# Step by step guide to learn and productionizing ML Application using Docker, MySQL, Flask, Gunicorn, and Nginx

## Open Terminal/Windows PowerShell - 1:

- Change Directory(cd) to MLFullDayLab folder.

```
cd <>/MLFullDayLab
```

- Navigate to App subfolder.

```
cd App
```

- Check whether mysql:5.7.25 image is locally available or not.

```
docker images
```

- If mysql:5.7.25 image is not listed, then pull mysql:5.7.25 image from the docker hub.

```
docker pull mysql:5.7.25
```

- Recheck for the images.

```
docker images
ls
cd AppMySQL
```

- View the Dockerfile

- Build app_mysql image from Dockerfile

```
docker build -t app_mysql .
docker images
```

- Run: Create and Start the container

```
docker run -p 3306:3306 -v
/home/jeevan/Desktop/MLFullDayLab/App/AppMySQL/:/AppMySQL --name App_MySQL -e
MYSQL_ROOT_PASSWORD=insofe -d app_mysql
```

- List running containers

  ```
  docker ps
  ```

- Inspect App_MySQL container to find its IPAddress

  ```
  docker inspect App_MySQL
  ```

  Observation: "IPAddress": "172.17.0.2"

- Runs a new command in a running container.

  ```
  docker exec -it App_MySQL /bin/bash
  ```

- Connect to mysql

  ```
  mysql -u root -pinsofe
  ```

- Show databases

  ```
  show databases;
  ```

- Create cust_db database if doesn't exist

  ```
  create database cust_db;
  show databases;
  ```

- Change database to cust_db

  ```
  use cust_db;
  show tables;
  ```

  `exit` -> This is to come out of MySQL

- Check whether cust_data.dump is there in current folder

  ```
  ls
  ```

- Create bank table and populate the data using cust_data.dump file

  ```
  mysql -u root -pinsofe cust_db < cust_data.dump
  ```

- Connect to mysql

  ```
  mysql -u root -pinsofe
  ```

- Execute following commands

  ```
  use cust_db;
  show tables;
  select * from bank limit 5;
  select count(*) as NumRec from bank;
  ```

  `exit` -> This is to come out of MySQL

  `exit` -> This is to come out of the App_MySQL container

- Change directory to AppPython

  ```
  cd ../AppPython
  ```

- Build app_python image from Dockerfile

  ```
  docker build -t app_python .
  ```

- List the Docker images

  ```
  docker images
  ```

- Create and Run the Docker container

```
docker run -p 1234:1234 -v /home/jeevan/Desktop/MLFullDayLab/App/AppPython:/AppPython --name App_Python -it app_python /bin/bash
```

## Open Terminal/Windows PowerShell - 2:

- List running containers

  ```
  docker ps
  ```

- Inspect and identify the MLApp_Python container IP address

  ```
  docker inspect App_Python
  ```

  Observation: "IPAddress": "172.17.0.3"

## Go to Terminal/Windows PowerShell - 1:

- Run jupyter notebook.

  ```
  jupyter notebook --no-browser --ip=0.0.0.0 --port=1234 --allow-root
  ```

  Open the browser and past following URL

  http://172.17.0.3:1234/?token=b4e8dd99627d1b072da61ce27cd95c3f891407e02e81393b

  (or)

  http://127.0.0.1:1234/?token=b4e8dd99627d1b072da61ce27cd95c3f891407e02e81393b

  (or)

  http://0.0.0.0:1234/?token=b4e8dd99627d1b072da61ce27cd95c3f891407e02e81393b

- Got to notebook directory and run 01_Python_MySQL.ipynb

  Press Ctrl+C and y

  ```
  cd code
  ```

- Minimal Flask application

  Run Flask

  ```
  python 01_hello.py
  ```

  *This launches a very simple builtin server, which is good enough for testing but probably not what you want to use in production.*

  Open the browser and past following URL

  http://0.0.0.0:1234/

  Ctrl + C → To kill the server

- Routing

  Modern web applications use meaningful URLs to help users. Use the route() decorator to bind a function to a URL.

  Run Flask

```
python 02_Routing.py
```

Open the browser and past following URL

```
http://0.0.0.0:1234/
http://0.0.0.0:1234/hello
```

- Variable Rules

  You can add variable sections to a URL by marking sections with <variable_name>. Your function then receives the <variable_name> as a keyword argument. Optionally, you can use a converter to specify the type of the argument like <converter:variable_name>.

  Run Flask

  ```
  python 03_VariableRules.py
  ```

  Open the browser and past following URL

  ```
  http://0.0.0.0:1234/user/Jeevan
  http://0.0.0.0:1234/post/1
  http://0.0.0.0:1234/path/insofe/blr/jeevan
  ```

- URL Binding

  To build a URL to a specific function, use the url_for() function. It accepts the name of the function as its first argument and any number of keyword arguments, each corresponding to a variable part of the URL rule

  Run Flask

  ```
  python 04_URLBuilding.py
  ```

  Open the browser and past following URL

  ```
  http://0.0.0.0:1234/admin
  http://0.0.0.0:1234/guest/Jeevan
  http://0.0.0.0:1234/user/Jeevan
  http://0.0.0.0:1234/user/admin
  ```

- HTTP Methods

  Run Flask

python 05_HTTPMethods.py

Open the browser and past following URL

http://0.0.0.0:1234/

Open the 05_login.html using browser

Using editor open 05_login.html, change method from POST to GET and reload 05_login.html browser

