

Assignment-4

Plasma Donor Application

Team ID : PNT2022TMID23657

Team Leader : Pooja M

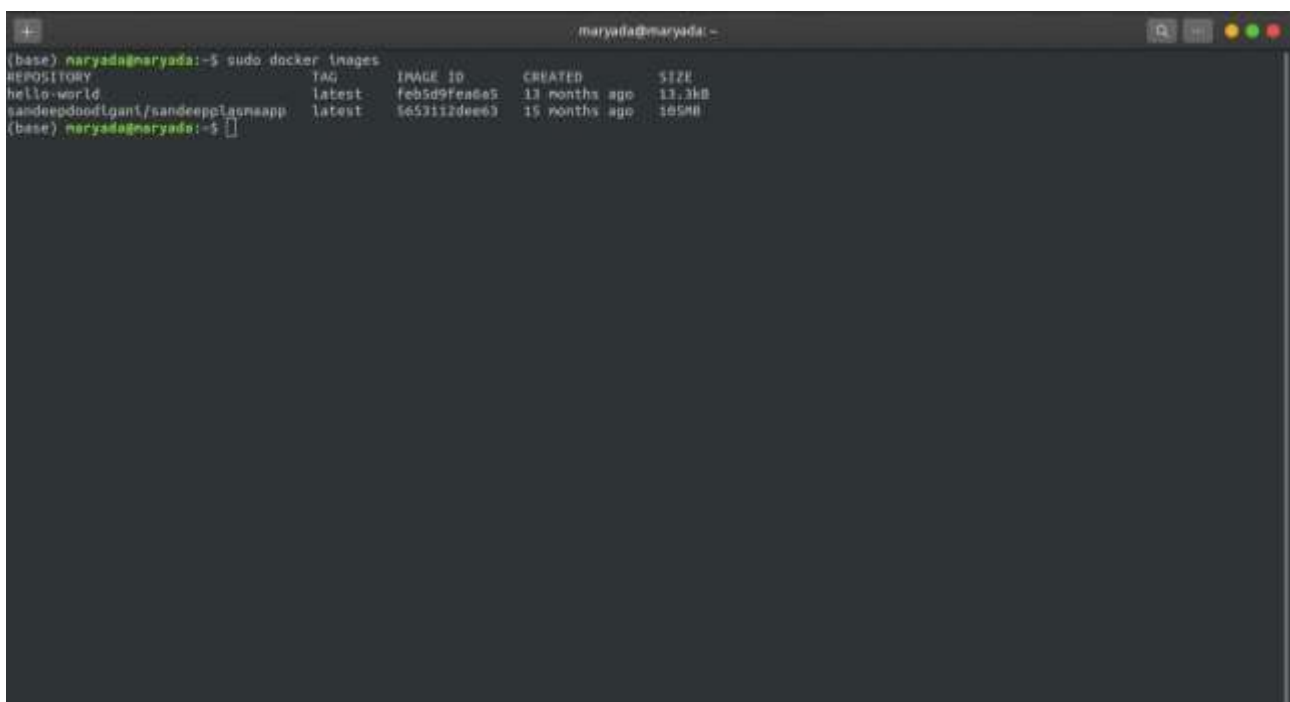
Team member : Nehaa V

Team member : Suvetha G

Team member : Srinithi S

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

Pulled sandeepdoodigani/plasmaapplication and running in docker:

A terminal window titled 'maryada@maryada: ~' showing the execution of 'sudo docker images'. The output lists two images: 'hello-world' and 'sandeepdoodigani/sandeepplasmaapp'.

```
(base) maryada@maryada:~$ sudo docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
hello-world         latest      feb5d9fea6a5 13 months ago 13.3kB
sandeepdoodigani/sandeepplasmaapp latest      5653112dee63 15 months ago 105MB
(base) maryada@maryada:~$
```

```
maryada@maryada:~$ sudo docker run -p 8080:8080 sandeepdoodigani/sandeeplasmaapp
* Serving Flask app 'app' (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: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.17.0.2:8080/ (Press CTRL+C to quit)
```



