Docker Networking – A Complete Detailed Tutorial

1. **IP Address**
   a. **Unique routable address given to a device in a network which follows IP protocol**

2. Ethernet Card (Network Interface Card – NIC)

What is it?

An **Ethernet card** or **NIC** is the physical hardware that connects a computer to a network.

Functions of NIC

- Converts digital data into electrical signals (Layer 1 & 2)
- Adds MAC addresses to frames
- Handles CSMA/CD (Carrier Sense Multiple Access with Collision Detection)
- Responsible for frame transmission & reception

Types

- Wired NICs (RJ45 port)
- Wireless NICs (Wi-Fi)

Why is NIC important?

Everything on your system (containers, VMs, pods, browsers) ultimately sends data **through the NIC** to reach the outside world.

---

3. Ethernet Cable

What is it?

A physical cable used to connect computers, switches, routers.

Common types

- **Cat5e** – 1 Gbps
- **Cat6** – 10 Gbps up to 55 meters

- **Cat6a** – 10 Gbps up to 100 meters
- **Cat7/8** – Higher speeds (datacenters)

What it carries?

Ethernet cables carry **electrical signals** representing bits.

---

4. Network Switch

What is a switch?

A Layer-2 device that connects multiple devices within the same LAN.

What it does?

- Forwards Ethernet frames based on **MAC addresses**
- Maintains a **MAC Address Table**
  - Example: IP 10.0.0.5 has MAC aa:bb:cc:dd:ee:ff and lives on port 3
- Reduces collisions
- High-speed packet switching (hardware-based)

How a switch works?

1. Device A sends a frame to B
2. Switch reads the destination MAC
3. Looks up in MAC table
4. Forwards frame only to that port (not to all)

This is why switches are much faster than hubs.

---

5. ARP Protocol (Address Resolution Protocol)

What is ARP?

ARP maps **IP address → MAC address** in a local network.

Why is ARP used?

NICs only understand MAC addresses, but applications use IP addresses.
So we need a translation mechanism.

Process

If machine A wants to talk to machine B:

1. A broadcasts:
   “Who has 192.168.1.20? Tell 192.168.1.10”

2. B replies:
   “192.168.1.20 is at aa:bb:cc:dd:ee:11”

3. A stores this in **ARP cache** & sends packet.

ARP Table Example

IP Address	MAC Address
192.168.1.5	aa:bb:cc:dd:ee:ff
192.168.1.20	11:22:33:44:55:66

---

6. Gateway

What is a gateway?

A **gateway** is the router through which your system sends traffic destined **outside its subnet**.

Why do we need it?

Your system can only directly communicate with devices **inside the same subnet**.
If traffic is going outside the subnet, it is sent to the **default gateway**.

Example

Your laptop IP: `192.168.1.10/24`
Destination IP: `8.8.8.8`

Since 8.8.8.8 is outside your /24 network, your system forwards the packet to the gateway (say 192.168.1.1).

---

7. DNAT & SNAT (NATing)

NAT = Network Address Translation
Used to modify IPs/ports at the network boundary.

---

a) SNAT / MASQUERADE

Source NAT changes the **source IP** of outgoing packets.

Use Case

Private IP → Internet communication

Example:
Packet from 192.168.1.10 (private IP) goes out to the internet.
Router changes source IP → public IP (e.g., 14.139.22.1)

Why?

Private IPs are not routable on internet.

---

b) DNAT

Destination NAT changes the **destination IP** of incoming traffic.

Use Case

Port forwarding / load balancing

Example:
Packet coming to public IP 14.139.22.1:80
Router rewrites destination IP → 192.168.1.50:80 (web server)

---

8. Subnetting

What is subnetting?

The process of dividing a large network into smaller networks.

Why subnet?

- Improve security
- Reduce broadcast domains
- Efficient IP allocation
- Better network control

Subnetting Example

Network: `192.168.1.0/24`
Dividing it into 4 subnets (/26)

Subnet	Range	Host Count
Subnet 1	192.168.1.0–63	62 hosts
Subnet 2	192.168.1.64–127	62 hosts
Subnet 3	192.168.1.128–191	62 hosts
Subnet 4	192.168.1.192–255	62 hosts

CIDR notation

- /24 → 255.255.255.0 (256 IPs)
- /25 → 255.255.255.128 (128 IPs)
- /26 → 255.255.255.192 (64 IPs)

Docker networking allows containers to communicate **with each other**, **with the host**, and with **external networks**. Understanding how Docker networking works is crucial for microservices, distributed systems, and DevOps workflows.

Docker uses Linux networking primitives like:

- Network namespaces
- Virtual Ethernet pairs (veth)
- Linux bridges
- iptables / NAT
- Overlay networks

Let's break everything down step-by-step.

---

1. What is a Network Namespace?

Every Docker container runs inside a **network namespace**, which provides:

- Its own network interface
- Its own routing table
- Its own firewall rules
- Isolation from other containers

Diagram: Container + Namespace

```
+-------------------------+
| Docker Container        |
|-------------------------|
| Network Namespace       |
|   - eth0 (private veth) |
|   - routing table       |
+-------------------------+
```

Namespaces ensure containers believe they have their own network stack.

