

## **FARMEASY ON AZURE**

The implementation of a database application on a cloud using docker.

**BY:**

**Name:** Shivaraj Koli

**USN:** 1MS20CS112

**Name:** Shreyas R Kaundinya

**USN:** 1MS20CS113

**Name:** Tejas Hegde

**USN:** 1MS20CS129

As part of the Course **Operating Systems– CS52**

SUPERVISED BY

Faculty

Dr. Dayananda R.B.  
ASSOCIATE PROFESSOR  
DEPARTMENT OF CSE



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAMAIAH INSTITUTE OF TECHNOLOGY

Sept 2022 – Dec 2022

## **Objective:**

The main objective of this document is to demonstrate the implementation of a database application on a public cloud using docker.

## **Background:**

Farameasy is the name of our database application. The name has been kept keeping in mind the users of this application; Farmers and Researchers. Basically, Farmers upload data regarding themselves and the crops which they want to sell. This data is stored in a database which is then accessed for displaying available products to buyers. Besides we also have useful links for farmers in the homepage, a scroller image widget, a clock, a link to videos and miscellaneous. We have used JavaScript, HTML and CSS for the front end and PHP for the back-end connectivity. MariaDB(MySQL) is used as back-end database.

first_name	last_name	gender	city	district	state	aadhar	phone	crop
Rajesh	hiremath	male	bengalore	bengalore	karnataka	476474787989	7876477899	1,3,apple,mango,nuts
Arsha	ranade	male	chikkamangalore	chikkamangalore	karnataka	796587789654	9786588965	2,tomato,radish
Ranadhir	sharm	male	gadag	gadag	karnataka	789749789788	9886890806	2,tomato,brinjal
Ghambhir	shetty	male	kodagu	kodagu	karnataka	565788939399	8985988958	2,tomato,brinjal
Ganesh	karma	male	gadag	gadag	karnataka	789579745934	9098589505	2,3,tomato,gram
Anusha	kantar	female	uttarkannada	uttarkannada	karnataka	894794778489	9745454544	4,corn,ragi,jowar
Radhika	singe	female	udapi	udapi	karnataka	893797903080	7662388823	2,3,radish,dal
Raman	ankani	male	koppal	koppal	karnataka	767348934999	9959044459	3,4,nuts,dal,wheat
Ramya	sharm	female	bengalore	bengalore	karnataka	676287388378	9699096566	1,2,3,mango,orange,tomato,dal
Gangadhar	kohli	male	bagalakote	bagalakote	karnataka	874783889900	6888968906	4,wheat,ragi
Krishna	rathode	male	raichur	raichur	karnataka	587658676858	9905695690	1,4,apple,wheat
karna	karte	male	udapi	udapi	karnataka	648768763499	7889457979	1,2,apple,tomato
anita	choudary	female	ramanagar	ramanagar	karnataka	664687899999	8789893499	2,tomato
bharat	chouhan	male	gulbarga	gulbarga	karnataka	848884984888	7492399090	1,4,apple,ragi

Fig. 1: Sample Of the Data In Database



Fig. 2: The homepage of our website

## Docker:

Our database application is executed using XAMPP server, which loads PHP scripts using Apache web server and MySQL server. This works fine for our local computers. The aim is to make our application platform independent. There are varieties of operating systems or runtime environments in which we may have to run our application and there is no guarantee that it will work there. Docker is treated as the solution for such situation. It is similar to virtual

machines, the difference being that it virtualizes the operating system, while Virtual Machine virtualizes hardware. Using Docker containers, we can run a container in virtually any platform.

