INT332

Docker version

Cmd= docker --version

```
PS C:\Users\harik> Docker --version
Docker version 29.2.0, build 0b9d198
PS C:\Users\harik>
```

Docker info

Cmd=docker info

```
PS C:\Users\harik> docker info
Client:
 Version:    29.2.0
 Context:    desktop-linux
 Debug Mode: false
 Plugins:
  ai: Docker AI Agent - Ask Gordon (Docker Inc.)
    Version:  v1.17.2
    Path:     C:\Program Files\Docker\cli-plugins\docker-ai.exe
```

1. Checking History

Cmd=docker history httpd

```
PS C:\Users\harik> docker history httpd
IMAGE          CREATED       CREATED BY                                      SIZE      COMMENT
b89c19a39051   2 weeks ago   CMD ["httpd-foreground"]                        0B        buildkit.dockerfile
.v0
<missing>      2 weeks ago   EXPOSE map[80/tcp:{}]                           0B        buildkit.dockerfile
.v0
<missing>      2 weeks ago   COPY httpd-foreground /usr/local/bin/ # buil…   20.5kB    buildkit.dockerfile
```

2. Downloading Image

Cmd=docker pull httpd

```
PS C:\Users\harik> docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
4f4fb700ef54: Pull complete
30973b009bab: Pull complete
85b0d820aa56: Pull complete
0c8d55a45c0d: Pull complete
fe1db8dd446f: Pull complete
996228ca33ab: Pull complete
```

3. Images

Cmd=docker images

```
PS C:\Users\harik> docker images
                                                                    i Info →  U  In Use
IMAGE          ID             DISK USAGE   CONTENT SIZE   EXTRA
httpd:latest   b89c19a39051        177MB         47.6MB
PS C:\Users\harik>
```

4. 1st example

Cmd=docker run httpd echo “Hello, World!”

```
PS C:\Users\harik> docker run httpd echo “Hello, World!”
Hello, World!
PS C:\Users\harik>
```

5. 2nd example

   Cmd= docker run --name my-container httpd echo “Hello, World!”

```
PS C:\Users\harik> docker run --name my-container httpd echo “Hello, World!”
Hello, World!
PS C:\Users\harik>
```

6. 3rd example : Setting environment

   Cmd=docker run -e MY_VAR=value httpd env

```
PS C:\Users\harik> docker run -e MY_VAR=value httpd env
PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=3655c410e041
MY_VAR=value
HTTPD_PREFIX=/usr/local/apache2
HTTPD_VERSION=2.4.66
HTTPD_SHA256=94d7ff2b42acbb828e870ba29e4cbad48e558a79c623ad3596e4116efcfea25a
HTTPD_PATCHES=
HOME=/root
```

7. 4th example: Example: Run a container from Image and display the variable name inside it.

   Cmd=docker run -it -e MY_NAME=Hari ubuntu bash

```
PS C:\Users\harik> docker run -it -e MY_NAME=Hari ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
a3629ac5b9f4: Pull complete
1baf05536e37: Download complete
Digest: sha256:cd1dba651b3080c3686ecf4e3c4220f026b521fb76978881737d24f200828b2b
Status: Downloaded newer image for ubuntu:latest
root@f3a877c65388:/#
```

8. 5th example :Multiple Environment Variables

   Cmd= docker run -e APP_ENV=production -e APP_VERSION=1.0 nginx

```
PS C:\Users\harik> docker run -e APP_ENV=production -e APP_VERSION=1.0 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
bae5a1799a80: Pull complete
46bf3a120c8e: Pull complete
f73400a233fd: Pull complete
7b6cb8ccac7b: Pull complete
47cd406a84ef: Pull complete
4f4efe02d542: Pull complete
a5d78d617315: Download complete
2e02dba24409: Download complete
```

