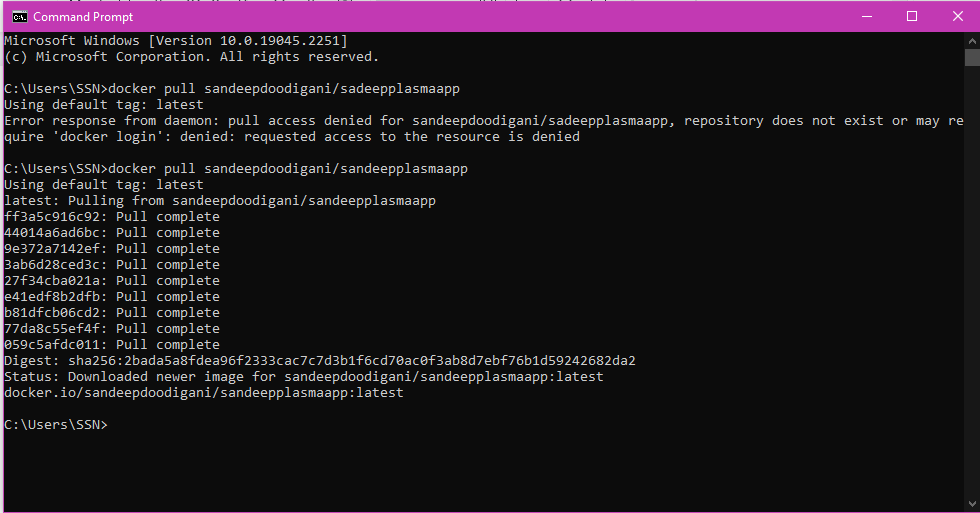
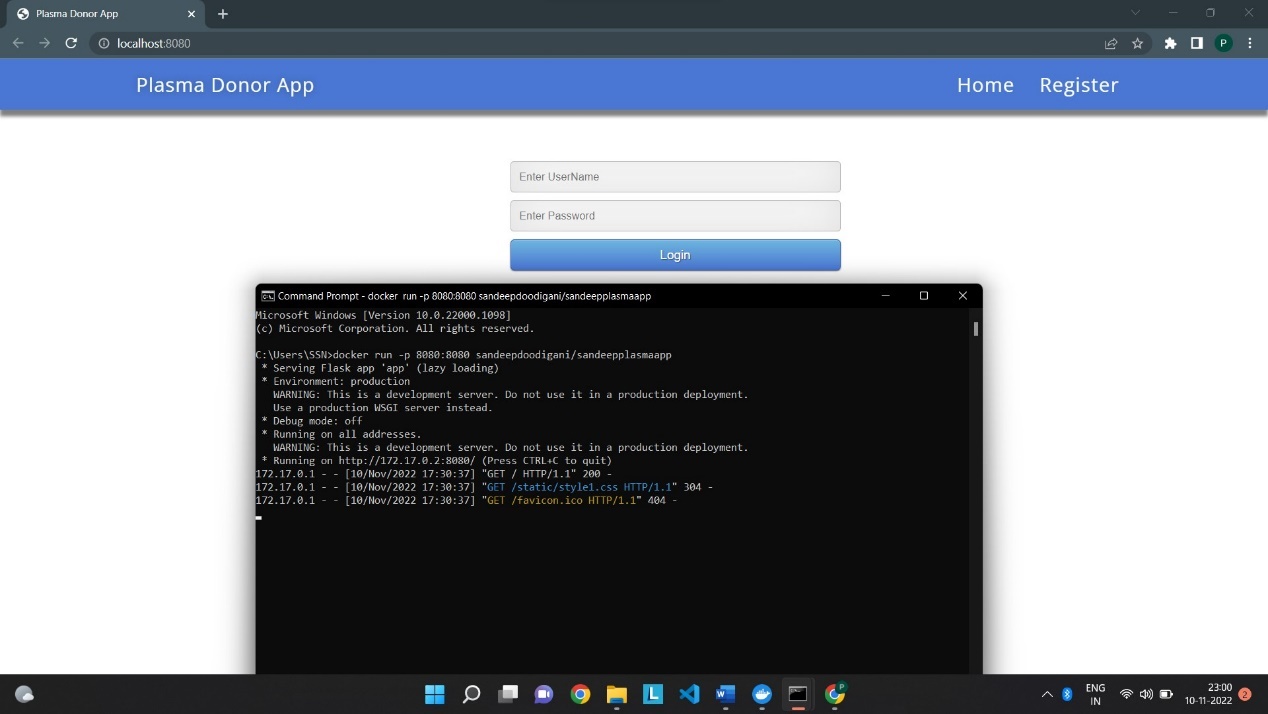
|  |  |  |
| --- | --- | --- |
| Name |  | Vignesh M |
| Roll No |  | SSNCE195001128 |
| Team ID |  | PNT2022TMID53102 |
| Project Name |  | Skill and Job Recommander |

