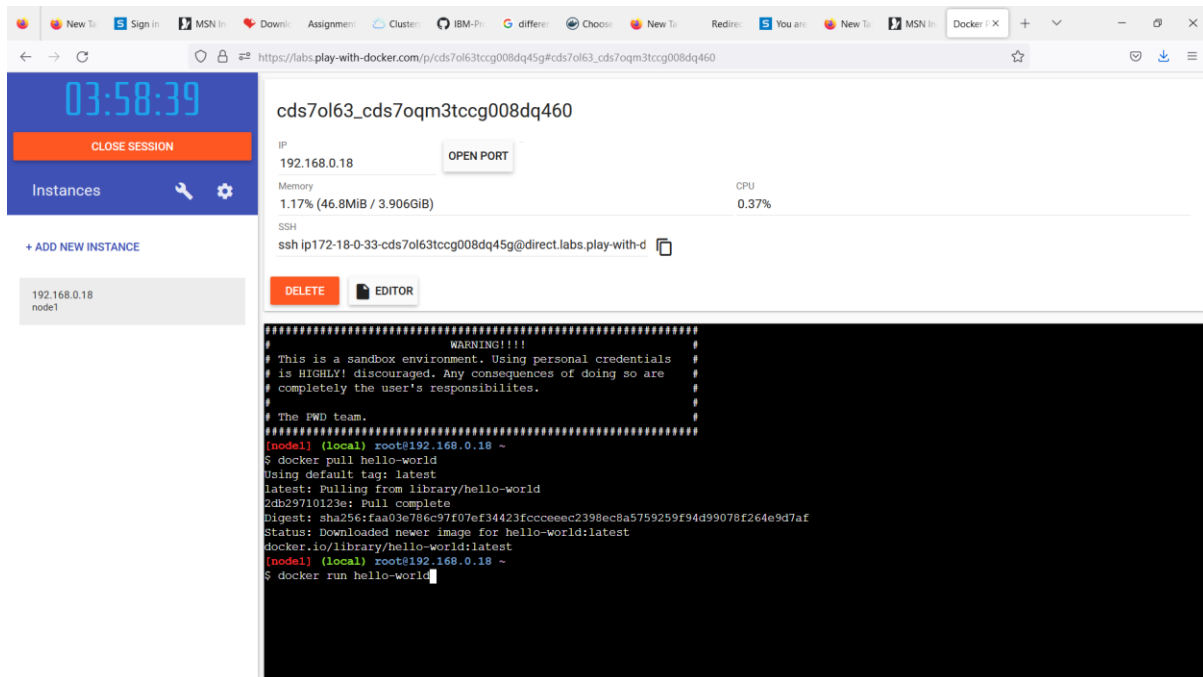


# 1.Pull an image from docker hub and run it in docker playground

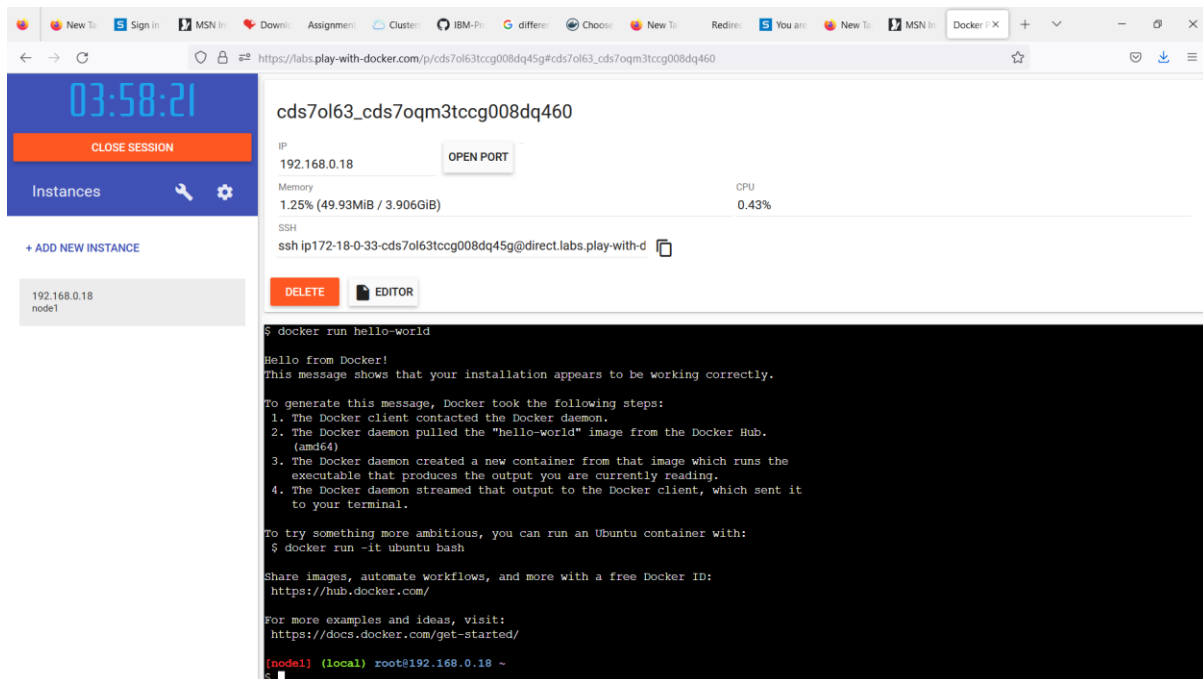


The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:58:39, a 'CLOSE SESSION' button, and a list of instances with one instance named 'node1' at IP 192.168.0.18. The main area displays the details for the instance 'node1' at IP 192.168.0.18. It shows memory usage at 1.17% (46.8MiB / 3.906GiB) and CPU usage at 0.37%. Below this, there's a terminal window with the following content:

```
WARNING!!!!
This is a sandbox environment. Using personal credentials
is HIGHLY discouraged. Any consequences of doing so are
completely the user's responsibilities.

The PWD team.

(node1) (local) root@192.168.0.18 ~
$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:faa03e766e97f07ef34423fccc0ec2398ec8a5759259f94d99078f264e9d7af
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
(node1) (local) root@192.168.0.18 ~
$ docker run hello-world
```