- Templates

Run Flask

python 06_Templates_00.py

Open the browser and past following URL

http://0.0.0.0:1234

Run Flask

python 06_Templates_01.py

Open the browser and past following URL

http://0.0.0.0:1234/hello/Jeevan

Run Flask

python 06_Templates_02.py

Open the browser and past following URL

http://0.0.0.0:1234/hello/75
http://0.0.0.0:1234/hello/45

Run Flask

python 06_Templates_03.py

Open the browser and past following URL

http://0.0.0.0:1234/result

- Sending Form Data two Template

  Run Flask

  ```
  python 07_SendingFormData2Template.py
  ```

  Open the browser and past following URL

  http://0.0.0.0:1234

- File Uploading

  Run Flask

  ```
  python 08_FileUploading.py
  ```

  Open the browser and past following URL

  http://0.0.0.0:1234/upload

  Browse INSOFE.png file and click on Submit Query

- Green Unicorn

  Run Flask

  ```
  python gunicorn_flask.py
  ```

  Open the browser and past following URL

  http://0.0.0.0:1234/

  Run with gunicorn

  ```
  gunicorn --bind 0.0.0.0:1234 wsgi:app
  ```

  Open the browser and past following URL

  http://0.0.0.0:1234/

  `exit` -> To come out of Container

- Navigate to App folder

```
cd ..
```

- Execute following two docker-compose commands

```
docker-compose build

docker-compose up
```

## Open Terminal/Windows PowerShell - 2:

- Check whether required containers are running or not

```
docker ps
docker ps -a
```

- Navigate to AppMySQL folder

```
cd AppMySQL
```

  - Runs a new command in a running container.

```
docker exec -it App_MySQL /bin/bash
```

- Create bank table and populate the data using cust_data.dump file

```
mysql -u root -pinsofe cust_db < cust_data.dump
```

- Connect to mysql

```
mysql -u root -pinsofe

use cust_db;

select count(*) as NumRec from bank;

exit -> To exit MySQL

exit -> To exit container
```

- Inspect App_MySQL to find its ip address

```
docker inspect App_MySQL
```

Note: Change the ip address in the notebook accordingly

- Inspect App_Python to find its ip address

```
docker inspect App_Python
```

- Open Notebook in the browser

```
docker-compose down
```

********************************************* BREAK *********************************************

## Go to Terminal/Windows PowerShell - 1:

- List available docker images

```
docker images

cd <>/MLApp/AppMySQL
```

- If app_mysql docker is not available, build it using the Dockerfile
- Build app_mysql image from Dockerfile

```
docker build -t app_mysql .
docker images
```

- Run: Create and Start the container

```
docker run -p 3306:3306 -v
/home/jeevan/Desktop/MLFullDayLab/MLApp/AppMySQL/:/AppMySQL --name App_MySQL
-e MYSQL_ROOT_PASSWORD=insofe -d app_mysql
```

- List running containers

```
docker ps
```

- Inspect App_MySQL container to find its IPAddress

```
docker inspect App_MySQL
```

Observation: "IPAddress": "172.17.0.2"

- Runs a new command in a running container.

```
docker exec -it App_MySQL /bin/bash
```

- Connect to mysql

```
mysql -u root -pinsofe
```

- Show databases

```
show databases;
```

- Change database to cust_db

```
use cust_db;
show tables;
```

`exit` -> This is to come out of MySQL

- Check whether cust_data.dump is there in current folder

```
ls
```

- Create bank table and populate the data using cust_data.dump file

```
mysql -u root -pinsofe cust_db < cust_data.dump
```

- Connect to mysql

```
mysql -u root -pinsofe
```

- Execute following commands

```
use cust_db;
show tables;
select * from bank limit 5;
select count(*) as NumRec from bank;
```

`exit` -> This is to come out of MySQL

`exit` -> This is to come out of the App_MySQL container

- Just confirm whether App_MySQL container is still running or not.

```
docker ps
```

- Navigate to AppPython folder

```
cd ../AppPython
```

- List available docker images.

```
docker images
```

- If App_Python image is not listed, then build App_Python image from Dockerfile

```
docker build -t app_python .
```

- List the Docker images

```
docker images
```

- Create and Run the Docker container

```
docker run -p 1234:1234 -v
/home/jeevan/Desktop/MLFullDayLab/MLApp/AppPython/:/AppPython --name
App_Python -it app_python /bin/bash
```

## Go to Terminal/Windows PowerShell - 2:

- List running containers

```
docker ps
```

- Inspect and identify the MLApp_Python container's IP address

```
docker inspect App_Python
```

Observation: "IPAddress": "172.17.0.3"

## Go to Terminal/Windows PowerShell - 1:

- Check whether reading the data MySQL and some pre-preprocess functions are working as expected

```
python Python_MySQL.py
```

- Read the data from MySQL, Pre-process, build the model and save everything as Pipeline

```
python build.py
```

- Make Predict on test data in .csv file

```
python predict.py
```

- Running Flask

```
python predict_flask.py
```

Open the browser and past following URL

http://0.0.0.0:1234/

- Run with gunicorn

```
gunicorn --bind 0.0.0.0:1234 wsgi:app
```

Open the browser and past following URL

http://0.0.0.0:1234/

```
Ctrl + c
```

`Exit` -> To exit the container

- Navigate to MLApp folder and execute following two docker-compose commands

```
docker-compose build
```

```
docker-compose up
```

## Go to Terminal/Windows PowerShell - 2:

```
docker-compose down
```