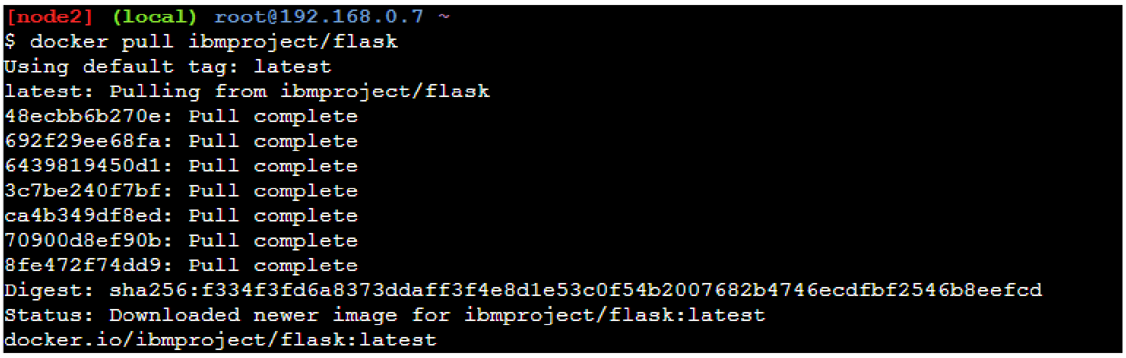
Assignment -4

Docker and Kubernetes

Student Name	Aravindan MK
Student Roll Number	910019106703

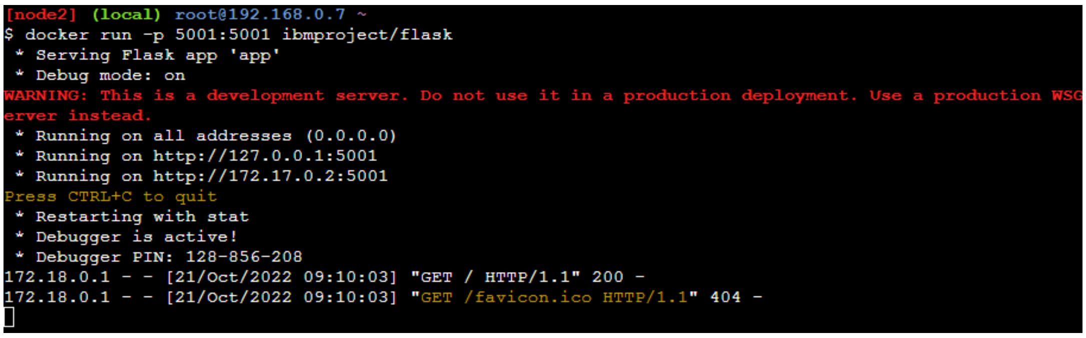
1. Pull an Image from docker hub and run it in docker playground.

Pull Image from docker hub



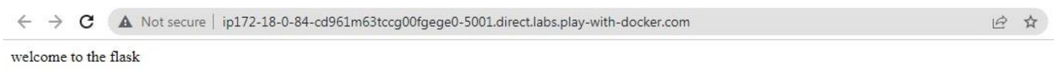
```
[node2] (local) root@192.168.0.7 ~
$ docker pull ibmproject/flask
Using default tag: latest
latest: Pulling from ibmproject/flask
48ecbb6b270e: Pull complete
692f29ee68fa: Pull complete
6439819450d1: Pull complete
3c7be240f7bf: Pull complete
ca4b349df8ed: Pull complete
70900d8ef90b: Pull complete
8fe472f74dd9: Pull complete
Digest: sha256:f334f3fd6a8373ddaff3f4e8d1e53c0f54b2007682b4746ecdfbf2546b8eefcd
Status: Downloaded newer image for ibmproject/flask:latest
docker.io/ibmproject/flask:latest
```

Running the image



```
[node2] (local) root@192.168.0.7 ~
$ docker run -p 5001:5001 ibmproject/flask
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSG
erver instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://172.17.0.2:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 128-856-208
172.18.0.1 - - [21/Oct/2022 09:10:03] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [21/Oct/2022 09:10:03] "GET /favicon.ico HTTP/1.1" 404 -
```

