Step by step guide to writing end-to-end production ready ML/AI application using Docker, MySQL, Python, Flask and Scikit-learn

Open Terminal/Windows PowerShell - 1:

- Navigate(cd) to the **DevProRedCodeFlask** folder.

  cd <>/DevProRedCodeFlask

- Navigate(cd) to the **App** subfolder.

  cd App

- Check whether mysql image is locally available or not.

  docker images

- If the mysql image is not listed, then pull the mysql 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 <name:tag> <dockerfile location>

  docker build -t app_mysql .
  docker images

- Run: Create and Start the container

  Docker run

  -d : Run container in background and print container ID

-i : Interactive
-p : Map port in the container to port on the Docker host.
e.g. -p <Docker host port>:<Container port >
-t : Allocate a pseudo-TTY
--mount : Attach a filesystem mount to the container
E.g. --mount <Docker host path>:<Container path>
Usually Container path will be container's WORKDIR
Note: Mount dependent on host machine directory structure

--name : Assign a name to the container

```
docker run -p 3306:3306 --mount
type=bind,source=/Users/jeevan/Desktop/DevProRedCodeFlask/App/AppMySQL/,tar
get=/AppMySQL --name App_MySQL -d app_mysql
```

(or)

```
docker run -p 3306:3306 -v
/Users/jeevan/Desktop/DevProRedCodeFlask/App/AppMySQL/:/AppMySQL --name
App_MySQL -d app_mysql
```

Note: If "docker: Error response from daemon: Conflict. The container name "/App_MySQL" is already in use …." error is shown, then kill the existing App_MySQL container and then try again.

```
docker ps -a
docker rm App_MySQL
```

- List running containers

```
docker ps
```

- Find the container IPAddress using **Inspect**.

```
docker inspect App_MySQL
```

E.g.: "IPAddress": "172.17.0.2"

Note: For **Docker Toolbox** installation types, find the VirtualBox IPAddress using **docker-machine**

```
docker-machine ls
```

E.g.: URL
tcp://192.168.99.100:2376

- Runs a new command in a running container.

-i : interactive
-t : Allocate a pseudo-TTY

```
docker exec -it App_MySQL bash
```

- Check whether cust_data.dump is there in the current folder or not.

  ```
  ls
  ```

  Note: If the file is not present, problem is with the volume mapping/mount

- Connect to mysql

  mysql -u <username> -p<password>

  ```
  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

- 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

- View the *Dockerfile and requirement.txt*

- 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 --mount type=bind,source=/Users/jeevan/Desktop/DevProRedCodeFlask/App/AppPython/,target=/AppPython --name App_Python -it app_python bash

  (or)

  docker run -p 1234:1234 -v /Users/jeevan/Desktop/DevProRedCodeFlask/App/AppPython/:/AppPython --name App_Python -it app_python bash

- Check whether the notebooks folder is there in the current folder or not.

  ls

  Note: If the file is not present, problem is with the volume mapping/mount

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"

Note: For **Docker Toolbox** installation types, find the VirtualBox IPAddress using **docker-machine**

docker-machine ls

E.g.: URL
tcp://192.168.99.100:2376

Go to Terminal/Windows PowerShell - 1:

- Run a jupyter notebook.

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

  E.g. Open the browser and past following URL

  http://0.0.0.0:1234/?token=b4e8dd99627d1b072da61ce27cd95c3f891407e02e81393b
  (or)
  http://172.17.0.3:1234/?token=b4e8dd99627d1b072da61ce27cd95c3f891407e02e81393b
  (or)
  http://127.0.0.1:1234/?token=b4e8dd99627d1b072da61ce27cd95c3f891407e02e81393b

  Note: For Toolbox Installation type, only virtual box IPaddress has to be used

  http://**192.168.99.100**:1234/?token=b4e8dd99627d1b072da61ce27cd95c3f891407e02e81393b

- Got to notebook directory and run 01_Python_MySQL.ipynb

  Press Ctrl+C and y

- Navigate code folder

  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

Using Docker Compose

- Navigate to App folder

  cd ..

- Check the correctness of the docker-compose.yml file

  docker-compose config

- Build the images defined in the docker-compose file using

  docker-compose build

- Start all the services using

  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 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

- Start all the services using

  docker-compose down

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

Go to Terminal/Windows PowerShell - 1:

- Navigate (cd) to MLApp subfolder in DevProRedCode folder

  cd <>/DevProRedCodeFlask/MLApp/

- Navigate to AppMySQL subfolder

  cd AppMySQL

- List available images

  docker images

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

  docker build -t app_mysql .
  docker images

- Run: Create and Start the container

