EXP 09

Kalpesh Jangir
T13/45

Aim:

To understand Docker Architecture and Container Life Cycle, install Docker andexecute docker commands to manage images and interact with containers

Theory:

1. Docker Architecture:

Docker employs a client-server model. The pivotal components encompass:

- Docker Daemon: Serving as a background process, it orchestrates Docker objects such as images, containers, networks, and volumes.
- Docker Client: Serving as the primary interface for users, it dispatches commands to the Docker Daemon for execution.
- Docker Images: Immutable blueprints housing filesystems and application code.
- Docker Containers: Lightweight, portable, and self-contained entities that execute applications.

2. Container Lifecycle:

- Create: A container materializes from an image using the `docker run` command.
- Start The container commences operation via the `docker start` command.
- Pause: Utilizing the `docker pause` command, a container can be halted.
- Unpause: A paused container can be reinstated to activity with the `docker unpause` command.
- Stop: Halting a container is accomplished via the `docker stop` command. This methodically sends a SIGTERM signal, prompting processes within the container to gracefully conclude.
- Kill: If exigencies arise, forcefully halting a container is achieved using the `docker kill` command. This instantaneously terminates processes within the container by dispatching a SIGKILL signal.
- Remove: Post-halt, a container can be eliminated from existence via the `docker rm` command.

3. Managing Docker Images and Containers:

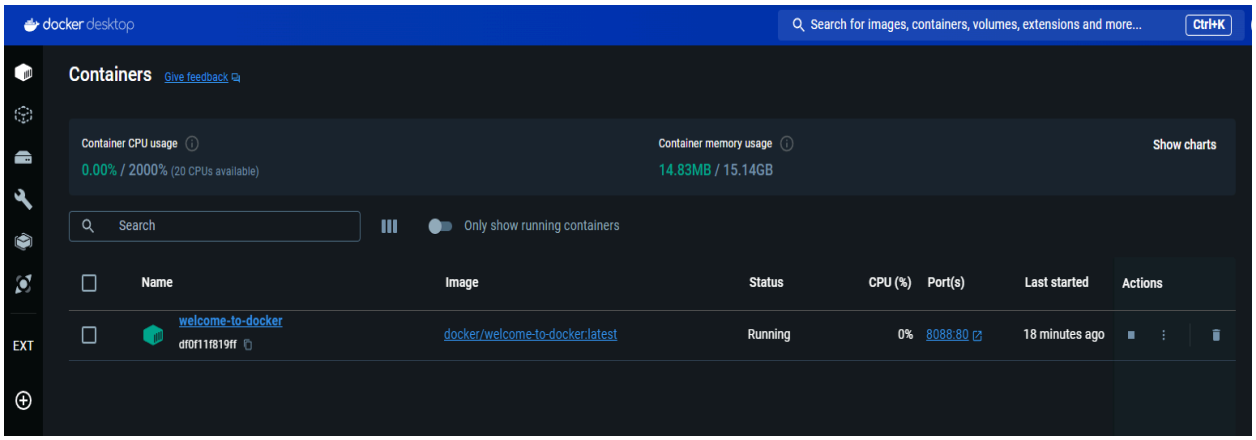
- Image Management: Docker offers a spectrum of commands:
  - docker pull`: Fetches images from registries.
  - docker build`: Constructs images from Dockerfiles.

