

Deploying a Multi-Tier Application on Docker Swarm

Project Overview:-

- Docker Swarm orchestrates microservices of a multi-tier app across the cluster, ensuring efficient resource usage and fault isolation.
- Automated deployment and rolling updates streamline operations, minimizing downtime and enhancing application resilience and scalability.
- Built-in service discovery and load balancing enable seamless communication and optimal traffic distribution between application tiers.

Services used:-

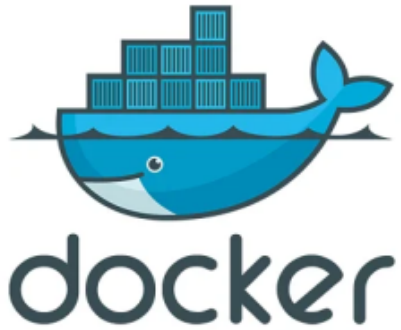
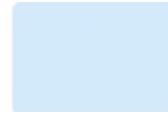
--> Docker, AWS cloud.

docker

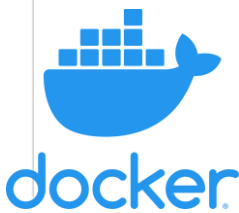




Architecture Diagram:-



**Amazon
EC2**



Explanation:-

Step #1 :- Launching the EC2-server on AWS Cloud

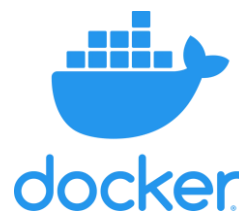
- Open the AWS Management console
- Launch the two (2) EC2 server (t2.micro) for free tier eligibility..
- Modify the security groups for the two EC2-server to allow the all the traffic ..
 - HTTP (80).
 - 2377 port for docker communication (Manager node and Worker node).
 - 8000 (For communicating a docker application through the local system).
- Connect the EC2 server with the putty through the public ip-address of the server
- Update the both server before installing docker on both servers

<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-0538fa14c481e926c	IPv4	Custom TCP	TCP	2377
<input type="checkbox"/>	-	sgr-02a1cf68b89950cf6	IPv4	SSH	TCP	22
<input type="checkbox"/>	-	sgr-0fb6e6a116f8b2fe5	IPv4	Custom TCP	TCP	8000
<input type="checkbox"/>	-	sgr-0b0d90337e8297...	IPv4	HTTP	TCP	80



docker

```
ubuntu@ip-172-31-84-174:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1413 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [277 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1492 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [245 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1050 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [237 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [22.1 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [42.1 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [10.1 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [472 B]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [41.7 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [10.5 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [388 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [24.3 kB]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.5 kB]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [644 B]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:29 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1194 kB]
Get:30 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [217 kB]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1456 kB]
Get:32 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [239 kB]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [845 kB]
Get:34 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [161 kB]
Get:35 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [16.8 kB]
Get:36 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [37.1 kB]
Get:37 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7476 B]
Get:38 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [260 B]
Fetched 29.8 MB in 5s (5449 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-84-174:~$
```



Step#2:- Installing the Docker on the EC2-server

- Run this command on both servers
-- sudo apt-get install docker.io

This will install all the software of the docker into the EC2 server

```
ubuntu@ip-172-31-84-174:~$ sudo apt-get install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 77 not upgraded.
Need to get 69.8 MB of archives.
After this operation, 267 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [35.2 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.7-0ubuntu1~22.04.2 [11.1 MB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.2-0ubuntu1~22.04.1 [17.5 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dns-root-data all 2023112702~ubuntu0.22.04.1 [10.1 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.90-0ubuntu0.22.04.1 [11.1 MB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.5-0ubuntu1~22.04.1 [35.2 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [10.1 kB]
Fetched 69.8 MB in 2s (44.2 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 64799 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.6-1_amd64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7-1ubuntu3_amd64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.1.7-0ubuntu1~22.04.2_amd64.deb ...
Unpacking runc (1.1.7-0ubuntu1~22.04.2) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.7.2-0ubuntu1~22.04.1_amd64.deb ...
Unpacking containerd (1.7.2-0ubuntu1~22.04.1) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../4-dns-root-data_2023112702~ubuntu0.22.04.1_all.deb ...
Unpacking dns-root-data (2023112702~ubuntu0.22.04.1) ...
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../5-dnsmasq-base_2.90-0ubuntu0.22.04.1_amd64.deb ...
Unpacking dnsmasq-base (2.90-0ubuntu0.22.04.1) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../6-docker.io_24.0.5-0ubuntu1~22.04.1_amd64.deb ...
Unpacking docker.io (24.0.5-0ubuntu1~22.04.1) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../7-ubuntu-fan_0.12.16_all.deb ...
Unpacking ubuntu-fan (0.12.16) ...
Setting up dnsmasq-base (2.90-0ubuntu0.22.04.1) ...
```