---

2. veth Pairs (virtual ethernet cables)

Docker connects containers to networks using **veth pairs**.

- A veth pair is like a **virtual cable** with two ends.

- One end goes inside the container (`eth0`)
- The other end goes to Docker's bridge (`vethXYZ`)

Diagram: veth pair

```
[Container eth0] <----> [veth0-br] ---- (bridge)
```

---

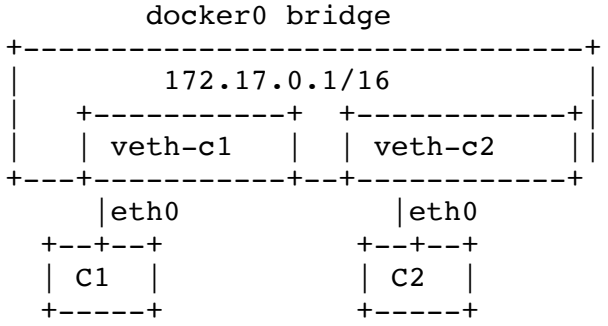
3. Docker Bridge Network (`bridge`)

When you install Docker, it creates a default network:

```
docker0  (a Linux bridge)
```

This is like a virtual switch.

Diagram: Bridge Network



Features:

- Containers can talk to each other
- Containers get internal IPs
- NAT (iptables) allows containers to reach the outside internet

---

4. Types of Docker Networks

Docker has **5** main network types:

Network Type	Key Use Case
bridge	default; containers communicate internally
host	container shares host's network stack
none	no networking; full isolation
overlay	multi-host communication (Swarm, Kubernetes)
macvlan	containers get their own MAC + real LAN presence

Let’s explain each.

---

5. Bridge Network (default)

Used for most local development/docker-compose setups.

Create your own bridge network:

```
docker network create mynet
```

Run containers inside it:

```
docker run -d --name app1 --network mynet nginx
docker run -d --name app2 --network mynet busybox
```

Containers can reach each other by name:

```
ping app1
```

---

6. Host Network

Container shares **host network directly**.

```
docker run --network host nginx
```

- No isolation
- Best performance (zero NAT)
- Used for performance-driven services (Prometheus, CNI plugins)

---

7. None Network

A container with no network:

```
docker run --network none alpine
```

Totally isolated.

Used for:

- Secure batch jobs
- Specialized use-cases

---

8. Overlay Network (multi-host networking)

Used when containers on **different hosts** must communicate.

```
docker network create -d overlay myoverlay
```

Works with:

- Docker Swarm
- Kubernetes (CNI handles it differently)

Diagram:

```
Host1                            Host2
+-----------+                  +-----------+
| Container | <-- Overlay --> | Container |
+-----------+                  +-----------+
```

Overlay networks use:

- VXLAN encapsulation
- Distributed key-value store for state

---

9. Macvlan Network

Gives each container a unique MAC address. Appears as real machines on LAN.

`docker network create -d macvlan ...`

Used when:

- You want containers visible to network like real servers
- Need DHCP or broadcast

---

10. How Does a Container Access the Internet? (NAT)

Docker configures iptables:

```
iptables -t nat -A POSTROUTING -s 172.17.0.0/16 ! -o docker0 -j MASQUERADE
```

Meaning:

- Container outbound traffic goes to docker0
- Docker NATs it to host IP
- Internet responds to host, traffic routed back into container

Diagram:

```
[Container] -> docker0 -> host eth0 -> Internet
                          NAT
```

---

11. Port Mapping (`-p`)

```
docker run -p 8080:80 nginx
```

Meaning:

- Host port 8080 → Container port 80

Diagram:

```
Internet --> host:8080 --> container:80
```

Docker uses:

- iptables DNAT rules
- virtual bridge forwarding

---

12. Inspecting Docker Networks

Check networks:

```
docker network ls
```

Inspect:

```
docker network inspect bridge
```

See container network details:

```
docker inspect container_name
```

---

13. Custom Bridge Network vs Default Bridge

Default bridge limitations:

- No DNS-based service discovery
- You must link containers or use IPs

Custom bridge advantages:

- Built-in DNS service discovery
- Better isolation
- Cleaner configuration

---

14. Practical Example: App + DB Network

Step 1: Create network

```
docker network create backend
```

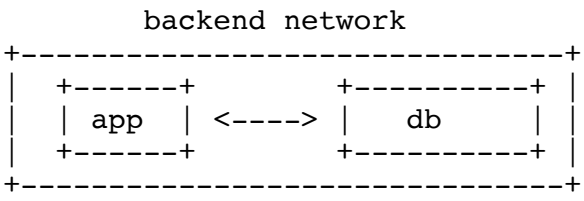
Step 2: Run DB

```
docker run -d --name db --network backend mysql
```

Step 3: App connects using service name:

```
db:3306
```

Diagram:



---

15. Complete Architecture Diagram

Here is a full conceptual diagram combining all aspects: