⚠️ Important Note on Docker Port Mapping

Docker containers run their own isolated network environment, which means the ports inside a container are separate from the ports on your host machine (your computer).

To allow your computer or other devices to access a service running inside a container (like a web server), you need to **map the container's port to a port on your host machine**. This is done using the -p (or --publish) option when you start a container.

How Port Mapping Works

- The syntax for port mapping is:

sudo docker run -d –name containername -p <host_port>:<container_port> img.name

where:

<container_port> is the port number inside the container where the service is running.
<host_port> is the port on your computer that you want to use to access the service.

Example:

sudo docker run -d -p 8080:80 nginx

This command maps port **80** inside the container (the default port nginx listens to) to port **8080** on your host machine.
So, when you visit http://localhost:8080 on your browser, you are accessing the nginx server running inside the container.

Important Points to Remember

- **One host port can only be mapped to one container at a time.**
  If you try to map two different containers to the same host port (for example, both containers mapping port 80 on the host), Docker will give an error because the port is already in use.
- You can map many containers to the same container port as long as the host ports are different. For example:
  - Container A: -p 8080:80
  - Container B: -p 8081:80
    Both containers use port 80 internally, but your host machine uses different ports (8080 and 8081) to access them separately.
- If you don't specify port mapping, the service inside the container will not be accessible from outside the container.
- Port mapping is essential when you want to access container services via your local machine or expose them over a network.