Docker Networking

Ritik Pansuriya
Information Technology Department
National Institute of Technology Karnataka
Surathkal, India
ritik.rjt@gmail.com

Karthik R
Information Technology Department
National Institute of Technology Karnataka
Surathkal, India
r.karthikdel@gmail.com

Shidharth S
Information Technology Department
National Institute of Technology Karnataka
Surathkal, India
s.shidharth@hotmail.com

***Abstract*—One of the reasons Docker containers and services are so powerful is that you can connect them together, or connect them to non-Docker workloads. Docker containers and services do not even need to be aware that they are deployed on Docker, or whether their peers are also Docker workloads or not. Whether your Docker hosts run Linux, Windows, or a mix of the two, you can use Docker to manage them in a platform-agnostic way.**

***Keywords*—Docker, Docker Networking**

I. Introduction

In the last couple of years, there has been a lot of excitement about Docker and many big & small companies have moved toward Docker and have incorporated it as part of their DevOps application deployment process.

Docker is a tool that enables you to create, deploy, and manage lightweight, stand-alone packages that contain everything needed to run an application (code, libraries, runtime, system settings, and dependencies). These packages are called containers.

Each container is deployed with its own CPU, memory, block I/O, and network resources, all without having to depend upon an individual kernel and operating system. While it may be easiest to compare Docker and virtual machines, they differ in the way they share or dedicate resources.

A. ADVANTAGES OF DOCKER CONTAINER

The demand and the advancement of Linux containers can be seen in the last few years. Docker has become popular very quickly, because of the benefits provided by docker container. The main advantages of docker are speed, portability, scalability, rapid delivery, and density.

1. Speed

Speed is one of the most exceedingly touted advantages of Containers. When the benefits of using docker are highlighted, it would be incredible not to mention about the speed of docker in the conversation (Chavis & Architect, 2015). The time required to build a container is very fast because they are really small. Development, testing, and deployment can be done faster as containers are small.Containers can be pushed for testing once they have been built and then from there, on to the production environment

2. Portability

Those applications that are built inside docker containers are extremely portable. These portable applications can easily be moved as a single element and the performance remains the same.

3. Scalability

Docker has the ability that it can be deployed in several physical servers, data servers, and cloud platforms. It can also be run on every Linux machine. Containers can easily be moved from a cloud environment to local host and from there back to cloud again at a fast pace. Adjustments can easily be done; the scale can simply be adjusted by the user according to the need.

4. Rapid Delivery

The format of a Docker Containers is standardized so programmers do not have to stress over one another's tasks. The responsibility of the administrator is to deploy and maintain the server with containers, whereas the responsibility of the programmer is to look after the applications inside the docker container. Containers can work in every environment as they have all the required dependencies embedded within the applications and they are all tested. Docker provides a reliable, consistent, and improved environment, so predictable results can be achieved when codes are moved between development, test and production systems (Chavis & Architect, 2015).

5. Density

Docker uses the resources that are available more efficiently because it does not use a hypervisor. This is the reason that more containers can be run on a single host as compared to virtual machines. The performance of a Docker Containers is higher because of higher density and no overhead wastage of resources.

B. Problem Statement

To create docker containers and understand how they communicate with each other , in terms of networking.

C. Objectives

- To demonstrate docker networking
- To show working of DNS in docker
- To show port forwarding in docker containers
- To configure a docker container to use proxy server.

II. Literature Survey

The following papers were an inspiration to this project, and have therefore been listed here:

1. An Introduction to Docker and Analysis of its Performance by Babak Bashari Rad, Harrison John Bhatti, Mohammad Ahmadi.

