Docker Intermediate Operations – Containers, Ports & Volumes

---

1. Naming Containers

Objective:

To give a meaningful name to a container instead of using the random auto-generated name.

Command Syntax:

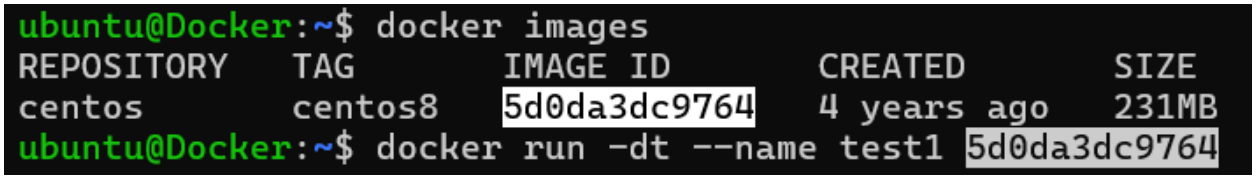
docker run -d --name <container_name> <image_name>

Example:

docker run -d --name test1 nginx

✅ Explanation:

- `--name` → Assigns a custom name to the container.




```
ubuntu@Docker:~$ docker run -dt --name test1 5d0da3dc9764
1810764e3b49e5341a51ed3b7935da452dc872a9c38ca6816e7c5b19d2e8d13c
ubuntu@Docker:~$ docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED          STATUS          PORTS     NAMES
1810764e3b49   5d0da3dc9764   "/bin/bash"   11 seconds ago   Up 10 seconds             test1
ubuntu@Docker:~$
```

```
ubuntu@Docker:~$ docker run -dt --name test2 centos:centos7
Unable to find image 'centos:centos7' locally
centos7: Pulling from library/centos
2d473b07cdd5: Pull complete
Digest: sha256:be65f488b7764ad3638f236b7b515b3678369a5124c47b8d32916d6487418ea4
Status: Downloaded newer image for centos:centos7
644a4b9602f7f72d09e61dd845a6008dfe68f5ddc692f6fc447fbed717aff0fa
ubuntu@Docker:~$ docker ps
CONTAINER ID   IMAGE           COMMAND       CREATED              STATUS              PORTS     NAMES
644a4b9602f7   centos:centos7  "/bin/bash"   6 seconds ago        Up 6 seconds                  test2
1810764e3b49   5d0da3dc9764    "/bin/bash"   About a minute ago   Up About a minute             test1
ubuntu@Docker:~$
```

---

2. Port Mapping (Port Forwarding and Assigning)

Objective:

Expose container ports to the host system so external users can access containerized applications.

Types of Port Mapping:

1. Port Forwarding(-P)

Maps a specific host port to a container port.

docker run -d -P nginx

```
ubuntu@Docker:~$ docker run -dt --name web1 -P httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
d7ecded7702a: Pull complete
fab98c44430d: Pull complete
4f4fb700ef54: Pull complete
13c22d886563: Pull complete
4870f70a8556: Pull complete
233dec01418c: Pull complete
Digest: sha256:ecfd5ca1bfe1fc5e44a5836c5188bde7f397b50c7a5bb603a017543e29948a01
Status: Downloaded newer image for httpd:latest
6038e4bf4b6d07ac79fd00ba9e20a11ccc08a1c23748d3ec0c1bd0e8f04e0572
ubuntu@Docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS         PORTS                                                  NAMES
6038e4bf4b6d   httpd     "httpd-foreground"   5 seconds ago   Up 4 seconds   0.0.0.0:32768->80/tcp, [::]:32768->80/tcp              web1
ubuntu@Docker:~$
```

```
ubuntu@Docker:~$ docker run -dt httpd
04ab8d181ae30705dd1f68a6c6c6b339cb8adae1dad2bc63190dfd79a495baa2
ubuntu@Docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS         PORTS     NAMES
04ab8d181ae3   httpd     "httpd-foreground"   9 seconds ago   Up 8 seconds   80/tcp    reverent_austin
ubuntu@Docker:~$
```

