Docker Tutorials

Compiled by Karma Dorji

Note: This tutorial is designed for beginners and mid-level learners of Docker. It includes step-by-step instructions for running containers, building Dockerfiles, and creating Docker Compose files.

These notes are structured for classroom teaching with detailed explanations, examples, and tips for each command.

---

1. `docker run`

Purpose: Create and start a new container from an image. Essential for running applications in isolated environments.

Syntax:

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Key Options: - `-d`: Run container in background (detached mode) - `--name <name>`: Assign a custom name to the container - `-p <host>:<container>`: Map host port to container port - `-e <VAR>=<value>`: Set environment variable - `--link <container>`: Connect to another container - `--restart <policy>`: Automatically restart the container (e.g., always, unless-stopped)

Example:

```
docker run -d --name webapp -p 8080:80 nginx:latest
```

Explanation: - Runs the `nginx:latest` image - Maps host port 8080 to container port 80 - Names the container `webapp` - Runs in detached mode

---

2. `docker ps`

Purpose: List running containers.

Syntax:

```
docker ps [OPTIONS]
```

Options: - `-a`: List all containers, including stopped ones

Example:

```
docker ps -a
```

Explanation: - Helps students see which containers are active or stopped.

---

3. `docker stop`

Purpose: Gracefully stop a running container.

Syntax:

```
docker stop <container_name_or_id>
```

Example:

```
docker stop webapp
```

Classroom Tip: - Teach students to always stop containers before removing them.

---

4. `docker rm`

Purpose: Remove containers that are no longer needed.

Syntax:

```
docker rm [OPTIONS] <container_name_or_id>
```

Options: - `-f`: Force remove running container

Example:

```
docker rm -f clickcounter
```

Explanation: - Removes container to free resources and avoid name conflicts.

---

5. `docker pull`

Purpose: Download images from Docker Hub or other registries.

Syntax:

```
docker pull <image_name>[:tag]
```

Example:

```
docker pull nginx:latest
docker pull httpd:2.4
```

Classroom Tip: - Explain difference between image and container. Pulling downloads the image without running it.

---

6. `docker images`

Purpose: List images available locally.

Syntax:

```
docker images
```

Explanation: - Shows REPOSITORY, TAG, IMAGE ID, and SIZE. - Helps students identify which images are present locally.

---

7. `docker rmi`

Purpose: Remove images to free up disk space.

Syntax:

```
docker rmi <image_name_or_id>
```

Example:

```
docker rmi nginx:latest
```

Classroom Tip: - Images must not be used by running containers before removal.

---

8. `docker tag`

Purpose: Tag an image with a new name for registry usage.

Syntax:

```
docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]
```

Example:

```
docker tag nginx:latest localhost:5000/nginx
```

Explanation: - Essential for pushing images to a registry.

---

9. `docker push`

Purpose: Push images to a registry (local or remote).

Syntax:

```
docker push <registry>/<image_name>:<tag>
```

Example:

```
docker push localhost:5000/nginx
```

Classroom Tip: - Demonstrate pushing to local registry first, then Docker Hub.

---

10. `docker inspect`

Purpose: View detailed information about containers or images.

Syntax:

```
docker inspect <container_or_image>
```

Example:

```
docker inspect webapp
```

Explanation: - Provides configuration, network settings, environment variables. - Useful for debugging.

---

11. `docker exec`

Purpose: Execute commands inside a running container.

Syntax:

```
docker exec [OPTIONS] <container> <command>
```

Example:

```
docker exec -it webapp bash
```

Classroom Tip: - Show interactive shell access inside the container.

---

12. `docker image prune`

Purpose: Remove dangling or unused images.

Syntax:

```
docker image prune -a
```

Classroom Tip: - Use regularly to free disk space during practical labs.

---

13. `docker-compose up`

Purpose: Start multi-container applications defined in a Compose file.

Syntax:

```
docker-compose up [OPTIONS]
```

Example:

```
docker-compose up -d
```

Classroom Tip: - Includes making Docker Compose files for linking multiple services. - Explain difference between `docker run` (single container) and `docker-compose` (multi-container).

---

14. `docker-compose down`

Purpose: Stop and remove containers, networks, volumes from Compose setup.

Syntax:

```
docker-compose down
```

---

15. `docker-compose config`

Purpose: Validate Compose file syntax and view full configuration.

Syntax:

```
docker-compose config
```

---

16. `docker network ls`

Purpose: List Docker networks.

Syntax:

```
docker network ls
```

Classroom Tip: - Explain bridge, host, and overlay networks.

---

17. `docker volume ls`

Purpose: List Docker volumes.

Syntax:

```
docker volume ls
```

Classroom Tip: - Explain persistent storage and its importance.

---

18. `docker logs`

Purpose: View logs of a container.

Syntax:

```
docker logs [OPTIONS] <container>
```

Example:

```
docker logs -f clickcounter
```

---

19. `docker restart`

Purpose: Restart a container.

Syntax:

```
docker restart <container>
```

---

20. `docker info`

Purpose: View system-wide Docker information.

Syntax:

```
docker info
```

Classroom Tip: - Useful for showing Docker version, storage driver, and resources.

Summary Tips for Teaching

1. Stop containers before removing images.
2. Use `docker ps -a` to view all containers.
3. Tag images before pushing to registry.
4. Use Docker Compose for multi-container apps.
5. Include practical exercises creating Dockerfiles and Compose files.
6. Regularly prune unused images and volumes.
7. Explain difference between image, container, and registry.
8. Include practical demos for each command.