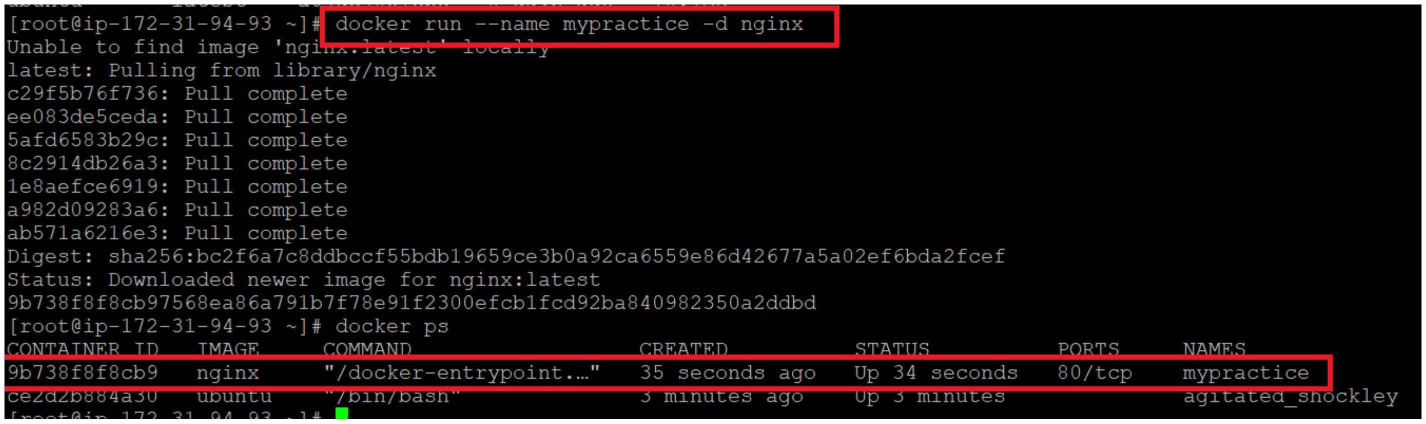
DOCKER Tasks:

31-01-2025

Install docker desktop in windows

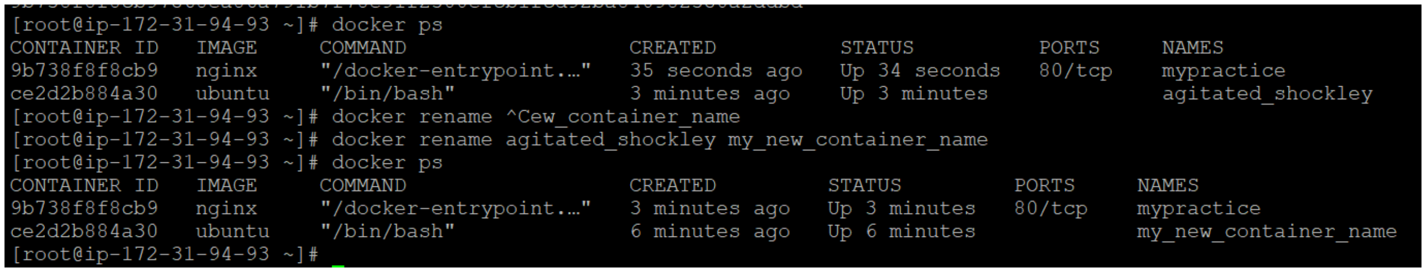
03-02-2025

How to name Docker container?



```
[root@ip-172-31-94-93 ~]# docker run --name mypractice -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
c29f5b76f736: Pull complete
ee083de5ceda: Pull complete
5afd6583b29c: Pull complete
8c2914db26a3: Pull complete
1e8aefce6919: Pull complete
a982d09283a6: Pull complete
ab571a6216e3: Pull complete
Digest: sha256:bc2f6a7c8ddbccf55bdb19659ce3b0a92ca6559e86d42677a5a02ef6bda2fcef
Status: Downloaded newer image for nginx:latest
9b738f8f8cb97568ea86a791b7f78e91f2300efcb1fcd92ba840982350a2ddbd
[root@ip-172-31-94-93 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
9b738f8f8cb9   nginx     "/docker-entrypoint.…"   35 seconds ago   Up 34 seconds   80/tcp    mypractice
ce2d2b884a30   ubuntu    "/bin/bash"              3 minutes ago    Up 3 minutes              agitated_shockley
```