9. Passing Environment Variable

Cmd= docker run -e APP_PORT nginx env

```
PS C:\Users\harik> docker run -e APP_PORT nginx env
HOSTNAME=989a2e3d628d
HOME=/root
PKG_RELEASE=1~trixie
DYNPKG_RELEASE=1~trixie
ACME_VERSION=0.3.1
```

```
HOSTNAME=989a2e3d628d
HOME=/root
PKG_RELEASE=1~trixie
DYNPKG_RELEASE=1~trixie
ACME_VERSION=0.3.1
NGINX_VERSION=1.29.5
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
NJS_VERSION=0.9.5
NJS_RELEASE=1~trixie
PWD=/
PS C:\Users\harik>
```

10. 7th example:

Cmd= docker run -it -d httpd

```
PS C:\Users\harik> docker run -it -d httpd
5a3a1bcae4fc1240b04a96f03b667da676de3efad0f8d916b6e999ddc772f80e
PS C:\Users\harik>
```

11. TASK #1:

Run Ubuntu container

Pass -e COLLEGE=CSE

Show echo $COLLEGE

Stop container then check what
Happened

Cmd= docker run -it -e COLLEGE=CSE ubuntu bash(inside container write this echo $COLLEGE)

```
PS C:\Users\harik> docker run -it -e COLLEGE=CSE ubuntu bash
root@0787fb72dc50:/# echo $COLLEGE
CSE
root@0787fb72dc50:/#
exit
PS C:\Users\harik> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS
```

```
CSE
root@0787fb72dc50:/#
exit
PS C:\Users\harik> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS                      PO
RTS     NAMES
0787fb72dc50   ubuntu         "bash"                   About a minute ago   Exited (0) 34 seconds ago
        bold_wu
92a5a74c759a   nginx          "/docker-entrypoint.…"   4 minutes ago        Exited (0) 4 minutes ago
        frosty_johnson
e1dba34a4c6b   nginx:latest   "/docker-entrypoint.…"   5 minutes ago        Up 5 minutes                80
```

12. Practice Questions on Basic Docker Commands

A.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?

Cmd= docker run -d --name DB-app -p 80:8082 mongo

```
PS C:\Users\harik> docker run -d --name DB-app -p 80:8082 mongo
Unable to find image 'mongo:latest' locally
latest: Pulling from library/mongo
5a8fa0179606: Pull complete
7bfa05e92f28: Pull complete
39291fd46646: Pull complete
8e6660ca41ae: Pull complete
d96ed84af028: Pull complete
04a8797a3d69: Pull complete
ca4018124b7d: Pull complete
```

```
39291fd46646: Pull complete
8e6660ca41ae: Pull complete
d96ed84af028: Pull complete
04a8797a3d69: Pull complete
ca4018124b7d: Pull complete
ea019893011f: Download complete
fd32ad5461df: Download complete
Digest: sha256:9e52bf8768ae4f59b9ca734845d15cee0e523ced14b900cc3692d241eec9154c
Status: Downloaded newer image for mongo:latest
7d9e3fc17182c408d035289605d389e5fe396ea328e75e49cdbce046bc67c17d
PS C:\Users\harik>
```

A.2.

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

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

•

Set an environment variable

APP_ENV=production.

•

Bind a local directory /app/data to

/data inside the container.

•

Name the container my_app.

•

Start an interactive terminal in an image called my_image?

•

Write the full command for the scenario above.

Cmd= docker run -it --name my_app -e APP_ENV=production -v C:\app\data:/data ubuntu bash(then cotainer open) root@5e808546f3ac:/# echo $APP_ENV

