
        - containerPort: 80
```

kubectl set image deployment/usn-nginx-deployment nginx=newImageusn --namespace=ms99cs001

kubectl describe deploy usn-nginx-deployment --namespace=ms99cs001 | grep newImageusn

3. Perform the following.
    1. Create 3 pods with names nginx1, nginx2,nginx3. All of them should have the label app=v1 Show all labels of the pods.
    2. Get only the 'app=v2' pods.
    3. Remove the 'app' label from the pods we created before

```
kubectl run ms99cs001-nginx1 --image=nginx --restart=Never --labels=app=ms99cs001-v1
--namespace=ms99cs001
kubectl run ms99cs001-nginx2 --image=nginx --restart=Never --labels=app=ms99cs001-v1
--namespace=ms99cs001
kubectl run ms99cs001-nginx3 --image=nginx --restart=Never --labels=app=ms99cs001-v1
--namespace=ms99cs001
kubectl get po --show-labels  --namespace=ms99cs001
kubectl get po -l app=ms99cs001-v2 --namespace=ms99cs001
kubectl label po ms99cs001-nginx1 ms99cs001-nginx2 ms99cs001-nginx3 app- --
namespace=ms99cs001
```

4. Create a Pod with ubuntu image and a command to echo “YOUR_NAME” which overrides the default CMD/ENTRYPOINT of the image. Delete pod.

dep_ubuntu_pod1.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu
  namespace: usn
  labels:
    app: ubuntu
spec:
  containers:
  - name: ubuntu
    image: 172.1.14.168:5001/ubuntu
    command: ["/bin/bash"]
    args: ["-c", "echo MSRIT"]
```

```
kubectl apply -f dep_ubuntu_pod1.yaml
kubectl logs ubuntu --namespace=usn
kubectl delete pod ubuntu –namespace=usn
```

5. Create a Pod that runs one container. The configuration file for the Pod defines a command and arguments by using environment variables and Delete pod.

dep_ubuntu_pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: ubuntunew
  namespace: usn
  labels:
    app: ubuntunew
spec:
  containers:
  - name: ubuntunew
    image: 172.1.14.168:5001/ubuntu
    env:
      - name: MESSAGE
        value: "Hello MSRIT"
    command: ["/bin/echo"]
    args: ["$(MESSAGE)"]
```

```
kubectl apply -f dep_ubuntu_pod.yaml
kubectl logs ubuntunew --namespace=usn
kubectl delete pod ubuntunew -namespace=usn
```

6. Create a manifest file for creating a pod with 2 containers. Demonstrate container to container communication and display “Ubuntu” container log entry.

`pod_nw.yaml`

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod
spec:
  containers:
    - name: c00
      image: 172.1.14.168:5001/ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello; sleep 5; done"]
    - name: c01
      image: 172.1.14.168:5001/httpd
      ports:
        - containerPort: 80
```

```
kubectl apply --namespace usn -f pod_nw.yml
kubectl get pods --namespace usn
kubectl exec --namespace usn testpod -it -c c00 -- /bin/bash
apt update
apt install curl
curl localhost:80
exit
kubectl logs testpod -c c00 --namespace usn
```

7. Create 2 Pods on the same node. Demonstrate Pod to Pod communication through Pod IP.

`pod1_nw.yaml`

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod1
spec:
  containers:
    - name: c02
      image: 172.1.14.168:5001/nginx
      ports:
        - containerPort: 80
```

`pod2_nw.yaml`

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod2
spec:
  containers:
    - name: c03
      image: 172.1.14.168:5001/httpd
      ports:
        - containerPort: 80
```

```
kubectl apply --namespace usn -f pod1_nw.yml
kubectl apply --namespace usn -f pod2_nw.yml

kubectl get pod testpod1 --namespace usn -o custom-
columns=NAME:metadata.name,IP:status.podIP

kubectl exec --namespace usn testpod1 -it -c c00 -- /bin/bash
apt update
apt install curl
curl testpod1:80
curl localhost:80
curl <tespod2_ip>:80

kubectl exec --namespace usn testpod2 -it -c c03 -- /bin/bash
apt update
apt install curl
curl <testpod1_ip>:80
```