

| | |
|--------------|------------------------------------|
| Date | 22 October 2022 |
| Team ID | PNT2022TMID50744 |
| Project Name | Project - Personal Expense Tracker |

Assignment - 4

Kubernetes and Docker

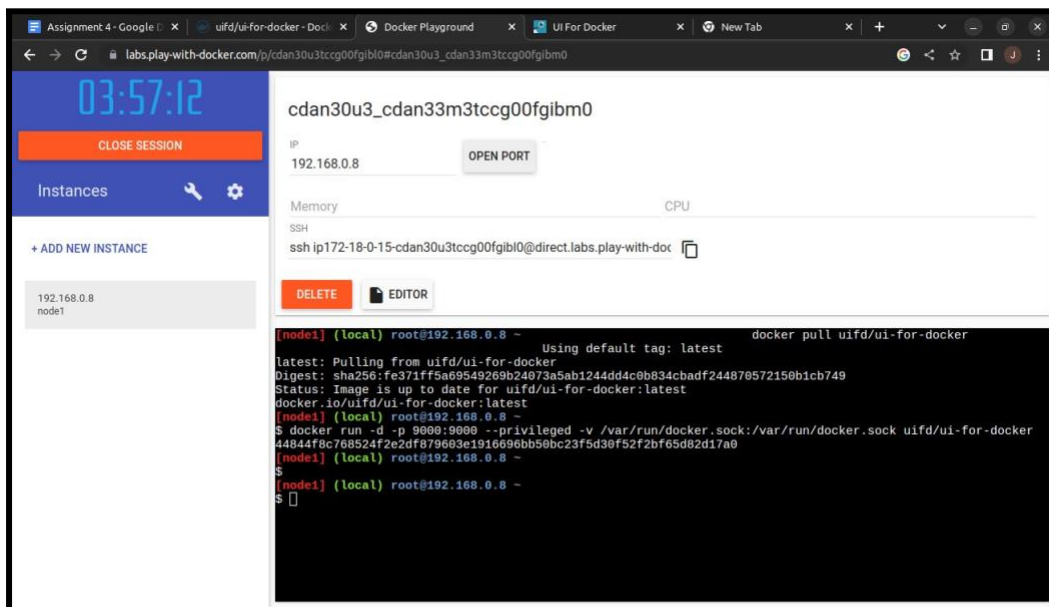
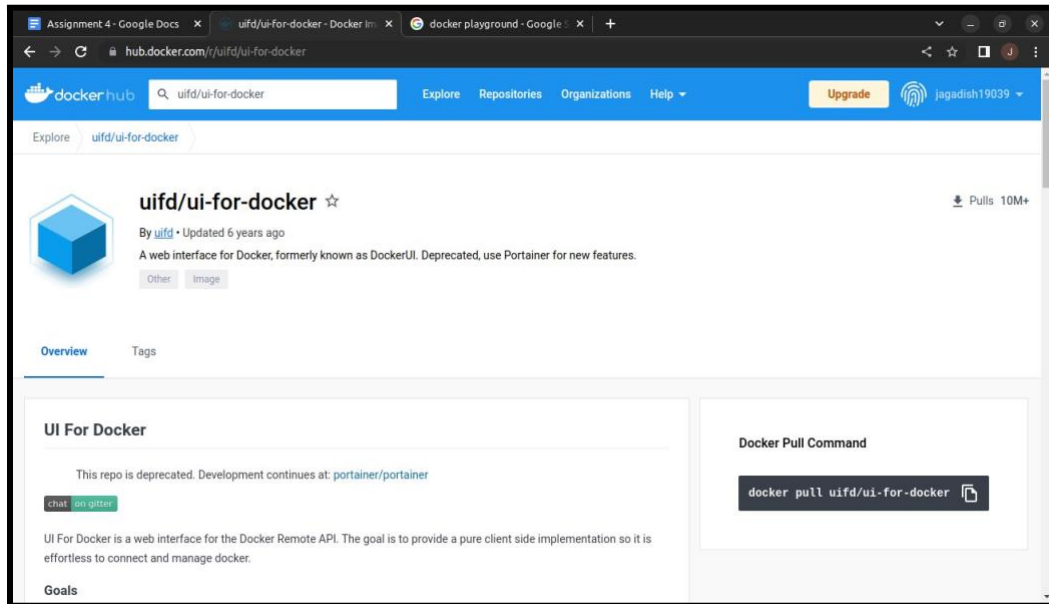
Question

