Pod Design

This lab consists of a list of exercises to demonstrate and understand the most commonly used kubernetes commands and concepts to ramp up your kubernetes competency skills.

Learning Outcomes

After completing the lab, you will be able to understand and use Kubernetes concepts related to the below topics:

1. Labels, Selectors, Annotations
2. Deployments
3. Services
4. Persistent Volumes

Start the minikube

1. Start minikube locally `minikube start --driver=virtualbox`
2. Verify the kubectl context `kubectl config get-contexts` is set to minikube. If not, set it to minikube `kubectl config use-context minikube`

Create all manifest resources in the directory `~/workspace/kubernetes-manifests/competencies`. Watch out for the right file names in the solution section.

Pod Design

Labels, Selectors and Annotations

1. Add a label `tier=service` to the pages application and display the label

   ▼ Click to see solution

   `~/workspace/kubernetes-manifests/competencies/pod-design/1.yaml`

   ```
   apiVersion: v1
   kind: Pod
   metadata:
   ```

```
  labels:
    run: pages
    tier: service
  name: pages
spec:
  containers:
    - image: dellcloud/pages:1.0
      imagePullPolicy: IfNotPresent
      name: pages
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/pod-design/1.yaml
```

```
kubectl get po pages --show-labels
```

2. Create a pod nginx and add a label `tier=frontend`

▼ Click to see solution

`~/workspace/kubernetes-manifests/competencies/pod-design/2.yaml`

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
    tier: frontend
  name: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      imagePullPolicy: IfNotPresent
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/pod-design/2.yaml
```

```
kubectl get po nginx --show-labels
```

3. Get all the pods with label `tier=frontend`

▼ Click to see solution

```
kubectl get po -l tier=frontend --show-labels
```

4. Fetch and delete all the pods which has label tier set to any value

▼ Click to see solution

```
kubectl get po -l tier --show-labels
kubectl delete po -l tier
```

5. Annotate nginx pod with annotation `team=yourteam` and `course=k8s`

▼ Click to see solution

`~/workspace/kubernetes-manifests/competencies/pod-design/3.yaml`

```yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    team: yourteam
    course: k8s
  labels:
    run: nginx
  name: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      imagePullPolicy: IfNotPresent
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/pod-design/3.yaml
kubectl describe po nginx
```

```
kubectl get po nginx -o jsonpath='{.metadata.annotations.c
ourse}'
kubectl get po nginx -o jsonpath='{.metadata.annotations.t
eam}'
```

```
kubectl delete po nginx
```

Deployments

1. Create pages deployment with 2 replicas using yaml manifest file

▼ Click to see solution

`~/workspace/kubernetes-manifests/competencies/pod-design/4.yaml`

```yaml
apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  labels:
    app: pages
  name: pages
spec:
  replicas: 2
  selector:
    matchLabels:
      app: pages
  strategy: {}
  template:
    metadata:
      labels:
        app: pages
    spec:
      containers:
      - image: dellcloud/pages:1.0
        name: pages
        imagePullPolicy: IfNotPresent
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/pod-design/4.yaml
```

```
kubectl get deployment pages
```

```
kubectl get po
```

2. Verify the first version of deployment has been deployed and rolled out

▼ Click to see solution

```
kubectl rollout status deployment pages
```

3. Update the manifest to use 3 replicas and set the image to `dellcloud/pages:service`

▼ Click to see solution

`~/workspace/kubernetes-manifests/competencies/pod-design/5.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: pages
  name: pages
spec:
  replicas: 3
  selector:
    matchLabels:
      app: pages
  strategy: {}
  template:
    metadata:
      labels:
        app: pages
    spec:
      containers:
      - image: dellcloud/pages:service
        name: pages
        imagePullPolicy: IfNotPresent
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competencies/pod-design/5.yaml
```

```
kubectl get deployment pages
```

```
kubectl get po
```

4. Verify the second version of deployment has been deployed and rolled out

▼ Click to see solution

```
kubectl rollout status deployment pages
```

```
kubectl rollout history deployment pages
```

```
kubectl rollout history deployment pages --revision=2
```

5. Rollback to the previous version. Notice the change by inspecting the deployment & the revision number

▼ Click to see solution

```
kubectl describe deployment pages | grep -i image
kubectl rollout undo deployment pages
kubectl describe deployment pages | grep -i image
kubectl rollout history deployment pages
```

6. Manually scale out/up to use 5 replicas and inspect the number of pods.

▼ Click to see solution

```
kubectl scale deployment pages --replicas=5
kubectl get po -w
```

7. Manually scale in/down to use 1 replicas and inspect the number of pods.

▼ Click to see solution

```
kubectl scale deployment pages --replicas=1
kubectl get po -w
```

```
kubectl delete deploy pages
```

Services

1. Before starting the next set of exercises, create the directory `services` inside `competencies`

```
mkdir ~/workspace/kubernetes-manifests/competencies/servic
es
```

