Assisted Practice: 4.2 Build a Custom Docker Image

This section will guide you to**:**

- Build a custom Docker image using a custom Dockerfile and deploy it on Docker host.

Prerequisites:

- Docker must be installed on your system.

This lab has three subsections, namely**:**

4.1.1 Clone Git repository
4.1.2 Docker Build
4.1.3 Pushing the code to GitHub repositories

Docker is already installed in your lab. (Refer FSD: Lab Guide - Phase 5)

Step 4.1.1: Clone Git repository

- First, clone the Git repository on a Docker host using the command below:

```
git clone https://github.com/Anuj1990/Docker.git
```

Step 4.1.2: Docker Build

- Then, proceed with the docker build command to build a custom docker image.

```
cd Docker
```

dockerbuild -t phpcode . -f Dockerfile

```
root@docker:~/Docker# docker build -t phpcode .
Sending build context to Docker daemon  337.9kB
Step 1/14 : FROM ubuntu
 ---> 93fd78260bd1
Step 2/14 : ENV DEBIAN_FRONTEND=non-interactive
 ---> Using cache
 ---> b21eb69f632a
Step 3/14 : RUN apt-get update -y
 ---> Using cache
 ---> d2e4866734b9
Step 4/14 : RUN apt-get install -y git curl apache2 php libapache2-mod-php php-mysql
 ---> Using cache
 ---> 85f084edfc0b
Step 5/14 : RUN rm -rf /var/www/html/*
 ---> Using cache
 ---> b56166da0f16
Step 6/14 : ADD src /var/www/html/
 ---> Using cache
 ---> ba9e5c5c651c
Step 7/14 : RUN a2enmod rewrite
 ---> Using cache
 ---> cff3e4bb8c42
Step 8/14 : RUN chown -R www-data:www-data /var/www/html
 ---> Using cache
 ---> 7a4314c7b69b
Step 9/14 : ENV APACHE_RUN_DIR /var/www/html
 ---> Using cache
 ---> 663a68663f90
```

- Once the image is built, check the image using docker run command and then run it to initialize custom container on Docker host.

  docker images

  docker run -d --name phpcode -p 80:80 phpcode

```
Step 10/14 : ENV APACHE_RUN_USER www-data
 ---> Using cache
 ---> 2915860f4df7
Step 11/14 : ENV APACHE_RUN_GROUP www-data
 ---> Using cache
 ---> c83847b098df
Step 12/14 : ENV APACHE_LOG_DIR /var/log/apache2
 ---> Using cache
 ---> c4428c1a14db
Step 13/14 : EXPOSE 80
 ---> Using cache
 ---> 7374debb3213
Step 14/14 : CMD ["/usr/sbin/apache2", "-D",  "FOREGROUND"]
 ---> Using cache
 ---> d35edda13537
Successfully built d35edda13537
Successfully tagged phpcode:latest
root@ip-172-31-25-208:~/Docker#
root@ip-172-31-25-208:~/Docker# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
phpcode             latest              d35edda13537        20 minutes ago      291MB
ubuntu              16.04               a51debf7e1eb        7 days ago          116MB
root@ip-172-31-25-208:~/Docker#
```

- Once the container is up and running, validate the connectivity using the **curl** command to see if **php code** is running on port 80 or not.

Step 4.1.3: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m “Changes have been committed.”

- Push the files to the folder you initially created using the following command:

  git push -u origin master

Assisted Practice: 4.4 Push an Image to Docker Hub

This section will guide you to**:**

- Build a custom Docker image using basic code of Python and push it to Docker Hub.

This lab has three subsections, namely**:**

4.2.1 Preparing a custom Docker image

4.2.2 Pushing the Docker image to Docker Hub

4.2.3 Push the code to GitHub repositories

- Docker is already installed in your lab. (Refer FSD: Lab Guide - Phase 5)

Step 4.2.1: Preparing a custom Docker image:

- Create a directory and write basic Python source code using the procedure given below:

```
mkdirdocker_python
        cddocker_python/
        vi Dockerfile
```

- Add the code given below to this Dockerfile

```
FROM python
        WORKDIR /app
        ADD . /app
        RUN pip install -r requirements.txt
        EXPOSE 80
        ENV NAME world
        CMD [“python”, “app.py”]
```

```
root@ip-172-31-17-73:~# mkdir docker_python
root@ip-172-31-17-73:~# cd docker_python/
root@ip-172-31-17-73:~/docker_python# vi Dockerfile
root@ip-172-31-17-73:~/docker_python# cat Dockerfile
FROM python
WORKDIR /app
ADD . /app
RUN pip install -r requirements.txt
EXPOSE 80
ENV NAME world
CMD ["python", "app.py"]
root@ip-172-31-17-73:~/docker_python#
```

- Create a Python app. Follow the steps below to create an app.py python file

  vi app.py

- Add the content below in app.py python source file

```
from flask import Flask
import os
import socket
app = Flask(__name__)@app.route("/")def hello():
    html = "<h3>Hello {name}!</h3>" \
           "<b>Hostname:</b> {hostname}<br/>"
    return html.format(name=os.getenv("NAME", "world"),
hostname=socket.gethostname())
if __name__ == "__main__":
app.run(host='0.0.0.0', port=80)
```

```
root@ip-172-31-17-73:~/docker_python# cat app.py
from flask import Flask
import os
import socket

app = Flask(__name__)

@app.route("/")

def hello():
        html = "<h3>Hello {name}!</h3>" \
               "<b>Hostname:</b> {hostname}<br/>"
        return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname())
if __name__ == "__main__":
app.run(host=.0.0.0.0', port=80)
root@ip-172-31-17-73:~/docker_python#
```

