<table>
<tr><td colspan="3">Name of Student: Pushkar Sane</td></tr>
<tr><td colspan="2">Roll Number: 45</td><td>Lab Assignment Number: 6</td></tr>
<tr><td colspan="3">Title of Lab Assignment: To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers.</td></tr>
<tr><td colspan="2">DOP: 18-03-2024</td><td>DOS: 22-03-2024</td></tr>
<tr><td>CO Mapped:
CO4</td><td>PO Mapped:
PO2, PO3, PO5, PSO1, PSO2</td><td>Signature:</td></tr>
</table>

Practical No. 6

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

Introduction:

Docker Architecture

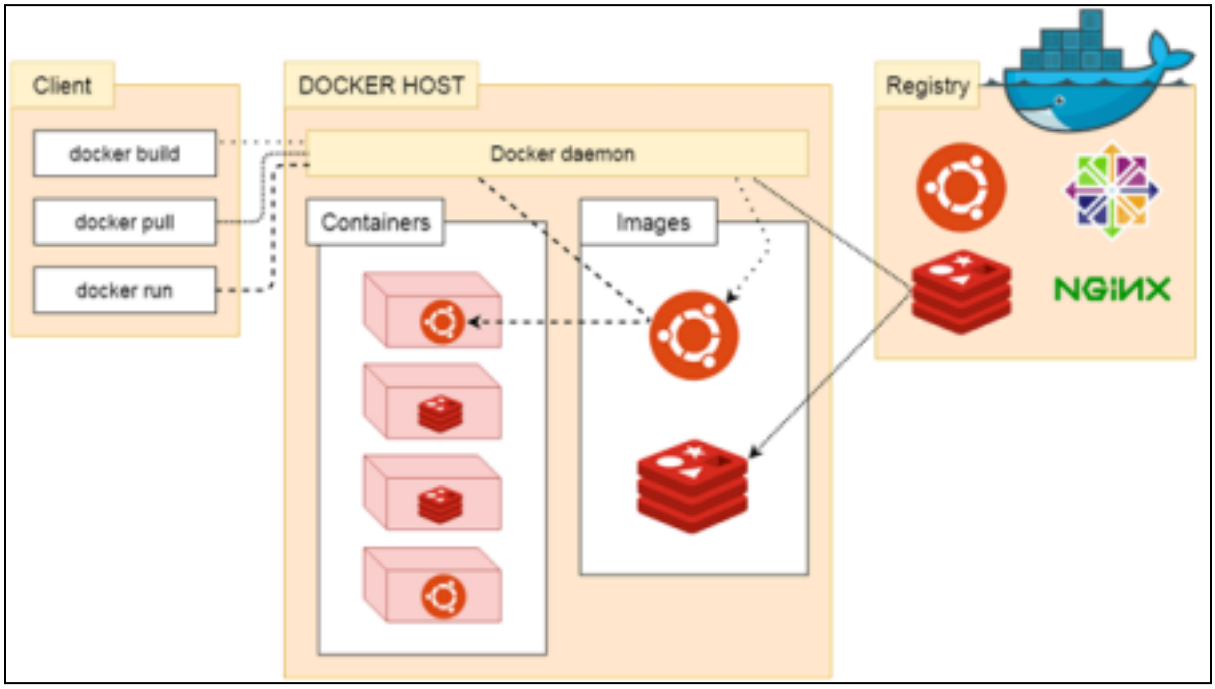
Before learning the Docker architecture, first, you should know about the Docker Daemon.

What is Docker daemon?

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storages.

Docker architecture

Docker follows Client-Server architecture, which includes the three main components that are Docker Client, Docker Host, and Docker Registry.



1. **Docker Client**

   Docker client uses **commands** and **REST APIs** to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

   Note: Docker Client has an ability to communicate with more than one docker daemon. Docker Client uses Command Line Interface (CLI) to run the following commands -

   docker build

   docker pull

   docker run

2. **Docker Host**

   Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

3. **Docker Registry**

   Docker Registry manages and stores the Docker images.

   There are two types of registries in the Docker -

   - Public Registry - Public Registry is also called Docker hub.
   - Private Registry - It is used to share images within the enterprise.

