Docker

1. Introduction to Docker

Docker is a platform for developing, shipping, and running applications in lightweight, portable containers. Containers package an application and its dependencies together, ensuring consistent environments across development, testing, and production.

2. Key Docker Definitions

- **Docker Image**: A read-only template used to create containers. It includes the application code, libraries, and dependencies.
- **Docker Container**: A runnable instance of an image. Containers are isolated and lightweight.
- **Dockerfile**: A script containing instructions to build a Docker image.
- **Docker Engine**: The core component that enables containerization on a system.
- **Docker Compose**: A tool to define and run multi-container Docker applications using YAML files.
- **Docker Volume**: A mechanism for persisting data generated by and used by Docker containers.
- **Docker Network**: A way to connect Docker containers with each other or the outside world.

3. Docker CLI Commands

Basic Commands

- docker --version – Check Docker version
- docker info – Display system-wide information
- docker help – Get help with Docker commands

Image Management

- docker build -t <image_name> . – Build an image from a Dockerfile
- docker pull <image_name> – Download an image from Docker Hub
- docker images – List all downloaded images
- docker rmi <image_id/image name> – Remove an image

Container Management

- docker run <image> – Run a container from an image
- docker run -it <image> – Run a container in interactive mode
- docker run -d <image> – Run a container in detached mode
- docker ps – List running containers
- docker ps -a – List all containers
- docker stop <container_id/container name> – Stop a running container
- docker start <container_id/container name> – Start a stopped container
- docker rm <container_id/container name> – Remove a container

4. Docker Hub

Docker Hub is a cloud-based registry service where Docker users and partners create, test, store, and distribute container images.

Basic Commands for Docker Hub

- docker login – Authenticate with Docker Hub
- docker tag <image_id> <username>/<repository>:<tag> – Tag an image for upload
- docker push <username>/<repository> – Upload image to Docker Hub
- docker pull <username>/<repository> – Download image from Docker Hub

5. Docker Desktop

Docker Desktop is an application for Mac and Windows that provides an easy-to-use GUI and CLI tools to build and share containerized applications.

Features

- **Built-in Kubernetes** – Easily enable and run Kubernetes locally
- **Volume and Container Management GUI** – View and manage containers, images, and volumes visually
- **Developer Tools Integration** – Integrates with IDEs and CI/CD pipelines
- **Cross-platform Support** – Consistent development environments across OSes

Common Uses

- Develop and test containerized applications locally
- Use Docker CLI and Compose commands without manual setup
- Build and share containers via Docker Hub or private registries

Creating Containers using a Docker Hub Image

Step-by-step process to create and run a NGINX container using Docker Desktop (GUI and CLI):

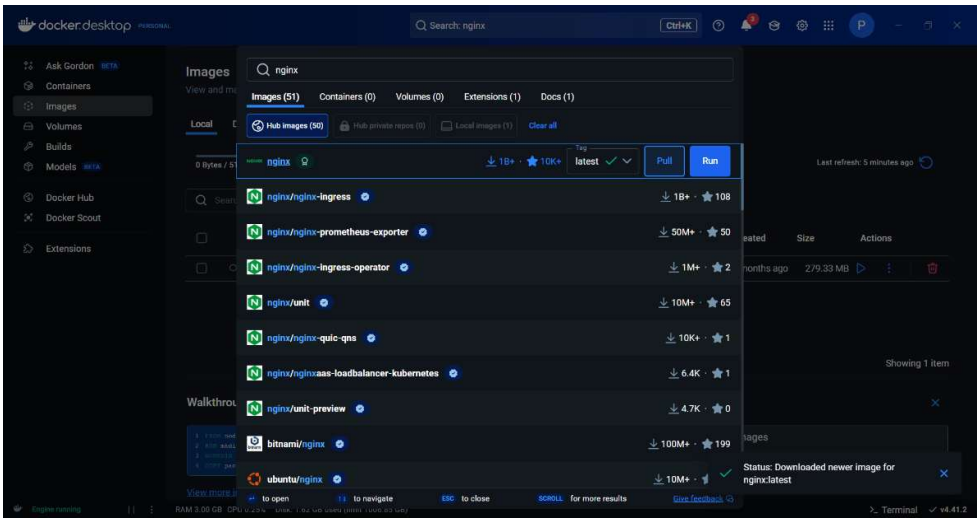
- **Using Docker Desktop GUI:**

  Step 1: Open Docker Desktop

  - Launch **Docker Desktop** on your Windows/Mac machine.

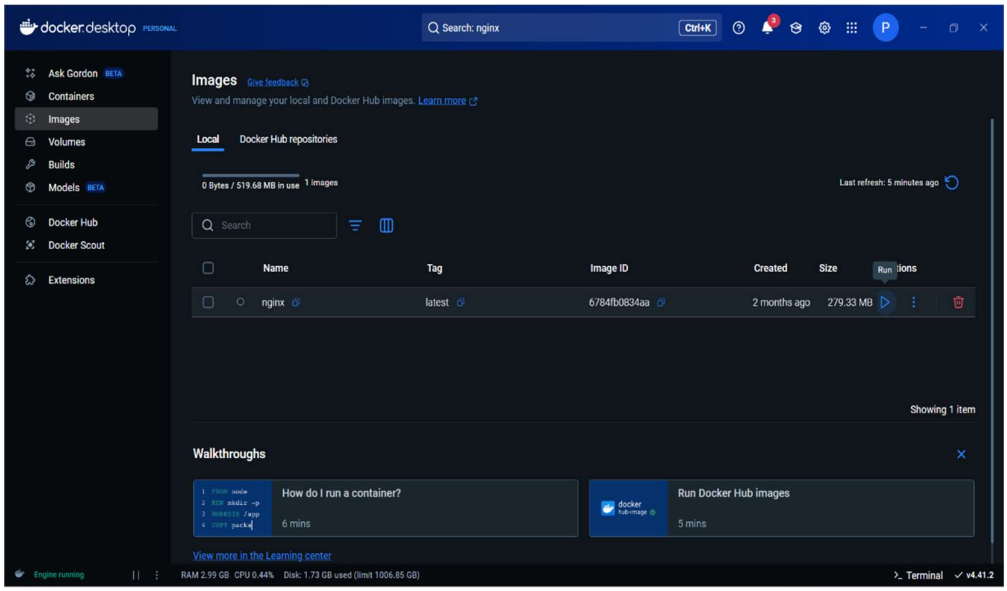
  Step 2: Search for the NGINX Image

  - Go to the **"Images"** tab on the left.
  - In the **search bar**, type nginx.
  - Select the official image (nginx) from **Docker Hub**.
  - Click **“Pull”** to download the image.



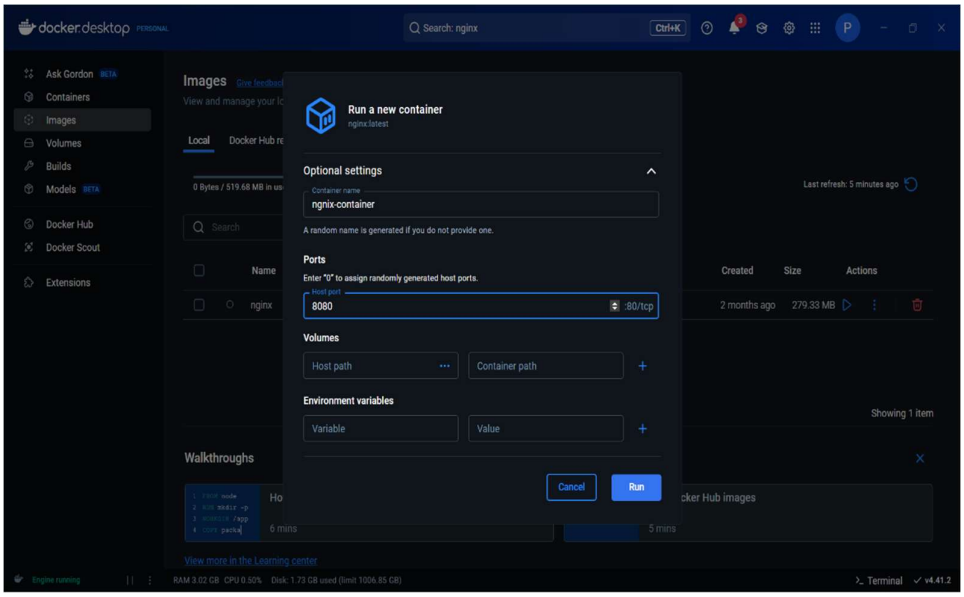
Step 3: Create a Container

- After the image is downloaded, go to the **“Containers”** or **“Images”** tab.
- Find the nginx image and click **“Run”**.



