

DOCKER – DOCUMENTATION

DOCKER FILE :

Dockerfile

| INSTRUCTION | ARGUMENT |
|-------------|---|
| FROM | Ubuntu |
| RUN | apt-get update && apt-get -y install python |
| RUN | pip install flask flask-mysql |
| COPY | ./opt/source-code |
| ENTRYPOINT | FLASK_APP=/opt/source-code/app.py flask run |

Annotations:

- Start from a base OS or another image (points to FROM)
- Install all dependencies (points to RUN apt-get)
- Copy source code (points to COPY)
- Specify Entrypoint (points to ENTRYPOINT)

It helps to run instructions and their arguments so that we can deploy Images in our local system, Or we can also post it on our docker hub.

LAYERED ARCHITECTURE :

Layered architecture

docker build . -f Dockerfile -t mmumshad/my-custom-app

| Layer | Size |
|---|--------|
| Layer 1. Base Ubuntu Layer | 120 MB |
| Layer 2. Changes in apt packages | 306 MB |
| Layer 3. Changes in pip packages | 6.3 MB |
| Layer 4. Source code | 229 B |
| Layer 5. Update Entrypoint with "flask" command | 0 B |

Help us to initialise or to understand all layers from base to the final layer with how many space they are occupying

DOCKER BUILD OUTPUT :

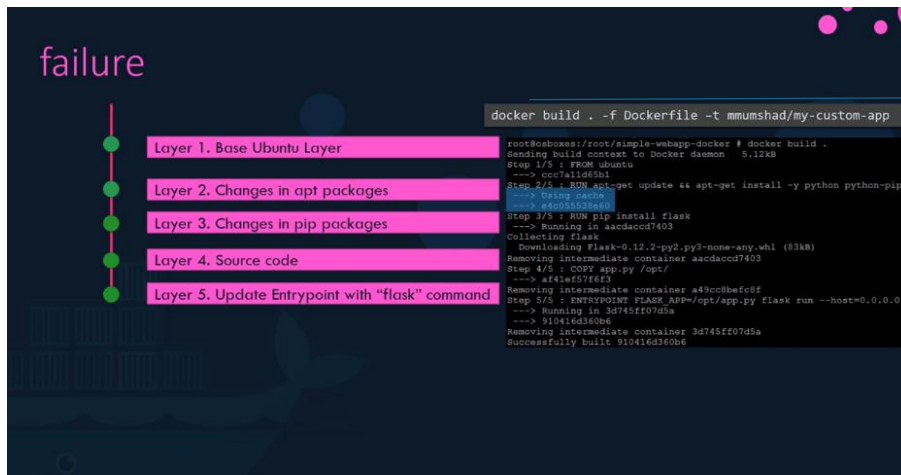
Docker build output

```
root@esboxes:/root/simple-webapp-docker # docker build .
Sending build context to Docker daemon 3.072kB
Step 1/5 : FROM ubuntu
--> ccc7a11d65b1
Step 2/5 : RUN apt-get update && apt-get install -y python python-setuptools python-dev
--> Running in a7840d8fad17
Get:1 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Get:4 http://security.ubuntu.com/ubuntu xenial-security/universe Sources [46.3 kB]
Get:5 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Get:6 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [440 kB]
Step 3/5 : RUN pip install flask flask-mysql
--> Running in a4a6c9190ba3
Collecting flask
  Downloading Flask-0.12.2-py3-none-any.whl (83kB)
Collecting flask-mysql
  Downloading Flask_MySQL-1.4.0-py3-none-any.whl
Removing intermediate container a4a6c9190ba3
Step 4/5 : COPY app.py /opt/
--> e7cdab17e782
Removing intermediate container f8aaef63c512
Step 5/5 : ENTRYPOINT FLASK_APP=/opt/app.py flask run --host=0.0.0.0
--> Running in d452c574a8bb
--> 9f27c36920bc
Removing intermediate container d452c574a8bb
Successfully built 9f27c36920bc
```

Output for the Docker Build

As you can see that Docker build commands run all steps from step 1 to step 5, So if you want to add some extra steps, you don't need to run it again from step one.

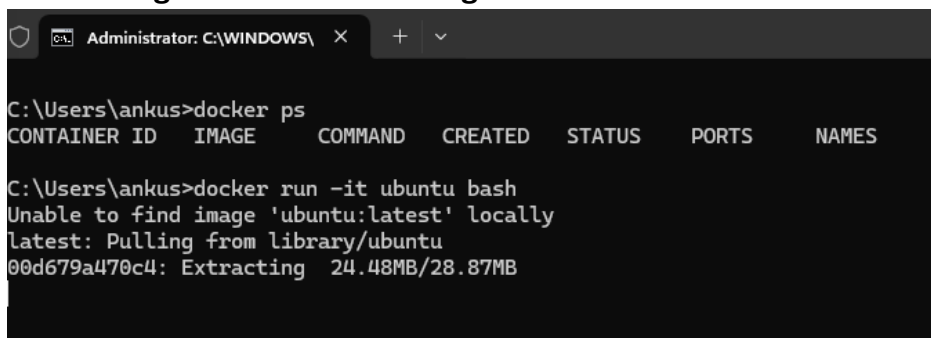
FAILURE :



In a case of failures, it will not run from the initial step again Just take the cache of the above steps and try to build the next step ,So if you want to add some extra steps also you don't need to run it again from step one.