How to rename it if already exists?



```
[root@ip-172-31-94-93 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
9b738f8f8cb9   nginx     "/docker-entrypoint.…"   35 seconds ago   Up 34 seconds   80/tcp    mypractice
ce2d2b884a30   ubuntu    "/bin/bash"              3 minutes ago    Up 3 minutes              agitated_shockley
[root@ip-172-31-94-93 ~]# docker rename ^Cew_container_name
[root@ip-172-31-94-93 ~]# docker rename agitated_shockley my_new_container_name
[root@ip-172-31-94-93 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
9b738f8f8cb9   nginx     "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes   80/tcp    mypractice
ce2d2b884a30   ubuntu    "/bin/bash"              6 minutes ago   Up 6 minutes             my_new_container_name
[root@ip-172-31-94-93 ~]#
```

Create account in DOckerhub!
Created

How to set max resource limitations?

To control how much CPU and memory a Docker container can use, you can specify resource limits using the `--memory` and `--cpus` flags.

1. Setting Memory Limits

Use the `--memory` flag to specify the maximum amount of RAM a container can use:

```
docker run --memory=512m -d nginx
```

- `512m` → **Limits memory usage to 512MB**
- **You can also use `g` for gigabytes (e.g., `1g` for 1GB)**

To prevent the container from exceeding its limit, use:

```
docker run --memory=512m --memory-swap=512m -d nginx
```

This ensures the container cannot swap beyond its memory limit.

2. Setting CPU Limits

Use the `--cpus` flag to limit CPU usage:

```
docker run --cpus="0.5" -d nginx (for 1 cpu)
```

How to remove docker image?

1. Remove a Specific Docker Image

Use the image ID or image name:

First need to stop the container if the image is running in that container before deleting the image.
1. Stop the Running Container

If a container is using the image, stop it:

docker stop 9b738f8f8cb9

2. Remove the Stopped Container

Once the container is stopped, remove it:

docker rm 9b738f8f8cb9

3. Remove the Image

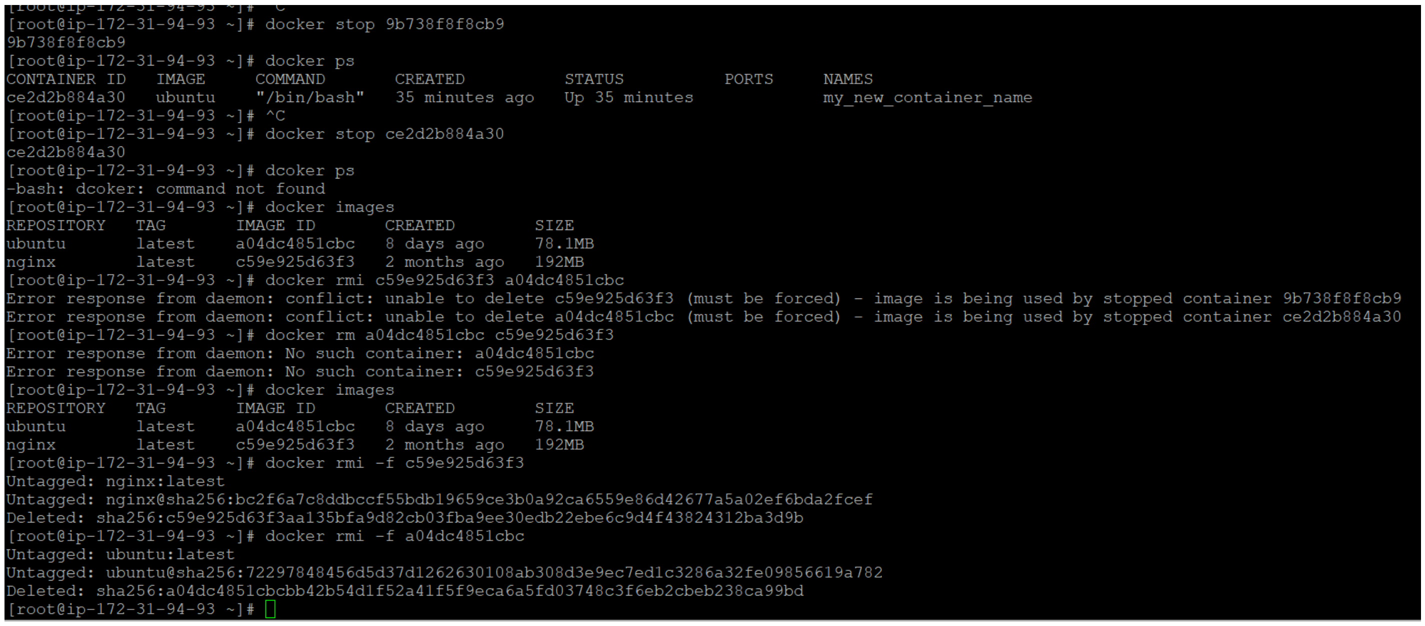
Now, try removing the image again:

