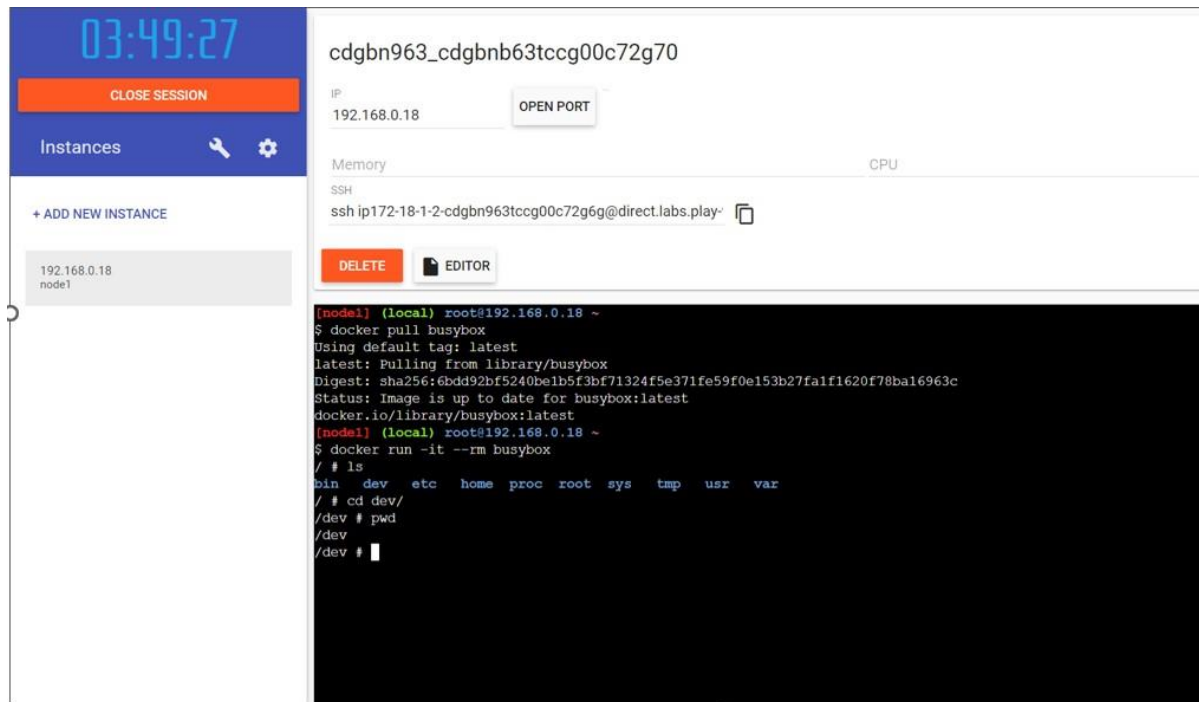


## ASSIGNMENT 4

PROJECT TITLE	SKILL AND JOB RECOMENDER
STUDENT NAME	PRAIMATHI.K
STUDENT ROLLNO	813819104064
MARKS	2 MARKS

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



The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:49:27, a 'CLOSE SESSION' button, and an 'Instances' section. Below 'Instances', there's a '+ ADD NEW INSTANCE' button and a list of instances showing '192.168.0.18' and 'node1'. The main area displays the session details for 'cdgbn963\_cdgbnb63tccg00c72g70'. It shows the IP '192.168.0.18' and an 'OPEN PORT' button. Below this, there's a 'Memory' and 'CPU' section, an 'SSH' section with a terminal window, and 'DELETE' and 'EDITOR' buttons. The terminal window shows the following commands and output:

```
(node1) (local) root@192.168.0.18 ~
$ docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:6bdd92bf5240be1b5f3bf71324f5e371fe59f0e153b27fa1f1620f78ba16963c
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest
(node1) (local) root@192.168.0.18 ~
$ docker run -it --rm busybox
/ # ls
bin dev etc home proc root sys tmp usr var
/ # cd dev/
/dev # pwd
/dev
/dev #
```

### 2. Create a docker file for the job portal application and deploy it in Docker desktop application.

```
FROM python:3-alpine3.15
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
EXPOSE 5000
CMD python ./app.py
```

C:\Windows\System32\cmd.exe - docker push icr.io/ibm-sahana/hello-flask:0.0.1.RELEASE

2 Dir(s) 431,001,137 bytes free

C:\Users\moham\Downloads\hello-flask-main\hello-flask-main>docker build -t sahanaparveen/hello-flask:0.0.1.RELEASE

"docker build" requires exactly 1 argument.

