Resource Quotas

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 in the area of `Resource Quotas`

Learning Outcomes

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

1. Resource Quota
2. Namespaces
3. Service Accounts for security

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.

Namespaces & ResourceQuotas

All the exercises in this section has to be completed in minikube & not on production cluster

1. Create a namespace called `[student-name]-test` either imperatively or declaratively.

   ▼ Click to see solution

   ```
   kubectl create namespace [student-name]-test
   ```

After creating it, when you try to create k8s resources inside the cluster, which namespace will it get created in? Lets create an `nginx` pod

```
kubectl run nginx --image=nginx --image-pull-policy=IfNotPresent
```

```
kubectl get pods
```

Do you see the pod?

```
kubectl get pods --all-namespaces
```

Was there an issue? Discuss with your pair.

```
kubectl delete po nginx
```

```
kubectl delete namespace [student-name]-test
```

a. What did you understand? Everytime you create a new namespace, you have to point the `kubectl context` to that particular namespace. Without doing that, you might end up creating it in a difference namespace.

b. What is the workaround? You can pass `--namespace` or `-n` argument while creating the resource by explicitly specifying the namespace.

2. Create a resource quota in the namespace `[student-name]` with the below requirement.

```
pods: 5
"requests.cpu": "2"
"requests.memory": 1024m
"limits.cpu": "4"
"limits.memory": 2048m
```

▼ Click to see solution

`~/workspace/kubernetes-manifests/competencies/resource-quota.yaml`

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: resource-quota
  namespace: [student-name]
spec:
  hard:
    pods: 5
    "requests.cpu": "2"
    "requests.memory": 1024Mi
    "limits.cpu": "4"
    "limits.memory": 2048Mi
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competencies/resource-quota.yaml
kubectl get resourcequota
```

Create an `nginx` pod within `[student-name]`

```
kubectl run nginx --image=nginx --image-pull-policy=IfNotPresent
```

Do you see an error while trying to create the pod?

Error from server (Forbidden): pods "nginx" is forbidden: failed quota: resource-quota: must specify cpu,memory

This time try to create the pod using the manifest file. Take the help of `kubectl explain` command to see the options for specifying *resource requests* & *limits* for a pod. Modify the manifest accordingly and create the pod.

`~/workspace/kubernetes-manifests/competencies/pod-with-resource-quota.yaml`

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
  namespace: [student-name]
spec:
  containers:
  - image: nginx
    imagePullPolicy: IfNotPresent
    name: nginx
    resources:
      requests:
        memory: 256Mi
        cpu: "0.5"
      limits:
        memory: 1024Mi
        cpu: "1"
  restartPolicy: Always
```

```
kubectl apply -f ~/workspace/kubernetes-manifests/competencies/pod-with-resource
-quota.yaml
```

Inspect the status of the pod and the resource quota

```
kubectl get po
kubectl get quota
kubectl describe quota resource-quota
```

Clean up!

```
kubectl delete po nginx
kubectl delete quota resource-quota
```

ServiceAccount

1. Create a secret called `docker-registry` of type `generic` in the namespace that you are currently working in. Use this secret to apply to all the pods in such a way that these credentials will be used by kubernetes while pulling images from dockerhub. One way to do that is by means of a patch operation updating the service account for the corresponding namespace.

   ▼ Click to see solution

   ```
   docker logout
   docker login
   ```

   ```
   cp ~/.docker/config.json config.json
   kubectl create secret generic docker-registry \
       --from-file=.dockerconfigjson=config.json \
       --type=kubernetes.io/dockerconfigjson -n default
   ```

   ```
   kubectl get serviceaccount default -o yaml -n default
   kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name": "docker-registry"}]}' -n default
   kubectl get serviceaccount default -o yaml -n default
   rm config.json
   ```