Name	Sai Sudarshan S
Roll No	SSNCE195001309
Date	22 October 2022
Team ID	P2022 MID 53149
Project Name	Project - Personal Expense Tracker

Assignment - 4
Kubernetes and Docker

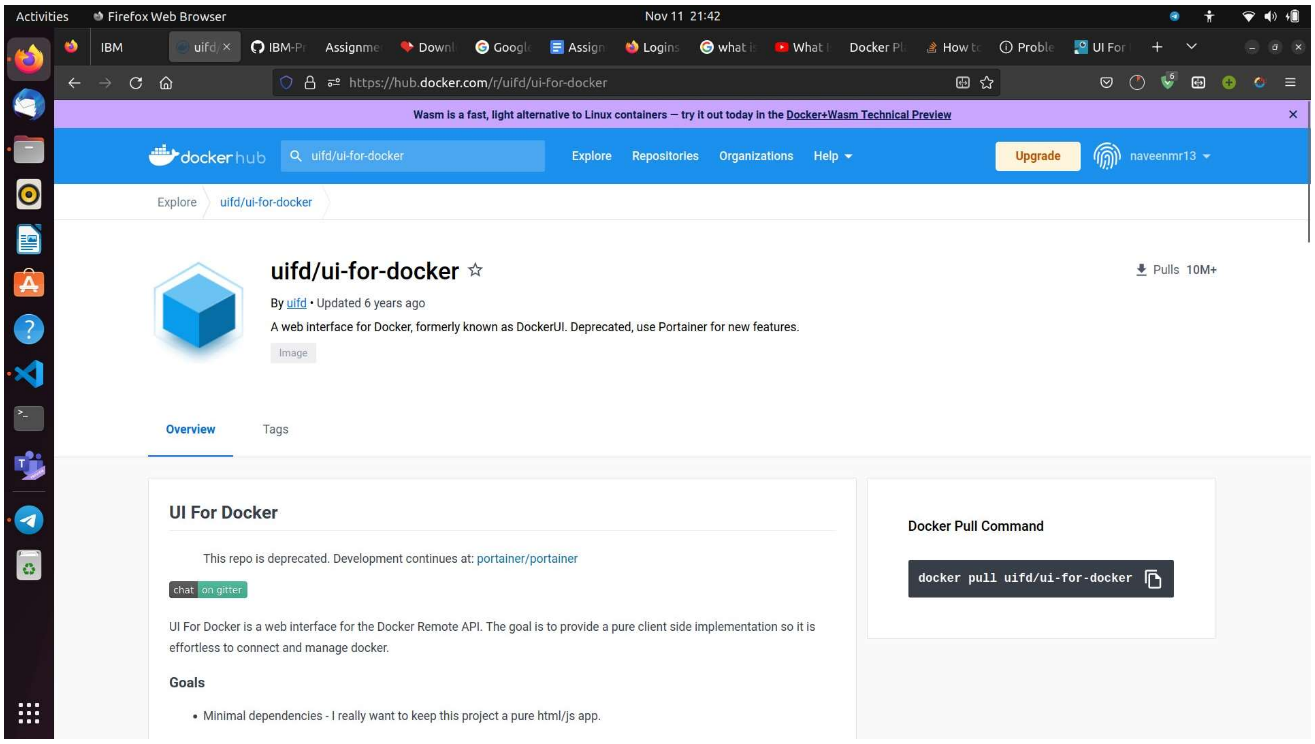
Question

1. Pull an Image from docker hub and run it in Docker Playground
2. Create a docker file for the job portal application and deploy it in Docker desktop application
3. Create a IBM container registry and deploy hello world app or job portal app
4. Create a Kubernetes cluster in IBM cloud and deploy hello world image or job portal 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



Wasm is a fast, light alternative to Linux containers — try it out today in the Docker+Wasm Technical Preview

uifd/ui-for-docker

By uifd • Updated 6 years ago

A web interface for Docker, formerly known as DockerUI. Deprecated, use Portainer for new features.