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

Dockerfile:

FROM python:3.6

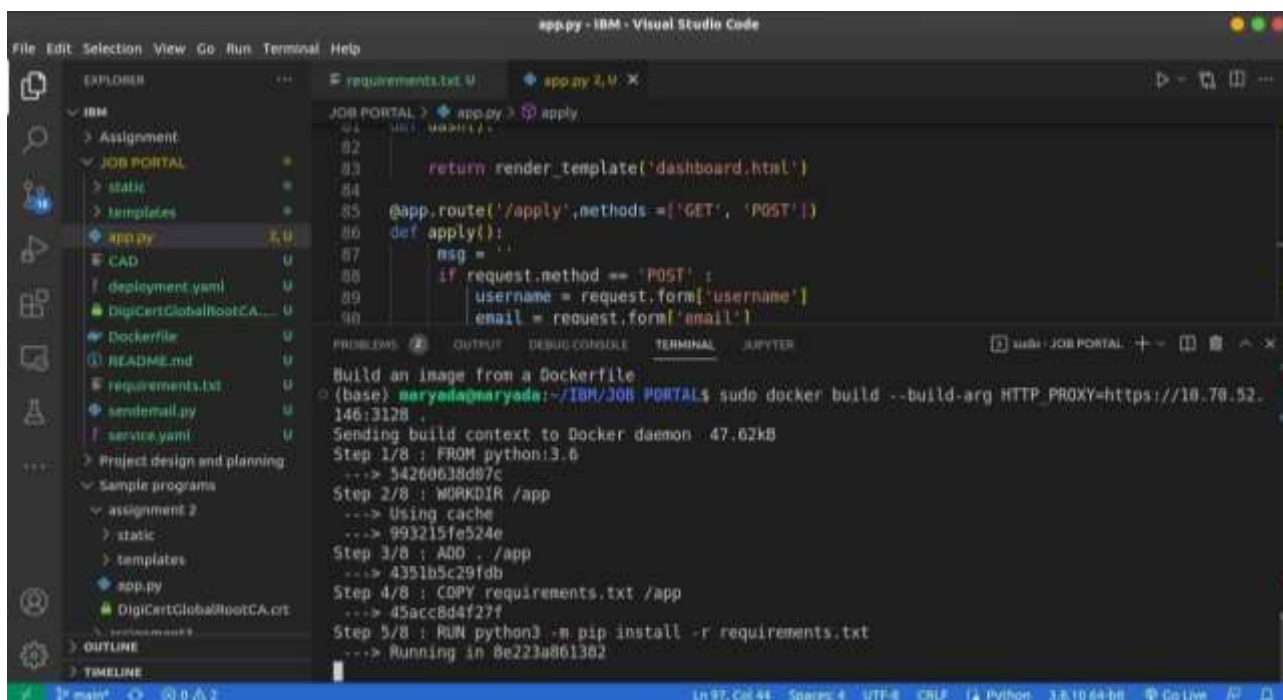
WORKDIR /app

ADD . /app

COPY requirements.txt /app

RUN python3 -m pip install -r requirements.txt

RUN `python3 -m pip install ibm_db`
EXPOSE 5000
CMD ["python","app.py"]