4. **Docker Objects**

   There are the following Docker Objects -

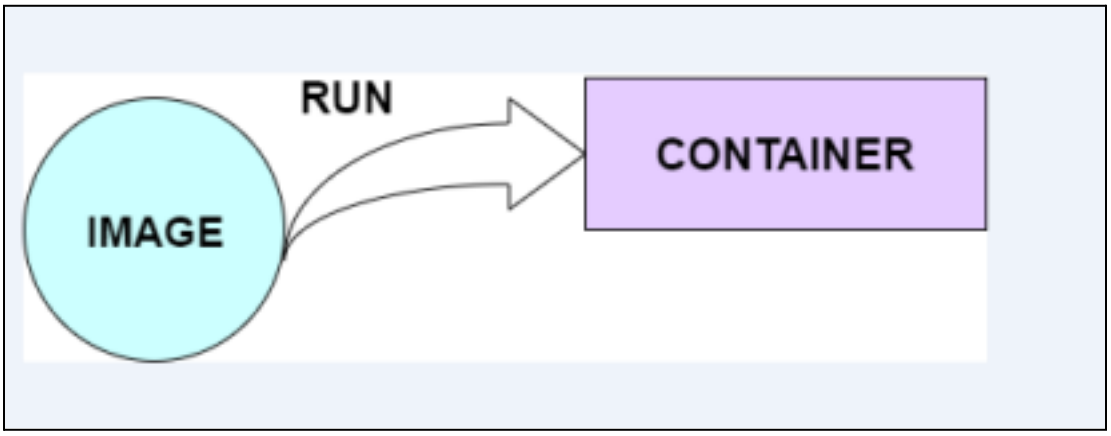
   Docker Images

   Docker images are the **read-only binary templates** used to create Docker Containers. It uses a private container registry to share container images within the enterprise and also uses a public container registry to share container images within the whole world. Metadata is also used by docket images to describe the container's abilities.

   Docker Containers

Containers are the structural units of Docker, which is used to hold the entire package that is needed to run the application. The advantage of containers is that it requires very less resources.

In other words, we can say that the image is a template, and the container is a copy of that template.



Docker Networking

Using Docker Networking, an isolated package can be communicated. Docker contains the following network drivers -

- Bridge - Bridge is a default network driver for the container. It is used when multiple dockers communicate with the same docker host.
- Host - It is used when we don't need network isolation between the container and the host. o None - It disables all the networking.
- Overlay - Overlay offers Swarm services to communicate with each other. It enables containers to run on the different docker hosts.

Macvlan - Macvlan is used when we want to assign MAC addresses to the containers.

Docker Storage

Docker Storage is used to store data on the container. Docker offers the following options for the Storage -

- Data Volume - Data Volume provides the ability to create persistence storage. It also allows us to name volumes, list volumes, and containers associates with the volumes.

- Directory Mounts - It is one of the best options for docker storage. It mounts a host's directory into a container.
- Storage Plugins - It provides an ability to connect to external storage platforms.

The life cycle of a Docker container includes the following stages:

1. Creating a Container: Containers are created from Docker images using the `docker run` command. When a container is created, it is in the "created" state.
2. Starting a Container: A container can be started using the `docker start` command. When a container is started, it transitions to the "running" state, and the application inside the container begins to execute.
3. Pausing a Container: The `docker pause` command can be used to pause the execution of processes inside a container. The container remains in the "running" state, but all processes are paused.
4. Resuming a Container: The `docker unpause` command resumes the execution of processes inside a paused container.
5. Stopping a Container: Containers can be stopped using the `docker stop` command. When stopped, the container transitions to the "exited" state, and the application inside the container stops executing.
6. Restarting a Container: The `docker restart` command stops and then starts a container.
7. Deleting a Container: Containers can be deleted using the `docker rm` command. Once deleted, all resources associated with the container, such as filesystem changes and network connections, are removed.

Installing Docker and Executing Docker Commands:

To install Docker, follow the installation instructions provided for your operating system on th e Docker website: https://docs.docker.com/get-docker/

After installing Docker, you can execute Docker commands to manage images and interact with containers. Here are some basic Docker commands:

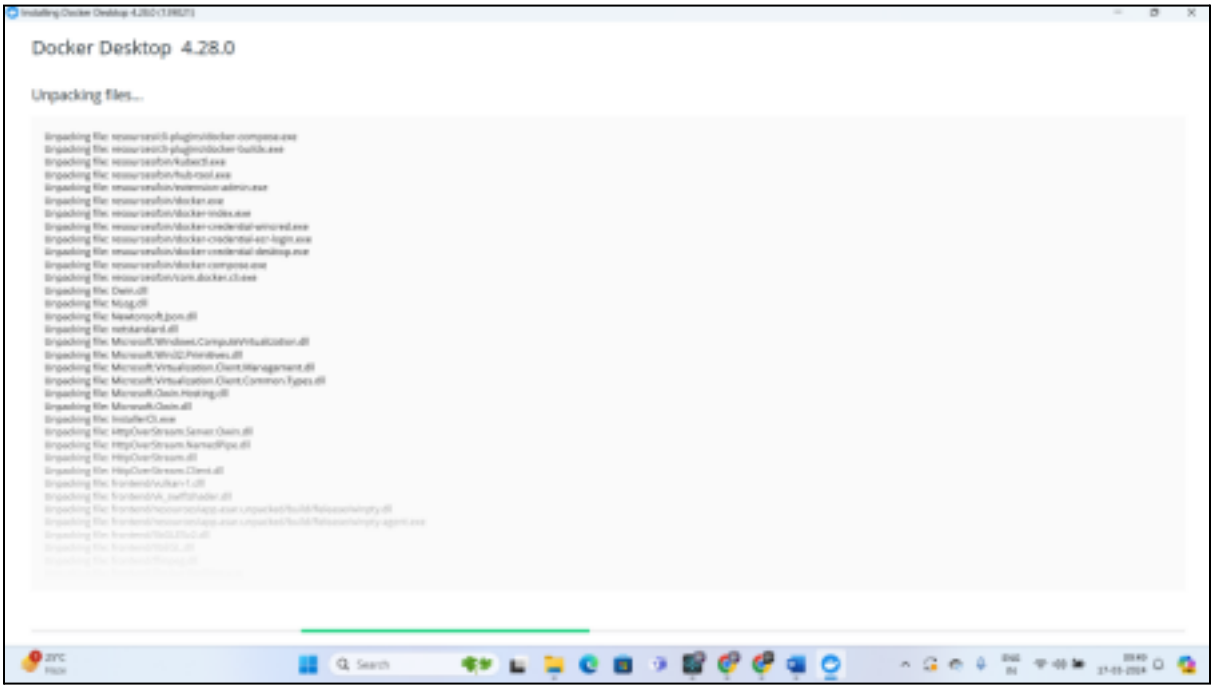
1. Pulling an Image: Use the `docker pull` command to download a Docker image from a registry. docker pull ubuntu
2. Listing Images: Use the `docker images` command to list all Docker images available on your system.
   docker images

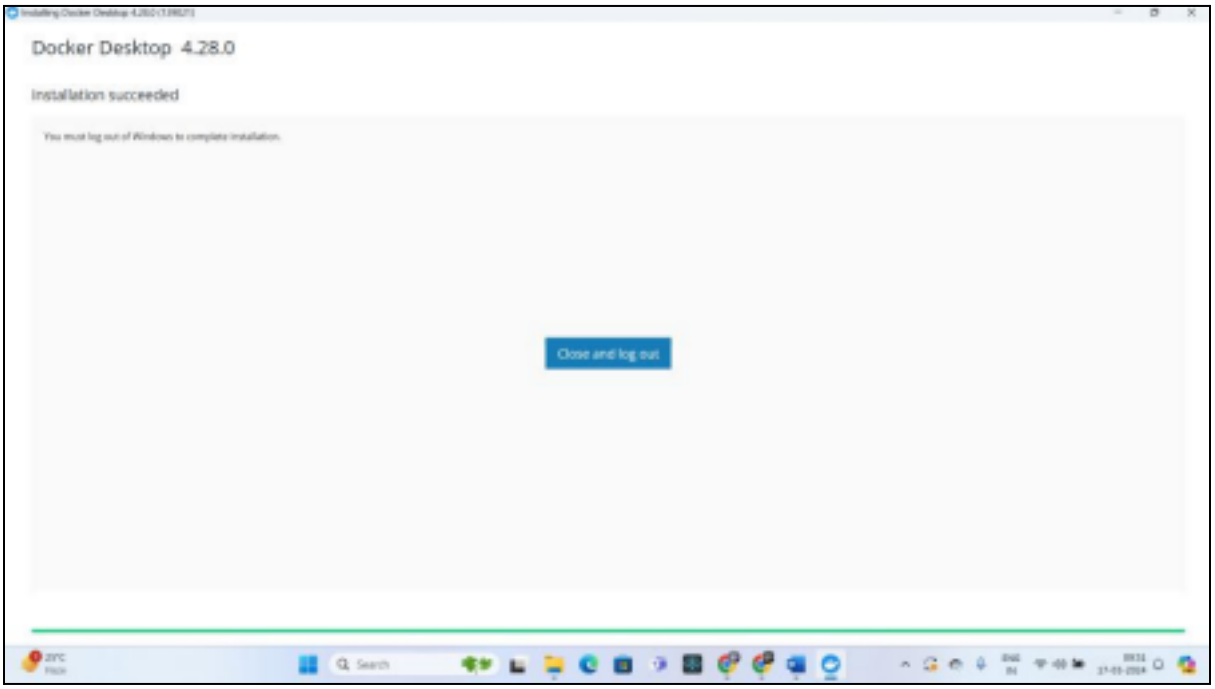
3. Running a Container: Use the `docker run` command to create and start a container from an image.
   docker run -d --name my-container ubuntu
4. Listing Containers: Use the `docker ps` command to list all running Docker containers.
   docker ps

