Objective: To deploy a Python application (flask app) on Kubernetes using Ubuntu subsystem for Linux

Steps:

1. Create a new directory using and give it a reasonable name ($ mkdir flask_app_deployment).

```
:~$ mkdir flask_app_deployment
```

2. Ensure that your flask app has a requirements.txt file containing the flask version.
3. Then to copy your flask app from your Windows machine to your ubuntu you need to mount it. To do so use the command ($ cd /mnt/c). In this case I'm using my C drive.
4. From there head to the location of your flask app and use the command ($ cp -r <directory name of flask app> ~/flask_app_deployment).

```
:~$ cd /mnt/c
:/mnt/c$ cd ./Users/Cleon/Desktop
:/mnt/c/Users/Cleon/Desktop$ cp -r Chore-Todo ~/flask_app_deployment
```

5. Change from current directory into your flask app deployment using ($ cd ~/flask_app_deployment).
6. Then create your Dockerfile using the command ($ nano Dockerfile). Then use the following with information.

```
FROM python:3.8

RUN apt-get update -y && apt-get install -y python3-pip python3-dev

COPY ./Chore-Todo/requirements.txt /Chore-Todo/requirements.txt

WORKDIR /Chore-Todo

RUN pip install -r requirements.txt

COPY Chore-Todo .

ENV FLASK_APP=application.py

EXPOSE 5000

CMD flask run --host=0.0.0.0
```

```
  GNU nano 4.8                                        Dockerfile
FROM python:3.8
RUN apt-get update -y && apt-get install -y python3-pip python3-dev
COPY ./Chore-Todo/requirements.txt /Chore-Todo/requirements.txt
WORKDIR /Chore-Todo
RUN pip install -r requirements.txt
COPY Chore-Todo .
ENV FLASK_APP=application.py
EXPOSE 5000
CMD flask run --host=0.0.0.0
```

7. Build your docker image and give it a tag ensuring that all dependencies were created. Use the command ($ docker build -t flask-chore-todo:lastest .)

```
eon:~/flask_app_deployment$ docker build -t flask-chore-todo:latest .
; 242.3s (12/12) FINISHED
1] load build definition from Dockerfile                                                    0.0s
ferring dockerfile: 38B                                                                     0.0s
1] load .dockerignore                                                                       0.0s
ferring context: 2B                                                                         0.0s
1] load metadata for docker.io/library/python:3.8                                           5.9s
ibrary/python:pull token for registry-1.docker.io                                           0.0s
1] load build context                                                                       0.0s
ferring context: 634B                                                                       0.0s
OM docker.io/library/python:3.8@sha256:b39ab988ac2fa749273cf6fdeb89fb3635850824419dc612c8a21ff4c796  159.2s
```

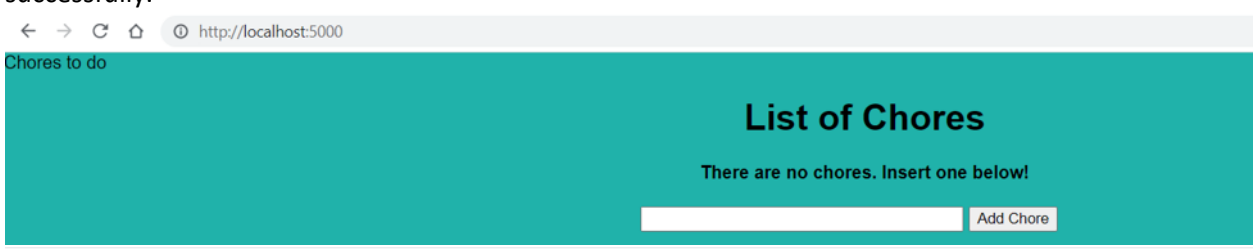
8. Use the command ($ docker images) to see the successful docker image that was built.