See 'docker build --help'.

Usage: docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile

C:\Users\moham\Downloads\hello-flask-main\hello-flask-main>docker build -t sahanaparveen/hello-flask:0.0.1.RELEASE

[+] Building 243.3s (10/10) FINISHED

>> [internal] load build definition from Dockerfile

>> transferring dockerfile: 157B

>> [internal] load .dockerignore

>> transferring context: 2B

>> [internal] load metadata for docker.io/library/python:alpine3.13

>> [auth] library/python:alpine3.13 token for registry-1.docker.io

>> [internal] load build context

>> transferring context: 44.0B

>> [1/4] FROM docker.io/library/python:alpine3.13@sha256:d89ac8cf42213699c792f0bec4f15c029709340a688600fdae13630f03c3be

>> resolve docker.io/library/python:alpine3.13@sha256:d89ac8cf42213699c792f0bec4f15c029709340a688600fdae13630f03c3be

>> sha256:7dcbef5879c0d6d007d13d5963d780245d3163917475e8d18dacc9ae27c76 673.38kB / 673.10kB

>> sha256:280f0944f7005372e3ff2e488c0911d3d359c1c1213d000e0c978e2d26c 13.07kB / 13.07kB

>> sha256:d89ac8cf42213699c792f0bec4f15c029709340a688600fdae13630f03c3be 1.52kB / 1.52kB

>> sha256:08502255173a653ae961ad02612e70d0a1526697d1dc09ee0d0f12bc627 1.37kB / 1.37kB

>> sha256:c8b5671607576027087c10313b1886a7a754b747c219c8144791f5b6886ed 7.01kB / 7.01kB

>> sha256:3621f4f0d005277968ff3417774002224761634bda481c7a7132c7792a 2.80kB / 2.80kB

>> sha256:078e084d20cc90773760d44f1224363802bc06a08f798bc029c4d613c4 2.0kB / 2.0kB

>> sha256:8652ff8a9cc2a72105ba9f1108019cb3f4e4c4230bf4d6a47165c5161d15 3.00kB / 3.00kB

>> extracting sha256:3621f4f0d005277968ff3417774002224761634bda481c7a7132c7792a

>> extracting sha256:7dcbef5879c0d6d007d13d5963d780245d3163917475e8d18dacc9ae27c76

>> extracting sha256:280f0944f7005372e3ff2e488c0911d3d359c1c1213d000e0c978e2d26c

>> extracting sha256:08502255173a653ae961ad02612e70d0a1526697d1dc09ee0d0f12bc627

>> extracting sha256:8652ff8a9cc2a72105ba9f1108019cb3f4e4c4230bf4d6a47165c5161d15

>> [2/4] WORKDIR /app

>> [4/4] RUN pip install -r requirements.txt

>> exporting to image

>> exporting layers

>> writing image sha256:3ea0e3a73470080f05189b0211888f1801c2e255442cc97311ef22ac06

>> naming to docker.io/sahanaparveen/hello-flask:0.0.1.RELEASE

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

C:\Users\moham\Downloads\hello-flask-main\hello-flask-main>ibmcloud plugin install container-registry -r "IBM Cloud"

looking up 'container-registry' from repository 'IBM Cloud'...

Plug-in 'container-registry[cr] 1.0.2' found in repository 'IBM Cloud'

Attempting to download the binary file...

30°C Cloudy

15:25 04-11-2022

Docker Desktop

Update to latest

sahanaparveen

Containers

Images

Volumes

Dev Environ...