  ```
  docker run -p 3306:3306 --mount
  type=bind,source=/Users/jeevan/Desktop/DevProRedCodeFlask/MLApp/AppMySQL/
  ,target=/AppMySQL --name App_MySQL -d app_mysql
  ```

  (or)

  ```
  docker run -p 3306:3306 -v
  /Users/jeevan/Desktop/DevProRedCodeFlask/MLApp/AppMySQL/:/AppMySQL
  --name App_MySQL -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 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 --mount
  type=bind,source=/Users/jeevan/Desktop/DevProRedCodeFlask/MLApp/AppPython/,
  target=/AppPython --name App_Python -it app_python bash

  (or)

  docker run -p 1234:1234 -v
  /Users/jeevan/Desktop/DevProRedCodeFlask/MLApp/AppPython/:/AppPython
  --name App_Python -it app_python bash

Go to Terminal/Windows PowerShell - 2:

- List running containers

  docker ps

- Inspect and identify the App_Python container's IP address

  docker inspect App_Python

  Observation: "IPAddress": "172.17.0.3"

- Inspect and identify the App_MySQL container's IP address

  docker inspect App_MySQL

  Observation: "IPAddress": "172.17.0.2"

  Note: For **Docker Toolbox** installation types, find the VirtualBox IPAddress using **docker-machine**

  docker-machine ls

  E.g.: URL
  tcp://**192.168.99.100**:2376

- Check whether the **config.ini** file in the conf folder is having an App_MySQL container/VirtualBox IP Address or not.

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

Tips and Troubleshooting Techniques

Tar all the required images for the lab

Open Terminal/Windows PowerShell - 1:

- Using cd command navigate to the folder where you want to save the images as .tar files, and execute following docker command to save the image.

  docker save -o <IMAGE_NAME>.tar <IMAGE_NAME>

  E.g. docker save -o app_mysql.tar app_mysql

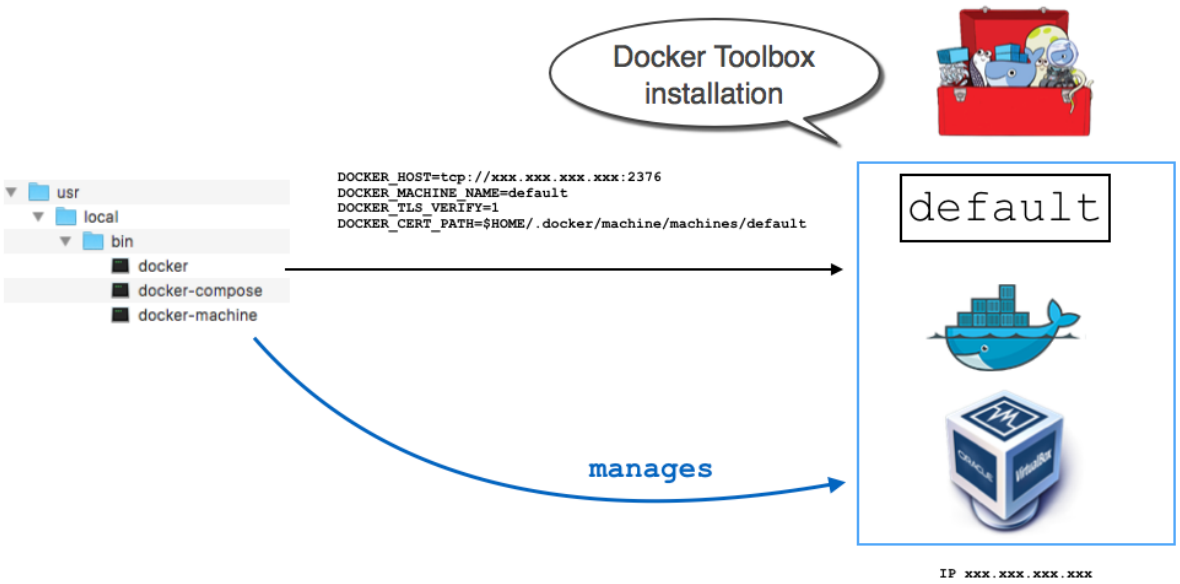
- Using cd command navigate to the folder where you have saved image tar files, and execute following docker command to load the images from the .tar file.

