Deploying python application in kubernetes

Create a python app

```
┌──(root㉿kali)-[~/Awake/Dockerapp]
└─# cat extract.py
import random
import requests
from bs4 import BeautifulSoup
from flask import Flask, jsonify
# crawl IMDB Top 250 and randomly select a movie

URL = 'http://www.imdb.com/chart/top'
app = Flask(__name__)
def main():
    response = requests.get(URL)

    soup = BeautifulSoup(response.text, 'html.parser')
    #soup = BeautifulSoup(response.text, 'lxml') # faster

    # print(soup.prettify())

    movietags = soup.select('td.titleColumn')
    inner_movietags = soup.select('td.titleColumn a')
    ratingtags = soup.select('td.posterColumn span[name=ir]')

    def get_year(movie_tag):
        moviesplit = movie_tag.text.split()
        year = moviesplit[-1] # last item
        return year

    years = [get_year(tag) for tag in movietags]
    actors_list =[tag['title'] for tag in inner_movietags] # access attribute 'title'
    titles = [tag.text for tag in inner_movietags]
    ratings = [float(tag['data-value']) for tag in ratingtags] # access attribute 'data-value'

    n_movies = len(titles)

    while(True):
        idx = random.randrange(0, n_movies)

        print(f'{titles[idx]} {years[idx]}, Rating: {ratings[idx]:.1f}, Starring: {actors_list[idx]}')

        user_input = input('Do you want another movie (y/[n])? ')
        if user_input ≠ 'y':
            break


if __name__ == '__main__':
   # main()
        app.run(host='0.0.0.0', debug=True)
```

Create a Dockerfile to containerize the application

```
┌──(root㉿kali)-[~/Awake/Dockerapp]
└─# cat Dockerfile
From python:3.8

ADD extract.py .

RUN pip install requests beautifulsoup4
RUN pip install requests flask

CMD ["python", "./extract.py"]
```

Build and run the application

```
┌──(root㉿kali)-[~/Awake/Dockerapp]
└─# docker build -t python-imdb .
Sending build context to Docker daemon  5.632kB
Step 1/5 : From python:3.8
 ---> 63c8db7db039
Step 2/5 : ADD extract.py .
 ---> Using cache
 ---> 6c9c9261dcd7
Step 3/5 : RUN pip install requests beautifulsoup4
 ---> Using cache
 ---> 6f818b6a8c0d
Step 4/5 : RUN pip install requests flask
 ---> Using cache
 ---> 0df0c9c8c3e1
Step 5/5 : CMD ["python", "./extract.py"]
 ---> Using cache
 ---> a42000a9a01d
Successfully built a42000a9a01d
Successfully tagged python-imdb:latest
```

```
┌──(root㉿kali)-[~/Awake/Dockerapp]
└─# docker run python-imdb
 * Serving Flask app 'extract' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000 (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 139-052-271
```

```
┌──(root㉿kali)-[~]
└─# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS
          NAMES
ab7db8188ecb   python-imdb                          "python ./extract.py"    12 seconds ago   Up 6 seconds
          intelligent_wu
2c5f1903b0fe   gcr.io/k8s-minikube/kicbase:v0.0.30   "/usr/local/bin/entr…"   10 days ago      Up 38 minutes   127.0.0.1:49157→22/tcp, 127.0.0
443/tcp   minikube
```

Create a deployment.yaml file to deploy it with kubernetes

```
┌──(root㉿kali)-[~/Awake/Dockerapp]
└─# cat deployment.yaml
apiVersion: v1
kind: Service
metadata:
  name: python-service
spec:
  selector:
    app: example-app
  ports:
  - protocol: "TCP"
    port: 6000
    targetPort: 5000
  type: LoadBalancer

---

apiVersion: v1
kind: Deployment
metadata:
  name: python-app
spec:
  selector:
    matchLabels:
      app: example-app
  replicas: 2
  template:
    metadata:
      labels:
        app: python-app
    spec:
      containers:
      - name: example-app
        image: python-imdb
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 5000
```

Create a Secret based on existing credentials

kubectl create secret generic regcred
--from-file=.dockerconfigjson=/root/.docker/config.json
--type=kubernetes.io/dockerconfigjson

Create a Secret by providing credentials on the command line

kubectl create secret docker-registry regcred --docker-server=https://index.docker.io/v1/ --docker-username=20732078 --docker-password=<password> --docker-email=shresthasudeep2073@gmail.com

Apply secret in deployment.yaml

```
    image: 20732078/kubernetes_python
  imagePullSecrets:
  - name: regcred
```

Start minikube

```
┌──(root㉿kali)-[~/Awake/Dockerapp]
└─# minikube start --force --driver=docker


😄  minikube v1.25.2 on Debian kali-rolling (vbox/amd64)
❗  minikube skips various validations when --force is supplied; this may lead to unexpected behavior
✨  Using the docker driver based on existing profile
🛑  The "docker" driver should not be used with root privileges.
💡  If you are running minikube within a VM, consider using --driver=none:
📘    https://minikube.sigs.k8s.io/docs/reference/drivers/none/
💡  Tip: To remove this root owned cluster, run: sudo minikube delete

❗  The requested memory allocation of 1981MiB does not leave room for system overhead (total system memory: 1981MiB).
You may face stability issues.
💡  Suggestion: Start minikube with less memory allocated: 'minikube start --memory=1981mb'

👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
    ▪ kubelet.housekeeping-interval=5m
🔎  Verifying Kubernetes components...
    ▪ Using image kubernetesui/metrics-scraper:v1.0.7
❗  Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 6.211135723s
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Using image kubernetesui/dashboard:v2.3.1
💡  Restarting the docker service may improve performance.
🌟  Enabled addons: default-storageclass, storage-provisioner, dashboard
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Apply deployment file with kubectl

```
┌──(root㉿kali)-[~/Awake/Dockerapp]
└─# kubectl apply -f deployment.yaml
service/python-service unchanged
```

View the kubernetes clusters in minikube dashboard

```
┌──(root㉿kali)-[~/Awake/Dockerapp]
└─# minikube dashboard
🤔  Verifying dashboard health ...
🚀  Launching proxy ...
🤔  Verifying proxy health ...
http://127.0.0.1:38773/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/
```