- Create a **requirements.txt** file with the content below

  vi requirements.txt

Flask

- You will get the file mentioned below for building a custom Docker image

```
root@ip-172-31-17-73:~/docker_python# ls -alrt
total 20
-rw-r--r--  1 root root  122 Jul 11 02:43 Dockerfile
-rw-r--r--  1 root root  348 Jul 11 02:48 app.py
-rw-r--r--  1 root root    6 Jul 11 02:49 requirements.txt
drwx------ 12 root root 4096 Jul 11 02:49 ..
drwxr-xr-x  2 root root 4096 Jul 11 02:49 .
root@ip-172-31-17-73:~/docker_python#
```

- Proceed with docker build command to build a custom Docker image

  dockerbuild -t docker_python . -f Dockerfile

```
root@ip-172-31-17-73:~/docker_python# docker build -t docker_python . -f Dockerfile
Sending build context to Docker daemon  4.096kB
Step 1/7 : FROM python
 ---> 4c0fd7901be8
Step 2/7 : WORKDIR /app
 ---> Using cache
 ---> 3e14f552d2a7
Step 3/7 : ADD . /app
 ---> Using cache
 ---> bb70b7abc0eb
Step 4/7 : RUN pip install -r requirements.txt
 ---> Using cache
 ---> 8a975e9d8faf
Step 5/7 : EXPOSE 80
 ---> Using cache
 ---> 71e2b3a4e69c
Step 6/7 : ENV NAME world
 ---> Using cache
 ---> 97135b098ad5
Step 7/7 : CMD ["python", "app.py"]
 ---> Using cache
 ---> 2ce510d91695
Successfully built 2ce510d91695
Successfully tagged docker_python:latest
root@ip-172-31-17-73:~/docker_python#
```

- Once the image is built, check the image using **docker run** command and run it to initialize the custom container on Docker host.

  docker images

  docker run -d --name docker_python-p 80:80 docker_python

```
root@ip-172-31-17-73:~/docker_python# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker_python       latest              a39ee4545a92        13 seconds ago      939MB
python              latest              4c0fd7901be8        2 days ago          929MB
root@ip-172-31-17-73:~/docker_python# docker run -d --name docker_python -p 80:80 docker_python
9e43a8fb15defe5c33463b31f403327225b956f7f4e8f3e4392c5df1e131b09d
root@ip-172-31-17-73:~/docker_python# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                   NAMES
9e43a8fb15de        docker_python       "python app.py"     2 seconds ago       Up 1 second         0.0.0.0:80->80/tcp      docker_python
root@ip-172-31-17-73:~/docker_python#
```

- Once the container is up and running, validate the connectivity using the **curl** command to see if Python code is running on port 80 or not.

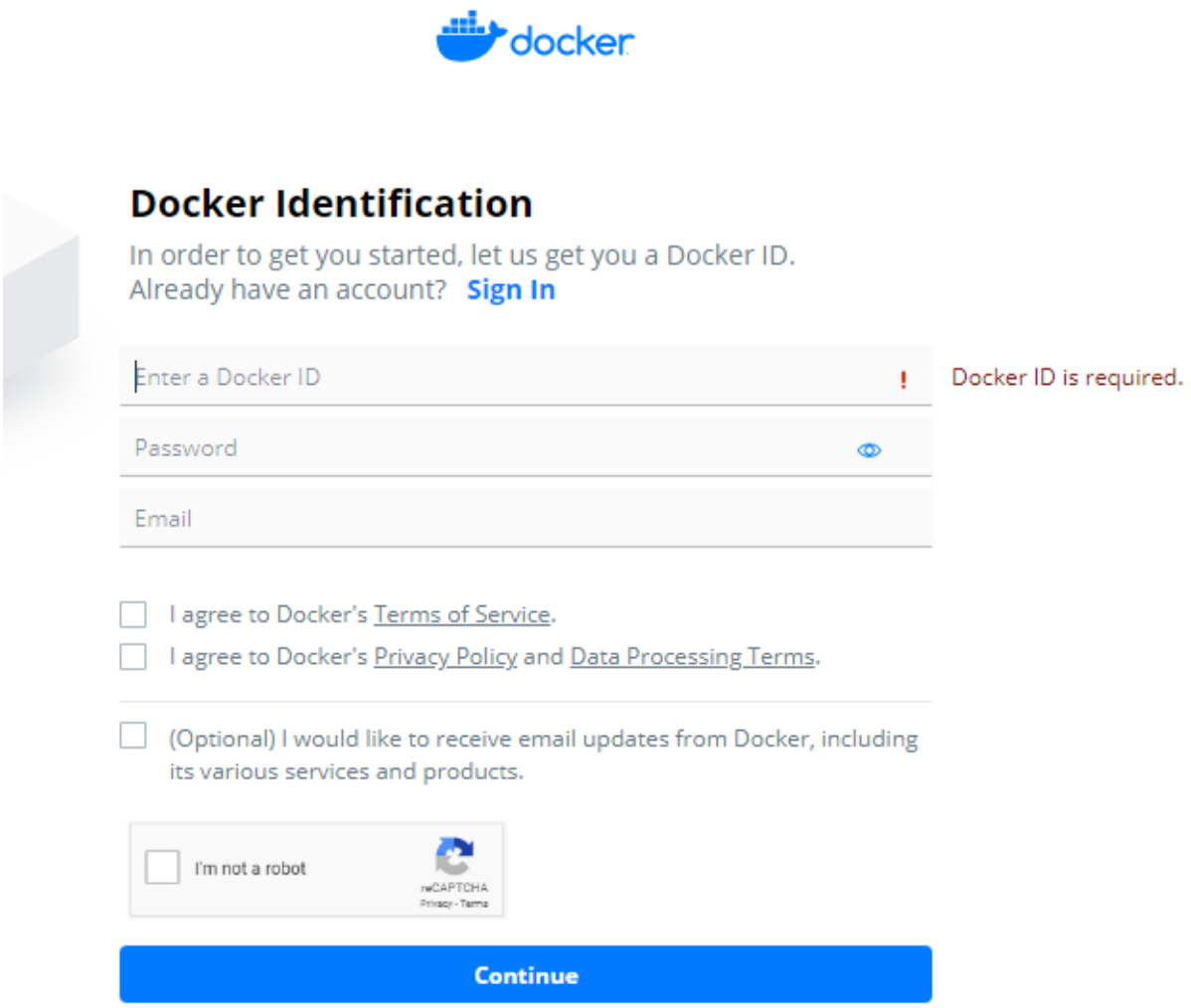
  curl localhost

```
root@ip-172-31-17-73:~/docker_python# curl localhost
<h3>Hello world!</h3><b>Hostname:</b> 9e43a8fb15de<br/>root@ip-172-31-17-73:~/docker_python#
root@ip-172-31-17-73:~/docker_python#
```

Step 4.2.2: Pushing Docker image to Docker Hub

- Once the Docker image is prepared, we need to push this custom Docker image to Docker Hub.

