Exercise 4.2: Working with CPU and Memory Constraints

Overview

We will continue working with our cluster, which we built in the previous lab. We will work with `resource limits`, more with `namespaces` and then a complex `deployment` which you can explore to further understand the architecture and relationships.

Use **SSH** or **PuTTY** to connect to the nodes you installed in the previous exercise. We will deploy an application called **stress** inside a container, and then use `resource limits` to constrain the resources the application has access to use.

1. Use a container called `stress`, in a deployment which we will name `hog`, to generate load. Verify you have the container running.

```
student@cp:~$ kubectl create deployment hog --image vish/stress
deployment.apps/hog created
```

```
student@cp:~$ kubectl get deployments
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
hog    1/1     1            1           13s
```

2. Use the `describe` argument to view details, then view the output in `YAML` format. Note there are no settings limiting resource usage. Instead, there are empty curly brackets.

```
student@cp:~$ kubectl describe deployment hog
Name:                   hog
Namespace:              default
CreationTimestamp:      Tue, 08 Jan 2019 17:01:54 +0000
Labels:                 app=hog
Annotations:            deployment.kubernetes.io/revision: 1
<output_omitted>
```

```
student@cp:~$ kubectl get deployment hog -o yaml
apiVersion: apps/v1
kind: Deployment
Metadata:

<output_omitted>

  template:
    metadata:
      creationTimestamp: null
      labels:
        app: hog
    spec:
      containers:
      - image: vish/stress
        imagePullPolicy: Always
        name: stress
        resources: {}
        terminationMessagePath: /dev/termination-log
<output_omitted>
```

3. We will use the YAML output to create our own configuration file.

```
student@cp:~$ kubectl get deployment hog -o yaml > hog.yaml
```

4. Probably good to remove the status output, creationTimestamp and other settings. We will also add in memory limits found below.

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

hog.yaml

```
....
        imagePullPolicy: Always
         name: hog
         resources:                    # Edit to remove {}
           limits:                     # Add these 4 lines
             memory: "4Gi"
           requests:
             memory: "2500Mi"
         terminationMessagePath: /dev/termination-log
         terminationMessagePolicy: File
....
```

5. Replace the deployment using the newly edited file.

```
student@cp:~$ kubectl replace -f hog.yaml
```

```
deployment.apps/hog replaced
```

6. Verify the change has been made. The deployment should now show resource limits.

```
student@cp:~$ kubectl get deployment hog -o yaml
```

```
....
        resources:
          limits:
            memory: 4Gi
          requests:
            memory: 2500Mi
        terminationMessagePath: /dev/termination-log
....
```

7. View the stdio of the hog container. Note how much memory has been allocated.

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

```
NAME                   READY     STATUS    RESTARTS   AGE
hog-64cbfcc7cf-lwq66   1/1       Running   0          2m
```

```
student@cp:~$ kubectl logs hog-64cbfcc7cf-lwq66
```

```
I1102 16:16:42.638972       1 main.go:26] Allocating "0" memory, in
   "4Ki" chunks, with a 1ms sleep between allocations
I1102 16:16:42.639064       1 main.go:29] Allocated "0" memory
```

8. Open a second and third terminal to access both cp and second nodes. Run **top** to view resource usage. You should not see unusual resource usage at this point. The **dockerd** and **top** processes should be using about the same amount of resources. The **stress** command should not be using enough resources to show up.

9. Edit the `hog` configuration file and add arguments for **stress** to consume CPU and memory. The `args:` entry should be indented the same number of spaces as `resources:`.

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

hog.yaml

```
....
        resources:
          limits:
            cpu: "1"
            memory: "4Gi"
          requests:
            cpu: "0.5"
            memory: "500Mi"
        args:
        - -cpus
        - "2"
        - -mem-total
        - "950Mi"
        - -mem-alloc-size
        - "100Mi"
        - -mem-alloc-sleep
        - "1s"
....
```

10. Delete and recreate the deployment. You should see increased CPU usage almost immediately and memory allocation happen in 100M chunks, allocated to the **stress** program via the running **top** command. Check both nodes as the container could deployed to either. Be aware that nodes with a small amount of memory or CPU may encounter issues. Symptoms include cp node infrastructure pods failing. Adjust the amount of resources used to allow standard pods to run without error.

```
student@cp:~$ kubectl delete deployment hog
deployment.apps "hog" deleted
```

```
student@cp:~$ kubectl create -f hog.yaml
deployment.apps/hog created
```

Only if top does not show high usage

Should the resources not show increased use, there may have been an issue inside of the container. Kubernetes may show it as running, but the actual workload has failed. Or the container may have failed; for example if you were missing a parameter the container may `panic`.

```
student@cp:~$ kubectl get pod
NAME                   READY     STATUS    RESTARTS   AGE
hog-1985182137-5bz2w   0/1       Error     1          5s
```

```
student@cp:~$ kubectl logs hog-1985182137-5bz2w
panic: cannot parse '150mi': unable to parse quantity's suffix

goroutine 1 [running]:
panic(0x5ff9a0, 0xc820014cb0)
        /usr/local/go/src/runtime/panic.go:481 +0x3e6
```

```
k8s.io/kubernetes/pkg/api/resource.MustParse(0x7ffe460c0e69, 0x5, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0)
        /usr/local/google/home/vishnuk/go/src/k8s.io/kubernetes/pkg/api/resource/quantity.go:134 +0x287
main.main()
        /usr/local/google/home/vishnuk/go/src/github.com/vishh/stress/main.go:24 +0x43
```

Here is an example of an improper parameter. The container is running, but not allocating memory. It should show the usage requested from the YAML file.

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

```
NAME                   READY     STATUS    RESTARTS   AGE
hog-1603763060-x3vnn   1/1       Running   0          8s
```

```
student@cp:~$ kubectl logs hog-1603763060-x3vnn
```

```
I0927 21:09:23.514921       1 main.go:26] Allocating "0" memory, in "4ki" chunks, with a 1ms sleep \
                              between allocations
I0927 21:09:23.514984       1 main.go:39] Spawning a thread to consume CPU
I0927 21:09:23.514991       1 main.go:39] Spawning a thread to consume CPU
I0927 21:09:23.514997       1 main.go:29] Allocated "0" memory
```