

Date	22 October 2022
Team ID	PNT2022TMID48162
Student Name	B.Sandhiya
Student Rollno	912619104022

Assignment - 4 Kubernetes and Docker

Question

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 jobportal app
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

Solutions

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

- a. Pull an image *uifd/ui-for-docker* from the docker hub
- b. This image is used for viewing and managing the docker engine
- c. Use `docker pull image_name` and `docker run -it image_name` commands to run the above image in the Docker Playground

The screenshot shows the 'lets-play-with-docker' website interface. At the top, there's a navigation bar with links like 'Home', 'About', 'Contact', 'FAQ', 'Privacy Policy', 'Terms of Service', and 'Help'. Below this, there's a large blue banner with the text 'lets-play-with-docker' and a 'GET STARTED' button. The main content area displays a list of Docker instances. The first instance is 'cdan30u3_cdan33m3tccg00fgibm0' with IP '192.168.0.8'. It is running the 'alpine:latest' image. Below the instance list, there's a terminal window showing the command 'docker pull alpine:latest' and the output 'latest: Pulling from alpine:latest'. The terminal also shows the command 'docker run -d --privileged --name lets-play-with-docker alpine:latest' and the output 'cdan30u3_cdan33m3tccg00fgibm0'.

The screenshot displays the 'UI For Docker' web interface. At the top, there is a navigation bar with tabs for 'Dashboard', 'Containers', 'Containers Network', 'Images', 'Networks', 'Volumes', and 'Info'. A 'Refresh' button is located on the right. The main content area is divided into two sections. The left section, titled 'Running Containers', shows a list of containers with one entry: 'trading_gemini' with a status of 'Up 10 seconds'. The right section, titled 'Status', features a donut chart showing the distribution of container states: 'Running' (green, approximately 90%), 'Stopped' (red, approximately 10%), and 'Ghost' (grey, 0%). Below these sections are two line graphs: 'Containers created' and 'Images created', both showing a single data point at the top of the y-axis.

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

- Create a docker file for build and deploy flask app.
- Use `docker build -t image_name .` in the current directory to start building the docker image and deploy in our local docker
- Use `docker run -p 5000:5000 image_name` to run in local system

Dockerfile

```
FROM ubuntu/apache2
FROM python
COPY ./requirements.txt /flaskApp/requirements.txt
WORKDIR /flaskApp
RUN pip install -r requirements.txt
COPY . /flaskApp
ENTRYPOINT [ "python" ]
CMD ["app.py" ]
Steps Involved
```

```
root@jagr-ma:/home/jagdish/Documents/DockerLearning# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
root@jagr-ma:/home/jagdish/Documents/DockerLearning# docker build -t job-portal-app .
Sending build context to Docker daemon 65.02kB
Step 1/8 : FROM ubuntu/apache2
latest: Pulling from ubuntu/apache2
b572d2b36365: Pull complete
3a1193867518: Pull complete
8bfd5261bf9e: Pull complete
Digest: sha256:d9b8fe0cbe6964368a1b4837e521ce326c287679bd1da6cd9989997dc2569382
Status: Downloaded newer image for ubuntu/apache2:latest
--> 8ca4f2c95e83
Step 2/8 : FROM python
latest: Pulling from library/python
f60d8928ed1: Pull complete
47db815c6a45: Pull complete
```

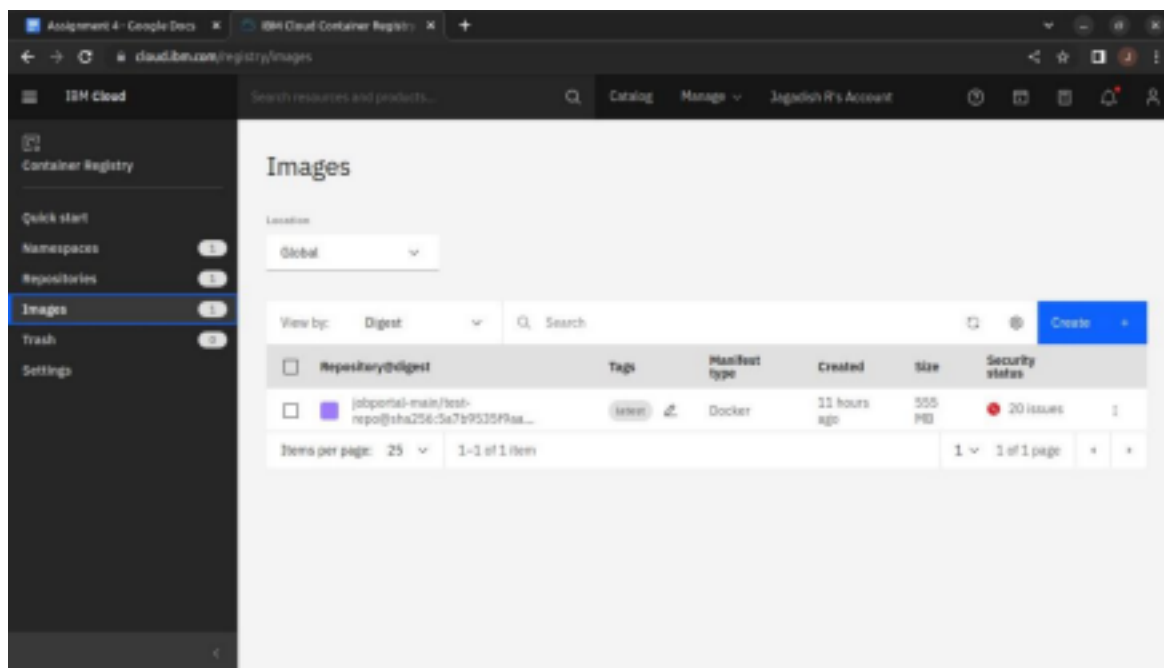
```
Step 7/8 : ENTRYPOINT [ "python" ]
--> Running in 64e98c363813
Removing intermediate container 64e98c363813
--> 7a8fadbf86d7
Step 8/8 : CMD ["app.py" ]
--> Running in 0380311c3e1d
Removing intermediate container 0380311c3e1d
--> 3b556e5956a8
Successfully built 3b556e5956a8
Successfully tagged job-portal-app:latest
root@jagr-ma:/home/jagdish/Documents/DockerLearning# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
job-portal-app latest 3b556e5956a8 About a minute ago 1.15GB
ubuntu/apache2 latest 8ca4f2c95e83 3 days ago 260MB
python latest f89c8762fe13 9 days ago 921MB
```

```
root@jagr-ma:/home/jagdish/Documents/DockerLearning# docker run -p 5000:5000 job-portal-app
* Serving Flask app "app"
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* 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: 481-117-531
```

Run locally using docker

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

- Log into IBM cloud
- Create a **container registry**
- Using IBM Cloud CLI, install the **container registry plugin** in our system
- Push our docker image into the created container registry using **docker push**
- So, our job portal app is deployed in the IBM container registry



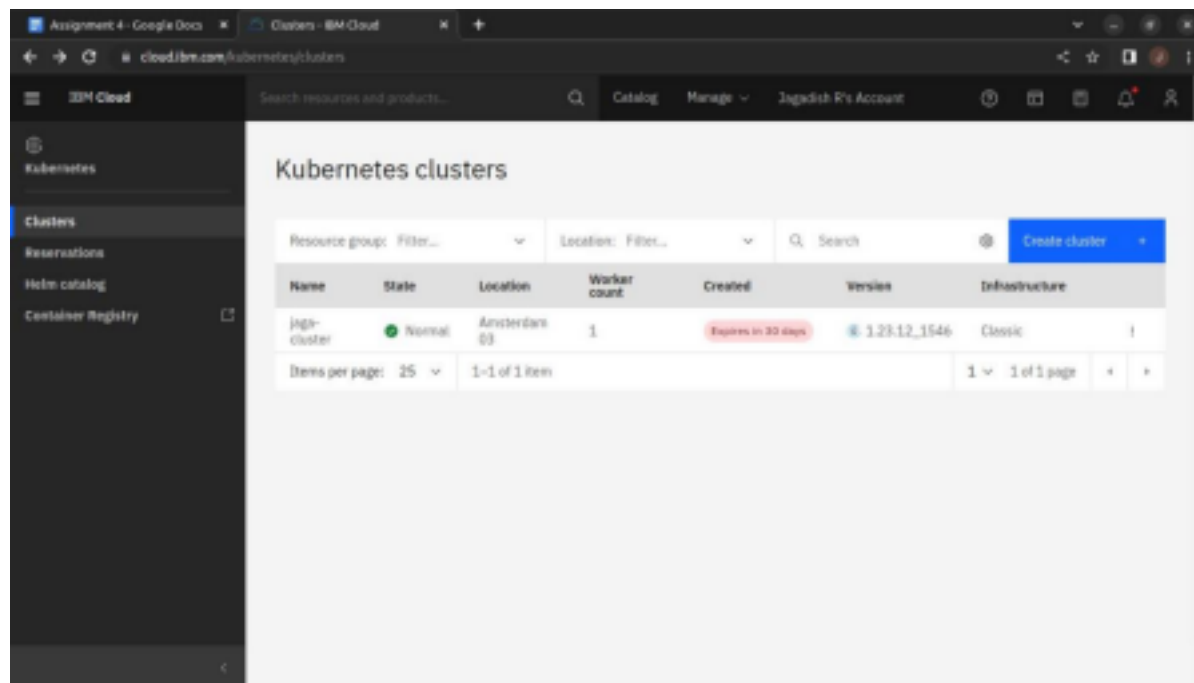
4. Create a Kubernetes cluster in IBM cloud and deploy helloworld image or jobportal image and also expose the same app to run in nodeport

