

Name	Sreenidhi S
Roll No	SSNCE195001310
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

Activities Firefox Web Browser Nov 11 21:42

IBM uifd: X IBM-P: Assignme Downl Google Assign Logins what What Docker Pl How to Proble UI For + v

← → ↻ 🔒 https://hub.docker.com/r/uifd/ui-for-docker

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

dockerhub 🔍 uifd/ui-for-docker Explore Repositories Organizations Help Upgrade naveenmr13

Explore uifd/ui-for-docker

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.

Image

Overview Tags

UI For Docker

This repo is deprecated. Development continues at: [portainer/portainer](#)

[chat on gitter](#)

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
```

Activities Firefox Web Browser Nov 11 21:42

IBM uifd/ui IBM-P: Assignme Downl Google Assign Logins what What Docker X How to Proble UI For + v

← → ↻ 🔒 https://labs.play-with-docker.com/p/cdn601n91rrg008ljhbg#cdn601n9_cdn603n91rrg008ljhc0

03:24:13

CLOSE SESSION

Instances

+ ADD NEW INSTANCE

192.168.0.28
node1

cdn601n9_cdn603n91rrg008ljhc0

IP: 192.168.0.28 OPEN PORT 9000

Memory: 1.62% (64.82MiB / 3.906GiB) CPU: 0.26%

SSH: [18-0-22-cdn601n91rrg008ljhbg@direct.labs.play-with-docker.com](#)

DELETE EDITOR

```
90092d4aef6d6da48fc963e7b46856490b40a62435c1d1e9351934533cf193f52
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 assign 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
43843cbeb552082e6438dad2465e64ae40975131d0bbe818375ed5df9fedbdce
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 assign 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
90eeal301752ae6bafa252d835b5a63e1b56a65215e84e4c2b9987160974ea10
[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

IBM uifd/ui IBM-P: Assignme Downl Googl Assign Logins what What Docker Pl How L Proble UI Fr X + v a x