Output



Not secure | ip172-18-0-84-cd961m63tccg00fgege0-5001.direct.labs.play-with-docker.com

welcome to the flask

Flask program Containerized and run

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
        return "welcome to the flask "
```

```
if __name__ == "__main__":
        app.run(host ='0.0.0.0', port = 5001, debug = True)
```

Dockerfile

```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
EXPOSE 5001
ENTRYPOINT [ "python" ]
CMD [ "app.py" ]
```

2.Create a dockerfile and deploy it in docker desktop

Flask application

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
        return "welcome to the flask"


if __name__ == "__main__":
        app.run(host ='0.0.0.0', port = 5001, debug = True)
```

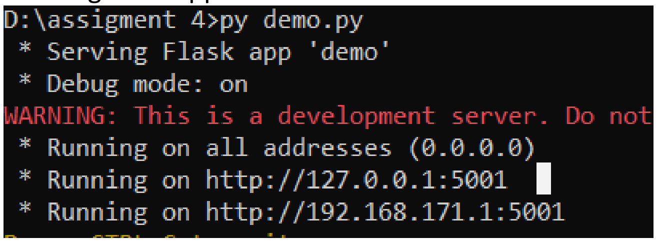
Dockerfile

```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
EXPOSE 5001
ENTRYPOINT [ "python" ]
CMD [ "demo.py" ]
```

Requirement.txt

```
Flask
```

Running Flask app

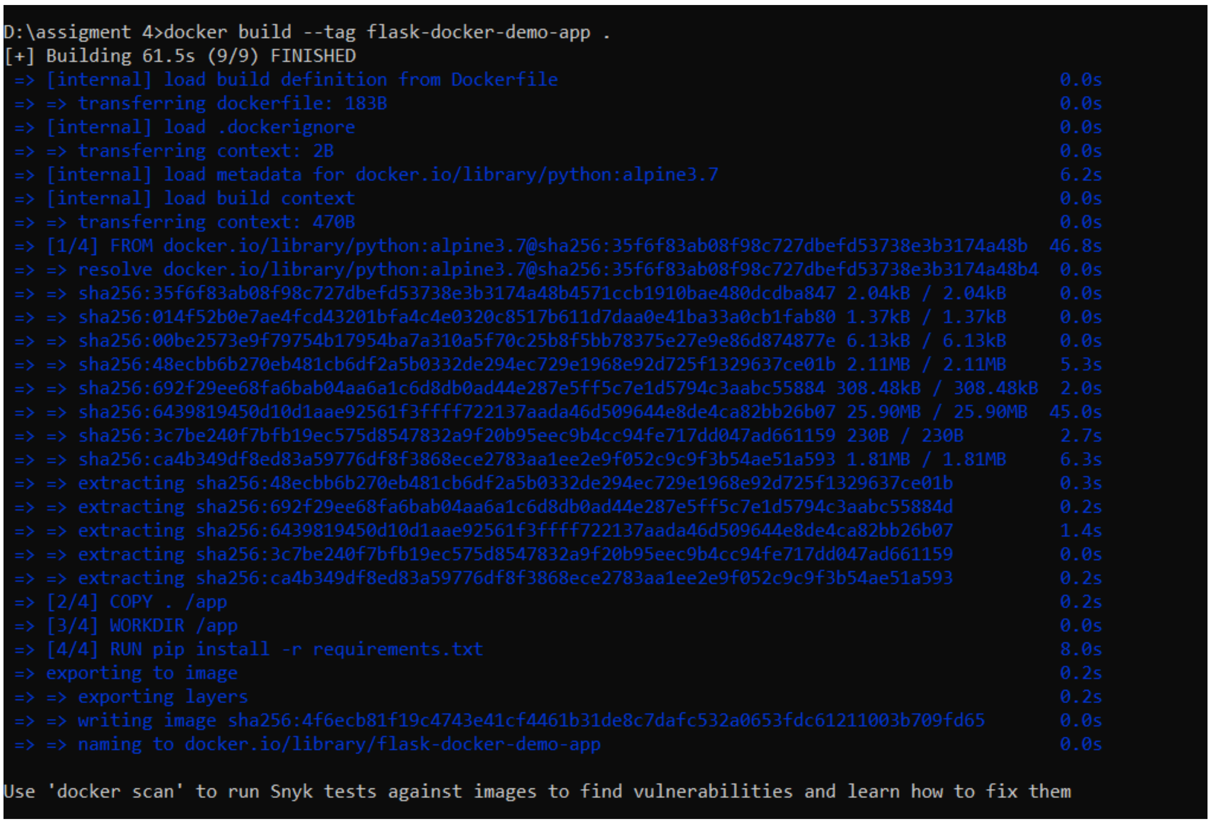


```
D:\assigment 4>py demo.py
 * Serving Flask app 'demo'
 * Debug mode: on
WARNING: This is a development server. Do not
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://192.168.171.1:5001
```

