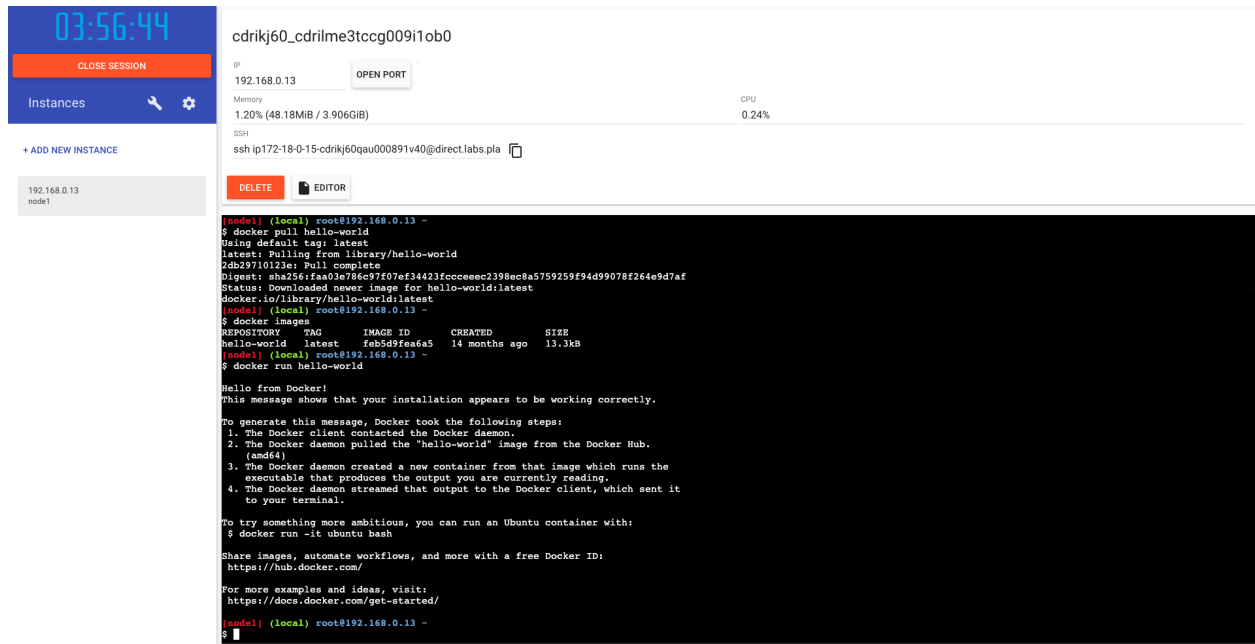


## 1. Pull an Image from docker hub and run it in docker playground.



The screenshot shows the Docker Playground interface. On the left, there's a sidebar with a clock showing 03:56:44, a 'CLOSE SESSION' button, and an 'Instances' section with a '+ ADD NEW INSTANCE' button. Below that, a list of instances shows '192.168.0.13 node1'. The main area displays the instance details for 'cdrikj60\_cdrilme3tccg009i1ob0' with IP '192.168.0.13', memory usage '1.20% (48.18MiB / 3.906GiB)', and CPU usage '0.24%'. There are 'DELETE' and 'EDITOR' buttons. The terminal window shows the following commands and output:

```
[node1] (local) root@192.168.0.13 ~