docker rmi c59e925d63f3

4. Force Remove (If Necessary)

If the image is still not removed, you can use:

docker rmi -f c59e925d63f3



```
[root@ip-172-31-94-93 ~]# docker stop 9b738f8f8cb9
9b738f8f8cb9
[root@ip-172-31-94-93 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
ce2d2b884a30   ubuntu    "/bin/bash"   35 minutes ago   Up 35 minutes             my_new_container_name
[root@ip-172-31-94-93 ~]# ^C
[root@ip-172-31-94-93 ~]# docker stop ce2d2b884a30
ce2d2b884a30
[root@ip-172-31-94-93 ~]# dcoker ps
-bash: dcoker: command not found
[root@ip-172-31-94-93 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       latest    a04dc4851cbc   8 days ago     78.1MB
nginx        latest    c59e925d63f3   2 months ago   192MB
[root@ip-172-31-94-93 ~]# docker rmi c59e925d63f3 a04dc4851cbc
Error response from daemon: conflict: unable to delete c59e925d63f3 (must be forced) - image is being used by stopped container 9b738f8f8cb9
Error response from daemon: conflict: unable to delete a04dc4851cbc (must be forced) - image is being used by stopped container ce2d2b884a30
[root@ip-172-31-94-93 ~]# docker rm a04dc4851cbc c59e925d63f3
Error response from daemon: No such container: a04dc4851cbc
Error response from daemon: No such container: c59e925d63f3
[root@ip-172-31-94-93 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       latest    a04dc4851cbc   8 days ago     78.1MB
nginx        latest    c59e925d63f3   2 months ago   192MB
[root@ip-172-31-94-93 ~]# docker rmi -f c59e925d63f3
Untagged: nginx:latest
Untagged: nginx@sha256:bc2f6a7c8ddbccf55bdb19659ce3b0a92ca6559e86d42677a5a02ef6bda2fcef
Deleted: sha256:c59e925d63f3aa135bfa9d82cb03fba9ee30edb22ebe6c9d4f43824312ba3d9b
[root@ip-172-31-94-93 ~]# docker rmi -f a04dc4851cbc
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:72297848456d5d37d1262630108ab308d3e9ec7ed1c3286a32fe09856619a782
Deleted: sha256:a04dc4851cbcbb42b54d1f52a41f5f9eca6a5fd03748c3f6eb2cbeb238ca99bd
[root@ip-172-31-94-93 ~]#
```

How can you access port forwarding without using port farwarding

Exit status code for docker container? 137

Common Docker Exit Status Codes:

1. **Exit Code 0:**
   - **Meaning: The process inside the container ran successfully and exited without errors.**
   - **Example: If you run a container and it completes the task (like running a script or application), it will exit with status code `0`.**
2. **Exit Code 1:**
   - **Meaning: The process inside the container encountered a general error. This is a common "failure" exit code for many types of errors in a program or script.**
   - **Example: A command in a Dockerfile or entrypoint script fails, or the program inside the container encounters an issue.**
3. **Exit Code 137:**
   - **Meaning: The container was terminated by a `SIGKILL` signal, typically due to the host system running out of resources, like memory. This is often seen when the container exceeds its memory limit or if the system administrator manually kills the container.**
   - **Example: If your container uses too much memory or if the system is low on memory, the process may be killed and return this exit code.**
4. **Exit Code 143:**
   - **Meaning: The container was terminated by a `SIGTERM` signal, indicating that the process inside was asked to gracefully shut down.**

