**Docker Networks: Commands to Create, Run, and Visualize Containers**

**1. Bridge Network (Default)**

📌 **Use Case:** Enables communication between containers on the same host using container names.

**Create a Custom Bridge Network**

docker network create my\_bridge

**Run Containers in the Bridge Network**

docker run -d --name mysql\_container --network my\_bridge mysql:5.7

docker run -d --name wordpress\_container --network my\_bridge wordpress

**Check the Network**

docker network ls # List all networks

docker network inspect my\_bridge # Inspect the bridge network

**Visualize (List Running Containers in the Network)**

docker network inspect my\_bridge | grep Name

* The output should show both mysql\_container and wordpress\_container.

**2. Host Network (No Isolation)**

📌 **Use Case:** Allows containers to use the **host’s network** directly.

**Run a Container in Host Network**

docker run -d --name nginx\_host --network host nginx

**Check if Nginx is Running on the Host**

curl http://localhost # Should return Nginx's default page

**Check the Network**

docker network inspect host

* Unlike bridge mode, the container **does not get its own IP** but uses the **host’s IP**.

**3. None Network (No Networking)**

📌 **Use Case:** Completely **isolates** the container from any network.

**Run a Container in None Network**

docker run -d --name isolated\_container --network none alpine sleep 3600

**Check the Network**

docker network inspect none

**Try Pinging Another Container (Fails)**

docker exec -it isolated\_container ping google.com # Should fail

**4. Overlay Network (For Multi-Host Communication)**

📌 **Use Case:** Used in **Docker Swarm** to connect containers across multiple hosts.

**Enable Swarm Mode**

docker swarm init

**Create an Overlay Network**

docker network create -d overlay my\_overlay

**Run a Service in the Overlay Network**

docker service create --name web\_service --network my\_overlay nginx

**Check the Network**

docker network inspect my\_overlay

**List Services Running**

docker service ls

* This shows the web\_service running in the **overlay network**.

**Visualizing Docker Networks with a GUI**

If you want a **visual representation** of Docker networks:

1. Install docker extension

docker extension install docker/dashboard-extension

1. Open the Docker Dashboard:

docker dashboard

1. Click **Networks** to see **all running networks and their connected containers**.

**Summary of Commands**

| **Network Type** | **Command to Create Network** | **Run Container** | **Check Network** |
| --- | --- | --- | --- |
| **Bridge** | docker network create my\_bridge | docker run -d --name app --network my\_bridge nginx | docker network inspect my\_bridge |
| **Host** | (No need to create, exists by default) | docker run -d --name nginx\_host --network host nginx | docker network inspect host |
| **None** | (No need to create, exists by default) | docker run -d --name isolated --network none alpine sleep 3600 | docker network inspect none |
| **Overlay** | docker network create -d overlay my\_overlay | docker service create --name web --network my\_overlay nginx | docker network inspect my\_overlay |

Would you like a **real-world example** using a **Docker Compose file** for networks? 🚀