Flask app output

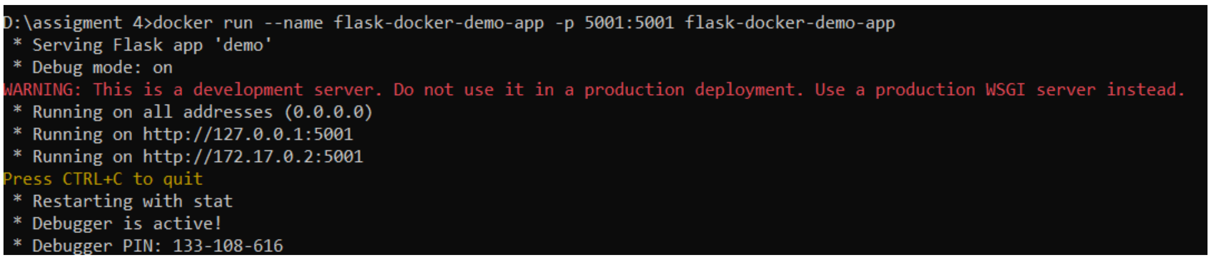
welcome to the flask

Build docker image



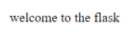
```
D:\assigment 4>docker build --tag flask-docker-demo-app .
[+] Building 61.5s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                               0.0s
 => => transferring dockerfile: 183B                                                               0.0s
 => [internal] load .dockerignore                                                                  0.0s
 => => transferring context: 2B                                                                    0.0s
 => [internal] load metadata for docker.io/library/python:alpine3.7                                6.2s
 => [internal] load build context                                                                  0.0s
 => => transferring context: 470B                                                                  0.0s
 => [1/4] FROM docker.io/library/python:alpine3.7@sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b  46.8s
 => => resolve docker.io/library/python:alpine3.7@sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4  0.0s
 => => sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571ccb1910bae480dcdba847 2.04kB / 2.04kB     0.0s
 => => sha256:014f52b0e7ae4fcd43201bfa4c4e0320c8517b611d7daa0e41ba33a0cb1fab80 1.37kB / 1.37kB     0.0s
 => => sha256:00be2573e9f79754b17954ba7a310a5f70c25b8f5bb78375e27e9e86d874877e 6.13kB / 6.13kB     0.0s
 => => sha256:48ecbb6b270eb481cb6df2a5b0332de294ec729e1968e92d725f1329637ce01b 2.11MB / 2.11MB     5.3s
 => => sha256:692f29ee68fa6bab04aa6a1c6d8db0ad44e287e5ff5c7e1d5794c3aabc55884 308.48kB / 308.48kB  2.0s
 => => sha256:6439819450d10d1aae92561f3ffff722137aada46d509644e8de4ca82bb26b07 25.90MB / 25.90MB  45.0s
 => => sha256:3c7be240f7bfb19ec575d8547832a9f20b95eec9b4cc94fe717dd047ad661159 230B / 230B         2.7s
 => => sha256:ca4b349df8ed83a59776df8f3868ece2783aa1ee2e9f052c9c9f3b54ae51a593 1.81MB / 1.81MB     6.3s
 => => extracting sha256:48ecbb6b270eb481cb6df2a5b0332de294ec729e1968e92d725f1329637ce01b          0.3s
 => => extracting sha256:692f29ee68fa6bab04aa6a1c6d8db0ad44e287e5ff5c7e1d5794c3aabc55884d          0.2s
 => => extracting sha256:6439819450d10d1aae92561f3ffff722137aada46d509644e8de4ca82bb26b07          1.4s
 => => extracting sha256:3c7be240f7bfb19ec575d8547832a9f20b95eec9b4cc94fe717dd047ad661159          0.0s
 => => extracting sha256:ca4b349df8ed83a59776df8f3868ece2783aa1ee2e9f052c9c9f3b54ae51a593          0.2s
 => [2/4] COPY . /app                                                                              0.2s
 => [3/4] WORKDIR /app                                                                             0.0s
 => [4/4] RUN pip install -r requirements.txt                                                      8.0s
 => exporting to image                                                                             0.2s
 => => exporting layers                                                                            0.2s
 => => writing image sha256:4f6ecb81f19c4743e41cf4461b31de8c7dafc532a0653fdc61211003b709fd65       0.0s
 => => naming to docker.io/library/flask-docker-demo-app                                           0.0s

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

Run docker image