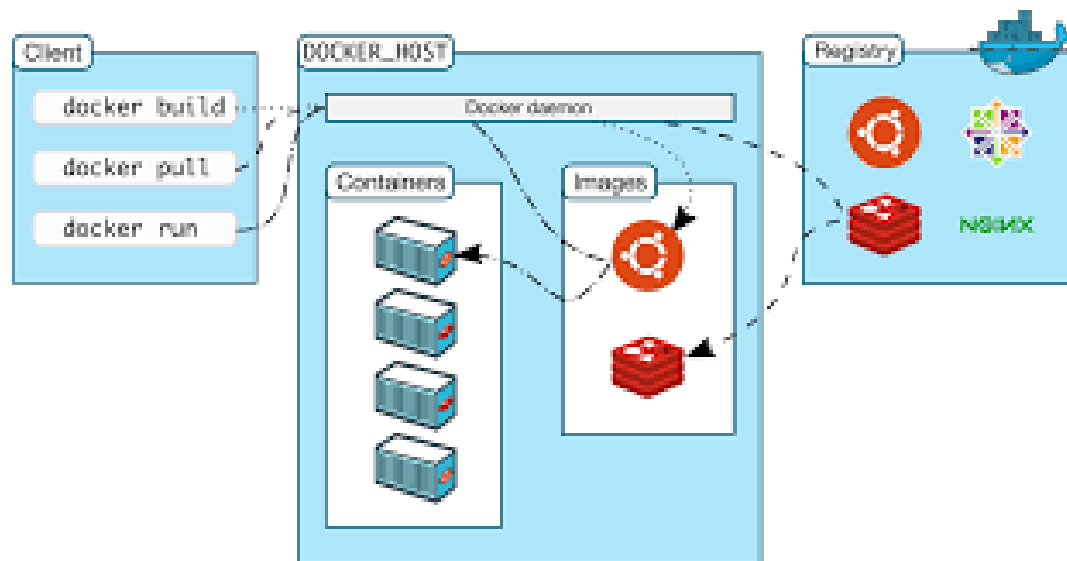


Fig. 3: Docker

## **Dockerizing the Database Application (By Shreyas):**

To dockerize our database application, we must have docker installed in our local system.

Dockerizing process was tested with with an Ubuntu:bionic Debian image, since docker is originally built for linux systems.

Here are the set of commands that we used in order to dockerize our database application. The following are the steps to dockerize.

1. Sudo apt install docker.io (Install Docker on the Local System)
2. sudo docker pull ubuntu:bionic (Pull the ubuntu/; bionic image)
3. sudo docker images (Check the images available in the Local System)
4. sudo docker run -idt --name os-project -p 8080:80 ubuntu:bionic /bin/bash (create a docker container in an interactive, detached, terminal mode with name “os-project”. Route requests to port 8080 on host machine through port 80 on the container.)
5. sudo docker container ls -a (Display the containers created)
6. sudo docker exec it os-project bash (Execute the container in bash shell)
7. Some of the net tools and other required tools are installed using the following commands
  - apt update && apt upgrade
  - apt install sudo
  - sudo apt-get install iputils-ping
  - sudo apt-get install net-tools
8. sudo docker cp /home/downloads/xampp-linux-x64-7.1.32-0-installer.run os-project:/opt/ (Copy the xampp installer to the newly created container)
9. chmod +x xampp-linux-x64-7.1.32-0-installer.run (Give permission to installer file for execution)
10. ./xampp-linux-x64-7.1.32-0-installer.run (Install xampp)

11. `#cd lampp` (Change to Xampp directory)

12. `#./lampp start` (Start Tomcat and mySql servers in Xampp)

13. `sudo docker cp /home/downloads/FarmEasy os-project:/opt/lampp/htdocs/FarmEasy` (Copy the webpages to xampp directory)

14. Check the website running using the container.



15. `sudo docker commit 21804891926d os-project-final` (Commit the changes to the image with a new name)

We have configured a running XAMPP server on an Ubuntu container. This container can now be deployed in any environment.

## **Azure:**

Azure is a popular public cloud, managed by the Microsoft Corporation. We chose Azure because of its ease of usage compared to other cloud platforms. It provides several free services for the first 12 months and also offers \$200 credit to be used in the first twelve months. Azure offers a broad variety of services like Web Apps, Virtual Machines, SQL storage and so on. For our project, we use the Web App service of Azure.