- For pushing the image to Docker Hub, create an account on Docker Hub. Follow simple sign up process to create a new account.



- Once the account is created, we need to login to Docker Hub to push the Docker image to Docker Hub.

docker login

```
root@ip-172-31-17-73:~/docker_python# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: anujsharma1990
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-172-31-17-73:~/docker_python#
```

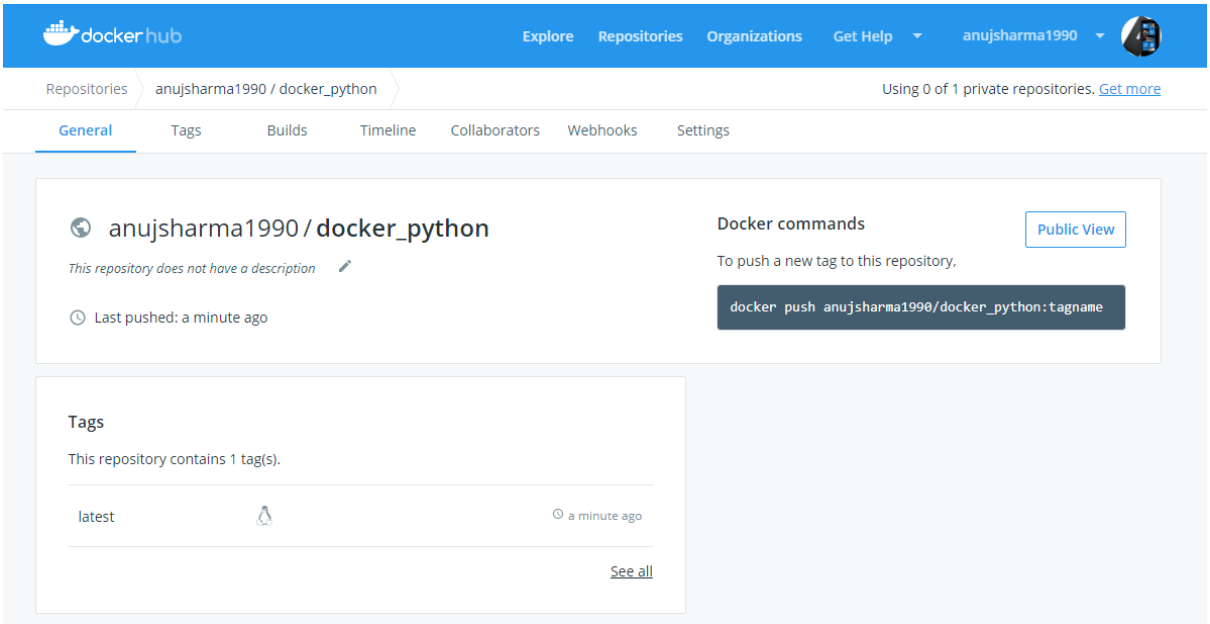
- Create a tag that can be used to push the custom image to Docker Hub.

docker tag docker_python anujsharma1990/docker_python

docker push anujsharma1990/docker_python

```
root@ip-172-31-17-73:~/docker_python# docker tag docker_python anujsharma1990/docker_python
root@ip-172-31-17-73:~/docker_python# docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
docker_python                 latest              a39ee4545a92        About an hour ago   939MB
anujsharma1990/docker_python  latest              a39ee4545a92        About an hour ago   939MB
python                        latest              4c0fd7901be8        2 days ago          929MB
root@ip-172-31-17-73:~/docker_python# docker push anujsharma1990/docker_python
The push refers to repository [docker.io/anujsharma1990/docker_python]
4ca3f54abbb7: Pushed
7ad993436b6a: Pushed
8514ba3cd21c: Pushed
fb32633979db: Mounted from library/python
a0595cebe89c: Mounted from library/python
a9ada1fd814d: Mounted from library/python
bb9c02680a15: Mounted from library/python
a637c551a0da: Mounted from library/python
2c8d31157b81: Mounted from library/python
7b76d801397d: Mounted from library/python
f32868cde90b: Mounted from library/python
0db06dff9d9a: Mounted from library/python
latest: digest: sha256:c448e76ad05db80de4f3dc4ff0e9b29176f330319e4deb14c311e6960d3d9e4a size: 2843
root@ip-172-31-17-73:~/docker_python#
```

- Shown below is the uploaded custom Docker image to Docker Hub.



Step 4.2.3: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m "Changes have been committed."

- Push the files to the folder you initially created using the following command:

  git push -u origin master

Assisted Practice: 4.5 Container Deployment Using Docker Swarm

This section will guide you to**:**

- Deploy a Docker container on Docker swarm for orchestration.

