

Learnings for lifetime:

1)Port publishing activity

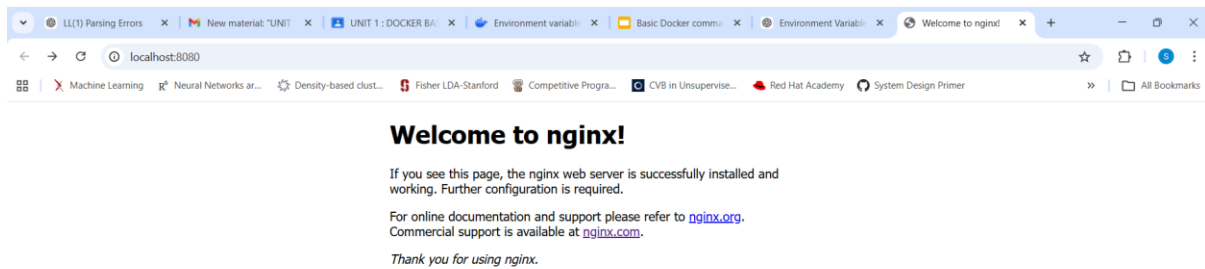
So basically, covered publishing port mechanism, I must say it's just intricately beautiful though it might be very basic it really showed me how powerful docker is even at basic levels, so according to what I understood of it, since containers run in isolated namespaces the ports they use are inaccessible to users outside of it so we publish the port sort of exposing it to outside world by **publishing a port** which is Docker's way of intentionally breaking that isolation in a controlled manner: Docker makes the **host machine listen on a chosen port** and then **forwards all traffic arriving on that host port to the container's internal port**.

In other words, the container still thinks it is listening privately, but Docker transparently acts as a bridge between the external world and the container, exposing only what we explicitly allow.

The screenshot displays the Docker Desktop application window. The top bar shows the 'docker.desktop' logo and a 'PERSONAL' label. The main interface is divided into a sidebar on the left with icons for containers, images, volumes, and settings. The central area is titled 'Containers' and includes a search bar, a filter for 'Only running' containers, and a table listing three containers: 'condescending_4e05c3605090' (httpd), 'magical_pascal_96586c4b2431' (ubuntu), and 'optimistic_wesc_7f6f9833a369' (nginx). The 'optimistic_wesc' container is highlighted with a green dot and shows port mapping '8080:80'. Below the table is a 'Terminal' window showing the command 'docker run -p 8080:80 nginx' and its output, which includes configuration details for the nginx container. The bottom status bar indicates system resources: RAM 0.92 GB, CPU 0.08%, and Disk 4.20 GB used (limit 1006.85 GB).

Name	Container ID	Image	Port(s)	Actions
condescending_	4e05c3605090	httpd		[Play] [More] [Delete]
magical_pascal	96586c4b2431	ubuntu		[Play] [More] [Delete]
optimistic_wesc	7f6f9833a369	nginx	8080:80	[Play] [More] [Delete]

```
root@96586c4b2431:/# exit
exit
PS C:\Users\sreev> docker run -p 8080:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
```