production

root@5e808546f3ac:/# cd /data

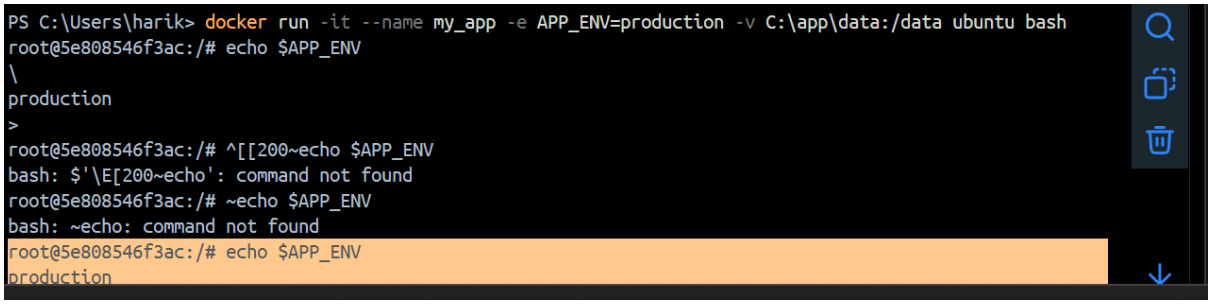
root@5e808546f3ac:/data# ls

root@5e808546f3ac:/data# echo "Hello Docker Volume" > test.txt

root@5e808546f3ac:/data# exit

exit

PS C:\Users\harik>



Docker-Container

1. 1. You need to start a new container using the nginx image

while setting an environment variable

ENV_MODE=production. Write the docker run command to

achieve this.

Cmd=docker run -d --name nginx_app -e ENV_MODE=production nginx

```
PS C:\Users\harik> docker run -d --name nginx_app -e ENV_MODE=production nginx
75d7ac675f0e270d33bde7bf6a7b9972d5ca72be887744f4c6265e87504c67a9
PS C:\Users\harik>
```

2. You have a running container named my_app, but it's not

behaving as expected. You want to check its logs to debug any

errors. Write the command to view its logs and follow new log

entries in real-time.

Cmd= docker logs -f my_app

```
PS C:\Users\harik> docker logs -f my_app
root@5e808546f3ac:/# echo $APP_ENV
\
production
>
root@5e808546f3ac:/# ^[[200~echo $APP_ENV
bash: $'\E[200~echo': command not found
root@5e808546f3ac:/# ~echo $APP_ENV
bash: ~echo: command not found
root@5e808546f3ac:/# echo $APP_ENV
production
```

```
bash: $'\E[200~echo': command not found
root@5e808546f3ac:/# ~echo $APP_ENV
bash: ~echo: command not found
root@5e808546f3ac:/# echo $APP_ENV
production
root@5e808546f3ac:/# cd /data
root@5e808546f3ac:/data# ls
root@5e808546f3ac:/data# echo "Hello Docker Volume" > test.txt
root@5e808546f3ac:/data# exit
exit
PS C:\Users\harik>
```

3. A container named web_server was stopped. Write the

command to:a) Start the container again.b) Stop the container

when needed.

Cmd= docker start web_server

```
PS C:\Users\harik> docker run -d --name web_server nginx
>>
e3b8e5412af13a35d57b8fb10e35a90f25d38e07f2545e36cd432b08d14e90fb
PS C:\Users\harik> ocker start web_server
ocker: The term 'ocker' is not recognized as a name of a cmdlet, function, script file, or executable prog
ram.
Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
PS C:\Users\harik> docker start web_server
web_server
PS C:\Users\harik>
```

Cmd= docker stop web_server

```
PS C:\Users\harik> docker stop web_server
>>
web_server
PS C:\Users\harik>
```

Volume:-

1. Create a Docker Volume

Cmd= docker volume create mydata

```
PS C:\Users\harik> docker volume create mydata
mydata
```

2. Inspect Volume Details

Cmd= docker volume inspect mydata

```
PS C:\Users\harik> docker volume inspect mydata
[
    {
        "CreatedAt": "2026-02-07T07:03:45Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/mydata/_data",
        "Name": "mydata",
        "Options": null,
        "Scope": "local"
    }
]
PS C:\Users\harik>
```

3. Run Container Using Volume

Cmd= docker run -d -v mydata:/app/data --name my_container ubuntu

```
PS C:\Users\harik> docker run -d -v mydata:/app/data --name my_container ubuntu
c1b1984eb554a753e5cee852778a298ba4572fbba1b3786a9853cc8742fcc46f
PS C:\Users\harik>
```

