Hands-On Lab (HOL)

Build Docker Image from a Running Container & Push to Docker Hub

Author: Dr. Sandeep Kumar Sharma

🎯Learning Objectives

By the end of this lab, participants will be able to:

- Understand what a Docker Image is and why it is required
- Understand the concept of creating an image from a running container
- Convert a running container into a reusable Docker image
- Tag a Docker image correctly for Docker Hub
- Push a custom Docker image to Docker Hub

Learning Outcomes

After completing this HOL, you will:

- Clearly differentiate between Docker Image and Docker Container
- Know real-world use cases for creating images from containers
- Gain hands-on experience with `docker commit`, `docker tag`, and `docker push`
- Be confident in sharing your custom images via Docker Hub

Conceptual Understanding

What is a Docker Image?

A **Docker Image** is a lightweight, immutable, and executable package that contains:

- Application code
- Runtime environment
- Libraries and dependencies
- Configuration files

👉Think of a Docker image as a **template or blueprint** used to create containers.

What is a Docker Container?

A **Docker Container** is a running instance of a Docker image. Containers are:

- Mutable
- Ephemeral (can be deleted and recreated anytime)

Why Create an Image from a Container?

Although best practice is to use a **Dockerfile**, there are valid scenarios where creating an image from a container makes sense:

🏐Use Cases

- When manual configuration is done inside a container (installing packages, tools)
- For quick prototyping or PoC environments
- For capturing a troubleshooting or debug environment
- For beginners to understand image layering concepts

👉`docker commit` captures the **current state** of a container and converts it into an image.

🧨Lab Prerequisites

- Ubuntu 22.04
- Docker installed and running
- Docker Hub account
- Internet connectivity

Hands-On Lab Steps

Step 1: Clean Up Existing Containers

```
docker rm -f $(docker ps -aq)
```

Explanation:

- Removes all running and stopped containers to start with a clean environment

---

Step 2: Verify No Containers Are Running

```
docker ps
```

Explanation:

- Confirms that no active containers are present

Step 3: Run a Base Ubuntu Container

```
docker run -it --name basecontainer ubuntu
```

Explanation:

- Creates and starts an interactive Ubuntu container
- This container will act as our base environment

Step 4: Customize the Container

Inside the container, you may run commands like:

```
apt update -y
touch file1 file2 file3
mkdir dir dir2 dir3
echo "this is my base container" >file1
apt install git -y
```

Explanation:

- Simulates real-world changes inside a container

Exit the container:

```
exit
```

Step 5: List Docker Images

```
docker images
```

Explanation:

- Displays available images (Ubuntu image will be listed)

Step 6: Verify Running Containers

```
docker ps
```

Explanation:

- Confirms container status

Step 7: Create an Image from the Container

```
docker container commit -a "sandeep" -m "this is my new image" basecontainer newimage
```

Explanation:

- `-a` → Author name
- `-m` → Commit message
- `basecontainer` → Source container
- `newimage` → Target image name

Step 8: Verify the New Image

```
docker images
```

Explanation:

- Confirms creation of the new custom image

Step 9: Inspect the Image

```
docker inspect newimage
```

Explanation:

- Displays detailed metadata of the image

Step 10: Run a Container from the New Image

```
docker run -it --name newcontainer newimage
```

Explanation:

- Validates that the image works correctly

Exit the container after verification.

Push Image to Docker Hub

Step 11: Login to Docker Hub

```
docker login
```

Explanation:

- Authenticates Docker CLI with Docker Hub

Step 12: Tag the Image for Docker Hub

```
docker tag newimage sandeep289/newimage
```

Explanation:

- Docker Hub format: `username/repository:tag`
- `latest` tag is used by default

Step 13: Verify Tagged Image

```
docker images
```

Explanation:

- Confirms correct tagging

Step 14: Push Image to Docker Hub

```
docker push sandeep289/newimage
```

Explanation:

- Uploads the image to Docker Hub
- Image becomes publicly available (unless repository is private)

Step 15: Verify on Docker Hub Portal

- Login to Docker Hub
- Navigate to **Repositories**
- Confirm `newimage` is available

Summary

- Docker images are reusable blueprints
- Containers are running instances of images
- `docker commit` captures container state as an image
- Docker Hub allows image sharing and versioning

Best Practice Note

In production environments, **Dockerfile-based image creation is recommended**. `docker commit` is best suited for learning, debugging, and rapid experimentation.

👂 **Congratulations! You have successfully created and pushed a Docker image to Docker Hub.**