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





Graphical user interface, application

Description automatically generated



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



**Program :**

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"]

Requirements.txt

Flask

ibm\_db

sendgrid

App.py

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

import ibm\_db

import re

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

app.secret\_key = 'a'

conn = ibm\_db.connect("DATABASE=bludb;HOSTNAME=b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32716;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=jzc43091;PWD=PI8VtGRvZlSVT65A",'','')

@app.route('/')

def homer():

    return render\_template('home.html')

@app.route('/login',methods =['GET', 'POST'])

def login():

    global userid

    msg = ''

    if request.method == 'POST' :

        username = request.form['username']

        password = request.form['password']

        sql = "SELECT \* FROM users WHERE username =? AND password=?"

        stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(stmt,1,username)

        ibm\_db.bind\_param(stmt,2,password)

        ibm\_db.execute(stmt)

        account = ibm\_db.fetch\_assoc(stmt)

        print (account)

        if account:

            session['loggedin'] = True

            session['id'] = account['USERNAME']

            userid=  account['USERNAME']

            session['username'] = account['USERNAME']

            msg = 'Logged in successfully !'

            msg = 'Logged in successfully !'

            return render\_template('dashboard.html', msg = msg)

        else:

            msg = 'Incorrect username / password !'

    return render\_template('login.html', msg = msg)

@app.route('/register', methods =['GET', 'POST'])

def registet():

    msg = ''

    if request.method == 'POST' :

        username = request.form['username']

        email = request.form['email']

        password = request.form['password']

        sql = "SELECT \* FROM users WHERE username =?"

        stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(stmt,1,username)

        ibm\_db.execute(stmt)

        account = ibm\_db.fetch\_assoc(stmt)

        print(account)

        if account:

            msg = 'Account already exists !'

        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

            msg = 'Invalid email address !'

        elif not re.match(r'[A-Za-z0-9]+', username):

            msg = 'name must contain only characters and numbers !'

        else:

            insert\_sql = "INSERT INTO  users VALUES (?, ?, ?)"

            prep\_stmt = ibm\_db.prepare(conn, insert\_sql)

            ibm\_db.bind\_param(prep\_stmt, 1, username)

            ibm\_db.bind\_param(prep\_stmt, 2, email)

            ibm\_db.bind\_param(prep\_stmt, 3, password)

            ibm\_db.execute(prep\_stmt)

            msg = 'You have successfully registered !'

    elif request.method == 'POST':

        msg = 'Please fill out the form !'

    return render\_template('register.html', msg = msg)

@app.route('/dashboard')

def dash():

    return render\_template('dashboard.html')

@app.route('/apply',methods =['GET', 'POST'])

def apply():

     msg = ''

     if request.method == 'POST' :

         username = request.form['username']

         email = request.form['email']

         qualification= request.form['qualification']

         skills = request.form['skills']

         jobs = request.form['s']

         sql = "SELECT \* FROM users WHERE username =?"

         stmt = ibm\_db.prepare(conn, sql)

         ibm\_db.bind\_param(stmt,1,username)

         ibm\_db.execute(stmt)

         account = ibm\_db.fetch\_assoc(stmt)

         print(account)

         if account:

            msg = 'there is only 1 job position! for you'

            return render\_template('apply.html', msg = msg)

         insert\_sql = "INSERT INTO  job VALUES (?, ?, ?, ?, ?)"

         prep\_stmt = ibm\_db.prepare(conn, insert\_sql)

         ibm\_db.bind\_param(prep\_stmt, 1, username)

         ibm\_db.bind\_param(prep\_stmt, 2, email)

         ibm\_db.bind\_param(prep\_stmt, 3, qualification)

         ibm\_db.bind\_param(prep\_stmt, 4, skills)

         ibm\_db.bind\_param(prep\_stmt, 5, jobs)

         ibm\_db.execute(prep\_stmt)

         msg = 'You have successfully applied for job !'

