

Now you can see that after running the app on terminal , it also provide localhost link. But if we click on it so we will not navigate to it. So why this happening ?

* Iski waja yeh haka yeh toh thk hai kay humnay container may port ko expose krdia and Docker also know kay humari application **5173** port pa run hogi but jo humari host machine/server hai it don’t know kay kis port pay our container will run kiu kay container ek isolated environment ma work krta hai isi lia it don’t expose its port , so we have to explicitly tell the host machine kay container kis port pa work krega.

You're absolutely on the right track with your explanation! Let me clarify and expand on it:

**\*\*Why Can't You Access the Application via Localhost?\*\***

**- \*\*Container Isolation\*\*:**

- Docker containers run in an isolated environment, which means that the network inside the container is separate from the host machine's network. The application inside the container might be running on port `5173`, but this port is not automatically accessible from the host machine because of this isolation.

**- \*\*Port Mapping\*\*:**

- Even though the application is running on `localhost:5173` inside the container, the host machine doesn’t know about it unless you explicitly map the container’s port to a port on the host machine.

- Without this mapping, trying to access `http://localhost:5173/` from your browser on the host machine won’t work because the host machine doesn’t know to forward traffic on port `5173` to the container.

**\*\*Solution: Port Mapping with `-p` Flag\*\***

To make the application accessible from your host machine, you need to map the container’s port to a port on the host. This is done using the `-p` flag when running the Docker container.

\*\*How to Map Ports\*\*:

```

**docker run -p 5173:5173 react-docker**

```

- \*\*`-p 5173:5173`\*\*: This maps port `5173` on your host machine to port `5173` in the container.

- The first `5173` is the port on your \*\*host machine\*\*.

- The second `5173` is the port inside the \*\*Docker container\*\* where your application is running.

**\*\*What Happens After Port Mapping?\*\***

- \*\*Host to Container Traffic\*\*: With this mapping, when you visit `http://localhost:5173/` on your host machine, Docker knows to forward this request to port `5173` inside the container, allowing you to access your application.

**Summary:**

Ab basic yeh hua yeh kay jab humnay port mapping ki using **-p** flag toh in simple terms humnay host machine ko yeh btadia kay jasay hi [**http://localhost:5173**](http://localhost:5173)iss URL say koi request ayegi toh port **5173** pa jo container hai usko request forward hojayegi, then Docker will receive the request and request to Container on 5173 port then Container will return the application running inside it as response.

**What is port Mapping in Docker in detail ?**

\*\*Port Mapping\*\* is a way to make an application running inside a Docker container accessible from outside the container (e.g., from your local machine or the internet).

**\*\*How It Works\*\*:**

- \*\*Inside the Container\*\*: Your application runs on a specific port, like `5173`.

- \*\*Outside the Container\*\*: By default, this port is not accessible from your local machine or the outside world because the container is isolated.

**\*\*Port Mapping with `-p`\*\*:**

- When you run a container, you use the `-p` flag to map a port on your local machine (host) to a port inside the container.

- Example: `-p 8080:80`

- This maps port `80` inside the container to port `8080` on your local machine.

- Now, if your application is running on port `80` inside the container, you can access it via `http://localhost:8080` on your local machine.

**\*\*Why It’s Important\*\*:**

- \*\*Access Your App\*\*: Port mapping allows you to interact with your application from outside the container, making it accessible for development, testing, or production use.

**\*\*Simple Analogy\*\*:**

Think of port mapping as connecting a specific "door" on your local machine to a "door" inside the container. When you knock on the door on your local machine, it automatically opens the corresponding door inside the container, allowing you to interact with the application running th

In summary, \*\*port mapping\*\* connects ports from your Docker container to your host machine, making containerized applications accessible from outside the container.

ere.

Q- it means that A/c to the above example which is : -p 8080:80 when we do port mapping means to tell the host machine that when anyone try to access http://localhost:8080 this URL you should send the request to container running at port 80 ?

**\*\*Explanation Based on Your Example (`-p 8080:80`)\*\*:**

- \*\*Host Machine\*\*: When someone tries to access `http://localhost:8080` on your host machine (your computer), the host machine receives the request on port `8080`.

- \*\*Port Mapping\*\*: The `-p 8080:80` option tells Docker to map port `8080` on the host machine to port `80` inside the Docker container.

- \*\*Request Forwarding\*\*: When the host machine receives a request on `http://localhost:8080`, it forwards that request to the Docker container's port `80`.

- \*\*Container\*\*: Inside the container, your application is listening on port `80`, so it receives the request and processes it.

**\*\*Summary\*\*:**

- Yes, you're correct! Port mapping like `-p 8080:80` instructs the host machine to forward requests from `http://localhost:8080` to the container's port `80`, where the application is running.