The screenshot shows the Docker Playground interface after running the 'hello-world' container. The sidebar is the same. The main area shows the instance 'node1' at IP 192.168.0.18 with memory usage at 1.25% (49.93MiB / 3.906GiB) and CPU usage at 0.43%. The terminal window now shows the output of the 'docker run hello-world' command:

```
$ docker run hello-world

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.

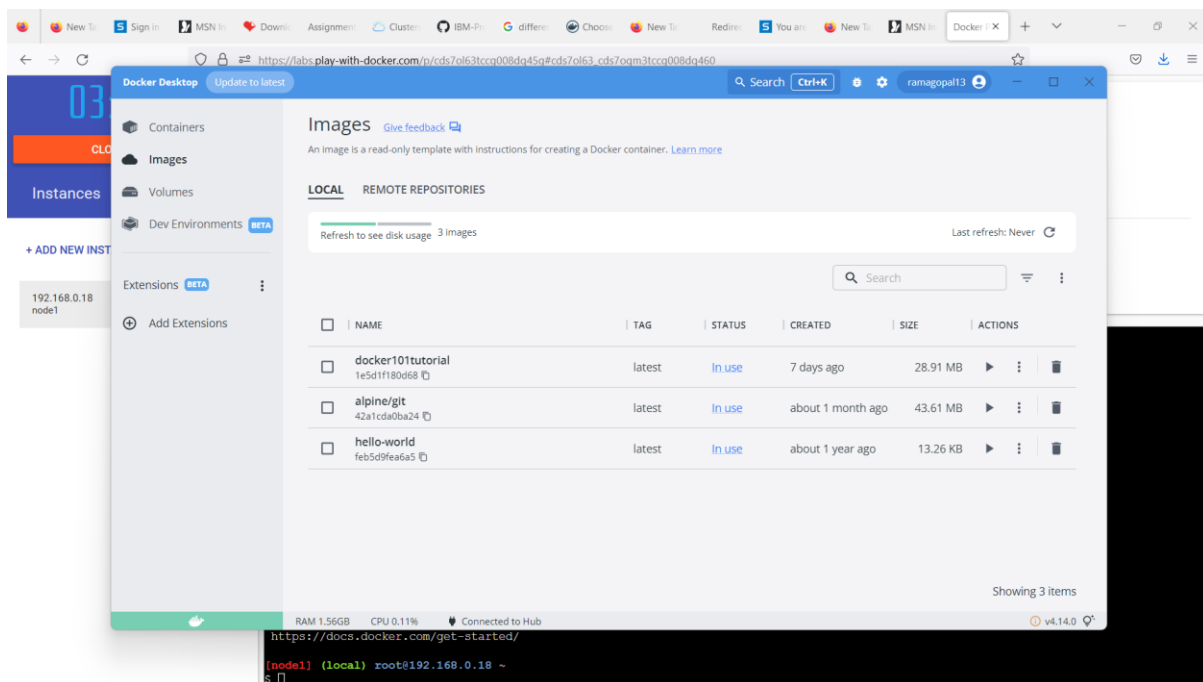
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

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

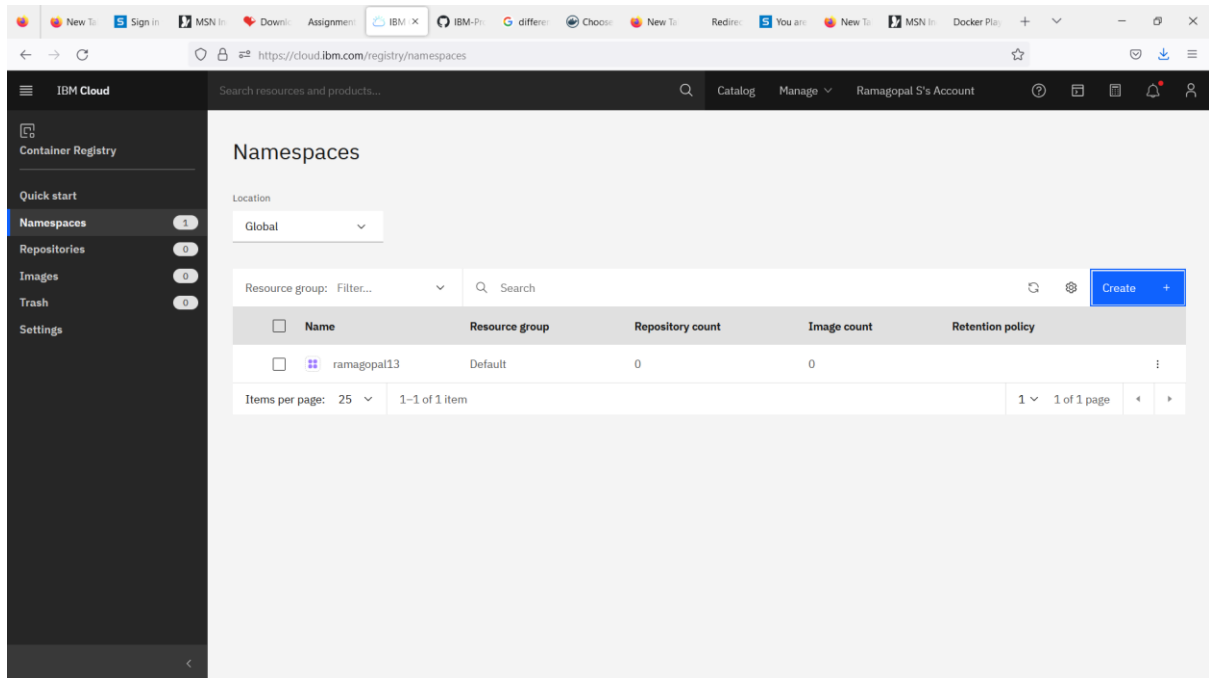
For more examples and ideas, visit:
https://docs.docker.com/get-started/
(node1) (local) root@192.168.0.18 ~
$
```

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

```
FROM
python:3.7
COPY.
/app
WORKDI
R /app
COPY requirements.txt /app
RUN python -m pip install -r
requirements.txtEXPOSE 5001
ENTRYPOINT [
"python" ]CMD [
"app.py" ]
```



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



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

The screenshot shows the IBM Cloud Kubernetes cluster overview page for a cluster named 'mycluster-free'. The cluster is in a 'Normal' state and will expire in 30 days. The page displays various status metrics and details.

**Cluster Overview:**

- Cluster Name: mycluster-free
- Status: Normal
- Expires in: 30 days
- Actions: [Kubernetes dashboard](#), [Actions...](#)

**Overview Metrics:**

- Node status: 1 of 1, Normal
- Add-on status: 0 of 0, Normal
- Master status: Normal
- Ingress status: Healthy

**Details:**

Cluster ID	Version	Infrastructure	Zones
cds75ahf8e0ek38o98fg	1.24.8_1544	Classic	Milan 01

Created	Resource group	Image security enforcement
11/19/2022, 11:37 AM	Default	<a href="#">Enable</a>

**Node health:** 1 total nodes (represented by a green bar)

[Worker node details](#)