Exercise 4.3: Resource Limits for a Namespace

The previous steps set limits for that particular `deployment`. You can also set limits on an entire `namespace`. We will create a new `namespace` and configure another `hog` deployment to run within. When set `hog` should not be able to use the previous amount of resources.

1. Begin by creating a new namespace called `low-usage-limit` and verify it exists.

```
student@cp:~$ kubectl create namespace low-usage-limit
```

```
namespace/low-usage-limit created
```

```
student@cp:~$ kubectl get namespace
```

```
NAME              STATUS    AGE
default           Active    1h
kube-node-lease   Active    1h
kube-public       Active    1h
kube-system       Active    1h
low-usage-limit   Active    42s
```

2. Create a YAML file which limits CPU and memory usage. The kind to use is `LimitRange`. Remember the file may be found in the example tarball.

```
student@cp:~$ vim low-resource-range.yaml
```

low-resource-range.yaml

```
1  apiVersion: v1
2  kind: LimitRange
3  metadata:
4    name: low-resource-range
5  spec:
6    limits:
7    - default:
8        cpu: 1
9        memory: 500Mi
10     defaultRequest:
11       cpu: 0.5
12       memory: 100Mi
13     type: Container
```

3. Create the LimitRange object and assign it to the newly created namespace low-usage-limit. You can use `--namespace` or `-n` to declare the namespace.

```
student@cp:~$ kubectl --namespace=low-usage-limit  \
    create -f low-resource-range.yaml
```

```
limitrange/low-resource-range created
```

4. Verify it works. Remember that every command needs a namespace and context to work. Defaults are used if not provided.

```
student@cp:~$ kubectl get LimitRange
```

```
No resources found in default namespace.
```

```
student@cp:~$ kubectl get LimitRange --all-namespaces
```

```
NAMESPACE         NAME                CREATED AT
low-usage-limit   low-resource-range   2023-02-23T10:23:57Z
```

5. Create a new deployment in the namespace.

```
student@cp:~$ kubectl -n low-usage-limit \
   create deployment limited-hog --image vish/stress
```

```
deployment.apps/limited-hog created
```

6. List the current deployments. Note `hog` continues to run in the default namespace. If you chose to use the **Calico** network policy you may see a couple more than what is listed below.

```
student@cp:~$ kubectl get deployments --all-namespaces
```

```
NAMESPACE        NAME                      READY  UP-TO-DATE   AVAILABLE   AGE
default          hog                       1/1    1            1           7m57s
kube-system      calico-kube-controllers   1/1    1            1           2d10h
kube-system      coredns                   2/2    2            2           2d10h
low-usage-limit  limited-hog               0/1    0            0           9s
```

7. View all pods within the namespace. Remember you can use the **tab** key to complete the namespace. You may want to type the namespace first so that tab-completion is appropriate to that namespace instead of the `default` namespace.

```
student@cp:~$ kubectl -n low-usage-limit get pods
```

```
NAME                            READY     STATUS    RESTARTS   AGE
limited-hog-2556092078-wnpnv    1/1       Running   0          2m11s
```

8. Look at the details of the pod. You will note it has the settings inherited from the entire namespace. The use of shell completion should work if you declare the namespace first.

```
student@cp:~$ kubectl -n low-usage-limit \
      get pod limited-hog-2556092078-wnpnv -o yaml
```

```
<output_omitted>
spec:
  containers:
  - image: vish/stress
    imagePullPolicy: Always
    name: stress
    resources:
      limits:
        cpu: "1"
        memory: 500Mi
      requests:
        cpu: 500m
        memory: 100Mi
    terminationMessagePath: /dev/termination-log
<output_omitted>
```

9. Copy and edit the config file for the original `hog` file. Add the namespace: line so that a new deployment would be in the `low-usage-limit` namespace. Delete the `selflink` line, if it exists.

```
student@cp:~$ cp hog.yaml hog2.yaml

student@cp:~$ vim hog2.yaml
```

hog2.yaml

```
1 ....
2   labels:
3     app: hog
4   name: hog
5   namespace: low-usage-limit     #<<--- Add this line, delete following
6   selfLink: /apis/apps/v1/namespaces/default/deployments/hog
7 spec:
8 ....
```

10. Open up extra terminal sessions so you can have **top** running in each. When the new deployment is created it will probably be scheduled on the node not yet under any stress.

    Create the deployment.

```
student@cp:~$ kubectl create -f hog2.yaml
```

```
deployment.apps/hog created
```

11. View the deployments. Note there are two with the same name, `hog` but in different namespaces. You may also find the `calico-typha` deployment has no pods, nor has any requested. Our small cluster does not need to add **Calico** pods via this autoscaler.

```
student@cp:~$ kubectl get deployments --all-namespaces
```

```
NAMESPACE         NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
default           hog                       1/1     1            1           24m
kube-system       calico-kube-controllers   1/1     0            0           4h
kube-system       coredns                   2/2     2            2           4h
low-usage-limit   hog                       1/1     1            1           26s
low-usage-limit   limited-hog               1/1     1            1           5m11s
```

12. Look at the **top** output running in other terminals. You should find that both `hog` deployments are using about the same amount of resources, once the memory is fully allocated. Per-deployment settings override the global namespace settings. You should see something like the following lines one from each node, which indicates use of one processor and about 12 percent of your memory, were you on a system with 8G total.

```
25128 root      20   0  958532 954672   3180 R 100.0 11.7   0:52.27 stress
24875 root      20   0  958532 954800   3180 R 100.3 11.7  41:04.97 stress
```

13. Delete the `hog` deployments to recover system resources.

```
student@cp:~$ kubectl -n low-usage-limit delete deployment hog
```

```
deployment.apps "hog" deleted
```

```
student@cp:~$ kubectl delete deployment hog
```

```
deployment.apps "hog" deleted
```