Managing Compute Resource Usage

Relevant Documentation

- Managing Resources for Containers
- Assign CPU Resources to Containers and Pods
- Resource Quotas

Exam Tips

- A resource request informs the cluster of the expected resource usage for a container. It is used to select a node that has enough resources available to run the Pod.
- A resource limit sets an upper limit on how many resources a container can use. If the container process attempts to go above this limit, the container process will be terminated.
- A ResourceQuota limits the amount of resources that can be used within a specific Namespace. If a user attempts to create or modify objects in that Namespace such that the quota would be exceeded, the request will be denied.

Lesson Reference

Log in to the **control plane node**.

Create a new Namespace.

```
kubectl create namespace resources-test
```

Create a Pod with resource requests and limits.

```
vi resources-pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: resources-pod
  namespace: resources-test
spec:
  containers:
  - name: busybox
    image: busybox:stable
    command: ['sh', '-c', 'while true; do echo Running...; sleep 5; done']
    resources:
      requests:
        memory: 64Mi
        cpu: 250m
      limits:
        memory: 128Mi
        cpu: 500m
```

```
kubectl apply -f resources-pod.yml
```

Check the status of the new Pod.

```
kubectl get pods -n resources-test
```

Enable the ResourceQuota admission controller.

```
sudo vi /etc/kubernetes/manifests/kube-apiserver.yaml
```

Locate the `--enable-admission-plugins` flag and add `ResourceQuota` to the list.

Note: If the `NamespaceAutoProvision` controller is still enabled from a previous lesson, you can remove it from the list to disable it, or leave it enabled if you wish.

```
    - --enable-admission-plugins=NodeRestriction,ResourceQuota
```

Create a ResourceQuota.

```
vi resources-test-quota.yml
```

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: resources-test-quota
  namespace: resources-test
spec:
  hard:
    requests.memory: 128Mi
    requests.cpu: 500m
    limits.memory: 256Mi
    limits.cpu: "1"
```

```
kubectl apply -f resources-test-quota.yml
```

Try to create a Pod that would exceed the memory limit quota for the Namespace.

```
vi too-many-resources-pod.yml
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: too-many-resources-pod
  namespace: resources-test
spec:
  containers:
  - name: busybox
    image: busybox:stable
    command: ['sh', '-c', 'while true; do echo Running...; sleep 5; done']
    resources:
      requests:
        memory: 64Mi
        cpu: 250m
      limits:
        memory: 200Mi
        cpu: 500m
```

```
kubectl apply -f too-many-resources-pod.yml
```

This should fail, since this Pod, alongside the existing Pod, would exceed the memory limit quota for the Namespace.