Extensions BETA

Add Extension

Add Extensions

Docker Desktop

Update to latest

Images on disk

Last refresh: Never 2 Images Refresh to see disk usage Clean up

Images Give feedback

LOCAL REMOTE REPOSITORIES

Search

In use only

NAME	TAG	IMAGE ID	CREATED	SIZE
icr.io/ibm-sahana/hello-fl...	hello-flask	3ea0e3a73470	41 minutes ago	70.72 MB
sahanaparveen/hello-flask	0.0.1.RELEASE	3ea0e3a73470	41 minutes ago	70.72 MB

RAM 1.66GB CPU 0.00% Connected to Hub

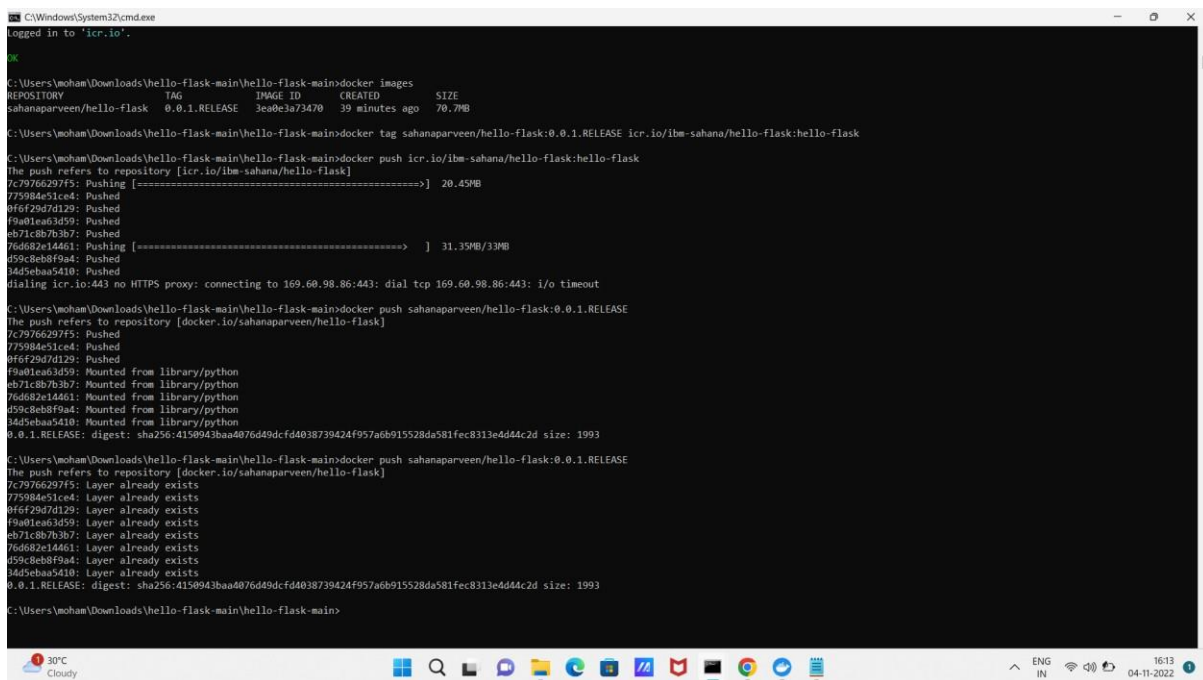
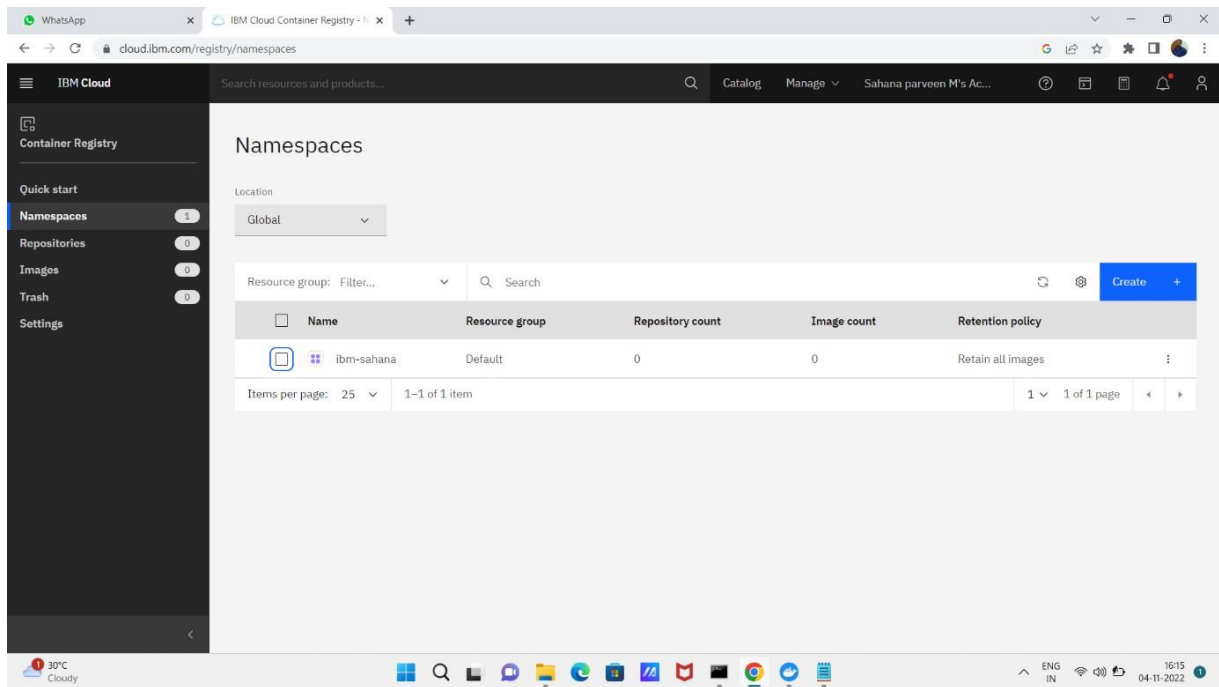
Run