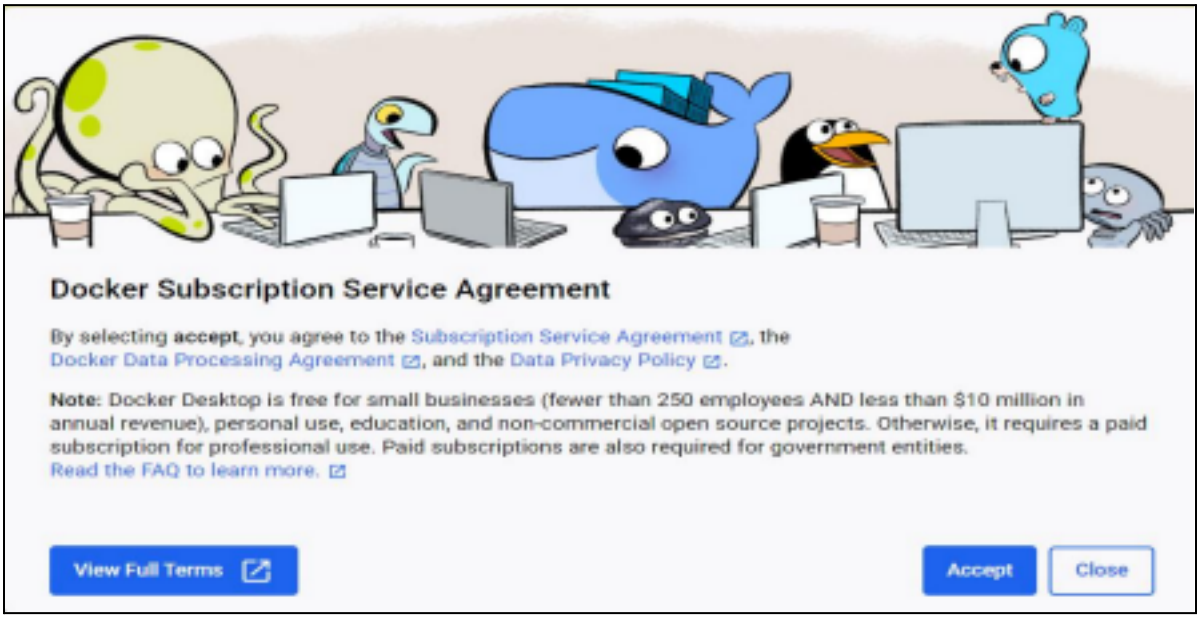
5. Stopping a Container: Use the `docker stop` command to stop a running container.
   docker stop my-container
6. Starting a Container: Use the `docker start` command to start a stopped container.
   docker start my-container
7. Removing a Container: Use the `docker rm` command to remove a container.
   docker rm my-container
8. Inspecting a Container: Use the `docker inspect` command to display detailed information about a container.
   docker inspect my-container

1. `docker version`: Check installed Docker version.
2. `docker search`: Search for Docker images on Docker Hub.
3. `docker pull`: Download a Docker image from a repository.
4. `docker run`: Create and start a new Docker container from an image.
5. `docker ps`: List all running Docker containers.
6. `docker stop`: Stop a running container.
7. `docker restart`: Restart a stopped container.
8. `docker kill`: Stop a container immediately.
9. `docker exec`: Execute a command inside a running container.
10. `docker login`: Log in to Docker Hub.
11. `docker commit`: Create a new image from a container's changes.
12. `docker push`: Push an image to a repository.
13. `docker network`: Manage Docker networks.
14. `docker history`: View the history of changes in an image.
15. `docker rmi`: Remove one or more images.
16. `docker ps -a`: List all containers (including stopped ones).
17. `docker copy`: Copy files between Docker container and local system.

