Introduction to Docker Networks

A brief introduction to Docker networks

We'll cover the following

In the fundamentals section, we discussed that we will add a database to the app in the coming section. We will add the database as a container and our app will connect to the database using the Docker network.

So, it is important to know a little bit about the Docker network so that you can troubleshoot any problem you face in future.

Network fundamentals

> The network is nothing but an area that allows somebody to reach others with defined permissions and protocols.

We will not go deep into networking, but just small important commands and concepts regarding Docker.

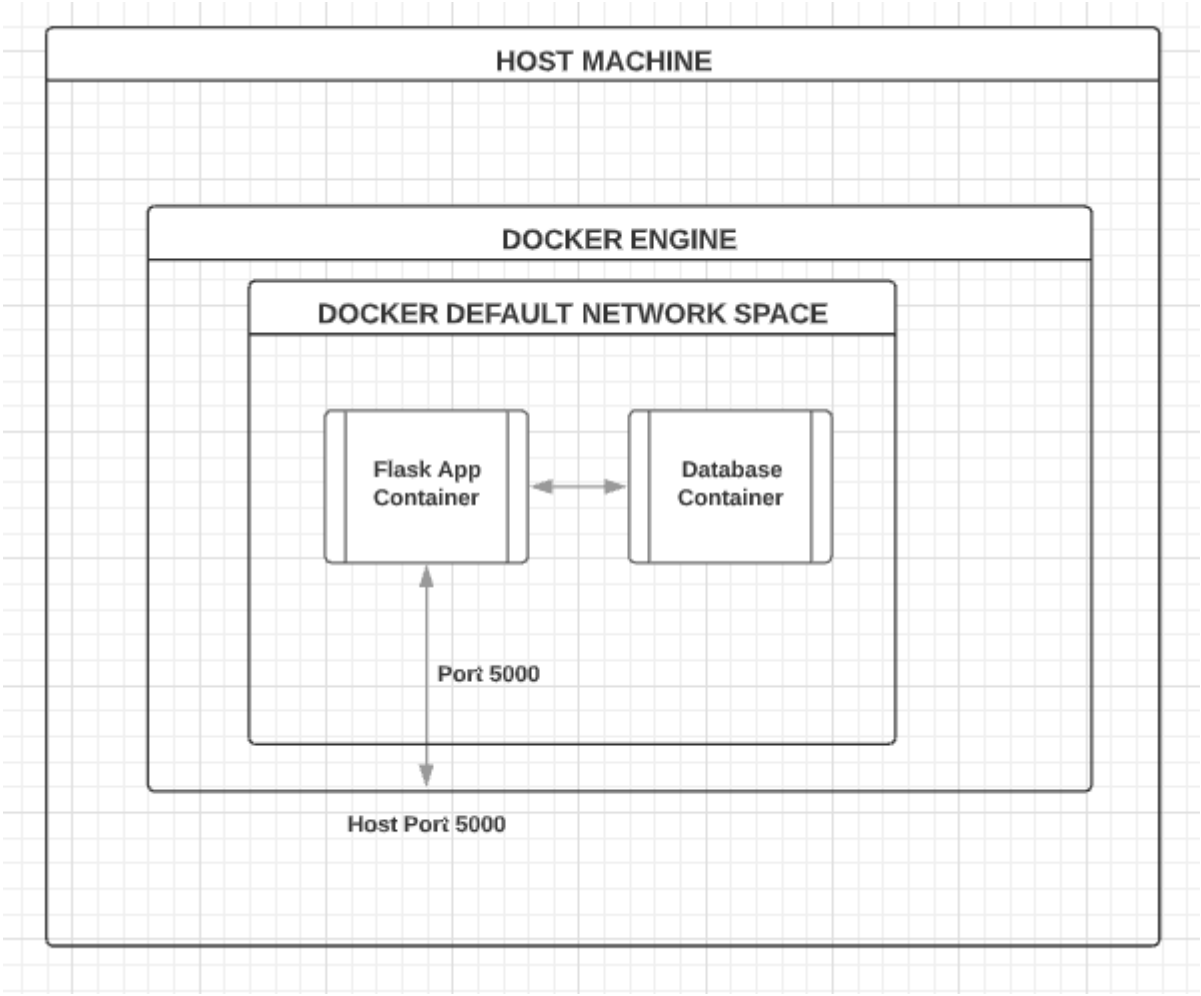
Network terminologies and commands

There are some terminologies used in Networking. Some of them are:

- Hostname: This is the most-frequently-used word. It's the name of the machine to identify it in a network.
- ping: `ping <hostname>/<ip>` command is used to check the connectivity of machine from the machine from which the command is run.

- subnet: This is a small isolated part of a network. It is like creating boundaries to a particular part in the same network.
- DNS: This is like a phone directory. All the reachable hosts are mapped in this domain name service/system using IP and hostname so that you can either reach them by name or IP.
- /etc/hosts: This is the important file. This file has all the reachable hosts with their IP addresses. We will frequently refer to this file.

Docker communication



As you can see in the diagram, we have connected the container's port 5000 to the host's 5000 and as a result, we can access whatever is running on the 5000 port on the container.

Coming to the database container, it will be in the same network as our app container. Hence, we can access the database from our app container.

Since there is no interface to the database container from the host machine, we won't be able to access the DB from the host machine. If needed, we can create an interface and access it from the host.

> By default, all containers run in the default network space of Docker. Hence, every container can communicate with others. We can create network isolation if it is needed.

Docker network types

By default, Docker creates three networks.

Run the `docker network ls` command like so:

```
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
915fe26ffbc8        bridge              bridge              local
3ac2c0505d62        host                host                local
753c94184c7b        none                null                local
```

When you initially install Docker, the platform automatically configures three different networks that are named none, host, and bridge.

The none and host networks cannot be removed, they're part of the network stack in Docker, but not useful to network administrators since they have no external interfaces to configure.

Admins can configure the bridge network, also known as the Docker0 network. This network automatically creates an IP subnet and gateway.

All containers that belong to this network are part of the same subnet, so communication between containers in this network can occur via IP addressing.

> A drawback of the default bridge network is that automatic service discovery of containers using DNS is not supported. Therefore, if you want containers that belong to the default network to be able to talk to each other, you must use the --link option to statically allow communications to occur. Additionally, communication requires port forwarding between containers.

I think this is enough knowledge to get started with networking. Once we link our container, you'll come to know the meaning of the note.

So, let’s proceed and add a database to our app.