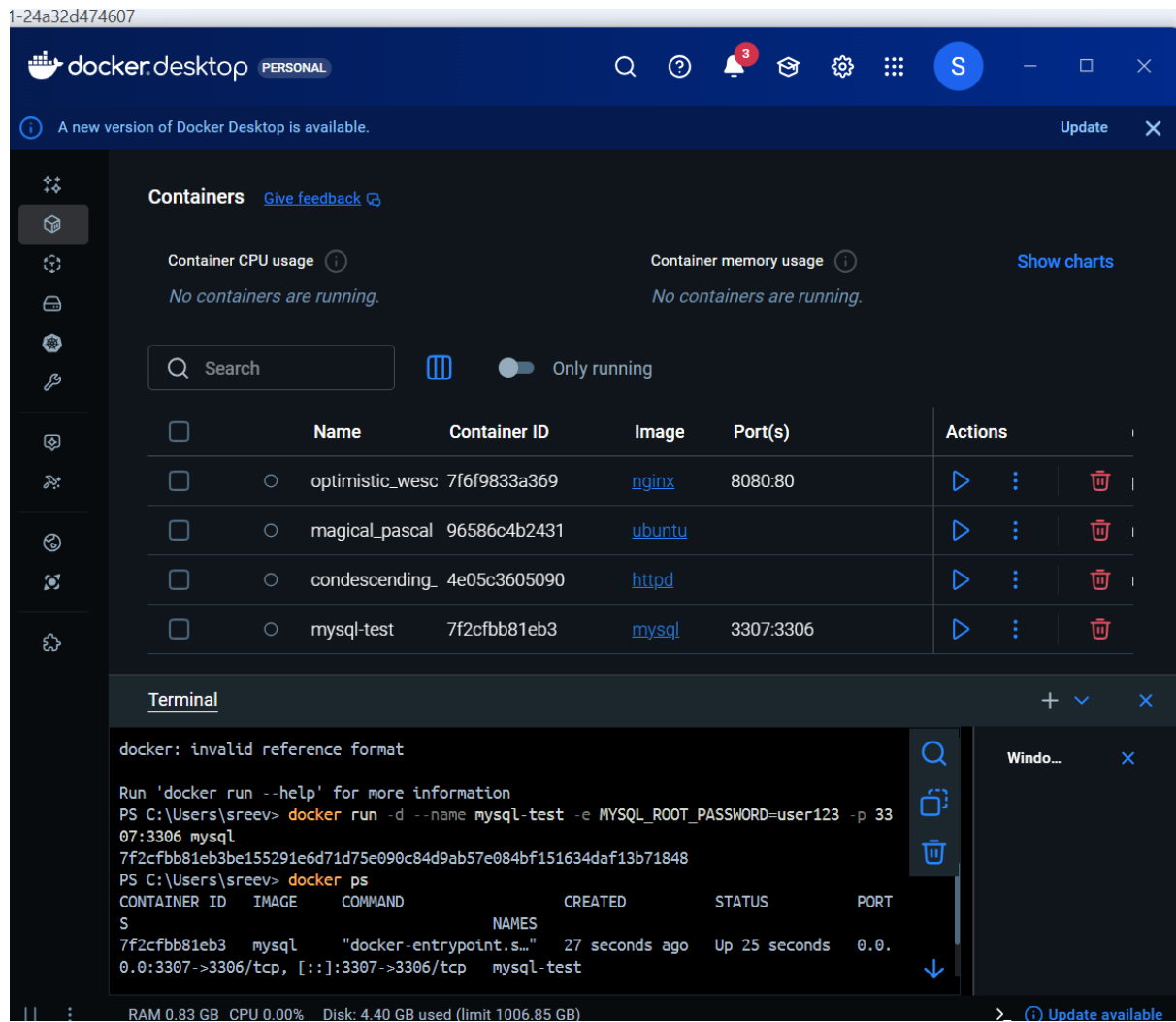


What happens internally?

Browser → localhost:8080 → Docker Host → Container:80 → Nginx

2)Some more interesting stuff ahead: started to work more on port publishing and MySQL container initialization.

Started exploring Docker port publishing and MySQL container initialization in more depth. By running a MySQL container in detached mode and setting the `MYSQL_ROOT_PASSWORD` environment variable, I understood how Docker images use environment variables during initialization to configure services automatically. I also learned that publishing a port (`-p 3307:3306`) does not affect how services work *inside* the container, but instead allows the host machine to access the MySQL server running in the container. Using `docker exec`, I was able to run the MySQL client inside the container without relying on port publishing, which clarified the difference between internal container access and external host access. Overall, this showed how Docker maintains isolation by default while still allowing controlled exposure of services when needed.



```
PS C:\Users\sreev> docker exec -it mysql-test mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.6.0 MySQL Community Server - GPL

Copyright (c) 2000, 2026, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
PS C:\Users\sreev> |
```

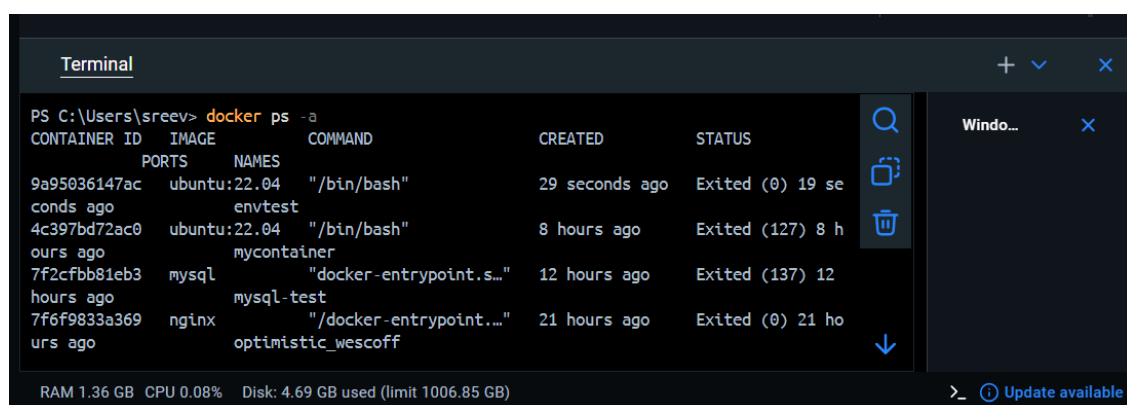
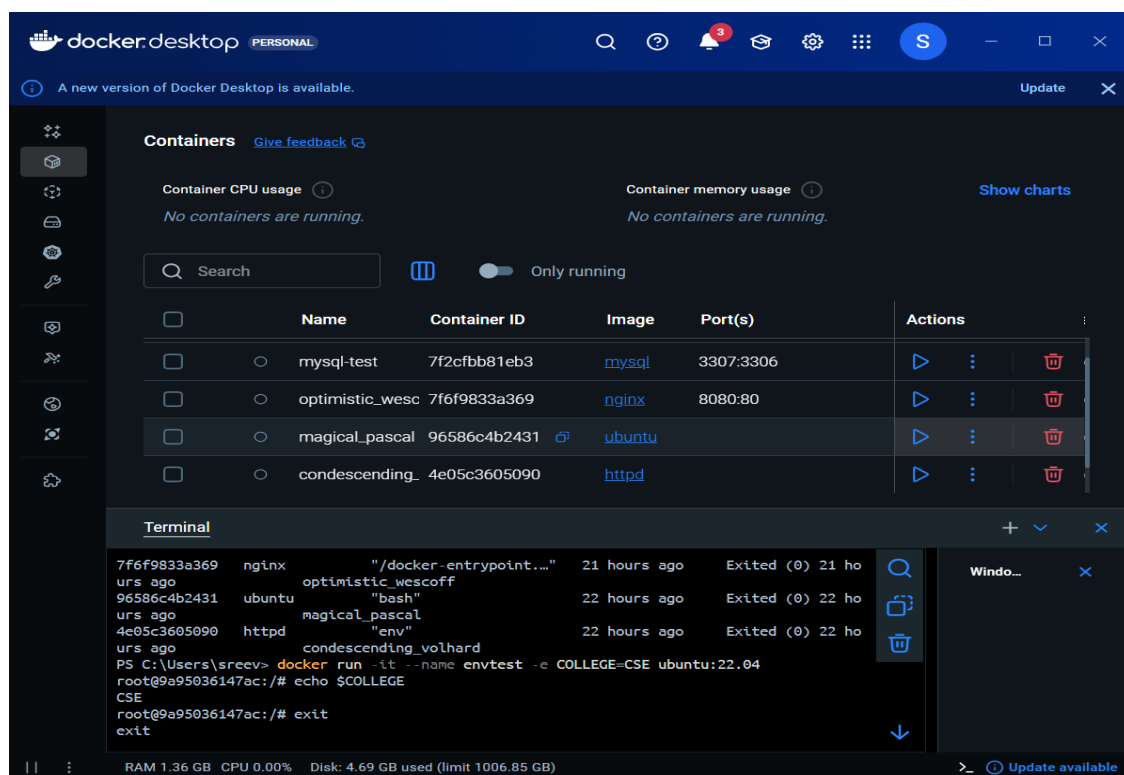
When I publish a port in Docker, I’m not “opening a port inside the container.” I’m asking Docker to stand at the host’s door, listen on my behalf, and quietly forward anything it hears into the container’s private world. Behind the scenes, Docker sets up all the low-level networking machinery—NAT rules, packet forwarding, and bridge connections—so the container

remains isolated, yet reachable, without me ever touching iptables or firewall configurations.

Let's go for our task-1:

Task: Environment Variable Test in Docker

- Run Ubuntu container
- Pass `-e COLLEGE=CSE`
- `SHOW echo $COLLEGE`
- Stop container, then check what happened



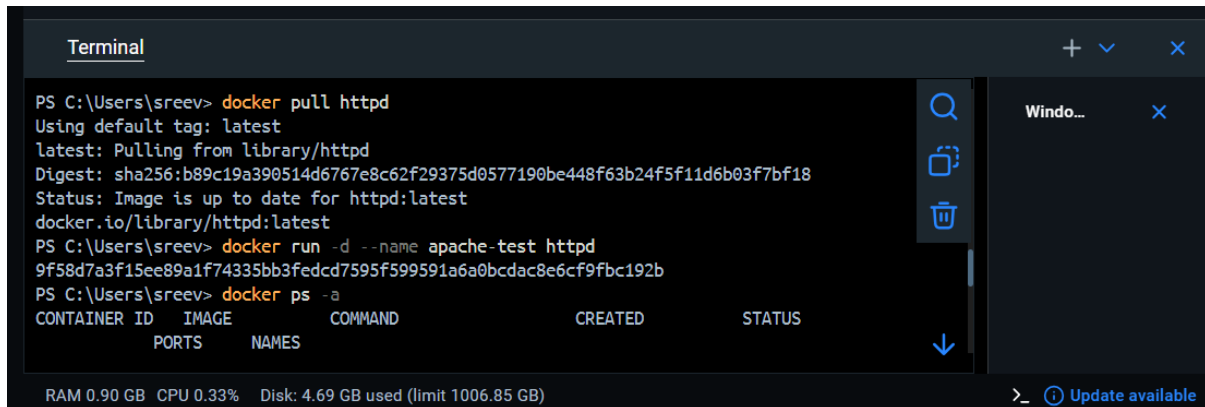
3) Here we go, exec on the way:

`docker exec` is a Docker command used to run a new command inside an already running container. It allows users to interact with a container without stopping or restarting it.