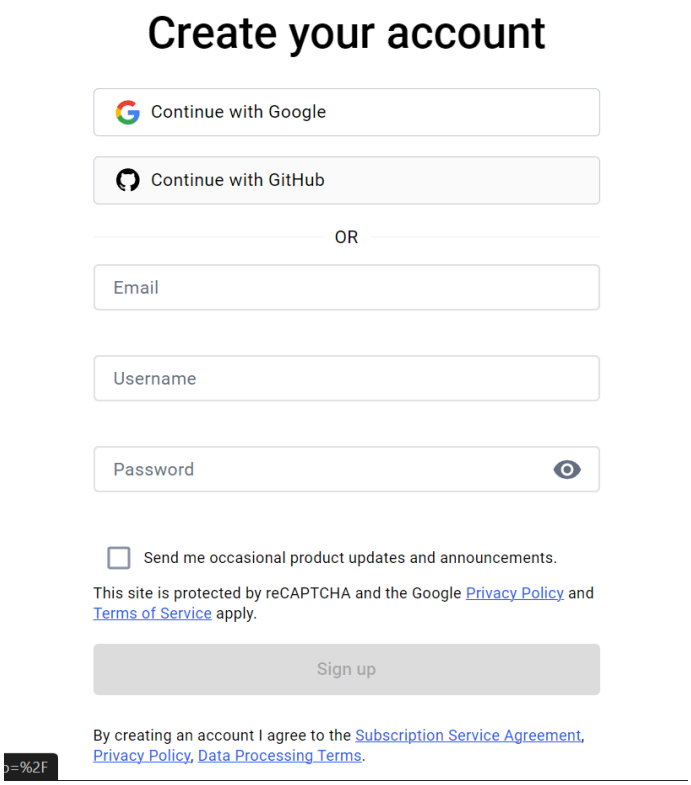
18. `docker logs`: View logs of a container.
19. `docker volume`: Manage Docker volumes for data persistence.
20. `docker logout`: Log out from Docker Hub.

