Docker Hands-On Lab 2 — Port Mapping

Trainer: Sandeep Kumar Sharma

Duration: 30–45 minutes

Goal: In this lab, you will learn about Docker port mapping — what it is, why we need it, and how to use it effectively.

---

1) Understanding Port Mapping

When you start a container, it runs in an isolated network environment. This means that the services running inside the container (like Nginx, Apache, or a Node.js app) are not directly accessible from your host machine or the internet unless you explicitly expose or map ports.

In simple terms: Port mapping allows us to connect a port on the host system (your computer or VM) to a port inside the container.

Example: If an Nginx container listens on port **80** inside the container, and you want to access it from your browser on port **8080**, you can map host port **8080** → container port **80**.

```
Host:8080  --->  Container:80
```

---

2) Why Do We Need Port Mapping?

- Containers are isolated by design — their internal network ports are not visible outside.
- Port mapping helps us access containerized applications from:
- The host machine
- Other machines on the same network
- The internet (if firewall rules allow)
- Without port mapping, your containerized app would run but be unreachable.

Example scenario: You deploy a web application using Docker. It works fine inside the container, but when you try to open `http://localhost:8080`, it fails — because the app is listening on port 80 inside the container. Port mapping solves this issue.

---

3) How to Do Port Mapping

The syntax to map ports when running a container is:

```
docker run -p <host_port>:<container_port> <image_name>
```

- **host_port:** The port on your local system.
- **container_port:** The port inside the container where your service listens.

You can map multiple ports using multiple `-p` flags.

Example:

```
docker run -p 8080:80 -p 4433:443 nginx
```

This will map:

- Host port 8080 → Container port 80 (HTTP)
- Host port 4433 → Container port 443 (HTTPS)

---

4) Lab Exercise — Demonstrating Port Mapping

Step 1: Pull and Run an Nginx container with Port Mapping

```
# Pull the latest nginx image
docker pull nginx:latest

# Run nginx in detached mode, mapping host port 8080 to container port 80
docker run -d --name web-demo -p 8080:80 nginx
```

Step 2: Verify the Container is Running

```
docker ps
```

You should see something like:

```
CONTAINER ID   IMAGE     COMMAND                  PORTS                   NAMES
1a2b3c4d5e6f   nginx     "/docker-entrypoint.…"   0.0.0.0:8080->80/tcp   web-
demo
```

Step 3: Test Access from the Host

Now open your browser and go to:

```
http://localhost:8080
```

You should see the **Nginx welcome page**.

If you are using a cloud VM, replace `localhost` with your VM's public IP.

Step 4: Inspect Port Mapping

To verify that the mapping was created correctly:

```
docker port web-demo
```

You should get output similar to:

```
80/tcp -> 0.0.0.0:8080
```

You can also inspect it in detail:

```
docker inspect web-demo | grep -i port -A 5
```

Step 5: Try Accessing Without Mapping (Optional Test)

Run the same image again but **without** specifying `-p`.

```
docker run -d --name web-demo2 nginx
```

Now, check its ports:

```
docker ps
```

You'll see no port mapping listed.

Try opening `http://localhost` or `http://<vm-ip>` — you won't be able to access it, because there's no host-to-container mapping.

This demonstrates the importance of port mapping.

Step 6: Stop and Remove Containers

```
docker stop web-demo web-demo2
docker rm web-demo web-demo2
```

---

5) Small Practice Exercises

1. Run an **httpd (Apache)** container and map port **8081** (host) → **80** (container). Test it from your browser.
2. Run two **nginx** containers simultaneously:
3. One on port 8080
4. Another on port 9090 Confirm both are reachable on their respective ports.
5. Inspect both containers and list their port mappings using `docker port`.

---

6) Common Troubleshooting

- **Port already in use:** Another container or service might already be using that port.

  ```
  docker ps
  sudo lsof -i :8080
  ```

- **Firewall blocking external access:** Allow inbound traffic for that port (e.g., on Azure or AWS security group).
- **App not reachable:** Check container logs.

  ```
  docker logs <container_name>
  ```

---

7) Summary

- Port mapping connects the outside world (your host) to services running inside containers.
- Syntax: `-p <host_port>:<container_port>`
- Always check running containers with `docker ps` to confirm mapping.
- Without port mapping, your containerized apps are isolated and cannot be accessed externally.

✅Lab Checklist

-

Next, we can explore **volume mapping** in Docker — where we map host directories into containers for data persistence. Would you like me to create that as the next lab?