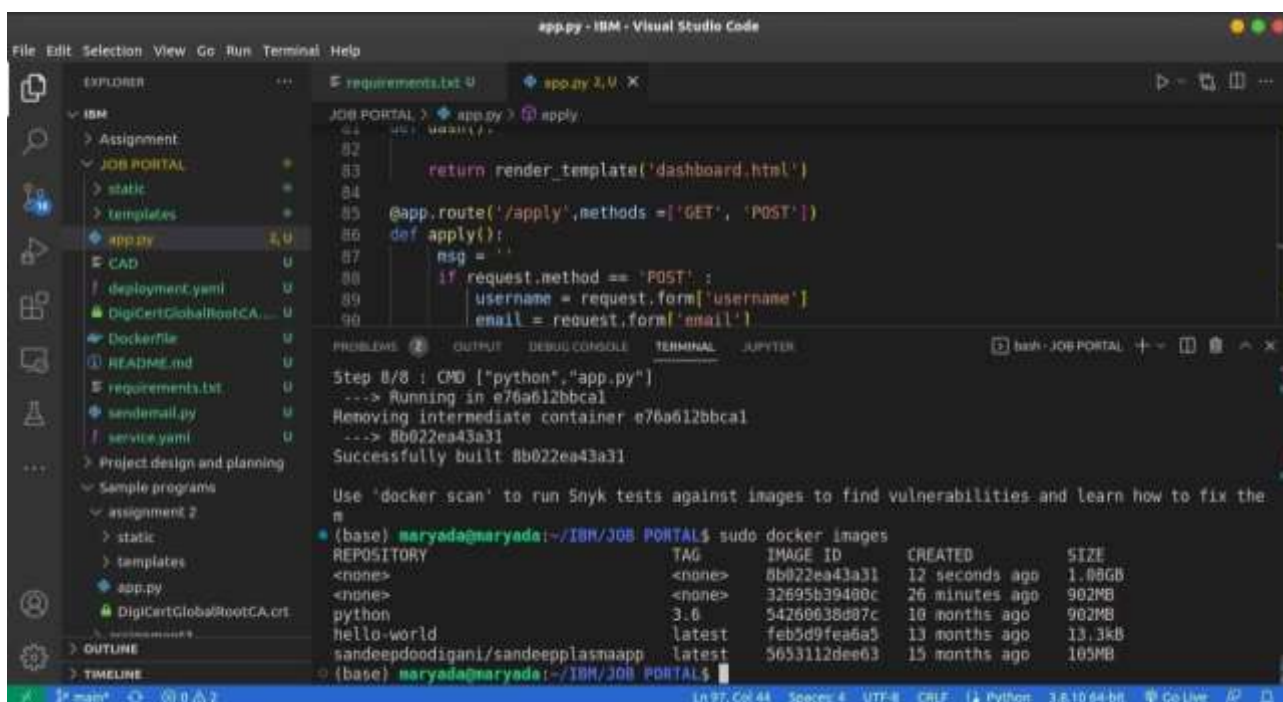


The screenshot shows the Visual Studio Code interface with the Explorer view on the left, the Editor view in the center, and the Terminal view at the bottom. The Explorer view shows a project structure with files like `requirements.txt`, `app.py`, `Dockerfile`, and `deployment.yaml`. The Editor view shows the `app.py` file with the following code:

```
11 from flask import Flask
12 app = Flask(__name__)
13
14 @app.route('/')
15 def index():
16     return render_template('dashboard.html')
17
18 @app.route('/apply', methods=['GET', 'POST'])
19 def apply():
20     msg = ''
21     if request.method == 'POST':
22         username = request.form['username']
23         email = request.form['email']
```

The Terminal view shows the output of the `docker build` command, which includes the following steps:

```
Build an image from a Dockerfile
(base) marya@marya:~/IBM/JOB PORTAL$ sudo docker build --build-arg HTTP_PROXY=https://10.70.52.146:3128 .
Sending build context to Docker daemon 47.62kB
Step 1/8 : FROM python:3.6
--> 54260638d87c
Step 2/8 : WORKDIR /app
--> Using cache
--> 993215fe524e
Step 3/8 : ADD . /app
--> 4351b5c29fdb
Step 4/8 : COPY requirements.txt /app
--> 45acc8d4f27f
Step 5/8 : RUN python3 -m pip install -r requirements.txt
--> Running in 8e223a861302
```



