Version : **2023.01**

Last update : 2023/12/27 08:34

DOF606 - Overlay Network Management with Docker in Swarm Mode

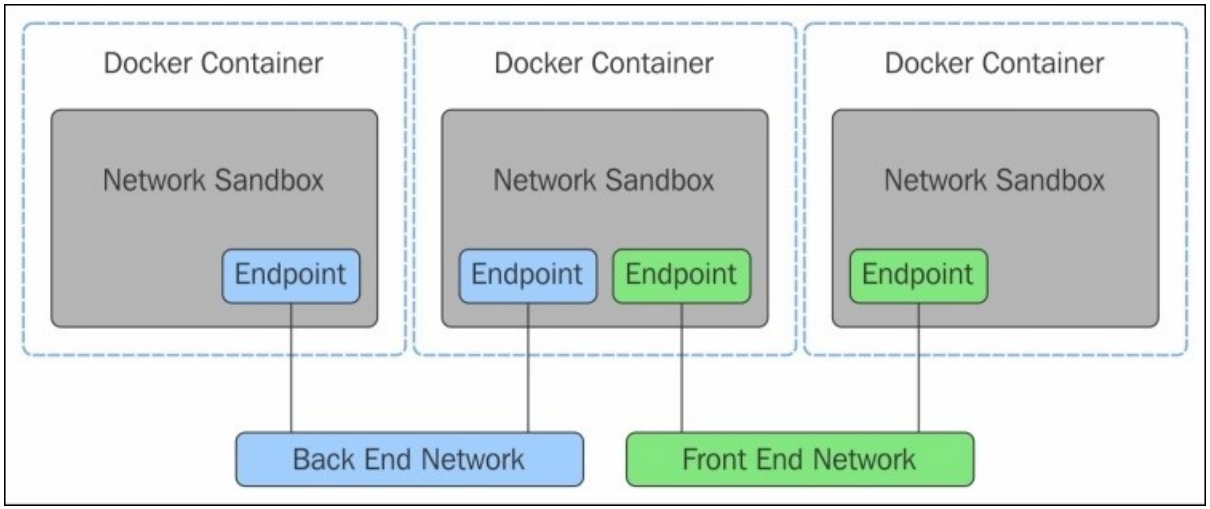
Contents

The Docker Network Model

The Docker network model is **libnetwork**, which implements the **C**ontainer **N**etwork **M**odel (CNM). There are three components in this model:

- Sandbox,
  - contains the container's network configuration, i.e. interface management, routing table and DNS,
- Endpoint,
  - connects a sandbox to a network,

- Network,
  - a group of endpoints which communicate directly.



LAB #1 - Network overlay management

In addition to the **bridge**, **host** and **none** networks, Docker offers two other types of network, namely **overlay** and **macvlan**. This module is about overlay. For more information about the **macvlan** type, see the Docker documentation site **ici**.

As the name suggests, an overlay network is a network that sits on top of the host network. When an overlay network is created, by default it is only available to swarm services. However, it is possible to connect autonomous containers to the overlay network if the **–attachable** option is specified when the network is created. This type of use of the overlay network is not recommended by Docker, which says that support for this feature may be withdrawn.

Traffic linked to the management of swarm services is encrypted by default using the AES algorithm in GCM mode. In order to encrypt application-related data traffic, it is possible to use the **–opt encrypted** option when creating the overlay network. In this case, Docker creates IPSEC tunnels between each node using the same algorithm as the swarm services traffic. There is therefore a performance degradation to be assessed before going into production. In both cases the keys are changed every 12 hours (see https://www.vaultproject.io/docs/internals/rotation.html)

CAUTION: Encryption of application-related data is not compatible with Windows™. When connecting the Windows™ node to an encrypted overlay network, no errors will be reported. However the node will be unable to communicate.