CHAPTER 1 : --

→ Creating our new docker image

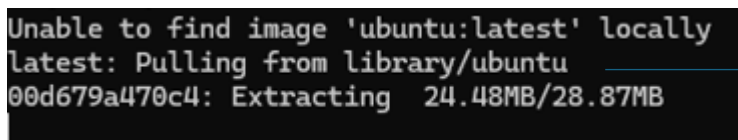


docker ps → This command lists all the running containers. The output includes the container ID, image name, command that was run, creation time, status, ports, and name of the container.

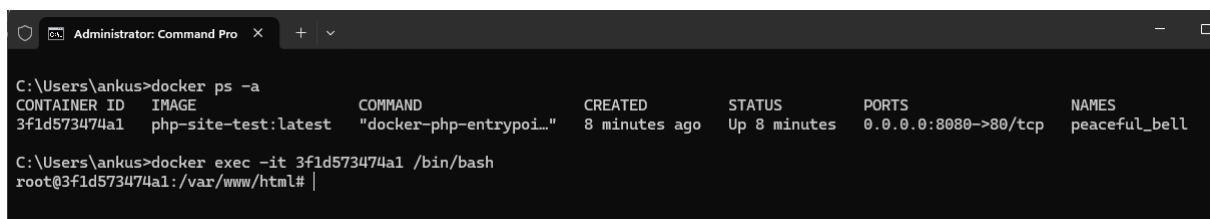
docker ps -a → List all containers (running and stopped)

docker ps -q → Display only container IDs

docker ps -f "status=exited" → Filter containers by status (e.g., exited)



This is pulling an Ubuntu Image from Docker hub



docker exec -it 3f1d573474a1 /bin/bash → To get inside a running container.

Hosting PHP website from the official php image →

In the image ***3f1d573474a1*** is the container id that is running through the image name that I created using ***docker build . -t php-site-test***

```
Administrator: Command Pro X + v
C:\Users\ankus>docker ps -q
3f1d573474a1

C:\Users\ankus>docker exec -it 3f1d573474a1 /bin/bash
root@3f1d573474a1:/var/www/html# ls -la
total 20
drwxrwxrwx 1 www-data www-data 4096 Jun 14 07:24 .
drwxr-xr-x 1 root      root    4096 Nov 15 2022 ..
-rwxr-xr-x 1 root      root      267 Jun 14 07:24 Dockerfile
-rwxr-xr-x 1 root      root     2534 May 23 05:19 index.php
root@3f1d573474a1:/var/www/html#
```

cat Dockerfile : →

```
root@3f1d573474a1:/var/www/html# cat Dockerfile
# Use the official PHP image with Apache as the base image
FROM php:7.4-apache

# Copy the PHP files to the Apache document root
COPY . /var/www/html/

# Expose port 80 to the outside world
EXPOSE 80

# Start the Apache server
CMD ["apache2-foreground"]

root@3f1d573474a1:/var/www/html#
```

