**TASK 16 - Docker Networking: Bridge Network, Host Network, and None Network**

Docker provides different types of networks to connect containers to each other and external systems. Understanding these networks, their purposes, definitions, and limitations is critical for containerized application design.

**Purpose**

1. **Bridge Network**: Provides network isolation between containers and enables communication on the same host.
2. **Host Network**: Allows containers to share the host machine's network stack for better performance or specific use cases.
3. **None Network**: Completely isolates containers from any network; no connectivity is provided.

**Definitions and Limitations**

1. **Bridge Network**:
   * **Definition**: A default network created by Docker that allows containers to communicate with each other through a virtual network bridge. Containers are assigned private IPs and can communicate using container names or IP addresses.
   * **Purpose**: To provide an isolated environment for containers to communicate internally while restricting direct access from external networks.
   * **Limitations**:
     + Communication is limited to containers on the same bridge unless explicitly routed.
     + Requires manual setup for external communication.
2. **Host Network**:
   * **Definition**: Removes network isolation between the container and the host. The container shares the same network namespace as the host machine.
   * **Purpose**: To achieve high performance or bypass network configuration challenges.
   * **Limitations**:
     + Containers lose isolation and share ports with the host.
     + Security risks increase as the container has direct access to the host’s network.
3. **None Network**:
   * **Definition**: Disables networking for the container, providing total network isolation. The container does not have any external or internal connectivity.
   * **Purpose**: To run tasks that don’t require networking, such as isolated data processing or application testing.
   * **Limitations**:
     + No access to external or internal networks.
     + Manual interaction is required for tasks involving network dependencies.

**Steps to Perform the Task**

1. **Preparation**:
   * Install Docker on your system if not already done.
   * Pull a simple web server image like nginx for testing:

“docker pull nginx”

1. **Task 1: Bridge Network**:
   * **Steps**:
     1. Create a custom bridge network:

“docker network create my\_bridge\_network”

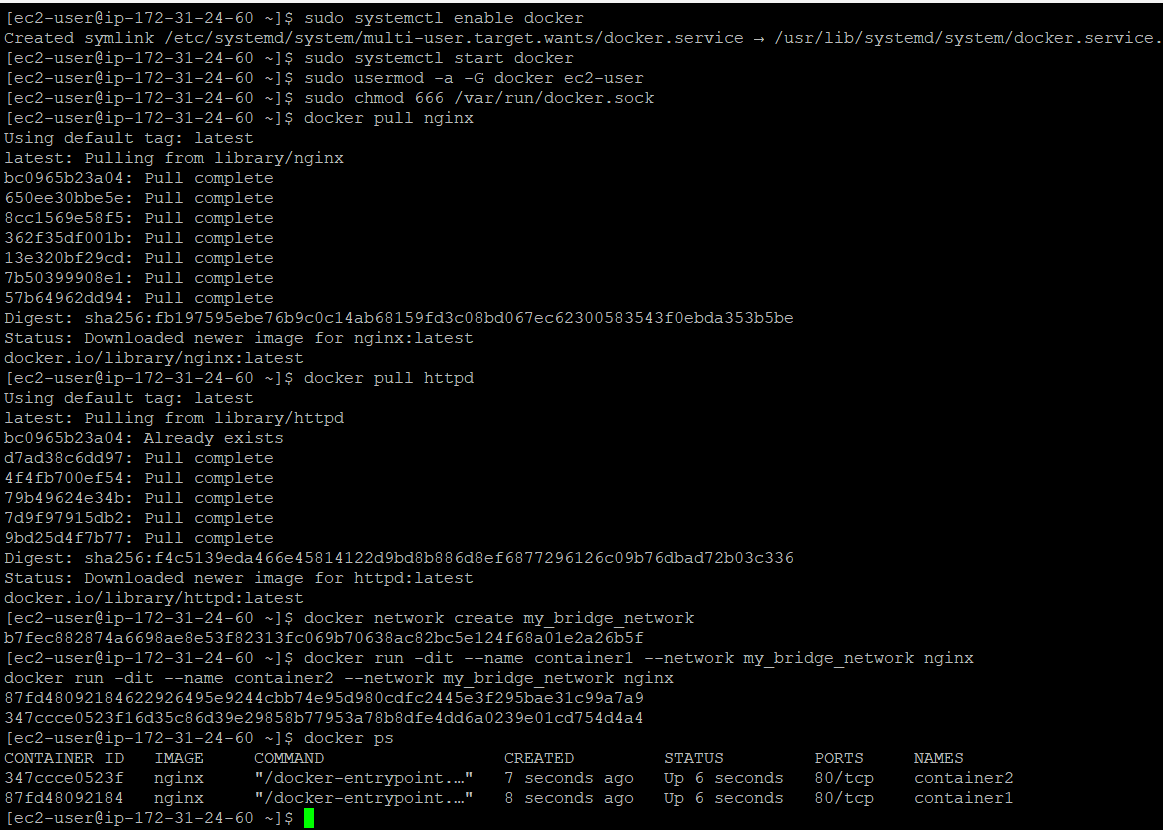
* + 1. Run two containers using the custom network:

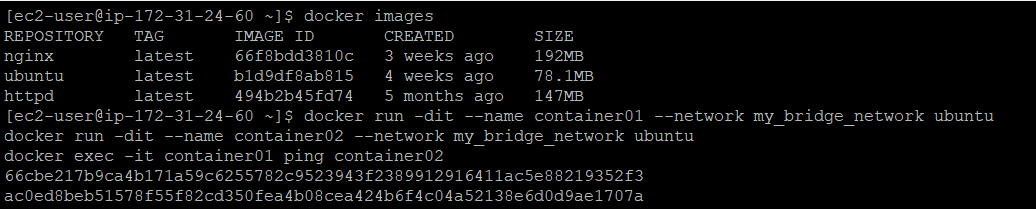
“docker run -dit --name container1 --network my\_bridge\_network nginx”

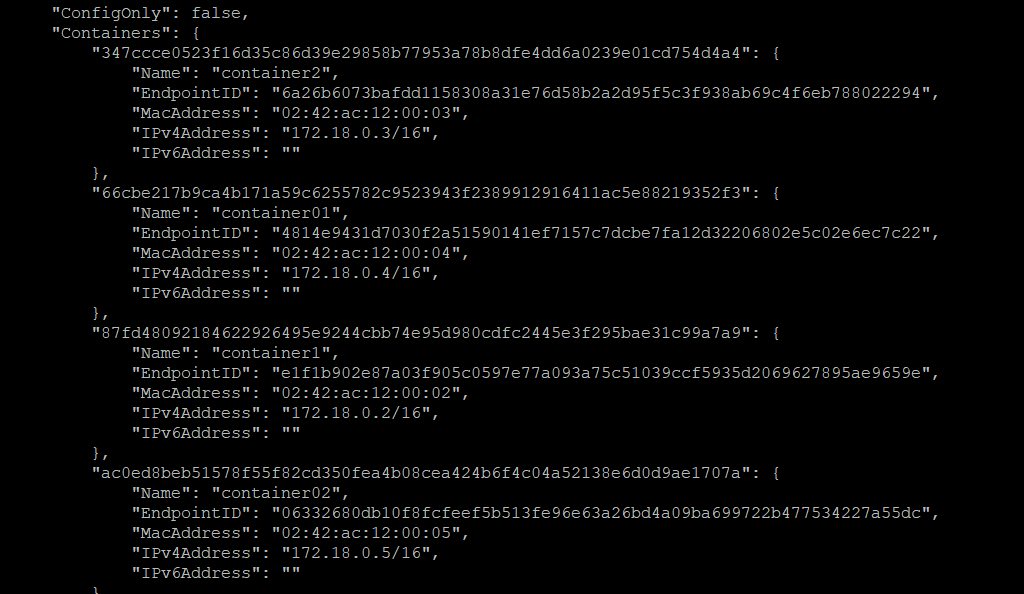
“docker run -dit --name container2 --network my\_bridge\_network nginx”