UI For Docker

This repo is deprecated. Development continues at: portainer/portainer

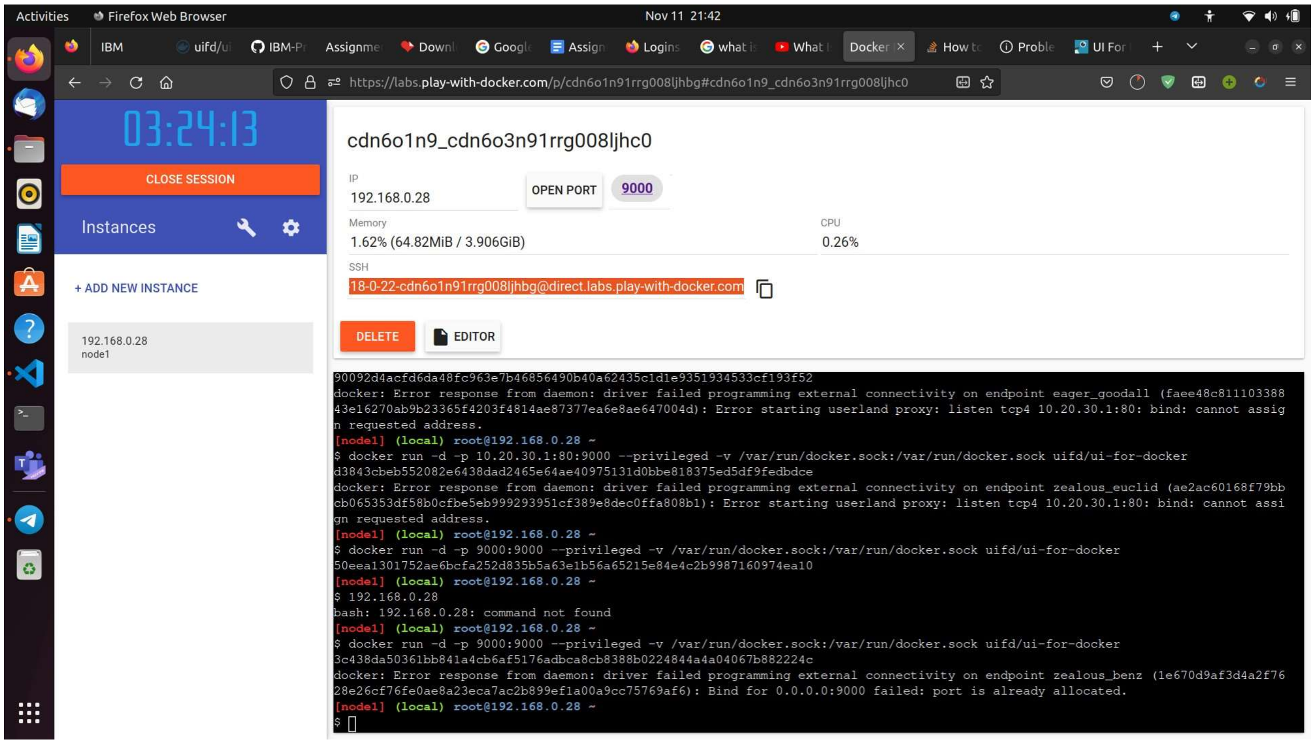
UI For Docker is a web interface for the Docker Remote API. The goal is to provide a pure client side implementation so it is effortless to connect and manage docker.

Goals

- Minimal dependencies - I really want to keep this project a pure html/js app.