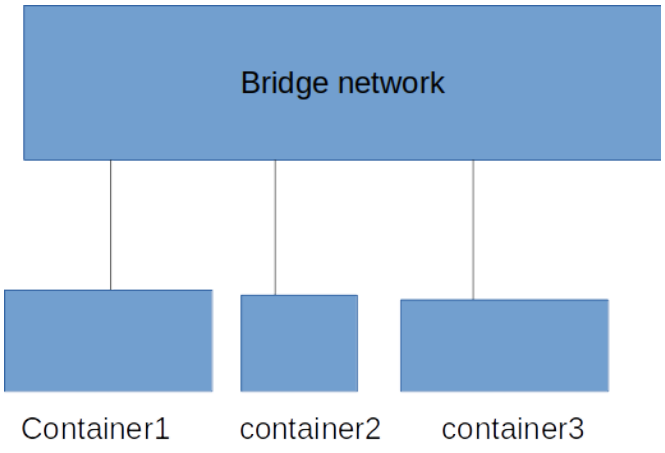
III. Hardware and Software Requirements

- 4GB RAM
- BIOS-level hardware virtualization support must be enabled in the BIOS settings.
- 64bit processor
- Docker has to be installed

IV. Methodology

A. *Docker Networking*

In terms of networking, a bridge network is a Link Layer device which forwards traffic between network segments. A bridge can be a hardware device or a software device running within a host machine's kernel.
In terms of Docker, a bridge network uses a software bridge which allows containers connected to the same bridge network to communicate, while providing isolation from containers which are not connected to that bridge network. The Docker bridge driver automatically installs rules in the host machine so that containers on different bridge networks cannot communicate directly with each other.
Bridge networks apply to containers running on the same Docker daemon host.

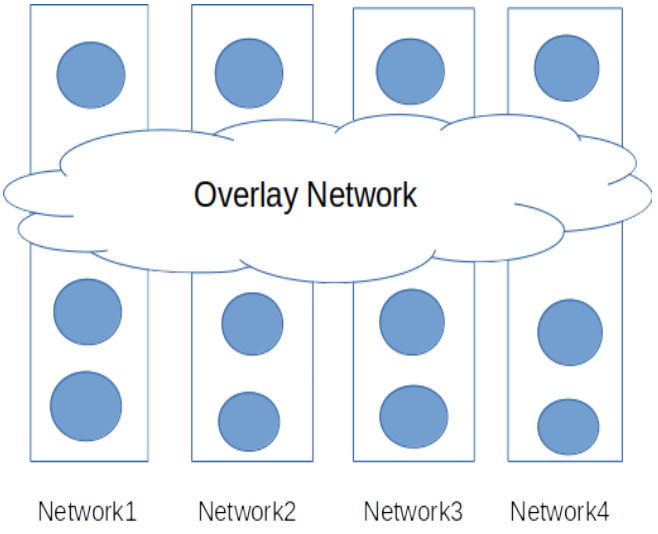


Containers by default connect to the default bridge network as shown above.

By default, the container is assigned an IP address for every Docker network it connects to. The IP address is assigned from the pool assigned to the network, so the Docker daemon effectively acts as a DHCP server for each container. Each network also has a default subnet mask and gateway.

B. *Docker Networking between different User Defined Networks*

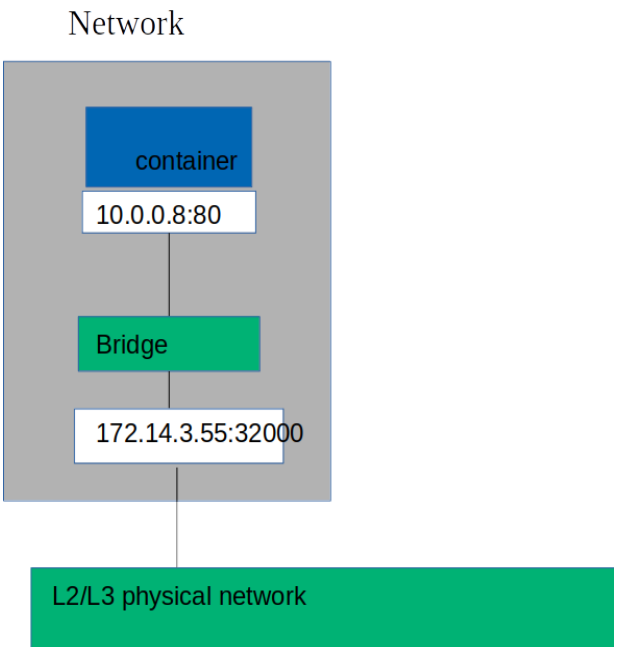
Overlay networks help docker containers across various networks communicate.The overlay network driver creates a distributed network among multiple Docker daemon hosts. This network sits on top of (overlays) the host-specific networks, allowing containers connected to it to communicate securely. Docker transparently handles routing of each packet to and from the correct Docker daemon host and the correct destination container.