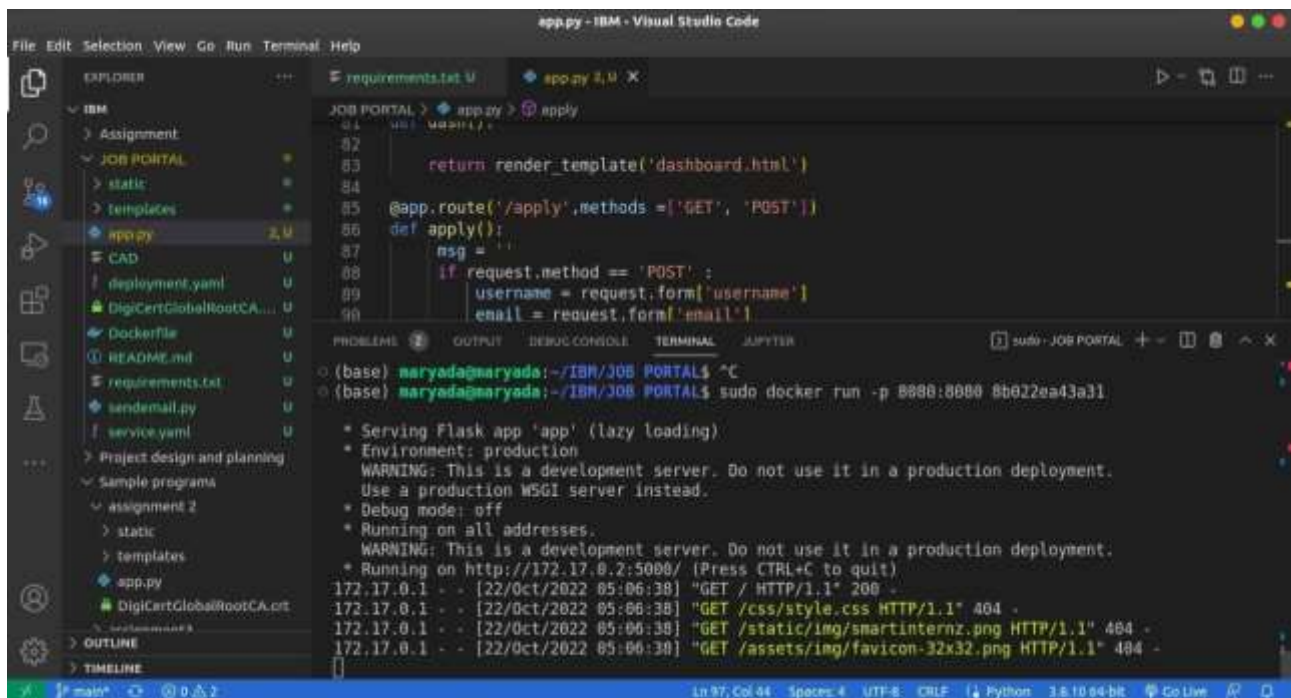
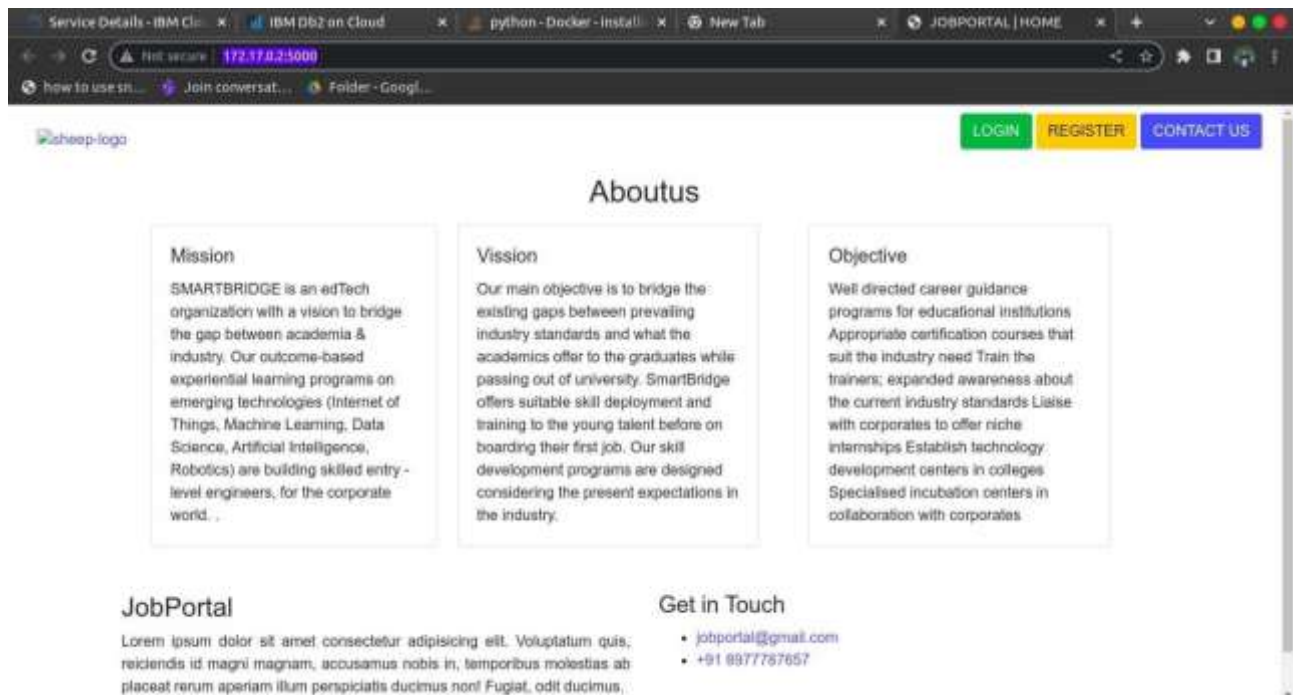
The screenshot shows the Visual Studio Code interface with the Explorer view on the left, the Editor view in the center, and the Terminal view at the bottom. The Explorer view shows a project structure with files like `requirements.txt`, `app.py`, `Dockerfile`, and `deployment.yaml`. The Editor view shows the `app.py` file with the following code:

```
11 from flask import Flask
12 app = Flask(__name__)
13
14 @app.route('/')
15 def index():
16     return render_template('dashboard.html')
17
18 @app.route('/apply', methods=['GET', 'POST'])
19 def apply():
20     msg = ''
21     if request.method == 'POST':
22         username = request.form['username']
23         email = request.form['email']
```

