Networking refresher

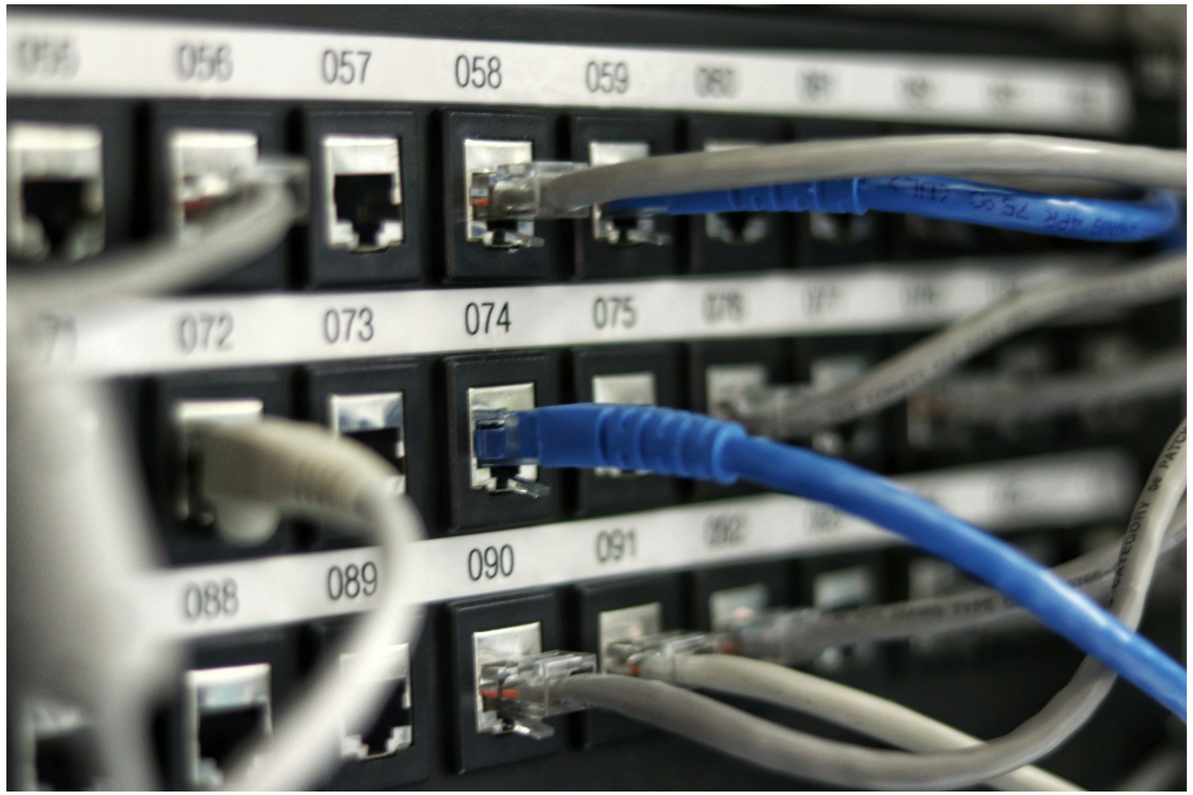
INTERMEDIATE DOCKER

Mike Metzger
Data Engineering Consultant

datacamp

What is networking?

- A computer network consists of systems communicating via a defined method
- Varying levels of communication, physical or logical, referred to as *protocols*
- Common physical networks include Ethernet and WiFi
- Logical networking includes TCP/IP, HTTP, and SMTP
- Networks are defined in various layers or levels. Often referred to as a networking stack



[1] Photo by Jordan Harrison on Unsplash.

Networking terms

- Host
  - General term for a computer
- Network
  - Group of hosts
- Interface
  - Actual connection from a host to a network, such as Ethernet or WiFi
  - Can be virtual, meaning entirely in software
- LAN
  - Local Area Network, or set of computers at a given location
- VLAN
  - Virtual LAN, or a software LAN

Internet Protocol

- IP
  - Internet protocol, method to connect between networks using IP addresses
- IPv4
  - Version of IP supporting 4.2 billion addresses, currently exhausted
- IPv6
  - Newer version of IP, supporting 2^128 addresses, still being deployed

```
IPv4: 10.10.10.1
IPv6: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
```

TCP / UDP

- TCP
  - Transmission Control Protocol, used to reliably communicate between hosts on IP networks
- UDP
  - User Datagram Protocol, used to communicate between hosts on IP where communication is not required.

Ports

- Port
  - Addresses services on a given host, a value between 0 and 65535, used to communicate between hosts via TCP or UDP
  - Ports below 1024 are typically reserved for privileged accounts
  - Values above 1024 are usually ephemeral or temporary ports
  - Applications **listen** on a port.

