Understanding Docker Networking and Container Management

Introduction:

Docker is a powerful containerization platform that not only simplifies application deployment but also provides robust networking capabilities. In this document, we will explore various aspects of Docker networking, container connectivity, IP management, and persistent data storage.

Docker Networking:

When Docker is installed, it automatically configures a software-defined network (SDN) that includes a router and switch. The networking component is handled by Docker itself, creating a seamless experience for users. The key element is the L3 Bridge, a single device that functions as both a router and a switch.

- Commands to Explore Docker Networking:
    - `docker network ls`: List all Docker networks.
    - `docker network inspect bridge`: View details of the bridge driver.

Network Address Translation (NAT) and IP Management:

Docker uses Source NAT (SNAT) to mask private IP addresses when containers communicate with external public IPs. The bridge, by default, has pre-enabled SNAT, ensuring connectivity to the outside world.

- Commands for IP Management:
    - `ifconfig`: Displays container IP addresses.
    - `route -n`: Shows the gateway address.
    - `docker run --net none`: Launches a container with no network connectivity.

Custom Docker Networks:

Users can create custom Docker networks using the `docker network create` command. This allows for more control over IP addressing and other network parameters.

docker network create –driver bridge –ipam-driver default –subnet 11.1.2.0/24 –gateway 11.1.2.1 lwnet1

Static IP and Connectivity:

Dynamic IPs are assigned by default, but static IPs can be set using the `--ip` command during container creation.

Example:

docker run -dit –ip 172.17.0.101 –name os10 ubuntu:14.04

Container Connectivity and Communication:

Containers can communicate using container names as hostnames. The `--link` option establishes connections between containers.

Example:

- docker run -dit –name client ubuntu:14.04
- docker run -dit –name server –link client ubuntu:14.04

Persistent Data Storage:

Understanding the difference between ephemeral and persistent data is crucial. Docker provides the ability to mount directories from the host system into containers, ensuring data persistence.

Example:

- docker run -it –name os1 -v /mydata:/cdata centos:7

ENTRYPOINT and CMD Keywords:

These keywords play a role in defining the command that is executed when a container starts. The `ENTRYPOINT` specifies the command, while `CMD` provides default arguments.

Example:-

```
FROM centos:7

RUN touch /a.txt

ENTRYPOINT ["ls"]

CMD ["–help"]
```

Image Sharing:

Docker images can be saved locally using `docker save` and shared through various means. Loading an image is done with `docker load`.

```
docker save aa:v1 -o myaa.tar

docker load -i myaa.tar
```