Docker Pull Command

```
docker pull uifd/ui-for-docker
```



cdn6o1n9_cdn6o3n91rrg008ljhc0

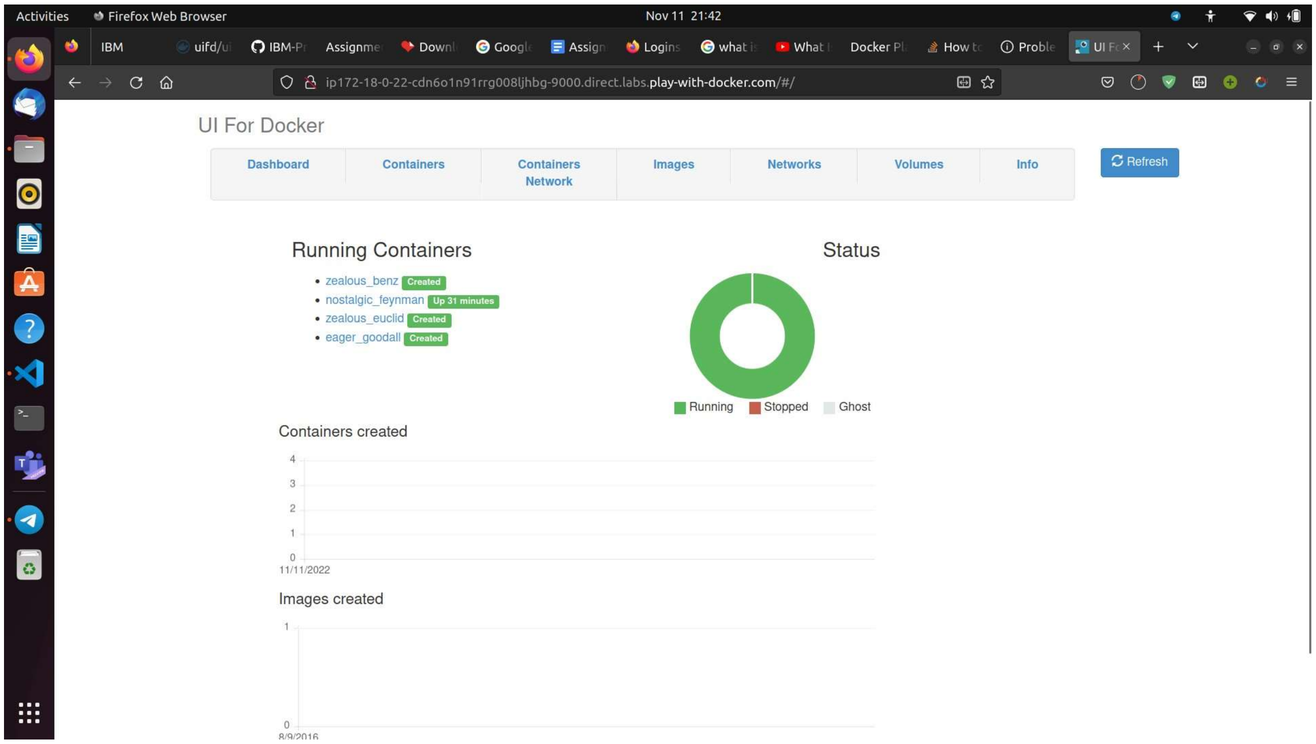
IP: 192.168.0.28

Memory: 1.62% (64.82MiB / 3.906GiB)

CPU: 0.26%

SSH: 18-0-22-cdn6o1n91rrg008ljhbg@direct.labs.play-with-docker.com

```
90092d4acfd6da48fc963e7b46856490b40a62435c1d1e9351934533cf193f52
docker: Error response from daemon: driver failed programming external connectivity on endpoint eager_goodall (faee48c811103388
43e16270ab9b23365f4203f4814ae87377ea6e8ae647004d): Error starting userland proxy: listen tcp4 10.20.30.1:80: bind: cannot assig
n requested address.
[node1] (local) root@192.168.0.28 ~
$ docker run -d -p 10.20.30.1:80:9000 --privileged -v /var/run/docker.sock:/var/run/docker.sock uifd/ui-for-docker
d3843cbeb552082e6438dad2465e64ae40975131d0bbe818375ed5df9fedbdce
docker: Error response from daemon: driver failed programming external connectivity on endpoint zealous_euclid (ae2ac60168f79bb
cb065353df58b0cfbe5eb999293951cf389e8dec0ffa808b1): Error starting userland proxy: listen tcp4 10.20.30.1:80: bind: cannot assi
gn requested address.
[node1] (local) root@192.168.0.28 ~
$ docker run -d -p 9000:9000 --privileged -v /var/run/docker.sock:/var/run/docker.sock uifd/ui-for-docker
50eea1301752ae6bcfa252d835b5a63e1b56a65215e84e4c2b9987160974ea10
[node1] (local) root@192.168.0.28 ~
$ 192.168.0.28
bash: 192.168.0.28: command not found
[node1] (local) root@192.168.0.28 ~
$ docker run -d -p 9000:9000 --privileged -v /var/run/docker.sock:/var/run/docker.sock uifd/ui-for-docker
3c438da50361bb841a4cb6af5176adbca8cb8388b0224844a4a04067b882224c
docker: Error response from daemon: driver failed programming external connectivity on endpoint zealous_benz (1e670d9af3d4a2f76
28e26cf76fe0ae8a23eca7ac2b899ef1a00a9cc75769af6): Bind for 0.0.0.0:9000 failed: port is already allocated.
[node1] (local) root@192.168.0.28 ~
$ 
```



Activities
Firefox Web Browser
Nov 11 21:42
ip172-18-0-22-cdn6o1n91rrg008ljhbg-9000.direct.labs.play-with-docker.com/#/

UI For Docker

Running Containers

- zealous_benz Created
- nostalgic_feynman Up 31 minutes
- zealous_euclid Created
- eager_goodall Created