```
D:\assigment 4>docker run --name flask-docker-demo-app -p 5001:5001 flask-docker-demo-app
 * Serving Flask app 'demo'
 * 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://127.0.0.1:5001
 * Running on http://172.17.0.2:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 133-108-616
```

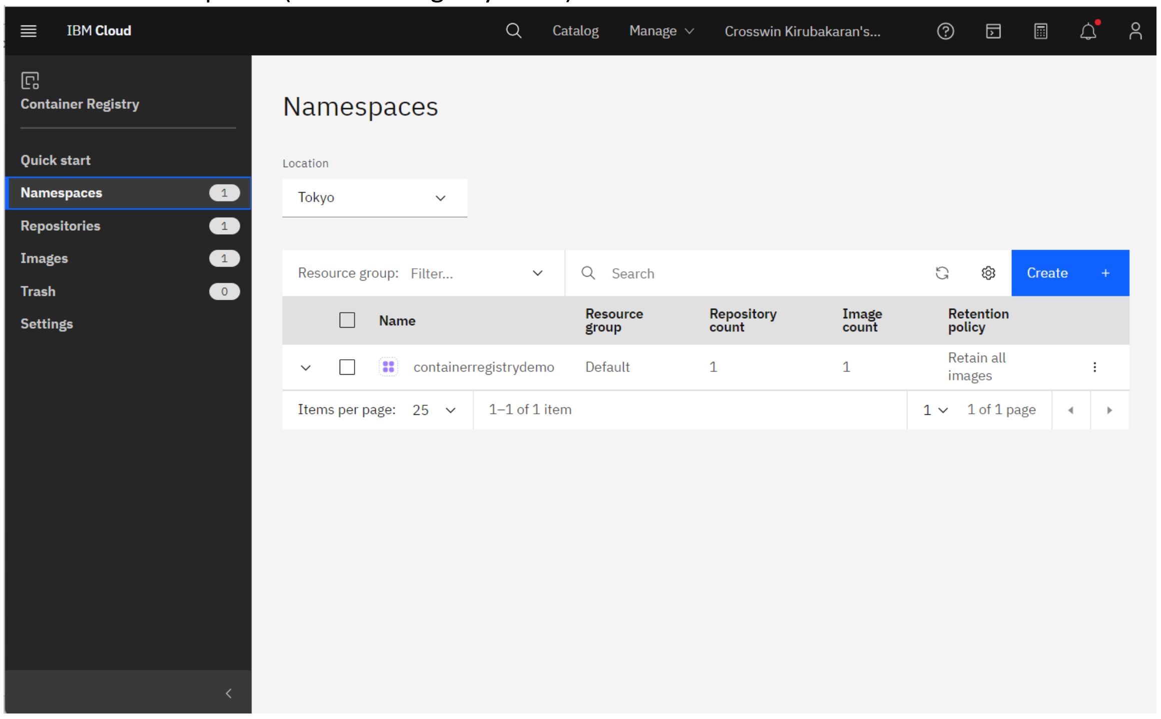
Container output



welcome to the flask

3.Create IBM container registry and push flask app

Create an namespace –(containerregistrydemo)



IBM Cloud | Catalog | Manage | Crosswin Kirubakaran's...

Container Registry

Quick start
Namespaces 1
Repositories 1
Images 1
Trash 0
Settings

Namespaces

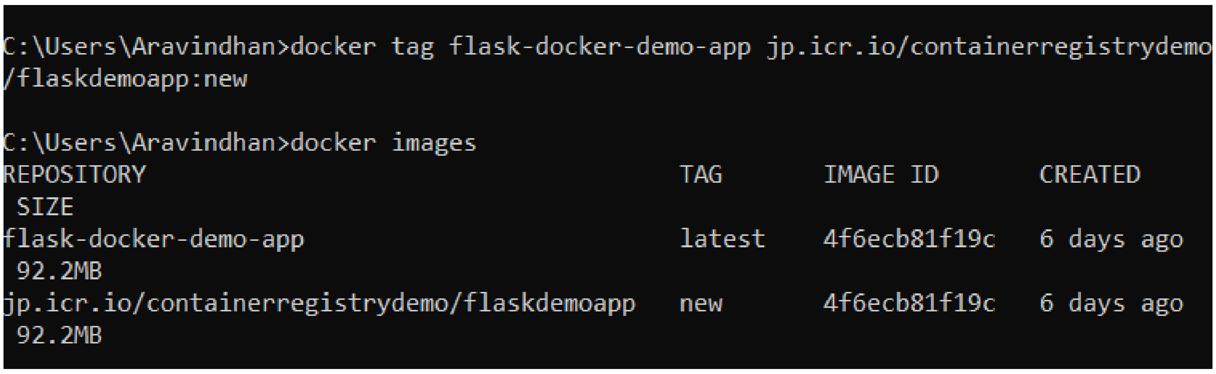
Location
Tokyo

Resource group: Filter... | Search | Create +

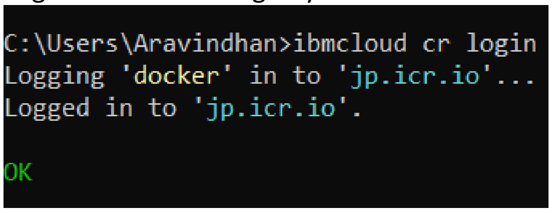
Name	Resource group	Repository count	Image count	Retention policy
containerregistrydemo	Default	1	1	Retain all images

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

Change docker image name