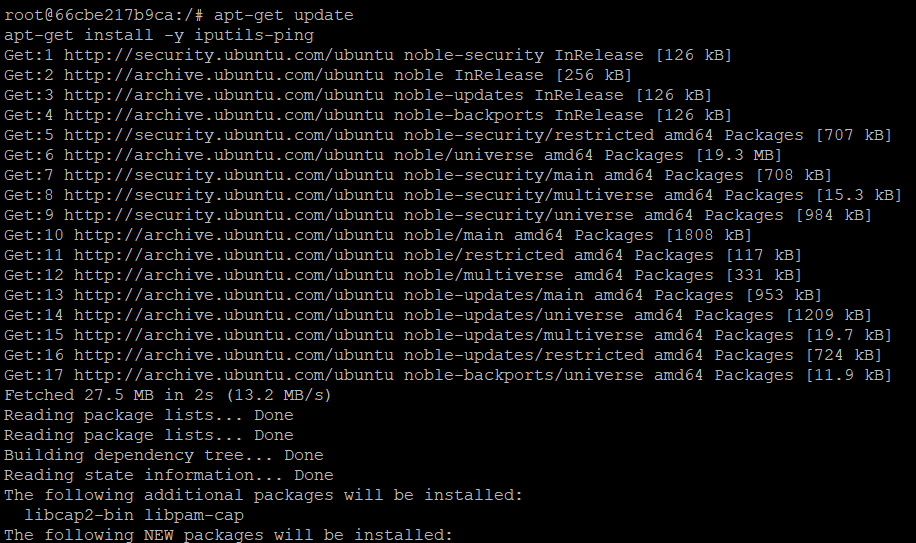
* + 1. Verify the containers' connectivity:

“docker exec -it container1 ping container2”