         session['loggedin'] = True

         TEXT = "Hello,a new application for job position" +jobs+"is requested"

     elif request.method == 'POST':

         msg = 'Please fill out the form !'

     return render\_template('apply.html', msg = msg)

@app.route('/display')

def display():

    print(session["username"],session['id'])

    cursor = mysql.connection.cursor()

    cursor.execute('SELECT \* FROM job WHERE userid = % s', (session['id'],))

    account = cursor.fetchone()

    print("accountdislay",account)

    return render\_template('display.html',account = account)

@app.route('/logout')

def logout():

   session.pop('loggedin', None)

   session.pop('id', None)

   session.pop('username', None)

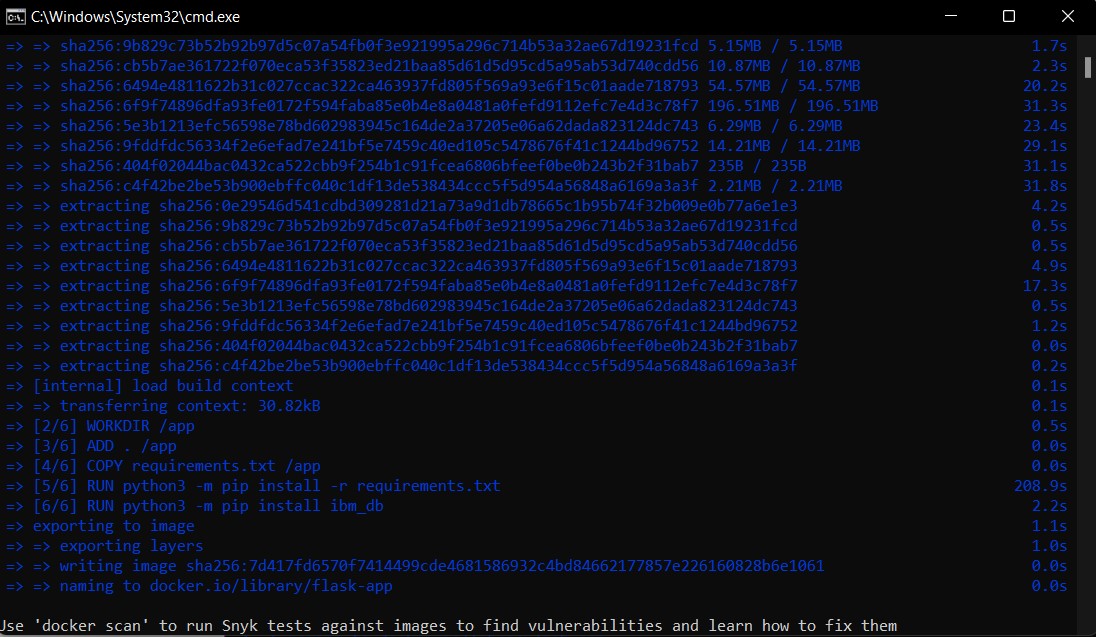
   return render\_template('home.html')

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

   app.run(host='0.0.0.0')