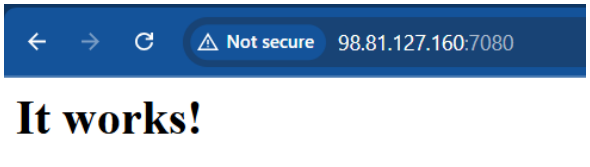
```
ubuntu@Docker:~$ docker run -dt --name web2 -P httpd
dffe16c99246a53a0556b0766d84bb7069413680494ff06422db78ab9f70ad93
ubuntu@Docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED             STATUS             PORTS                                                  NAMES
dffe16c99246   httpd     "httpd-foreground"   4 seconds ago       Up 3 seconds       0.0.0.0:32769->80/tcp, [::]:32769->80/tcp              web2
04ab8d181ae3   httpd     "httpd-foreground"   About a minute ago  Up About a minute  80/tcp                                                 reverent_austin
```

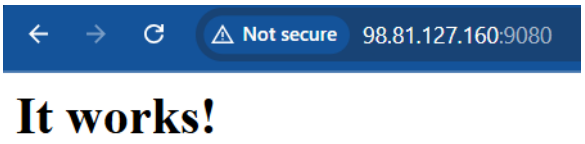
2. Port Assignment(-p)

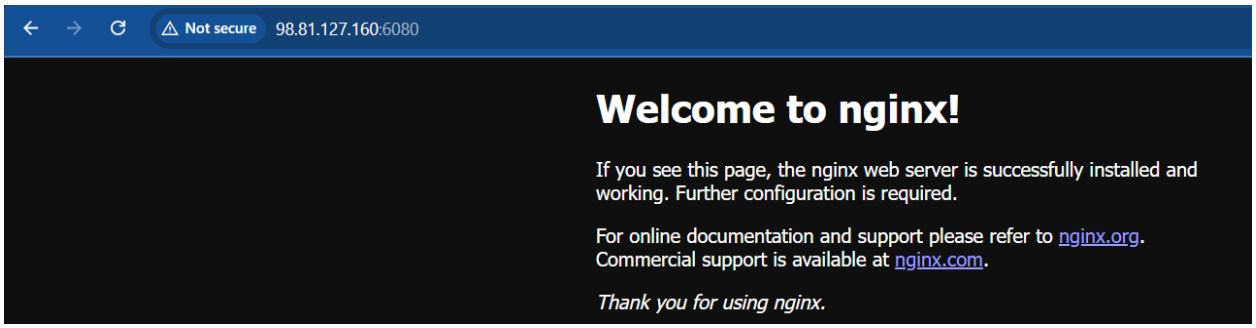
Docker automatically assigns a random available port on the host.

docker run -d -p 8080:80 nginx

```
ubuntu@Docker:~$ docker run -dt --name test1 -p 7080:80 httpd
49960f2e4367f47a4296c6377308e9196342171439df9d99d0cb51f1cd3134e8
ubuntu@Docker:~$ docker run -dt --name test2 -p 9080:80 httpd
745584302af5c84817eb15fe7db682966330d64791c5a6eff2ea5fe64ab09f2d
ubuntu@Docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED          STATUS          PORTS                                                  NAMES
745584302af5   httpd     "httpd-foreground"   33 seconds ago   Up 33 seconds   0.0.0.0:9080->80/tcp, [::]:9080->80/tcp                test2
49960f2e4367   httpd     "httpd-foreground"   48 seconds ago   Up 48 seconds   0.0.0.0:7080->80/tcp, [::]:7080->80/tcp                test1
ubuntu@Docker:~$
```








```
ubuntu@Docker:~$ docker run -dt --name test3 -p 6080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
d7ecded7702a: Already exists
266626526d42: Pull complete
320b0949be89: Pull complete
d921c57c6a81: Pull complete
9def903993e4: Pull complete
52bc359bcbd7: Pull complete
e2f8e296d9df: Pull complete
Digest: sha256:1beed3ca46acebe9d3fb62e9067f03d05d5bfa97a00f30938a0a3580563272ad
Status: Downloaded newer image for nginx:latest
5f9d075caaa66e0e0b754351dc42449cbef0bebbb9ff03b2f842574f37b4f29c
ubuntu@Docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                       NAMES
5f9d075caaa6   nginx     "/docker-entrypoint.…"   11 seconds ago   Up 10 seconds   0.0.0.0:6080->80/tcp, [::]:6080->80/tcp     test3
745584302af5   httpd     "httpd-foreground"       4 minutes ago    Up 4 minutes    0.0.0.0:9080->80/tcp, [::]:9080->80/tcp     test2
49960f2e4367   httpd     "httpd-foreground"       4 minutes ago    Up 4 minutes    0.0.0.0:7080->80/tcp, [::]:7080->80/tcp     test1
ubuntu@Docker:~$
```

