13. Docker Networking

Friday, May 19, 2023 5:23 PM

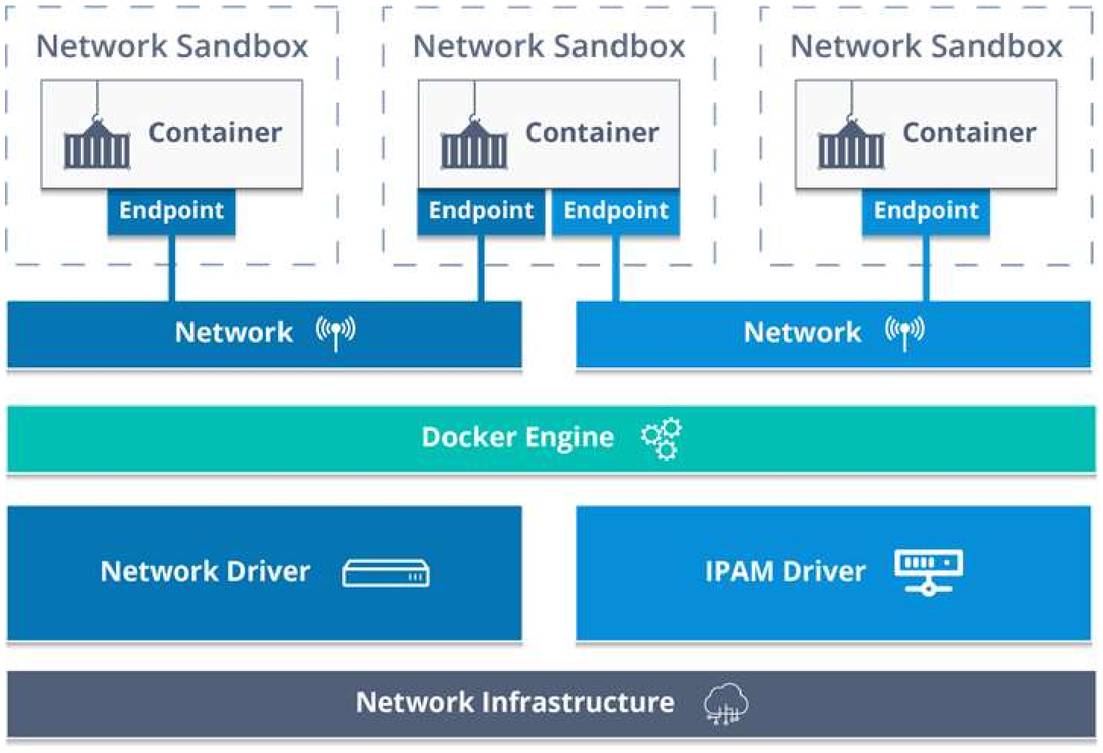
Docker Networking refers to the networking capabilities and features provided by Docker, an open-source containerization platform. Docker Networking allows containers to communicate with each other and with the outside world, enabling seamless connectivity and network integration for applications running in Docker containers.

Here are some key concepts and features related to Docker Networking:

- ✓ **Containers**:
  - Docker allows you to create and run containers, which are lightweight, isolated environments that package an application and its dependencies.
  - Each container can have its own network stack, including network interfaces, IP addresses, and routing tables.
- ✓ **Docker Network**:
  - A Docker network is a virtual network that provides communication between containers.
  - Docker offers various network drivers, such as bridge, overlay, host, and macvlan, each with its own characteristics and use cases.
  - By default, Docker creates a bridge network called "bridge" that allows containers to communicate with each other.
- ✓ **Bridge Network**:
  - The bridge network driver connects containers to a bridge on the host.
  - Containers attached to the same bridge can communicate with each other using IP addresses.
  - This is the default network driver for Docker containers and is suitable for most use cases.
- ✓ **Overlay Network**:
  - The overlay network driver allows containers to communicate across multiple Docker hosts.
  - It helps in the creation of distributed applications spanning multiple machines or even different data centers.
  - It uses a network overlay technology, such as VXLAN, to encapsulate and transmit network traffic between hosts.
- ✓ **Container-to-Container Communication**:
  - Containers within the same Docker network can communicate with each other using their IP addresses or container names.
  - Docker assigns each container a unique IP address within the network, allowing them to discover and communicate with other containers by their IP or hostname.
- ✓ **Exposing Ports**:
  - Docker allows you to expose ports on containers to the host system or external networks.
  - This enables access to containerized applications or services from the outside world. You can specify port mappings when running a container to bind a container port to a host port.
- ✓ **DNS Resolution**:
  - Docker provides an embedded DNS server that allows containers to resolve each other's IP addresses using container names.
  - This enables seamless service discovery and communication between containers without the need to know the IP addresses explicitly.

✓ **Docker Compose**:

- Docker Compose is a tool that simplifies the orchestration of multi-container applications.
- It provides a way to define and manage multiple containers as a single service using a YAML file.
- Docker Compose allows you to specify networking configurations, including network creation, container-to-container communication, and port mappings.



Architecture of Container Networking Model

✓ **Network sandbox**

- It is isolated sandbox that holds the network configurations of containers
- Sandbox is created when a user requests to generate an endpoint on the network.

✓ **Endpoints**

- It can have several endpoints in a network.
- EP established connectivity for the container services with other services.

✓ **Network**

- Provides networking components

✓ **Docker Engine**

- Is the base engine installed on host machine to build & run containers using docker.

✓ **Network driver**

- Its task is to manage the network with multiple drivers.

✓ **IPAM drivers**

- Manages the allocation and assignment of IP addresses to containers within a Docker network.

- ✓ **Network architecture**
    - Provides connectivity between endpoints

IPAM driver:

An IPAM (IP Address Management) driver is a component that manages the allocation and assignment of IP addresses to containers within a Docker network.

The IPAM driver is responsible for the following tasks:

1. **IP Address Allocation**: The IPAM driver manages the pool of available IP addresses within a Docker network. When a container is created, the IPAM driver assigns an IP address to it from the available pool. This ensures that containers are assigned unique and non-conflicting IP addresses.
2. **Subnet Management**: The IPAM driver defines the subnet configuration for the Docker network, including the range of IP addresses that can be assigned to containers. It ensures that the IP addresses allocated to containers fall within the defined subnet.
3. **Gateway Configuration**: The IPAM driver sets up the default gateway for the Docker network. The gateway is the IP address used by containers to reach external networks. The IPAM driver assigns the appropriate gateway IP address to containers when they are connected to the network.
4. **Custom IP Address Assignment**: Some IPAM drivers allow for custom IP address assignment, where you can specify a particular IP address to be assigned to a container. This can be useful when you require specific IP address management for certain containers.
5. **Integration with External IPAM Systems**: Docker IPAM drivers can integrate with external IP Address Management systems, such as IPAM systems used in large-scale network deployments. This allows Docker to leverage existing IPAM infrastructure and management processes.