DOCKER NETWORKING

1. Creating Containers (Login & Logout) Using Bridge Network

Docker's default bridge network enables containers to communicate with each other within a host. I created two containers (login, logout) using the official nginx:latest image.

Screenshot

```
malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examples
(main)
$ docker run -d --name login nginx:latest
4d22c402a9a9ca9708bed27d4e5e2be94abd862649c88be2878dde7b8b0a57e5
```

```
malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examples (main)
$ docker run -d --name logout nginx:latest
392a7e2b8771b0c68e2640c0abc7c533b52793cffb37e09ba2400c1f02794723
```

2. Testing Connectivity Between Containers

After creating the containers, I used the docker exec command to open a shell in one container and ping the IP address of the other. The successful ping results confirmed that the containers were able to communicate within the bridge network.

Screenshot

```
        }
    }
]

malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examples (main)
$ ^C

malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examples (main)
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
392a7e2b8771   nginx:latest   "/docker-entrypoint.…"   4 minutes ago    Up 4 minutes    80/tcp    logout
4d22c402a9a9   nginx:latest   "/docker-entrypoint.…"   14 minutes ago   Up 14 minutes   80/tcp    login

malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examples (main)
$ docker exec -it 4d22c402a9a9 bash
root@4d22c402a9a9:/# ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3) 56(84) bytes of data.
64 bytes from 172.17.0.3: icmp_seq=1 ttl=64 time=0.751 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.108 ms
64 bytes from 172.17.0.3: icmp_seq=3 ttl=64 time=0.081 ms
64 bytes from 172.17.0.3: icmp_seq=4 ttl=64 time=0.062 ms
64 bytes from 172.17.0.3: icmp_seq=5 ttl=64 time=0.126 ms
64 bytes from 172.17.0.3: icmp_seq=6 ttl=64 time=0.160 ms
64 bytes from 172.17.0.3: icmp_seq=7 ttl=64 time=0.084 ms
64 bytes from 172.17.0.3: icmp_seq=8 ttl=64 time=0.045 ms
64 bytes from 172.17.0.3: icmp_seq=9 ttl=64 time=0.069 ms
64 bytes from 172.17.0.3: icmp_seq=10 ttl=64 time=0.100 ms
64 bytes from 172.17.0.3: icmp_seq=11 ttl=64 time=0.147 ms
```

Shows the result of a ping between containers.

3. Creating and Using a Custom Bridge Network

To enhance network isolation and management, I created a custom bridge network named secure-network. Then, I launched the finance container (nginx) attached to this network. This demonstrated the process of building more secure, segmented Docker networks for different applications.

Screenshot

```
malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examples (main)
$ docker network create secure-network
d6be2ebab09845a7ef0b14128555ba7dbd23dd974048f609a3768379828f112a
```

```
malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examples (main)
$ docker run -d --name finance --network=secure-network nginx:latest
fc170587d4af9f49e4ae0b6ada4eab41ddc9abfcee254f81651eaea240279042
```

```
malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examples (main)
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS     NAMES
fc170587d4af   nginx:latest   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    finance
392a7e2b8771   nginx:latest   "/docker-entrypoint.…"   14 minutes ago       Up 14 minutes       80/tcp    logout
4d22c402a9a9   nginx:latest   "/docker-entrypoint.…"   24 minutes ago       Up 24 minutes       80/tcp    login
```

```
            "Networks": {
                "secure-network": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "MacAddress": "ee:c6:5f:46:87:4d",
                    "DriverOpts": null,
                    "GwPriority": 0,
                    "NetworkID": "d6be2ebab09845a7ef0b14128555ba7dbd23dd974048f609a3768379828f112a",
                    "EndpointID": "afdf104a12acb501fb1ad66ff576d07e9a62309f6f20f318ba90567ddef3a080",
                    "Gateway": "172.18.0.1",
                    "IPAddress": "172.18.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "DNSNames": [
                        "finance",
                        "fc170587d4af"
                    ]
                }
            }
        }
    }
```