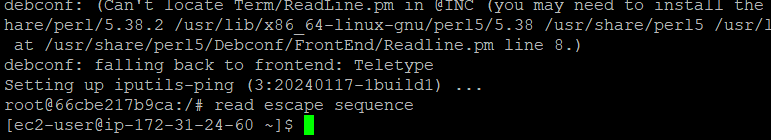


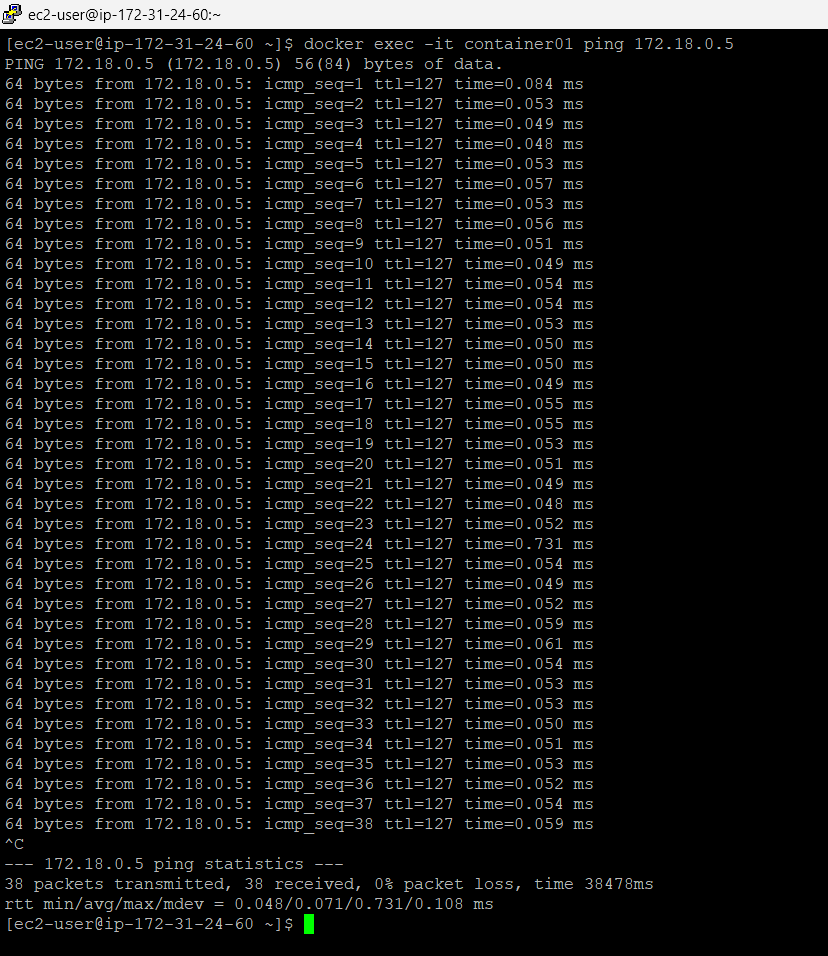






CTRL+P+Q





**Observation**: Containers on the same bridge network can communicate using container names or IP addresses

**Task 2: Host Network**:

* Run a container with the host network

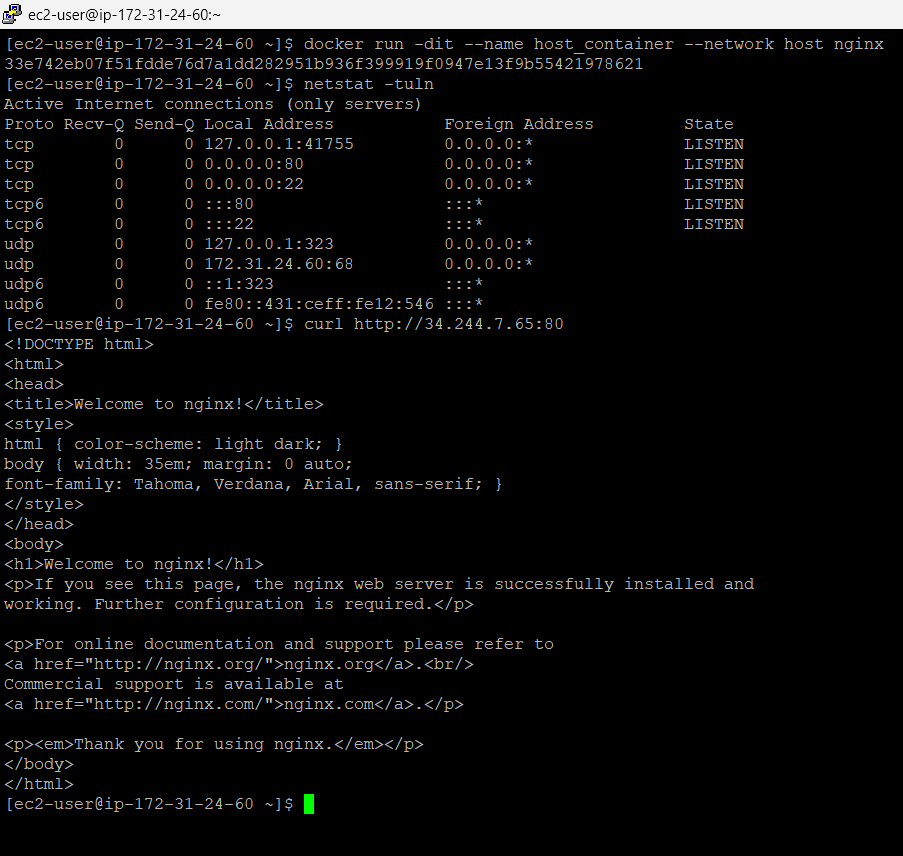
“docker run -dit --name host\_container --network host nginx”