- Log into IBM cloud
- Create a **kubernete**
- Using IBM Cloud CLI, install the **ks plugin** in our system
- Create a **cluster** in the kubernetes
- Now, go to the **kubernetes dashboard** where we need to create a service based on a yml file (given below)
- In that file, we have to mention *which image we are going to use* and the *app name*
- Take the **public IP address** and **Nodeport** since we exposed the *flask app in nodeport*
- Finally, we got the **url address** where our flask app is hosted

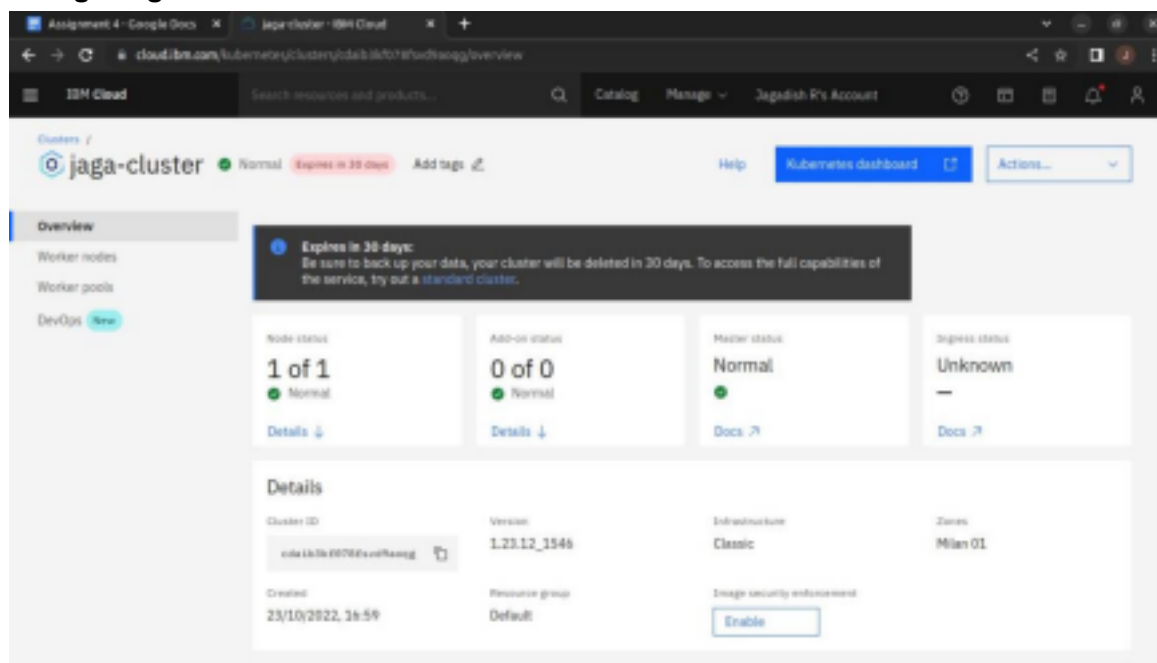
job-portal-app.yml

```
apiVersion: v1
kind: Service
metadata:
  name: job-portal-app
spec:
  selector:
    app: job-portal-app
  ports:
    - port: 5000
      type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: job-portal-app
  labels:
    app: job-portal-app
spec:
  selector:
    matchLabels:
      app: job-portal-app
  replicas: 1
  template:
    metadata:
      labels:
        app: job-portal-app
    spec:
      containers:
        - name: job-portal-app
          image: image_name
          ports:
            - containerPort: 5000
          env:
            - name: DISABLE_WEB_APP
              value: "false"
```