Step#3:- Importing a repository from public docker hub

- Login in to the docker hub account
- Open terminal of the server and perform this command
 - docker pull hshar/webapp
 - docker pull hshar/mysql:5.6

```
ubuntu@ip-172-31-84-174:~$ sudo docker pull hshar/webapp
Using default tag: latest
latest: Pulling from hshar/webapp
a48c500ed24e: Pull complete
1e1de00ff7e1: Pull complete
0330ca45a200: Pull complete
471db38bcfbf: Pull complete
0b4aba487617: Pull complete
c2e32ec79cfd: Pull complete
a18d6ba75273: Pull complete
4c2cc0ff3ce8: Pull complete
Digest: sha256:3c7cbcab1a26c01410dcc9cbc57252b50d9ed2f31a2dc24e3f066c61b88e839b
Status: Downloaded newer image for hshar/webapp:latest
docker.io/hshar/webapp:latest
ubuntu@ip-172-31-84-174:~$
```

```
ubuntu@ip-172-31-84-174:~$ sudo docker pull hshar/mysql:5.6
5.6: Pulling from hshar/mysql
5e6ec7f28fb7: Pull complete
4140e62498e1: Pull complete
e7bc612618a0: Pull complete
1af808cf1124: Pull complete
ff72a74ebb66: Pull complete
852cfe5dca55: Pull complete
e27e60fa86d5: Pull complete
ab7c1c7d8dd6: Pull complete
bb9fcaf41441: Pull complete
0c4bda3739a6: Pull complete
e22ee1bc1b20: Pull complete
bd82affc86f8: Pull complete
Digest: sha256:f0c9455c7406134380d7c73ba022a63bfc7499aa514b3b626532434a1c468aa6
Status: Downloaded newer image for hshar/mysql:5.6
docker.io/hshar/mysql:5.6
ubuntu@ip-172-31-84-174:~$
```



Step#4:- Creating a docker Swarm Cluster

- In the terminal of the server perform this command

→ `sudo docker swarm init --advertise-addr=<ip-address of the 1st server>`

```
ubuntu@ip-172-31-84-174:~$ sudo docker swarm init --advertise-addr=3.84.134.22
Swarm initialized: current node (5s3pbd5i9mu9whu3sjhmz4rx) is now a manager.

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

docker swarm join --token SWMTKN-1-21xwagkkg7v84vrzbl0sexfpu4qiqveh9hlz2sfxki5diwnk7-61oxvtvi9hpke76a6aiol5uq 3.84.134.22:2377

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

ubuntu@ip-172-31-84-174:~$
```

- Here it provides the SWARM join TOKEN
---- Which is used to join the slave node to the worker node (Manager Node)

Paste the Swarm join in the another server to make it as a worker node



Step#5:- Adding the worker node to the Manager node

- After pasting the Swarm TOKEN in the 2nd server

```
ubuntu@ip-172-31-35-210: ~  
ubuntu@ip-172-31-35-210:~$ sudo docker swarm join --token SWMTKN-1-4hjaaujiioh92nlypg6detikw  
2tjtqd3epvgs43hg6xstunlk1-c7m6jt8fbvolg1rp605ck2pac 3.84.134.22:2377  
This node joined a swarm as a worker.  
ubuntu@ip-172-31-35-210:~$
```

- This one shows the node is successfully joined as worker node.....



Step#6:- Creating a overlay network on docker

- Creating a docker overlay network named demo-overlay
- With the help of a command
→ `sudo docker create --driver overlay demo-overlay`

```
root@ip-172-31-84-174: /home/ubuntu
root@ip-172-31-84-174:/home/ubuntu# docker network create --driver overlay demo-overlay
w0i9jyht4xyceibxkyej0y7fo
root@ip-172-31-84-174:/home/ubuntu# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
a17a8fee41c8        bridge             bridge              local
w0i9jyht4xyc        demo-overlay       overlay             swarm
610d83e83f07        docker_gwbridge    bridge              local
6fdf83336d88        host               host                local
0lg8atbejxgh        ingress            overlay             swarm
0423dbdd5edd        none               null                local
root@ip-172-31-84-174:/home/ubuntu#
```

- We can also see the entire networks list on the docker ...
--- by using this command
→ `sudo docker network ls`



Step#7:- Creating a Service on Docker Swarm

- To create a service on docker swarmuse this comamd.