The blue-filled circles represent docker containers.

C. *Port Forwarding in Docker Containers*

The type of network a container uses, whether it is a bridge, an overlay, or a custom network plugin, is transparent from within the container. From the container's point of view, it has a network interface with an IP address, a gateway, a routing table, DNS services, and other networking details.
By default, when you create a container, it does not publish any of its ports to the outside world.



Port forwarding in docker where port 80 on docker is mapped to port 32000 on the host,thus any query to port 32000 on the host will be forwarded to the port 80 of the container

D.*DNS in docker*

By default, a container inherits the DNS settings of the Docker daemon these settings can be overridden on a per-container basis. The default bridge network does not have a pre-defined DNS however user defined networks contain embedded DNS.

E.*Configure docker to use Proxy Server*

By default, Docker assumes that the system running Docker and executing Docker commands has general access to the internet. Often in large corporate networks this is simply not the case. More often than not, a corporate network will route all internet traffic through a proxy.
While this setup is generally transparent to the end user, this type of network environment often neglects command line tools such as Docker.

V. Docker Commands

- Create and run a docker container from the given image:
  Docker container run <image_name> -
- List all the running container :
  docker container ls
- Create a network with specified name and type:
  docker network create <type> <network_name>
- Lists all networks: docker network ls
- Kill the mentioned process:
  docker kill <process_name>
- Container of given id is removed(deleted):
  docker container rm <container_id>
- Gives port information about the given container: docker container port <container_id>
- Gives details about the network mentioned:
  docker network inspect <network_name>

DISADVANTAGES OF DOCKER CONTAINER

There are some drawbacks of docker containers, which are listed below:

- Complete virtualization is not provided by a docker because it depends on the Linux kernel, which is provided by the local host.
- Currently, docker does not run on older machines. It only supports 64-bit local machines.
- The complete virtualized environment must be provided by the docker container for Windows and Mac machines. Even though the boot2docker tool fills this gap, but still, it should be checked whether it makes obstructions to acceptance by users of these systems or the integration and performance with the host machine's operating system are adequate .
- It is necessary that the possibility of security issues should be evaluated. Building off trusting binaries could be made easier by digitally signing docker images, for future support.
- An important concern is to check if the teaching community or scientific researcher will significantly think of adopting docker.

VI. Conclusion

Docker automates the applications when they are containerized. An extra layer of docker engine is added to the host operating system. The performance of docker is faster than virtual machines as it has no guest operating system and less resource overhead. Although docker networking is same as individual machines but contains much lighter and easier to implement.

Acknowledgment

We would like to thank our professors in our department, particularly Prof Geetha V, Prof Thanmayee S, who helped us shape the idea of our project.

Authors and Affiliations

This report is a part of mini-project for Computer Communications and Networking. Following are the members who contributed:

- Ritik Pansuriya, Roll No. 181IT237
- Karthik R, Roll No. 181IT235
- Shidharth S, Roll No. 181IT243

References

[1] [1] Boettiger, C. (2015). An introduction to Docker for reproducible research. ACM SIGOPS Operating Systems Review, 49(1), 71-79.

[2]] Scheepers, M. J. (2014). Virtualization and containerization of application infrastructure: A comparison.

[3] Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2014). An updated performance comparison of virtual machines and linux containers. technology, 28, 32.

[4] Turnbull, J. (2014). The Docker Book: Containerization is the new virtualization.

[5] Vase, T. (2015). Advantages of Docker.