Application protocols

- HTTP/HTTPS
  - Application protocol, defaulting to TCP port 80 for web communication. Secure version on TCP 443
- SMTP
  - Email transfer protocol, over TCP port 25
- SNMP
  - Network management protocol, over UDP port 161

Docker and networking

- Can communicate between containers
- Can communicate with the host system
- Depending on settings can communicate with external hosts
- Typical communication is handled by exposing ports from container to host
- Acts like a translation between containers and hosts

Docker and IP

- Containers can have IP addresses
- Use `ifconfig <interface>` or `ip addr show <interface>` from within container to find addresses
- Use `ping -c <x> <host>` to verify connectivity
  - `ping -c 3 myhost`

```
repl@4221eef4-d3b6-45ef-b95a-71e78ac62619:~$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 10.4.77.205  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fe80::5457:73ff:fe94:f6e4  prefixlen 64  scopeid 0x20<link>
        ether 56:57:73:94:f6:e4  txqueuelen 0  (Ethernet)
        RX packets 1811  bytes 272039 (272.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1295  bytes 137826 (137.8 KB)
        TX errors 0  dropped 1 overruns 0  carrier 0  collisions 0
```

Let's practice!

INTERMEDIATE DOCKER

Making network services available in Docker

INTERMEDIATE DOCKER

Mike Metzger
Data Engineering Consultant

Network services

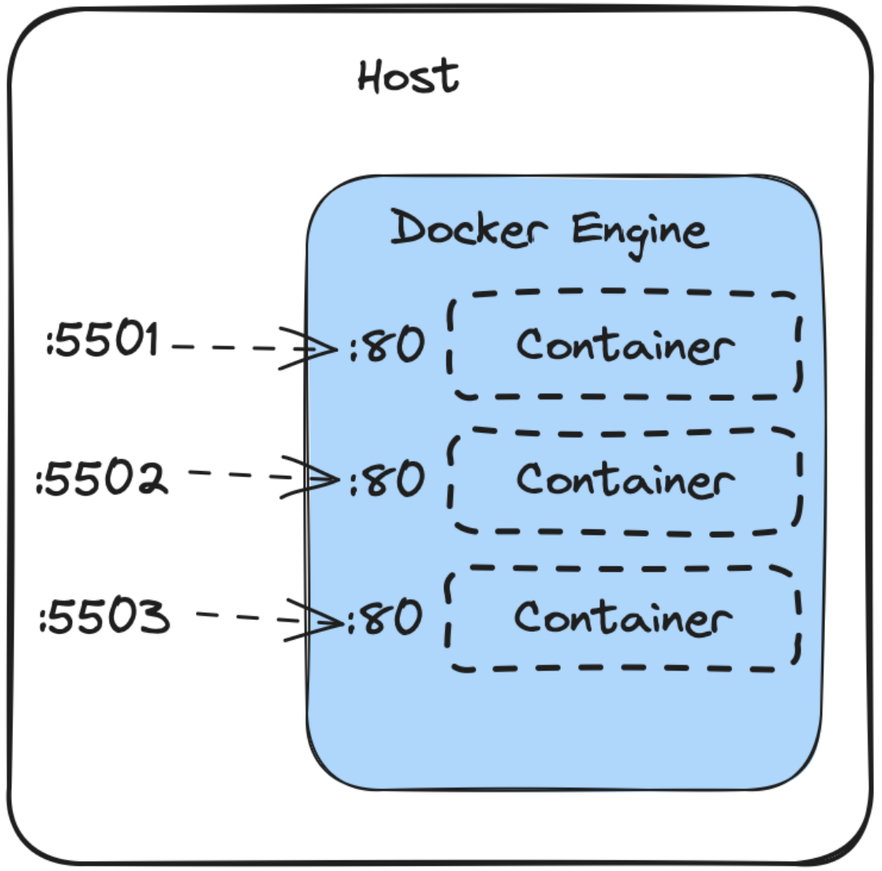
- Network services **listen** on a given port
- Only one program can listen on an IP:port combo at a given time
  - For example, 10.1.2.3:80 would be listening on 10.1.2.3 on port 80.
- Consider trying to debug different versions of a web server that listens on port 80
  - Could only run one copy of the application at a time given that it listens only on that port

Containerized services

- Wrapping application in a container means that each container can now listen on that port (as the IP:port combo is different, each container has a different IP)
- Can have multiple copies of the containers running at once
- But how to connect to container's version of application from the host?