Text

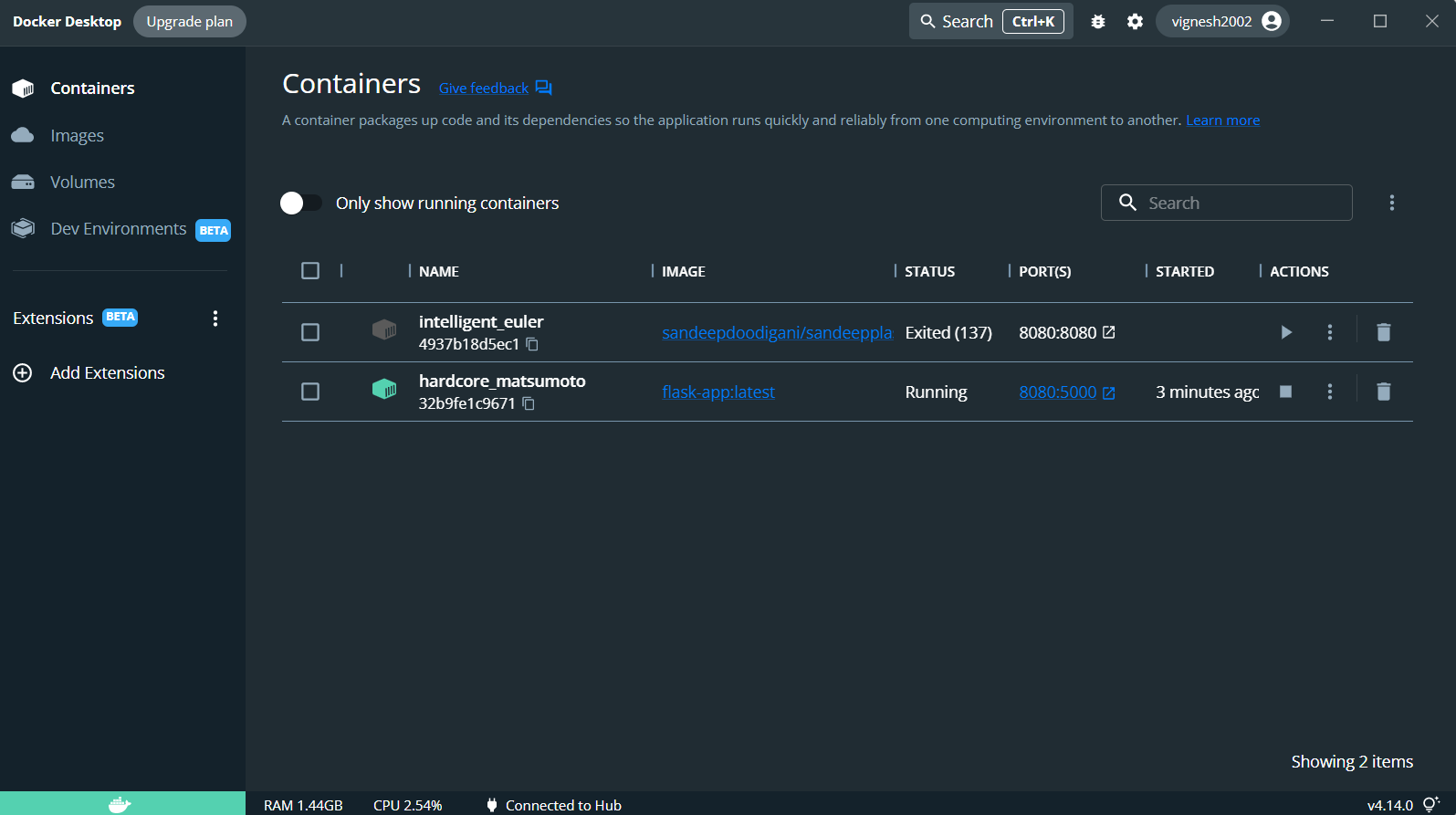
Description automatically generated



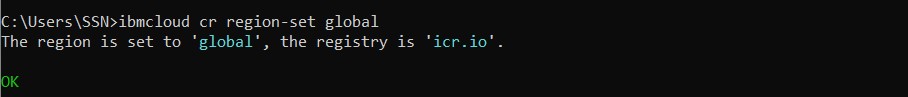
Text

Description automatically generated