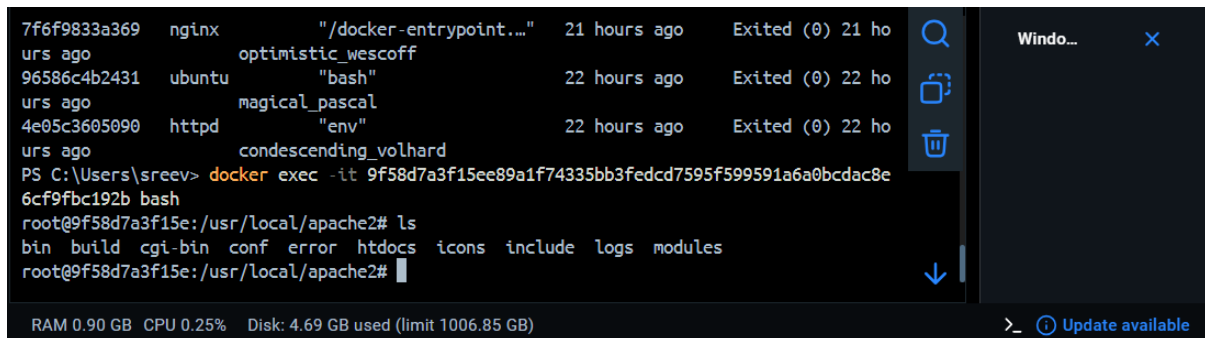
This command is commonly used for:

- Debugging running containers
- Inspecting files and logs
- Checking environment variables
- Running administrative commands inside containers

Unlike `docker run`, which creates a new container, `docker exec` works only with existing and running containers.

A terminal window titled "Terminal" with standard window controls. It shows the execution of `docker pull httpd` and `docker run -d --name apache-test httpd`. The output of the run command is a long container ID. Below this, the `docker ps -a` command is run, displaying a table of containers.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
9f58d7a3f15ee89a1f74335bb3fedcd7595f599591a6a0bcdac8e6cf9fbc192b	httpd			

The bottom status bar shows "RAM 0.90 GB CPU 0.33% Disk: 4.69 GB used (limit 1006.85 GB)" and an "Update available" notification.A terminal window showing the output of `docker ps -a` and the execution of `docker exec` to enter a container. The `docker ps -a` output lists several containers, including the one created in the previous screenshot.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
7f6f9833a369	nginx	"/docker-entrypoint..."	21 hours ago	Exited (0) 21 ho
urs ago	optimistic_wescoff			
96586c4b2431	ubuntu	"bash"	22 hours ago	Exited (0) 22 ho
urs ago	magical_pascal			
4e05c3605090	httpd	"env"	22 hours ago	Exited (0) 22 ho
urs ago	condescending_volhard			

The `docker exec` command is used to enter the container with ID `9f58d7a3f15ee89a1f74335bb3fedcd7595f599591a6a0bcdac8e6cf9fbc192b`. The prompt changes to `root@9f58d7a3f15e:/usr/local/apache2#`, and the `ls` command is executed, showing the contents of the `/usr/local/apache2` directory. The bottom status bar shows "RAM 0.90 GB CPU 0.25% Disk: 4.69 GB used (limit 1006.85 GB)" and an "Update available" notification.

```
Terminal
root@9f58d7a3f15e:/usr/local/apache2# exit
exit
PS C:\Users\sreev> docker exec apache-test env
PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=9f58d7a3f15e
HTTPD_PREFIX=/usr/local/apache2
HTTPD_VERSION=2.4.66
HTTPD_SHA256=94d7ff2b42acbb828e870ba29e4cbad48e558a79c623ad3596e4116efcfea25a
HTTPD_PATCHES=
HOME=/root
PS C:\Users\sreev>

RAM 0.90 GB CPU 0.17% Disk: 4.69 GB used (limit 1006.85 GB) >_ ⓘ Update available
```

Practice Question 1:

Run a Docker container named DB-app based on the mongodb image, and expose port 80 on the host to port 8082 on the container.

```
Terminal
HOSTNAME=9f58d7a3f15e
HTTPD_PREFIX=/usr/local/apache2
HTTPD_VERSION=2.4.66
HTTPD_SHA256=94d7ff2b42acbb828e870ba29e4cbad48e558a79c623ad3596e4116efcfea25a
HTTPD_PATCHES=
HOME=/root
PS C:\Users\sreev> docker stop apache-test
apache-test
PS C:\Users\sreev> docker run -d --name DB-app -p 80:8082 mongo
72d0bb17c193ec0f0816118bf7d9e021f9717f4efcade0cf899d1c1ca96623e9
PS C:\Users\sreev>

RAM 1.00 GB CPU 0.25% Disk: 4.89 GB used (limit 1006.85 GB) >_ ⓘ Update available
```