Command Syntax for Port Assignment:

docker run -d -p <host_port>:<container_port> <image_name>

Example:

docker run -d -p 8080:80 nginx

✅ Explanation:

- `8080` → Host machine port
- `80` → Container's internal port (used by Nginx)
- Access app via `http://<VM-IP>:8080`

To check which port was assigned:

- docker ps

---

3. Docker Volumes

Objective:

Persist container data beyond container lifecycle (stop/remove).

Why Volumes?

- Data remains even if the container is deleted.
- Enables data sharing between containers.
- Helps maintain application data consistency.

3.1. Types of Volumes

A. Anonymous (Unnamed) Volume

- Docker automatically creates it when you use the `-v` flag without specifying a name.
- Stored under `/var/lib/docker/volumes/` with a random hash name.

Command:

docker run -d -v /data nginx

Check created volume:

docker volume ls

```
ubuntu@Docker:~$ docker volume ls
DRIVER    VOLUME NAME
ubuntu@Docker:~$ docker run -dt --name web1 -v /app/data httpd
05ef585b19025ea386cb1153e16fa5fe0d845e4e721d991adf33acb93ccaac41
ubuntu@Docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS         PORTS     NAMES
05ef585b1902   httpd     "httpd-foreground"   4 seconds ago   Up 3 seconds   80/tcp    web1
ubuntu@Docker:~$
```

```
ubuntu@Docker:~$ docker exec -it web1 /bin/bash
root@05ef585b1902:/usr/local/apache2# ls
bin  build  cgi-bin  conf  error  htdocs  icons  include  logs  modules
root@05ef585b1902:/usr/local/apache2# cd
root@05ef585b1902:~# ls
root@05ef585b1902:~# cd /
root@05ef585b1902:/# ls
app  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@05ef585b1902:/# cd app
root@05ef585b1902:/app# ls
data
root@05ef585b1902:/app# cd data/
root@05ef585b1902:/app/data# ls
root@05ef585b1902:/app/data# touch file.txt
root@05ef585b1902:/app/data#
```

```
ubuntu@Docker:~$ docker volume ls
DRIVER    VOLUME NAME
local     dd45eed2ff82604fe34c79fc08cf2ff35a15454667c090bcc48a75c7c6f00789
ubuntu@Docker:~$
```

B. Named Volume

- You explicitly assign a name to the volume for easy management.
- Easier to reuse across multiple containers.

Command:

docker run -dt –name web2 -v web2:/app/data httpd

✅ **Explanation:**

- `web2` → Volume name (stored in Docker-managed storage).
- `/app/data` → Mount path inside the container.