```
C:\Users\Aravindhan>docker tag flask-docker-demo-app jp.icr.io/containerregistrydemo
/flaskdemoapp:new

C:\Users\Aravindhan>docker images
REPOSITORY                                    TAG       IMAGE ID       CREATED
 SIZE
flask-docker-demo-app                         latest    4f6ecb81f19c   6 days ago
 92.2MB
jp.icr.io/containerregistrydemo/flaskdemoapp   new       4f6ecb81f19c   6 days ago
 92.2MB
```

Login to container registry


```
C:\Users\Aravindhan>ibmcloud cr login
Logging 'docker' in to 'jp.icr.io'...
Logged in to 'jp.icr.io'.

OK
```

Push the image to container registry

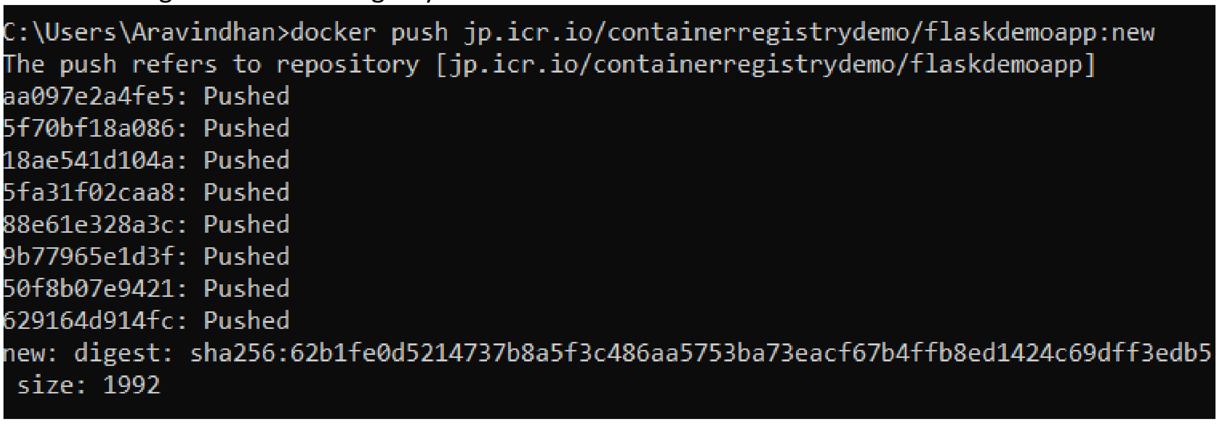
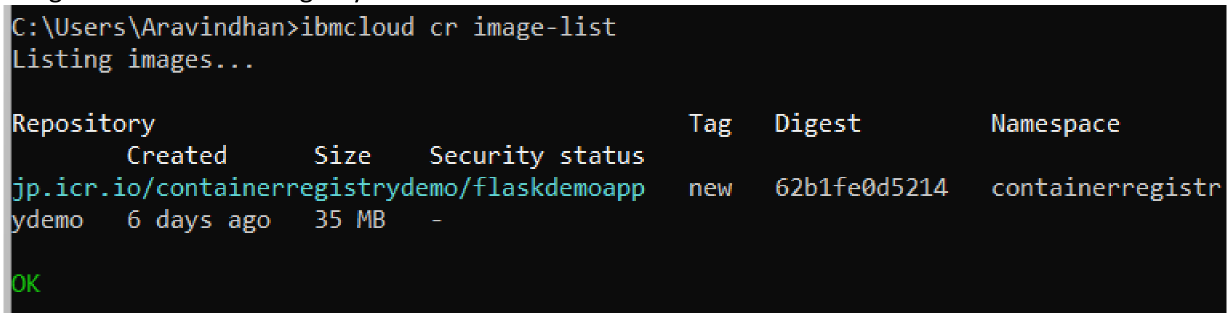

```
C:\Users\Aravindhan>docker push jp.icr.io/containerregistrydemo/flaskdemoapp:new
The push refers to repository [jp.icr.io/containerregistrydemo/flaskdemoapp]
aa097e2a4fe5: Pushed
5f70bf18a086: Pushed
18ae541d104a: Pushed
5fa31f02caa8: Pushed
88e61e328a3c: Pushed
9b77965e1d3f: Pushed
50f8b07e9421: Pushed
629164d914fc: Pushed
new: digest: sha256:62b1fe0d5214737b8a5f3c486aa5753ba73eacf67b4ffb8ed1424c69dff3edb5
 size: 1992
```

Image list in container registry


```
C:\Users\Aravindhan>ibmcloud cr image-list
Listing images...

Repository                                    Tag   Digest         Namespace
        Created       Size    Security status
jp.icr.io/containerregistrydemo/flaskdemoapp   new   62b1fe0d5214   containerregistr
ydemo   6 days ago    35 MB   -