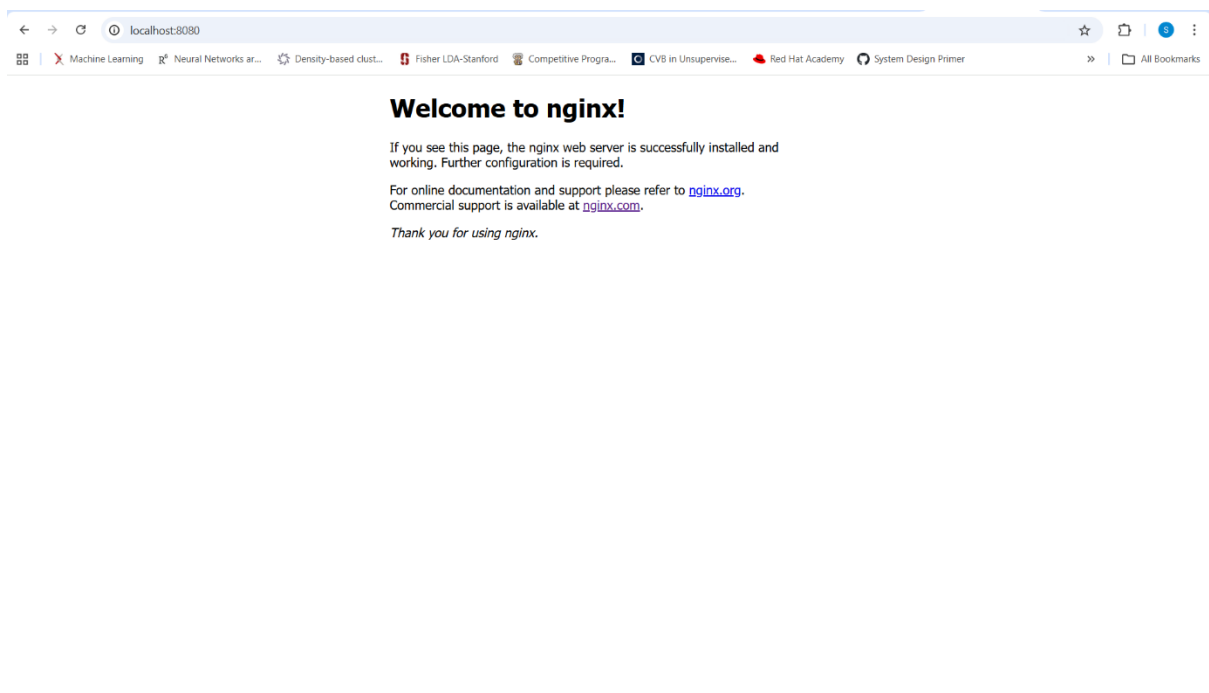
Practice Question 2:

Run a Docker container based on the nginx image, exposing port 8080 on the host to port 80 on the container. Set an environment variable NGINX_PORT=8080 inside the container and start the container interactively.

```
DB-app
PS C:\Users\sreev> docker run -it --name nginx-portpublishing -p 8080:80 -e NGINX_PORT=8080 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf

RAM 0.98 GB CPU 0.00% Disk: 4.89 GB used (limit 1006.85 GB)
```

```
2026/02/10 15:24:05 [notice] 1#1: start worker process 31
2026/02/10 15:24:05 [notice] 1#1: start worker process 32
2026/02/10 15:24:05 [notice] 1#1: start worker process 33
2026/02/10 15:24:05 [notice] 1#1: start worker process 34
2026/02/10 15:24:05 [notice] 1#1: start worker process 35
2026/02/10 15:24:05 [notice] 1#1: start worker process 36
2026/02/10 15:24:05 [notice] 1#1: start worker process 37
2026/02/10 15:24:05 [notice] 1#1: start worker process 38
2026/02/10 15:24:05 [notice] 1#1: start worker process 39
2026/02/10 15:24:05 [notice] 1#1: start worker process 40
```



NOTE: EXPOSE documents container ports for internal use, while -p publishes container ports to the host for external access.

Task 1:

How would you use the docker run command with -it, -e, -v, and --name to:

- Set an environment variable APP_ENV=production
- Name the container my_app
- Start an interactive terminal
- Use an image called my_image

```
PS C:\Users\sreev> docker run -it --name my_app -e APP_ENV=production -v my_volume:/app ubuntu
root@bd37f134f9c4:/# echo $APP_ENV
production
root@bd37f134f9c4:/#
```

RAM 0.87 GB CPU 0.00% Disk: 4.89 GB used (limit 1006.85 GB)

The command- **docker run -it --name my_app -e APP_ENV=production -v my_volume:/app ubuntu**

creates and starts an interactive Docker container named my_app from the ubuntu image. It sets the environment variable APP_ENV to production at runtime.

