Assignment 4

Name:	Hemavarshini B
Roll No:	718019Z217
Project:	Inventory Management System for Retailers
Team ID:	PNT2022TMID12716

Dockerfile:

Dockerfile

```
FROM python:3.10.4
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
ENTRYPOINT ["python"]
CMD ["app.py"]
```

Building Image:

docker build -t <image_name> .
Execute the above command in the working directory where Dockerfile is present

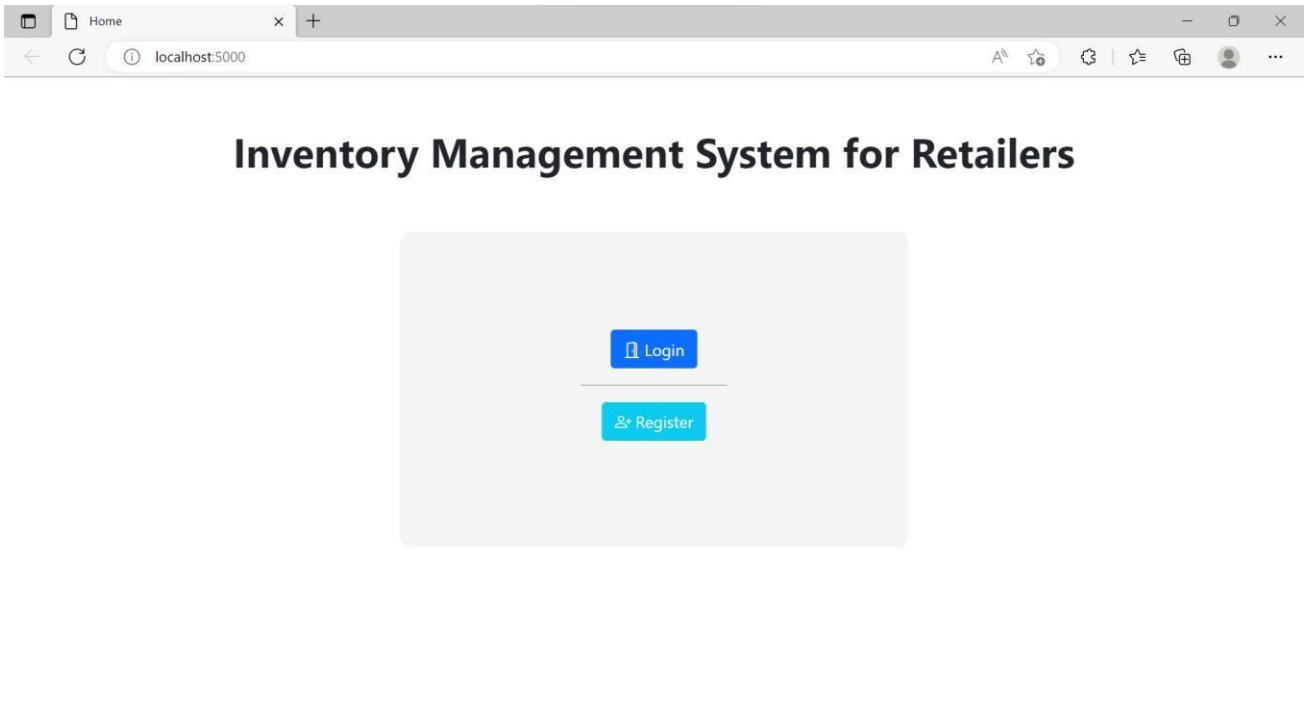
```
(myApp) D:\Inventory_Management_System_for_Retailers>docker build -t inventory-mgmt .
[+] Building 217.2s (12/12) FINISHED
 => [internal] load build definition from Dockerfile                                                0.1s
 => => transferring dockerfile: 252B                                                                0.0s
 => [internal] load .dockerignore                                                                   0.0s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load metadata for docker.io/library/python:3.10.4                                    2.6s
 => [3/6] ADD . /app                                                                                0.5s
 => [4/6] COPY requirements.txt /app                                                                0.1s
 => [5/6] RUN python3 -m pip install -r requirements.txt                                          208.1s
 => [6/6] RUN python3 -m pip install ibm_db                                                         1.0s
 => exporting to image                                                                              1.1s
 => => exporting layers                                                                             1.0s
 => => writing image sha256:82d0d6d55d7e358b0b69a8f2a38ea24804dcc39a81c5d3a6554bd1a1dcf8dc78        0.0s
 => => naming to docker.io/library/inventory-mgmt                                                   0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

(myApp) D:\Inventory_Management_System_for_Retailers>docker images
REPOSITORY                          TAG       IMAGE ID       CREATED              SIZE
inventory-mgmt                      latest    82d0d6d55d7e   About a minute ago   1.21GB
```