cat index.php : →

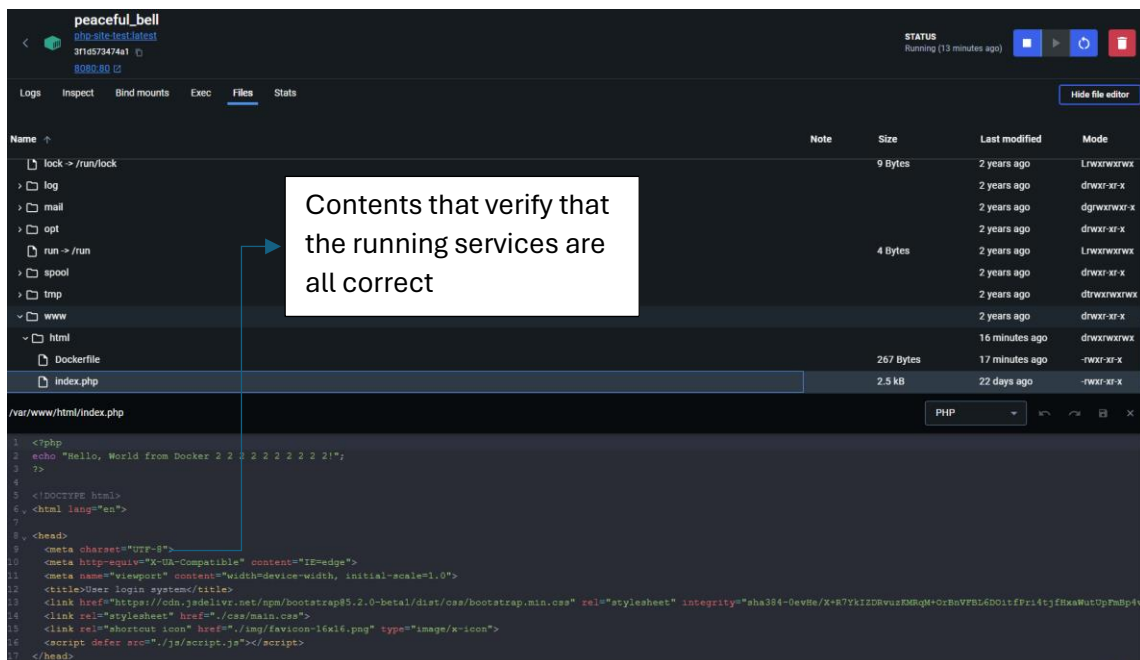
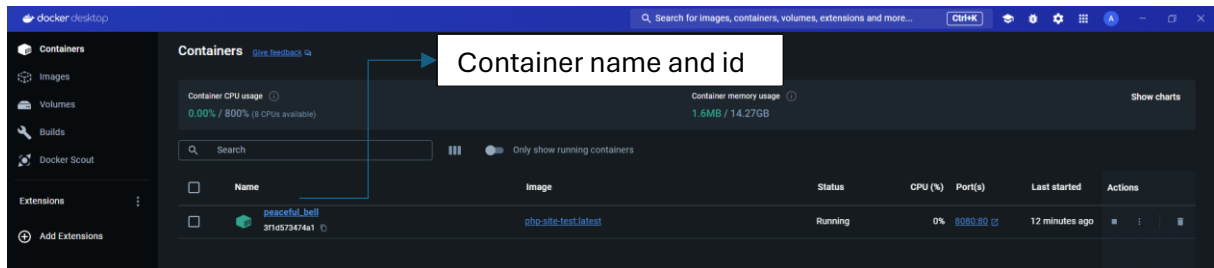
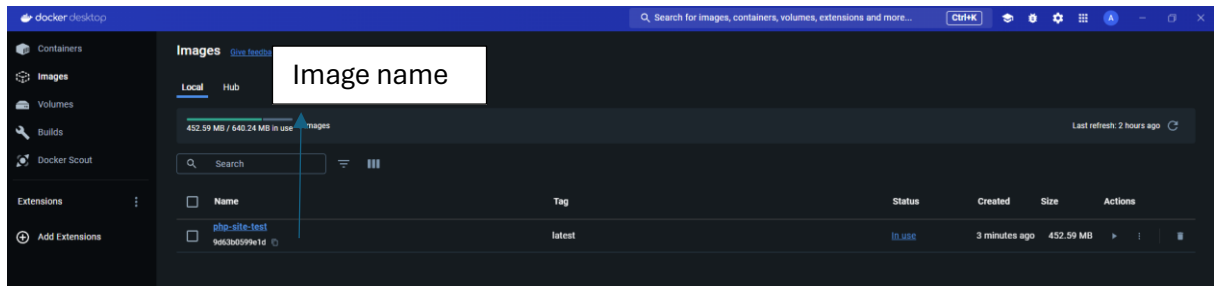
```
root@3f1d573474a1:/var/www/html# cat index.php
<?php
echo "Hello, World from Docker 2 2 2 2 2 2 2 2 2!";
?>

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>User login system</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-MqrW1XZYx1Lq7Mg967E8+MDtAwEs9PthHjGwBf0qIwZ2bY71X1TQWZp38Wz7" crossorigin="anonymous">
  <link rel="stylesheet" href="/css/main.css">
  <link rel="shortcut icon" href="/img/favicon-16x16.png" type="image/x-icon">
  <script defer src="/js/script.js"></script>
</head>

<body>
  <div class="container">
    <div class="row min-vh-100 justify-content-center align-items-center">
      <div class="col-lg-5">
        <?php
        if (!empty($login_err)) {
          echo "<div class='alert alert-danger'" . $login_err . "</div>";
        }
        ?>
      </div>
    </div>
  </div>
</body>
</html>
```

DOCKER DESKTOP : →



github Link with workflow file that automatically push Image Into My Docker Hub Account

https://github.com/ankushjha-aj/DOCKER_TESTS/tree/docker-php-image

OUTPUT : --

NOTE : → no MYSQL Connected, just a DEMO page

