

Assignment Number	4
Assignment Date	28th October 2022
Team id	PNT2022TMID08777
Student Name	Kanimozhi.V
Student Roll Number	727619BEC010
Maximum marks	2 MARKS

Question:

- 1. Pull an Image from docker hub and run it in docker playground.**
- 2. Create a dockerfile for the job portal / flask application and deploy it in Docker desktop application.**
- 3. Create an IBM container registry and push a docker image of a flask application or job portal app.**
- 4. Create a Kubernetes cluster in IBM cloud and deploy flask application image or job portal image and also expose the same app to run in nodeport.**

Answers:

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

```
Command Prompt

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

C:\Users\prath>docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
docker101tutorial   latest         1310d78c8eb8   2 minutes ago   28.9MB
alpine/git          latest         42a1cda0ba24   13 days ago     43.6MB
hello-world         latest         feb5d9fea6a5   13 months ago   13.3kB

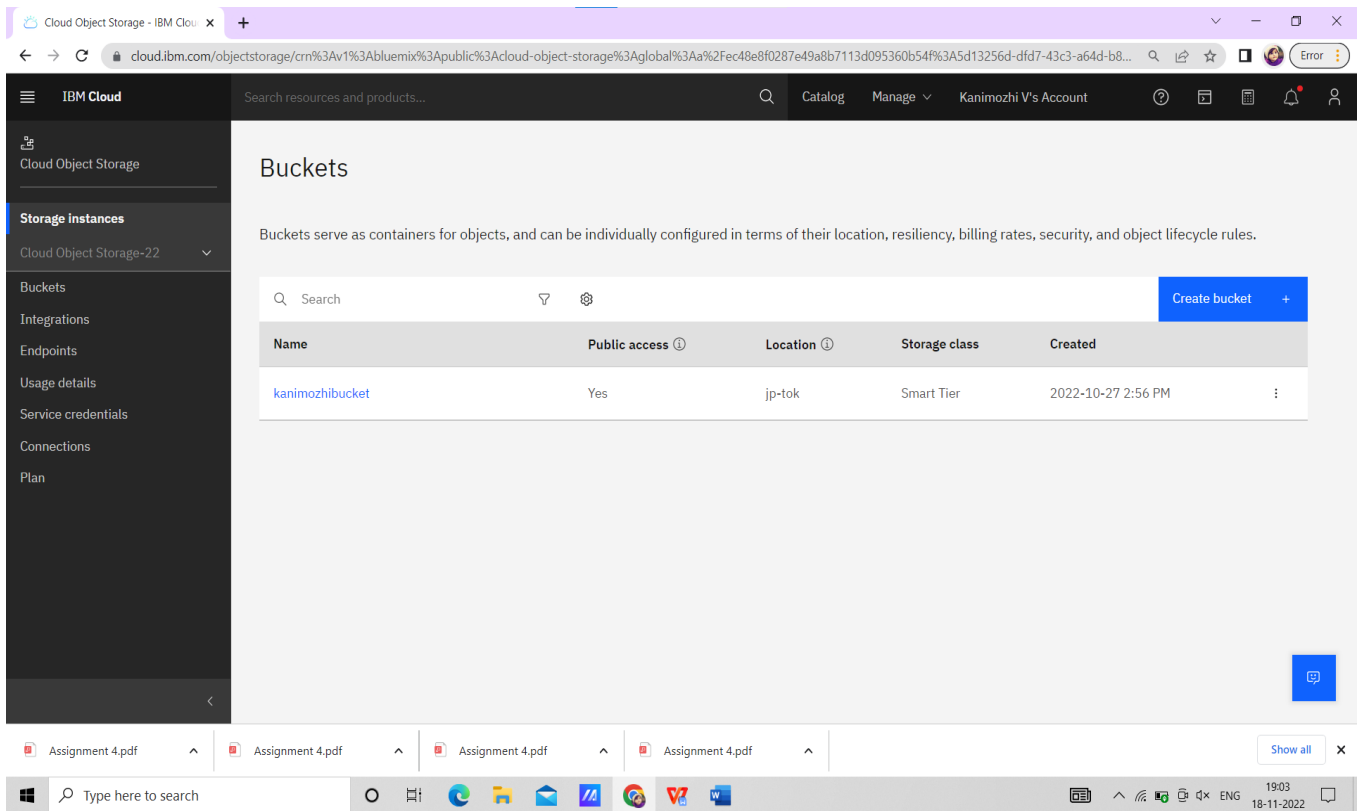
C:\Users\prath>
```

2.Create a dockerfile for the job portal / flask application and deploy it in Docker desktop application.

The screenshot shows the Docker Desktop application interface. On the left is a sidebar with navigation options: Containers, Images, Volumes, Dev Environments (marked BETA), Extensions (marked BETA), and Add Extensions. The main panel is titled 'Images on disk' and shows a list of local Docker images. At the top right of the main panel, it indicates 'Last refresh: Never', '4 Images', and a 'Refresh to see disk usage' button. Below this, there's a search bar and a checkbox for 'In use only'. The image list has columns for NAME, TAG, IMAGE ID, CREATED, and SIZE. Four images are listed: 'alpine', 'alpine/git', 'docker101tutorial', and 'hello-world'. Each image has a green 'IN USE' badge. The bottom status bar shows system information: RAM 2.49GB, CPU 0.03%, Connected to Hub, and version v4.13.1. The Windows taskbar is visible at the very bottom.

NAME	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	9c6f07244728	3 months ago	5.54 MB
alpine/git	latest	42a1cda0ba24	14 days ago	43.61 MB
docker101tutorial	latest	1310d78c8eb8	23 minutes ago	28.94 MB
hello-world	latest	feb5d9fea6a5	about 1 year ago	13.26 KB

3. Create an IBM container registry and push a docker image of a flask application or job portal app.



4. Create a Kubernetes cluster in IBM cloud and deploy flask application image or job portal image and also expose the same app to run in nodeport

2. Change directory to Lab 1:

```
cd "Lab 1"
```

3. Log in to the IBM Cloud CLI:

```
ibmcloud login
```

To specify an IBM Cloud region, include the API endpoint.

4. In order to upload images to the IBM Cloud Container Registry, you first need to create a namespace with the following command:

```
ibmcloud cr namespace-add <my_namespace>
```

5. Build the container image with a `1` tag and push the image to the IBM Cloud

Registry: `ibmcloud cr build --tag
us.icr.io/<my_namespace>/hello-world:1 .`

6. Verify the image is built:

```
ibmcloud cr images
```