4. Store Data in the Volume

Cmd= docker run -it -v mydata:/app/data ubuntu sh

#

cd /app/data

echo "Hello Docker Volume" > file.txt

exit

PS C:\Users\harik

```
PS C:\Users\harik> docker run -it -v mydata:/app/data ubuntu sh
#
# cd /app/data
# echo "Hello Docker Volume" > file.txt
# exit
PS C:\Users\harik> ^C
PS C:\Users\harik>
```

5. Verify Data Persistence

Cmd= docker run -it -v mydata:/app/data ubuntu sh

cat /app/data/file.txt

```
PS C:\Users\harik> docker run -it -v mydata:/app/data ubuntu sh
# cat /app/data/file.txt
Hello Docker Volume
# ^C
#
```

6. docker volume rm mydata

cmd= docker volume rm mydata

```
PS C:\Users\harik> docker volume rm mydata
Error response from daemon: remove mydata: volume is in use - [c1b1984eb554a753e5cee852778a298ba4572fbba1b
3786a9853cc8742fcc46f, f90dc88fe16ce52d3910336d8d6a6a9dd4cdba9b0445cc75a508a9fedae64b07, 9e5614fe02cbc97b4
4ed3bd370e367dfe0a4c14de702a239fa022d8ffea0e60c]
PS C:\Users\harik>
```

7. Delete All Unused Volumes

Cmd= docker volume prune

```
S C:\Users\harik> docker volume prune
ARNING! This will remove anonymous local volumes not used by at least one contai
re you sure you want to continue? [y/N] y
eleted Volumes:
25ee7b1b2da562391ba3c5691f116d7800e1c83ff013676cbbfd01302cfb8b4
e5fa56104493e69e274fe0ecf84071bae60f86e108aa257488a683cdb9189c6
9244fc9c7d8945e60c899b0592ffbf8d09e55c2d374e0403b4b791f0fccd9a6
80ec7c9bf67f2b10ac8075045c04bf7504436a39eeb19a2d5af8a66d9c38cc5

otal reclaimed space: 649MB
S C:\Users\harik>
```

Volume-part-2

Example#1

1. mkdir C:\docker-volume-demo

2. echo Hello from HOST machine > C:\docker-volume-demo\hostfile.txt

3. docker run -it --name cv1 -v c:\docker-volume-demo:/data ubuntu bash

4. Inside container

ls

cd data

cat hostfile.txt

5. Modify file INSIDE container

echo "Modified inside container" >> /data/hostfile.txt

exit

```
PS C:\Users\harik> cd C:\docker-volume-demo
PS C:\docker-volume-demo> echo Hello from HOST machine > C:\docker-volume-demo\hostfile.txt
PS C:\docker-volume-demo> docker run -it --name cv1 -v C:\docker-volume-demo:/data ubuntu bash
root@a522497e3d98:/# ls
cd /data
cat hostfile.txt
echo "Modified inside container" >> /data/hostfile.txt
exit
```

```
PS C:\docker-volume-demo> docker run -it --name cv1 -v C:\docker-volume-demo:/data ubuntu bash
exit
ls
cd /data
exit "Modified inside container" >> /data/hostfile.txt
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
Hello
from
HOST
machine
exit
```

Practice Question

- Create a Docker volume named studentdata.

