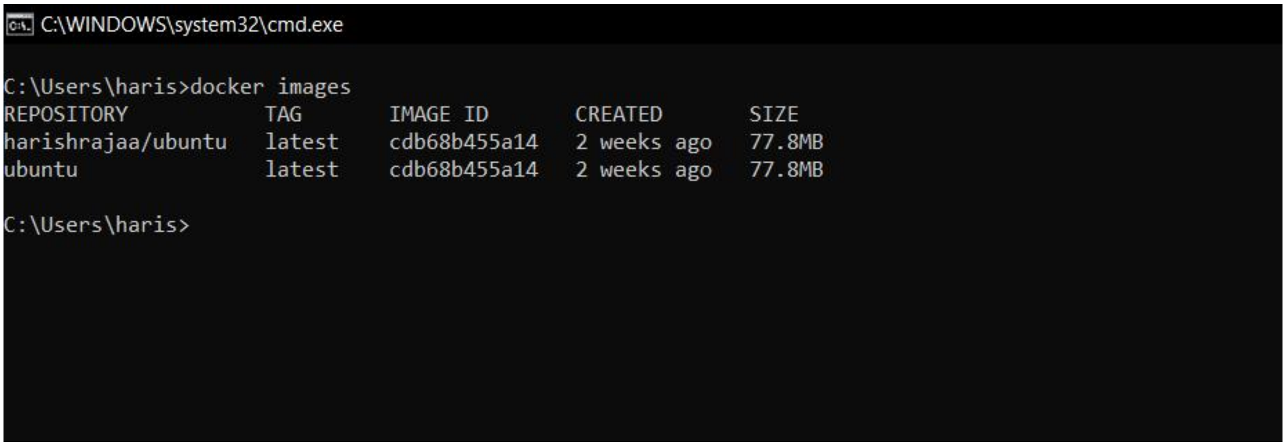
ASSIGNMENT -4
(PNT2022TMID02140)

HARISH RAJAA M
HEMANATHAN C
HARISH G.V
JAGADESH P

1) PULL AN IMAGE FROM DOCKER HUB AND RUN IT IN DOCKER PLAYGROUND.