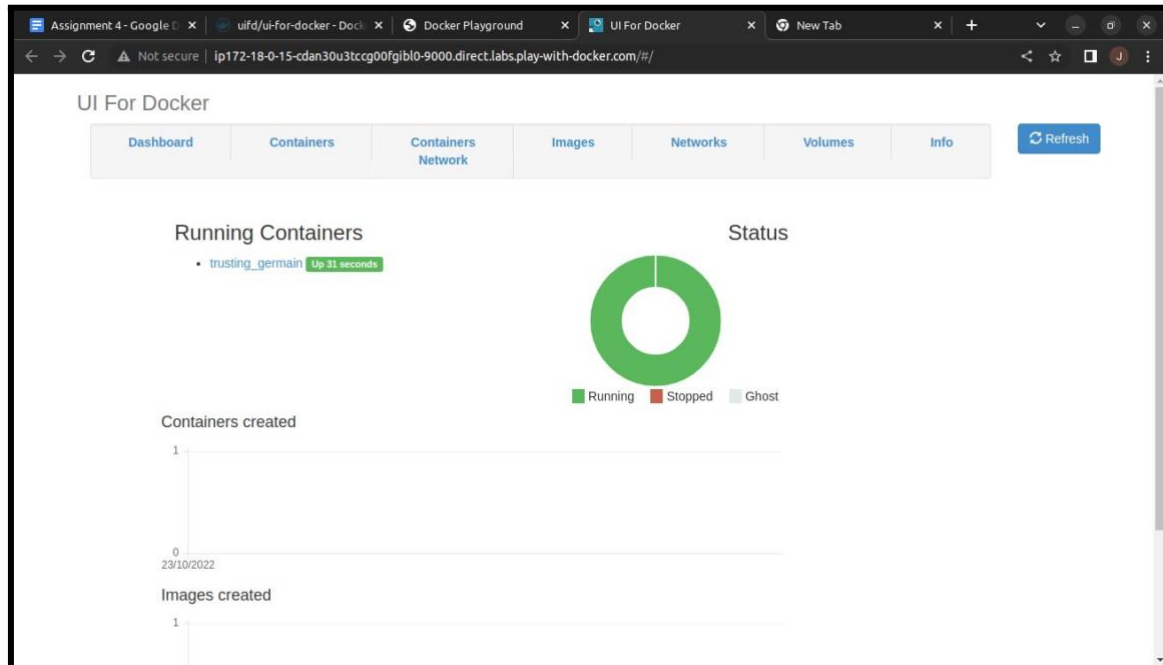
1. Pull an Image from docker hub and run it in Docker Playground
2. Create a docker file for the jobportal application and deploy it in Docker desktop application
3. Create a IBM container registry and deploy helloworld app or jobportal app
4. Create a Kubernetes cluster in IBM cloud and deploy helloworld image or jobportal image and also expose the same app to run in nodeport