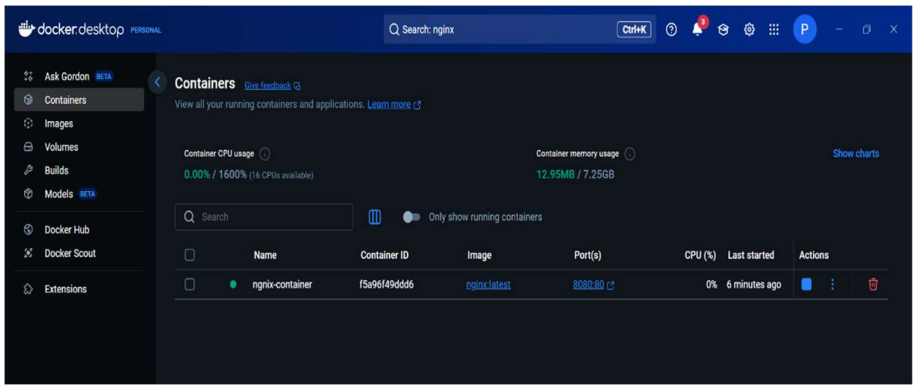
Step 4: Configure the Container

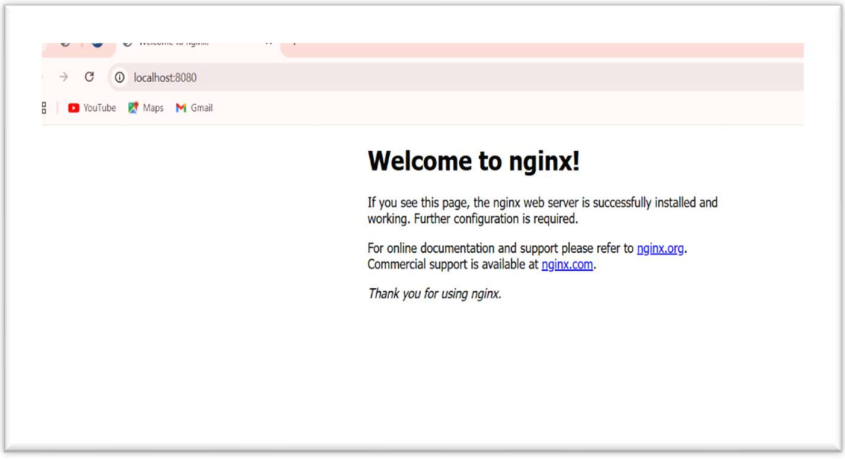
- Set **container name** (e.g., nginx-container).
- Set **ports**: expose container port 80 to host port 8080.
    - Host: 8080, Container: 80
- (Optional) Add volume mappings if you want to serve custom content.
- Click **“Run”**.



Step 5: Test the Container

- Open your browser and go to:
  - http://localhost:8080
- You'll see the **default NGINX welcome page**.





- **Using Docker CLI**
  (Terminal in Docker Desktop or system terminal):

  # Step 1: Pull the official NGINX image with tag latest

  docker pull nginx:latest

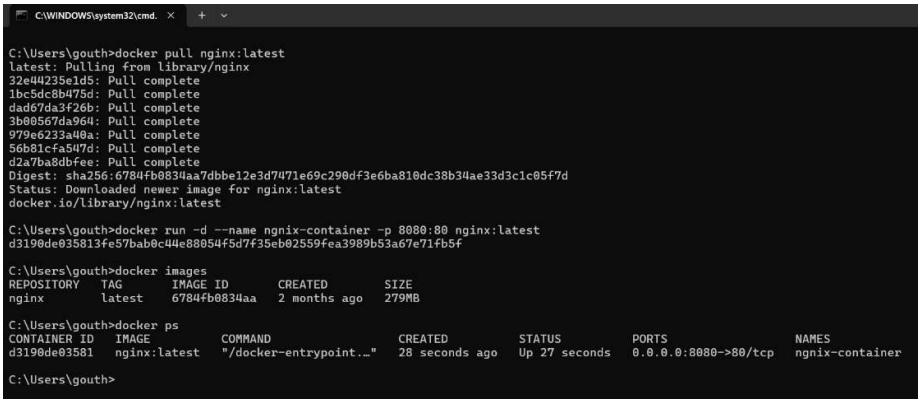
  # Step 2: Run the NGINX container

  docker run -d -p 8080:80 --name nginx-container nginx:latest

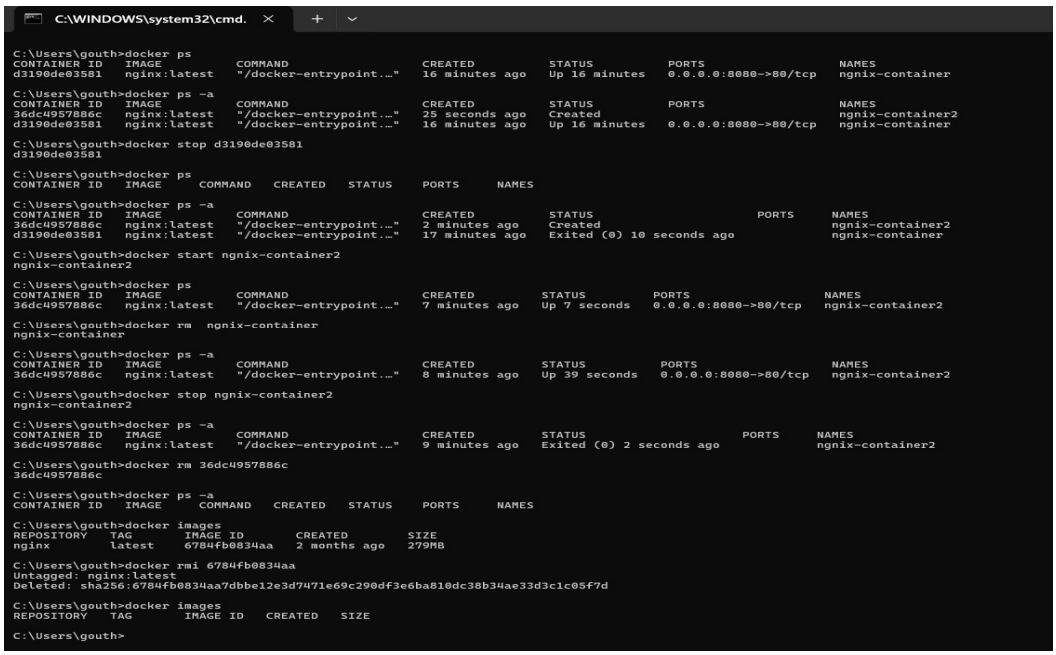
  # Step 3: Verify container is running

  docker ps

  # Step 4: Open browser and go to http://localhost:8080



- **CLI Commands for containers:**



Hosting a To-Do List Application using Docker

Prerequisites

- Docker Desktop installed on your system
- Docker Hub account (https://hub.docker.com)
- A GitHub repository for your To-Do List application (e.g., Node.js/Express app)
- Basic knowledge of terminal/command line usage