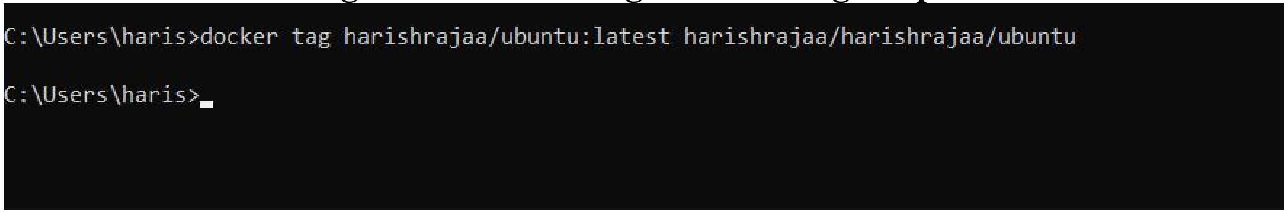
STEP-1 : List out the< docker images>



C:\WINDOWS\system32\cmd.exe

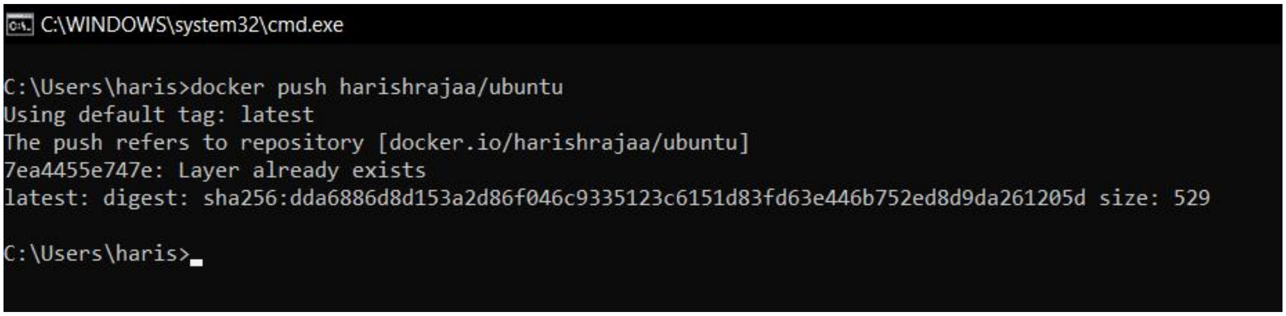
```
C:\Users\haris>docker images
REPOSITORY          TAG       IMAGE ID       CREATED       SIZE
harishrajaa/ubuntu   latest    cdb68b455a14   2 weeks ago   77.8MB
ubuntu              latest    cdb68b455a14   2 weeks ago   77.8MB

C:\Users\haris>
```

STEP-2:Add the tag name for existing docker image to push the docker hub



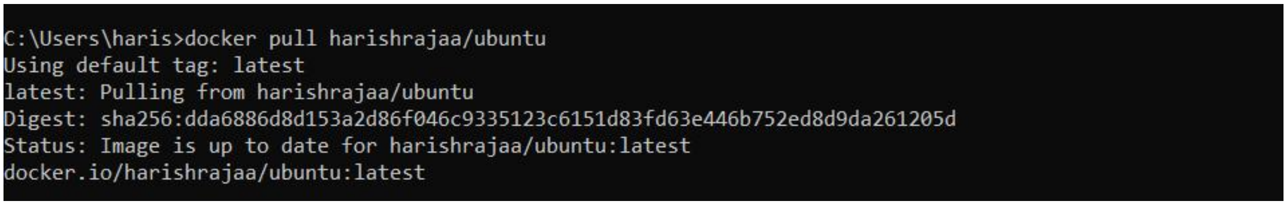
```
C:\Users\haris>docker tag harishrajaa/ubuntu:latest harishrajaa/harishrajaa/ubuntu

C:\Users\haris>
```

STEP-3: Push to docker hub



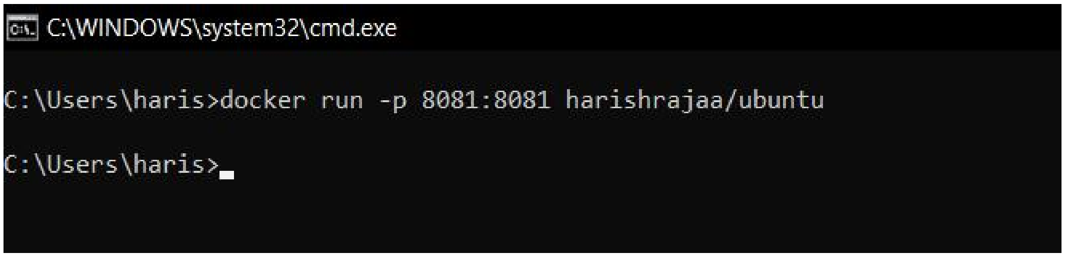
```
C:\Users\haris>docker push harishrajaa/ubuntu
Using default tag: latest
The push refers to repository [docker.io/harishrajaa/ubuntu]
7ea4455e747e: Layer already exists
latest: digest: sha256:dda6886d8d153a2d86f046c9335123c6151d83fd63e446b752ed8d9da261205d size: 529

C:\Users\haris>
```

STEP-4: Pull the docker images from the docker hub to docker container.