This lab has four subsections, namely**:**

4.3.1 Setting up a Docker instance
4.3.2 Building a custom Docker image to be deployed
4.3.3 Initializing a Docker swarm cluster and deploying a container to the cluster

4.3.4 Pushing the code to GitHub repositories

Step 4.3.1: Setting up a Docker instance

- Docker version 18.09.7 is installed in your practice lab. (Refer FSD: Lab Guide - Phase 5)
- Type the following command to check the docker version installed on lab:

  docker version

```
root@ip-172-31-17-73:~# apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
docker.io is already the newest version (18.09.7-0ubuntu1~18.04.3).
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
root@ip-172-31-17-73:~# docker version
Client:
 Version:           18.09.7
 API version:       1.39
 Go version:        go1.10.1
 Git commit:        2d0083d
 Built:             Wed Jul  3 12:13:59 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.09.7
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.1
  Git commit:       2d0083d
  Built:            Mon Jul  1 19:31:12 2019
  OS/Arch:          linux/amd64
  Experimental:     false
root@ip-172-31-17-73:~#
```

Step 4.3.2: Building a custom Docker image to be deployed

- First, clone the Git repository on Docker host using the command below:

  git clone https://github.com/Anuj1990/Docker.git

- Run with docker build command to build a custom Docker image

  cd Docker

docker build -t phpcode . -f Dockerfile

```
root@docker:~/Docker# docker build -t phpcode .
Sending build context to Docker daemon  337.9kB
Step 1/14 : FROM ubuntu
 ---> 93fd78260bd1
Step 2/14 : ENV DEBIAN_FRONTEND=non-interactive
 ---> Using cache
 ---> b21eb69f632a
Step 3/14 : RUN apt-get update -y
 ---> Using cache
 ---> d2e4866734b9
Step 4/14 : RUN apt-get install -y git curl apache2 php libapache2-mod-php php-mysql
 ---> Using cache
 ---> 85f084edfc0b
Step 5/14 : RUN rm -rf /var/www/html/*
 ---> Using cache
 ---> b56166da0f16
Step 6/14 : ADD src /var/www/html/
 ---> Using cache
 ---> ba9e5c5c651c
Step 7/14 : RUN a2enmod rewrite
 ---> Using cache
 ---> cff3e4bb8c42
Step 8/14 : RUN chown -R www-data:www-data /var/www/html
 ---> Using cache
 ---> 7a4314c7b69b
Step 9/14 : ENV APACHE_RUN_DIR /var/www/html
 ---> Using cache
 ---> 663a68663f90
```

- Once the image is built, check if it is built properly or not. You can see a Docker image entry using Docker images command

```
Removing intermediate container 66720df3cf7e
 ---> b914fd976a06
Successfully built b914fd976a06
Successfully tagged phpcode:latest
root@ip-172-31-17-73:~/Docker# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
phpcode             latest              b914fd976a06        3 minutes ago       251MB
ubuntu              latest              4c108a37151f        2 weeks ago         64.2MB
root@ip-172-31-17-73:~/Docker#
```

Step 4.3.3: Initializing a Docker swarm cluster and deploying a container to the cluster

- First, we need to initialize Docker swarm using the set of commands given below:

docker swarm init

docker node ls

```
root@ip-172-31-17-73:~# docker swarm init
Swarm initialized: current node (rv82bet81vwyaic3r18y3mu0n) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3c53o3x6p1rny8f8i89lp34s8lm6lpd2uy7shn3excvk1d8a4y-1peyvlf7d4y02y253d5cqaarh 172.31.17.73:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@ip-172-31-17-73:~# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
rv82bet81vwyaic3r18y3mu0n *   ip-172-31-17-73     Ready               Active              Leader              18.09.7
root@ip-172-31-17-73:~#
```

- Once the node is configured, deploy the custom Docker image on the Docker swarm cluster following the process shown below

docker service create -p 80:80 --name webserver phpcode

docker service ls

curl localhost

```
root@ip-172-31-17-73:~# docker service create -p 80:80 --name webserver phpcode
image phpcode:latest could not be accessed on a registry to record
its digest. Each node will access phpcode:latest independently,
possibly leading to different nodes running different
versions of the image.

p1sows6zdl801fr36ld9533uv
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
root@ip-172-31-17-73:~# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
p1sows6zdl80        webserver           replicated          1/1                 phpcode:latest      *:80->80/tcp
root@ip-172-31-17-73:~# curl localhost
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <title>Simple PHP App</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link href="assets/css/bootstrap.min.css" rel="stylesheet">
        <style>body {margin-top: 40px; background-color: #333;}</style>
        <link href="assets/css/bootstrap-responsive.min.css" rel="stylesheet">
        <!--[if lt IE 9]><script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script><![endif]-->
    </head>

    <body>
        <div class="container">
            <div class="hero-unit">
                <h1>Simple PHP App</h1>
                <h2>Congratulations</h2>
                <p>Your PHP application is now running on a container in Amazon ECS.</p>
                <p>The Kubernetes Docker container is running PHP version 7.2.19-0ubuntu0.18.04.1.</p>
                root@ip-172-31-17-73:~#
```

Step 4.3.4: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m “Changes have been committed.”

- Push the files to the folder you initially created using the following command:

  git push -u origin master

Assisted Practice: 4.6 Container Scaling with Docker Swarm

This section will guide you to**:**

- Scale Docker containers on Docker swarm.

This lab has two subsections, namely**:**

4.4.1 Creating service for scaling

4.4.2 Pushing the code to GitHub repositories

- Docker is already installed in your lab. (Refer FSD: Lab Guide - Phase 5)

Step 4.4.1: Creating service for scaling

Please Note: Docker containers deployed on Docker swarm cluster can be scaled up and down to implement high availability of Docker containers. If in case any Docker container gets crashed, we can get a new one created and other containers can easily handle the load.