  Cmd = docker volume create studentdata

  docker volume ls

docker volume inspect studentdata

```
PS C:\docker-volume-demo> docker volume create studentdata
>> docker volume ls
>> docker volume inspect studentdata
>>
studentdata
DRIVER    VOLUME NAME
local     0a70d66252496da4cf79d846c2fd0c706984b5160a1081d486d8d87c4b89a8cf
local     68a56098f44062bbb2ea1e8e960e701b5639ccf67a0d944643a7448bd347a94c
local     0087c2c444c7a8e59eaedacb675cf9c9a56bb86207502827627739cfdd01aaa6
local     6245df7e05b39020b192edc5b2d3ffaab00ded54c029653f98ad4eaa10531f67
local     mydata
local     studentdata
[
    {
        "CreatedAt": "2026-02-05T07:12:16Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/studentdata/_data",
        "Name": "studentdata",
        "Options": null,
        "Scope": "local"
    }
]
PS C:\docker-volume-demo>
```

- Run an Ubuntu container and mount studentdata at /student.

Cmd= docker run -it --name c1 -v studentdata:/student ubuntu bash

root@f00c3069d592:/# cd /student

echo "Hello from container c1" > note.txt

ls

cat note.txt

exit

```
PS C:\docker-volume-demo> docker run -it --name c1 -v studentdata:/student ubuntu bash
root@f00c3069d592:/# cd /student
echo "Hello from container c1" > note.txt
ls
cat note.txt
exit
info.txt  note.txt  testfile.txt  testfile2.txt
Hello from container c1
exit
PS C:\docker-volume-demo>
```

- Create a file inside the container and verify it persists after container deletion.

Cmd= docker rm c1

```
PS C:\docker-volume-demo> docker rm c1
c1
PS C:\docker-volume-demo>
```

- Attach the same volume to another container and verify the file exists.

Cmd= docker run -it --name c2 -v studentdata:/student ubuntu bash

root@5dbc3b1c57b1:/# ls /student

cat /student/note.txt

exit

```
PS C:\docker-volume-demo> docker run -it --name c2 -v studentdata:/student ubuntu bash
root@5dbc3b1c57b1:/# ls /student
cat /student/note.txt
exit
info.txt  note.txt  testfile.txt  testfile2.txt
Hello from container c1
exit
PS C:\docker-volume-demo>
```

- Demonstrate data sharing between two containers using a shared volume.

Cmd= docker run -d --name c3 -v studentdata:/student ubuntu sleep infinity

docker run -d --name c4 -v studentdata:/student ubuntu sleep infinity

```
PS C:\docker-volume-demo> docker run -d --name c3 -v studentdata:/student ubuntu sleep infinity
>> docker run -d --name c4 -v studentdata:/student ubuntu sleep infinity
>>
f1b8399775aa618a7ec134686ae883de8c17d90d4308c283d138ba2131e413cc
682ea5b5075477a4121659ec51d9ba5cd20ccde0a9de77a9bb8111a9cbd9edfa
PS C:\docker-volume-demo> docker exec -it c3 bash -lc "echo 'Data written by c3' >> /student/shared.txt && cat /student/shared.txt"
>>
Data written by c3
PS C:\docker-volume-demo>
```

- Show that deleting a container does NOT delete the volume data.

Cmd= docker rm -f c2 c3 c4

```
>>
Data written by c3
PS C:\docker-volume-demo> docker rm -f c2 c3 c4
c2
c3
c4
PS C:\docker-volume-demo>
```

Bind Mount

Host folder

C:\docker\website\index.html

Run container

docker run -it --mounttype=bind,source="C:\docker\website",target=/usr/share/nginx/html -p 8080:80 nginx

Edit file on HOST

<h1>Hello from Host</h1>

Browser

http://localhost:8080

Effects .....

Host directory is **directly mapped**

Changes on host reflect **instantly**

Container restart does NOT affect data

Best for development, live code editing

Pratice question

To demonstrate **real-time file synchronization between host and container**.

i. On the **host system**, create a directory named **webdata**.

ii. Inside it, create a file **index.html** with the content:

<h1>Welcome from Host</h1>

iii. Run an **Nginx container** and bind mount the host directory to **/usr/share/nginx/html** using --mount.

iii. Expose the container on port **8080**.

iii. Open a browser and verify the page loads.

iii. Modify index.html on the host to:

iii. <h1>Updated from Host</h1>

iii. Refresh the browser and observe the change.