OK
```

4.Deploy the flask app in IBM Kubernetes cluster and expose it in nodeport

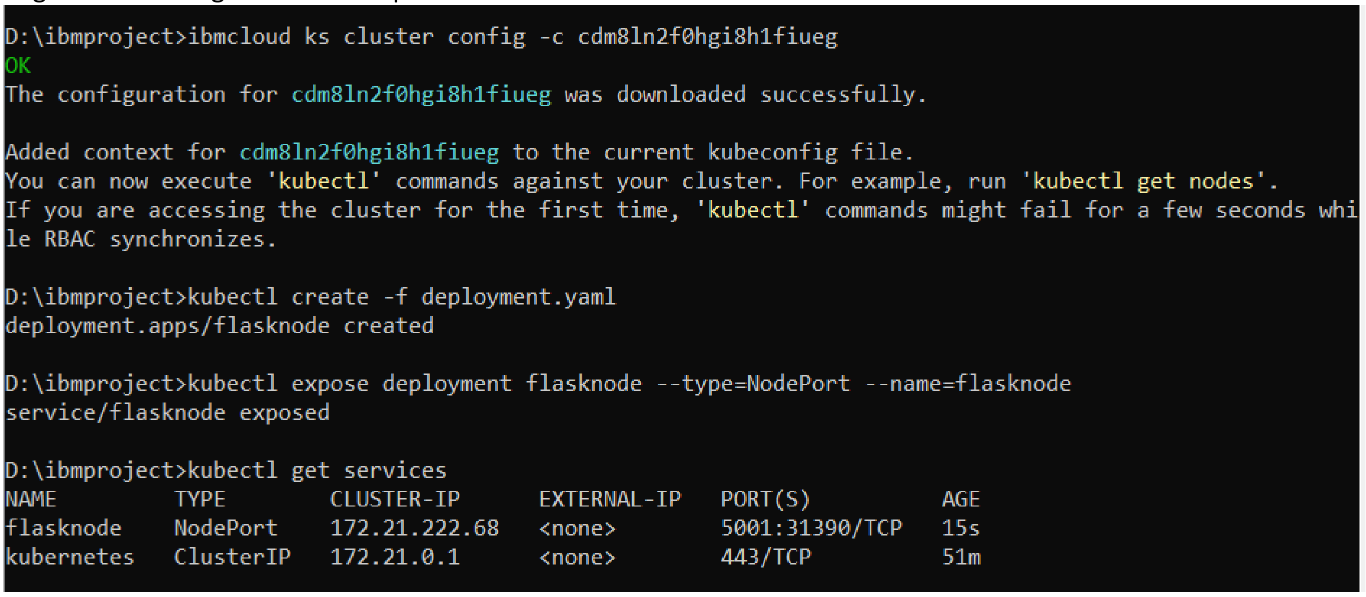
Deployment.yaml file

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flasknode
spec:
  replicas: 2
  selector:
    matchLabels:
      app: flasknode
  template:
    metadata:
      labels:
        app: flasknode
    spec:
      containers:
      - name: flasknode
        image: jp.icr.io/containerregistrydemo/flaskdemoapp:new
        imagePullPolicy: Always
        ports:
        - containerPort: 5001
```