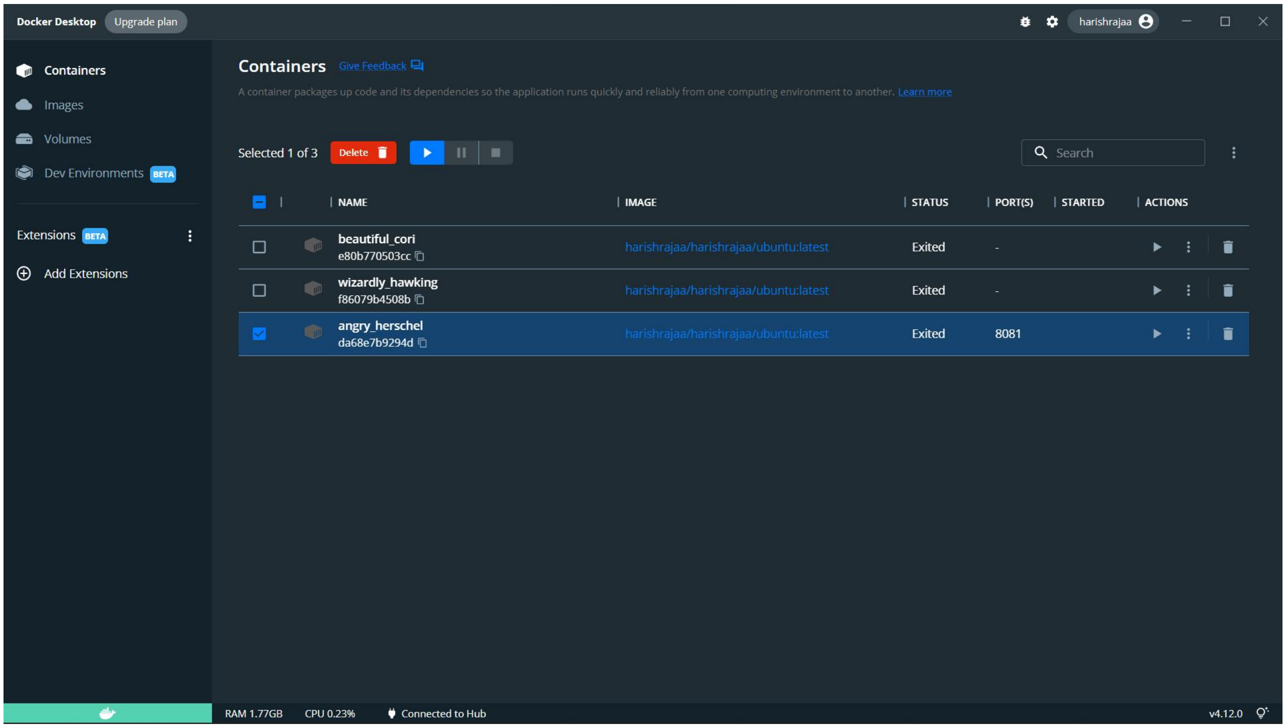


```
C:\Users\haris>docker pull harishrajaa/ubuntu
Using default tag: latest
latest: Pulling from harishrajaa/ubuntu
Digest: sha256:dda6886d8d153a2d86f046c9335123c6151d83fd63e446b752ed8d9da261205d
Status: Image is up to date for harishrajaa/ubuntu:latest
docker.io/harishrajaa/ubuntu:latest
```

STEP-5 :Run the docker container.



```
C:\Users\haris>docker run -p 8081:8081 harishrajaa/ubuntu

C:\Users\haris>
```