The **-v my_volume:/app** option mounts a Docker-managed volume named **my_volume** into the container at the **path /app**. This volume is stored on the host system and exists independently of the container. Any files created or modified inside **/app** are written to the volume and persist even after the container is stopped or deleted, while all other container files remain temporary.

Task-2: Copy files between container and host

Docker cp is used to **copy files or directories between a Docker container and the host system.**

- **Syntax**

Container → Host

docker cp <container_id|name>:<container_path> <host_path>

Host → Container

docker cp <host_path> <container_id|name>:<container_path>

- **Key Characteristics**

- Works with **running and stopped containers**
- Copies data **one time only** (no live sync)
- Does **not require volumes**
- Container filesystem must exist (container must not be removed)
- Files are copied, not mounted

- **Permissions Note**

- Destination path must be **writable**

- /root/ is commonly used because it avoids permission issues
- Any valid container path can be used
- **Persistence Clarification**
 - Files copied **into a container** persist only until the container is deleted
 - Files copied **from a container** are permanently saved on the host
 - docker cp does **not** provide persistent storage like volumes.

```
PS C:\Users\sreev> docker run -it --name cp-practice ubuntu
root@fcd3b9b311f0:/# echo "hello from container">/root/container.txt
root@fcd3b9b311f0:/# ls/root
bash: ls/root: No such file or directory
root@fcd3b9b311f0:/# ls /root
container.txt
root@fcd3b9b311f0:/# exit
exit
PS C:\Users\sreev>
```

RAM 0.87 GB CPU 2.92% Disk: 4.89 GB used (limit 1006.85 GB)

```
PS C:\Users\sreev> docker run -it --name cp-practice ubuntu
root@fcd3b9b311f0:/# echo "hello from container">/root/container.txt
root@fcd3b9b311f0:/# ls/root
bash: ls/root: No such file or directory
root@fcd3b9b311f0:/# ls /root
container.txt
root@fcd3b9b311f0:/# exit
exit
PS C:\Users\sreev> docker cp cp-practice:/root/container.txt C:\Users\sreev
Successfully copied 2.05kB to C:\Users\sreev
PS C:\Users\sreev>
```

RAM 0.80 GB CPU 0.00% Disk: 4.89 GB used (limit 1006.85 GB)

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\sreev> echo "hello from host">hello.txt
PS C:\Users\sreev> docker cp hello.txt cp-practice:/root/
Successfully copied 2.05kB to cp-practice:/root/
PS C:\Users\sreev>
```

```
Terminal
container.txt
root@fcd3b9b311f0:/# exit
exit
PS C:\Users\sreev> docker cp cp-practice:/root/container.txt C:\Users\sreev
Successfully copied 2.05kB to C:\Users\sreev
PS C:\Users\sreev> docker start -ai cp-practice
root@fcd3b9b311f0:/# ls /root/
container.txt  hello.txt
root@fcd3b9b311f0:/# cat /root/hello.txt
hello from host
root@fcd3b9b311f0:/#

RAM 0.86 GB CPU 6.72% Disk: 4.89 GB used (limit 1006.85 GB) >_ Update available
```

Practice question 3:

You have a running container named `my_app`, but it is not behaving as expected. You want to check its logs to debug errors and follow new log entries in real time. Write the command to view its logs and continuously monitor them.

```
hority

Run 'docker run --help' for more information
PS C:\Users\sreev> docker run -d --name my_app ubuntu sh -c "while true;do echo App is
running;sleep 2;done"
3dd172be10cfb826d2c6d8e018c9fd3aca7461bf09e6b44ca5c70232f8b79d70
PS C:\Users\sreev> docker logs -f my_app
App is running
App is running
```

```
App is running
App is running
App is running
App is running
App is running
App is running
App is running
App is running
App is running
App is running
App is running
```

Note: Docker logs are generated only while the container's main process is running, and `docker logs -f` streams them in real time.

When the container is stopped, the process ends and no new logs are produced, though previous logs remain accessible.

Practical on Bind Mount:

```
Command Prompt
Microsoft Windows [Version 10.0.26200.7705]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sreev>mkdir C:\docker-volume-demo