Port mapping

- The answer is the use of **port mapping**, or port forwarding / translation
- Port mapping takes a connection to a given IP:port and automatically forwards it to another IP:port combo
- In this case, we could map an unused port on our host and point it to port 80 on the container(s)
- The Docker engine can handle this automatically if we configure it to



Enabling port mapping

- To enable port mapping on a given container, we use the `docker run` command, and the `-p` flag
- `-p <host port>:<container port>`
- `-p 5501:80`
- Can have multiple `-p` flags for different ports

```
repl@host:~$ docker run -p 5501:80 nginx
repl@host:~$ docker ps -a
CONTAINER ID   IMAGE     ... PORTS                                      NAMES
84266724ff47   nginx     ... 0.0.0.0:5501->80/tcp, :::5501->80/tcp   coiled_elgamal
```

Let's practice!

INTERMEDIATE DOCKER

Exposing ports with Dockerfiles

INTERMEDIATE DOCKER

Mike Metzger
Data Engineer

Exposing services

- `EXPOSE` command
- Defines which ports the container will use at runtime
- Can be defined as `<number>` , `<number>/tcp` , or `<number>/udp`
  - Such as `EXPOSE 80` or `EXPOSE 80/tcp`
- Multiple entries permitted
- Used as a documentation method

Using the -p / -P flags

- Still requires use of `-p` or `-P` options to `docker run` to make the ports available outside the container
- The `-P` option will automatically map an ephemeral port to the exposed port(s). Must use `docker ps -a` to see which ports are mapped.
- Using `-p<host port>:<container port>` allows use of specific ports.

EXPOSE example

```
# Dockerfile
FROM python:3.11-slim
ENTRYPOINT ["python","-mhttp.server"]
# Let the Docker engine know
# port 8000 should be available
EXPOSE 8000
```

- Create a container from the image

```
docker run pyserver
```

- Print the state of the container

```
docker ps -a
```

```
CONTAINER ID   IMAGE          ...   PORTS      NAMES
8c3d320255ae   pyserver       ...   8000/tcp   angry_chaum
```

Making ports reachable

- Automatically map temporary port from host to the container

```
docker run -P pyserver
```

```
docker ps -a
```

```
CONTAINER ID   IMAGE       ...   PORTS                     NAMES
6bb458ef25da   pyserver    ...   0.0.0.0:55001->8000/tcp   beautiful_lamarr
```

Finding exposed ports

- `docker inspect` provides a lot of information

```
docker inspect <id>
```

```
"NetworkSettings": {
  "Bridge": "",
  "Ports": {
    "8000/tcp": [{
                  "HostIp": "0.0.0.0",
                  "HostPort": "55001"
                   }]
           },
...
```

Let's practice!

INTERMEDIATE DOCKER

Docker networks

INTERMEDIATE DOCKER

Mike Metzger
Data Engineering Consultant

Docker networking

- Docker has extensive networking options
- Can create networks to communicate between containers, host, and external systems
- Will cover various commands to interact with networks.

Docker networking types

- Docker supports different networking types, using drivers
  - **bridge**: Default driver, allows connections out, connections in if exposed
  - **host**: Allows full communication between host and containers
  - **none**: Isolate container from network communications
  - Many others, including custom drivers
- Will mostly use the bridge driver to create our own networks

Working with Docker networks

Several commands:

- `docker network`
  - `docker network <command>`
  - `docker network <command> --help`
  - `docker network ls` to list all docker networks on the host
  - `docker network create` to create a network
  - `docker network rm` to remove a network

Docker network example

- Create a Docker network named `mynetwork`

```
repl@host:~$ docker network create mynetwork
```

```
5ff0febab98f73b74dd753eb44a30f7d7291052b3b1d58b0134589221cb8e33d
```

```
repl@host:~$ docker network ls
NETWORK ID     NAME        DRIVER    SCOPE
2edc5ae4838c   bridge      bridge    local
a92988382711   host        host      local
5ff0febab98f   mynetwork   bridge    local
5464ed866dad   none        null      local
```

Attaching containers to networks

- How to connect container to a network?
- `docker run --network <networkname> ...`
- `docker run --network mynetwork ubuntu bash`
- Can also connect containers later
  - `docker network connect <networkname> <container>`
  - `docker network connect mynetwork ubuntu-B`

docker network inspect

- How to check details of network?
- `docker network inspect <networkname>`
- Provides configuration info and IP addresses assigned to containers

```
repl@host:~$ docker network inspect mynetwork
```

docker network inspect example

```
        "Name": "mynetwork",
        ...
        "Driver": "bridge",
        ...
        Containers": {    "2be08aa942029191350d4bceb8816254af8713dd6f7dcbadcab8f068f
                "Name": "unruffled_kare",
                "EndpointID": "29739356ae200e1e901d2eabef05efaca0fb37e1a4e1a4c3bf369
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
```

Let's practice!

INTERMEDIATE DOCKER