Containers

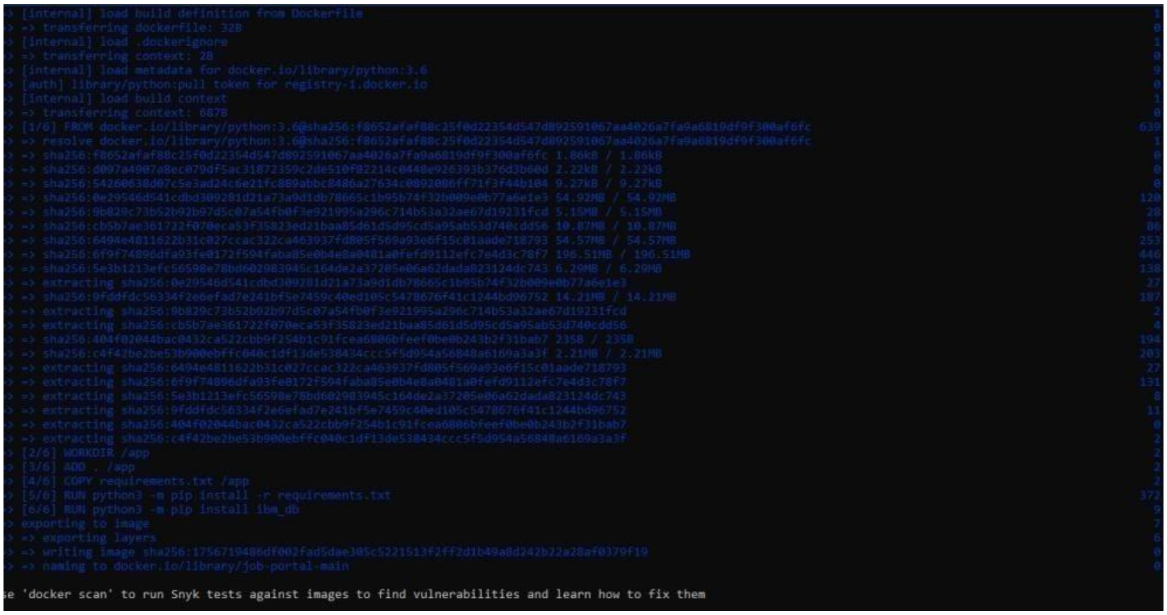
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Selected 1 of 3

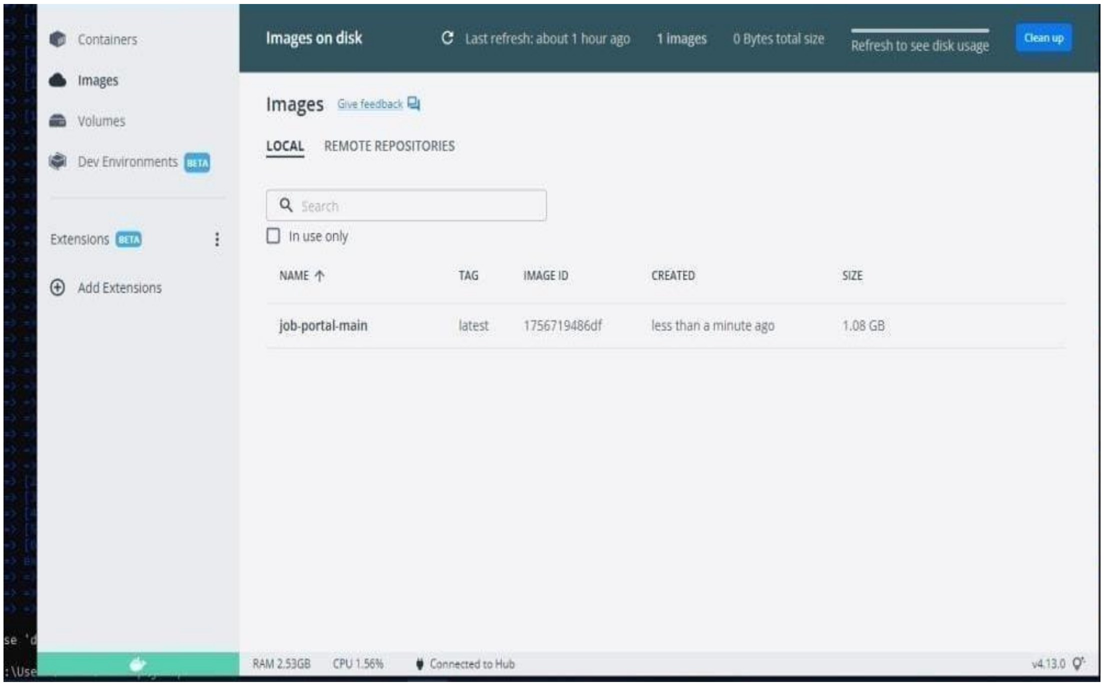
☐	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
☐	beautiful_cori e80b770503cc	harishrajaa/harishrajaa/ubuntu:latest	Exited	-		
☐	wizardly_hawking f86079b4508b	harishrajaa/harishrajaa/ubuntu:latest	Exited	-		
☑	angry_herschel da68e7b9294d	harishrajaa/harishrajaa/ubuntu:latest	Exited	8081		