Start by re-creating a swarm using the **manager**, **worker1** and **worker2** virtual machines:

```
root@debian11:~# ssh -l trainee 10.0.2.62
The authenticity of host '10.0.2.62 (10.0.2.62)' can't be established.
ECDSA key fingerprint is SHA256:sEfHBv9azmK60cjqF/aJgUc9jg56slNaZQdAUcvBOvE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.62' (ECDSA) to the list of known hosts.
trainee@10.0.2.62's password: trainee
Linux manager.i2tch.loc 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jul 17 08:27:29 2022 from 10.0.2.1
trainee@manager:~$ su -
Mot de passe : fenestros
root@manager:~#

root@manager:~# docker swarm leave
Node left the swarm.
root@manager:~# docker swarm init --advertise-addr 10.0.2.62
Swarm initialized: current node (tpn1zsk20sfsfafmk2cvefqjc) is now a manager.
```

```
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-23d7n1fkkk9rvlhty106q9390bfpf9daljjguq3s807le6c5qs-
e0s1yqsajvmi7s8t9l9mw48ao 10.0.2.62:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

Connect to **worker1** :

```
root@manager:~# ssh -l trainee 10.0.2.63
The authenticity of host '10.0.2.63 (10.0.2.63)' can't be established.
ECDSA key fingerprint is SHA256:sEfHBv9azmK60cjqF/aJgUc9jg56slNaZQdAUcvBOvE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.63' (ECDSA) to the list of known hosts.
trainee@10.0.2.63's password: trainee
Linux worker1.i2tch.loc 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Mar 21 16:34:26 2021 from 10.0.2.11
trainee@worker1:~$ su -
Mot de passe : fenestros
root@worker1:~#

root@worker1:~# docker swarm leave
Node left the swarm.

root@worker1:~# docker swarm join --token SWMTKN-1-23d7n1fkkk9rvlhty106q9390bfpf9daljjguq3s807le6c5qs-
e0s1yqsajvmi7s8t9l9mw48ao 10.0.2.62:2377
This node joined a swarm as a worker.
```

```
root@worker1:~# exit
déconnexion

trainee@worker1:~$ exit
déconnexion
Connection to 10.0.2.63 closed.

root@manager:~#
```

Connect to **worker2** :

```
root@manager:~# ssh -l trainee 10.0.2.64
The authenticity of host '10.0.2.64 (10.0.2.64)' can't be established.
ECDSA key fingerprint is SHA256:sEfHBv9azmK60cjqF/aJgUc9jg56slNaZQdAUcvBOvE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.64' (ECDSA) to the list of known hosts.
trainee@10.0.2.64's password: trainee
Linux worker2.i2tch.loc 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Mar 21 16:18:25 2021 from 10.0.2.11
trainee@worker2:~$ su -
Mot de passe : fenestros
root@worker2:~#

root@worker2:~# docker swarm leave
Node left the swarm.

root@worker2:~# docker swarm join --token SWMTKN-1-23d7n1fkkk9rvlhty106q9390bfpf9daljjguq3s807le6c5qs-
```

```
e0s1yqsajvmi7s8t9l9mw48ao 10.0.2.62:2377
This node joined a swarm as a worker.

root@worker2:~# exit
déconnexion

trainee@worker2:~$ exit
déconnexion
Connection to 10.0.2.64 closed.

root@manager:~#
```

Check the state of the swarm:

```
root@manager:~# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
ENGINE VERSION
b85hxlixbr1mh1txd1hrfe4us *   manager.i2tch.loc   Ready               Active              Leader
19.03.4
4sui75vvdhmet4qvt0zbvzlzl     worker1.i2tch.loc   Ready               Active
19.03.4
lbjtg5o9kw3x6xg7frm07jfuw     worker2.i2tch.loc   Ready               Active
19.03.4
root@manager:~# docker node ls --filter role=manager
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
ENGINE VERSION
b85hxlixbr1mh1txd1hrfe4us *   manager.i2tch.loc   Ready               Active              Leader
19.03.4
root@manager:~# docker node ls --filter role=worker
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
ENGINE VERSION
4sui75vvdhmet4qvt0zbvzlzl     worker1.i2tch.loc   Ready               Active
19.03.4
lbjtg5o9kw3x6xg7frm07jfuw     worker2.i2tch.loc   Ready               Active
```

```
19.03.4
```

Check the presence of the overlay network **ingress** as well as the bridged network **docker_gwbridge** :