    - **Example: This can happen when the container is stopped using `docker stop`, which sends a `SIGTERM` signal to gracefully shut down the process.**
5. **Exit Code 137 (Container OOM Kill):**
    - **Meaning: The container was killed because it was using too much memory (OOM - Out of Memory). This typically happens when Docker's memory limit is exceeded, and the container gets killed by the operating system.**
    - **Example: A container running a memory-intensive application might exceed the memory limit, leading to this exit code.**

How to Check the Exit Code:

To check the exit code of a container, you can use the following Docker command:

```
docker inspect <container_id> --format '{{.State.ExitCode}}'
```

Alternatively, you can check the logs of a container after it has exited:

```
docker logs <container_id>
```

Example:

```
$ docker run --name my-container my-image
$ docker inspect my-container --format '{{.State.ExitCode}}'
```

This will output the exit code of the container. If the container completed successfully, it will return `0`. If there was an error, it will return a different exit code (e.g., `1` for a general error).

Summary of Some Key Exit Status Codes:

- **`0`: Success.**
- **`1`: General error.**
- **`137`: Killed by `SIGKILL` (out of memory or forcefully killed).**
- **`143`: Killed by `SIGTERM` (graceful shutdown).**

04-02-2025

1. How can we do port forwarding for an existing container?

If you have an **existing Docker container** running without exposing a port, you cannot modify its port mapping directly. However, you have a few workarounds:

Option 1: Use `docker network` with a New Container

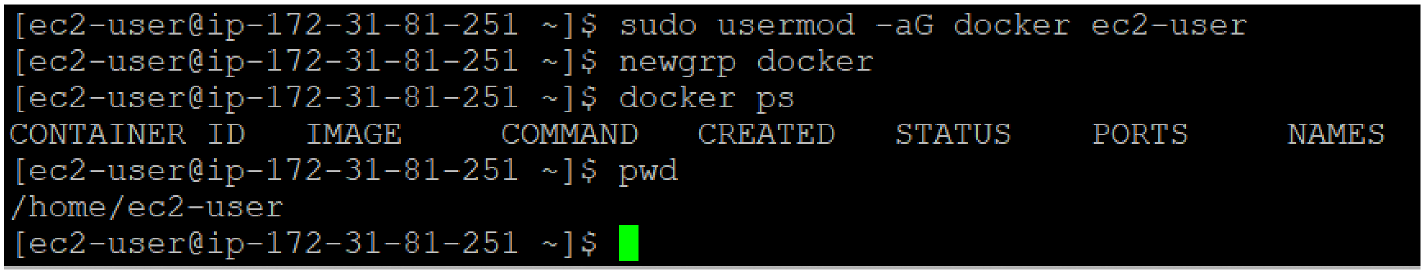
If you cannot restart the existing container, you can create another container in the same network and forward ports to it.

```
docker network create my_network
docker network connect my_network existing_container
docker run -d --name port_forwarder --network my_network -p 8080:80 nginx
```

This allows you to access the service running in `existing_container` via `port_forwarder`.

05-02-2025

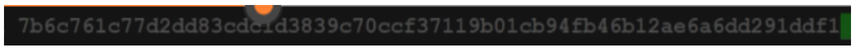
1. How to access docker commands by ec2-user?



```
[ec2-user@ip-172-31-81-251 ~]$ sudo usermod -aG docker ec2-user
[ec2-user@ip-172-31-81-251 ~]$ newgrp docker
[ec2-user@ip-172-31-81-251 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-81-251 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-81-251 ~]$
```