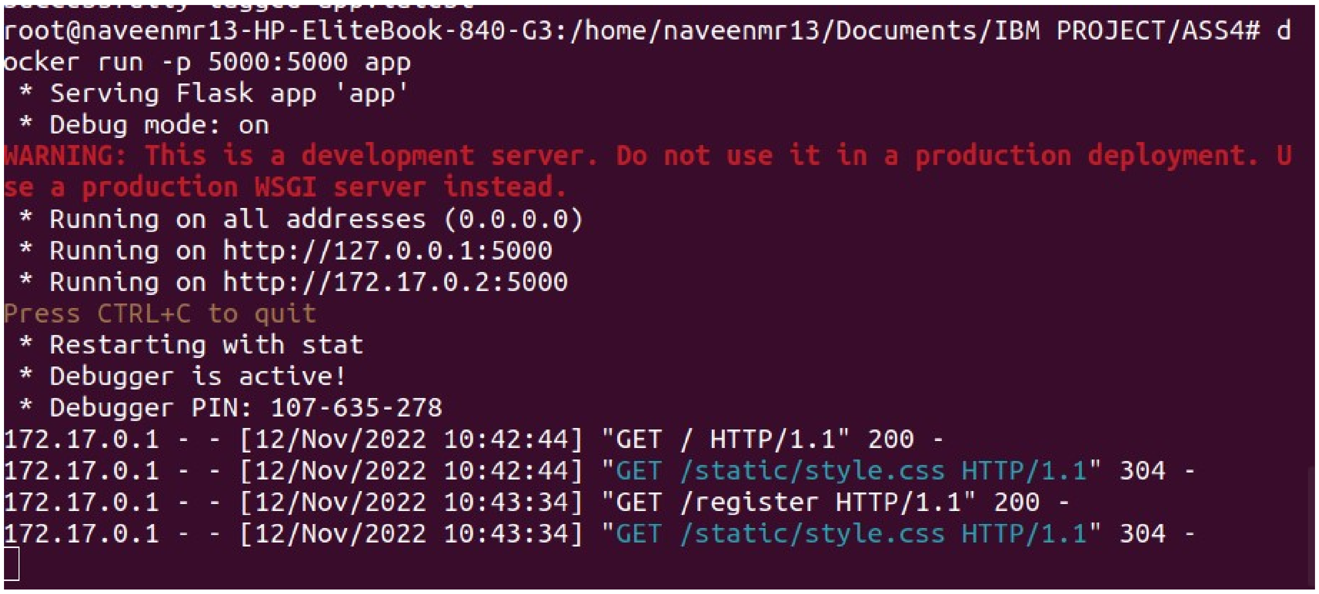
Status

2. **Create a docker file for the job portal application and deploy it in Docker desktop application**
    a. Create a docker file for build and deploy flask app.
    b. Use **`docker build -t image_name .`** in the current directory to start building the docker image and deploy in our local docker
    c. Use **`docker run -p 5000:5000 image_name`** to run in local system

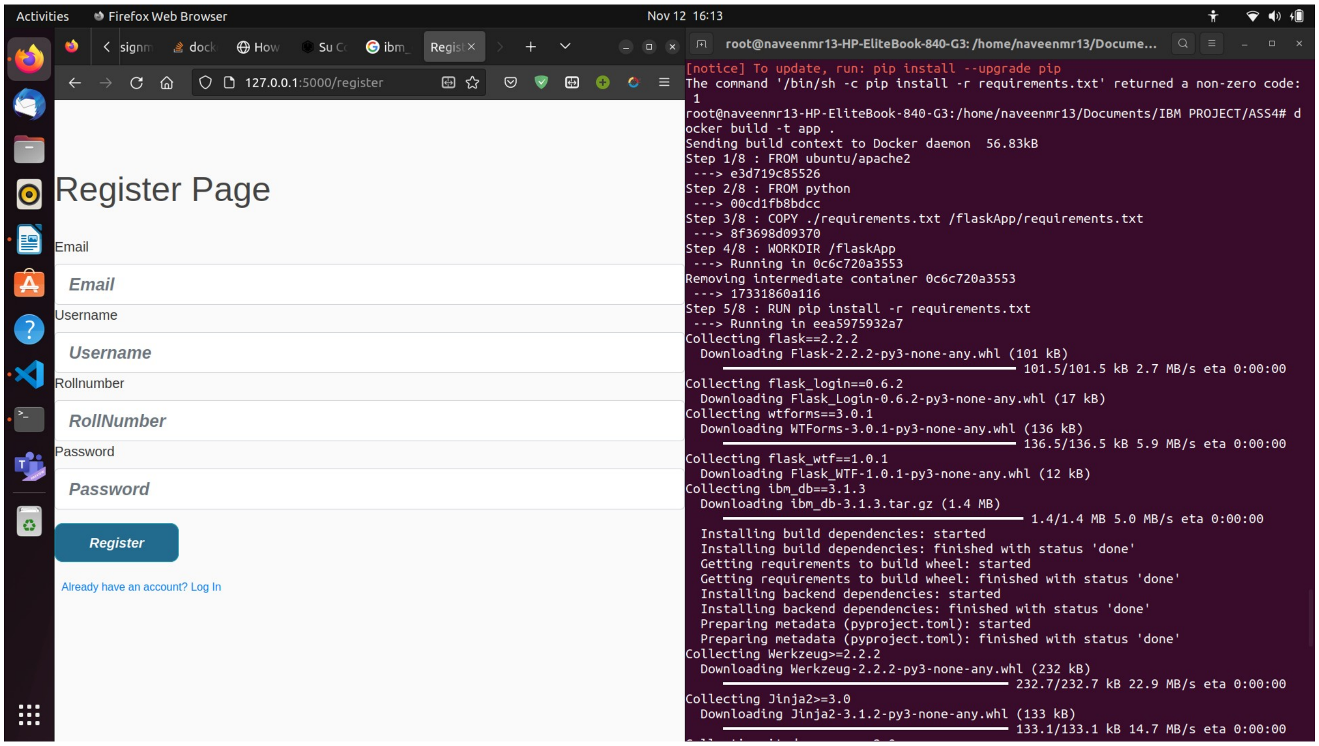
Dockerfile

```
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”]
```

Run locally using docker



```
root@naveenmr13-HP-EliteBook-840-G3:/home/naveenmr13/Documents/IBM PROJECT/ASS4# d
ocker run -p 5000:5000 app
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. U
se 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: 107-635-278
172.17.0.1 - - [12/Nov/2022 10:42:44] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2022 10:42:44] "GET /static/style.css HTTP/1.1" 304 -
172.17.0.1 - - [12/Nov/2022 10:43:34] "GET /register HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2022 10:43:34] "GET /static/style.css HTTP/1.1" 304 -
```



