** Predefined Images

- A ready-made Docker image provided Docker hub or other registries.

Source: Mostly from docker hub (Official images or community image).

- Example :→

```
docker pull nginx
docker pull node:20
docker pull mysql:8
```

usage :

```
docker run nginx
```

Note :→ (Tip)→ Always usage check tags for specific version instead of using latest.

** Container in Interactive Terminal mode

- Purpose: Allow you to interact directly with a running container (like SSH into it).

Ex `docker run -it python_myapp`

∴ python_myapp → sum of two no.
need interact with terminal

∴ flags : -it → -i (interactive)
→ -t (allocate terminal

* Sharing Image with docker hub

Step A :- `docker login` Login into docker account.

Step-B :- Tag image for docker hub

`docker tag local_image_name dockerhub_username/repo:tag`

Step:C Push Image / Pull Image

`docker push dockerhub_username/reponame:tag`

D Pull Image

`docker pull dockerhub_username/repo_name:tag`

* Volumes (Managed by docker)

- Stored :- In docker's internal storage location (usually /var/lib/docker/volumes).
- Managed by docker.
- Best for: When you want docker to handle data location

Why need :- By default, data inside a docker container is temporary — if the container is deleted, the data is gone.
→ This is bad for things like database data, uploaded files, or logs.

When we use Volumes & / Bind mounts to:
1. Persist data outside of container lifecycle
2. Shared data b/w multiple containers
3. Enable development (edits files on host and see changes instantly in container).

Ex) Anonymous volume (automatic)

```
docker run -v /data myapp
```

- /data in container is mapped to an auto created volume. (/data → WORKDIR)

Ex) Named Volume (manual)

```
docker volume create mydata
docker run -v mydata:/data myapp
```

adv: Named volumes survive container deletion.

To list

```
docker volume ls
```

To inspect

```
docker volume inspect mydata.
```

* Bind Mounts (Direct Host Path Mapping)

- Stored at a specific path you choose on the host machine.
- Not managed by docker.

- Best for : local development (sync code changes live).
- Downside :- harder to migrate; depends on host file system structure.

Eg :-

```
docker run -v /host/path:/container/path myapp
```

or newer system

```
docker run --mount type=bind,source=/host/path,target=/container/path myapp
```