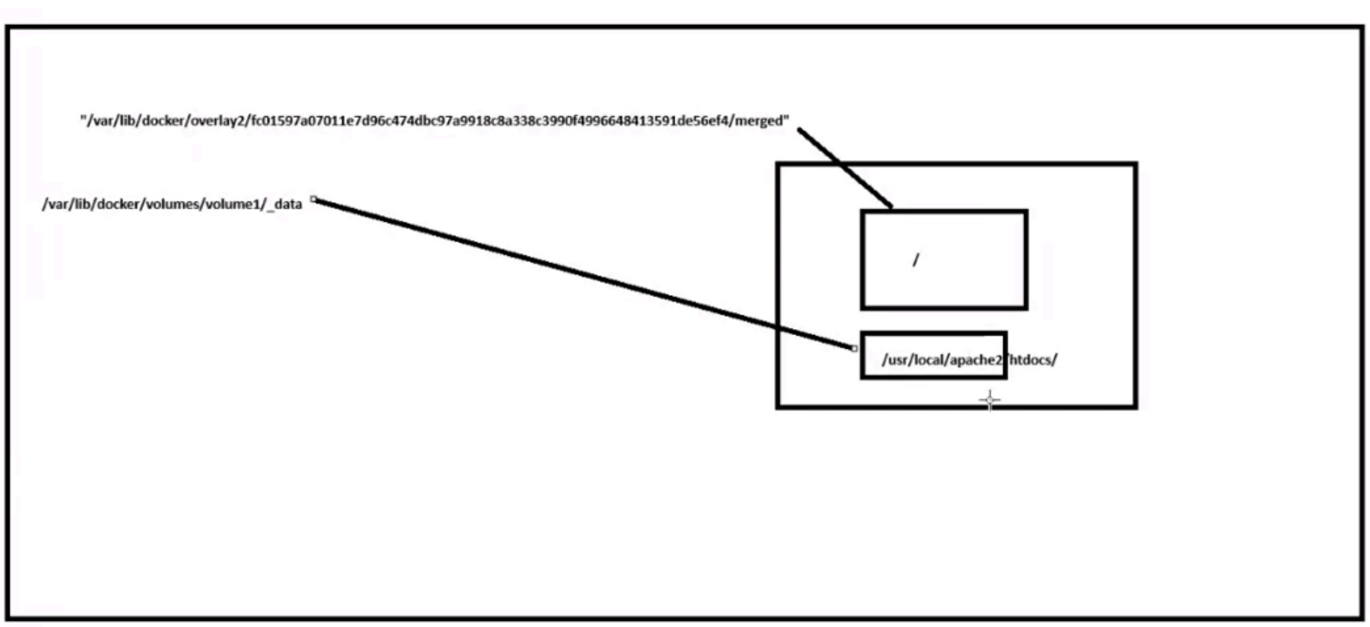
2. **How to persists the data in docker container?** By default one storage gets created for the container. The default storage is not persistent storage. So, additionally we attach volumes to manage the data and this additional disk or volume is persistent storageeven if you delete the container also still that persistent storage will be available

Persistant storage
Docker ps
Ls
Docker rm -f containerid
Docker ps
Docker run -dit -p 80:80 -v volume1:usr/local/apache2/htdocs/ httpd (-v is volume or additional disk to the container you can give any name to that harddisk)
Mount this harddisk to application folder. Here for httpd application folder is “usr/local/apache2/htdocs/” now the container will have 2 harddisk
After creating the container /var/lib/docker/volumes/volume1/_data (this folder gets created in the docker host)