$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:faa02a7d6c97107ef34423fccc0002398ec8a5759259f94d99078f264e9d7af
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
[node1] (local) root@192.168.0.13 ~
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest             feb5d9fea6a5        14 months ago      13.3kB
[node1] (local) root@192.168.0.13 ~
$ docker run hello-world

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

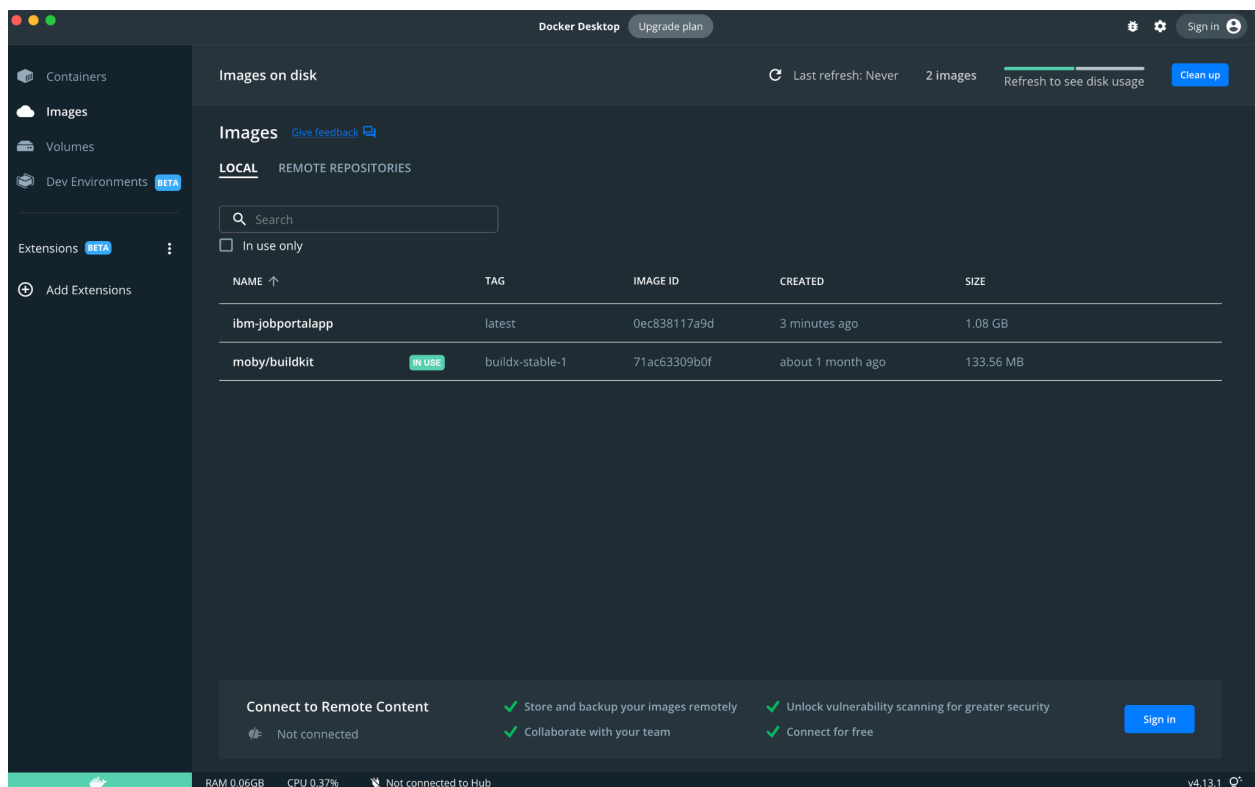
To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (and64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the 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@192.168.0.13 ~
$
```