ip172-18-0-22-cdn6o1n91rrg008ljbhg-9000.direct.labs.play-with-docker.com/#/

UI For Docker

Dashboard

Containers

Containers Network

Images

Networks

Volumes

Info

Refresh

Running Containers

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

Status

Running Stopped Ghost

Containers created

11/11/2022

Images created

8/8/2016

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 -
```

The screenshot shows a web browser window with the URL `127.0.0.1:5000/register`. The page is titled "Register Page" and contains a registration form with the following fields:

- Email
- Username
- Rollnumber
- Password

Below the form is a "Register" button and a link: "Already have an account? Log In".

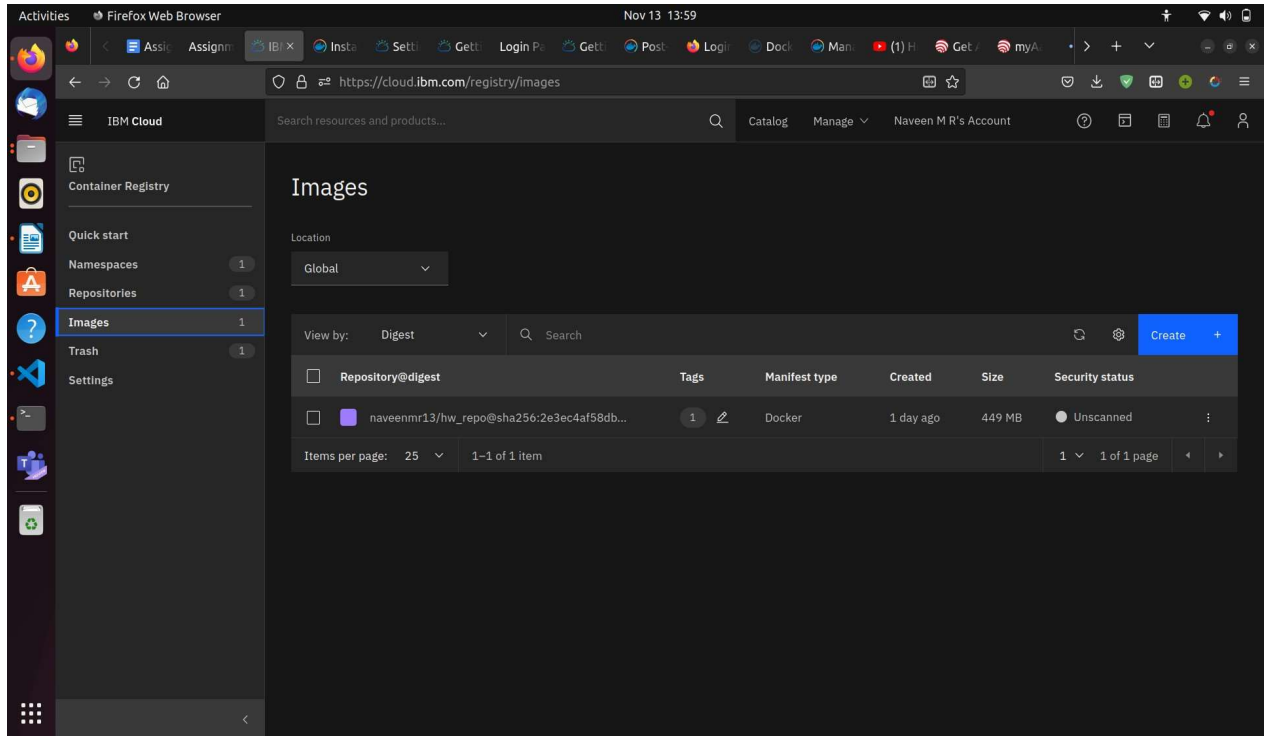
In the background, a terminal window shows the Docker build and run process. The build process includes the following steps:

- Step 1/8 : FROM ubuntu/apache2
- Step 2/8 : FROM python
- Step 3/8 : COPY ./requirements.txt /FlaskApp/requirements.txt
- Step 4/8 : WORKDIR /FlaskApp
- Step 5/8 : RUN pip install -r requirements.txt

The terminal output shows the installation of various dependencies, including Flask, Flask_Login, WTForms, Flask_WTF, IBM_DB, Werkzeug, and Jinja2.

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 job portal app is deployed in the IBM container registry



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

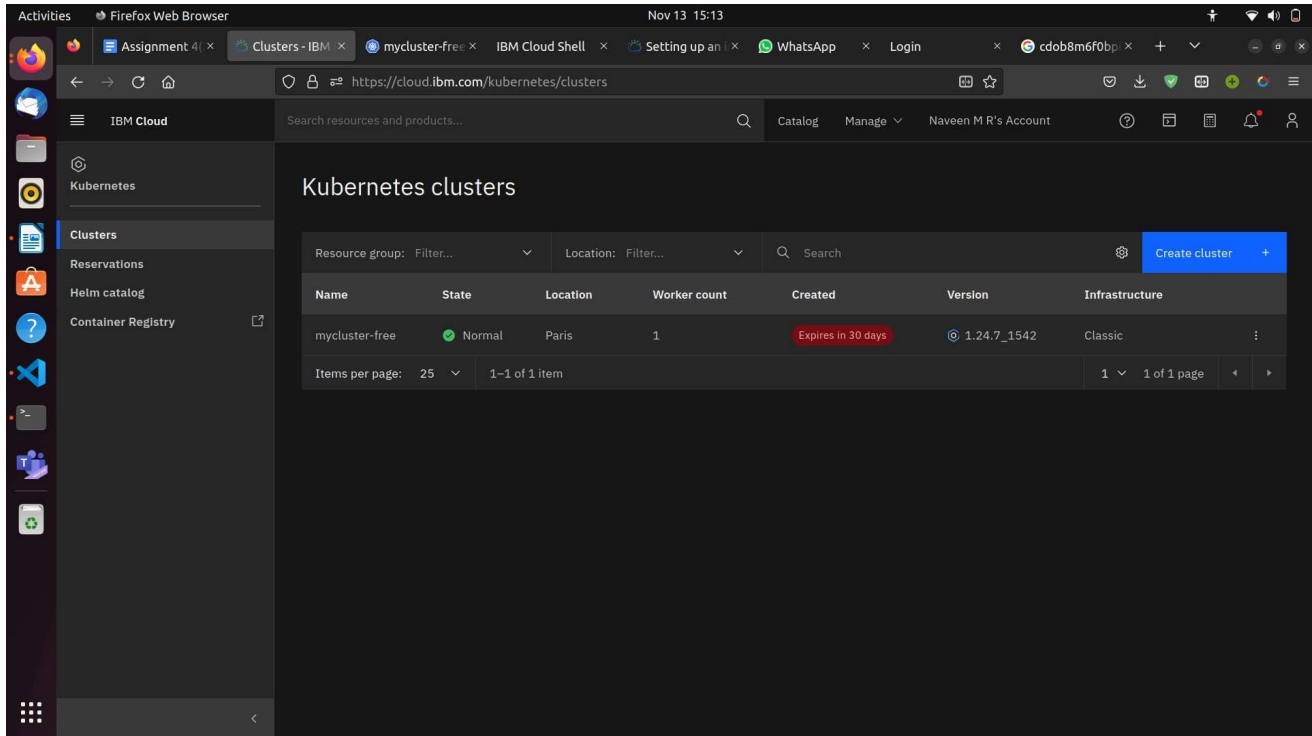
- a. Log into IBM cloud
- b. Create a **kubernete**
- c. Using IBM Cloud CLI, install the **ks plugin** in our system
- d. Create a **cluster** in the kubernetes
- e. Now, go to the **kubernetes dashboard** where we need to create a service based on a yml file (given below)
- f. In that file, we have 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"
```