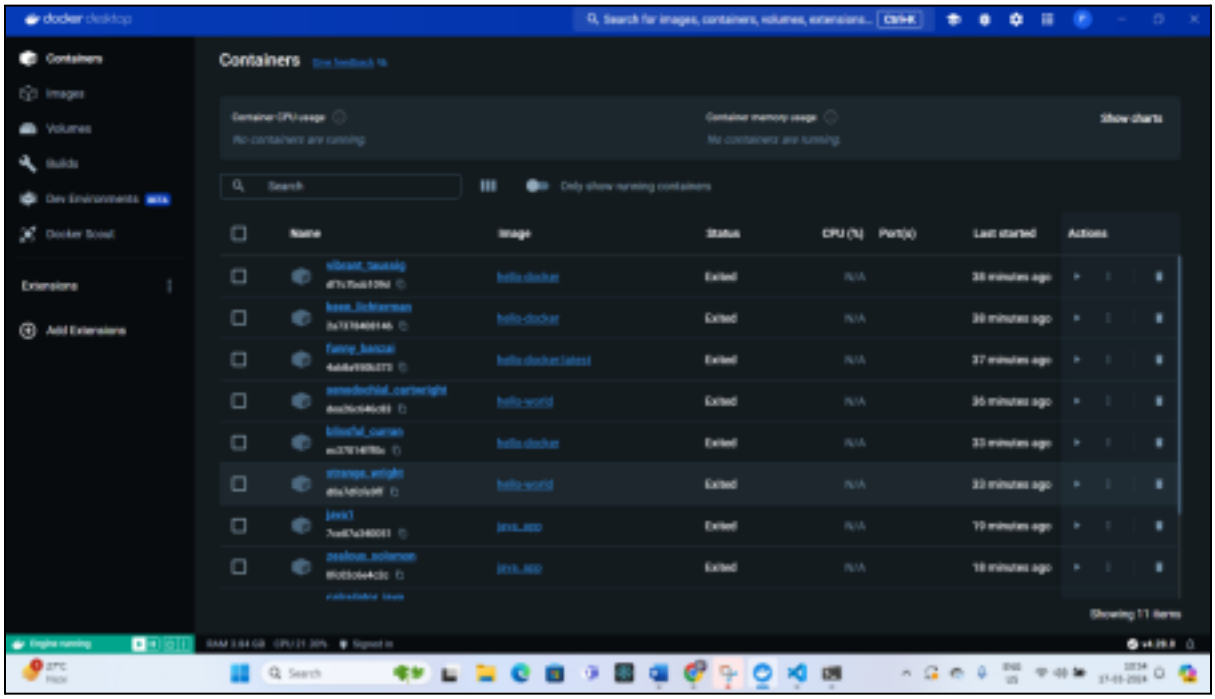


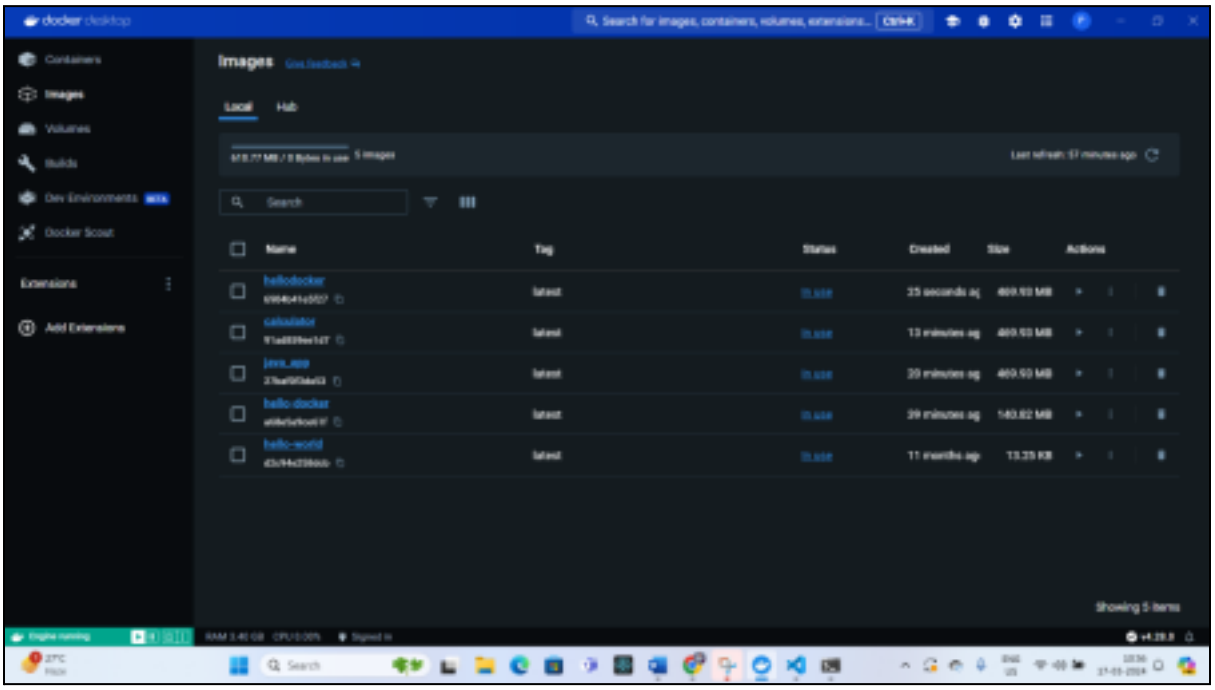


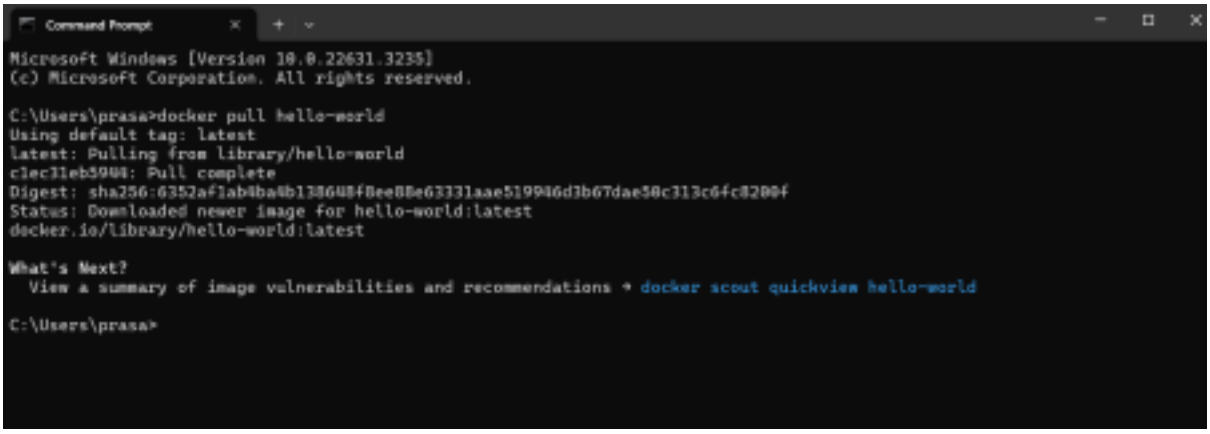

Docker Subscription Service Agreement
By selecting accept, you agree to the Subscription Service Agreement, the Docker Data Processing Agreement, and the Data Privacy Policy.
Note: Docker Desktop is free for small businesses (fewer than 250 employees AND less than $10 million in annual revenue), personal use, education, and non-commercial open source projects. Otherwise, it requires a paid subscription for professional use. Paid subscriptions are also required for government entities.
Read the FAQ to learn more.
View Full Terms
Accept
Close


Finish setting up Docker Desktop
version 4.28.0 (139021)
Complete the installation of Docker Desktop.
Use recommended settings (requires administrator password)
Docker Desktop automatically sets the necessary configurations that work for most developers.
Use advanced settings
You manually set your preferred configurations.
Finish


Create your account
Continue with Google
Continue with GitHub
OR
Email
Username
Password
Send me occasional product updates and announcements.
This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.
Sign up
By creating an account I agree to the Subscription Service Agreement, Privacy Policy, Data Processing Terms.


Containers
Images
Volumes
Builds
Extensions
Add Extensions
Name
Image
Status
CPU (%)
Port(s)
Last started
Actions


docker desktop
Containers
Images
Volumes
Builds
Dev Environments
Docker Scout
Extensions
Add Extensions
Images
Local
Hub
Search
Name
Tag
Status
Created
Size
Actions
hellodocker
latest
calculator
latest
hello-docker
latest
hello-world
latest
Showing 5 items


Command Prompt
Microsoft Windows [Version 10.0.22631.3235]
(c) Microsoft Corporation. All rights reserved.
C:\Users\prasa>docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:6352af1ab4ba4b1386408f8ee8be63331aae519946d3b67dae50c313c6fc8200f
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview hello-world
C:\Users\prasa>

```
C:\Users\prasa>docker --version