→ `docker service create --name website --replicas 2 --network demo-overlay -p 8000:80 hshar/webapp`

- Here this command indicated that

1. Importing a hshar/webapp from the docker hub.
 2. To connect the docker application it is using the system port(8000) to connect to the application on docker.
- Creating this services on the demo-overlay network (Which was created by the use).

 root@ip-172-31-84-174: /home/ubuntu

```
root@ip-172-31-84-174:/home/ubuntu# docker service create --name website --replicas 2 --network demo-overlay -p 8000:80 hshar/webapp
bnvw86r0zbhrhg5kc5c41gg3g
overall progress: 2 out of 2 tasks
1/2: running [=====>]
2/2: running [=====>]
verify: Service converged
root@ip-172-31-84-174:/home/ubuntu#
```



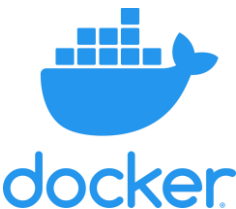
Step#8:- Installing the MYSQL server on the EC2-server

- Install the mysql server on the EC2-server.

→ `sudo apt-get install mysql-server`

- This command install's all the packages of MYSQL-Server.

```
root@ip-172-31-84-174: /home/ubuntu
root@ip-172-31-84-174: /home/ubuntu# sudo apt-get install mysql-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcgi-fast-perl libcgi-pm-perl libclone-perl libencode-locale-perl libevent-pthreads-2.1-7 libfcgi-bin libfcgi-perl libfcgi0ldbl libhtml-parser-perl libhtml-tagset-perl
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libmecab2 libprotobuf-lite23 libtimedate-perl liburi-perl mecab-ipadic
  mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server-8.0 mysql-server-core-8.0
Suggested packages:
  libdata-dump-perl libipc-sharedcache-perl libbusiness-isbn-perl libwww-perl mailx tinyca
The following NEW packages will be installed:
  libcgi-fast-perl libcgi-pm-perl libclone-perl libencode-locale-perl libevent-pthreads-2.1-7 libfcgi-bin libfcgi-perl libfcgi0ldbl libhtml-parser-perl libhtml-tagset-perl
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libmecab2 libprotobuf-lite23 libtimedate-perl liburi-perl mecab-ipadic
  mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server mysql-server-8.0 mysql-server-core-8.0
0 upgraded, 28 newly installed, 0 to remove and 77 not upgraded.
Need to get 29.5 MB of archives.
After this operation, 243 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 mysql-common all 5.8+1.0.8 [7212 B]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-client-core-8.0 amd64 8.0.36-0ubuntu0.22.04.1 [2692 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-client-8.0 amd64 8.0.36-0ubuntu0.22.04.1 [22.7 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libevent-pthreads-2.1-7 amd64 2.1.12-stable-1build3 [7642 B]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libmecab2 amd64 0.996-14build9 [199 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libprotobuf-lite23 amd64 3.12.4-1ubuntu7.22.04.1 [209 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-server-core-8.0 amd64 8.0.36-0ubuntu0.22.04.1 [17.5 MB]
```



Step#9:- Creating a DataBase on the Docker swarm

- To create a data-base on the docker ...perform this command to create a mysql database on docker.
→ `sudo docker service create --name db --network demo-overlay hshar/mysql:5.6`
1. Importing a mysql:5.6 from docker hub of (hshar/mysql:5.6)
 2. Here this creates a database named "db".
 3. On the demo-overlay network.

```
root@ip-172-31-84-174:/home/ubuntu# docker service create --name db --network demo-overlay hshar/mysql:5.6
d0xfezfspemt2jjuz5bdhoa7x
overall progress: 1 out of 1 tasks
1/1: running [=====>]
verify: Service converged
root@ip-172-31-84-174:/home/ubuntu#
```



Step#10:- Running the services on DOCKER SWARM

1. List out the all the services in the docker
→ `sudo docker service ls`
2. To see all the running container on the docker

→ `sudo docker ps`
3. Copy the container id of the hshar/mysql
→ `sudo docker exec -it <container-id> bash`

After performing this command we entered into the container..

4. Now perform `mysql -u root -p` to enter into the database ..

```
root@ip-172-31-84-174:/home/ubuntu# docker exec -it b68a0b2da45b bash
root@b68a0b2da45b:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.62 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```



Step#11:- Creating a MYSQL database

1. After executing the `mysql -u root -p`
2. Enter the password :- intelli
3. To display all the databases

```
mysql> show databases;
+-----+
| Database                |
+-----+
| information_schema      |
| mysql                   |
| performance_schema      |
+-----+
3 rows in set (0.00 sec)
```

4. Creating a database named **docker**