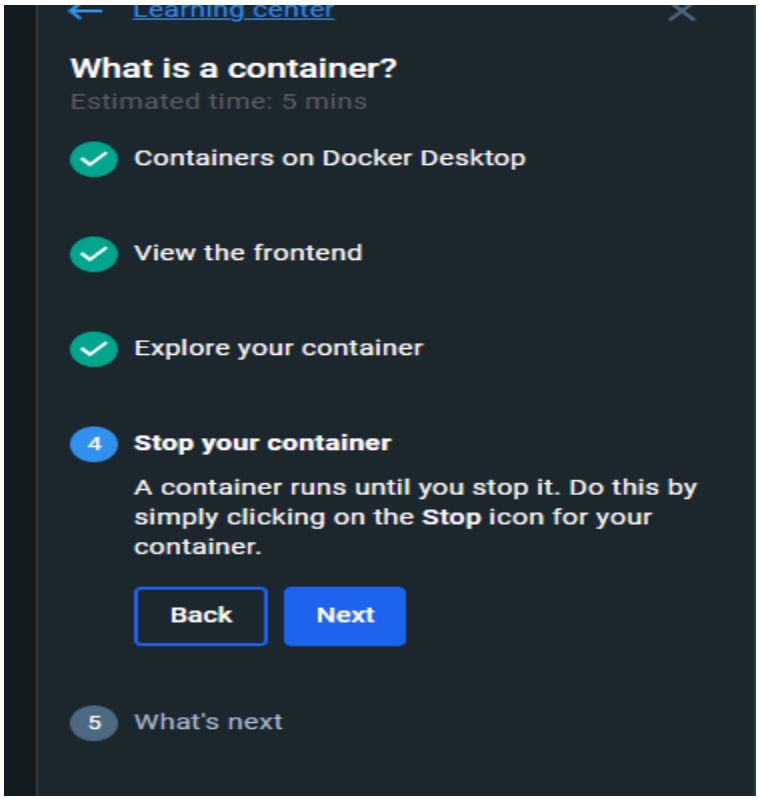
Kalpesh Jangir
T13/45

- docker push`: Uploads images to registries.

- Container Management: Docker commands wielded in container administration encompass:

- docker run: Instantiates and launches containers.
- docker ps: Enumerates extant containers.
- docker stop: Halts a running container.
- docker rm: Deletes a container.