Docker version 25.0.3, build 4debf41

C:\Users\prasa>docker image ls
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
hellodocker    latest    6904b41e5f27   12 minutes ago   470MB
calculator     latest    91ad859ee1d7   25 minutes ago   470MB
java_app       latest    27baf0f3da53   31 minutes ago   470MB
hello-docker   latest    a68e5a9ce61f   50 minutes ago   141MB
hello-world    latest    d2c94e258dcb   10 months ago    13.3kB

C:\Users\prasa>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

```
C:\Users\prasa>docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS                      PORTS     NAM
ES
408a57158e26   hellodocker          "java sample"            12 minutes ago   Exited (0) 11 minutes ago             mon
derful_shannon
f8f622220804   calculator           "java sample"            16 minutes ago   Exited (1) 16 minutes ago             aff
ectionate_sinoussi
a61e93590c37   calculator           "java sample"            17 minutes ago   Exited (1) 16 minutes ago             cal
culator_java
0fc02c6e4c2c   java_app             "java sample"            29 minutes ago   Exited (0) 28 minutes ago             zea
lous_solomon
7ce87a340051   java_app             "java sample"            30 minutes ago   Exited (0) 29 minutes ago             jav
a1
d8a7dfcfe9ff   hello-world          "/hello"                 43 minutes ago   Exited (0) 43 minutes ago             str
ange_wright
ec37014fff0c   hello-docker         "docker-entrypoint.s…"   43 minutes ago   Exited (0) 43 minutes ago             bli
ssful_curran
dea26c646c83   hello-world          "/hello"                 46 minutes ago   Exited (0) 46 minutes ago             xen
odochial_cartwright
4ab8a950b373   hello-docker:latest  "docker-entrypoint.s…"   47 minutes ago   Exited (0) 47 minutes ago             fun
ny_banzai
2a7378408146   hello-docker         "docker-entrypoint.s…"   48 minutes ago   Exited (0) 48 minutes ago             kee
n_lichterman
df7c7bcb109d   hello-docker         "docker-entrypoint.s…"   49 minutes ago   Exited (0) 49 minutes ago             vib
rant_taussig
```

