

# Experiment -7

**Aim-** To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers.

**Theory:-**

## Docker Architecture:

Docker is a containerization platform that allows developers to package applications and their dependencies into lightweight containers, ensuring consistency across different environments. Understanding Docker architecture is crucial for effectively utilizing its capabilities:

### 1. Docker Engine:

- At the heart of Docker is the Docker Engine, which comprises three major components:
  - Docker Daemon: This is a background process responsible for managing Docker objects such as images, containers, networks, and volumes. It listens for Docker API requests and handles them.
  - Docker CLI (Command Line Interface): It provides a command-line interface that allows users to interact with the Docker daemon. Developers use the CLI to build, run, and manage Docker containers and images.
  - REST API: The Docker Daemon exposes a REST API that allows interaction with the Docker Engine remotely.

### 2. Docker Images:

- Docker images are read-only templates that contain application code, libraries, dependencies, and other files needed for an application to run.
- Images are built using a Dockerfile, which contains instructions for building the image layer by layer.
- Images can be stored locally or published to a Docker registry like Docker Hub for sharing and distribution.

### 3. Docker Containers:

- Docker containers are lightweight, standalone, and executable packages that contain everything needed to run a piece of software, including the application code, runtime, system tools, system libraries, and settings.
- Containers are created from Docker images and can be started, stopped, moved, and deleted.
- Each container runs in isolation from others, but they can communicate with each other and with the host system through well-defined channels.

### 4. Docker Registries:

- Docker registries are repositories for Docker images. They allow users to store and distribute Docker images.
- Docker Hub is the default public registry provided by Docker, but users can also set up private registries for internal use.
- Docker images can be pushed to and pulled from registries, enabling seamless sharing and distribution across teams and environments.

## Container Lifecycle:

Understanding the container lifecycle is essential for managing Docker containers effectively:

#### 1. Creation:

- Containers are created using Docker images. When a container is created, it is in the "created" state.

#### 2. Running:

- Once created, containers can be started using the ``docker run`` command. A running container is in the "running" state.
- Running containers execute the application or service they contain.

#### 3. Paused:

- Containers can be paused using the ``docker pause`` command. In the paused state, a container's processes are paused, but the container's state is retained.

#### 4. Stopped:

- Containers can be stopped using the ``docker stop`` command. In the stopped state, a container's processes are halted, and the container's resources are released.

#### 5. Restarted:

- Stopped containers can be restarted using the ``docker start`` command. The container returns to the "running" state.

#### 6. Deletion:

- Containers can be deleted using the ``docker rm`` command. Once deleted, the container and its filesystem are removed from the host system.

### Installing Docker and Executing Docker Commands:

To install Docker and interact with Docker commands, follow these steps:

#### 1. Installation:

- Visit the official Docker website (<https://www.docker.com/>) and download the appropriate Docker package for your operating system.
- Follow the installation instructions provided on the website to install Docker on your system.

#### 2. Verification:

- After installation, open a terminal or command prompt and run the command ``docker --version`` to verify that Docker has been installed correctly.
- Additionally, you can run ``docker info`` to get detailed information about your Docker installation.

#### 3. Executing Docker Commands:

- Use the Docker CLI to execute various commands for managing images and interacting with containers.
- Some common Docker commands include:
  - ``docker pull <image_name>``: Pull an image from a Docker registry.
  - ``docker build -t <image_name> <path_to_Dockerfile>``: Build a Docker image using a Dockerfile.


- ``docker run <image_name>``: Create and start a container from a Docker image.
- ``docker ps``: List running containers.
- ``docker images``: List locally available Docker images.
- ``docker stop <container_id>``: Stop a running container.
- ``docker rm <container_id>``: Remove a container.
- ``docker rmi <image_id>``: Remove an image.
- Refer to the Docker documentation or use ``docker --help`` for more commands and options.