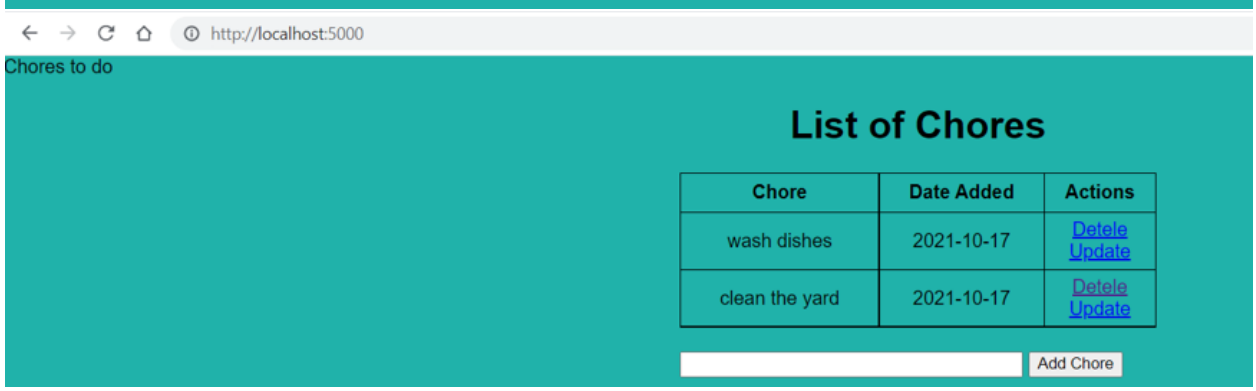
```
REPOSITORY          TAG       IMAGE ID       CREATED              SIZE
flask-chore-todo    latest    1a575066fb32   About a minute ago   990MB
```

9. Now run your docker image ensuring that it has a port using the command ($ docker run -d -p 5000:5000 flask-chore-todo).

```
:~/flask_app_deployment$ docker run -d -p 5000:5000 flask-chore-todo
```

10. Then go to your local host using the url (http://localhost:5000) to ensure that it runs successfully.





NOTE: if experiencing errors, use the command the following commands

i. In case of 500 errors use ($ docker logs CONTAINER_ID) to view the logs.
ii. In case to need to browse a container use the command ($ docker container exec -it CONTAINER_ID ls).

11. Now rebuild and tag your image when pushing it to Dockerhub. Use the command ($ docker build -t mastercle/flask-chore-todo:latest .). Format being “<dockerhub username>/<flask name>:<tag>”.

```
:~/flask_app_deployment$ docker build -t mastercle/flask-chore-todo:latest .
5s (12/12) FINISHED
load build definition from Dockerfile                                    0.6s
ring dockerfile: 38B                                                     0.0s
load .dockerignore                                                       0.0s
ring context: 2B                                                         0.0s
load metadata for docker.io/library/python:3.8                          8.6s
```

12. Once successfully built, push the image to your docker account using the command ($ docker push mastercle/flask-chore-todo:latest).

```
:~/flask_app_deployment$ docker push mastercle/flask-chore-todo:latest
 to repository [docker.io/mastercle/flask-chore-todo]
```

13. Then create your Kubernetes cluster with a load balancer that accesses port 8081 and maps to the container on port 8080. Use the command ($ k3d cluster create chore-todo-cluster -p “8081:8080@loadbalancer).

```
loyment$ k3d cluster create chore-todo-cluster -p "8081:8080@loadbalancer"
```

NOTE I had a container running on port 8080 so I used the command ($ docker ps -a) to see what was running on the port and the command ($ docker stop CONTAINER_ID) to stop the container.

```
INFO[0163] Deleting image volume 'k3d-chore-todo-cluster-images'
FATA[0163] Cluster creation FAILED, all changes have been rolled back!
```

```
:~/flask_app_deployment$ docker ps -a
IMAGE                       COMMAND                  CREATED          STATUS                     PORTS
                                    NAMES
flask-chore-todo            "/bin/sh -c 'flask r…"   42 minutes ago   Up 42 minutes              0.0.0.0:
                                    reverent_robinson
rancher/k3d-proxy:5.0.1     "/bin/sh -c nginx-pr…"   14 hours ago     Up 14 hours                80/tcp,
443/tcp, 0.0.0.0:8081->8080/tcp     k3d-cluster01-serverlb
rancher/k3s:v1.21.5-k3s2    "/bin/k3s server --t…"   14 hours ago     Up 14 hours
                                    k3d-cluster01-server-0
rancher/k3d-proxy:5.0.1     "/bin/sh -c nginx-pr…"   14 hours ago     Up 14 hours                80/tcp,
443/tcp                             k3d-cluster1-serverlb
rancher/k3s:v1.21.5-k3s2    "/bin/k3s server --t…"   14 hours ago     Up 14 hours
                                    k3d-cluster1-server-0
git                         "bash"                   12 days ago      Exited (0) 11 days ago
                                    amazing_mcnulty
docker/getting-started      "/docker-entrypoint.…"   2 weeks ago      Exited (255) 12 days ago   0.0.0.0:
                                    reverent_haslett
:~/flask_app_deployment$ docker stop 6dc0

:~/flask_app_deployment$ k3d cluster create chore-todo-cluster -p "8081:8080@loadbalancer"
```