```
PS C:\Users\prasa\hello-docker\Hello-Docker> docker build -t hellodocker .
 [+] Building 3.8s (9/9) FINISHED
  => [internal] load build definition from Dockerfile
  => => transferring dockerfile: 123B
  => [internal] load metadata for docker.io/library/openjdk:latest
  => [internal] load .dockerignore
  => => transferring context: 2B
  => [1/4] FROM docker.io/library/openjdk:latest@sha256:9b640d0897d211c9e0ec635a485650ae6e28d4eca1efbc3404060a400b3f1f
  => [internal] load build context
  => => transferring context: 239B
  => CACHED [2/4] WORKDIR /app
  => [3/4] COPY . /app
  => [4/4] RUN javac sample.java
  => exporting to image
  => => exporting layers
  => => writing image sha256:6904b41e5f27e060fcb006cabcdb08f2bb2cbf62fcf9e3632b1db07cf6109439
  => => naming to docker.io/library/hellodocker

 View build details: docker-desktop://dashboard/build/default/default/y6814ym2qa1kots77dfcpgabb
```

```
C:\Users\prasa>docker logs calculator_java
Error: LinkageError occurred while loading main class sample
        java.lang.UnsupportedClassVersionError: sample has been compiled by a more recent version of the Java Runtime (
lass file version 65.0), this version of the Java Runtime only recognizes class file versions up to 62.0
Error: LinkageError occurred while loading main class sample
        java.lang.UnsupportedClassVersionError: sample has been compiled by a more recent version of the Java Runtime (
lass file version 65.0), this version of the Java Runtime only recognizes class file versions up to 62.0
```

```
C:\Users\prasa>docker run hellodocker
Hello Docker Using JAVA !

C:\Users\prasa>docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

```
C:\Users\prasa>docker search MySQL
NAME                            DESCRIPTION                                     STARS     OFFICIAL
mysql                           MySQL is a widely used, open-source relation…   14938     [OK]
phpmyadmin                      phpMyAdmin - A web interface for MySQL and M…   959       [OK]
mariadb                         MariaDB Server is a high performing open sou…   165       [OK]
percona                         Percona Server is a fork of the MySQL relati…   63        [OK]
bitnami/mysql                   Bitnami MySQL Docker Image                      110
bitnami/mysqld-exporter                                                         6
cimg/mysql                                                                      3
ubuntu/mysql                    MySQL open source fast, stable, multi-thread…   61
rapidfort/mysql                 RapidFort optimized, hardened image for MySQL   25
rapidfort/mysql8-ib             RapidFort optimized, hardened image for MySQ…   9
google/mysql                    MySQL server for Google Compute Engine          25
elestio/mysql                   Mysql, verified and packaged by Elestio         0
rapidfort/mysql-official        RapidFort optimized, hardened image for MySQ…   9
bitnamicharts/mysql                                                             0
hashicorp/mysql-portworx-demo                                                   0
databack/mysql-backup           Back up mysql databases to... anywhere!         111
linuxserver/mysql               A Mysql container, brought to you by LinuxSe…   41
mirantis/mysql                                                                  0
docksal/mysql                   MySQL service images for Docksal - https://d…   0
linuxserver/mysql-workbench                                                     55
vitess/mysqlctld                vitess/mysqlctld                                1
eclipse/mysql                   Mysql 5.7, curl, rsync                          1
drupalci/mysql-5.5              https://www.drupal.org/project/drupalci         3
drupalci/mysql-5.7              https://www.drupal.org/project/drupalci         0
datajoint/mysql                 MySQL image pre-configured to work smoothly …   2
```

```
C:\Users\prasa>docker network

Usage:  docker network COMMAND

Manage networks

Commands:
  connect     Connect a container to a network
  create      Create a network
  disconnect  Disconnect a container from a network
  inspect     Display detailed information on one or more networks
  ls          List networks
  prune       Remove all unused networks
  rm          Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.

C:\Users\prasa>docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
6bfd8a525900   bridge    bridge    local
e12711773ad3   host      host      local
927a0521e61c   none      null      local

C:\Users\prasa>
```

Conclusion: In conclusion, Docker's architecture, container life cycle, and commands empower users to efficiently manage containerized applications. By understanding Docker's components, stages like creation, start, stop, and delete of containers, users can leverage Docker effectively. With commands like `docker pull`, `docker run`, `docker stop`, and others, users can interact with containers and images seamlessly, optimizing application deployment and management.

Mastering Docker fundamentals enhances development and operational workflows, ensuring smoother containerized application deployment.