Run the container:

Docker run -d -p <port>:<port> <image_name>
Execute the command to run the container in the mentioned port number

```
(myApp) D:\Inventory_Management_System_for_Retailers>docker run -d -p 5000:5000 inventory-mgmt
25eb65cdf5cc7db4171174c02a8a73024a9228d72d440fce1c0556eb764d686f

(myApp) D:\Inventory_Management_System_for_Retailers>docker ps
CONTAINER ID   IMAGE           COMMAND           CREATED         STATUS         PORTS                    NAMES
25eb65cdf5cc   inventory-mgmt   "python app.py"   7 seconds ago   Up 6 seconds   0.0.0.0:5000->5000/tcp   affectionate_chatelet

(myApp) D:\Inventory_Management_System_for_Retailers>
```

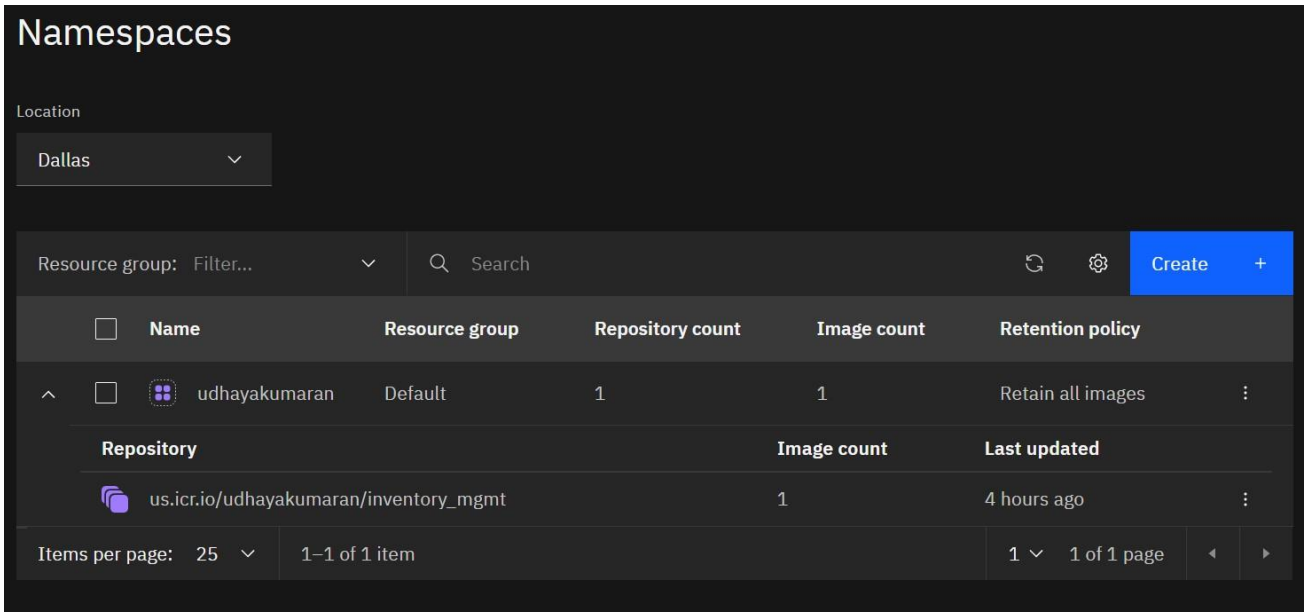
Open localhost:5000 or 127.0.0.1:5000 to view the running container