```
mysql> create database docker;
Query OK, 1 row affected (0.00 sec)
```

5. After performing the pervious step perform the command to use the docker database (created by the user)
→ `use docker;`
6. Create a table name emp
→ `create a table emp (name varchar(20) , phone number int);`



Step#12:-

1. Copy the ip-address of the 1st server and paste it on the browser

→ <ip-address>:8000

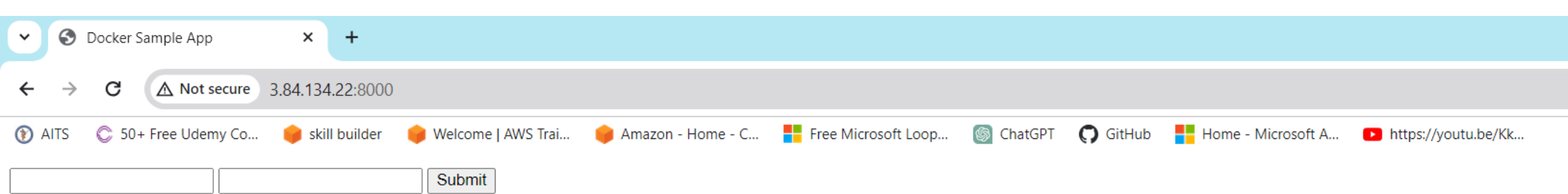
2. Here 8000 is the port number of the system to connect to the docker application.

3. To see the data in the database ... perform the

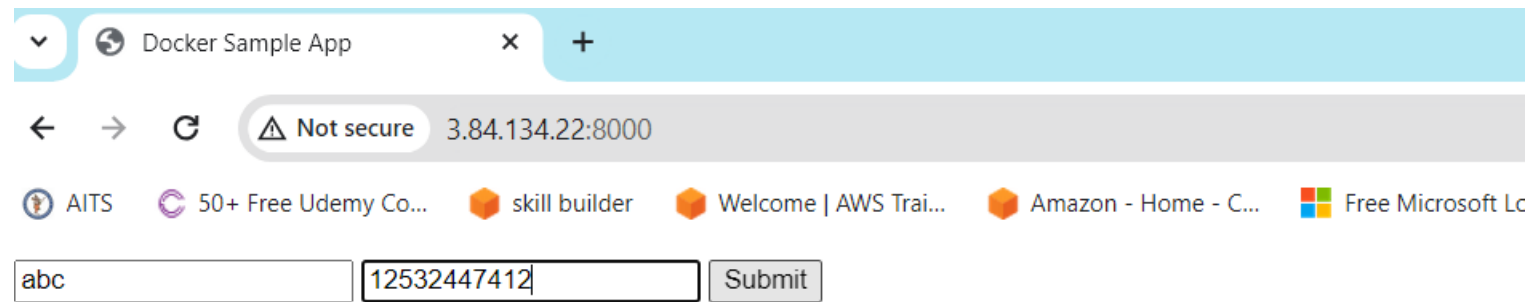
→ select * form emp;

```
mysql> select * from emp;
+-----+-----+
| name | phone      |
+-----+-----+
| abc  | 12532447412 |
+-----+-----+
1 row in set (0.00 sec)
```





Entering values into the webpage:-



▼

Docker Sample App

×

+


←


→


↻


⚠ Not secure


3.84.134.22:8000/index.php

 AITS

 50+ Free Udemy Co...

 skill builder

 Welcome | AWS Trai...

 Amazon - Hor

New record created successfully

Submit



To see the webapp page code :-

1. Enter into the /var/www/html path of the webapp container
2. And perform the

→ cat index.php

```
root@b9cc6b3d3187:/var/www/html# cat index.php
<html>
<head>
<title>Docker Sample App</title>

<?php
if($_SERVER['REQUEST_METHOD'] == "POST")
{
    $servername = "db";
    $username = "root";
    $password = "intelli";
    $dbname = "docker";
    $name=$_POST["name"];
    $phone=$_POST["phone"];

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    $sql = "INSERT INTO emp (name, phone)
VALUES ('".$name."', '".$phone."')";

    if ($conn->query($sql) === TRUE) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    $conn->close();
}
}
</head>
<body>
    <form action="index.php" method="POST">
        <input type="text" name="name">
        <input type="text" name="phone">
        <input type="submit" name="submit">
    </form>
</body>
</html>
```



Challenges faced during this project:-

- **Networking Complexity:** Managing networking between containers and across nodes in the cluster can be complex, especially when dealing with multiple tiers of the application.
- Monitoring the health and performance of containers, services, and nodes in the Docker Swarm cluster is essential for maintaining uptime and diagnosing issues promptly.
- Determining the optimal resource allocation for each tier of the application and scaling them dynamically based on demand requires careful planning and monitoring.



Conclusion:-

- Docker Swarm simplifies operations while enhancing reliability and scalability.

Embracing Docker Swarm empowers teams to focus on innovation and development, to handle the modern distributed applications.

- In conclusion, deploying a multi-tier application on Docker Swarm offers a powerful solution for managing complex architectures .

