

ASSIGNMENT -4

JAYAPRAKSH P

JOYSON SAMUEL P

JAYA SIVAAKRISHNA B

SHAIKH FIRAAS A

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

STEP-1 : List out the< docker images>

```
C:\Users\jayap>docker images
REPOSITORY                                TAG      IMAGE ID      CREATED        SIZE
jayaprakash2019/docker101tutorial         latest   94ab66b2cb4d  3 weeks ago    28.9MB
docker/getting-started                    latest   cb90f98fd791  6 months ago   28.8MB
```

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

```
C:\Users\jayap>docker tag jayaprakash2019/docker101tutorial:latest jayaprakash2019/jayaprakash2019/docker101tutorial
```

STEP-3: Push to docker hub

<https://hub.docker.com/r/jayaprakash2019/docker101tutorial>

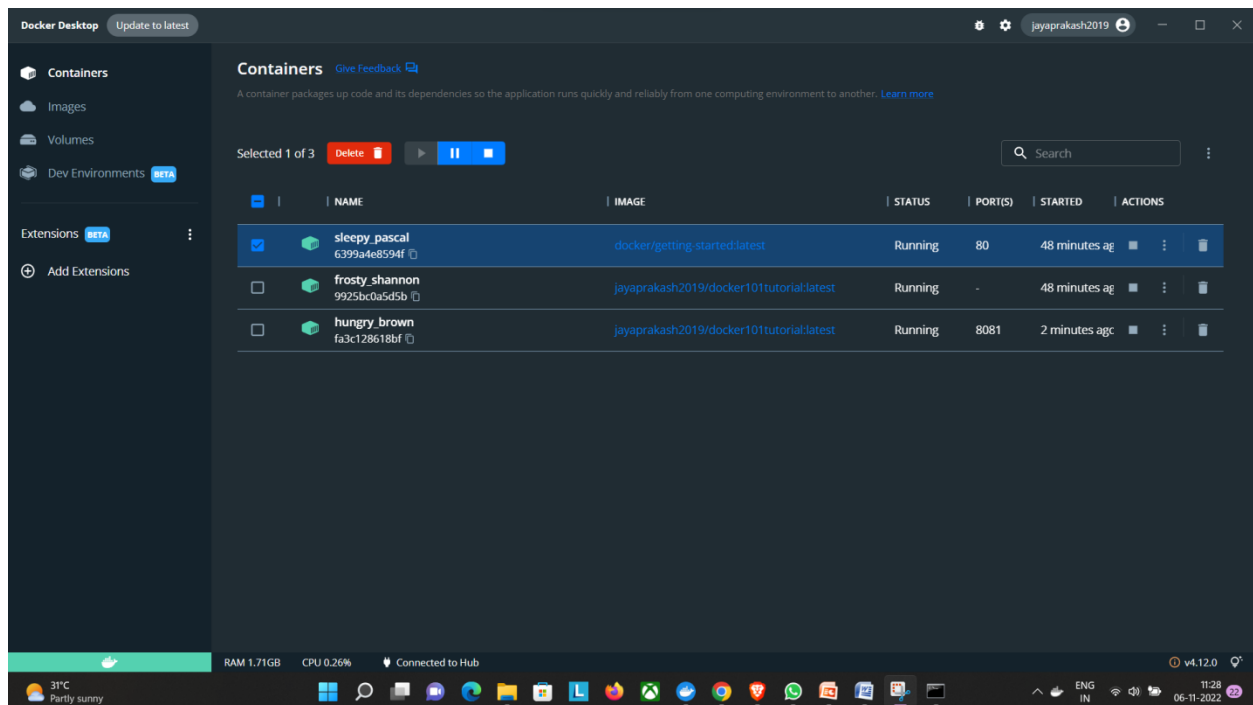
```
C:\Users\jayap>docker push jayaprakash2019/docker101tutorial
Using default tag: latest
The push refers to repository [docker.io/jayaprakash2019/docker101tutorial]
6bfa574ae9d8: Pushed
26be16e2f5ce: Pushed
00e62d52c55c: Layer already exists
403a87f26438: Layer already exists
b26a89445b38: Layer already exists
0f7dbcfcd2a: Layer already exists
492c31ac001f: Pushed
994393dc58e7: Layer already exists
latest: digest: sha256:ec5151b5ed5ba01931614d029ba15e99e319d514534fb93528b9d358b75e8b0f size: 1990
```

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

```
C:\Users\jayap>docker pull jayaprakash2019/docker101tutorial
Using default tag: latest
latest: Pulling from jayaprakash2019/docker101tutorial
Digest: sha256:ec5151b5ed5ba01931614d029ba15e99e319d514534fb93528b9d358b75e8b0f
Status: Image is up to date for jayaprakash2019/docker101tutorial:latest
docker.io/jayaprakash2019/docker101tutorial:latest
```

STEP-5 :Run the docker container

```
C:\Users\jayap>docker run -p 8081:8081 jayaprakash2019/docker101tutorial
```