  docker load -i <IMAGE_NAME>.tar

  E.g. docker load -i app_mysql.tar

Docker Toolbox for Windows

Docker Toolbox installs docker, docker-compose, and docker-machine. It also installs VirtualBox. At installation time, Toolbox uses docker-machine to provision a VirtualBox VM called default, running the boot2docker Linux distribution.



In this type of installation Container IP is not applicable. To get the IP

docker-machine ls

Docker Desktop for Windows/ Mac OS/ Linux

docker inspect <CONTAINER_NAME>

Docker ToolBox

To use Docker Command Line Interface open Docker QuickStart Terminal

Docker Desktop for Windows/ Mac OS/ Linux

To use Docker Command Line Interface open Termincal/CMD/Windows PowerShell

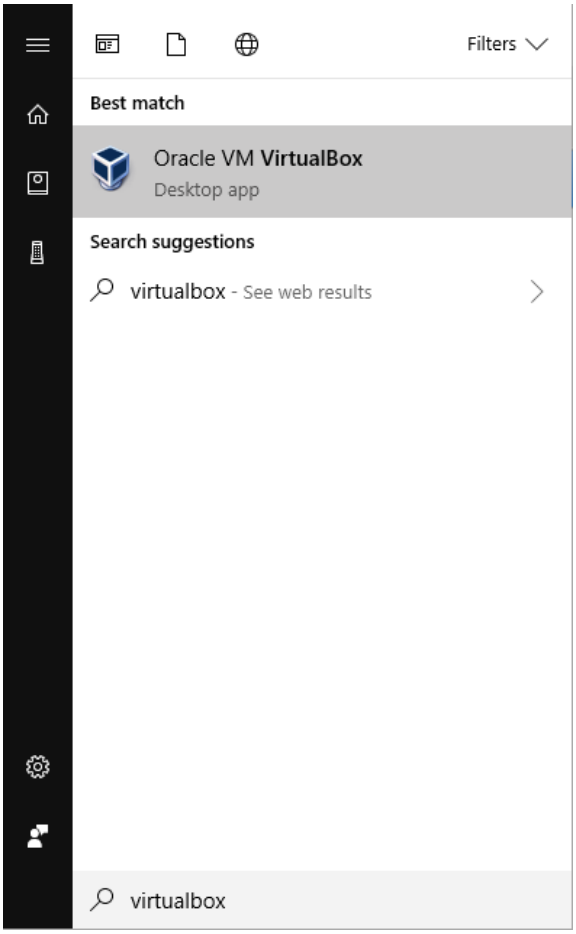
Fixing Volumes in Docker Toolbox

When running your container, use //x/ to reference your drive, where x is your lowercase drive letter. To reference C:\Users\admin\volume, you would use //c/Users/admin/volume.

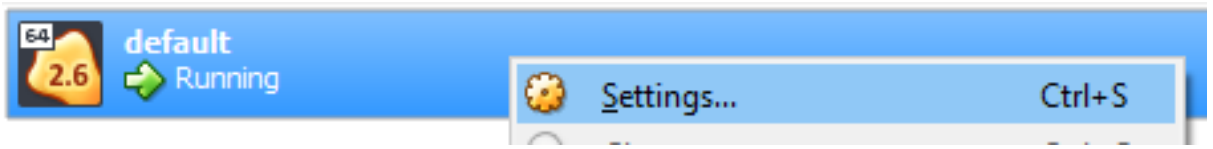
Note: If the path has spaces wrap path in between “ ”.

Configure Oracle VirtualBox to have access to that folder. This is an important solution that is not well documented. Oracle VirtualBox, the virtual machine behind Docker Toolbox, does not have access to your folders by default. Mounting C:\MyFolder\ *does nothing* until you’ve given VirtualBox access to that folder.