C:\Users\sreev>echo "Hello from host" > C:\docker-volume-demo\hostfile.txt

C:\Users\sreev>type C:\docker-volume-demo\hostfile.txt
"Hello from host"
```

```
Terminal
cv1
PS C:\Users\sreev> docker run -it --name cv1 -v C:\docker-volume-demo:\data ubuntu bash

root@decc5f3ade6c:/# cd /data
bash: cd: /data: No such file or directory
root@decc5f3ade6c:/# cd \data
bash: cd: data: No such file or directory
root@decc5f3ade6c:/# ls
\data  boot  etc   lib   media  opt   root  sbin  sys   usr
bin    dev   home  lib64 mnt    proc  run   srv   tmp   var
root@decc5f3ade6c:/# cd '\data'
```

RAM 0.86 GB CPU 7.53% Disk: 4.89 GB used (limit 1006.85 GB)

```
root@decc5f3ade6c:/# cd \data
bash: cd: data: No such file or directory
root@decc5f3ade6c:/# ls
\data  boot  etc   lib   media  opt   root  sbin  sys   usr
bin    dev   home  lib64 mnt    proc  run   srv   tmp   var
root@decc5f3ade6c:/# cd '\data'
root@decc5f3ade6c:/\data# ls
hostfile.txt
root@decc5f3ade6c:/\data# cat hostfile.txt
"Hello from host"
root@decc5f3ade6c:/\data#
```

RAM 0.86 GB CPU 6.36% Disk: 4.89 GB used (limit 1006.85 GB)

Note:

What is bind mount?

A **bind mount** directly maps a specific directory from the host system into a container.

The container and host share the same files, and changes are reflected in real time on both sides.

Bind mounts are commonly used in development but are not managed by Docker for data safety.

Practice Question 4:

- Create a Docker volume named studentdata.
- Run an Ubuntu container and mount studentdata at /student.
- Create a file inside the container and verify it persists after container deletion.
- Attach the same volume to another container and verify the file exists.
- Demonstrate data sharing between two containers using a shared volume.
- Show that deleting a container does NOT delete the volume data.

```
Terminal
# exit
PS C:\Users\sreev> docker volume create studentdata
studentdata
PS C:\Users\sreev> docker volume ls
DRIVER      VOLUME NAME
local       292d36e61a3fbc725b68a1c41f1ed5f8d959552e31e01c755764dea00fd01883
local       a06a9de2a6c86004aec5446f449cf76bf91ebd327650368478c0443fe8cc1eaa
local       a2277fd67f2d4a253809ed020c54ba48196a7cd8a2e2b530c1ed977d0abeece0
local       data
local       my_volume
local       studentdata

RAM 0.92 GB  CPU 0.00%  Disk: 4.89 GB used (limit 1006.85 GB)  >_ Update a
```

```
local       studentdata
PS C:\Users\sreev> docker run -it --name container1 -v studentdata:/student ubuntu sh
# cd /student
# echo "hi from stud">filestud.txt
# exit
PS C:\Users\sreev> delete rm container1
```

```
Terminal
PS C:\Users\sreev> docker rm container1
container1
PS C:\Users\sreev> docker volum els
docker: unknown command: docker volum

Run 'docker --help' for more information
PS C:\Users\sreev> docker volume ls
DRIVER      VOLUME NAME
local       292d36e61a3fbc725b68a1c41f1ed5f8d959552e31e01c755764dea00fd01883
local       a06a9de2a6c86004aec5446f449cf76bf91ebd327650368478c0443fe8cc1eaa
local       a2277fd67f2d4a253809ed020c54ba48196a7cd8a2e2b530c1ed977d0abeece0

RAM 0.93 GB  CPU 0.33%  Disk: 4.89 GB used (limit 1006.85 GB)  >_ Update a

local       studentdata
PS C:\Users\sreev> docker run -it --name container2 -v studentdata:/student ubuntu sh
# ls /student
filestud.txt
# cat /student/filestud.txt
hi from stud
#
```

```
PS C:\Users\sreev> docker run -it --name writer -v studentdata:/student ubuntu sh
# cd /student
# echo "sharing practice"
sharing practice
# echo "sharing practice" > fileprac.txt
#
```

```
Terminal
You have to remove (or rename) that container to be able to reuse that name.

Run 'docker run --help' for more information
PS C:\Users\sreev> docker run -it --name reader1 -v studentdata:/student ubuntu sh
# ls /student
fileprac.txt  filestud.txt
# cd /student/fileprac.txt
sh: 2: cd: can't cd to /student/fileprac.txt
# cat /student/fileprac.txt
sharing practice
#
```

RAM 0.91 GB CPU 2.59% Disk: 4.89 GB used (limit 1006.85 GB) >_ Update a

```
Terminal
PS C:\Users\sreev> docker rm writer
writer
PS C:\Users\sreev> docker rm reader
reader
PS C:\Users\sreev> docker rm container2
container2
PS C:\Users\sreev> docker volume ls
DRIVER    VOLUME NAME
local     292d36e61a3fbc725b68a1c41f1ed5f8d959552e31e01c755764dea00fd01883
local     a06a9de2a6c86004aec5446f449cf76bf91ebd327650368478c0443fe8cc1eaa
local     a2277fd67f2d4a253809ed020c54ba48196a7cd8a2e2b530c1ed977d0abeece0
```

RAM 0.99 GB CPU 0.00% Disk: 4.89 GB used (limit 1006.85 GB) >_ Update a