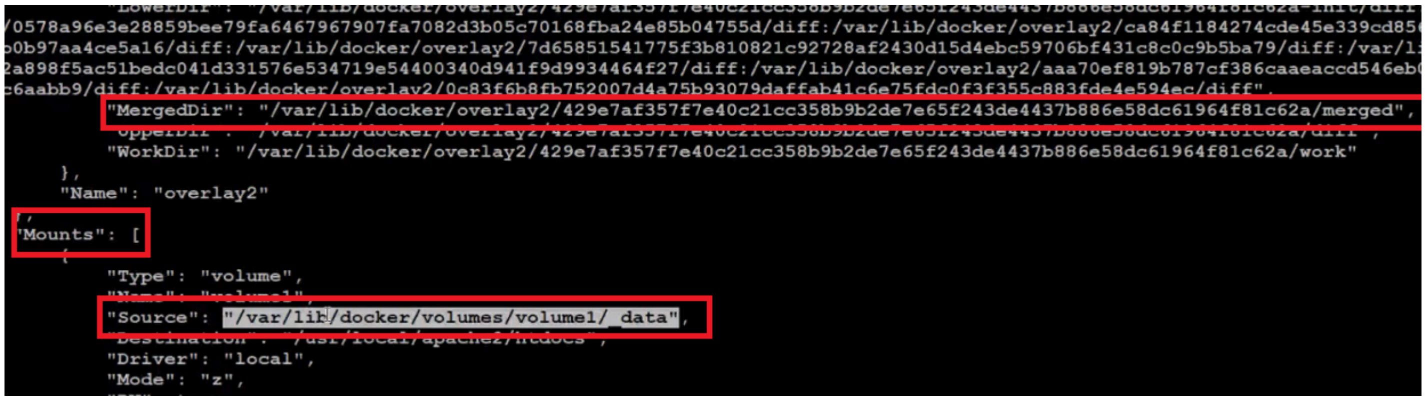
So now first folder /var/lib/docker/overlay2/ 

```
7b6c761c77d2dd83cdc1d3839c70ccf37119b01cb94fb46b12ae6a6dd291ddf1
```

/merged acts as “/” and second harddisk is mounted on ““usr/local/apache2/htdocs/”


"/var/lib/docker/overlay2/fc01597a07011e7d96c474dbc97a9918c8a338c3990f4996648413591de56ef4/merged"
/var/lib/docker/volumes/volume1/_data
/
/usr/local/apache2/htdocs/

Docker inspect containerid (check for merged folder(defaultstorage) and also under mounts there will be volume created as an additional harddisk(container data))



```
                "LowerDir": "/var/lib/docker/overlay2/429e7af357f7e40c21cc358b9b2de7e65f243de4437b886e58dc61964f81c62a-init/diff
/0578a96e3e28859bee79fa6467967907fa7082d3b05c70168fba24e85b04755d/diff:/var/lib/docker/overlay2/ca84f1184274cde45e339cd85
b0b97aa4ce5a16/diff:/var/lib/docker/overlay2/7d65851541775f3b810821c92728af2430d15d4ebc59706bf431c8c0c9b5ba79/diff:/var/l
2a898f5ac51bedc041d331576e534719e54400340d941f9d9934464f27/diff:/var/lib/docker/overlay2/aaa70ef819b787cf386caaeaccd546eb
c6aabb9/diff:/var/lib/docker/overlay2/0c83f6b8fb752007d4a75b93079daffab41c6e75fdc0f3f355c883fde4e594ec/diff",
                "MergedDir": "/var/lib/docker/overlay2/429e7af357f7e40c21cc358b9b2de7e65f243de4437b886e58dc61964f81c62a/merged",
                "UpperDir": "/var/lib/docker/overlay2/429e7af357f7e40c21cc358b9b2de7e65f243de4437b886e58dc61964f81c62a/diff",
                "WorkDir": "/var/lib/docker/overlay2/429e7af357f7e40c21cc358b9b2de7e65f243de4437b886e58dc61964f81c62a/work"
            },
            "Name": "overlay2"
        },
        "Mounts": [
            {
                "Type": "volume",
                "Name": "volume1",
                "Source": "/var/lib/docker/volumes/volume1/_data",
                "Destination": "/usr/local/apache2/htdocs",
                "Driver": "local",
                "Mode": "z",
```

Cd “/var/lib/docker/volumes/volumes1/_data”
Ls
Vi index.html
Edit something in the .html file
Check in the browser (now the file will be permanent storage until we delete it manually it will not be deleted even if the container is deleted try this)
Docker ps
Docker rm -f containerid (even though u delete container the data will still exists under volumes folder)
Docker ps
Cd “/var/lib/docker/volumes/volumes1/_data”
Ls
Cat index.html

10-02-2025

1. **Create a private network and Make the private network with 10.0.0.0/16 series not the default one**

To create a custom Docker network with a specific IP range like `10.0.0.0/16` (a private IP range), you can specify the subnet when creating the network. Here's how to do it:

Step 1: Create the Custom Private Network with a Specific Subnet

You can specify the subnet `10.0.0.0/16` when creating the network:

```
docker network create --driver bridge --subnet 10.0.0.0/16 my_private_network
```

- `--subnet 10.0.0.0/16` tells Docker to use the subnet `10.0.0.0/16`.
- `my_private_network` is the name of the network (you can replace it with your preferred name).