RAM 1.77GB CPU 0.23% Connected to Hub v4.12.0

2) CREATE A DOCKER FILE FOR THE JOBPORTAL APPLICATION AND DEPLOY IT IN DOCKER DESKTOP APPLICATION



```
> [internal] load build definition from Dockerfile                                                                                    1
> => transferring dockerfile: 32B                                                                                                     0
> [internal] load .dockerignore                                                                                                       1
> => transferring context: 2B                                                                                                         0
> [internal] load metadata for docker.io/library/python:3.6                                                                           9
> [auth] library/python:pull token for registry-1.docker.io                                                                           0
> [internal] load build context                                                                                                       1
> => transferring context: 687B                                                                                                       0
> [1/6] FROM docker.io/library/python:3.6@sha256:f8652afaf88c25f0d22354d547d892591067aa4026a7fa9a6819df9f300af6fc                     639
> => resolve docker.io/library/python:3.6@sha256:f8652afaf88c25f0d22354d547d892591067aa4026a7fa9a6819df9f300af6fc                     1
> => sha256:f8652afaf88c25f0d22354d547d892591067aa4026a7fa9a6819df9f300af6fc 1.86kB / 1.86kB                                          0
> => sha256:d097a4907a8ec079df5ac31872359c2de510f82214c0448e926393b376d3b60d 2.22kB / 2.22kB                                          0
> => sha256:54260638d07c5e3ad24c6e21fc889abbc8486a27634c0892086ff71f3f44b104 9.27kB / 9.27kB                                          0
> => sha256:0e29546d541cdbd309281d21a73a9d1db78665c1b95b74f32b009e0b77a6e1e3 54.92MB / 54.92MB                                        120
> => sha256:9b829c73b52b92b97d5c07a54fb0f3e921995a296c714b53a32ae67d19231fcd 5.15MB / 5.15MB                                          28
> => sha256:cb5b7ae361722f070eca53f35823ed21baa85d61d5d95cd5a95ab53d740cdd56 10.87MB / 10.87MB                                        86
> => sha256:6494e4811622b31c027ccac322ca463937fd805f569a93e6f15c01aade718793 54.57MB / 54.57MB                                        253
> => sha256:6f9f74896dfa93fe0172f594faba85e0b4e8a0481a0fefd9112efc7e4d3c78f7 196.51MB / 196.51MB                                      446
> => sha256:5e3b1213efc56598e78bd602983945c164de2a37205e06a62dada823124dc743 6.29MB / 6.29MB                                          138
> => extracting sha256:0e29546d541cdbd309281d21a73a9d1db78665c1b95b74f32b009e0b77a6e1e3                                               27