```
Terminal
container2
PS C:\Users\sreev> docker volume ls
DRIVER    VOLUME NAME
local     292d36e61a3fbc725b68a1c41f1ed5f8d959552e31e01c755764dea00fd01883
local     a06a9de2a6c86004aec5446f449cf76bf91ebd327650368478c0443fe8cc1eaa
local     a2277fd67f2d4a253809ed020c54ba48196a7cd8a2e2b530c1ed977d0abeece0
local     data
local     my_volume
local     student
local     studentdata
PS C:\Users\sreev>
```

RAM 0.99 GB CPU 0.08% Disk: 4.89 GB used (limit 1006.85 GB) >_ Update a

Practice Question-5:

1. On the **host system**, create a directory named **webdata**.
2. Inside it, create a file **index.html** with the content:

<h1>Welcome from Host</h1>

3. Run a **Nginx container** and bind mount the host directory to **/usr/share/nginx/html** using **--mount**.
4. Expose the container on port **8080**.
5. Open a browser and verify the page loads.
6. Modify **index.html** on the host to:
7. **<h1>Updated from Host</h1>**
8. Refresh the browser and observe the change.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\sreev> mkdir C:\webdata

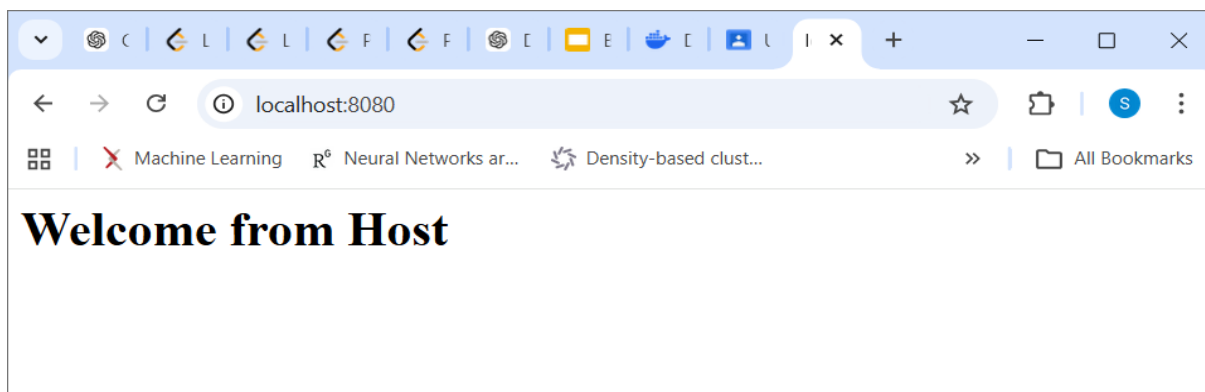
Directory: C:\

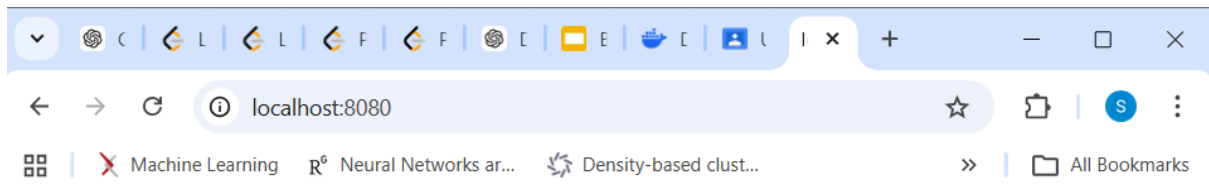
Mode                LastWriteTime         Length Name
----                -
d-----         11-02-2026    22:45         webdata

PS C:\Users\sreev> cd C:\webdata
PS C:\webdata> notepad index.html
PS C:\webdata>
```

```
local studentdata
PS C:\Users\sreev> docker run -d --name mynginx --mount type=bind,source="C:\webdata",target=/usr/share/nginx/html -p 8080:80 nginx
3ad189382cb1ef6e4dc54ecc2b4307216646deda6f19c5cbe43ad62b97c9714c
PS C:\Users\sreev>

RAM 0.93 GB CPU 0.00% Disk: 4.89 GB used (limit 1006.85 GB)
```





Updated from Host

```
3ad189382cb1ef6e4dc54ecc2b4307216646deda6f19c5cbe43ad62b97c9714c
PS C:\Users\sreev> docker exec -it mynginx sh
# cat /usr/share/nginx/html/index.html
<h1>Updated from Host</h1>#
```

RAM 0.96 GB CPU 0.25% Disk: 4.89 GB used (limit 1006.85 GB)