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





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

- Create a docker file for build and deploy flask app.
- Use `docker build -t image_name .` in the current directory to start building the docker image and deploy in our local docker
- 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" ]
```

Steps Involved

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
root@jagz-mx:/home/jagadish/Documents/DockerLearning# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
root@jagz-mx:/home/jagadish/Documents/DockerLearning# docker build -t job-portal-app .
Sending build context to Docker daemon 65.02kB
Step 1/8 : FROM ubuntu/apache2
latest: Pulling from ubuntu/apache2
b572d2b36365: Pull complete
3a1193867518: Pull complete
8bfd5261bf9e: Pull complete
Digest: sha256:d9b8fe0cdbc6964360a1b4037d521ce326c287679bd1da6cd909997dc2509302
Status: Downloaded newer image for ubuntu/apache2:latest
----> 6ca4f2c95e83
Step 2/8 : FROM python
latest: Pulling from library/python
f606d8928ed3: Pull complete
47db815c6a45: Pull complete
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Step 7/8 : ENTRYPOINT [ "python" ]
----> Running in d4e98c363815
Removing intermediate container d4e98c363815
----> 7a8fadbf86d7
Step 8/8 : CMD ["app.py"]
----> Running in 0380311c3e1d
Removing intermediate container 0380311c3e1d
----> 3b55665956a8
Successfully built 3b55665956a8
Successfully tagged job-portal-app:latest
root@jagz-mx:/home/jagadish/Documents/DockerLearning# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
job-portal-app latest 3b55665956a8 About a minute ago 1.13GB
ubuntu/apache2 latest 6ca4f2c95e83 3 days ago 208MB
python latest f05c8762fe15 9 days ago 921MB
root@jagz-mx:/home/jagadish/Documents/DockerLearning#
```

```
root@jagz-mx:/home/jagadish/Documents/DockerLearning# docker run -p 5000:5000 job-portal-app
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI serv
er 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: 481-117-531
```

Run locally using docker

← → ↺ ⓘ localhost:5000/application

Job Application Page

Applicant Name

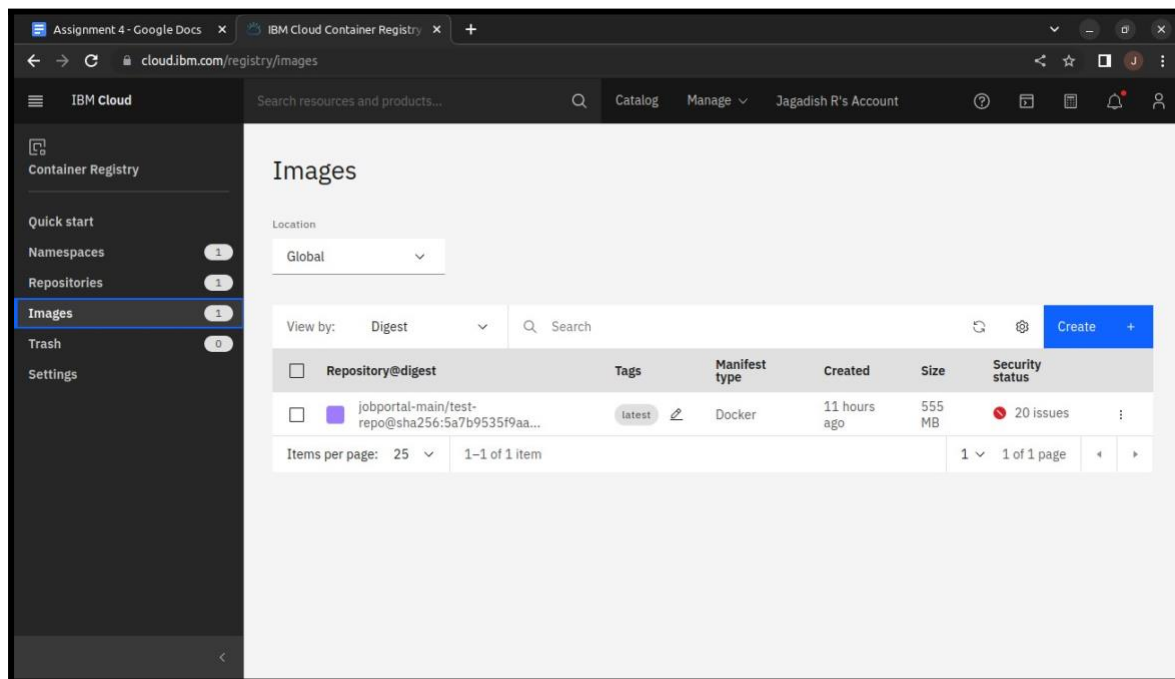
Name

Email

Email

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

- Log into IBM cloud
- Create a **container registry**
- Using IBM Cloud CLI, install the **container registry plugin** in our system
- Push our docker image into the created container registry using **docker push**
- So, our job portal app is deployed in the IBM container registry