Fig. 5: Azure Cloud

## **Deploying Container To Azure:**

The first part of the deployment was to create a **Dockerfile**. It is a standard way of dynamically creating images. To explain this, Dockerfile is a file containing a set of instructions that tells docker how to produce an image. For our project, I prepared the Dockerfile as given below:

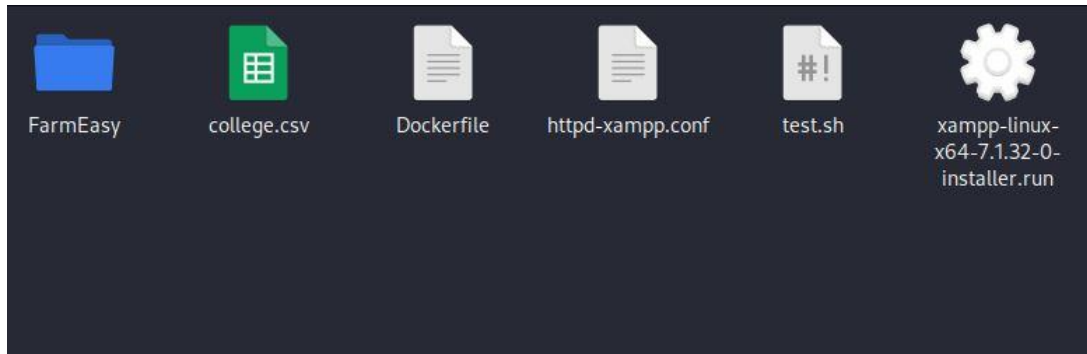
```
FROM ubuntu:bionic
COPY xampp-linux-x64-7.1.32-0-installer.run /opt
COPY test.sh /opt
WORKDIR /opt
RUN apt update
RUN apt-get update
RUN apt-get install net-tools
RUN chmod +x xampp-linux-x64-7.1.32-0-installer.run
RUN chmod +x test.sh
RUN ./xampp-linux-x64-7.1.32-0-installer.run
WORKDIR /opt/lampp/htdocs
RUN mkdir FarmEasy
COPY FarmEasy /opt/lampp/htdocs/FarmEasy
WORKDIR /opt
RUN rm /opt/lampp/etc/extra/httpd-xampp.conf
COPY httpd-xampp.conf /opt/lampp/etc/extra
EXPOSE 80
COPY test.sh /usr/local/bin/
RUN chmod u+x /usr/local/bin/test.sh
ENTRYPOINT ["test.sh"]
```

The httpd-conf is used to configure phpmyadmin for public access.

The test.sh has a very important function to perform. The XAMPP server, unlike other servers like nginx, does not run in the foreground. To keep the server running in foreground (otherwise the container exits immediately after run), we need something that runs forever. test.sh does exactly this. It runs an infinite while loop.

To build an image using the dockerfile, we must have a directory in a Linux system with docker installed like the one shown below. Note that college.csv is optional. It is required only for filling data in our database. Shivaraj has prepared this file.





Then, within that directory, open a terminal and issue the following command

***docker build .***

It takes a while to build the image. A message showing successful completion will be displayed.

Then I used ***docker tag*** command to give a name and tag to the created image. In my case, it was:

***docker tag <image-id> dockercoolexp/osproject:latest***

Basically, I will push my image to my Docker Hub registry named dockercoolexp (username) for further steps.

To run the container, we can issue the following command:

***docker run -p 8000:80 --name test dockercoolexp/osproject:latest***

To see the container in action, open a browser and type localhost:8000. You should be able to see the XAMPP welcome page. This shows the successful configuration of the container. Note that this would not be possible without the test.sh script that I mentioned before. I simply mapped port 8000 on the host to port 80 on the container where our XAMPP server is listening.

To push the image to DockerHub, I used the following commands:

***docker login***

***docker push dockercoolexp/osproject:latest***

After these commands, I have a docker image in my docker registry for public use. To pull the image simply use ***docker pull dockercoolexp/osproject:latest***.