Inspect volume:

docker volume inspect web2

```
ubuntu@Docker:~$ docker run -dt --name web2 -v web2:/app/data httpd
2f7bdef4ebe365dda9cefdb405ea8c3816bf2ae98928e24874a332228d137f70
ubuntu@Docker:~$ docker exec -it web2 /bin/bash
root@2f7bdef4ebe3:/usr/local/apache2# cd /
root@2f7bdef4ebe3:/# ls
app  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@2f7bdef4ebe3:/# cd app/data/
root@2f7bdef4ebe3:/app/data# ls
root@2f7bdef4ebe3:/app/data# touch file.txt
root@2f7bdef4ebe3:/app/data# ls
file.txt
root@2f7bdef4ebe3:/app/data#
```

```
ubuntu@Docker:~$ docker volume ls
DRIVER    VOLUME NAME
local     dd45eed2ff82604fe34c79fc08cf2ff35a15454667c090bcc48a75c7c6f00789
local     web2
ubuntu@Docker:~$
```

C. Host (Bind Mount) Volume

- Links a specific host directory to a container directory.
- Useful for development where you want real-time file access.

Command:

docker run -dt –name host-vol -v /home/ubuntu/scripts:/app/data nginx

✅ **Explanation:**

- `/home/ubuntu/html` → Directory on host
- `/app/data` → Directory inside container
- Any changes on the host reflect in the container (and vice versa).

```
ubuntu@Docker:~$ mkdir scripts
ubuntu@Docker:~$ ls
scripts
ubuntu@Docker:~$ cd scripts/
ubuntu@Docker:~/scripts$ touch host.txt
ubuntu@Docker:~/scripts$ cat > host.txt
This is about host volume.
^C
ubuntu@Docker:~/scripts$ cat host.txt
This is about host volume.
ubuntu@Docker:~/scripts$
```

```
ubuntu@Docker:~/scripts$ docker run -dt --name host-vol -v /home/ubuntu/scripts:/app/data nginx
f44956d3d70451e0e933be8f0760e2b36981c622fe60b279a95ae11a84e85749
ubuntu@Docker:~/scripts$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
f44956d3d704   nginx     "/docker-entrypoint.…"   4 seconds ago    Up 4 seconds    80/tcp    host-vol
2f7bdef4ebe3   httpd     "httpd-foreground"       11 minutes ago   Up 11 minutes   80/tcp    web2
05ef585b1902   httpd     "httpd-foreground"       18 minutes ago   Up 18 minutes   80/tcp    web1
ubuntu@Docker:~/scripts$ docker volume ls
DRIVER    VOLUME NAME
local     dd45eed2ff82604fe34c79fc08cf2ff35a15454667c090bcc48a75c7c6f00789
local     web2
ubuntu@Docker:~/scripts$
```

Data getting reflected from Container to Host VM:

```
ubuntu@Docker:~$ docker exec -it host-vol /bin/bash
root@f44956d3d704:/# ls
app  boot  docker-entrypoint.d   etc   lib    media  opt   root  sbin  sys  usr
bin  dev   docker-entrypoint.sh  home  lib64  mnt    proc  run   srv   tmp  var
root@f44956d3d704:/# cd app/
root@f44956d3d704:/app# ls
data
root@f44956d3d704:/app# cd data/
root@f44956d3d704:/app/data# ls
host.txt
root@f44956d3d704:/app/data# cat host.txt
This is about host volume.
root@f44956d3d704:/app/data#
```

```
root@f44956d3d704:/app/data# touch Hosted-volume.txt
root@f44956d3d704:/app/data# cat > Hosted-volume.txt
This is from the Container to the host VM.
^C
root@f44956d3d704:/app/data# cat Hosted-volume.txt
This is from the Container to the host VM.
root@f44956d3d704:/app/data# exit
exit
ubuntu@Docker:~$ cd scripts/
ubuntu@Docker:~/scripts$ ls
Hosted-volume.txt  host.txt
ubuntu@Docker:~/scripts$ cat Hosted-volume.txt
This is from the Container to the host VM.
ubuntu@Docker:~/scripts$
```

D. tmpfs Volume

- Stores data in memory (RAM), **not persisted** on disk.
- Fast access but data is lost once container stops.