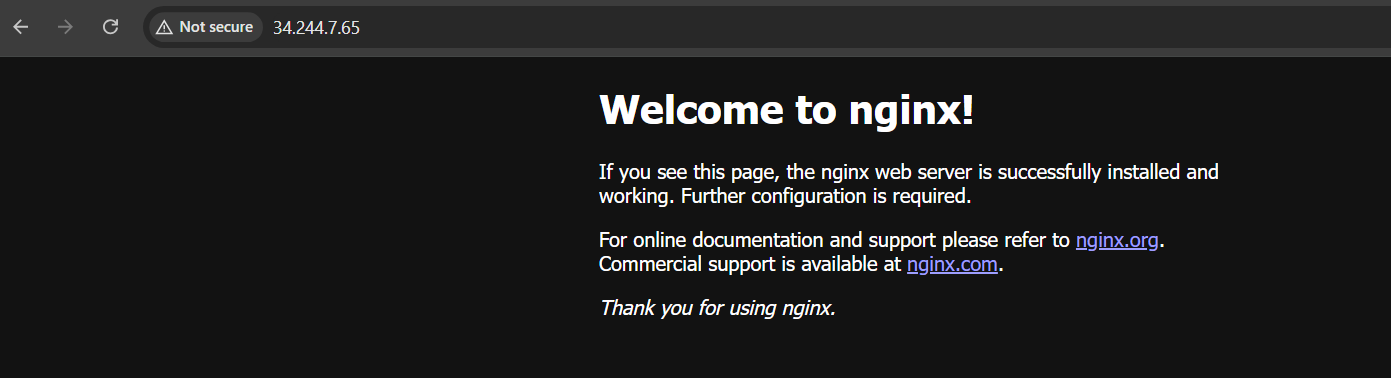
* Check open ports on the host:

“netstat -tuln”

* Access the web server using the host’s IP:

“curl http://<host-ip>:80”

