LAB ACTION PLAN FOR WEEK 5

Objectives:

Students will able to:

- Learn how to pull, run, stop, start, remove, and inspect containers and images.
- Gain the ability to create, monitor, and troubleshoot running containers.
- Configure and manage networks for container communication.
- Create and manage persistent storage for containers.
- Learn how to list, remove, and manage images efficiently.

Docker:

Docker is an open-source platform that automates the process of building, packaging, and running applications inside lightweight, portable containers, ensuring they run consistently across different environments.

Docker Images:

A read-only template that contains the application code, runtime, libraries, and dependencies. It is like a blueprint for creating containers.

Docker Containers:

A running instance of a Docker image.

Working with Docker CLI Commands:

Docker CLI Commands with redis

Step 1: Pull the redis Image

```
docker pull redis
```

Step 2: Run a Redis Container

```
docker run --name my-redis -d redis
```

What It Does:

Creates and starts a container named my-redis from the redis image.

The -d flag runs the container in the background.

Step 3: Check Running Containers

```
docker ps
```

What It Does:

Lists all running containers.

Step 4: Access Redis

```
docker exec -it my-redis redis-cli
```

Opens the Redis command-line tool (redis-cli) inside the container.

Example Redis Commands:

```
127.0.0.1:6379> SET name "Alice"
OK
127.0.0.1:6379> GET name
"Alice"
```

Step 5: Stop the Redis Container

```
docker stop my-redis
```

What It Does:

- Stops the Redis container but doesn’t delete it.

Step 6: Restart the Redis Container

```
docker start my-redis
```

What It Does:

- Restarts the stopped container.

Step 7: Remove the Redis Container

```
docker rm my-redis
```

What It Does:

- Deletes the container permanently.

Step 8: Remove the Redis Image

docker rmi redis

What It Does:

・Deletes the Redis image from your local system.

2.Working with Docker file

A Docker file is a text file with instructions to create a custom Docker image.

Step 1: Set Up Your Folder

1. Windows:

o Create a folder like C:\DockerProjects\Redis.

o Open Git Bash and navigate to the folder:

cd /c/DockerProjects/Redis

2. Mac/Linux:

o Create a folder:

mkdir ~/DockerProjects/Redis

cd ~/DockerProjects/Redis

Step 2: Write the Dockerfile

1. Inside the folder, create a file named Dockerfile (no extension).

2. Add the following content:

FROM redis:latest

CMD ["redis-server"]

Docker Commands (Step-by-step):

1. docker build -t redisnew .

What it does:

・This creates (builds) a Docker image using the recipe (Dockerfile) in the current

folder (.).

・-t redisnew: Gives the image a name/tag ("redisnew"), so you can find it easily.

2. docker run --name myredisnew -d redisnew

What it does:

・Starts a new container (mini computer) from the redisnew image.

・--name myredisnew: Names the container "myredisnew" so it’s easy to identify.

・-d: Runs the container in the background.

3. docker ps

What it does:

・Shows a list of containers that are running right now.

4. docker stop myredisnew

What it does:

・Stops the container named "myredisnew" (like turning off a computer).

5. docker login

What it does:

・Logs you into your Docker Hub account, so you can upload images.

6. docker ps -a

What it does:

・Shows a list of all containers, including stopped ones.

7. docker commit 0e993d2009a1 budarajumadhurika/redis1

What it does:

・Takes a snapshot (saves changes) of the container with ID 0e993d2009a1 and creates a new image called budarajumadhurika/redis1.

8. docker images

What it does:

・Lists all images saved on your system.

9. docker push budarajumadhurika/redis1

What it does:

・Uploads the image budarajumadhurika/redis1 to Docker Hub, so others can download it.

10. docker rm 0e993d2009a1

What it does:

・Deletes the container with ID 0e993d2009a1.

11. docker rmi budarajumadhurika/redis1

What it does:

・Deletes the image budarajumadhurika/redis1 from your system.

12. docker ps -a

What it does:

Shows all containers again to confirm changes.

13. docker logout

What it does:

・Logs you out of Docker Hub.

14. docker pull budarajumadhurika/redis1

What it does:

・Downloads the image budarajumadhurika/redis1 from Docker Hub.

15. docker run --name myredis -d budarajumadhurika/redis1

What it does:

・Starts a new container using the image budarajumadhurika/redis1.

16. docker exec -it myredis redis-cli

What it does:

・Opens the Redis command-line interface (like a terminal) inside the running container

myredis.

17. SET name "Abcdef"

What it does:

・Saves a key-value pair in Redis (key = name, value = Abcdef).

18. GET name

What it does:

・Retrieves the value of the key name from Redis (it will return "Abcdef").

19. exit

What it does:

・Exits the Redis CLI.

20. docker ps -a

What it does:

・Shows all containers again to check their status.

21. docker stop myredis

What it does:

・Stops the container myredis.

22. docker rm 50a6e4a9c326

What it does:

・Deletes the container with ID 50a6e4a9c326.

23. docker images

What it does:

・Lists all images again to confirm which ones remain.

24. docker rmi budarajumadhurika/redis1

What it does:

・Deletes the image budarajumadhurika/redis1 again.

Step 3: Remove Login Credentials (Optional)

If you no longer need to be logged in, you can log out:

docker logout

What It Does:

・Logs you out from Docker Hub and removes your stored credentials.

Scenario based Questions:

1.Your application is running inside Docker, but you're not sure if the container is active.
Which command helps you check running containers?

2. You notice one of your containers is consuming high CPU and want to stop it.
How do you stop a running container named web_app?

3. You've written a Dockerfile in your project directory and want to build an image named myapi.

What Docker CLI command do you use?

4. You want to run your web_app container and expose its internal port 5000 on host port 8080.
Which command should you use?

5. Your container is running, but you want to go inside and debug it using a shell. What CLI command helps you enter the container's shell?

6. You've built a few test images and want to delete one called old_api. How would you do this?

7. Your app crashed in the container. You need to check the logs. How do you check this?

8. You want to start a container and let it run in the background. Which command should you use?

9. You're not sure which container is using port 3000. How do you check?

10. You built an image but forgot to tag it. How do you tag it now?

11. You want to export an image to a .tar file for offline transfer. How do you export it?

12. You want to restart a container automatically if it crashes. What Docker CLI command do you use?

13. You want to see how much CPU/RAM each container uses. What Docker CLI command do you use? How to Control or Limit RAM in Docker? How to Measure Actual Memory Usage?

14. You want your container to run a shell command before starting the app. What do you include in docker file?

15. You want to create a custom Nginx image with your static files. What do you do?

Conclusion:

Through this hands-on exploration of Docker commands and Dockerfile usage, students will get a clear picture about the end-to-end container workflow—from pulling images, creating and running containers, to customizing them with Dockerfiles. By practicing key commands such as docker build, docker run, docker ps, and docker exec, students will gain practical skills in managing containerized environments. The Dockerfile exercise reinforced the importance of layered builds, efficient image creation, and environment reproducibility. Overall, these activities strengthened the understanding of containerization principles, enabling building, testing and deploying applications in isolated, consistent environments.