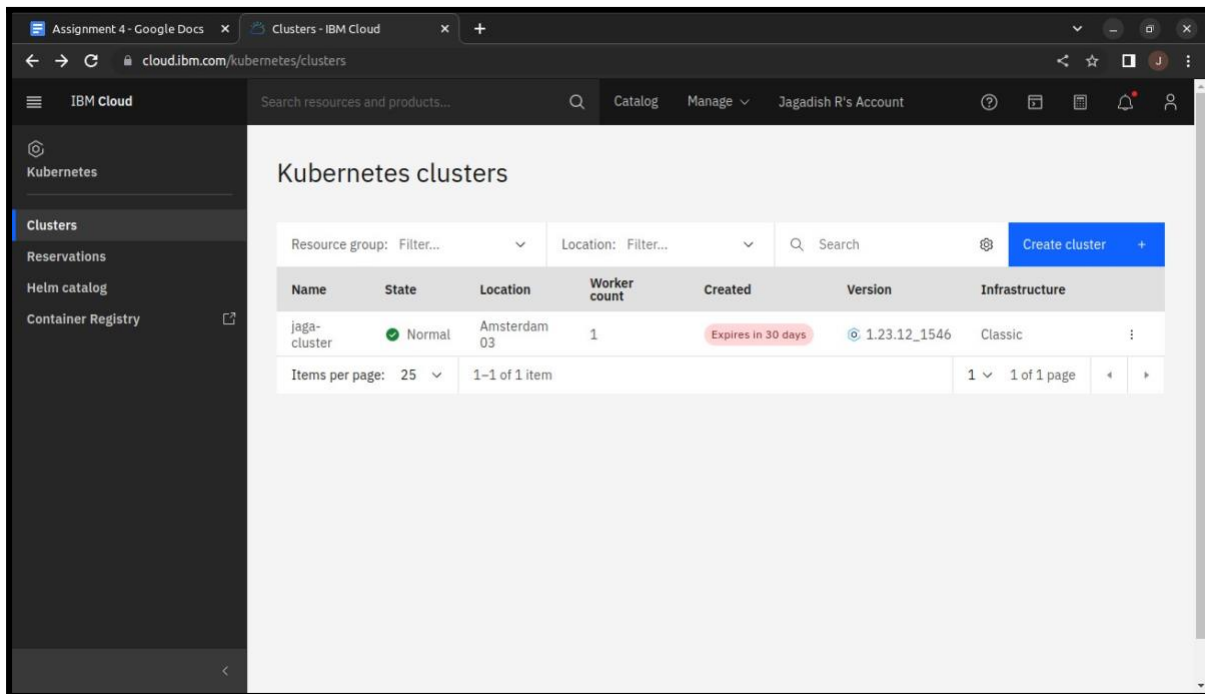
4. Create a Kubernetes cluster in IBM cloud and deploy helloworld image or jobportal image and also expose the same app to run in nodeport