Register Page

Email

Email

Username

Username

Rollnumber

RollNumber

Password

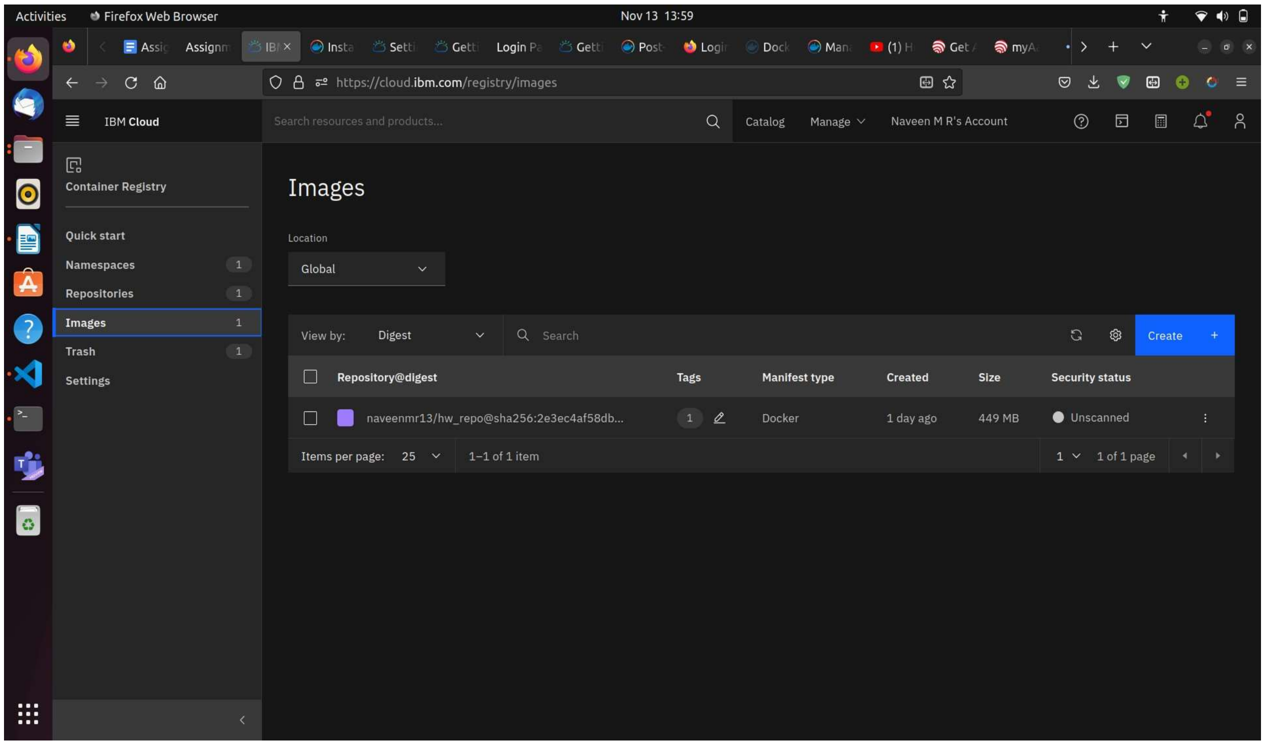
Password

Register

Already have an account? Log In

```
[notice] To update, run: pip install --upgrade pip
The command '/bin/sh -c pip install -r requirements.txt' returned a non-zero code:
 1
root@naveenmr13-HP-EliteBook-840-G3:/home/naveenmr13/Documents/IBM PROJECT/ASS4# d
ocker build -t app .
Sending build context to Docker daemon  56.83kB
Step 1/8 : FROM ubuntu/apache2
 ---> e3d719c85526
Step 2/8 : FROM python
 ---> 00cd1fb8bdcc
Step 3/8 : COPY ./requirements.txt /flaskApp/requirements.txt
 ---> 8f3698d09370
Step 4/8 : WORKDIR /flaskApp
 ---> Running in 0c6c720a3553
Removing intermediate container 0c6c720a3553
 ---> 17331860a116
Step 5/8 : RUN pip install -r requirements.txt
 ---> Running in eea5975932a7
Collecting flask==2.2.2
  Downloading Flask-2.2.2-py3-none-any.whl (101 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.5/101.5 kB 2.7 MB/s eta 0:00:00
Collecting flask_login==0.6.2
  Downloading Flask_Login-0.6.2-py3-none-any.whl (17 kB)
Collecting wtforms==3.0.1
  Downloading WTForms-3.0.1-py3-none-any.whl (136 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 136.5/136.5 kB 5.9 MB/s eta 0:00:00
Collecting flask_wtf==1.0.1
  Downloading Flask_WTF-1.0.1-py3-none-any.whl (12 kB)
Collecting ibm_db==3.1.3
  Downloading ibm_db-3.1.3.tar.gz (1.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 5.0 MB/s eta 0:00:00
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
  Installing backend dependencies: started
  Installing backend dependencies: finished with status 'done'
  Preparing metadata (pyproject.toml): started
  Preparing metadata (pyproject.toml): finished with status 'done'
Collecting Werkzeug>=2.2.2
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.7/232.7 kB 22.9 MB/s eta 0:00:00
Collecting Jinja2>=3.0
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.1/133.1 kB 14.7 MB/s eta 0:00:00
```

3. **Create a IBM container registry and deploy helloworld app or jobportal app**
    a. Log into IBM cloud
    b. Create a **container registry**
    c. Using IBM Cloud CLI, install the **container registry plugin** in our system
    d. Push our docker image into the created container registry using **docker push**
    e. So, our <u>job portal app is deployed</u> in the IBM container registry
    f.



