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 master and worker nodes. You should find a `stress` command consuming the majority of the CPU on on node. Use **ctrl-c** to exit from top. Delete the deployment.
3. Edit the `stress.yaml` file add in the following limits and requests.

```
student@ckad-1:~$
```

```
....
        name: stressmeout
        resources:
          limits:                  #<<-- Add this and following five lines
            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@ckad-1:~$ 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. You will find also be killed.

```
student@ckad-1:~$ kubectl delete -f stress.yaml

student@ckad-1:~$ vim stress.yaml
```

```
....
        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. The command details have been omitted. Use previous steps to figure out the commands.