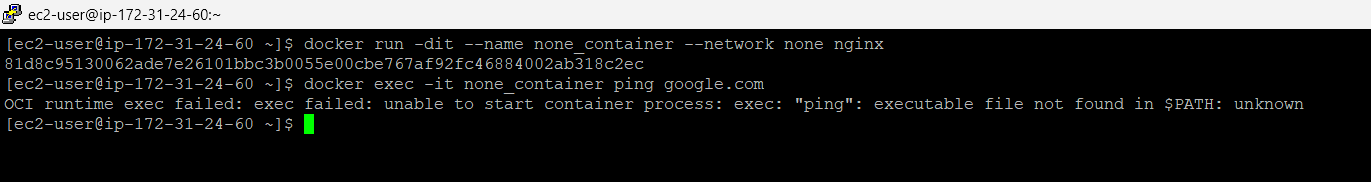


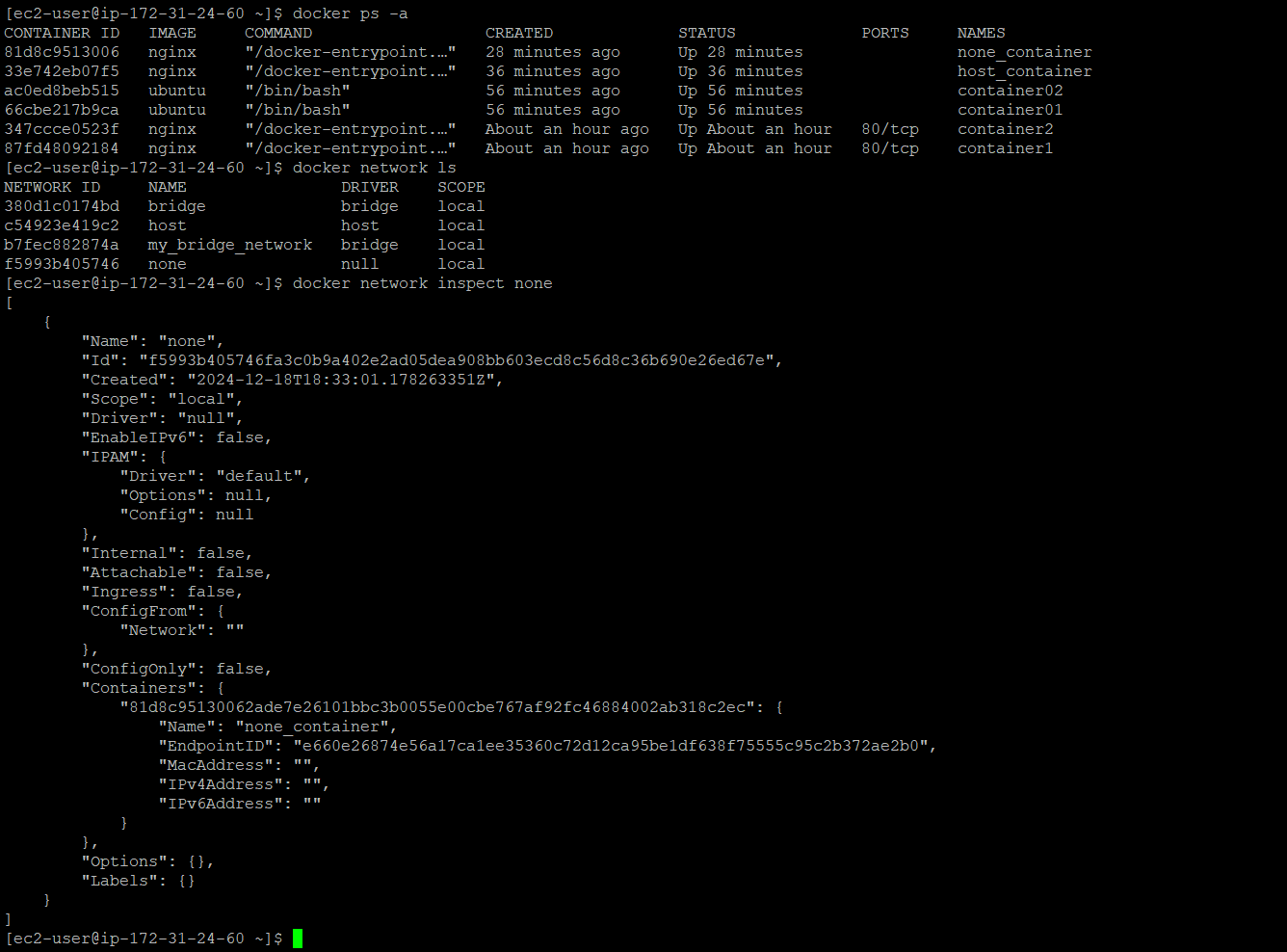


**Observation**: The container shares the host’s network stack, and the web server is accessible directly on the host's IP address.