Now lets come to Azure.

First, I open a cloud shell and then, I clone my Github repository containing all the required files using: ***git clone https://github.com/txh2020/DockerProject***.

Then issue ***cd DockerProject***.

Note that I haven't included the xampp installation file in my GitHub repo because it is too large (152MB) to fit in a normal GitHub repository. In Azure I dealt with the problem using wget:

***wget <https://sourceforge.net/projects/xampp/files/XAMPP%20Linux/7.1.32/xampp-linux-x64-7.1.32-0-installer.run/download>***

and then I renamed the file as xampp-linux-x64-7.1.32-0-installer.run using:

***mv download xampp-linux-x64-7.1.32-0-installer.run*** .

Basically, our project folder contents should be as follows:



```
college.csv  dc.yml  Dockerfile  FarmEasy  httpd-xampp.conf  'OS Project Report.pdf'  README.md  test.sh  xampp-linux-x64-7.1.32-0-installer.run
```

I decided to use **docker compose** to deploy to the image to Azure. After a lot of unsuccessful attempts, I found out that it was not possible to implement **VOLUME** using Dockerfile. A volume is used to persist the data stored in the database. The problem lies in the fact that we can create a volume in Dockerfile, but we cannot dynamically initialize it for storage. I also tried other techniques such as **Azure File Share** but was not successful. Then I came across Docker compose. We can build a docker image using a file known as **docker-compose.yml** using the following command:

**docker compose up**

The command automatically searches for the docker-compose.yml file and builds the image. For the deployment to Azure, my docker-compose.yml file, which is present as dc.yml in my GitHub repo is as follows:

```
version: '3.3'
```

```
services:
```

```
  db:
```

```
    image: dockercool/exp/osproject:latest
```

```
  volumes:
```

```
    - db_data:/opt/lampp/var/mysql
```

```
volumes:
```

```
  db_data:
```

This is where the image that I pushed to the DockerHub helps. I am pulling that image here since Azure cannot build it from scratch using **build .** option in the dc.yml file to itself build the file from scratch.

The goal is to deploy our dockerized web application as a web application in the cloud. In the cloud terminal, issue the following commands. Note that you can give any name to the resource group, service plan and webapp:

```
az group create --name myResourceGroup --location "South Central US"
```

```
az appservice plan create --name myAppServicePlan --resource-group myResourceGroup --sku F1 --is-linux
```

```
az webapp create --resource-group myResourceGroup --plan myAppServicePlan --name styapp2--multicontainer-config-type compose --multicontainer-config-file dc.yml
```

Once these steps are successfully completed, then issue the following command.

```
az webapp config appsettings set --resource-group myResourceGroup --name styapp2 --  
settings WEBSITES_ENABLE_APP_SERVICE_STORAGE=TRUE
```

This step is crucial because it gives persistency to the database.

Once all these steps are done, go to the Azure Portal. Search for styapp2. Click on the app. Then in the dashboard, we have the link to our webapp. Click on it. Initially, the application takes a lot of time to load. It may gateway timeout or application error. Do not heed these errors. Reload the browser as and when these errors appear. Finally, you will find the XAMPP welcome page appearing. Our web application is running successfully.

Then I just created the required database using phpMyAdmin and populated it using college.csv.

And there we go. The web application is finally ready.

In my case the link to the application is as follows:

<https://styapp2.azurewebsites.net/FarmEasy/proj.php>

## **References:**

<https://learn.microsoft.com/en-us/azure/app-service/tutorial-multi-container-app>

<https://taufanlubis.wordpress.com/2020/06/05/docker-tutorial-how-to-install-xampp-server-in-docker-container/>

<https://docs.docker.com/engine/reference/builder/>

<https://www.cloudbees.com/blog/using-docker-push-to-publish-images-to-dockerhub>

[https://www.tutorialspoint.com/unix/shell\\_scripting.htm](https://www.tutorialspoint.com/unix/shell_scripting.htm)