2. Create a service which routes the request to `nginx` pod using selectors in the yaml file.

▼ Click to see solution

`~/workspace/kubernetes-manifests/competencies/pod-design/2.yaml`

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
    tier: frontend
  name: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      imagePullPolicy: IfNotPresent
```

`~/workspace/kubernetes-manifests/competencies/services/1.yaml`

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
    tier: frontend
  type: ClusterIP
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competencies/services/1.yaml
kubectl apply -f ~/workspace/kubernetes-manifests/competencies/pod-design/2.yaml
```

```
kubectl get svc nginx -o wide
kubectl get ep
kubectl get po nginx --show-labels
```

```
kubectl port-forward svc/nginx 8080:8080
```

```
curl localhost:8080
```

```
kubectl delete po nginx
```

```
kubectl delete svc nginx
```

3. Create a service (color) of type `nodeport` and expose `port 8080` and `target port 80` , with the selector app=colorful. Create appropriate `nginx` pod to be accessible by the service.

▼ Click to see solution

`~/workspace/kubernetes-manifests/competencies/services/2-pod.yaml`

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: colorful
  name: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      imagePullPolicy: IfNotPresent
```

~/workspace/kubernetes-manifests/competencies/services/2.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: colorful
  name: color
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 80
  selector:
    app: colorful
  type: NodePort
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/services/2.yaml
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/services/2-pod.yaml
```

```
kubectl get svc color -o wide
kubectl get ep
kubectl get po nginx --show-labels
```

```
kubectl port-forward svc/color 8080:8080
```

```
curl localhost:8080
```

```
kubectl delete po nginx
```

```
kubectl delete svc color
```

4. Create a service named `nginx` of type `NodePort` which routes the traffic to any pod which has the label `app=colorful` exposing `target port 80` on `port 80`. The service does not route traffic to any pods yet. We shall use it in later exercises.

▼ Click to see solution

```
mkdir -p ~/workspace/kubernetes-manifests/competencies/services/green
```

`~/workspace/kubernetes-manifests/competencies/services/green/service.yaml`

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: colorful
  name: nginx
spec:
  type: NodePort
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: colorful
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competencies/services/green/service.yaml

kubectl get svc nginx
kubectl get ep
```

5. Create a `nginx-green` pod deriving from `nginx` image, which prints a custom message `Green` (HINT:: echo Green > /usr/share/nginx/html/index.html) instead of its original message, having the label `app=colorful` & Create a `nginx-blue` pod deriving from `nginx` image, which prints a custom message `Blue` (HINT:: echo Blue > /usr/share/nginx/html/index.html) instead of its original message having the label `app=colorful` Make multiple requests to the `nginx` service created in the previous exercise and ensure that the service is routing requests between the two different pods. Delete the pods and services before moving on to the next exercise.

▼ Click to see solution

```
cd ~/workspace/kubernetes-manifests/competencies/services/green
```

`~/workspace/kubernetes-manifests/competencies/services/green/Dockerfile`

```
FROM nginx:latest
RUN echo Green > /usr/share/nginx/html/index.html
```

```
docker build -t [docker-username]/nginx:green .
docker push [docker-username]/nginx:green
```

```
mkdir ~/workspace/kubernetes-manifests/competencies/services/blue
cd ~/workspace/kubernetes-manifests/competencies/services/blue
```

`~/workspace/kubernetes-manifests/competencies/services/blue/Dockerfile`

```
FROM nginx:latest
RUN echo Blue > /usr/share/nginx/html/index.html
```

```
docker build -t [docker-username]/nginx:blue .
docker push [docker-username]/nginx:blue
```

`~/workspace/kubernetes-manifests/competencies/services/green/pod.yaml`

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: colorful
  name: nginx-green
spec:
  containers:
  - image: dellcloud/nginx:green
    name: nginx-green
    imagePullPolicy: IfNotPresent
```

`~/workspace/kubernetes-manifests/competencies/services/blue/pod.yaml`

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: colorful
  name: nginx-blue
spec:
  containers:
  - image: dellcloud/nginx:blue
    name: nginx-blue
    imagePullPolicy: IfNotPresent
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/services/green/pod.yaml
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/services/blue/pod.yaml

kubectl get all
```

```
kubectl get svc nginx
```

The next set of instructions are provided for minikube. But if you want to run it on production cluster you will have to provide appropriate security policies for inbound access on the node port

Copy the 5 digit NODEPORT under the column PORT(S)

```
kubectl get nodes -o wide
```

Copy the INTERNAL-IP

```
curl http://INTERNAL-IP:NODEPORT
**Run the command for a few times to understand the concep
t of kubernetes service discovery**
```

Clean up

```
kubectl delete po -l app=colorful
kubectl delete svc -l app=colorful
```

6. Create 2 deployments of the nginx application from the previous exercise. The first deployment is `nginx-blue` application, scaled to 3 replicas, and the second deployment is a single replica of `nginx-green` application. Create the service, which will forward network requests to any pod with the label `app=colorful`

▼ Click to see solution

