

## 1. Docker Registry

### ◆ What is a Docker Registry?

A **Docker Registry** is a storage and distribution system for Docker images. It allows you to **push**, **pull**, and **manage** images.

There are two types:

- **Public Registry:** Like DockerHub
- **Private Registry:** Self-hosted or cloud-based (e.g., AWS ECR, Azure ACR, Google GCR)

## 2. DockerHub

### What is DockerHub?

DockerHub is the **default public registry** for Docker images.

It lets users:

- Search and download pre-built images
- Host their own public or private images
- Version control with tags
- Collaborate on teams and projects

### Common DockerHub Commands:

Command	Description
<code>docker login</code>	Authenticate with DockerHub
<code>docker pull nginx</code>	Download image from DockerHub
<code>docker tag local-image username/image-name:tag</code>	Tag image for DockerHub
<code>docker push username/image-name:tag</code>	Push image to DockerHub
<code>docker search image-name</code>	Search DockerHub images

```
C:\Users\prate>docker login
Authenticating with existing credentials... [Username: prateek2004]

Info → To login with a different account, run 'docker logout' followed by 'docker login'

Login Succeeded

C:\Users\prate>
```

```
C:\Users\prate>docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
dad67da3f26b: Already exists
3b00567da964: Pull complete
56b81cfa547d: Pull complete
1bc5dc8b475d: Pull complete
979e6233a40a: Pull complete
d2a7ba8dbfee: Pull complete
32e44235e1d5: Pull complete
Digest: sha256:6784fb0834aa7dbbe12e3d7471e69c290df3e6ba810dc38b34ae33d3c1c05f7d
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

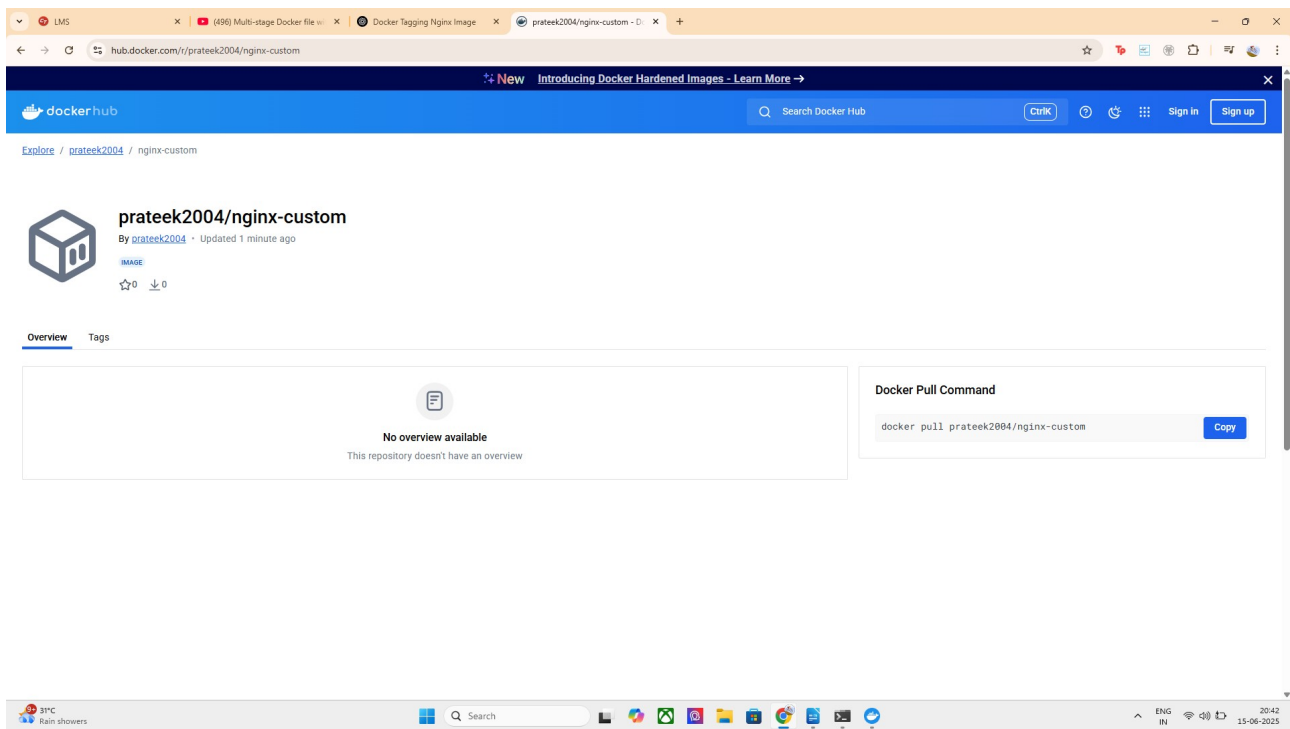
```
C:\Users\prate>docker tag nginx prateek2004/nginx-custom:latest