Service.yaml file

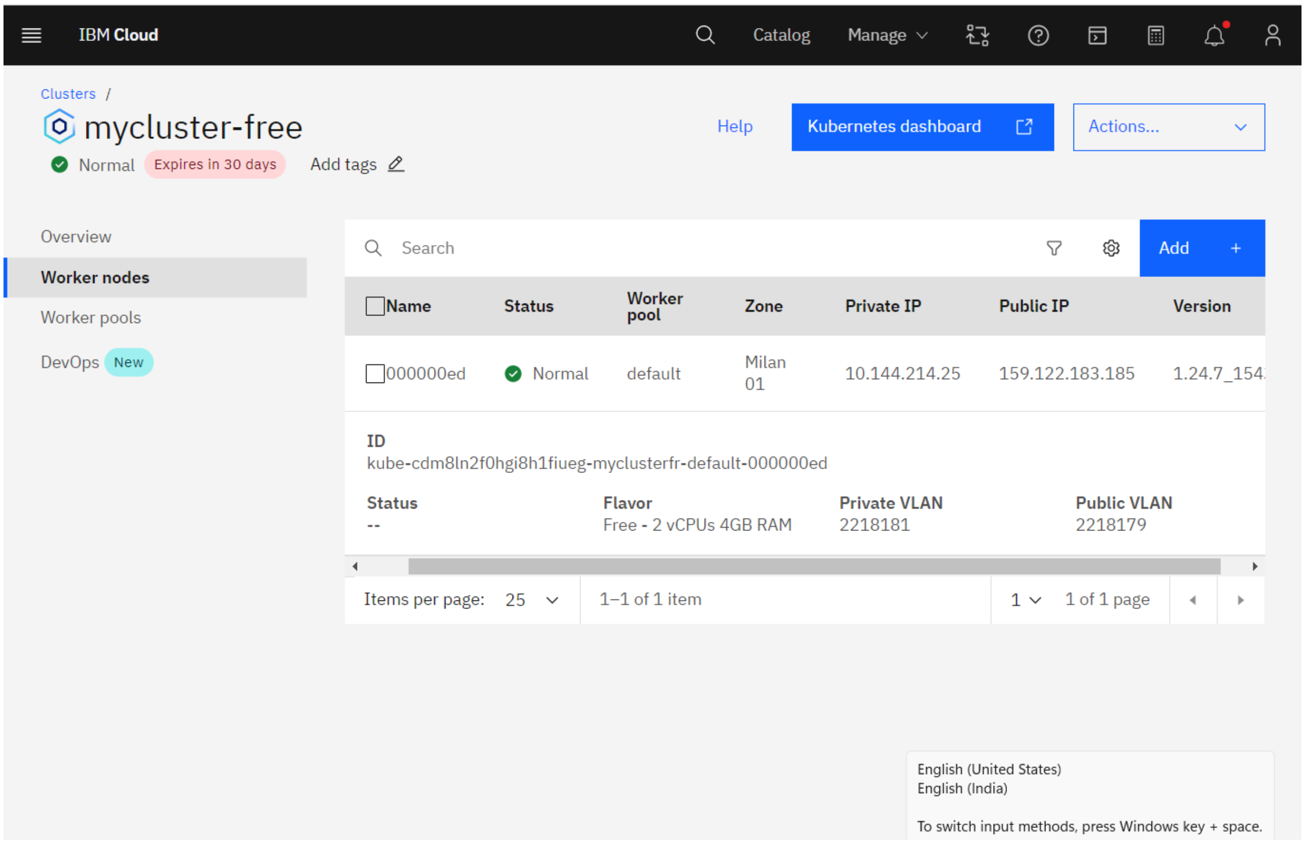
```
apiVersion: v1
kind: Service
metadata:
  name: flasknode
spec:
  ports:
  - port: 5001
    targetPort: 5001
  selector:
    app: flasknode
    protocol: TCP
```

1.connect to IBM ks cluster
2 create deployment
3. create a service
4. get services to get the service port



```
D:\ibmproject>ibmcloud ks cluster config -c cdm8ln2f0hgi8h1fiueg
OK
The configuration for cdm8ln2f0hgi8h1fiueg was downloaded successfully.

Added context for cdm8ln2f0hgi8h1fiueg 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 whi
le RBAC synchronizes.

D:\ibmproject>kubectl create -f deployment.yaml
deployment.apps/flasknode created

D:\ibmproject>kubectl expose deployment flasknode --type=NodePort --name=flasknode
service/flasknode exposed