- Log into IBM cloud
- Create a **kubernete**
- Using IBM Cloud CLI, install the **ks plugin** in our system
- Create a **cluster** in the kubernetes
- Now, go to the **kubernetes dashboard** where we need to create a service based on a yaml file (given below)
- In that file, we have to mention *which image we are going to use* and the *app name*
- Take the **public IP address** and **Nodeport** since we exposed the *flask app in nodeport*
- 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"
```

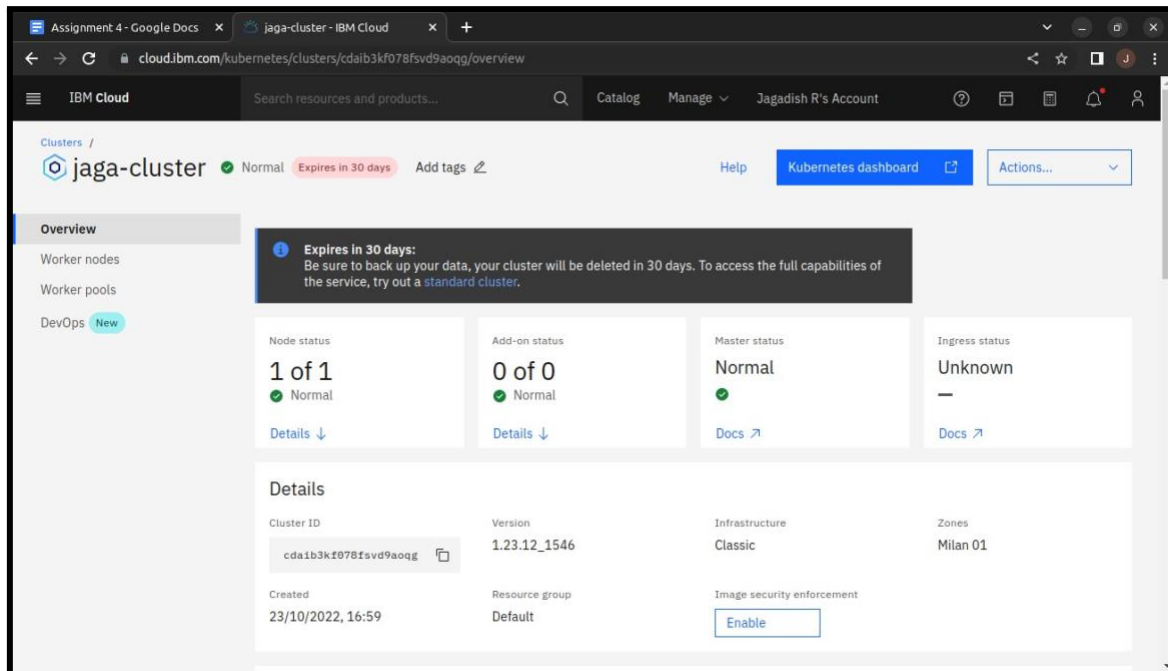
Cluster creation



The screenshot shows the IBM Cloud console interface for managing Kubernetes clusters. The left sidebar contains navigation links for Kubernetes, Clusters, Reservations, Helm catalog, and Container Registry. The main content area is titled "Kubernetes clusters" and features a table with the following columns: Name, State, Location, Worker count, Created, Version, and Infrastructure. A single cluster, "jaga-cluster", is listed with a "Normal" state, located in "Amsterdam 03", with 1 worker. A red banner indicates it "Expires in 30 days". The version is "1.23.12_1546" and the infrastructure is "Classic". A "Create cluster" button is visible in the top right of the table area.

| Name | State | Location | Worker count | Created | Version | Infrastructure |
|--------------|--------|--------------|--------------|--------------------|--------------|----------------|
| jaga-cluster | Normal | Amsterdam 03 | 1 | Expires in 30 days | 1.23.12_1546 | Classic |

Configuring the cluster



The screenshot displays the "Overview" page for the "jaga-cluster". At the top, there's a header with the cluster name, state ("Normal"), and expiration notice ("Expires in 30 days"). Below this, a "Details" section provides key information: Cluster ID (cda1b3kf078fsvd9aoqg), Version (1.23.12_1546), Infrastructure (Classic), and Zones (Milan 01). It also shows the creation time (23/10/2022, 16:59) and Resource group (Default). A button to "Enable" Image security enforcement is present. On the left, a sidebar lists navigation options: Overview, Worker nodes, Worker pools, and DevOps (marked as "New").

Expires in 30 days:
Be sure to back up your data, your cluster will be deleted in 30 days. To access the full capabilities of the service, try out a [standard cluster](#).

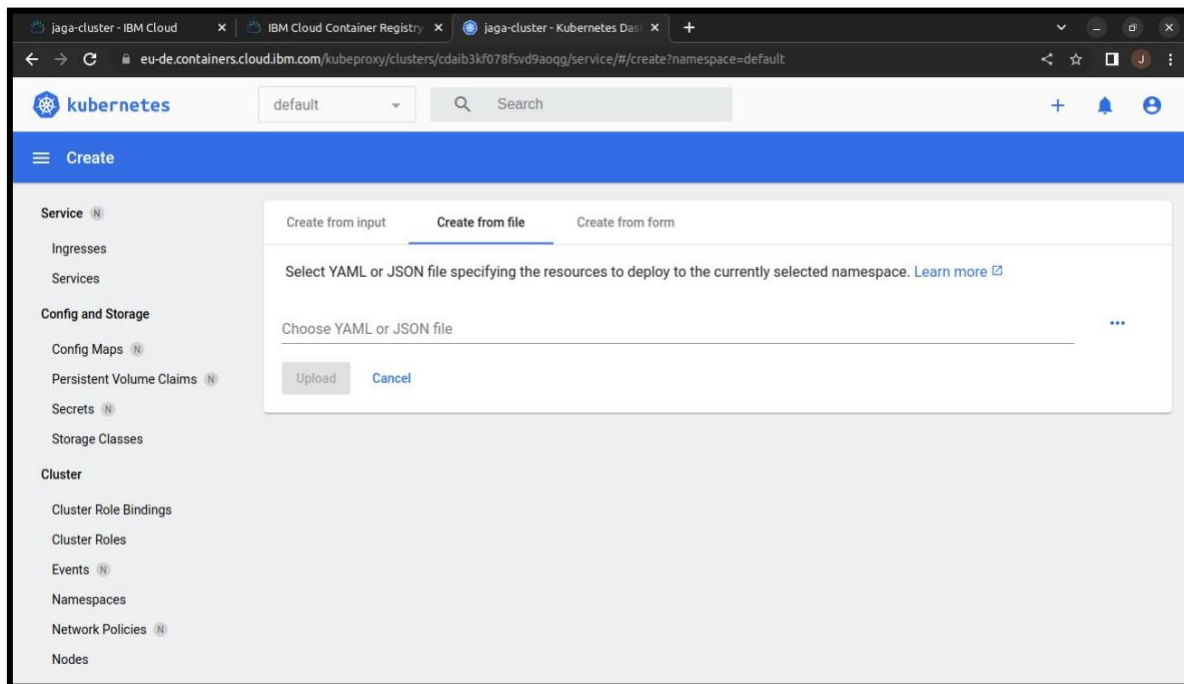
| Node status | Add-on status | Master status | Ingress status |
|------------------|------------------|---------------|----------------|
| 1 of 1 Normal | 0 of 0 Normal | Normal | Unknown |

Details

| Cluster ID | Version | Infrastructure | Zones |
|----------------------|--------------|----------------|----------|
| cda1b3kf078fsvd9aoqg | 1.23.12_1546 | Classic | Milan 01 |

Created: 23/10/2022, 16:59 | Resource group: Default | Image security enforcement: [Enable](#)

Creating a service based on the yml file



Procedure to find the exposed url