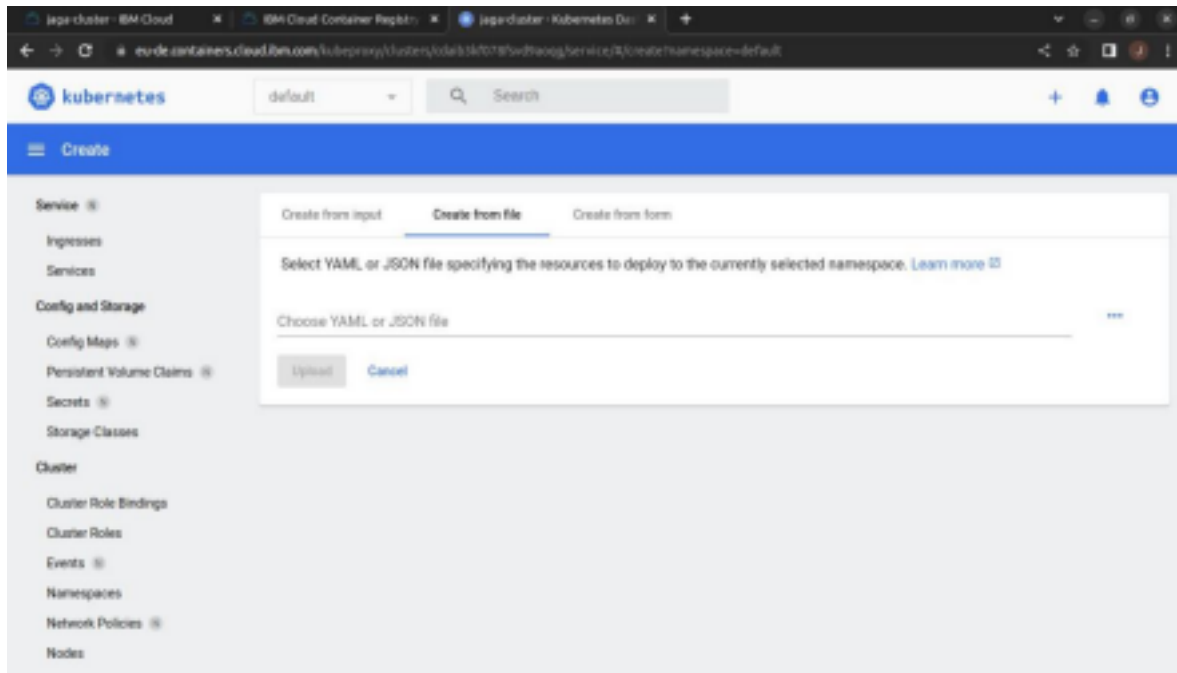
Cluster creation



Configuring the cluster



Creating a service based on the yml file



Procedure to find the exposed url

```

root@ravi-HP-58m-Desktop-290-p8xx:-
/ # ibmcloud ks cluster config --cluster cda1b3kf078fsvd9aoqg
OK
The configuration for cda1b3kf078fsvd9aoqg was downloaded successfully.
added context for cda1b3kf078fsvd9aoqg to the current kubecfg file.
You can now execute 'kubectl' commands against your cluster. For example, run 'kubectl get nodes'.
If you are accessing the cluster for the first time, 'kubectl' commands might fail for a few seconds while RBAC synchronizes.
/ # kubectl config current-context
jaga-cluster/cda1b3kf078fsvd9aoqg
/ # kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
10.144.183.00       Ready    <none>    6h7m   v1.23.12+IKS
/ # kubectl get pods
NAME                READY     STATUS    RESTARTS   AGE
job-portal-app-57f769b6b6-4rggl   1/1       Running    0           5h51m
/ # ibmcloud cs workers --cluster cda1b3kf078fsvd9aoqg
OK
IP
kube-cda1b3kf078fsvd9aoqg-jagacuster-default-00000073   Public IP   Private IP   Flavor   State   Status   Zone   Version
159.122.179.243   10.144.183.00   free       normal   Ready   nil/01   1.23.12_1540
/ # kubectl describe service job-portal-app
Name:                 job-portal-app
Namespace:            default
Labels:               <none>
Annotations:          <none>
Selector:             app=job-portal-app
Type:                 NodePort
IP Family Policy:     SingleStack
IP Families:          IPv4
IPs:                  172.21.183.254
172.21.183.254
Port:                 <unset> 5000/TCP
TargetPort:           5000/TCP
NodePort:             <unset> 30508/TCP
Endpoints:            172.30.146.139:5000
Session Affinity:     None
External Traffic Policy: Cluster
Events:              <none>
/ #

```

Run our flask app in the IBM kubernetes