Step 2: Run Containers on the Custom Network

Once the network is created, you can run your containers on it using the `--network` flag:

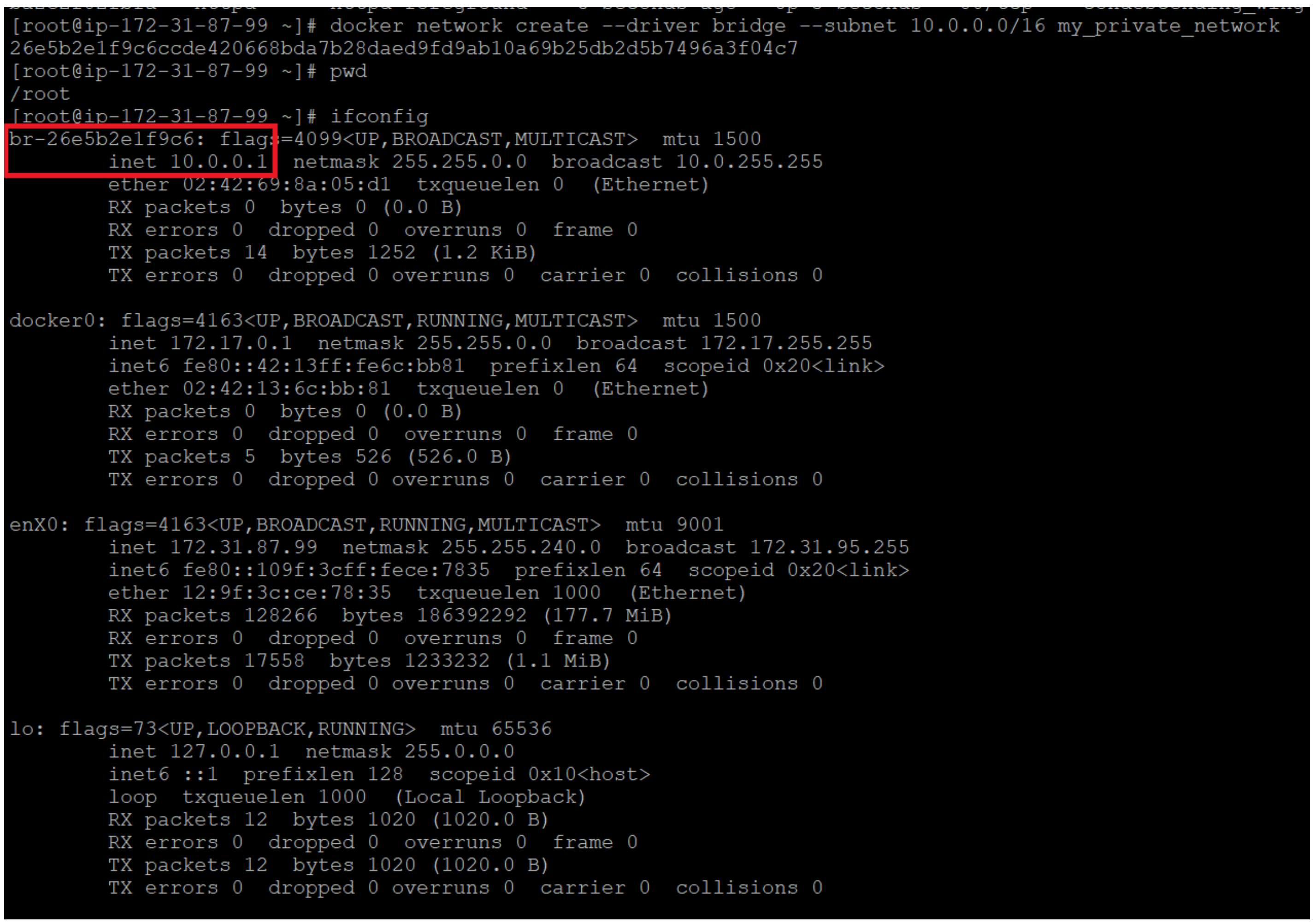
```
docker run -d --name my_container --network my_private_network my_image
```

This command will run `my_container` using the custom `my_private_network`, which has the subnet `10.0.0.0/16`.