```
root@ravi-HP-Slim-Desktop-290-p0xxx: ~  
/ # ibmcloud ks cluster config --cluster cdaib3kf078fsvd9aoqg  
OK  
The configuration for cdaib3kf078fsvd9aoqg was downloaded successfully.  
  
Added context for cdaib3kf078fsvd9aoqg 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.  
/ # kubectl config current-context  
jaga-cluster/cdaib3kf078fsvd9aoqg  
/ # kubectl get nodes  
NAME                STATUS    ROLES    AGE    VERSION  
10.144.183.60        Ready    <none>    6h7m   v1.23.12+IKS  
/ # kubectl get pods  
NAME                READY    STATUS    RESTARTS   AGE  
job-portal-app-57f769b8b6-4rggl  1/1      Running    0           5h51m  
/ # ibmcloud cs workers --cluster cdaib3kf078fsvd9aoqg  
OK  
ID  
kube-cdaib3kf078fsvd9aoqg-jagacuster-default-00000073  Public IP    Private IP    Flavor    State    Status    Zone    Version  
159.122.179.243    10.144.183.60    free    normal    Ready    mil01    1.23.12_1548  
/ # kubectl describe service job-portal-app  
Name: job-portal-app  
Namespace: default  
Labels: <none>  
Annotations: <none>  
Selector: app=job-portal-app  
Type: NodePort  
IP Family Policy: SingleStack  
IP Families: IPv4  
IP: 172.21.183.254  
IPs: 172.21.183.254  
Port: <unset> 5000/TCP  
TargetPort: 5000/TCP  
NodePort: <unset> 30508/TCP  
Endpoints: 172.30.146.139:5000  
Session Affinity: None  
External Traffic Policy: Cluster  
Events: <none>  
/ #
```


Run our flask app in the IBM kubernetes