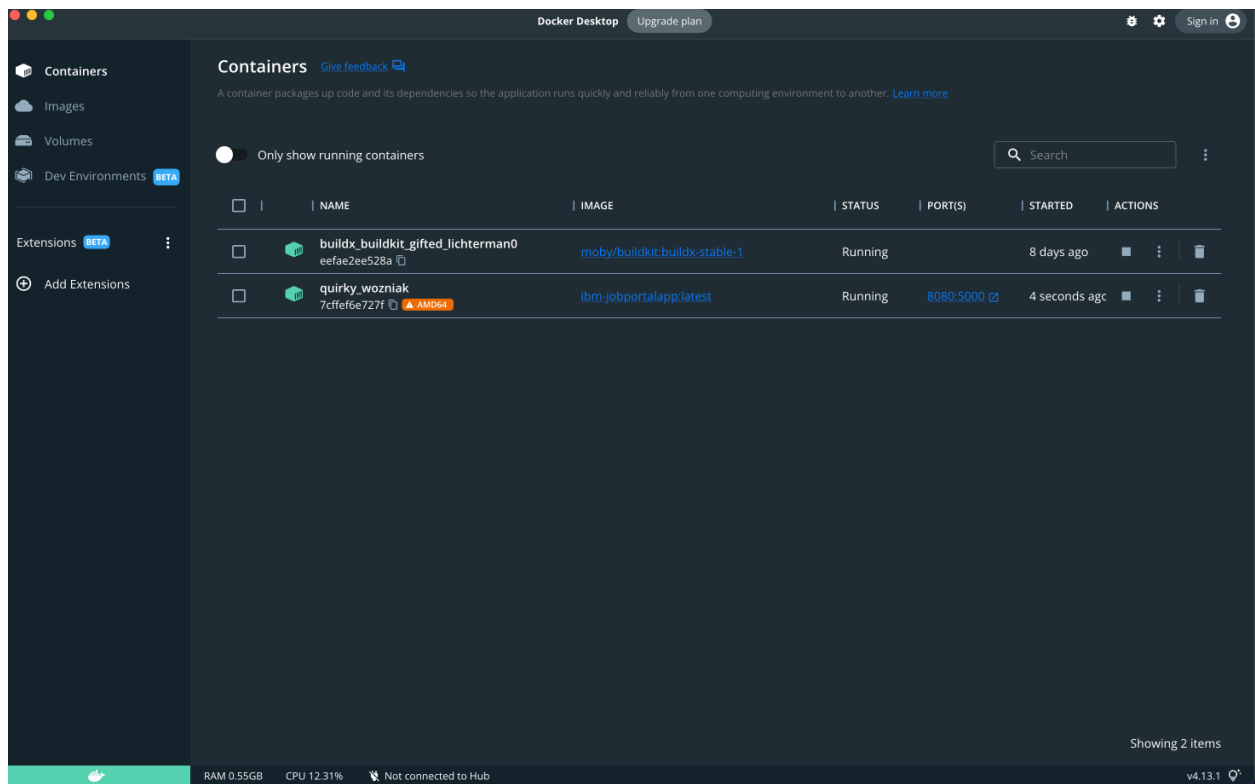
## 2. Create a docker file for the jobportal application and deploy it in the Docker desktop application.



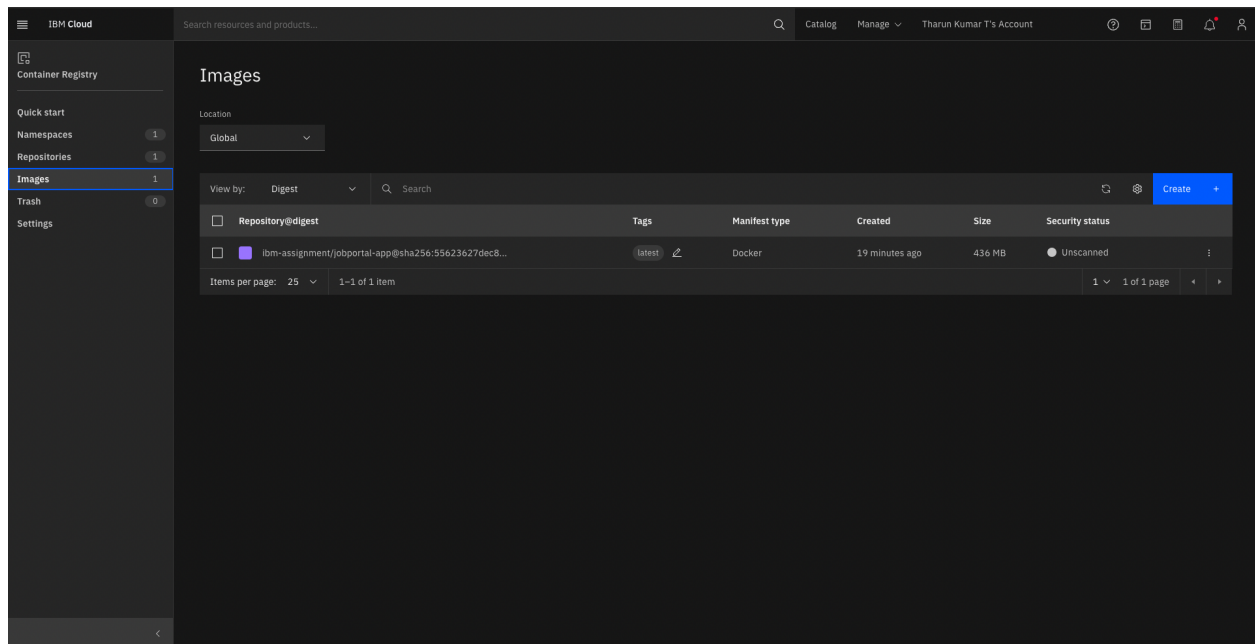
The screenshot shows the Docker Desktop interface. On the left, there's a sidebar with 'Containers', 'Images', 'Volumes', 'Dev Environments', 'Extensions', and 'Add Extensions'. The main area displays 'Images on disk' with a search bar and a table of installed images. The table has columns for NAME, TAG, IMAGE ID, CREATED, and SIZE. The images listed are 'ibm-jobportalapp' and 'moby/buildkit'.

