Exercise 4.5: Setting Pod Resource Limits and Requirements

1. Create a new pod running the `vish/stress` image. A YAML `stress.yaml` file has been included in the course tarball.

2. Run the **top** command on the cp and worker nodes. You should find a `stress` command consuming the majority of the CPU on one node, the `worker`. Use **q** to exit from top. Delete the deployment.

3. Edit the `stress.yaml` file add in the following limits and requests.

```
student@cp:~$
```

```
....
        name: stressmeout
        resources:                  #<<-- Add this and following six lines
          limits:
            cpu: "1"
            memory: "1Gi"
          requests:
            cpu: "0.5"
            memory: "500Mi"
        args:
        - -cpus
....
```

4. Create the deployment again. Check the status of the pod. You should see that it shows an `OOMKilled` status and a growing number of restarts. You may see a status of `Running` if you catch the pod in early in a restart. If you wait long enough you may see `CrashLoopBackOff`.

```
student@cp:~$ kubectl get pod stressmeout-7fbbbcc887-v9kvb
```

```
NAME                           READY   STATUS      RESTARTS   AGE
stressmeout-7fbbbcc887-v9kvb   0/1     OOMKilled   2          32s
```

5. Delete then edit the deployment. Change the `limit:` parameters such that pod is able to run, but not too much extra resources. Try setting the memory limit to exactly what the stress command requests.

   As we allow the pod to run on the cp node, this could cause issues, such as the `kube-apiserver` restarting due to lack of resources. We will also add a `nodeSelector` to use the built in label of `kubernetes.io/hostname`.

```
student@cp:~$ kubectl delete -f stress.yaml
```

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

```
....
    spec:
      nodeSelector:                     #<-- Uncomment and edit
        kubernetes.io/hostname: worker  #<-- to by YOUR worker hostname
      containers:

....
        resources:
          limits:
            cpu: "2"
```

```
            memory: "2Gi"
          requests:
....
```

6. Create the deployment and ensure the pod runs without error. Use **top** to verify the stress command is running on one of the nodes and view the pod details to ensure the CPU and memory limits are in use. Also use the **kubectl describe node** command to view the resources your cp and worker node are using. The command details have been omitted. Use previous steps to figure out the commands.

7. Change `limits` and `requests` to numbers higher than your node resources, and evaluate how the container and pod is handled. It may take a while for resources to be fully allocated.

8. Remove the `stressmeout` deployment when done.

```
student@cp:~$ kubectl delete deploy stressmeout
```

```
deployment.apps "stressmeout" deleted
```