Tagging and pushing the image to the IBM Container Registry:

```
(myApp) D:\Inventory_Management_System_for_Retailers>ibmcloud cr login
Logging 'docker' in to 'us.icr.io'...
Logged in to 'us.icr.io'.

OK

(myApp) D:\Inventory_Management_System_for_Retailers>docker tag inventory-mgmt us.icr.io/udhayakumaran/inventory-mgmt:1.1

(myApp) D:\Inventory_Management_System_for_Retailers>docker push us.icr.io/udhayakumaran/inventory_mgmt:1.1
The push refers to repository [us.icr.io/udhayakumaran/inventory_mgmt]
b90e32367499: Layer already exists
2f535d5fc5c0: Layer already exists
0d2f265ad776: Layer already exists
dc00774c89ec: Layer already exists
fc9886f4d896: Layer already exists
9fda40ddc568: Layer already exists
428e1f341db7: Layer already exists
9ea8d200cd5d: Layer already exists
13b045a1dfd2: Layer already exists
2fbabeba902e: Layer already exists
ee509ed6e976: Layer already exists
9177197c67d0: Layer already exists
7dbadf2b9bd8: Layer already exists
e7597c345c2e: Layer already exists
1.1: digest: sha256:384ea4d17e3c6ca5099c39156b144c16e3e76065fcc77f0f9120878ba3d02152 size: 3263
```

IBM Container Registry UI:



deployment.yaml:

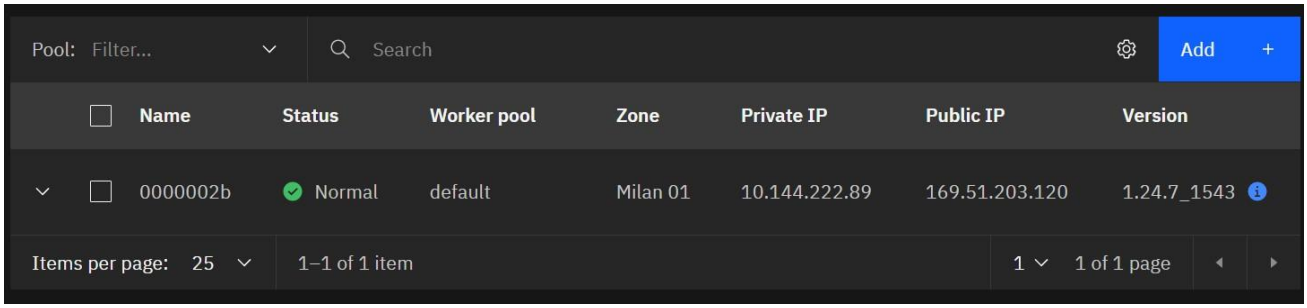
```yaml
! deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inventory-mgmt
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flasknode
  template:
    metadata:
      labels:
        app: flasknode
    spec:
      containers:
      - name: flasknode
        image: us.icr.io/udhayakumaran/inventory_mgmt:1.1
        imagePullPolicy: Always
        ports:
        - containerPort: 5000
```

Enter *ibmcloud ks cluster config -c <cluster_id>* to connect with the kubernetes cluster created in the IBM Cloud