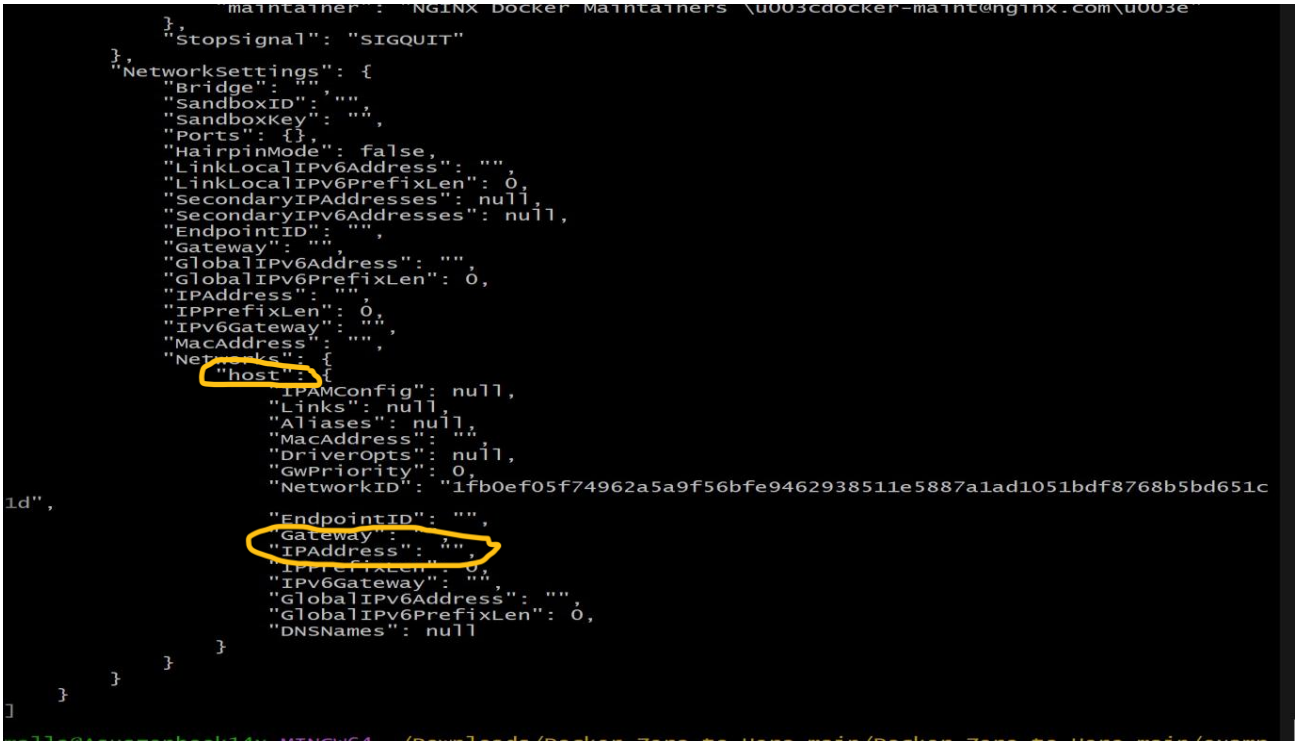
```
malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examples (main)
$ docker exec -it 4d22c402a9a9 bash
root@4d22c402a9a9:/# ping 172.18.0.2
PING 172.18.0.2 (172.18.0.2) 56(84) bytes of data.
```

Displays the network configuration details (docker inspect finance), confirming the container is mapped to the correct bridge and has a unique IP.

4. Host Networking Mode

I also experimented with Docker's host network mode, which allows containers to use the host's networking stack directly (removing network isolation). A container (host-demo) was run with the --network=host option.

```
malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examp
les (main)
$ docker run -d --name host-demo --network=host nginx latest
4be058675b810e9381ed6330b1de8001398355cc1bc4d7fe14e8f60527149134
```



Displays creation and status of the container using host networking.

5. Listing all Containers and Networks

Finally, I reviewed the running containers and their network details with docker ps and docker inspect.

Screenshot

```
malla@Asuszenbook14x MINGW64 ~/Downloads/Docker-Zero-to-Hero-main/Docker-Zero-to-Hero-main/examples (main)
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS     NAMES
fc170587d4af   nginx:latest   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    finance
392a7e2b8771   nginx:latest   "/docker-entrypoint.…"   14 minutes ago       Up 14 minutes       80/tcp    logout
4d22c402a9a9   nginx:latest   "/docker-entrypoint.…"   24 minutes ago       Up 24 minutes       80/tcp    login
```

Displays the final state of Docker containers and their network attachments.

Key Learnings

- **Understood the difference between bridge, custom bridge, and host networking in Docker.**
- **Practically validated container connectivity using ping and shell access.**
- **Explored network isolation and sub-network creation for enhanced security and management.**
- **Learned how to inspect networks and containers for detailed configuration review.**

Conclusion

This hands-on lab reinforced my Docker networking knowledge and practical skills important for real-world DevOps and cloud infrastructure roles.