Images

Location: Global

View by: Digest | Search | Create +

Repository@digest	Tags	Manifest type	Created	Size	Security status
naveenmr13/hw_repo@sha256:2e3ec4af58db...	1	Docker	1 day ago	449 MB	Unscanned

Items per page: 25 | 1–1 of 1 item | 1 | 1 of 1 page

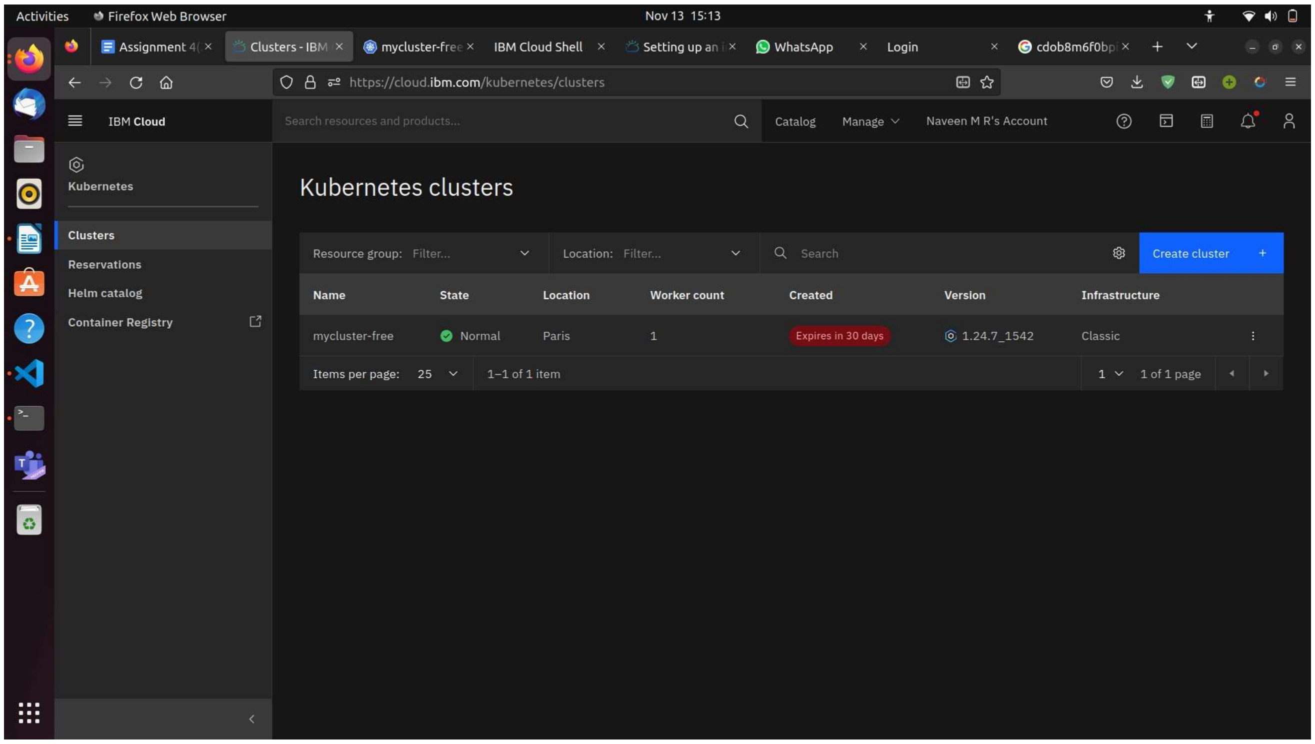
4. **Create a Kubernetes cluster in IBM cloud and deploy helloworld image or job portal image and also expose the same app to run in nodeport**
    a. Log into IBM cloud
    b. Create a **kubernete**
    c. Using IBM Cloud CLI, install the **ks plugin** in our system
    d. Create **a cluster** in the kubernetes
    e. Now, go to the **kubernetes dashboard** where we need to create a service based on a yml file (given below)
    f. In that file, we have to mention *which image we are going to use* and the *app name*
    g. Take the **public IP address** and **Nodeport** since we exposed the *flask app in nodeport*
    h. 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