- Use the commands below to create a service and scale the service up and down to increase or decrease Docker containers

```
docker service create -p 8080:8080 --name bootcampjocatalin/kubernetes-bootcamp:v1

docker service ls

docker service psbootcamp

curl localhost:8080
```

```
root@ip-172-31-17-73:~# docker service create -p 8080:8080 --name bootcamp jocatalin/kubernetes-bootcamp:v1
pklb5vzestjjg0ejhbhzas3vi
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
root@ip-172-31-17-73:~# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE
pklb5vzestjj        bootcamp            replicated          1/1                 jocatalin/kubernetes-bootcamp:v1
p1sows6zdl80        webserver           replicated          1/1                 phpcode:latest
root@ip-172-31-17-73:~# docker service ps bootcamp
ID                  NAME                IMAGE                               NODE                DESIRED STATE
70jbzbm9mgll        bootcamp.1          jocatalin/kubernetes-bootcamp:v1    ip-172-31-17-73     Running
root@ip-172-31-17-73:~# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: 453e2d4bf870 | v=1
root@ip-172-31-17-73:~# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: 453e2d4bf870 | v=1
root@ip-172-31-17-73:~#
```

- Once the container deployed, we can scale up and down the Docker swarm service following the process shown below

docker service scale bootcamp=3

docker service psbootcamp

curl localhost:8080

```
root@ip-172-31-17-73:~# docker service scale bootcamp=3
bootcamp scaled to 3
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
root@ip-172-31-17-73:~# docker service ps bootcamp
ID                  NAME                IMAGE                               NODE                DESIRED STATE
70jbzbm9mgll        bootcamp.1          jocatalin/kubernetes-bootcamp:v1    ip-172-31-17-73     Running
j9ijttsqiwpc        bootcamp.2          jocatalin/kubernetes-bootcamp:v1    ip-172-31-17-73     Running
0nu6c719dmtl        bootcamp.3          jocatalin/kubernetes-bootcamp:v1    ip-172-31-17-73     Running
root@ip-172-31-17-73:~# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: 2899104e3c94 | v=1
root@ip-172-31-17-73:~# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: 8c8d2c6a855a | v=1
root@ip-172-31-17-73:~# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: 453e2d4bf870 | v=1
```

Please Note: In the screenshot above, we can see that when we are trying to access swarm service on 8080 port, we are getting different ids in the response. This means that our request is going to different containers in round robin manner.

Step 4.4.2: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m “Changes have been committed.”

- Push the files to the folder you initially created using the following command:

  git push -u origin master

Assisted Practice: 4.7 Distribute Your App Across a Swarm Cluster

This section will guide you to**:**

- Deploy a Docker container on Docker swarm and have multiple nodes inside a cluster.

This lab has four subsections, namely**:**

4.5.1 Setting up a Docker instance
4.5.2 Setting up a Docker swarm with multiple nodes
4.5.3 Deploying a custom Docker image to a Docker swarm cluster

4.5.4 Pushing the code to GitHub repositories

Step 4.5.1: Setting up a Docker instance

- Docker version 18.09.7 is installed in your practice lab. (Refer FSD: lab Guide - Phase 5)
- To verify the installation:
    1. Open the command-line interface
    2. Type in the command:

        docker --version

```
root@ip-172-31-17-73:~# apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
docker.io is already the newest version (18.09.7-0ubuntu1~18.04.3).
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
root@ip-172-31-17-73:~# docker version
Client:
 Version:           18.09.7
 API version:       1.39
 Go version:        go1.10.1
 Git commit:        2d0083d
 Built:             Wed Jul  3 12:13:59 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.09.7
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.1
  Git commit:       2d0083d
  Built:            Mon Jul  1 19:31:12 2019
  OS/Arch:          linux/amd64
  Experimental:     false
root@ip-172-31-17-73:~#
```

Step 4.5.2: Setting up Docker swarm with multiple nodes

- Edit the **/etc/hosts** file across the two nodes via **gedit** or **vim** and make the following changes:

172.31.17.73dockermanager

172.31.86.69dockerworker1

- After modifying the host file with the details mentioned above, check the connectivity with **ping** between all the nodes
  - From Docker Manager Host instance:

```
root@ip-172-31-17-73:~# ping dockerworker1
PING dockerworker1 (172.31.86.69) 56(84) bytes of data.
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=1 ttl=64 time=0.637 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=2 ttl=64 time=0.727 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=3 ttl=64 time=0.673 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=4 ttl=64 time=5.00 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=5 ttl=64 time=0.674 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=6 ttl=64 time=0.647 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=7 ttl=64 time=0.751 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=8 ttl=64 time=0.663 ms
^C
--- dockerworker1 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7136ms
rtt min/avg/max/mdev = 0.637/1.222/5.005/1.430 ms
root@ip-172-31-17-73:~#
```

- From Docker Worker Node instance:

```
root@ip-172-31-86-69:~# ping dockermanager
PING dockermanager (172.31.17.73) 56(84) bytes of data.
64 bytes from dockermanager (172.31.17.73): icmp_seq=1 ttl=64 time=0.669 ms
64 bytes from dockermanager (172.31.17.73): icmp_seq=2 ttl=64 time=0.693 ms
64 bytes from dockermanager (172.31.17.73): icmp_seq=3 ttl=64 time=0.693 ms
64 bytes from dockermanager (172.31.17.73): icmp_seq=4 ttl=64 time=0.713 ms
64 bytes from dockermanager (172.31.17.73): icmp_seq=5 ttl=64 time=0.697 ms
^C
--- dockermanager ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4100ms
rtt min/avg/max/mdev = 0.669/0.693/0.713/0.014 ms
root@ip-172-31-86-69:~#
```

- Initialize the Docker swarm mode by running the following docker command on the **dockermanager** node

docker swarm init --advertise-addr<manager node IP address>

docker swarm init --advertise-addr172.31.17.73

```
root@ip-172-31-17-73:~# docker swarm init --advertise-addr 172.31.17.73
Swarm initialized: current node (ba8j0ti2lols6f8pbxfyqy5lc) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-2o9yesj2p0jk65wory232wthdrec38yeg1r037ryoxe6duuy4n-ant41o3e6xkdociyk9ut5ky4j 172.31.17.73:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@ip-172-31-17-73:~#
```

- Once the swarm cluster is initialized, allow the ports mentioned below in security groups

SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
All ICMP - IPv	ICMP	0 - 65535	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP F	TCP	2377	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop

- While initializing the Docker swarm cluster, you will get docker swarm join command which can be executed on node manager to add node to swarm

```
root@ip-172-31-86-69:~# docker swarm join --token SWMTKN-1-2o9yesj2p0jk65wory232wthdrec38yeg1r037ryoxe6duuy4n-ant41o3e6xkdociyk9ut5ky4j 172.31.17.73:2377
This node joined a swarm as a worker.
root@ip-172-31-86-69:~#
```

cluster

- Run the command below to see the node status

docker node ls

```
root@ip-172-31-17-73:~# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
ba8j0ti2lols6f8pbxfyqy5lc *   ip-172-31-17-73     Ready               Active              Leader              18.09.7
sk9btki9xk5gr39jj5j7kdztz     ip-172-31-86-69     Ready               Active                                  18.09.7
root@ip-172-31-17-73:~#
```

Step 4.5.3: Deploying a custom Docker image to a Docker swarm cluster

- Create service in Docker swarm cluster

docker service create --name webapp --publish 8080:8080 --replicas 2 jocatalin/kubernetes-bootcamp:v1

```
root@ip-172-31-17-73:~# docker service create --name webapp --publish 8080:8080 --replicas 2 docker.io/jocatalin/kubernetes-bootcamp:v1
oh0u8450pritrl1hwicjgbqjy
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
root@ip-172-31-17-73:~# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE                               PORTS
oh0u8450prit        webapp              replicated          2/2                 jocatalin/kubernetes-bootcamp:v1    *:8080->8080/tcp
root@ip-172-31-17-73:~#
```

- You can now validate if Docker containers got deployed on both nodes or not using the command below

docker service pswebapp

```
root@ip-172-31-17-73:~# docker service ps webapp
ID                  NAME                IMAGE                               NODE                DESIRED STATE
S
kxlfdaa25vol        webapp.1            jocatalin/kubernetes-bootcamp:v1    ip-172-31-17-73     Running
wouv28ypnnje        webapp.2            jocatalin/kubernetes-bootcamp:v1    ip-172-31-86-69     Running
root@ip-172-31-17-73:~#
```

Please Note: We can validate the application using the **curl** command to see if the application is up and running.

```
root@ip-172-31-17-73:~# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: dda6e7f30789 | v=1
root@ip-172-31-17-73:~#
```

Step 4.5.4: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m "Changes have been committed."

- Push the files to the folder you initially created using the following command:

git push -u origin master

Assisted Practice: 4.8 Setting Up Jenkins Pipeline

This section will guide you to**:**

- Build a Docker Jenkins pipeline to implement CI/CD workflow.

  This lab has three subsections, namely**:**

  4.6.1 Installing Docker plugin and configuring Docker cloud

  4.6.2 Configuring Jenkins job

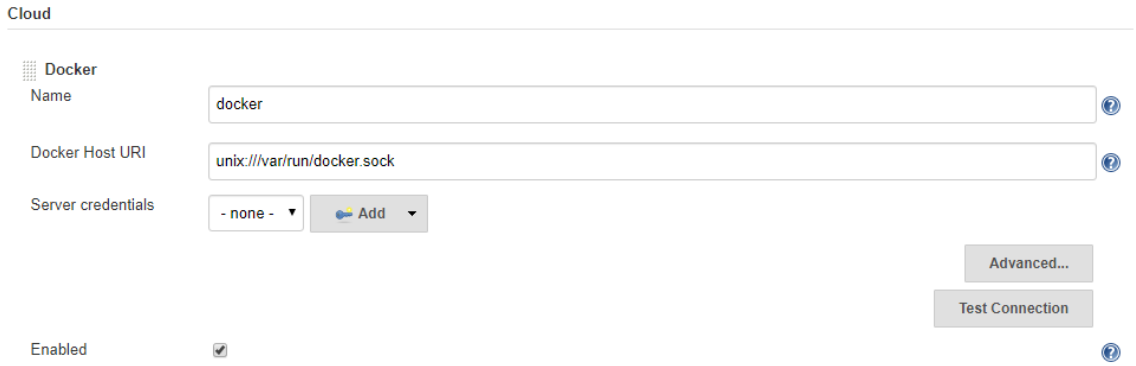
  4.6.3 Pushing the code to GitHub repositories

- Docker is already installed in your lab. (Refer FSD: Lab Guide - Phase 5)

Step 4.6.1: Installing Docker plugin and configuring Docker cloud

- Add Docker cloud by accessing Manage Jenkins Configure system. Then, you have to add Docker cloud details as shown below:

Docker Host URI: unix:///var/run/docker.sock



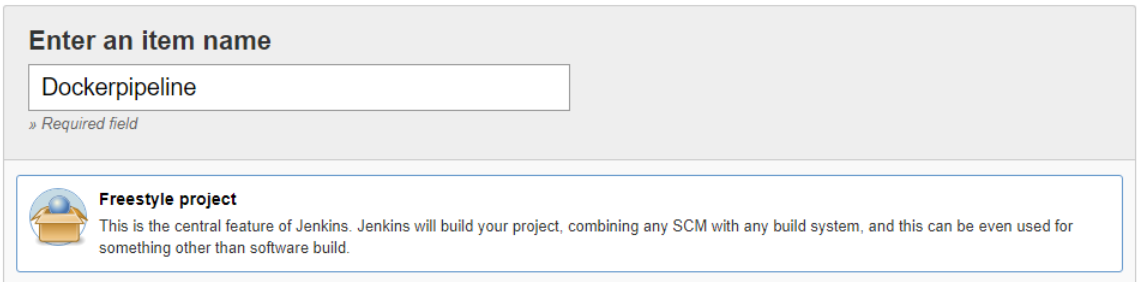
- Configure the Docker cloud to give complete access to docker.sock file so that Jenkins will be able to connect to Docker process

chmod 777 /var/run/docker.sock

```
root@ip-172-31-17-73:/var/run# chmod 777 docker.sock
```

