Recap Sheet : Optimize Your Deployment Using Docker Containers

System & Networks

1. Understand the basics of containerization to distinguish it from virtualization.
2. Install Docker on your machine by following the setup instructions.
3. Launch your first container using the **docker run command.**
4. Manage Docker images with **docker pull and docker build.**
5. Deploy an application using Docker Compose to orchestrate services.

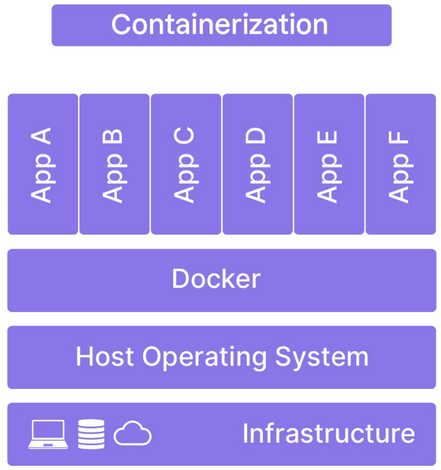
Commands 💻

```
# Install Docker on Ubuntu
sudo apt-get install docker-ce docker-ce-cli containerd.io

# Launch an interactive container
docker run -it ubuntu bash

# List running containers
docker ps

# Create an image from a Dockerfile
docker build -t myimage .
```



Best Practices 👍

- ✅ Use lightweight images to improve performance.
- ✅ Check for Docker updates to ensure security.
- ✅ Apply resource usage limits to your containers.
- ✅ Tag images with clear versions to make management easier.
- ✅ Schedule regular backups to prevent data loss.
- ✅ Document your Dockerfiles to simplify maintenance.
- ✅ Test containers locally before deploying to production.

Typical errors 👎

- ❌ Forgetting to secure exposed ports during deployment
- ❌ Failing to limit container resource usage
- ❌ Using unverified images from Docker Hub
- ❌ Ignoring container log management
- ❌ Not isolating container networks
- ❌ Overloading Dockerfiles with too many RUN instructions
- ❌ Skipping persistent volumes for critical data
- ❌ Neglecting container backups during updates

Definitions 🔍

Containerization
Isolating processes in a shared environment without a full OS.

Virtualization
Creating fully isolated virtual machines with complete operating systems.

Cgroups
A Linux feature that limits and controls resource usage by containers.

Namespaces
A mechanism that isolates processes, files, and networks within containers.

Docker Hub
An image-sharing platform for developers using Docker.