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

```
[internal] load build definition from Dockerfile
--> transferring dockerfile: 32B
[internal] load .dockerignore
--> transferring context: 2B
[internal] load metadata for docker.io/library/python:3.6
[auth] library/python:pull token for registry-1.docker.io
[internal] load build context
--> transferring context: 687B
[1/6] FROM docker.io/library/python:3.6@sha256:f8052aaf88c25fd22354d547d892501067aa026a7fa0a0819df9f300af6fc
--> resolve docker.io/library/python:3.6@sha256:f8052aaf88c25fd22354d547d892501067aa026a7fa0a0819df9f300af6fc
--> sha256:f8052aaf88c25fd22354d547d892501067aa026a7fa0a0819df9f300af6fc 1.86kB / 1.86kB
--> sha256:0097aa007d8ec070d75ac31872359c2de510f82214c0448e926392b37cd30e0d 2.22kB / 2.22kB
--> sha256:54200830097c5e2a24c6e22fc889dbcc9408a276d410092008ff7415f44b104 9.22kB / 9.22kB
--> sha256:8e39566d541c8d380181021a73a0d1db7865c1b95b74f32b009a8c77ade1a3 54.92MB / 54.92MB
--> sha256:9b89c73b52b02b9795c87a54fb0f3e921995a296c714b53a32ae7d19211fcd 5.10MB / 5.10MB
--> sha256:c85b7ae3b173f070eca53f35823ed21baa85d61d5d95cd5a95ab53d746cdd5e 10.87MB / 10.87MB
--> sha256:6404e4811622b31c027ccac322ca63937fd805f560a9306f15c81aade718793 54.57MB / 54.57MB
--> sha256:6f9f7480ddfa93fe8172f504faba85e0bae8a041a0fef0d112efc7e4d3c78f7 196.51MB / 196.51MB
--> sha256:5e3b1213efc56598e78b0d02983045c164de2a37205e0ea62dada823134dc743 6.20MB / 6.20MB
--> extracting sha256:0e29546d541c8d380181021a73a0d1db7865c1b95b74f32b009a8c77ade1a3
--> sha256:9f5d9dc56334f2e6efad7e241bf5e7459c40ed105c5478676f41c1244bd96752 14.21MB / 14.21MB
--> extracting sha256:0b820c73b02b02b9705c87454f0e7e0921995a296c714b53a32ae7d19211fcd
--> extracting sha256:c85b7ae3b173f070eca53f35823ed21baa85d61d5d95cd5a95ab53d746cdd5e
--> sha256:404f02044bac0432ca522cbb9f254b1c91fca080b0feef0be0b243b2f31bab7 235B / 235B
--> sha256:c4f42be2b53b900ebff04ec1d0f13de538434cc3f5d0954a5048ae109a3af 2.21MB / 2.21MB
--> extracting sha256:6404e4811622b31c027ccac322ca63937fd805f560a9306f15c81aade718793
--> extracting sha256:6f9f7480ddfa93fe8172f504faba85e0bae8a041a0fef0d112efc7e4d3c78f7
--> extracting sha256:5e3b1213efc56598e78b0d02983045c164de2a37205e0ea62dada823134dc743
--> extracting sha256:9f5d9dc56334f2e6efad7e241bf5e7459c40ed105c5478676f41c1244bd96752
--> extracting sha256:404f02044bac0432ca522cbb9f254b1c91fca080b0feef0be0b243b2f31bab7
--> extracting sha256:c4f42be2b53b900ebff04ec1d0f13de538434cc3f5d0954a5048ae109a3af
[2/6] WORKDIR /app
[3/6] ADD . /app
[4/6] COPY requirements.txt /app
[5/6] RUN python3 -m pip install -r requirements.txt
[6/6] RUN python3 -m pip install ibm_db
--> exporting to image
--> exporting layers
--> writing image sha256:1756719486df002fad5dae305c5221513f2ff2d1b49a8d242b22a28af0379f10
--> naming to docker.io/library/job-portal-main
```

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

The screenshot shows the Docker Desktop application window. On the left is a sidebar with navigation icons for Containers, Images, Volumes, and Dev Environments (marked as BETA). Below these are Extensions (also BETA) and an 'Add Extensions' button. The main panel is titled 'Images on disk' and includes a refresh button and statistics: 'Last refresh: about 1 hour ago', '1 Images', and '0 Bytes total size'. A 'Clean up' button is in the top right. Under the 'LOCAL' tab, there is a search bar and a checkbox for 'In use only'. A table displays the local images:

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

At the bottom of the window, a status bar shows 'RAM 2.53GB', 'CPU 1.56%', 'Connected to Hub', and the version 'v4.13.0'.

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.github:simple-website-#{random}
  name: simple-website-
  #{random}memory:64M
  stack:cflinuxfs2
```

The screenshot shows the IBM Cloud Deploy console interface. At the top, there's a 'DEPLOY' header with a 'DELETE' button. Below it are tabs for 'INPUT', 'JOBS', and 'ENVIRONMENT PROPERTIES'. The 'JOBS' tab is active, showing a 'Rolling Deploy' section with a 'ROLLING DEPLOY' button. The 'Rolling Deploy' section contains a 'Deploy configuration' table with the following fields:

Field	Value
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-ae715ff6

```

1  {
2    "ServiceId": "com.ibm.cloudoe.orion.client.deploy",
3    "Params": {
4      "Target": {
5        "Url": "https://api.ng.bluemix.net",
6        "Org": "bluemix_devops@ibm.com",
7        "Space": "demo"
8      },
9      "Name": "simple-website-ae7f5ff6",
10     "Instrumentation": {}
11   },
12   "Path": "manifest.yml",
13   "Type": "Cloud Foundry"
14 }

```

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/cloudatabases-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 registry
is'registry.ng.bluemix.net'.ibmcloudcr build -t registry.ng.bluemix.net/<namespace>/icdpg
.ibmcloudcrimages

```

env:

```

    - name:

```

```

        BINDINGvalue

```

```

    From:

```

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:30081