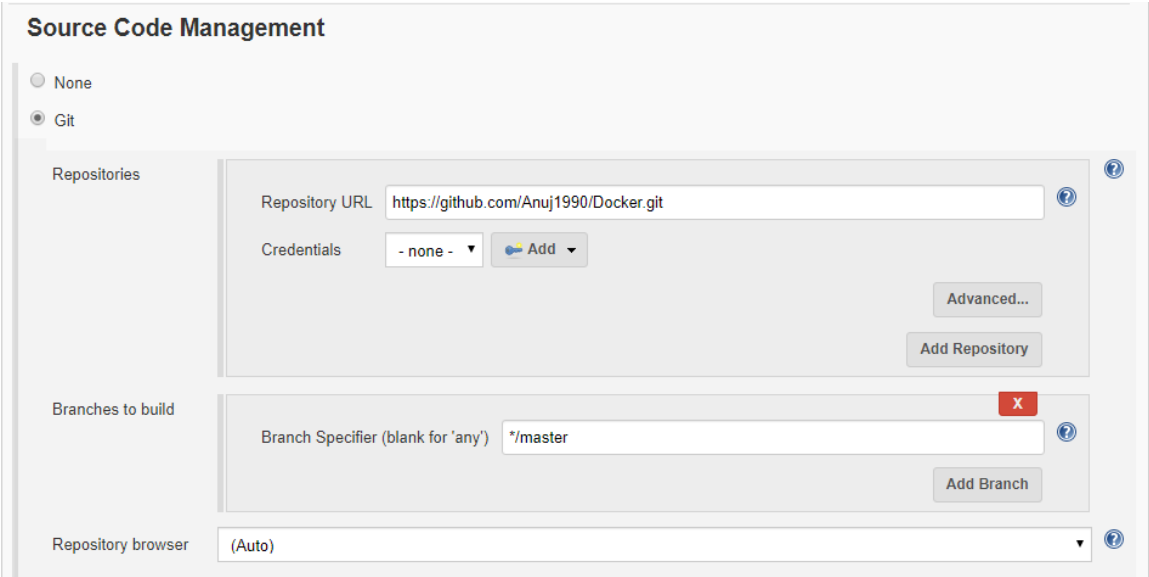
Step 4.6.2: Configuring Jenkins job

- Create a new Jenkins pipeline job for supporting CI/CD workflow

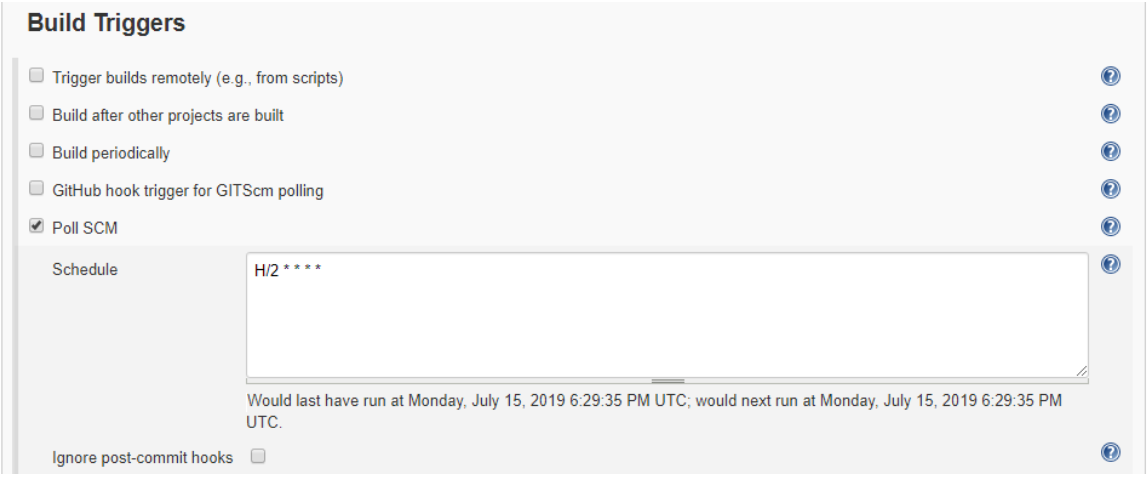


- Configure Git repository so that we can have Dockerfile to build Docker container and push it to Docker Hub

https://github.com/Anuj1990/Docker.git



- Configure build triggers to enable Poll SCM feature so that once any push is detected



- Configure Docker build option to configure build configurations. Some of the configurations are mentioned below:

  Directory for Dockerfile . (Represents current location)

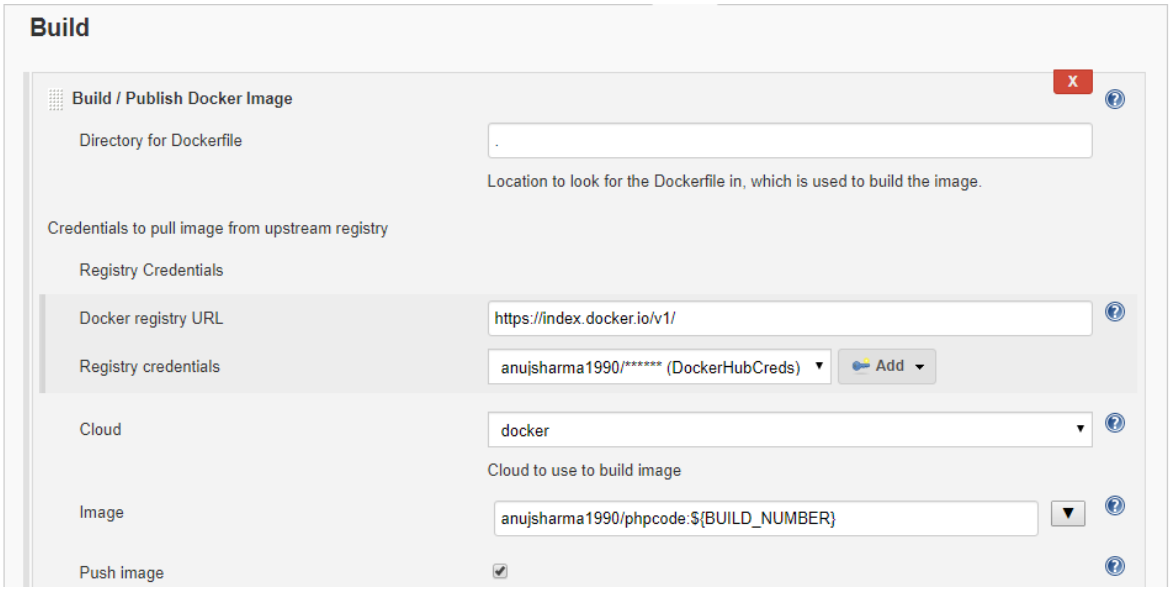
  Docker Registry URL
  https://index.docker.io/v1/