Command:

docker run -d --tmpfs /app/cache nginx

✅ **Use Case:**
For temporary data like caches, session files, or buffers.

```
ubuntu@Docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
f44956d3d704   nginx     "/docker-entrypoint.…"   18 minutes ago   Up 18 minutes   80/tcp    host-vol
2f7bdef4ebe3   httpd     "httpd-foreground"       30 minutes ago   Up 30 minutes   80/tcp    web2
05ef585b1902   httpd     "httpd-foreground"       36 minutes ago   Up 36 minutes   80/tcp    web1
ubuntu@Docker:~$ docker run -d --name httpd-temp --mount type=tmpfs,destination=/usr/local/apache2/data,tmpfs-size=10m httpd
6c8863f0fee3918cb5ee38516684561b61e0873a70af299e7eef463acd28f68a
ubuntu@Docker:~$ docker exec -it httpd-temp /bin/bash
```

```
root@6c8863f0fee3:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay         6.8G  3.4G  3.4G  50% /
tmpfs            64M     0   64M   0% /dev
shm              64M     0   64M   0% /dev/shm
/dev/root       6.8G  3.4G  3.4G  50% /etc/hosts
tmpfs            10M     0   10M   0% /usr/local/apache2/data
tmpfs           479M     0  479M   0% /proc/acpi
tmpfs           479M     0  479M   0% /proc/scsi
tmpfs           479M     0  479M   0% /sys/firmware
root@6c8863f0fee3:/#
```

---

4. Volume Management Commands

List All Volumes:

docker volume ls

```
ubuntu@Docker:~$ docker volume ls
DRIVER    VOLUME NAME
local     dd45eed2ff82604fe34c79fc08cf2ff35a15454667c090bcc48a75c7c6f00789
local     web2
```

Inspect a Volume:

docker volume inspect <volume_name>

```
ubuntu@Docker:~$ docker volume inspect web2
[
    {
        "CreatedAt": "2025-11-12T11:35:49Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/web2/_data",
        "Name": "web2",
        "Options": null,
        "Scope": "local"
    }
]
ubuntu@Docker:~$
```

Remove a Specific Volume:

docker volume rm <volume_name>

```
ubuntu@Docker:~$ docker volume rm dd45eed2ff82604fe34c79fc08cf2ff35a15454667c090bcc48a75c7c6f00789
dd45eed2ff82604fe34c79fc08cf2ff35a15454667c090bcc48a75c7c6f00789
ubuntu@Docker:~$ docker volume ls
DRIVER    VOLUME NAME
local     web2
ubuntu@Docker:~$
```

Remove All Unused Volumes (Prune):

docker volume prune or docker system prune --all --volumes(Never Use this it's very powerfull).

```
ubuntu@Docker:~$ docker volume prune
WARNING! This will remove anonymous local volumes not used by at least one container.
Are you sure you want to continue? [y/N] Y
Total reclaimed space: 0B
```

```
ubuntu@Docker:~$ docker volume ls
DRIVER    VOLUME NAME
ubuntu@Docker:~$
```

⚠️ **Caution:**
This deletes all volumes **not used by any container**.

---

5. Examples Summary

Task	Command	Description
Create named container	`docker run -d --name web nginx`	Assigns container name
Map port	`docker run -d -p 8080:80 nginx`	Forward host port 8080 to container port 80
Auto port mapping	`docker run -d -P nginx`	Assigns random host port
Anonymous volume	`docker run -d -v /data nginx`	Creates unnamed volume
Named volume	`docker run -d -v myvol:/app nginx`	Creates reusable volume
Host bind mount	`docker run -d -v /home/charan/data:/app nginx`	Mounts local directory
tmpfs volume	`docker run -d --tmpfs /cache nginx`	Stores temporary data in memory
Remove one volume	`docker volume rm myvol`	Deletes selected volume
Remove all unused	`docker volume prune`	Deletes unused volumes

---

6. Best Practices

- Always use **named volumes** for data you want to keep.
- It is not a good practice to use **named** or **unnamed volumes** in **production**; it's best to go with **Host/bind volumes** as the **named or unnamed volumes** are managed by **Docker**. If it crashes then the data is gone. So use the **Host/Bind** volume.

- Use **bind mounts** for development to sync local code changes.

- Use **tmpfs volumes** for fast, temporary data (no persistence).

- Regularly prune unused volumes to save disk space.