**Task 3: None Network**:

Run a container with no network and try to access the container’s network



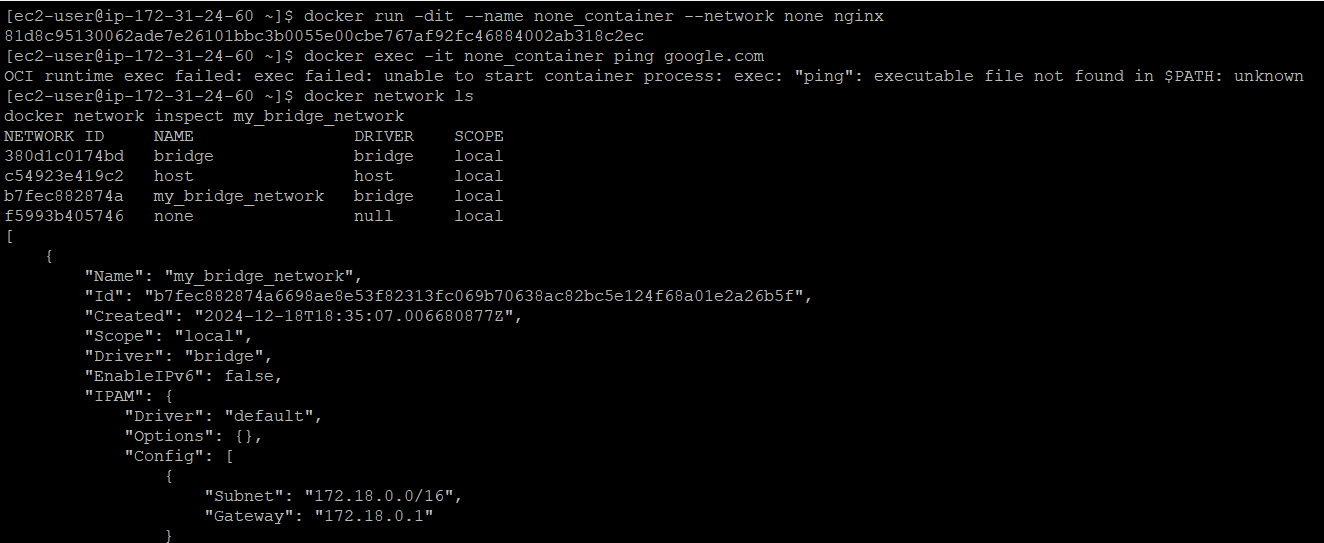


**Observe the isolation:**

* + The ping command will fail, showing no network access.

**Observation**: The container is fully isolated from any network connectivity.

List the networks and inspect their details

.

### **1. Bridge Network (Default Network)**

#### **Definition:**

- A private, isolated network for containers on the same Docker host. Containers can communicate with each other using container names or IP addresses.

#### **Impact on Containers:**

- Isolation: Containers in a bridge network cannot communicate with containers outside the network unless explicitly connected.

- Internal Connectivity: Containers on the same bridge network can communicate with each other using their container names.

- Access to Host/Internet: Containers can access the internet if the host has a default route and NAT is enabled.

- Port Binding: You must map ports (`-p` or `-P`) to access the container externally.

#### **Use Case:**

- Ideal for local development where you want containers to communicate only with each other and not the host system or external containers.

#### **Example:**

- Two containers in a custom bridge network (`my\_bridge\_network`) can ping each other using their container names or IPs:

docker network create my\_bridge\_network

docker run -dit --name container1 --network my\_bridge\_network ubuntu

docker run -dit --name container2 --network my\_bridge\_network ubuntu

docker exec -it container1 ping container2

### **2. Host Network**

**#### Definition:**

- The container shares the host's network namespace directly, meaning it uses the host's IP address and network stack.

#### **Impact on Containers:**

- No Isolation: Containers bypass Docker's virtual network layer and use the host's network interfaces.

- Performance: Provides better performance because packets bypass Docker’s network stack.

- Port Conflicts: Containers using the host network cannot bind to ports already in use on the host.

- Internet Access: Containers can access the internet and external networks without any NAT setup.

#### **Use Case:**

- Best suited for high-performance applications (e.g., real-time apps or network-heavy workloads) where the overhead of Docker’s networking is not acceptable.

#### **Example:**

- Run a container with host networking:

docker run --rm --network host nginx

- Access the container via the host's IP:

curl http://localhost:80

### **3. None Network**

#### **Definition:**

- The container has no network interface and cannot communicate with other containers, the host, or the internet.

#### **Impact on Containers:**

- Full Isolation: The container is entirely isolated from all networks.

-No Connectivity: Containers cannot connect to the host, other containers, or the internet.

- Use of Local Communication: Only accessible via the container's terminal or mounted volumes.

#### **Use Case:**

- Suitable for containers that don’t need network access, such as batch jobs, security tests, or isolated tasks.