> => sha256:9fddfdc56334f2e6efad7e241bf5e7459c40ed105c5478676f41c1244bd96752 14.21MB / 14.21MB                                        187
> => extracting sha256:9b829c73b52b92b97d5c07a54fb0f3e921995a296c714b53a32ae67d19231fcd                                               2
> => extracting sha256:cb5b7ae361722f070eca53f35823ed21baa85d61d5d95cd5a95ab53d740cdd56                                               4
> => sha256:404f02044bac0432ca522cbb9f254b1c91fcea6806bfeef0be0b243b2f31bab7 235B / 235B                                              194
> => sha256:c4f42be2be53b900ebffc040c1df13de538434ccc5f5d954a56848a6169a3a3f 2.21MB / 2.21MB                                          203
> => extracting sha256:6494e4811622b31c027ccac322ca463937fd805f569a93e6f15c01aade718793                                               27
> => extracting sha256:6f9f74896dfa93fe0172f594faba85e0b4e8a0481a0fefd9112efc7e4d3c78f7                                               131
> => extracting sha256:5e3b1213efc56598e78bd602983945c164de2a37205e06a62dada823124dc743                                               0
> => extracting sha256:9fddfdc56334f2e6efad7e241bf5e7459c40ed105c5478676f41c1244bd96752                                               11
> => extracting sha256:404f02044bac0432ca522cbb9f254b1c91fcea6806bfeef0be0b243b2f31bab7                                               0
> => extracting sha256:c4f42be2be53b900ebffc040c1df13de538434ccc5f5d954a56848a6169a3a3f                                               2
> [2/6] WORKDIR /app                                                                                                                  2
> [3/6] ADD . /app                                                                                                                    2
> [4/6] COPY requirements.txt /app                                                                                                    2
> [5/6] RUN python3 -m pip install -r requirements.txt                                                                                372
> [6/6] RUN python3 -m pip install ibm_db                                                                                             9
> exporting to image                                                                                                                  7
> => exporting layers                                                                                                                 6
> => writing image sha256:1756719486df002fad5dae305c5221513f2ff2d1b49a8d242b22a28af0379f19                                            0
> => naming to docker.io/library/job-portal-main                                                                                      0

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

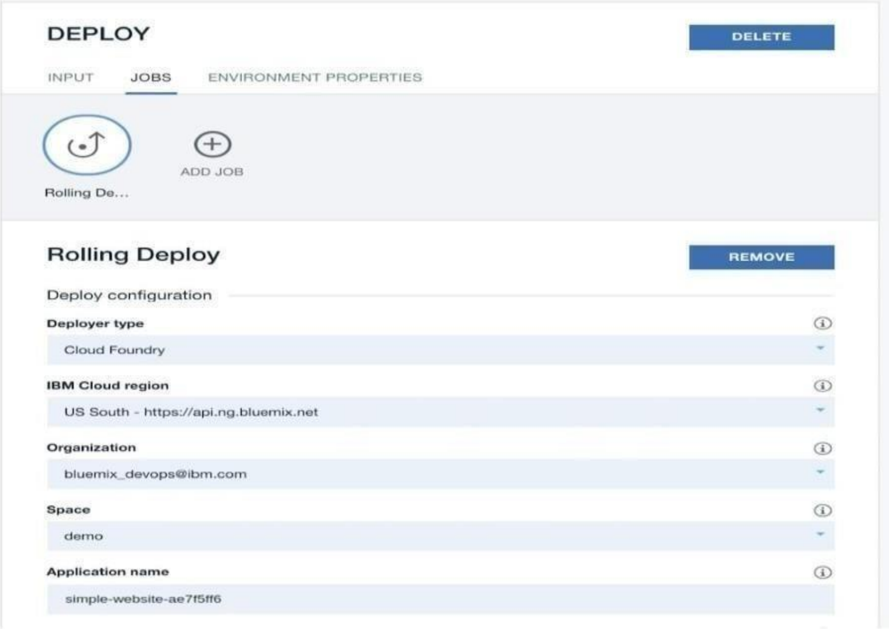


NAME ↑	TAG	IMAGE ID	CREATED	SIZE
job-portal-main	latest	1756719486df	less than a minute ago	1.08 GB

3) CREATE A IBM CONTAINER REGISTRY AND DEPLOY HELLOWORLD APP OR JOBPORTALAPP.

Solution:

```
         <html>
   <body>
      Hello,IBMCloudWorld!
   </body>
</html>---
application
s:
 - buildpack: https://github.com/cloudfoundry/staticfile-
 buildpack.githost:simple-website-${random}
 name: simple-website-
 ${random}memory:64M
 stack:cflinuxfs2
```



DEPLOY

Rolling Deploy

Deploy configuration

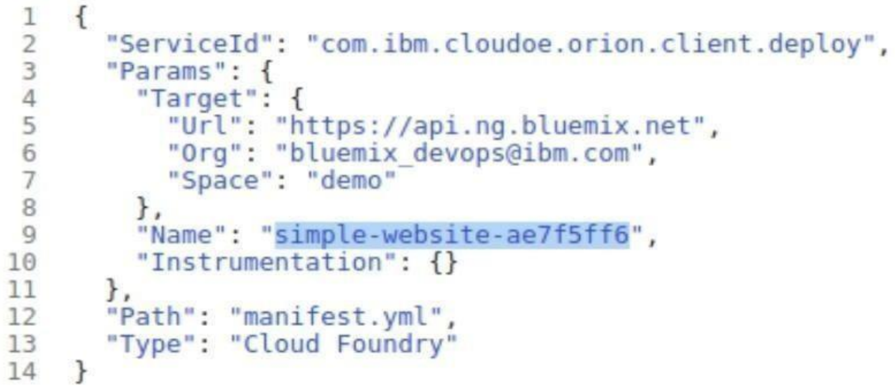
Deployer type: Cloud Foundry

IBM Cloud region: US South - https://api.ng.bluemix.net

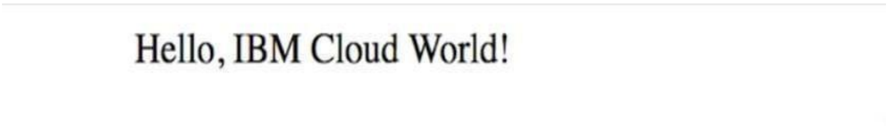
Organization: bluemix_devops@ibm.com

Space: demo

Application name: simple-website-ae7f5ff6



```
{
  "ServiceId": "com.ibm.cloudoe.orion.client.deploy",
  "Params": {
    "Target": {
      "Url": "https://api.ng.bluemix.net",
      "Org": "bluemix_devops@ibm.com",
      "Space": "demo"
    },
    "Name": "simple-website-ae7f5ff6",
    "Instrumentation": {}
  },
  "Path": "manifest.yml",
  "Type": "Cloud Foundry"
}
```



Hello, IBM Cloud World!

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.

Solution:

ibmcloudtarget-g<resource_group_name>ibmcloudcrnishanthc-add <your_nishanthc>ibmcloudresource service-instance-create example-postgresql databases-for-postgresql standard us- southibmcloudks cluster-service-bind mycluster default example-postgresqlgit clone -b node git@github.com:IBM-Cloud/clouddatabases-helloworld-kubernetes-examples.gitspec:

replicas:3name:cloudpostgres-nodejs-app

image:"registry.<region>.bluemix.net/<namespace>/icdpg"#Editme

imagePullPolicy: AlwaysibmcloudcrregionYou are targeting region 'us-south', the registryis'registry.ng.bluemix.net'.ibmcloudcr build -t
registry.ng.bluemix.net/<namespace>/icdpg
.ibmcloudcrimages

env:

- name:

BINDINGval

ueFrom:

```
        secretKeyRef:
         name: <postgres-secret-name> # Edit
         mekey:binding
apiVersion:
v1kind:
Servicemeta
data:
 name: cloudpostgres-
 servicelabels:
  run: clouddb-
demospec: type:
NodePortsele
ctor:
  run: clouddb-
 demoports:
 -
  protocol:TC
  Pport:8080
  nodePort
```