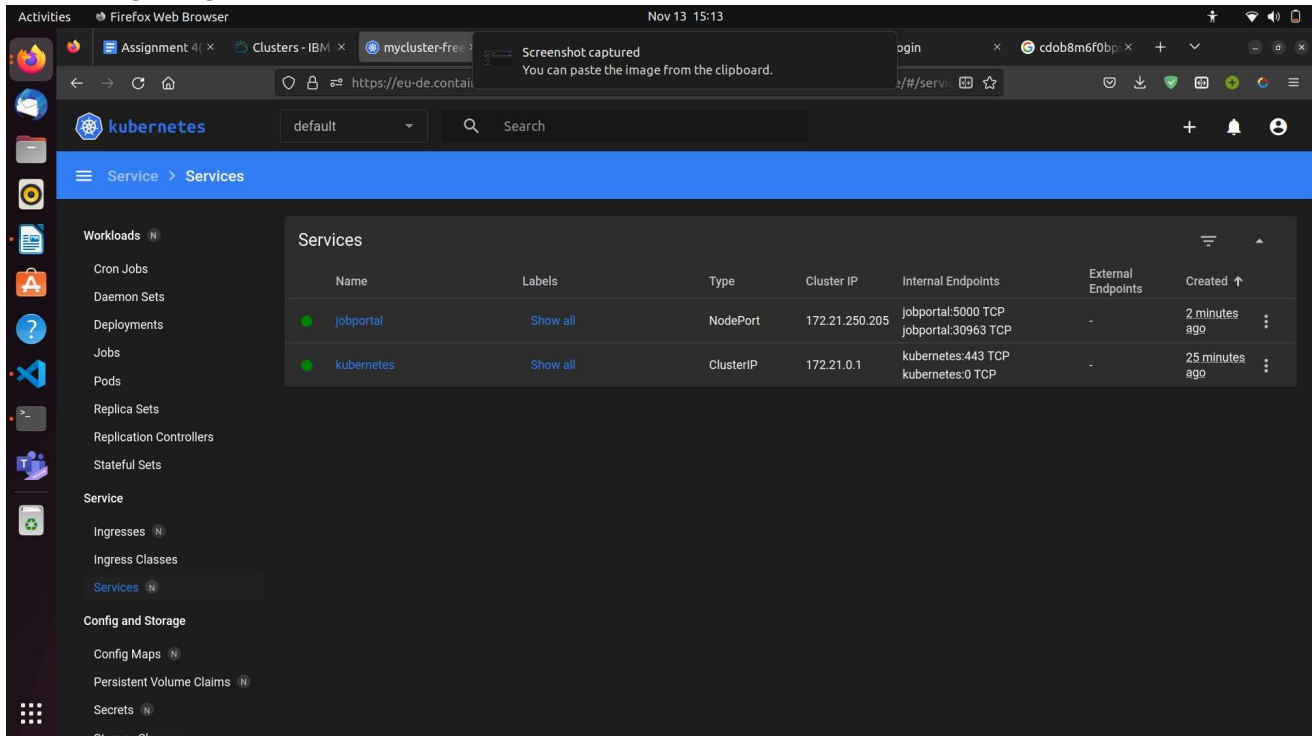

Cluster creation



The screenshot shows the IBM Cloud console interface in a Firefox web browser. The browser's address bar displays the URL `https://cloud.ibm.com/kubernetes/clusters`. The left-hand navigation menu includes options for IBM Cloud, Kubernetes, Clusters, Reservations, Helm catalog, and Container Registry. The main content area is titled "Kubernetes clusters" and features a search bar and a "Create cluster" button. Below this, a table lists the existing clusters. The table has columns for Name, State, Location, Worker count, Created, Version, and Infrastructure. One cluster, "mycluster-free", is listed with a state of "Normal", located in "Paris", with a worker count of 1. The "Created" column for this cluster shows a red warning icon and the text "Expires in 30 days". The "Version" is "1.24.7_1542" and the "Infrastructure" is "Classic". At the bottom of the table, there are pagination controls showing "Items per page: 25" and "1-1 of 1 item".

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

Configuring the cluster



The screenshot displays the Kubernetes dashboard within a Firefox web browser. The browser's address bar shows the URL `https://eu-de.containerservice.cloud.ibm.com/k8s/clusters/mycluster-free`. The dashboard's left-hand navigation menu includes sections for Workloads (Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets), Service (Ingresses, Ingress Classes, Services), and Config and Storage (Config Maps, Persistent Volume Claims, Secrets, Storage Classes). The main content area is titled "Services" and contains a table with columns for Name, Labels, Type, Cluster IP, Internal Endpoints, External Endpoints, and Created. Two services are listed: "jobportal" and "kubernetes". The "jobportal" service has a type of "NodePort", a cluster IP of "172.21.250.205", and internal endpoints for "jobportal:5000 TCP" and "jobportal:30963 TCP". The "kubernetes" service has a type of "ClusterIP", a cluster IP of "172.21.0.1", and internal endpoints for "kubernetes:443 TCP" and "kubernetes:0 TCP". Both services have a "Created" time of "2 minutes ago".

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

Activities Firefox Web Browser Nov 13 15:13

Assignment 4 Clusters - IBM mycluster-free IBM Cloud Shell Setting up an WhatsApp Login cdob8m6f0bp

169.51.207.205:30963/update

Update Password

Username

Username

Oldpassword

Previous Password

Password

Password

Update

