Docker Networking

npNOG 10

November 25 - 28, 2024

Docker Networking

Docker networking is an essential aspect of using Docker containers, as it allows containers to communicate with each other, with the host system, and with external networks.

Docker provides several networking options to facilitate these communications.

Docker Networking Drivers

Docker networking drivers are the underlying technologies that enable communication between containers and the host system. Docker includes several network drivers, each serving different purposes:

- Bridge Network (default)
- Host Network
- Overlay Network
- Macvlan Network
- Ipvlan Network
- None Network

Bridge Network (default):

- The default network driver for containers.
- Creates an isolated network on a single Docker host.
- Containers on the same bridge network can communicate with each other.
- Example command:

```
docker network create --driver bridge
my_bridge_network
```

Host Network:

- Shares the host’s networking namespace.
- Containers use the host's IP address.
- Useful for performance optimization or running services that need direct network access.
- Example command:

```
docker run --network host my_container
```

Overlay Network:

- Used for multi-host communication, useful in Docker Swarm or Kubernetes.
- Enables containers running on different Docker hosts to communicate.
- Example command:

```
docker network create --driver overlay
my_overlay_network
```

Macvlan Network:

- Assigns a MAC address to each container to make them look like physical devices.
- Useful for legacy applications that require direct network access.
- Example command:

```
docker network create --driver macvlan --
subnet=192.168.1.0/24 --gateway=192.168.1.1 -o
parent=eth0 my_macvlan_network
```

Ipvlan Network:

- IPVLAN is similar to MACVLAN, but instead of assigning each container a unique MAC address, it assigns a unique IP address while sharing the MAC address of the parent interface.
- Useful for performance optimization or running services that need direct network access.
- Example command:

```
docker network create --driver ipvlan --
subnet=192.168.1.0/24 --gateway=192.168.1.1 -o
parent=eth0 my_ipvlan_network
```

None Network:

- No network is attached to the container.
- Useful for highly isolated containers where networking is not required.
- Example command:

```
docker run --network none my_container
```

Networking Commands

- **List Networks:**
  - `docker network ls`
  - Lists all networks available on the Docker host.
- **Inspect Network:**
  - `docker network inspect [network_name]`
  - Provides detailed information about a specific network.
- **Connect Container to Network:**
  - `docker network connect [network_name] [container_name]`
  - Connects an existing container to a specific network.
- **Disconnect Container from Network:**
  - `docker network disconnect [network_name] [container_name]`
  - Disconnects a container from a specific network.

Networking Use Cases

- **Single-host Networking:**
  - For applications running on a single Docker host, the bridge network is usually sufficient.
  - Example: Running a web server and database on the same host.
- **Multi-host Networking:**
  - For applications distributed across multiple Docker hosts, the overlay network is suitable.
  - Example: A microservices architecture deployed using Docker Swarm or Kubernetes.
- **High-performance Networking:**
  - For performance-sensitive applications, the host network can reduce network latency.
  - Example: Network monitoring tools or real-time applications.

Custom Docker Networks

- **Creating a Custom Network:**
  - Example:

```
docker network create --driver bridge
my_custom_network
```

- **Running a Container on a Custom Network:**
  - Example:

```
docker run --network my_custom_network my_container
```

- **DNS Resolution:**
  - Docker provides internal DNS resolution, so containers can resolve each other by name when using user-defined networks.
  - Example: Container `web` can resolve the name `db` if both are on the same user-defined network.

Practical Examples

```
# Create a custom bridge network
docker network create --driver bridge my_bridge_network

# Run a container on the custom bridge network
docker run -d --name my_container --network my_bridge_network nginx

# List all networks
docker network ls

# Inspect the custom bridge network
docker network inspect my_bridge_network

# Connect an existing container to another network
docker network connect my_bridge_network my_container

# Disconnect a container from a network
docker network disconnect my_bridge_network my_container
```