Docker Hands-On Lab 1 — Docker Basics

Trainer: Sandeep Kumar Sharma

Duration: 30–45 minutes

Goal: By the end of this lab you will be able to: - Install Docker (Ubuntu example) - Pull a Docker image - Create (run) a container - Describe and inspect containers - Attach / enter a running container - Stop and start a container

---

Lab setup

- You need a Linux VM (Ubuntu 20.04 / 22.04 recommended) with sudo access.
- Terminal access (SSH or local terminal).

Note: commands below use `sudo`. If your user is in the `docker` group and you can run docker without `sudo`, you may omit it.

---

1) Install Docker (Ubuntu example)

Run these commands step-by-step.

```
# 1. Update package index
sudo apt update

# 2. Install prerequisites
sudo apt install -y ca-certificates curl gnupg lsb-release

# 3. Add Docker's official GPG key
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -
o /etc/apt/keyrings/docker.gpg

# 4. Set up the repository
echo
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/
docker.gpg]
  https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
  | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 5. Install Docker Engine
sudo apt update
```

```
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

# 6. Start and enable the docker service
sudo systemctl enable --now docker

# 7. Verify installation
sudo docker version
sudo docker run --rm hello-world
```

Checkpoint: `hello-world` should print a message indicating Docker can pull and run containers.

---

2) Pull an image and create a container

We'll use the lightweight `nginx` image as an example.

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

# Run a container from the image (named `my-nginx`) in detached mode and map
port 8080 to container port 80
sudo docker run -d --name my-nginx -p 8080:80 nginx:latest

# List running containers
sudo docker ps
```

How to test: open a browser and go to `http://<VM-IP>:8080` — you should see the default Nginx page.

---

3) Describe (inspect) a container

```
# Show running containers (short summary)
sudo docker ps

# Show all containers (including stopped)
sudo docker ps -a

# Get detailed low-level information about the container
sudo docker inspect my-nginx

# View recent logs from the container
sudo docker logs my-nginx
```

```
# Follow logs in realtime
sudo docker logs -f my-nginx
```

Tip: `docker inspect` returns JSON — it contains IP addresses, mounts, environment variables, and other detailed metadata.

---

4) Attach / enter a running container

There are two common approaches: `docker attach` and `docker exec`. `docker exec` is safer for interactive shells.

```
# Enter a running container with an interactive shell (recommended)
sudo docker exec -it my-nginx /bin/bash

# If /bin/bash isn't available, try /bin/sh
sudo docker exec -it my-nginx /bin/sh

# When done inside the container, type 'exit' to go back to your host shell

# Alternative: attach the container's main process STDIN/STDOUT (less used for
shells)
sudo docker attach my-nginx
# To detach from attach without stopping the container: press Ctrl-p then Ctrl-q
```

Checkpoint: You should be able to run commands inside the container (e.g., `ls /`, `cat /etc/os-release`).

---

5) Stop and start a container

```
# Stop the container
sudo docker stop my-nginx

# Confirm it's stopped
sudo docker ps -a

# Start the container again
sudo docker start my-nginx

# Restart the container
sudo docker restart my-nginx
```

Note: Stopping keeps the container on disk; removing deletes it (see cleanup section).

---

6) Cleanup (optional)

```
# Stop the container if running
sudo docker stop my-nginx

# Remove the container
sudo docker rm my-nginx

# Remove the image (if you want to free space)
sudo docker rmi nginx:latest
```

---

7) Quick troubleshooting

- If `docker` commands say `permission denied`, either re-run with `sudo` or add your user to the `docker` group:

```
sudo usermod -aG docker $USER
# then log out and log in again (or restart the shell)
```

- If the service won't start:

```
sudo systemctl status docker
sudo journalctl -u docker --no-pager --since "10 minutes ago"
```

---

8) Small exercises for practice

1. Run an interactive Ubuntu container, create a file `/hello.txt` inside it, then exit and inspect the file using `docker cp` or by starting the container and `exec` into it.
2. Run a container named `my-test` from `alpine` image, sleep for 60 seconds (`docker run --rm --name my-test alpine sh -c "sleep 60"`) and watch it appear and disappear in `docker ps` and `docker ps -a`.
3. Start `nginx` on port `8081` as another container and ensure both nginx containers don't conflict on host ports.

---

9) Lab checklist (mark when done)

- [ ] Docker installed and `hello-world` succeeded
- [ ] Pulled `nginx` and started `my-nginx` on port 8080
- [ ] Inspected container metadata with `docker inspect`
- [ ] Entered container with `docker exec -it`
- [ ] Stopped and started the container
- [ ] Performed cleanup

If you want, I can add additional steps: mounting host volumes, creating a Dockerfile and building an image, or demonstrating `docker-compose` — tell me which one to add next.