C:\Users\prate>docker push yourusername/nginx-custom:latest
The push refers to repository [docker.io/yourusername/nginx-custom]
An image does not exist locally with the tag: yourusername/nginx-custom

C:\Users\prate>docker push prateek2004/nginx-custom:latest
The push refers to repository [docker.io/prateek2004/nginx-custom]
7e893c1b6ee8: Mounted from library/nginx
463308bed0c9: Mounted from library/nginx
4197a611afec: Mounted from library/nginx
3e96162769d5: Mounted from library/nginx
892e805f6f4f: Mounted from library/nginx
626ab8a5d57b: Mounted from library/nginx
7fb72a7d1a8e: Mounted from library/nginx
|
```

You can use to check :

<https://hub.docker.com/r/prateek2004/nginx-custom>



### 3. Create a Multi-Stage Build

#### ◆ What is a Multi-Stage Build?

Multi-stage builds help:

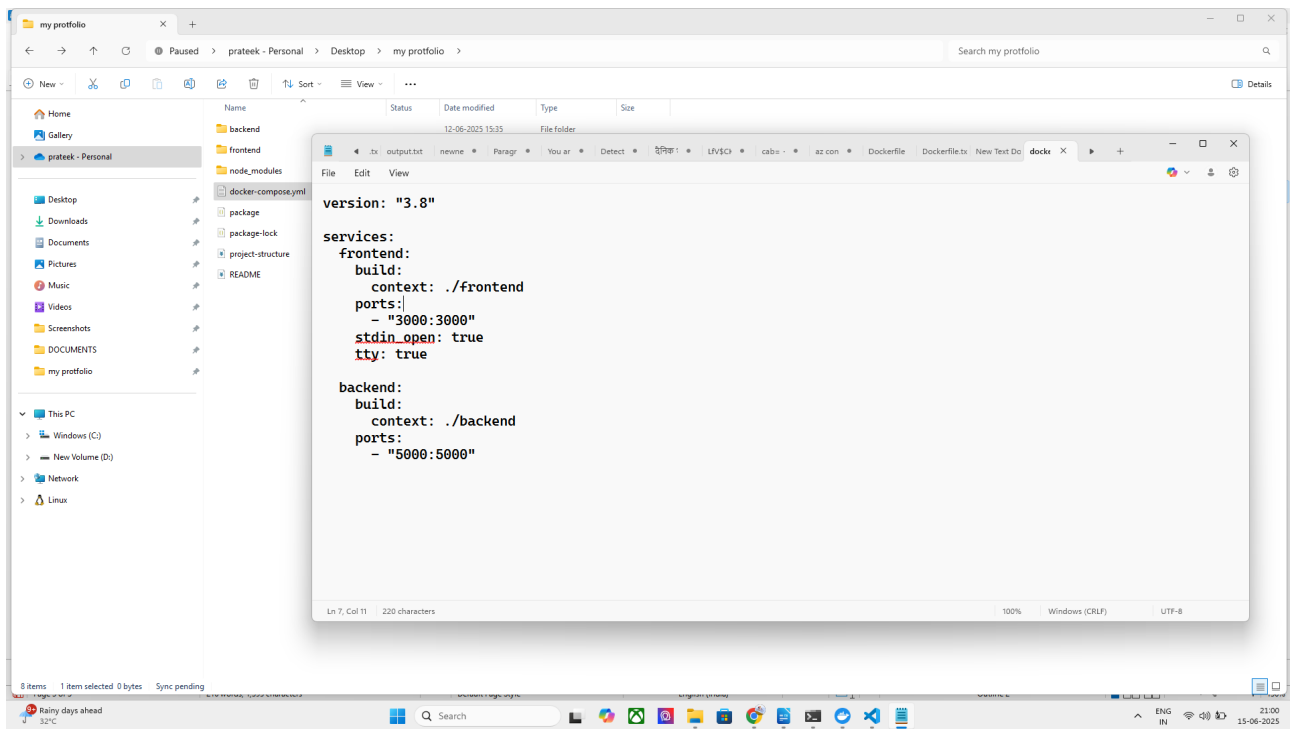
- Optimize final image size
- Separate **build dependencies** from **runtime**
- Improve security and performance

Lets create 1<sup>st</sup> without multi bulid to check the efficiency :

#### 1. Dockerfile for Frontend (No Multi-Stage)

