Basic Networking with Docker

Networks can be configured to provide complete isolation for containers, which enable building web applications that work together *securely*.

Docker network

To view Docker networks, run:

```
docker network ls
```

To get further details on networks, run:

```
docker network inspect
```

Default behavior

Docker creates three networks automatically on install: `bridge`, `none`, and `host`. Specify which network a container should use with the `--net` flag. If you create a new network `my_network` (more on this later), you can connect your container (`my_container`) with:

```
docker run my_container --net=my_network
```

Bridge

All Docker installations represent the `docker0` network with `bridge`; Docker connects to `bridge` by default. Run `ifconfig` on the Linux host to view the `bridge` network.

When you run the following command in your console, Docker returns a JSON object describing the `bridge` network (including information regarding which containers run on the network, the options set, and listing the subnet and gateway).

```
docker network inspect bridge
```

Docker automatically creates a subnet and gateway for the `bridge` network, and `docker run` automatically adds containers to it. If you have containers running on your network, `docker network inspect` displays networking information for your containers.

Any containers on the same network may communicate with one another via IP addresses. Docker does not support automatic service discovery on `bridge`. You must connect containers with the `--link` option in your `docker run` command.

The Docker `bridge` supports port mappings and `docker run --link` allowing communications between containers on the `docker0` network. However, these error-prone techniques require unnecessary complexity. Just because you can use them, does not mean you *should*. It's better to define your own networks instead.

None

This offers a container-specific network stack that lacks a network interface. This container only has a local loopback interface (i.e., no external network interface).

Host

This enables a container to attach to your host's network (meaning the configuration *inside* the container **matches** the configuration *outside* the container).

Defining your own networks

You can create multiple networks with Docker and add containers to one or more networks. Containers can communicate within networks but not *across* networks. A container with attachments to multiple networks can connect with all of the containers on all of those networks. This lets you build a "hub" of sorts to connect to multiple networks and separate concerns.

Creating a bridge network

Bridge networks (similar to the default `docker0` network) offer the easiest solution to creating your own Docker network. While similar, you do not simply clone the `default0` network, so you get some new features and lose some old ones. Follow along below to create your own `my_isolated_bridge_network` and run your Postgres container `my_psql_db` on that network:

```
$ docker network create --driver bridge my_isolated_bridge_network
3b7e1ad19ee8bec9628b18f9f3691adecd2ea3395ec248f8fa57a2ec85aa71c1
```

```
$ docker network inspect my_isolated_bridge_network
[
    {
        "Name": "my_isolated_bridge_network",
        "Id": "3b7e1ad19ee8bec9628b18f9f3691adecd2ea3395ec248f8fa57a2ec8
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
```

```
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1/16"
                }
            ]
        },
        "Internal": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

```
$ docker network ls
NETWORK ID          NAME                          DRIVER
fa1ff6106123        bridge                        bridge
803369ddc1ae        host                          host
3b7e1ad19ee8        my_isolated_bridge_network    bridge
01cc882aa43b        none                          null
```

```
$ docker run --net=my_isolated_bridge_network --name=my_psql_db postgres
```

```
$ docker network inspect my_isolated_brige_network
[
    {
        "Name": "my_isolated_bridge_network",
        "Id": "3b7e1ad19ee8bec9628b18f9f3691adecd2ea3395ec248f8fa57a2ec8
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1/16"
                }
            ]
        },
        "Internal": false,
        "Containers": {
```

```
            "b4ba8821a2fa3d602ebf2ff114b4dc4a9dbc178784dad340e78210a1318
                "Name": "my_psql_db",
                "EndpointID": "4434c2c253afed44898aa6204a1ddd9b758ee66f7
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

Any other container you create on this network can immediately connect to any other container on this network. The network isolates containers from other (including external) networks. However, you can expose and publish container ports on the network, allowing portions of your `bridge` access to an outside network.

Creating an overlay network

If you want native multi-host networking, you need to create an overlay network. These networks require a valid key-value store service, such as Consul, Etcd, or ZooKeeper. You must install and configure your key-value store service **before** creating your network. Your Docker hosts (you can use multiple hosts with overlay networks) must communicate with the service you choose. Each host needs to run Docker. You can provision the hosts with Docker Machine.

Open the following ports between each of your hosts:

Protocol	Port	Purpose
udp	4789	data
tcp/udp	7946	control

Check your key-value store service documentation; your service may need more ports open.

Create an overlay network by configuring options on each Docker daemon you wish to use with the network. You may set the following options:

Option	Description
`--cluster-store=PROVIDER://URL`	Describes the location of the key-value store service
`--cluster-advertise=HOST_IP` or `--cluster-advertise=HOST_IFACE:PORT`	The IP address or interface corresponding to the clustering host
`--cluster-store-opt=KEY-VALUE OPTIONS`	Additional options, like a TLS certificate

1. Create the overlay network in a similar manner to the bridge network (network name `my_multi_host_network`):

```
docker network create --driver overlay my_multi_host_network
```

2. Launch containers on each host; make sure you specify the network name:

```
docker run -itd -net=my_multi_host_network my_python_app
```

Once you connect, every container on the network has access to all the other containers on the network, *regardless of the Docker host* serving the container.

Further information

Normally when you detach from a container, the container stops. You can leave it running by pressing `CTRL-p + CTRL-q`.

The Linux `screen` tool might become your new best friend. You can use it to start containers in other "windows" (in your terminal) and jump between the action in one window and the next. Try out these really helpful commands:

Command	Description
`screen -S my_name`	Creates a new screen called "my_name" (so you can later reference it by name rather than ID)
`screen -x`	If you have only one screen open, jumps to that screen; if you have multiple screens open, lists available screens
`screen -x my_name`	Switches to screen with name `my_name`

None and Host

These have specific uses for the Docker installation. Do not bind containers to these networks.

Custom Bridge Networks

Use bridge networks when you need a relatively small network on a single host. Containers you create on your custom bridge network must exist on the same host. Docker *does not* support linking on bridge networks you define.

Custom Overlay Networks

You must *install and configure* your key-value store service (either Consul, Etcd, or ZooKeeper) **before** creating your network.

Next: Securing your System

Best practices for using Docker securely.