Introduction To Docker Networking: Advantages and Working

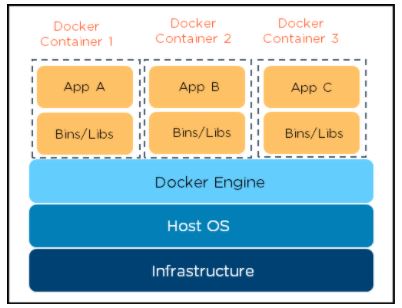


In the Docker world, Network admins have a huge responsibility of understanding the network components found in virtualization platforms like Microsoft, Red Hat, etc. But, deploying a container isn’t simple; it requires strong networking skills to configure a container architecture correctly. To solve this issue, Docker Networking was introduced.

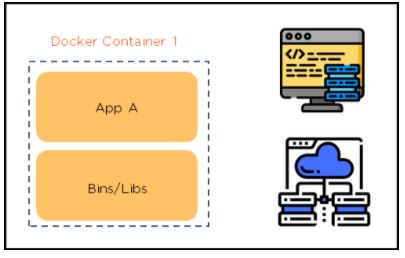
Before understanding Docker Networking, let’s quickly understand the term ‘Docker’ first.

What is Docker?

[Docker](https://www.simplilearn.com/tutorials/docker-tutorial/what-is-docker) is a platform that utilizes OS-level virtual software, to help users to develop, deploy, manage, and run applications in a Docker Container with all their library dependencies.



Note: [Docker Container](https://www.simplilearn.com/tutorials/docker-tutorial/what-is-docker-container" \o "Docker Container" \t "_blank) is a standalone package that includes all the dependencies (frameworks, libraries, etc.) required to execute an application.



Now, let’s dig into what Docker networking is, and then understand its advantages.

What is Docker Networking?

Docker networking enables a user to link a Docker container to as many networks as he/she requires. Docker Networks are used to provide complete isolation for Docker containers.

Note: A user can add containers to more than one network.

Let’s move forward and look at the Advantages of networking.

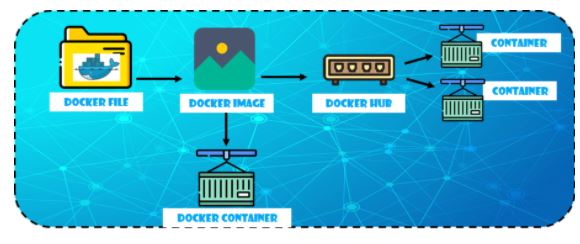
Advantages of Docker Networking

Some of the major benefits of using Docker Networking are:

* They share a single operating system and maintain containers in an isolated environment.
* It requires fewer OS instances to run the workload.
* It helps in the fast delivery of software.
* It helps in application portability.

How Does Docker Networking Work?

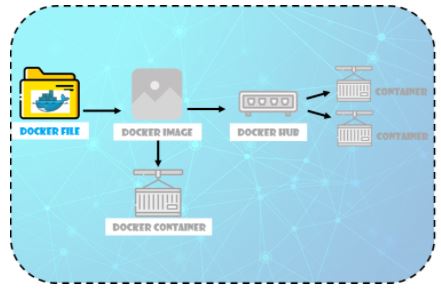
For a more in-depth understanding, let’s have a look at how Docker Networking works. Below is a diagrammatic representation of the Docker Networking workflow:



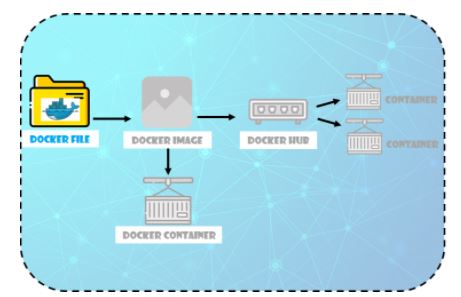
* [Docker File](https://www.simplilearn.com/tutorials/docker-tutorial/what-is-dockerfile) builds the [Docker Image.](https://www.simplilearn.com/tutorials/docker-tutorial/docker-images" \o "Docker Image." \t "_blank)
* Docker Image is a template with instructions, which is used to build Docker Containers.
* Docker has its own cloud-based registry called Docker Hub, where users store and distribute container images.
* Docker Container is an executable package of an application and its dependencies together.

Functionalities of the different components:

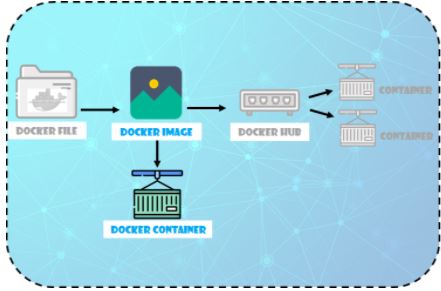
* Docker File has the responsibility of building a Docker Image using the build command



* Docker Image contains all the project’s code.