```
cd ~/workspace/kubernetes-manifests/competencies/services/
green
```

~/workspace/kubernetes-manifests/competencies/services/green/deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: colorful
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: colorful
  template:
    metadata:
      labels:
        app: colorful
    spec:
      containers:
      - image: dellcloud/nginx:green
        name: nginx
        imagePullPolicy: IfNotPresent
```

~/workspace/kubernetes-manifests/competencies/services/blue/deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: colorful
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: colorful
  template:
    metadata:
      labels:
        app: colorful
    spec:
      containers:
      - image: dellcloud/nginx:blue
```

```
        name: nginx
        imagePullPolicy: IfNotPresent
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/services/green/deployment.yaml
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/services/blue/deployment.yaml
```

Let's reuse the service created in the previous exercise as it serves our purpose

```
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/services/green/service.yaml
```

```
kubectl get all --show-labels
kubectl get svc nginx
```

Copy the 5 digit NODEPORT under the column PORT(S)

```
kubectl get nodes -o wide
```

Copy the INTERNAL-IP

```
curl http://INTERNAL-IP:NODEPORT
```

Run the command few times and watch how the Kubernetes service automatically load balances our request between the running pods

Persistent Volumes

1. Create a Persistent Volume which is used as a long term storage solution. Create a Persistent Volume Claim to use the persistent volume. Create a pod that defines an application container which writes the current date to a log file every five seconds and this pod will eventually use persistent volume claim when mounting the log file to persistent volume.

▼ Click to see solution
`~/workspace/kubernetes-manifests/competencies/volumes/pv-1.yaml`

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: log-pv-[student-name]
  labels:
    type: local
spec:
  storageClassName: document
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/etc/kal-directory"
```

~/workspace/kubernetes-manifests/competencies/volumes/pvc-1.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: log-pvc-[student-name]
spec:
  storageClassName: document
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

~/workspace/kubernetes-manifests/competencies/volumes/pod-1.yaml

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: alpine
  name: alpine
spec:
  volumes:
    - name: log-date-vol
      persistentVolumeClaim:
            claimName: log-pvc-[student-name]
  containers:
  - image: alpine
    name: alpine
    imagePullPolicy: IfNotPresent
    command: ["/bin/sh"]
    args: ["-c", "while true; do date >> /etc/kal-director
y/date-file.txt; sleep 5; done"]
    volumeMounts:
      - name: log-date-vol
        mountPath: /etc/kal-directory
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/volumes/pv-1.yaml
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/volumes/pvc-1.yaml
kubectl apply -f ~/workspace/kubernetes-manifests/competen
cies/volumes/pod-1.yaml
```

```
kubectl get po alpine
```

```
kubectl exec -it alpine -- cat /etc/kal-directory/date-fil
e.txt
```

```
kubectl delete po alpine
kubectl delete pvc log-pvc-[student-name]
kubectl delete pv log-pv-[student-name]
```

2. Re-design the pod that was created in Multicontainer section: Exercise 3, such that the written files will be mounted on Persistent Volume. Create required Persistent Volume and Persistent Volume Claim. Storage capacity of Persistent Volume shoud not exceed 500M

▼ Click to see Notes

Create `~/workspace/kubernetes-manifests/competencies/volumes/pod-2.yaml` file with the below content and **modify the manifest to use PV & PVC**

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: multi-container-pod
  name: multi-container-pod
spec:
  volumes:
    - name: shared-vol
      emptyDir: {}
  containers:
    - image: ubuntu
      name: ubuntu
      imagePullPolicy: IfNotPresent
      command: ["/bin/sh"]
      args: ["-c", "while true; do date > /logs/output.tx
t; free -tw --giga >> /logs/output.txt; sleep 10; done"]
      volumeMounts:
        - name: shared-vol
```

```
          mountPath: /logs
    - image: alpine
      name: alpine
      imagePullPolicy: IfNotPresent
      command: ["/bin/sh"]
      args: ["-c", "while true; do echo  'Date: ' $(cat /logs/output.txt |  head -1) > /logs/report.txt; echo 'Total Memory:' $(cat /logs/output.txt |  grep Total: | tr -s ' ' | cut -d ' ' -f 2) GB >> /logs/report.txt; echo 'Free Memory:' $(cat /logs/output.txt |  grep Total: | tr -s ' ' | cut -d ' ' -f 3) GB >> /logs/report.txt; sleep 10; done"]
      volumeMounts:
        - name: shared-vol
          mountPath: /logs
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competencies/volumes/pod-2.yaml
```

```
kubectl get po multi-container-pod
```

```
kubectl exec -it multi-container-pod -c alpine -- cat /logs/report.txt
```

```
kubectl delete po  multi-container-pod
```