C

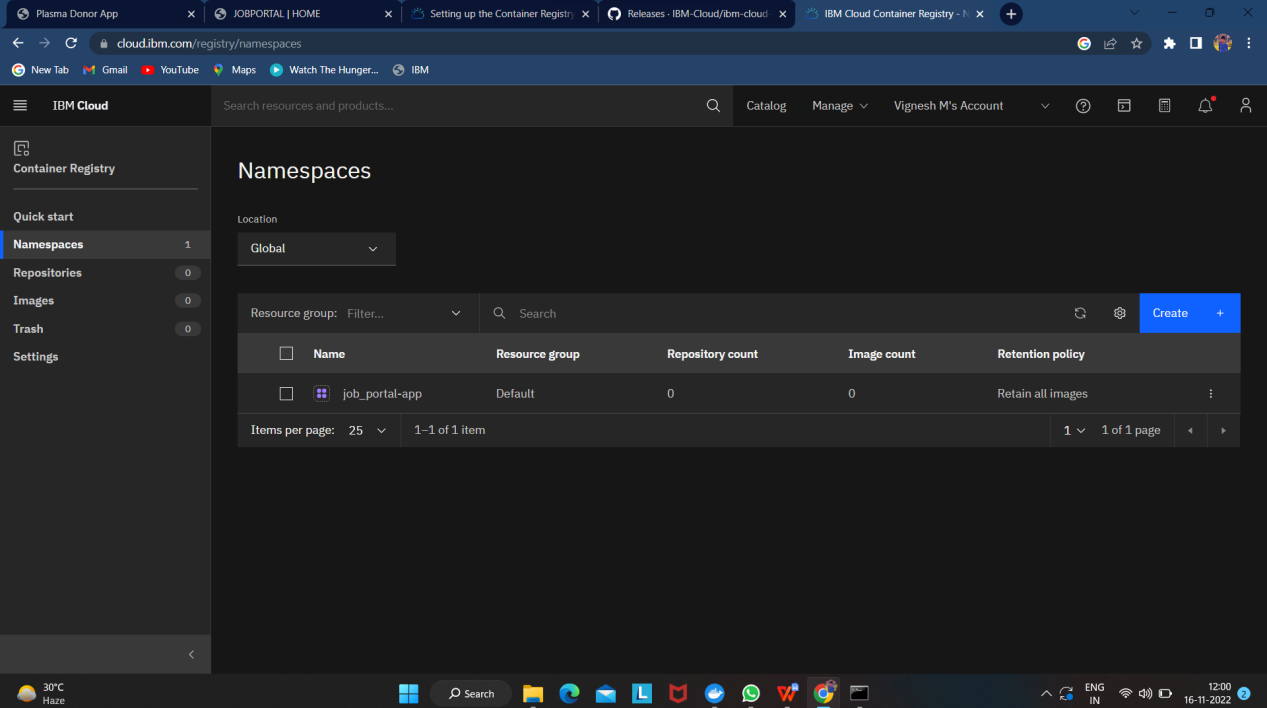


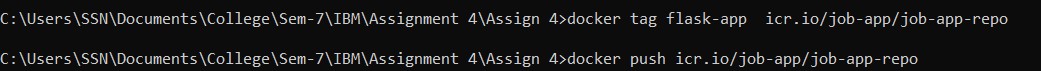
1. Create a IBM container registry and deploy helloworld app or jobportalapp.