Step 3: Inspect the Network (Optional)

If you want to inspect the details of your custom network and check if the subnet is correctly set, you can run:

```
docker network inspect my_private_network
```



```
[root@ip-172-31-87-99 ~]# docker network create --driver bridge --subnet 10.0.0.0/16 my_private_network
26e5b2e1f9c6ccde420668bda7b28daed9fd9ab10a69b25db2d5b7496a3f04c7
[root@ip-172-31-87-99 ~]# pwd
/root
[root@ip-172-31-87-99 ~]# ifconfig
br-26e5b2e1f9c6: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.255.0.0  broadcast 10.0.255.255
        ether 02:42:69:8a:05:d1  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14  bytes 1252 (1.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:13ff:fe6c:bb81  prefixlen 64  scopeid 0x20<link>
        ether 02:42:13:6c:bb:81  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5  bytes 526 (526.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enX0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 172.31.87.99  netmask 255.255.240.0  broadcast 172.31.95.255
        inet6 fe80::109f:3cff:fece:7835  prefixlen 64  scopeid 0x20<link>
        ether 12:9f:3c:ce:78:35  txqueuelen 1000  (Ethernet)
        RX packets 128266  bytes 186392292 (177.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17558  bytes 1233232 (1.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 12  bytes 1020 (1020.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 1020 (1020.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. How to make a private network default as doker0

1. Check Existing Docker Network

By default, Docker uses the `docker0` network bridge for containers, unless a custom network is specified during container creation. First, check your current Docker network settings by running:

```
docker network ls
```

You should see something like this:

```
NETWORK ID     NAME      DRIVER    SCOPE
123456abcdef   bridge    bridge    local
```

The `bridge` network corresponds to `docker0`.

2. Configure Docker to Use `docker0` as the Default Network

Docker containers will use the `bridge` network by default unless you explicitly specify another network. However, you can ensure this behavior by adjusting the Docker configuration file.

Edit the Docker configuration file:
The configuration file is typically located at `/etc/docker/daemon.json`. If it doesn't exist, create it.
Open the file using a text editor (e.g., `nano`):
bash
Copy
```
sudo nano /etc/docker/daemon.json
Rm -rf /etc/docker/daemon.json (to troubleshoot)
Systemctl start docker (troubleshoot)
Systemctl statusdocker
Docker network psdocker network ls
```
https://www.ispsystem.com/docs/dcimanager-kb/how-to-guides/how-to-change-the-network-used-by-docker

```
{
     "live-restore": true,
     "bip": "10.10.0.1/16",
     "default-address-pools": [{
          "base": "10.0.0.0/8",
          "size": 16
     }]
}
```

Add or Modify the `default-address-pools` option:
If you want to ensure that Docker uses the `docker0` network by default, you can explicitly define a default address pool in the configuration.
Example of the contents of `daemon.json`:
json
Copy
```
{
  "default-address-pools": [
    {
      "base": "172.17.0.0/16",
      "size": 24
    }
  ]
}
```

1. In this example, the base address of `docker0` is `172.17.0.0/16`, and the subnet size for each container is set to `/24`.

Restart Docker: After saving the changes to `daemon.json`, restart the Docker service for the changes to take effect.
```
sudo systemctl restart docker
ifconfig
```

3. Verify the Configuration

To verify that the `docker0` network is the default, run:

```
docker network inspect bridge
```

This should show details about the `docker0` bridge network.

You can also run a container without specifying a network and see if it uses `docker0`:

```
docker run -d --name test-container busybox
```

Then, check which network the container is attached to:

```
docker inspect test-container
```

Look for the `NetworkMode` or `Networks` section to confirm that the container is using the `docker0` network.

---

4. Optional: Configure a Custom Default Network

If you want to create and use a different default network, you can create a custom network with a specific subnet and set it as the default network for Docker.

Example:

```
docker network create --driver=bridge --subnet=192.168.1.0/24 my_custom_network
```

Then, use this network as the default by adjusting the `daemon.json` file accordingly.