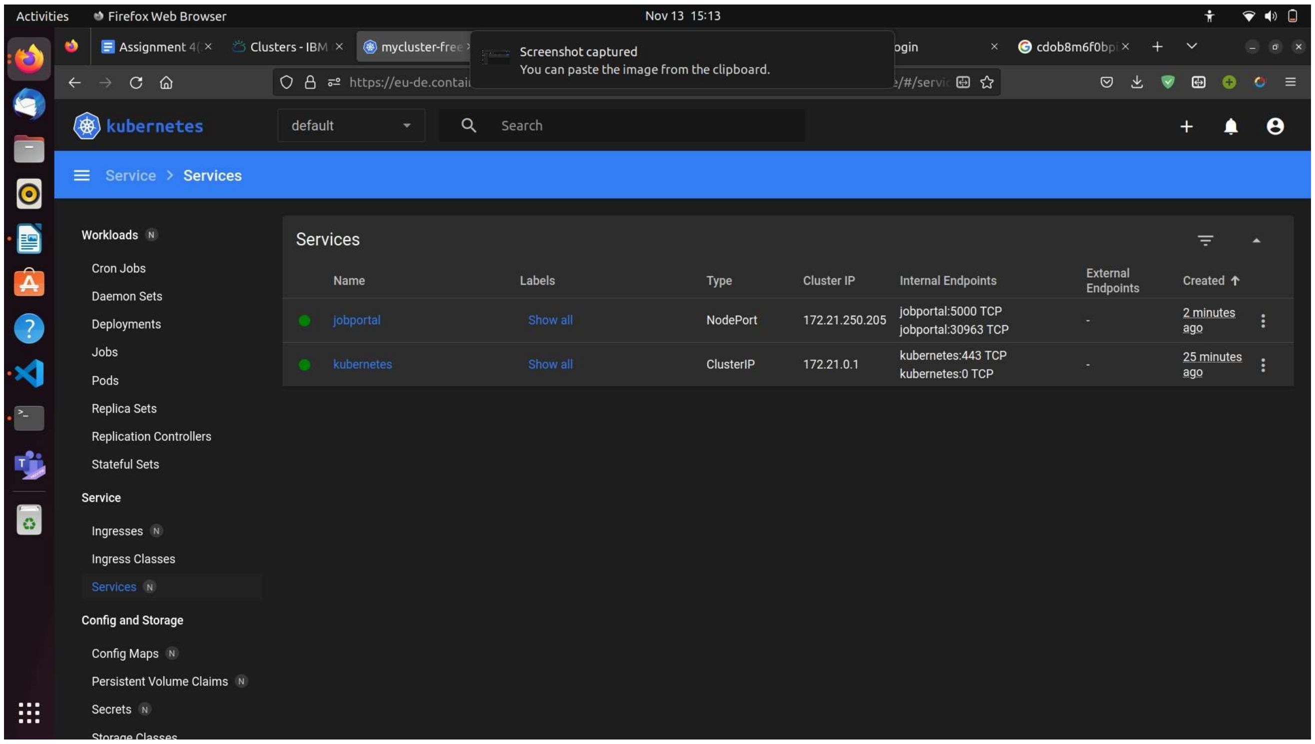


Kubernetes clusters

Name	State	Location	Worker count	Created	Version	Infrastructure
mycluster-free	Normal	Paris	1	Expires in 30 days	1.24.7_1542	Classic