#### **Example:**

- Run a container with no network:

docker run --rm --network none ubuntu

- Inside the container, no networking tools will function:

ping google.com

# Output: ping: unknown host

While **bridge**, **host**, and **none** networks are foundational for Docker networking and are sufficient for basic container communication, **overlay** and **macvlan** networks are crucial for scaling and integrating Docker into real-world, enterprise-level environments. They address specific needs like multi-host communication, performance optimization, and direct integration with physical networks, which are often encountered in advanced use cases.

Adding an understanding of these networks will help you:

1. Prepare for **real-world scenarios**, such as working with distributed applications or legacy systems.
2. Showcase a deeper grasp of Docker networking, which is valuable for interviews and professional growth.
3. Expand your toolkit for solving challenges that the basic network types might not handle efficiently.

### **4. Overlay Network**

#### **Definition:**

- A network that spans multiple Docker hosts, allowing containers running on different hosts to communicate securely.

#### **Impact on Containers:**

- Inter-host Communication: Containers can communicate across multiple hosts without additional configuration.

- Requires Docker Swarm or Orchestration: Needs a Docker Swarm cluster or similar orchestration tools.

- Isolated by Default: Only containers in the same overlay network can communicate.

#### **Use Case:**

- Best for distributed applications in multi-host Docker environments (e.g., microservices).

### **5. Macvlan Network**

#### **Definition:**

- Assigns a MAC address to each container, making them appear as physical devices on the network.

#### **Impact on Containers:**

- Direct Access to Host Network: Containers appear as separate devices on the physical network.

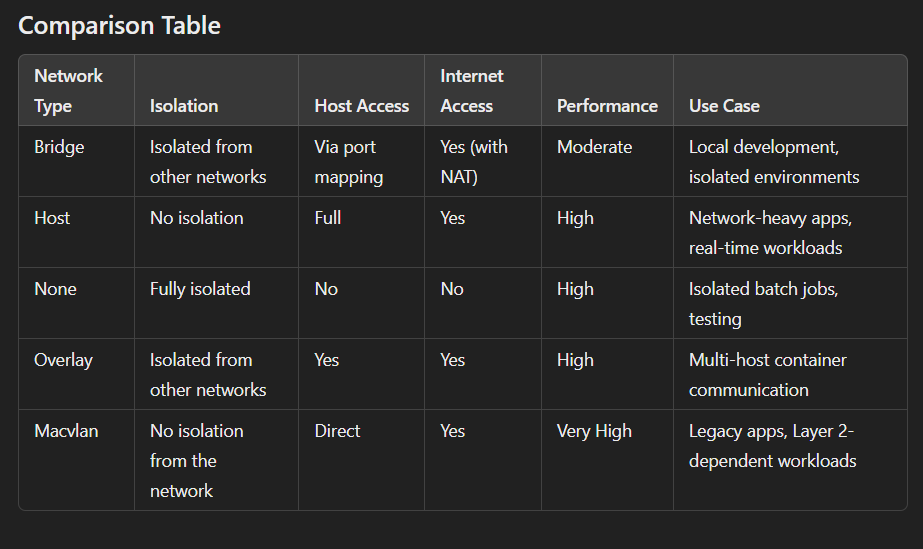
- No NAT Overhead: Offers high performance because there’s no NAT or port binding involved.

- No Internal Communication: Containers on a macvlan network cannot communicate with the host by default.

#### **Use Case:**

- Ideal for legacy applications that require direct Layer 2 access to the network.

### **Comparison Table**



**Steps to Demonstrate Bridge, Host, and None Networking**

1. **Bridge Network:**

docker network create my\_bridge\_network

docker run -dit --name container1 --network my\_bridge\_network ubuntu

docker run -dit --name container2 --network my\_bridge\_network ubuntu

docker exec -it container1 ping container2

2. **Host Network:**

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

curl http://localhost:80

3. **None Network:**

docker run -d --name container\_none --network none ubuntu

docker exec -it container\_none ping google.com

# Output: ping: unknown host