NAME	TAG	IMAGE ID	CREATED	SIZE
ibm-jobportalapp	latest	0ec838117a9d	3 minutes ago	1.08 GB
moby/buildkit	buildx-stable-1	71ac63309b0f	about 1 month ago	133.56 MB

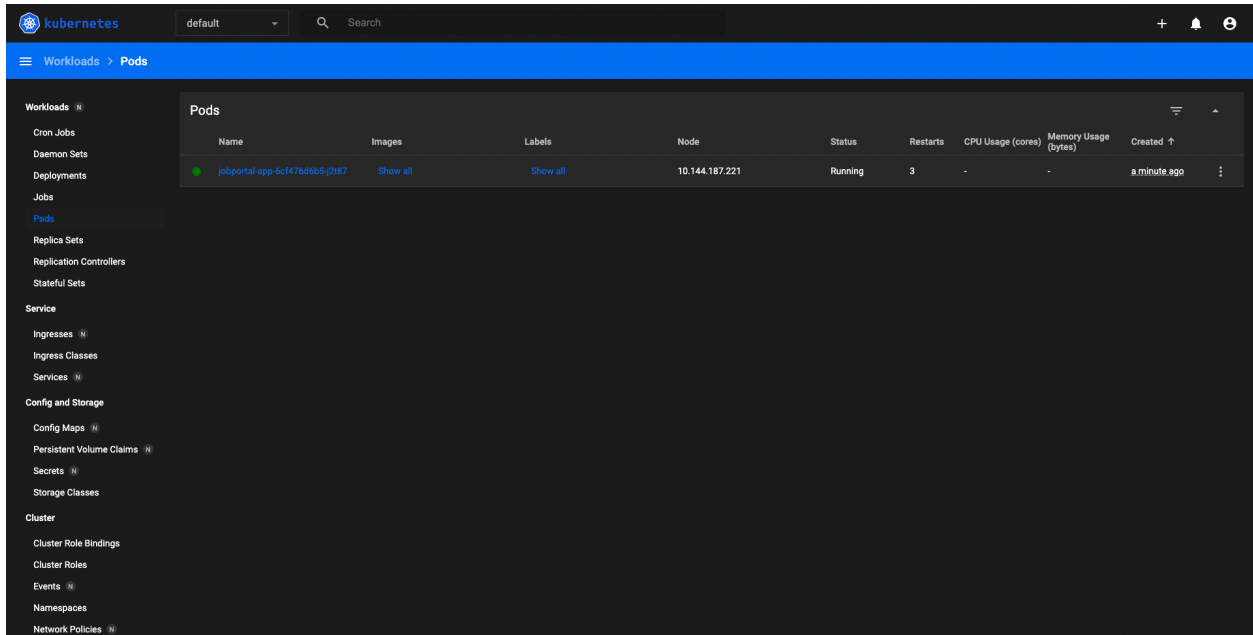
At the bottom, there's a 'Connect to Remote Content' section with a 'Sign in' button. The status bar at the bottom shows 'RAM 0.06GB', 'CPU 0.37%', and 'Not connected to Hub'.



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 jobportal image and also expose the same app to run in nodeport.



The screenshot shows the Kubernetes dashboard interface. The top navigation bar includes the Kubernetes logo, a namespace dropdown set to 'default', a search bar, and user profile icons. The left sidebar lists various Kubernetes resources under categories like Workloads, Service, Config and Storage, and Cluster. The 'Pods' resource is selected, and the main panel displays a table of pods. One pod is listed with the name 'jobportal-app-6cf476d6b5-j2t87', running on node '10.144.187.221' with a status of 'Running'.

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
jobportal-app-6cf476d6b5-j2t87	<a href="#">Show all</a>	<a href="#">Show all</a>	10.144.187.221	Running	3	-	-	a minute ago