D:\ibmproject>kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
flasknode    NodePort    172.21.222.68   <none>        5001:31390/TCP   15s
kubernetes   ClusterIP   172.21.0.1      <none>        443/TCP          51m
```

Public ip is available in Kubernetes cluster dashboard



mycluster-free

Normal Expires in 30 days Add tags

Name	Status	Worker pool	Zone	Private IP	Public IP	Version
000000ed	Normal	default	Milan 01	10.144.214.25	159.122.183.185	1.24.7_154

ID
kube-cdm8ln2f0hgi8h1fiueg-myclusterfr-default-000000ed

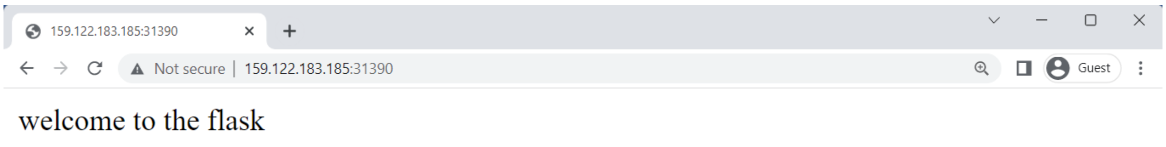
Status	Flavor	Private VLAN	Public VLAN
--	Free - 2 vCPUs 4GB RAM	2218181	2218179

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

English (United States)
English (India)

To switch input methods, press Windows key + space.

Output



welcome to the flask