  Docker credentials Docker hub username
password

  Cloud Select Docker Cloud created in dropdown

  Image
  anujsharma1990/phpcode:${BUILD_NUMBER}

  Registry Credentials Docker hub username
password

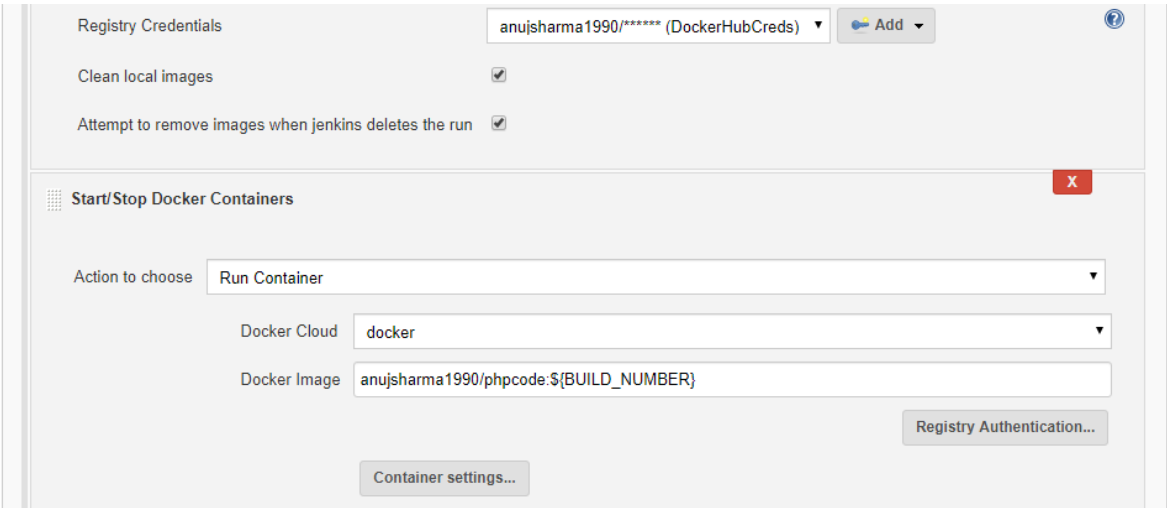


Please Note: We have configured how our custom Docker image will be pushed to Docker Hub. Now, we have to configure the Docker container deployment.

- Follow the steps mentioned below to configure how Docker containers will be deployed

Docker cloud Select from dropdown Docker cloud details

Docker image anujsharma1990/phpcode:${BUILD_NUMBER}



- Once job configuration is done, save the configuration and proceed with triggering build in order to build custom container and deploy the container

```
Successfully built 57c0eeb63850

Tagging built image with anujsharma1990/phpcode:4
Docker Build Response : 57c0eeb63850
Pushing [anujsharma1990/phpcode:4]
The push refers to repository [docker.io/anujsharma1990/phpcode]
```

```
4: digest: sha256:4a9404ab7b26b05fdcd0aee10538a43667fc2b9d0834d7dffb4d330356afd106 size: 2408
Cleaning local images [57c0eeb63850]
Docker Build Done
Pulling image anujsharma1990/phpcode:4
4:Pulling from anujsharma1990/phpcode5b7339215d1d:Already exists14ca88e9f672:Already existsa31c3b1caad4:Already
existsb054a26005b7:Already existsd4db4c3dd692:Pulling fs layer42cbd6016189:Pulling fs layer58e44124d930:Pulling fs
layercb727fa74bc1:Pulling fs layer51e9e9911579:Pulling fs layeree860aead8f8:Pulling fs
layer51e9e9911579:Waitingee860aead8f8:Waitingcb727fa74bc1:Waiting58e44124d930:Verifying Checksum58e44124d930:Download
completecb727fa74bc1:Verifying Checksumcb727fa74bc1:Download complete51e9e9911579:Verifying Checksum51e9e9911579:Download
completed4db4c3dd692:Verifying Checksumd4db4c3dd692:Download completeee860aead8f8:Verifying Checksumee860aead8f8:Download
completed4db4c3dd692:Pull complete42cbd6016189:Verifying Checksum42cbd6016189:Download complete42cbd6016189:Pull
complete58e44124d930:Pull completecb727fa74bc1:Pull complete51e9e9911579:Pull completeee860aead8f8:Pull completenull:Digest:
sha256:4a9404ab7b26b05fdcd0aee10538a43667fc2b9d0834d7dffb4d330356afd106null:Status: Downloaded newer image for
anujsharma1990/phpcode:4Starting container for image anujsharma1990/phpcode:4
Started container aec3ed34f43c7d14e1166aed67b99e9f211aed2fe105b10ee5604c046522e8cc
Finished: SUCCESS
```

- Once the build is successful, validate the Docker container deployment on Docker host which will help us to implement complete CI/CD workflow for Docker container

```
root@ip-172-31-17-73:/var/run# docker ps
CONTAINER ID        IMAGE                       COMMAND                  CREATED
aec3ed34f43c        anujsharma1990/phpcode:4    "/usr/sbin/apache2 -â¦"   44 seconds ago
root@ip-172-31-17-73:/var/run#
```

Step 4.6.3: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize your repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m "Changes have been committed."

- Push the files to the folder you initially created using the following command:

  git push -u origin master