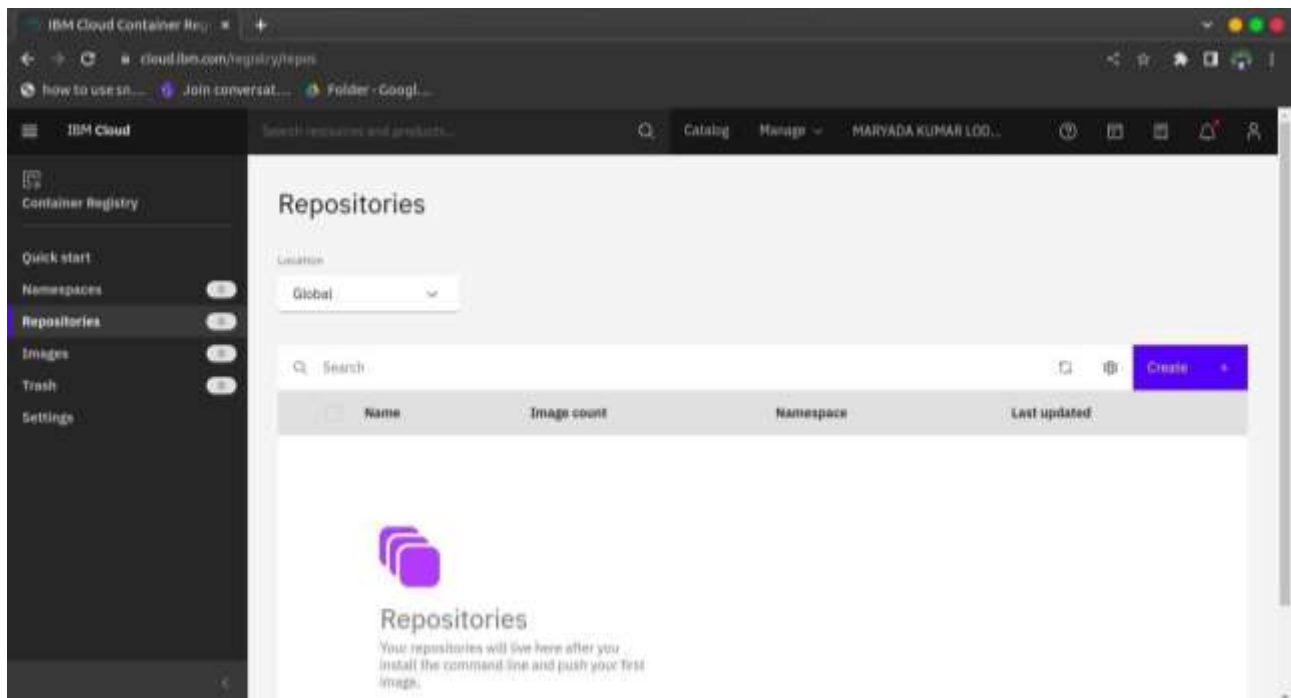
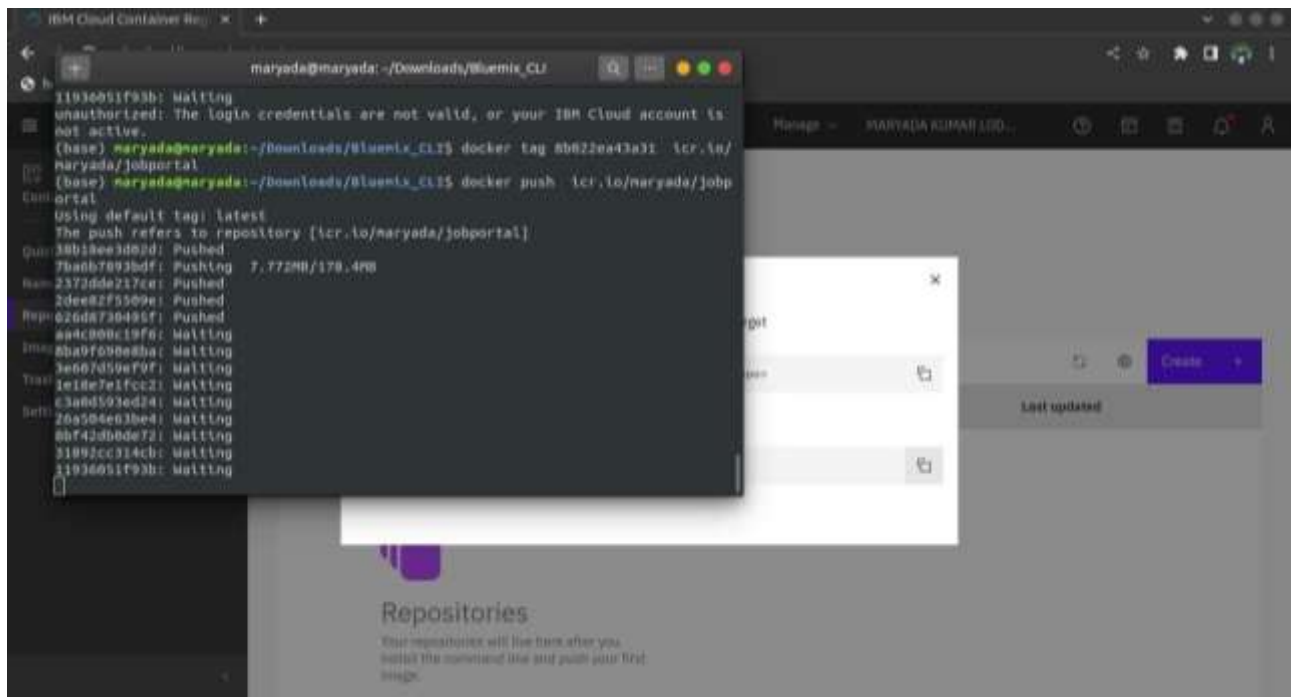
The Terminal view shows the output of the `docker build` command, which includes the following steps:

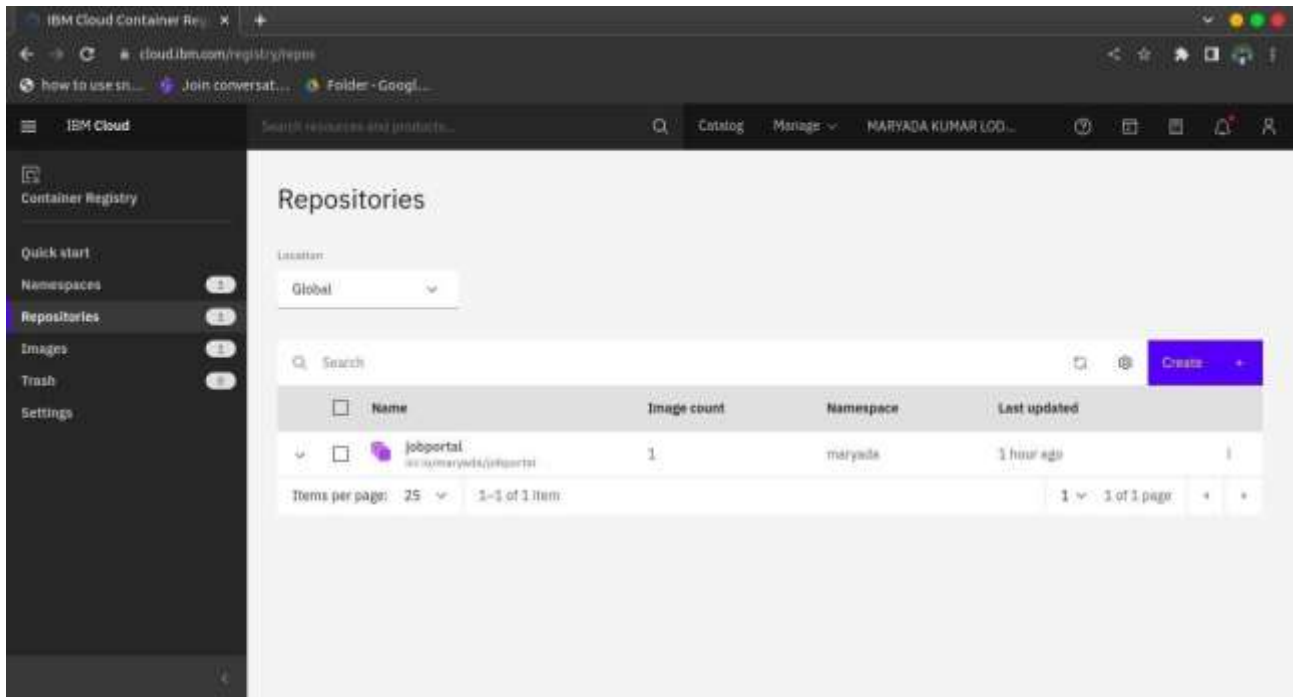
```
Step 6/8 : CMD ["python","app.py"]
--> Running in e76a612bbca1
Removing intermediate container e76a612bbca1
--> 8b022ea43a31
Successfully built 8b022ea43a31

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix the m
(base) marya@marya:~/IBM/JOB PORTAL$ sudo docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
<none>              <none>       8b022ea43a31     12 seconds ago  1.08GB
<none>              <none>       32695b39400c     26 minutes ago  902MB
python              3.6         54260638d87c     10 months ago  902MB
hello-world         latest      feb5d9fea6a5     13 months ago  13.3kB
sandeepdoodigani/sandeepplasmaapp latest      5653112dee63     15 months ago  105MB
(base) marya@marya:~/IBM/JOB PORTAL$
```

