|  |  |
| --- | --- |
| NAME | M.MUTHU SELVI |
| REG.NO: | 960519205010 |

ASSIGNMENT 4

QUESTION:

Assignment Kubernetes / Docker

1.Pull an Image from docker hub and run it in docker playground.

2.Create a docker file for the jobportal application and deploy it in Docker desktop application.

3.Create a IBM container registry and deploy helloworld app or jobportalapp.

4.Create a Kubernetes cluster in IBM cloud and deploy helloworld image or jobportal image and also expose the same app to run in nodeport

SOLUTION:

Kubernetes

dashboard\_adminuser.yaml:

apiVersion: v1

kind: ServiceAccount

metadata:

name: admin-user

namespace: kubernetes-dashboard

---

apiVersion: v1

kind: Secret

metadata:

name: admin-user-token

namespace: kubernetes-dashboard

annotations:

kubernetes.io/service-account.name: admin-user

type: kubernetes.io/service-account-token

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: admin-user

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: admin-user

namespace: kubernetes-dashboard

flask\_deployment.yaml:

apiVersion: apps/v1

kind: Deployment

metadata:

name: flask-app

spec:

replicas: 3

selector:

matchLabels:

app: flask-app

template:

metadata:

labels:

app: flask-app

spec:

containers:

- name: webpage

image: rajkiranss/flask

imagePullPolicy: Never

ports:

- containerPort: 5000

protocol: TCP

flask\_ingress.yaml:

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: flask-app-ingress

annotations:

kubernetes.io/ingress.class: nginx

nginx.ingress.kubernetes.io/ssl-redirect: "false"

spec:

# ingressClassName: nginx

rules:

- http:

paths:

- backend:

service:

name: flask-app-service

port:

number: 5000

path: /

pathType: Prefix

flask\_service.yaml:

apiVersion: v1

kind: Service

metadata:

name: flask-app-service

spec:

type: ClusterIP

ports:

- port: 5000

selector:app: flask-app

ibm\_deployment.yaml:

apiVersion: apps/v1

kind: Deployment

metadata:

name: flask-deploy

spec:

replicas: 3

selector:

matchLabels:

app: flask-deploy

template:

metadata:

labels:

app: flask-deploy

spec:

containers:

- name: web

image: jp.icr.io/webpage/web

imagePullPolicy: Always

ports:

- containerPort: 5000

protocol: TCP

templates:

base.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title> FlaskApp</title>

<style>

.message {

padding: 10px;

margin: 5px;

background-color: #f3f3f3

}

nav a {

color: #d64161;

font-size: 25px;

margin-left: 50px;

text-decoration: none;

text-transform: uppercase;

}

.alert {

padding: 20px;

margin: 5px;

color: #970020;

background-color: #ffd5de;

}

.button {

background-color:#970020; /\* Green \*/

border: none;

color: white;

padding: 16px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

transition-duration: 0.4s;

cursor: pointer;

}

.button1 {

background-color: white;

color: black;

border: 2px solid #970020;

}

.button1:hover {

background-color: #970020;

color: white;

}

input[type=text],textarea {

width: 50%;

padding: 12px 20px;

margin: 8px 0;

box-sizing: border-box;

border: 2px solid red;

border-radius: 4px;

}

</style>

</head>

<body>

<nav>

<a href="{{ url\_for('index') }}">FlaskApp</a>

<a href="{{ url\_for('create') }}">Create</a>

</nav>

<hr>

<div class="content">

{% for message in get\_flashed\_messages() %}

<div class="alert">{{ message }}</div>

{% endfor %}

{% block content %} {% endblock %}

</div>

</body>

</html>

create.html:

{% extends 'base.html' %}

{% block content %}

<h1>{% block title %} Add a New Message {% endblock %}</h1>

<form method="post" style="text-align: center;">

<label for="title">TITLE</label>

<br>

<input type="text" name="Welcome"

placeholder="Hello World!"

value="{{ request.form['Welcome'] }}"></input>

<br>

<label for="content">Hello World!</label>

<br>

<textarea name="content"

placeholder="Hello World!"

rows="15"

cols="60"

>{{ request.form['content'] }}</textarea>

<br>

<button type="submit" class="button button1">Submit</button>

</form>

<a href="{{ url\_for('index') }}" class="button button1">Home</a>

{% endblock %}

index.html:

{% extends 'base.html' %}

{% block content %}

<h1 style="text-align: center;">HELLO !</h1>

<h1> {% block title %} <span style="color: red;">WRITE YOUR MESSAGES</span> {% endblock %}</h1>

{% for message in messages %}

<div class='message'>

<h3><span style="color: blue;">welcome </span> {{ message['title'] }}</h3>

<p><span style="color: blue;">Hello World! </span>{{ message['content'] }}</p>

</div>

{% endfor %}

{% endblock %}

dockerfile:

FROM python:3.10.4

WORKDIR /app

COPY requirements.txt ./

RUN pip install -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python","./app.py"]

app.py

from flask import Flask, render\_template, request, url\_for, flash, redirect

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'df0331cefc6c2b9a5d0208a726a5d1c0fd37324feba25506'

@app.route('/create/', methods=('GET', 'POST'))

def create():

if request.method == 'POST':

title = request.form['title']

content = request.form['content']

if not title:

flash('Title is required!')

elif not content:

flash('Content is required!')

else:

messages.append({'title': title, 'content': content})

name = "Shyam Mohan"

return redirect(url\_for('index', messages=name ))

return render\_template('create.html')

messages = [{'title': 'Message One',

'content': 'Message One Content'},

{'title': 'Message Two',

'content': 'Message Two Content'}

]

@app.route('/')

def index():

return render\_template('index.html', messages=messages)

@app.route('/admin')

def hello\_admin():

return 'Hello Admin'

@app.route('/guest/<guest>')

def hello\_guest(guest):

return 'Hello %s as Guest' % guest

@app.route('/user/<name>')

def hello\_user(name):

if name =='admin':

return redirect(url\_for('hello\_admin'))

else:

return redirect(url\_for('hello\_guest',guest = name))

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5000, debug=True)