* Using Docker Image, any user can run the code to create Docker Containers.



* Once Docker Image is built, it’s either uploaded in a registry or a Docker Hub

Now that you know how Docker networking works, it is important to understand the container network model.

Container Network Model

This concept will help you to build and deploy your applications in the  Docker tool.

Let’s discuss the components of the container network model in detail:

Network Sandbox

* It is an isolated sandbox that holds the network configuration 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, as it represents a container’s network configuration such as IP-address, MAC-address, DNS, etc.
* The endpoint establishes the connectivity for container services (within a network) with other services
* It helps in providing connectivity among the endpoints that belong to the same network and isolate them from the rest. So, whenever a network is created, or configuration is changed, the corresponding Network Driver will be notified with an event

Docker Engine

* It is the base engine installed on your host machine to build and run containers using Docker components and services
* Its task is to manage the network with multiple drivers
* It provides the entry-point into libnetwork to maintain networks, whereas libnetwork supports multiple virtual drivers

So, those were the key concepts in the container network model. Going ahead, let’s have a look at the network drivers.

Network Drivers

Docker supports networking for its containers via network drivers. These drivers have several network drivers.

In this article, we will be discussing how to connect your containers with suitable network drivers. The network drivers used in Docker are below:

* Bridge
* Host
* None
* Overlay
* Macvlan

Bridge

* It is a private default network created on the host
* Containers linked to this network have an internal IP address through which they communicate with each other easily
* The Docker server (daemon) creates a virtual ethernet bridge docker0 that operates automatically, by delivering packets among various network interfaces
* These are widely used when applications are executed in a standalone container

Host

* It is a public network
* It utilizes the host’s IP address and TCP port space to display the services running inside the container
* It effectively disables network isolation between the docker host and the docker containers, which means using this network driver a user will be unable to run multiple containers on the same host

None

* In this network driver, the Docker containers will neither have any access to external networks nor will it be able to communicate with other containers
* This option is used when a user wants to disable the networking access to a container
* In simple terms, None is called a loopback interface, which means it has no external network interfaces

Overlay

* This is utilized for creating an internal private network to the Docker nodes in the [Docker swarm](https://www.simplilearn.com/tutorials/docker-tutorial/docker-swarm" \o "Docker swarm" \t "_blank) cluster
* Note: Docker Swarm is a service for containers which facilitates developer teams to build and manage a cluster of swarm nodes within the Docker platform
* It is an important network driver in Docker networking. It helps in providing the interaction between the stand-alone container and the Docker swarm service

Macvlan

* It simplifies the communication process between containers
* This network assigns a MAC address to the Docker container. With this Mac address, the Docker server (daemon) routes the network traffic to a router
* Note: Docker Daemon is a server which interacts with the operating system and performs all kind of services
* It is suitable when a user wants to directly connect the container to the physical network rather than the Docker host

FREE DevOps Certification Training

Basic Docker Networking Commands

Let’s discuss some of the important networking commands that are widely used by the developer teams.

* List down the Networks associated with Docker

docker network ls

The above command displays all the networks available on the Docker ecosystem

* Connect a Running Container to a Network

$ docker network connect multi-host-network container

In the command shown above, You can also use the docker network option to start a container and immediately connect it to multiple host networks.

* Specify the IP Address that you want to assign to the Container

$ docker network connect --IP 10.10.36.122 multi-host-network container

In the above command, a user can specify the IP address (for example, 10.10.36.122) that he/she wants to assign to the container interface.

* Create a Network alias for a Container

$ docker network connect --alias db --alias mysql multi-host-network container2

In the above command, we have specified Aliases to define new commands and to rectify incorrect input

* Disconnect a Container from a Network

$ docker network disconnect multi-host-network container1

In the above command, the disconnect option is used to stop the running docker containers on multiple host network

* Remove a Network

$ docker network rm network\_name

In the above command, the rm option is used to remove a network from the Docker ecosystem

* Remove Multiple Network

$ docker network rm 3695c422697f network\_name

The above command can be used when a user wants to remove multiple networks at a time

* Remove all Unused Networks

$ docker network prune

The above ‘prune’ command can be used when a user wants to remove all unused networks at a time

Conclusion

That concludes the Docker Networking article. In this write-up, we learned what Docker and Docker Networking are, some of its benefits, how Docker networking works, the Container network model, network drivers, and finally, we saw some of the basic Docker networking commands.