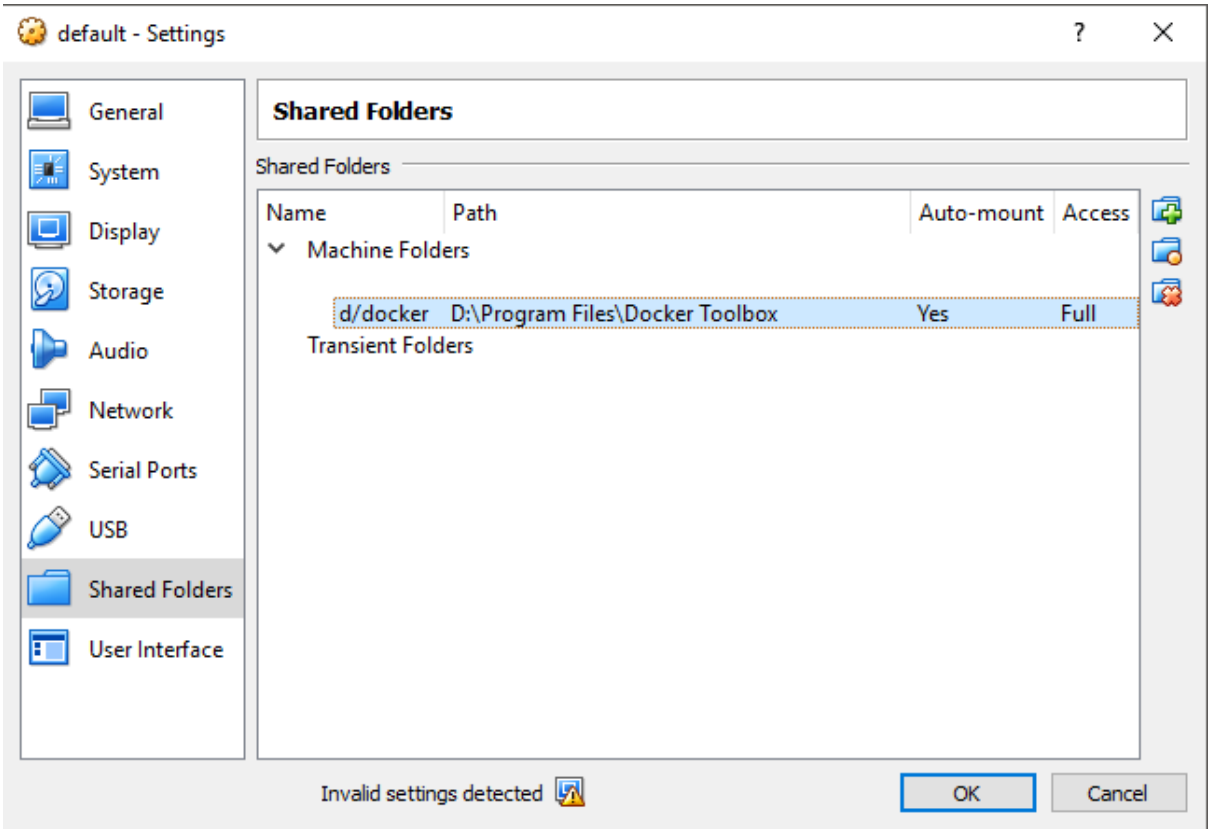
1. Find VirtualBox on your machine and run it.



2. Right-Click the default machine and choose Settings.



3. Choose the **Shared Folders** category and add a shared folder.



Give the folder name something convenient. It is what you will type when you mount it via --volume or -v. Make it auto-mount and permanent.

In my case, if I docker run --volume //d/docker/nginx:/etc/nginx, I will be binding the /etc/nginx directory in my container to D:\Program Files\Docker Toolbox\nginx. I am not suggesting you use the Docker Toolbox directory, as you can use any that you desire; I only used it as an example.

You may need to restart the default machine for your changes to take affect. You may do so by right-clicking it and choosing the Reset option.

Binding your volumes should now work.

Ref : https://medium.com/@Charles_Stover/fixing-volumes-in-docker-toolbox-4ad5ace0e572

Note: As images are already shared and loaded, don't execute **docker build** and **docker pull** and **docker-compose build** commands. Those commands are just for your reference.