By understanding Docker architecture, the container lifecycle, and how to install Docker and execute Docker commands, you can effectively utilize Docker for containerization and application deployment.

```
C:\Users\ASHIS\Desktop>mkdir docker
C:\Users\ASHIS\Desktop>cd docker
C:\Users\ASHIS\Desktop\docker>git clone https://github.com/docker/getting-started-app.git
Cloning into 'getting-started-app'...
remote: Enumerating objects: 68, done.
remote: Counting objects: 100% (37/37), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 68 (delta 12), reused 10 (delta 10), pack-reused 31
Receiving objects: 100% (68/68), 1.75 MiB | 7.57 MiB/s, done.
Resolving deltas: 100% (12/12), done.
C:\Users\ASHIS\Desktop\docker>S|
```

```
C:\Users\ASHIS\Desktop\docker>cd getting-started-app
C:\Users\ASHIS\Desktop\docker\getting-started-app>type nul >Dockerfile
C:\Users\ASHIS\Desktop\docker\getting-started-app>Dockerfile code.
'Dockerfile' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\ASHIS\Desktop\docker\getting-started-app>ode Dockerfile
'ode' is not recognized as an internal or external command,
operable program or batch file.
```

## Dockerfile ●

C: > Users > ASHIS > Desktop > docker > getting-started-app >  Dockerfile

💡 Click here to ask Blackbox to help you code faster

```
1 # syntax=docker/dockerfile:1
2
3 FROM node:18-alpine
4 WORKDIR /app
5 COPY . .
6 RUN yarn install --production
7 CMD ["node", "src/index.js"]
8 EXPOSE 3000
```

```
C:\Users\ASHIS\Desktop\docker\getting-started-app>docker build -t getting-started .
[+] Building 0.1s (1/1) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile             0.1s
=> => transferring dockerfile: 31B                             0.0s
ERROR: failed to solve: the Dockerfile cannot be empty
```

```
C:\Users\ASHIS\Desktop\docker\getting-started-app>docker run -dp 127.0.0.1:3000:3000 getting-started
```

```

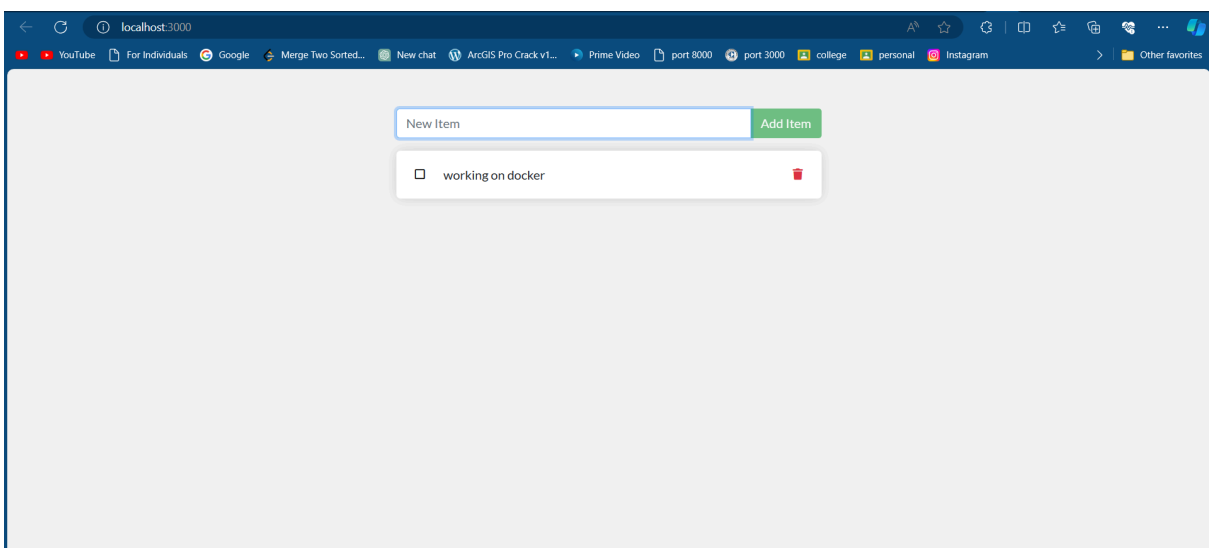
C:\Users\ASHIS\Desktop\docker\getting-started-app>docker build -t getting-started .
[+] Building 28.0s (11/11) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 188B                                0.0s
=> resolve image config for docker.io/docker/dockerfile:1        2.5s
=> docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b1 1.8s
=> => resolve docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfc 0.0s
=> => sha256:9d9c93f4b00be908ab694a4df732570bced3b 11.80MB / 11.80MB 1.5s
=> => sha256:ac85f380a63b13dfcefa89046420e1781752bab 8.40kB / 8.40kB 0.0s
=> => sha256:657fcc512c7369f4cb3d94ea329150f8daf626bc838 482B / 482B 0.0s
=> => sha256:a17ee7fff8f5e97b974f5b48f51647d2cf28d54 1.27kB / 1.27kB 0.0s
=> => extracting sha256:9d9c93f4b00be908ab694a4df732570bced3b8a96b75 0.1s