14. In your flask_app_deployment directory, create a YAML file using the command ($ nano chore-todo.yml). There you will input the following ensuring that the name matches your app:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chore-todo-deployment
spec:
  selector:
    matchLabels:
      app: chore_todo
  replicas: 1 # tells deployment to run 1 pods matching the template
  template: # create pods using pod definition in this template
    metadata:
      labels:
        app: chore_todo
    spec:
      containers:
      - name: chore-todo-container
        image: mastercle/flask-chore-todo:latest
        ports:
        - containerPort: 5000

---

apiVersion: v1
kind: Service
metadata:
  name: chore-todo-service
spec:
```

type: LoadBalancer

ports:

- port: 8080

  protocol: TCP

  targetPort: 5000

selector:

  app: chore_todo

```
  GNU nano 4.8                                        chore-todo.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chore-todo-deployment
spec:
  selector:
    matchLabels:
      app: chore_todo
  replicas: 1 # tells deployment to run 1 pods matching the template
  template: # create pods using pod definition in this template
    metadata:
      labels:
        app: chore_todo
    spec:
      containers:
      - name: chore-todo-container
        image: mastercle/flask-chore-todo:latest
        ports:
        - containerPort: 5000
```

```
    spec:
      containers:
      - name: chore-todo-container
        image: mastercle/flask-chore-todo:latest
        ports:
        - containerPort: 5000

---

apiVersion: v1
kind: Service
metadata:
  name: chore-todo-service
spec:
  type: LoadBalancer
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 5000
  selector:
    app: chore_todo
```

NOTE: The Kubernetes pod/Docker container uses port 5000. However, Kubernetes uses port 8080 and the load balancer uses port 8081.

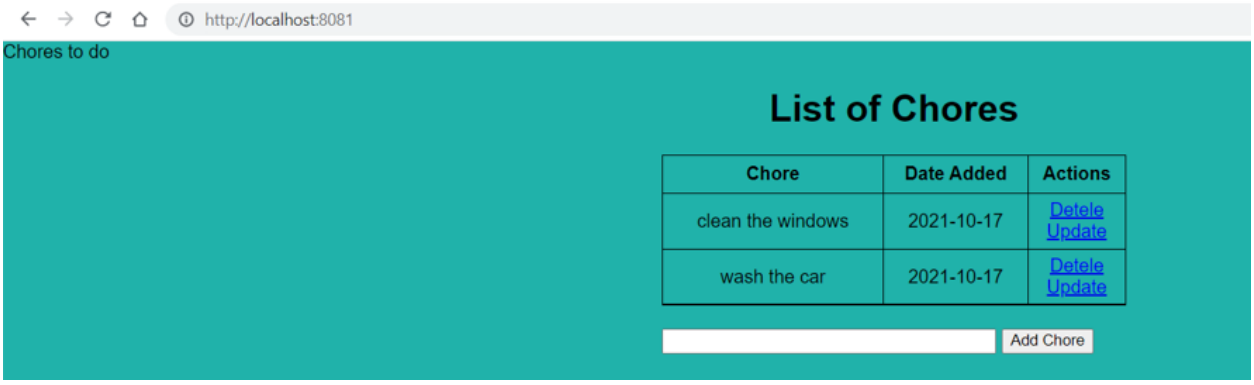
15. Use the command ($ kubectl create -f chore-todo.yml) to run your YAML file.

```
~/flask_app_deployment$ nano chore-todo.yml
~/flask_app_deployment$ kubectl create -f chore-todo.yml
chore-todo-deployment created
do-service created
```

16. Once created use the command ($ kubectl get all) to see the status of the containers.
17. Then use the command ($ kubectl get pod) to see the status of the pods. This is important because both containers' statuses need to say "running" in order for the local host to be active.

```
:~/flask_app_deployment$ kubectl get pod
                         READY   STATUS              RESTARTS   AGE
oyment-688c68cc8-vrnjj   0/1     ContainerCreating   0          2m8s
o-service-gnfk7          1/1     Running             0          2m8s
:~/flask_app_deployment$ kubectl get pod
                         READY   STATUS    RESTARTS   AGE
o-service-gnfk7          1/1     Running   0          3m3s
oyment-688c68cc8-vrnjj   1/1     Running   0          3m3s
```

18. Head to your browser and use the url(http://localhost:8081).



19. Once done delete the deployment and service.

```
:~/flask_app_deployment$ kubectl delete deployment chore-todo-deployment
 "chore-todo-deployment" deleted
:~/flask_app_deployment$ kubectl delete service chore-todo-service
todo-service" deleted
```