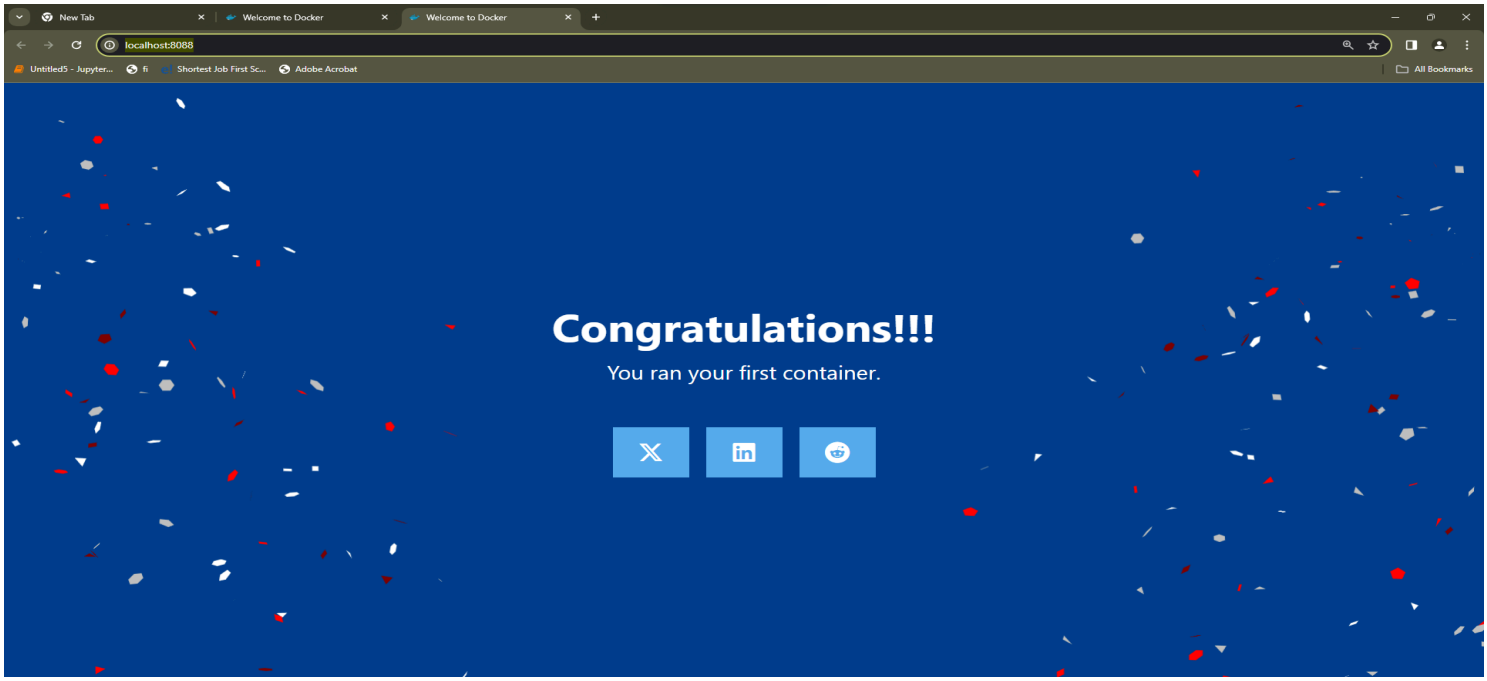


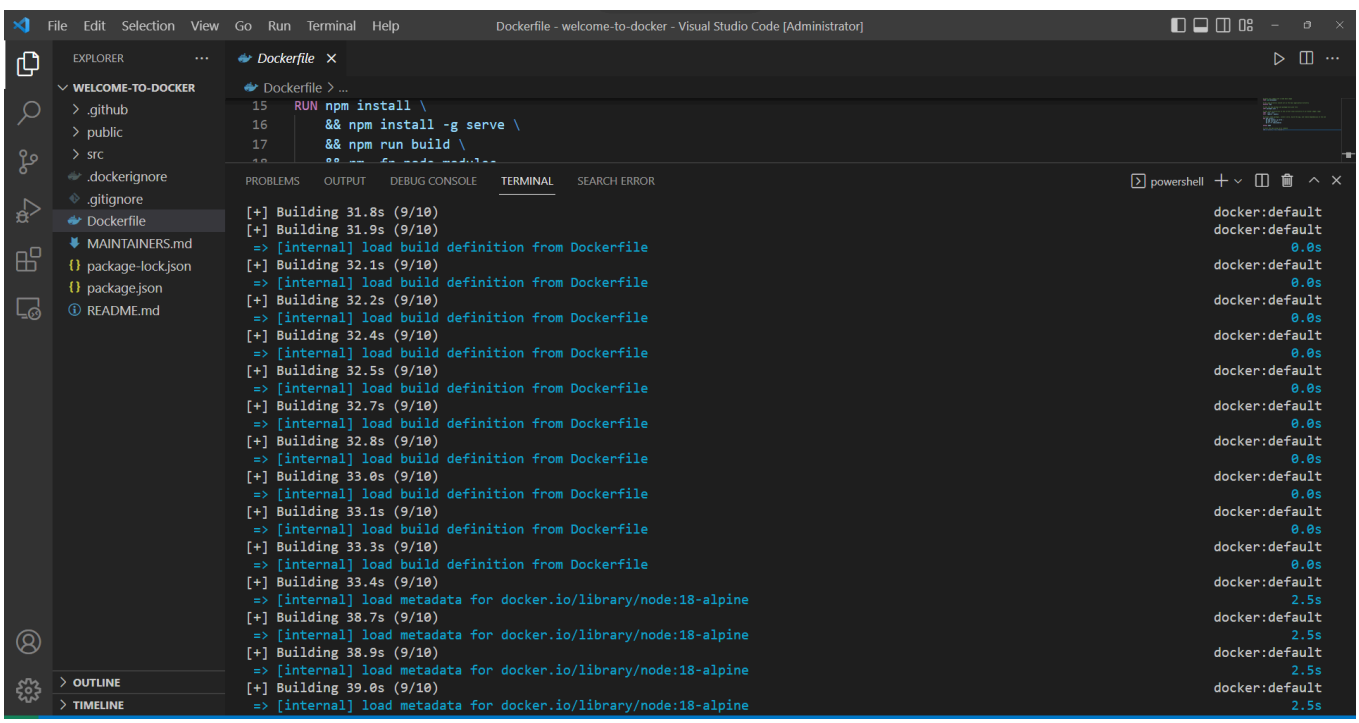


EXP 09

Kalpesh Jangir

T13/45



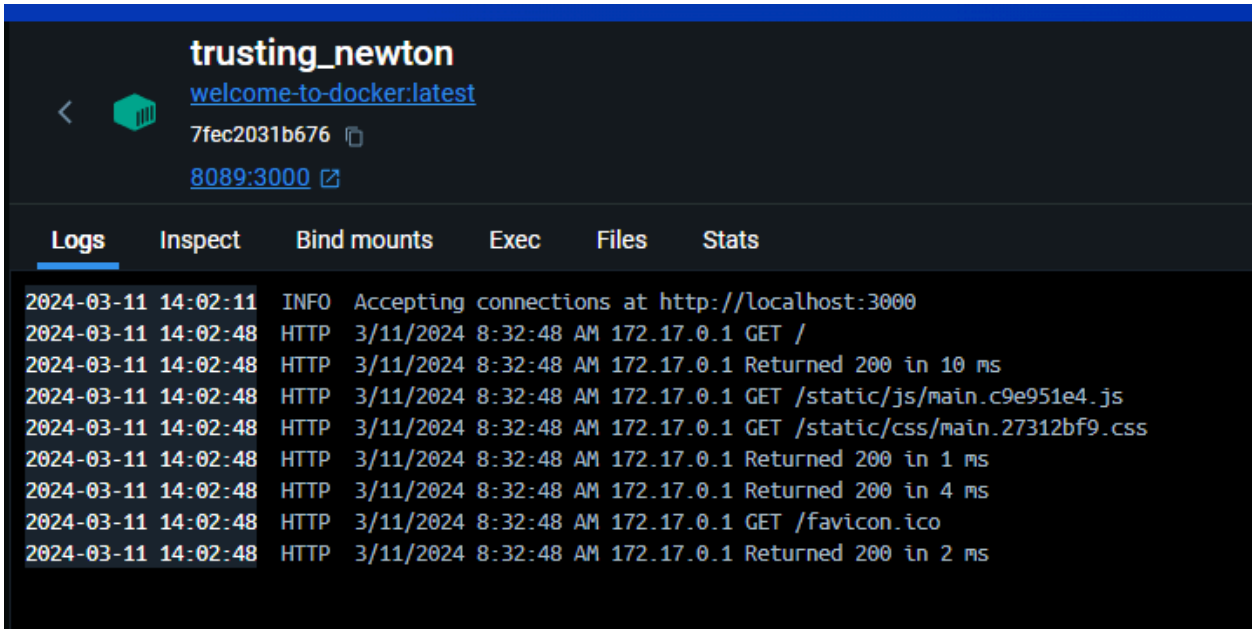


```
 => => naming to docker.io/library/welcome-to-docker

What's Next?
  1. Sign in to your Docker account → docker login
  2. View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\15L\Desktop\welcome-to-docker> docker scout quickview
    i New version 1.5.1 available (installed version is 1.4.1) at https://github.com/docker/scout-cli
    v SBOM of image already cached, 307 packages indexed

  Target                  local://welcome-to-docker:latest   0C   0H   1M   0L
    digest                0e7372b5f723
  Base image              node:18-alpine                     0C   0H   1M   0L
  Updated base image      node:18.19.0-alpine                0C   0H   1M   0L

What's Next?
  View vulnerabilities → docker scout cves local://welcome-to-docker:latest
  View base image update recommendations → docker scout recommendations local://welcome-to-docker:latest
  Include policy results in your quickview by supplying an organization → docker scout quickview local://welcome-to-docker:la
anization>
```



Conclusion:

Comprehending Docker architecture and the lifecycles of containers is pivotal for leveraging Docker proficiently in software development and deployment. Armed with a profound understanding of these concepts and adeptness with Docker commands, one can seamlessly manage Docker images and containers, streamline developmental workflows, and deploy applications with utmost efficacy.