```
# React dev environment (not optimized)
FROM node:18-alpine
```

```
WORKDIR /app
COPY . .
RUN npm install
CMD ["npm", "start"]
EXPOSE 3000
```



IN TTHIS APPROACH IT HAS MADE THE FILE OF ( 551 Mb ) for the frontend only

```
Run 'docker image COMMAND --help' for more information on a command.
PS D:\OneDrive\Desktop\my protfolio> docker image ls
REPOSITORY              TAG                IMAGE ID           CREATED            SIZE
myprotfolio-frontend    latest            da30a22769d8       3 minutes ago     551MB
my-python-app           latest            44c082b8ea65       2 hours ago       127MB
ubuntu                  latest            bf16bdcff9c9       2 weeks ago       78.1MB
prateek2004/nginx-custom latest            1e5f3c5b981a       2 months ago      192MB
nginx                   latest            1e5f3c5b981a       2 months ago      192MB
nginx                   alpine            6769dc3a703c       2 months ago      48.2MB
PS D:\OneDrive\Desktop\my protfolio>
```

LETS GO TO THE MULTI STAGE BUILT FOR THE SAME

NOW THE DOCKER FILE WILL BE

FOR FRONTEND IN 2 STAGES

```
# Stage 1: Build React app
FROM node:18-alpine AS builder
WORKDIR /app
COPY . .
RUN npm install
RUN npm run build

# Stage 2: Serve with Nginx
FROM nginx:alpine
COPY --from=builder /app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

## **AND DOCKER-COMPOSE.YML FILE WILL BE**

```
version: "3.8"
```

```
services:
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "3000:80"
    depends_on:
      - backend

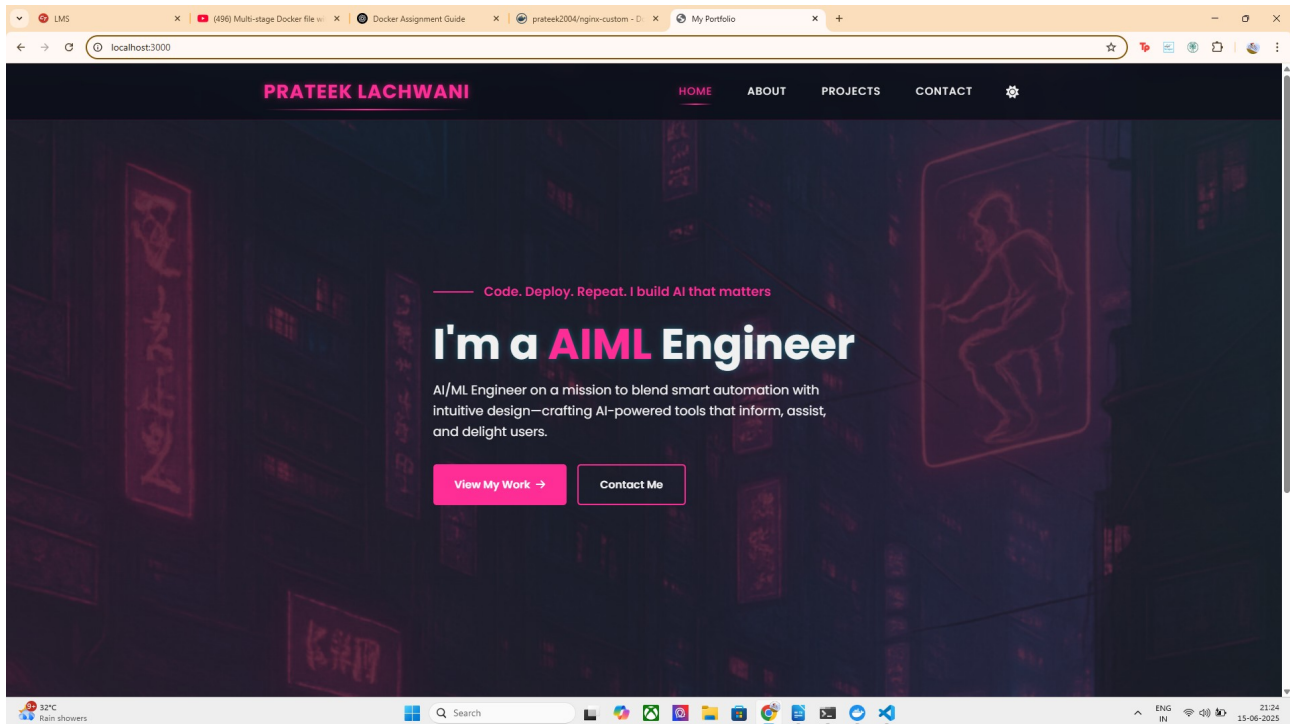
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - "5000:5000"
```