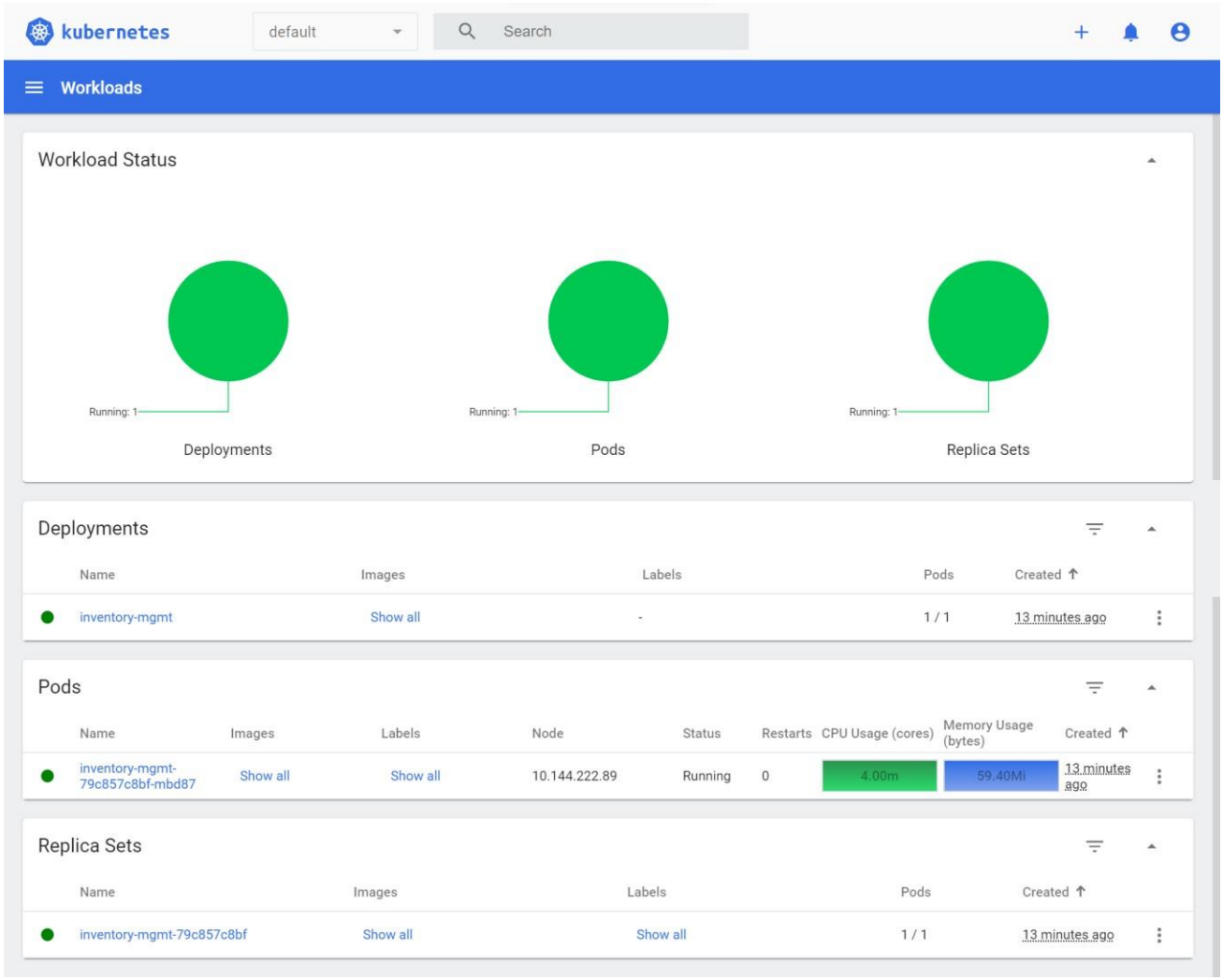
```
(myApp) D:\Inventory_Management_System_for_Retailers>ibmcloud ks cluster config -c [redacted]
OK
The configuration for cdticv7f05afjutac0hg was downloaded successfully.

Added context for cdticv7f05afjutac0hg to the current kubeconfig 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.

(myApp) D:\Inventory_Management_System_for_Retailers>kubectl create -f deployment.yaml
deployment.apps/inventory-mgmt created

(myApp) D:\Inventory_Management_System_for_Retailers>kubectl get deployment
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
inventory-mgmt   1/1     1            1           38s
```

After deploying, expose the application to public using the command; *kubectl expose deployment inventory-mgmt --type=NodePort --name=inventory-mgmt*

Kubernetes Cluster Worker Node



Kubernetes Dashboard



Application running on IBM Kubernetes Cluster