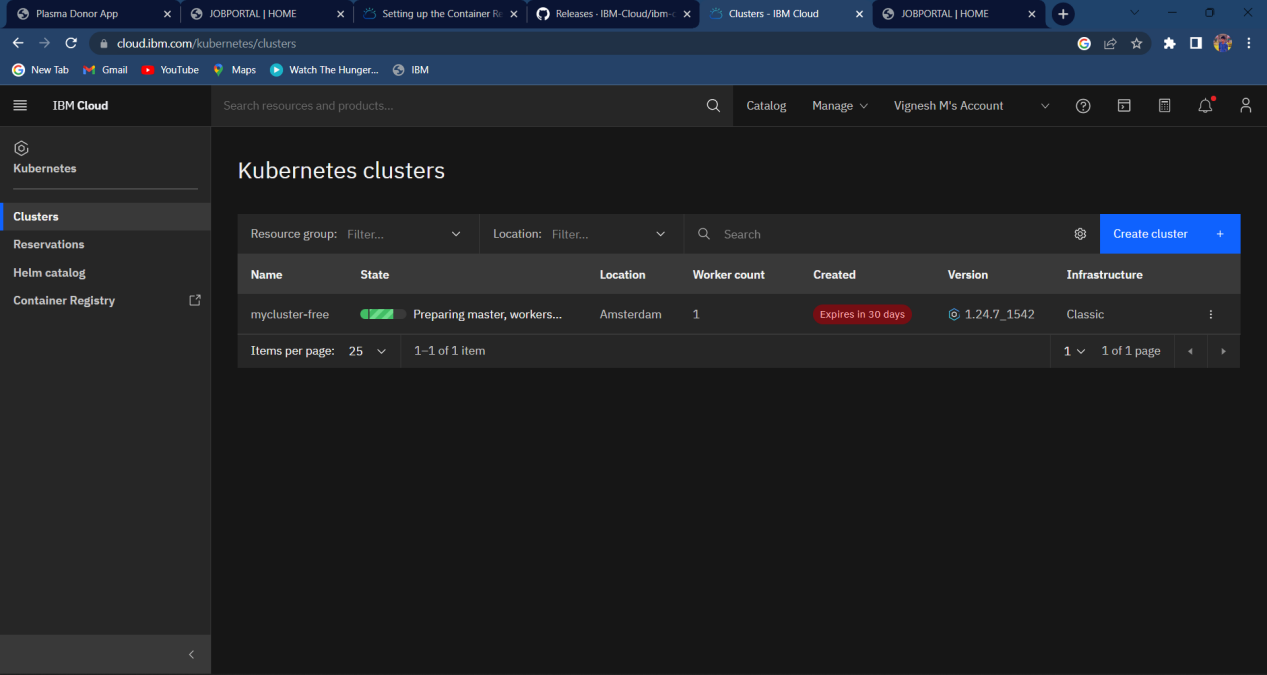


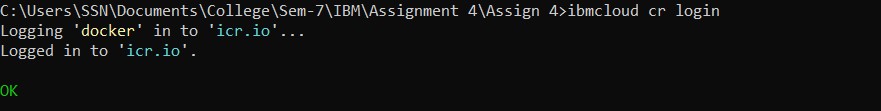
Text

Description automatically generated



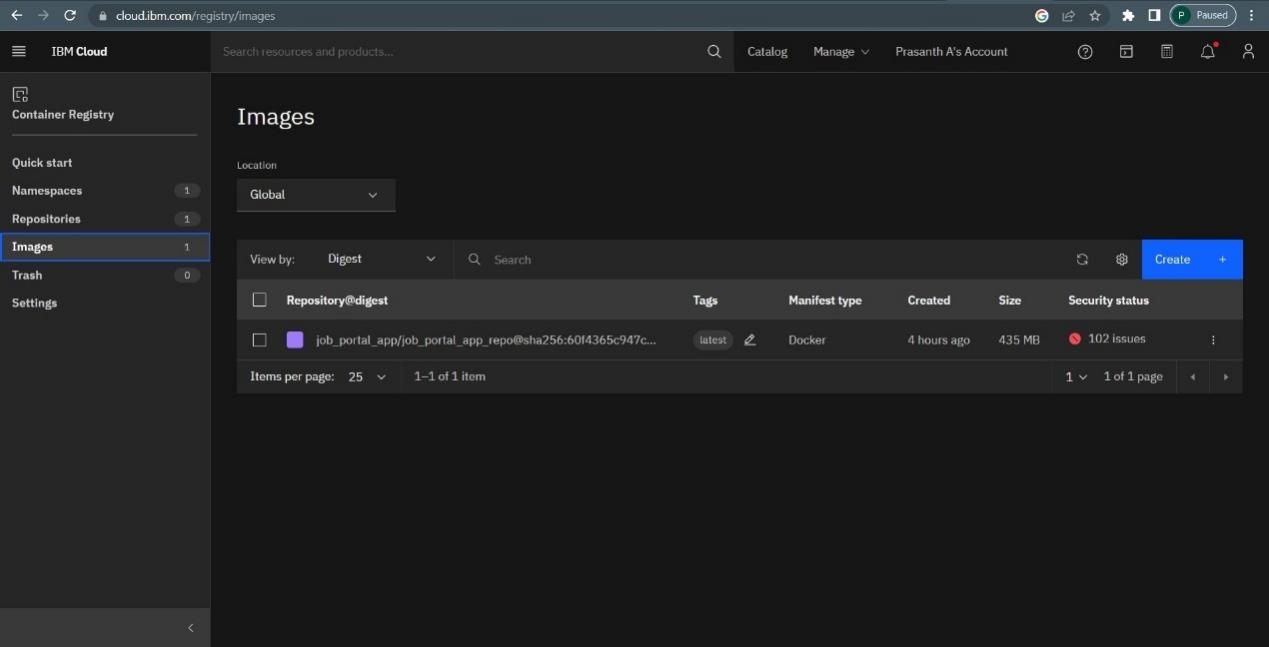






Text

Description automatically generated



Graphical user interface, text

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

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



Graphical user interface, application

Description automatically generated

Text

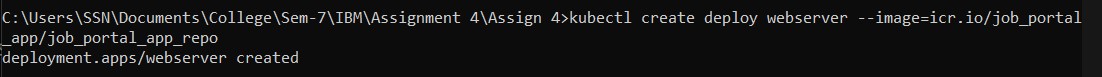
Description automatically generated

Text

Description automatically generated

Text

Description automatically generated



Graphical user interface, text

Description automatically generated