v4.13.0

30°C Cloudy

15:29 04-11-2022

### 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 node port.

mycluster-free

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: 1 of 1 Normal

Add-on status: 0 of 0 Normal

Master status: Normal

Ingress status: Pending

Details

Cluster ID	Version	Infrastructure	Zones
cdj9p7lf0ndbf115jdn0	1.24.7_1542	Classic	Milan 01
Created	Resource group	Image security enforcement	
05/11/2022, 12:41	Default	Enable	

Node health

Worker node details

Help

- Log in to your cluster
- Deploy your app
  - 1. Set up your image registry
    - Store your images in a registry that the cluster can access to run containers.
  - 2. Deploy your app
    - Kubernetes dashboard
  - 3. Manage your app lifecycle
    - Keep your apps up to date with Kubernetes container orchestration tools.
- Expose your app
- Add storage to your app
- Connect integrations
- Install add-ons
- Troubleshoot

default

Search

Create

Workloads

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Service

- Ingresses
- Ingress Classes
- Services

Config and Storage

- Config Maps
- Persistent Volume Claims
- Secrets

Create from input

App name \*

hello-flask

Container image \*

sahanaparveen/hello-flask:0.0.1.RELEASE

Number of pods \*

1

Service \*

External

Port \*

5000

Target port \*

5000

Protocol \*

TCP

Namespace \*

default

Deploy Preview Cancel Show advanced options

An 'app' label with this value will be added to the Deployment and Service that get deployed. [Learn more](#)

Enter the URL of a public image on any registry, or a private image hosted on Docker Hub or Google Container Registry. [Learn more](#)

A Deployment will be created to maintain the desired number of pods across your cluster. [Learn more](#)

Optionally, an internal or external Service can be defined to map an incoming Port to a target Port seen by the container. [Learn more](#)

Namespaces let you partition resources into logically named groups. [Learn more](#)

WhatsAppmycluster-free - IBM Cloudmycluster-free - Kubernetes DashboardIBMIBM-Project-17016-1659626790

eu-de.containers.cloud.ibm.com/kubeproxy/clusters/cdj0p7tf0ndbf15jdn0/service/#/deployment?namespace=default

kubernetes

default

Search

+🔔👤

Workloads > Deployments

Workloads

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

Service

Ingresses

Ingress Classes

Services

Config and Storage

Config Maps

Persistent Volume Claims

Secrets

Deployments

Name	Images	Labels	Pods	Created
hello-flask	Show all	Show all	1 / 1	48 seconds ago

29°C  
Mostly sunny

ENG  
IN

13:10  
05-11-2022