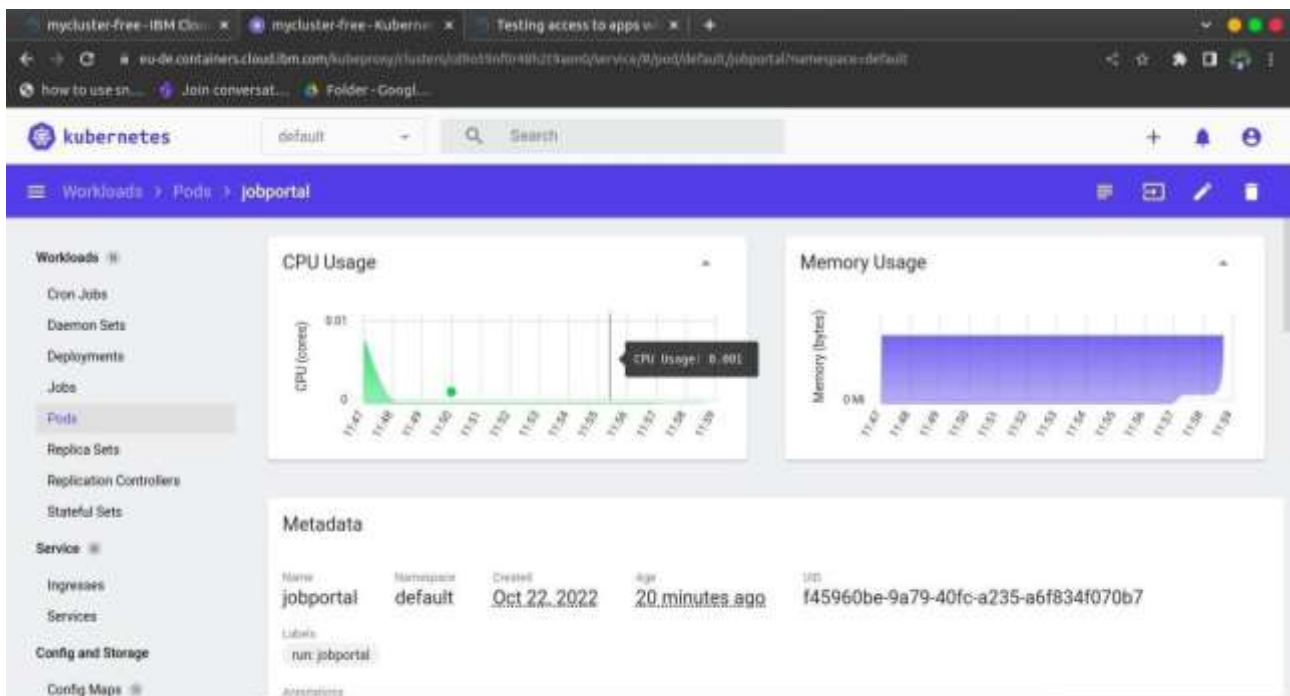


3. Create a IBM container registry and deploy helloworld app or jobportalapp.





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.



The screenshot shows the Kubernetes dashboard interface. At the top, there's a navigation bar with the 'kubernetes' logo and a search bar. Below this, the 'Pods' section is selected under 'Workloads'. On the left sidebar, there are links for 'Workloads', 'Service', 'Config and Storage', and 'Config Maps'. The main content area displays two graphs: 'CPU Usage' and 'Memory Usage'. The 'CPU Usage' graph shows a flat line at 0.01 cores over time. The 'Memory Usage' graph shows a flat line at 0 MB over time. Below these graphs is a table titled 'Pods' with columns for Name, Images, Labels, Node, Status, Restarts, and CPU Usage (cores). The table lists two pods: 'jobportal' and 'lb4-simple web-app-deployment'.

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)
jobportal	Show all	Show all	10.144.216.52	Running	0	0.01
lb4-simple web-app-deployment	Show all	Show all	10.144.216.52	ImagePullBackOff	-	-