=> [internal] load metadata for docker.io/library/node:18-alpine 2.4s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                      0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:c7620fdecfefb9 9.6s
=> => resolve docker.io/library/node:18-alpine@sha256:c7620fdecfefb9 0.0s
=> => sha256:c7620fdecfefb96813da6251989780877523038 1.43kB / 1.43kB 0.0s
=> => sha256:17b35d6932abf679c57caa7b8297ded19d89760 1.16kB / 1.16kB 0.0s
=> => sha256:995e68c9d946f454675dd18204619ccd3b8e065 7.14kB / 7.14kB 0.0s
=> => sha256:4abcf20661432fb2d719aaf90656f55c287f8ca 3.41MB / 3.41MB 1.4s
=> => sha256:ff171c16ee4e8ed3a0ae3edb6fb9558f28113 40.25MB / 40.25MB 8.2s
=> => sha256:7c215e7ef3945bd4fdac5a20f1e4a29c62f28c8 2.34MB / 2.34MB 5.4s
=> => extracting sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc 0.1s
=> => sha256:3f72a72846173b6a02c1f73db9a74aca2af50fb39bf 451B / 451B 1.8s
=> => extracting sha256:ff171c16ee4e8ed3a0ae3edb6fb9558f281134a3879e 1.0s
=> => extracting sha256:7c215e7ef3945bd4fdac5a20f1e4a29c62f28c8493ad 0.0s
=> => extracting sha256:3f72a72846173b6a02c1f73db9a74aca2af50fb39bf1 0.0s
=> [internal] load build context                                  0.6s
=> => transferring context: 6.50MB                                  0.5s
=> [2/4] WORKDIR /app                                           0.1s
=> [3/4] COPY . .                                               0.1s
=> [4/4] RUN yarn install --production                          10.5s
=> exporting to image                                           0.8s
=> => exporting layers                                           0.8s

```

```

C:\Users\ASHIS\Desktop\docker\getting-started-app>docker run -dp 127.0.0.1:3000:3000 getting-started
f2048488c3f9918f18a99a810b1d02176a614d7fa836826a2037d7d98963fe24

```



New Item

Add Item

☐

working on docker

☒

Working on Networking C

docker desktop

Search for images, containers, volumes...

Ctrl+K

Sign in

Containers

Images

Volumes

Builds

Dev Environments 

BETA

Docker Scout

Extensions

Add Extensions

Containers

Container CPU usage ⓘ

0.00% / 1600% (16 CPUs available)

Container memory usage ⓘ

19.97MB / 6.54GB

Show charts

Search

Only show running containers

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>		<div><div>focused_he</div><div>f2048488c3f9</div></div> <div>getting-started</div>	Running	0%	3000:3000	2 minutes ago	<div><div></div><div></div><div></div></div>

Showing 1 item

Walkthroughs

Multi-container applications

Containerize your application

```
C:\Users\ASHIS\Desktop\docker\getting-started-a
CONTAINER ID    IMAGE           COMMAND
f2048488c3f9    getting-started "docker-entryp
```