Items per page: 25 1–1 of 1 item 1 of 1 page

Configuring the cluster

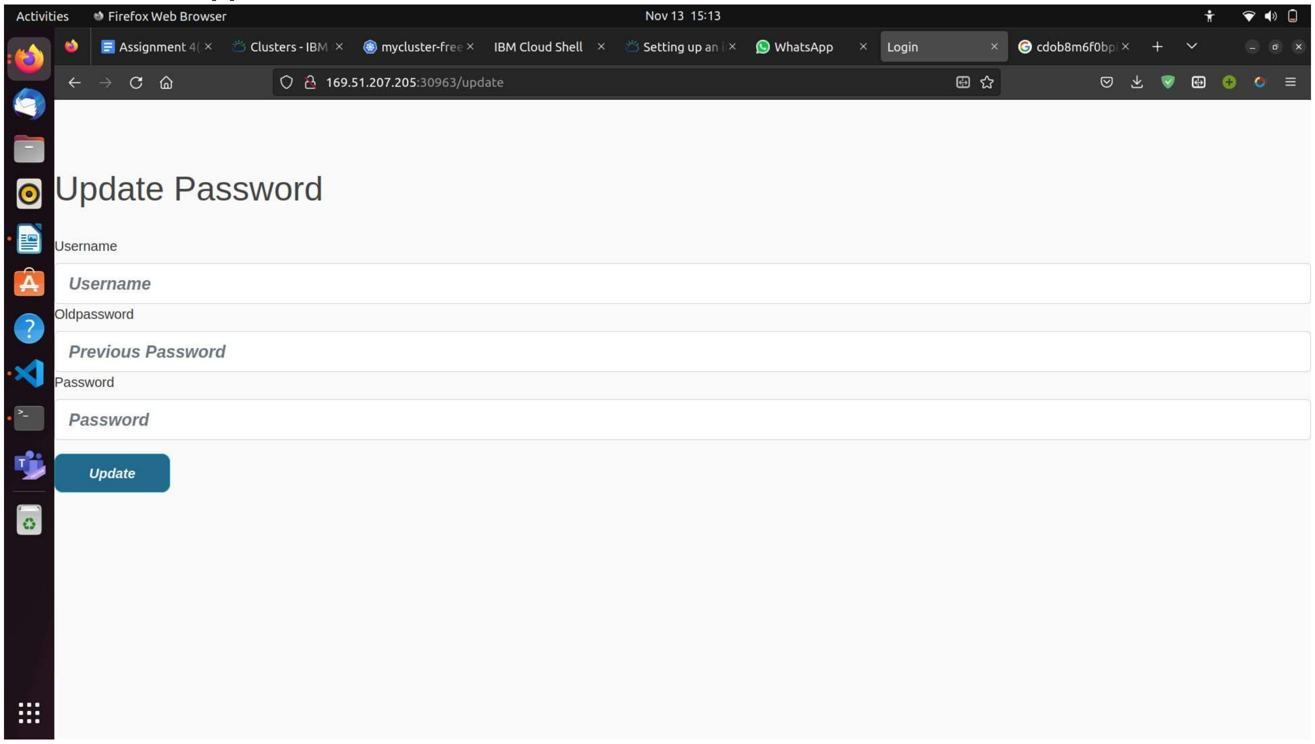


Screenshot captured
You can paste the image from the clipboard.

Services

Name	Labels	Type	Cluster IP	Internal Endpoints	External Endpoints	Created
jobportal	Show all	NodePort	172.21.250.205	jobportal:5000 TCP jobportal:30963 TCP	-	2 minutes ago
kubernetes	Show all	ClusterIP	172.21.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	25 minutes ago

Run our flask app in the IBM kubernetes



Update Password

Username

Username

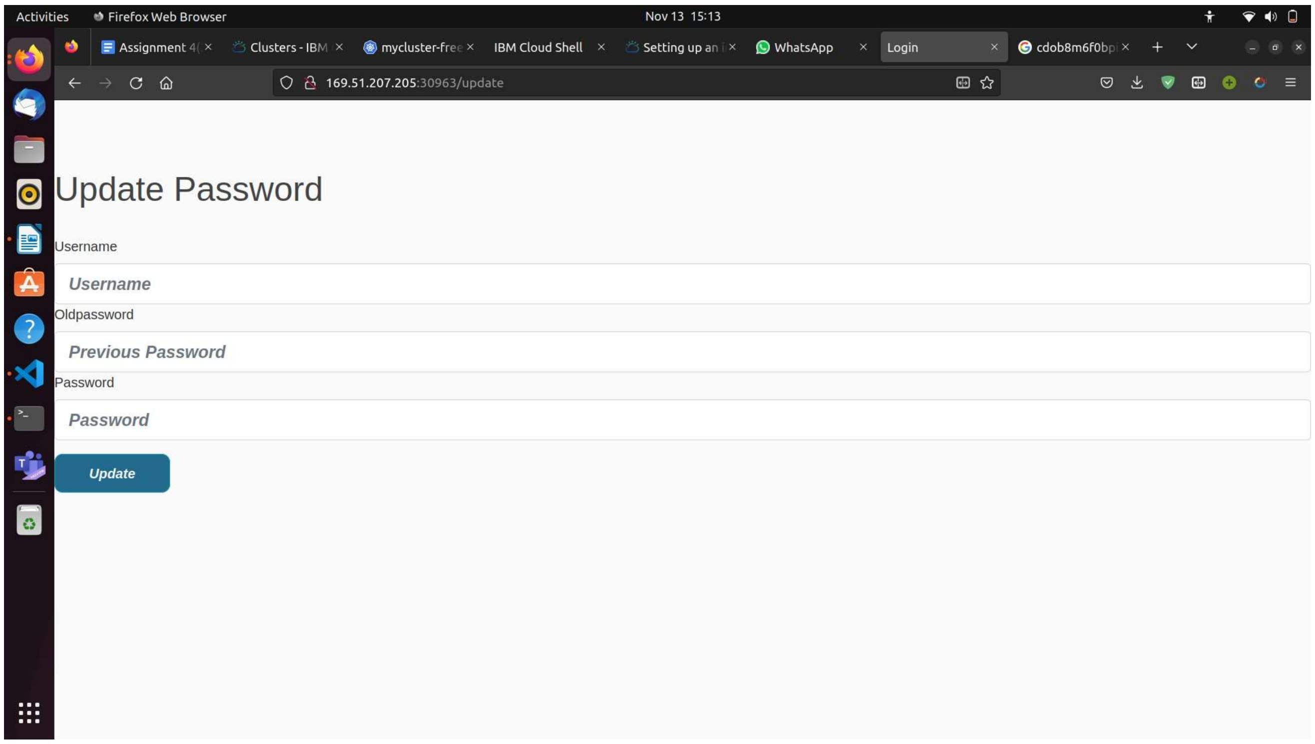
Oldpassword

Previous Password

Password

Password

Update



Update Password

Username

Username

Oldpassword

Previous Password

Password

Password

Update