## **FOR BUILDING AND RUNNING THE CONTAINER**

```
docker-compose up --build
```

## NOW THE STEPS HAVE BEEN INCREASED TO 20

```
PS D:\OneDrive\Desktop\my portfolio> docker-compose up --build
time="2025-06-15T21:21:33+05:30" level=warning msg="D:\OneDrive\Desktop\my portfolio\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[*] Building 43.6s (22/22) FINISHED
=> [backend internal] load build definition from Dockerfile
=> => transferring dockerfile: 137B
=> [frontend internal] load metadata for docker.io/library/node:18-alpine
=> [backend auth] library/node:pull token for registry-1.docker.io
=> [backend internal] load .dockerignore
=> => transferring context: 2B
=> [frontend builder 1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca89d9e
=> [backend internal] load build context
=> => transferring context: 3.35MB
=> CACHED [frontend builder 2/5] WORKDIR /app
=> [backend 3/4] COPY . .
=> [backend 4/4] RUN npm install
=> [backend] exporting to image
=> => exporting layers
=> => writing image sha256:f3c8e4a635f3e59b044ce394bbd851958353d81178d75c3b855dce91c598aef1
=> => naming to docker.io/library/myportfolio-backend
=> [backend] resolving provenance for metadata file
=> [frontend internal] load build definition from Dockerfile
=> => transferring dockerfile: 318B
=> [frontend internal] load metadata for docker.io/library/nginx:alpine
=> [frontend internal] load .dockerignore
=> => transferring context: 2B
=> CACHED [frontend stage-1 1/2] FROM docker.io/library/nginx:alpine
=> [frontend internal] load build context
=> => transferring context: 3.61MB
=> [frontend builder 3/5] COPY . .
=> [frontend builder 4/5] RUN npm install
=> [frontend builder 5/5] RUN npm run build
=> [frontend stage-1 2/2] COPY --from=builder /app/build /usr/share/nginx/html
=> [frontend] exporting to image
=> => exporting layers
=> => writing image sha256:cb66c6ca0a59f20883bd0b74b4157fe0977ba86f98bcb694d9afd8c862cb4c63
=> => naming to docker.io/library/myportfolio-frontend
=> [frontend] resolving provenance for metadata file
[*] Running 4/4
✓ backend Built 0.0s
✓ frontend Built 0.0s
✓ Container myportfolio-backend-1 Created 0.1s
✓ Container myportfolio-frontend-1 Recreated 0.1s
Attaching to backend-1, frontend-1
backend-1 | Server running on port 5000
backend-1 | Frontend being served from: /frontend/build
frontend-1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
frontend-1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
frontend-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
```



**NOW THE NEW IMAGE IS OPTIMIZED WHERE FRONTEND ONLY MAKES UP  
( 59.4 MB ) THATS WHY MULTI STAAGE BUILT ARE PREFFERED**

```
No images found matching "ls": did you mean "docker image ls"?
PS D:\OneDrive\Desktop\my protfolio> docker image ls
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
myprotfolio-frontend latest       cb66c6ca0e59  3 minutes ago 59.4MB
myprotfolio-backend  latest       f3c8e4a635f3  4 minutes ago 131MB
<none>              <none>       da30a22769d8  14 minutes ago 551MB
```