Step-by-Step Deployment

1. Clone the GitHub Repository

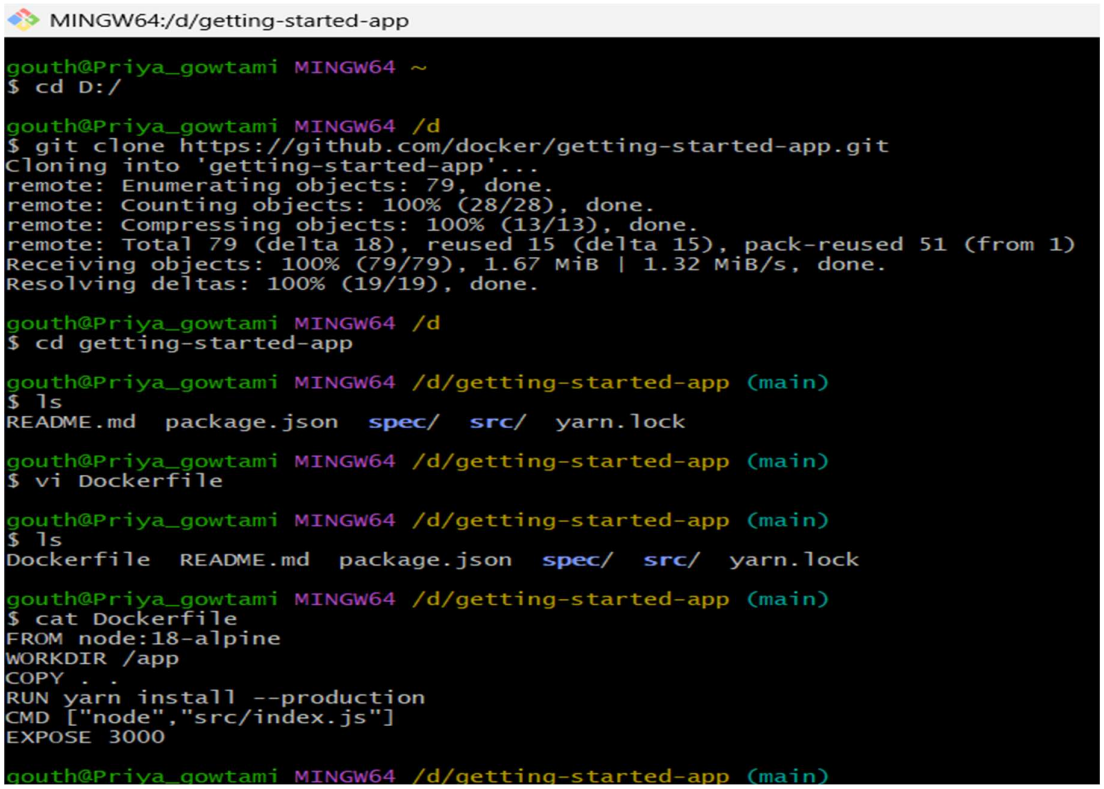
- git clone https://github.com/docker/getting-started-app.git
- cd getting-started-app

2. Create a Dockerfile

- Inside the root of your project, create a file named Dockerfile with the following content (example for a Node.js app):

```
FROM node:18-alpine   # Use an official Node.js image as the base

WORKDIR /app          # Set working directory

COPY . .              # Copy all remaining source code

RUN yarn install –production   # Install dependencies

CMD ["node","src/index.js"]    # Command to run the application

EXPOSE 3000           # Expose the application port
```

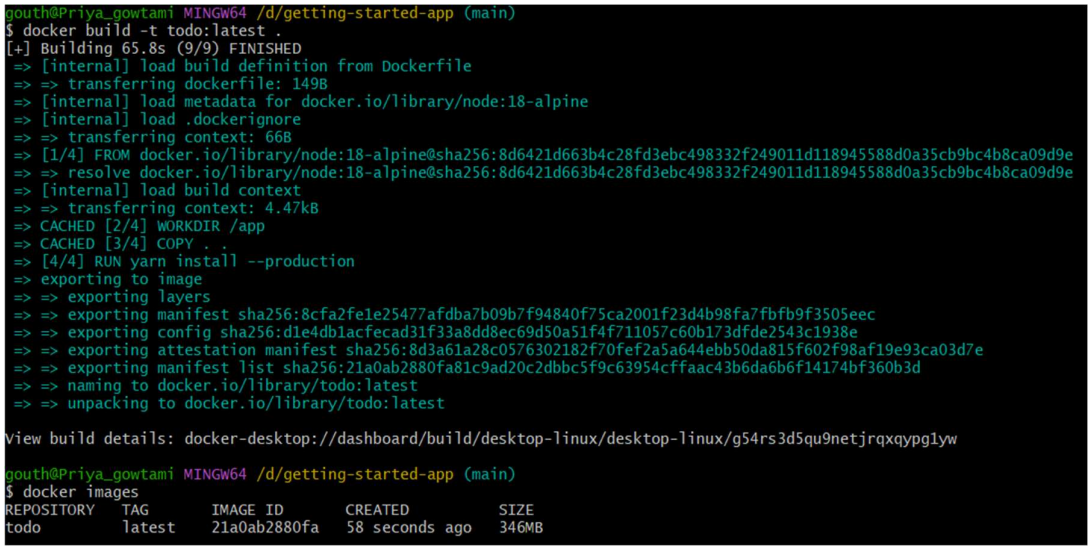
Make sure this file is saved in the root directory of your application folder.



3. Build a Docker Image from the Dockerfile

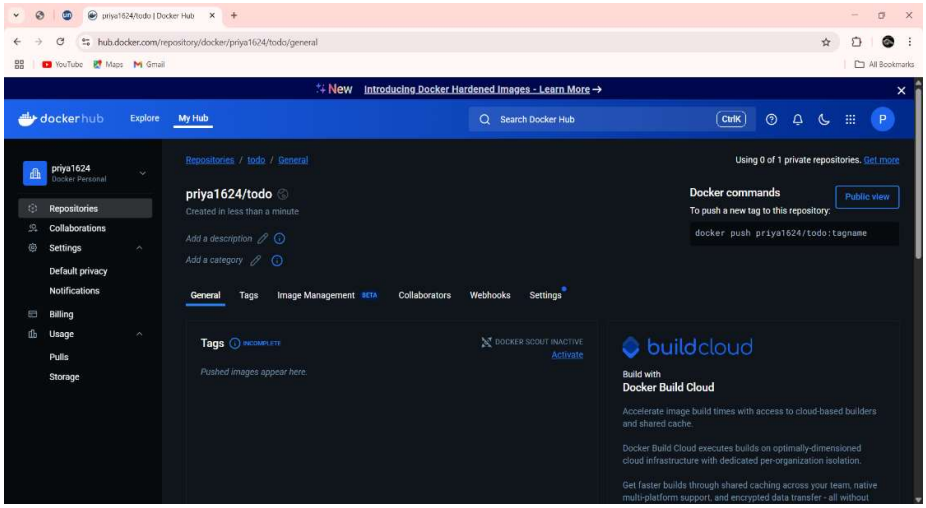
`docker build -t todo:latest .`

- This command will build the image with the tag latest using the current directory (.) and the Dockerfile.



4. Tag the Docker Image for Docker Hub

- Create a repo in docker hub and tag that name to the image.



docker tag todo:latest priya1624/todo:latest

- This command will create an image with the tagged repo name.

5. Log in to Docker Hub

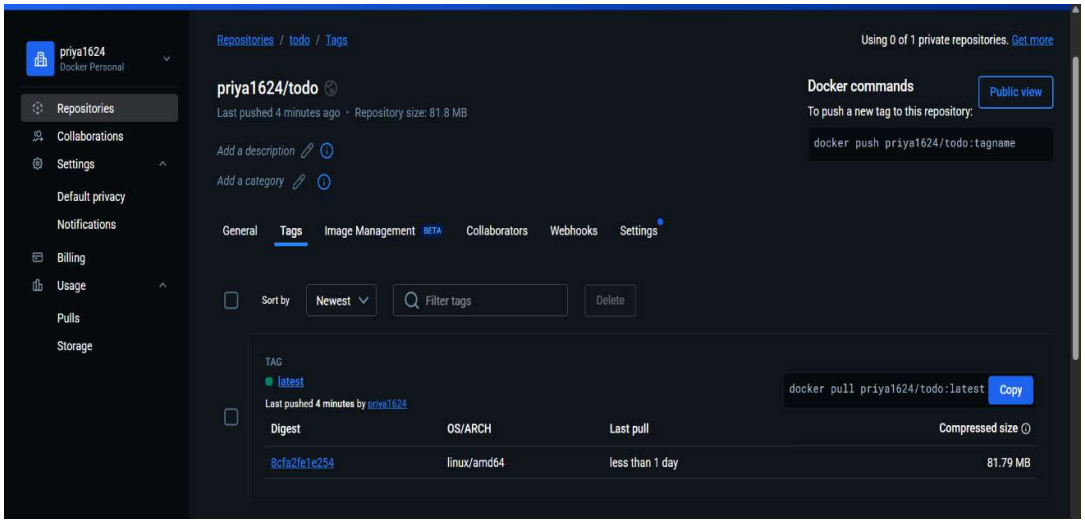
- docker login
- Provide your Docker Hub username and password when prompted.

6. Push the Docker Image to Docker Hub

docker push priya1624/todo:latest

- After the push completes, the image will be available in your Docker Hub repository.

```
gouth@Priya_gowtami MINGW64 /d/getting-started-app (main)
$ docker tag todo:latest priya1624/todo:latest

gouth@Priya_gowtami MINGW64 /d/getting-started-app (main)
$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
todo                latest    21a0ab2880fa   7 minutes ago    346MB
priya1624/todo      latest    21a0ab2880fa   7 minutes ago    346MB

gouth@Priya_gowtami MINGW64 /d/getting-started-app (main)
$ docker login
Authenticating with existing credentials... [Username: priya1624]

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


Login Succeeded

gouth@Priya_gowtami MINGW64 /d/getting-started-app (main)
$ docker push priya1624/todo:latest
The push refers to repository [docker.io/priya1624/todo]
f18232174bc9: Mounted from priya1624/priya-repo1
dd71dde834b5: Mounted from priya1624/priya-repo1
8b221aaf91b2: Pushed
7372b603f87d: Pushed
b02f5c4fb64b: Pushed
1e5a4c89cee5: Mounted from priya1624/priya-repo1
25ff2da83641: Mounted from priya1624/priya-repo1
a098521b0b23: Mounted from priya1624/priya-repo1
latest: digest: sha256:21a0ab2880fa81c9ad20c2dbbc5f9c63954cffaac43b6da6b6f14174bf360b3d size: 856
```



Deploying the App on Another Machine or Server

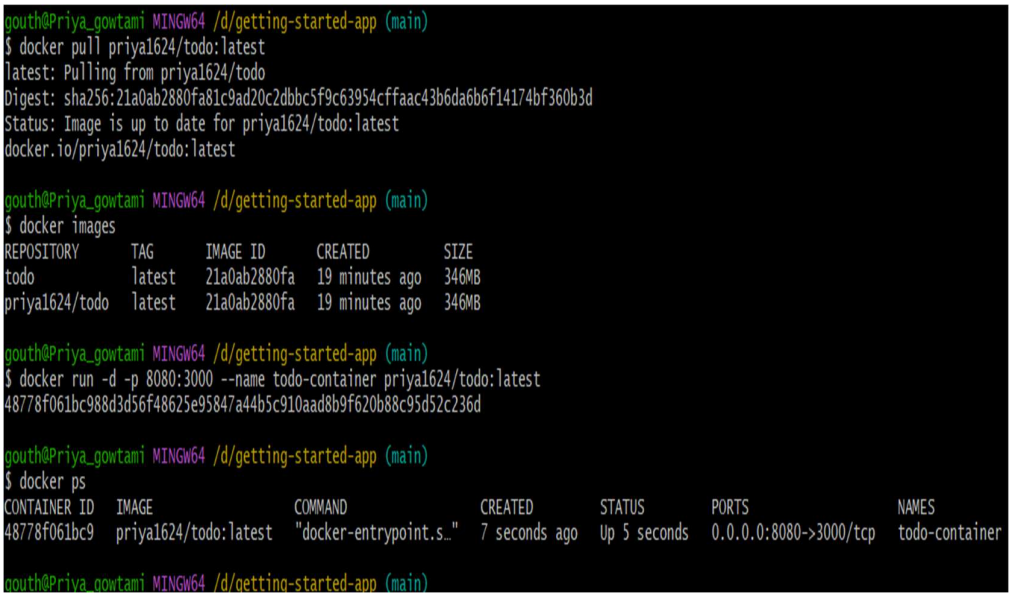
7. Pull the Docker Image from Docker Hub

```
docker pull priya1624/todo:latest
```

8. Run the Container from the Pulled Image

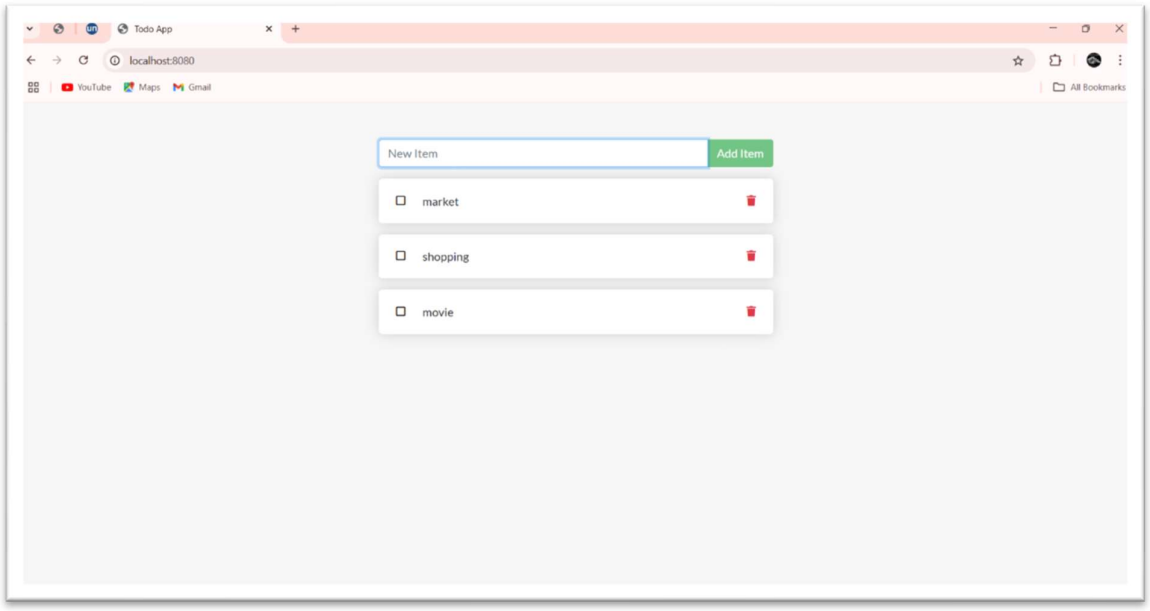
```
docker run -d -p 8080:3000 --name todo-container priya1624/todo-app:latest
```

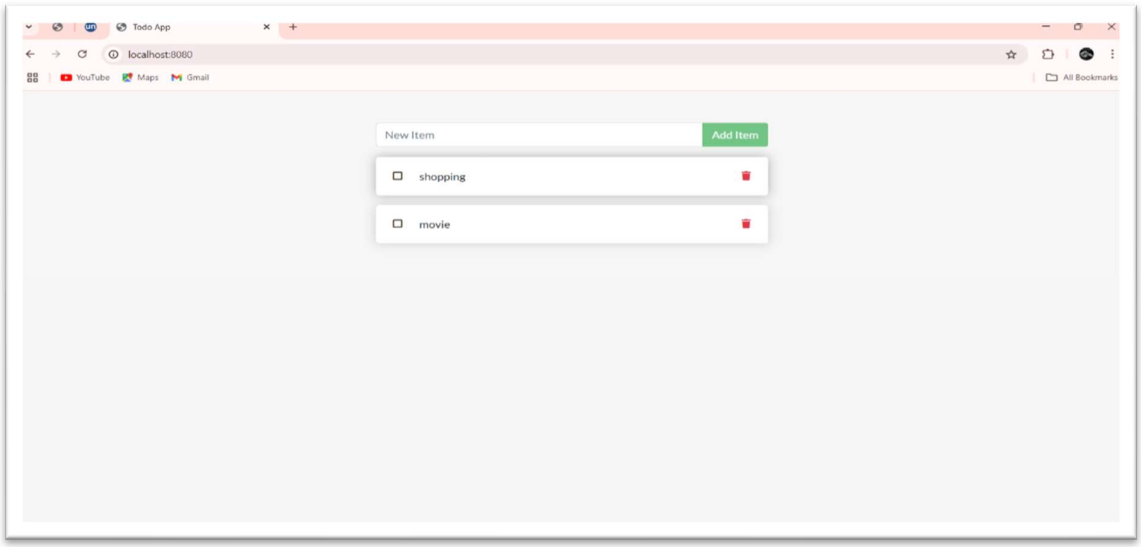
- This will start the container in detached mode and expose it on port 8080 (host) mapping to port 3000 (container).



9. Access the Application

- Open your browser and visit:
- http://localhost:8080
- You should see your To-Do List application running.





✓ Hosted the To-Do application successfully by using Docker