```
root@manager:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
4edb7186dcc9        bridge              bridge              local
d4c9b0c9437a        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
r8htcvc8oxmz        ingress             overlay             swarm
de563e30d473        none                null                local
```

Info: The **docker_gwbridge** network connects the **ingress** network to the host's network adapter and therefore connects the Docker daemon to the other Docker daemons participating in swarm.

Best Practice : Docker recommends using different overlay networks for each application or group of applications.

.1 - Creating an Overlay Network

From the Manager, create an overlay type network called **nginx-net** :

```
root@manager:~# docker network create -d overlay nginx-net
j57jhtug4kjxp22ai1y664lqr
root@manager:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
```

```
dde514eea83f        bridge              bridge              local
d4c9b0c9437a        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
r8htcvc8oxmz        ingress             overlay             swarm
j57jhtug4kjx        nginx-net           overlay             swarm
de563e30d473        none                null                local
```

1.2 - Creating a Service

Create a nginx service that uses the **nginx-net** network:

```
root@manager:~# docker service create --name my-nginx --publish target=80,published=80 --replicas=5 --network
nginx-net nginx
fpydgix3e1rc1qum72gvwcb7f
overall progress: 5 out of 5 tasks
1/5: running [==================================================>]
2/5: running [==================================================>]
3/5: running [==================================================>]
4/5: running [==================================================>]
5/5: running [==================================================>]
verify: Service converged
```

> **Info** : The service publishes port 80, which is visible from the outside.
> Containers communicate with each other without opening additional ports.

Check that the service is working before continuing:

```
root@manager:~# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
```

```
fpydgix3e1rc        my-nginx            replicated          5/5                 nginx:latest        *:80->80/tcp
```

Now take a look at the service details:

```
root@manager:~# docker service inspect my-nginx
[
    {
        "ID": "fpydgix3e1rc1qum72gvwcb7f",
        "Version": {
            "Index": 40
        },
        "CreatedAt": "2019-10-28T06:23:29.17883246Z",
        "UpdatedAt": "2019-10-28T06:23:29.183438696Z",
        "Spec": {
            "Name": "my-nginx",
            "Labels": {},
            "TaskTemplate": {
                "ContainerSpec": {
                    "Image":
"nginx:latest@sha256:922c815aa4df050d4df476e92daed4231f466acc8ee90e0e774951b0fd7195a4",
                    "Init": false,
                    "StopGracePeriod": 10000000000,
                    "DNSConfig": {},
                    "Isolation": "default"
                },
                "Resources": {
                    "Limits": {},
                    "Reservations": {}
                },
                "RestartPolicy": {
                    "Condition": "any",
                    "Delay": 5000000000,
                    "MaxAttempts": 0
                },
```

```
                "Placement": {
                    "Platforms": [
                        {
                            "Architecture": "amd64",
                            "OS": "linux"
                        },
                        {
                            "OS": "linux"
                        },
                        {
                            "Architecture": "arm64",
                            "OS": "linux"
                        },
                        {
                            "Architecture": "386",
                            "OS": "linux"
                        },
                        {
                            "Architecture": "ppc64le",
                            "OS": "linux"
                        },
                        {
                            "Architecture": "s390x",
                            "OS": "linux"
                        }
                    ]
                },
                "Networks": [
                    {
                        "Target": "j57jhtug4kjxp22ai1y664lqr"
                    }
                ],
                "ForceUpdate": 0,
                "Runtime": "container"
```

```
            },
            "Mode": {
                "Replicated": {
                    "Replicas": 5
                }
            },
            "UpdateConfig": {
                "Parallelism": 1,
                "FailureAction": "pause",
                "Monitor": 5000000000,
                "MaxFailureRatio": 0,
                "Order": "stop-first"
            },
            "RollbackConfig": {
                "Parallelism": 1,
                "FailureAction": "pause",
                "Monitor": 5000000000,
                "MaxFailureRatio": 0,
                "Order": "stop-first"
            },
            "EndpointSpec": {
                "Mode": "vip",
                "Ports": [
                    {
                        "Protocol": "tcp",
                        "TargetPort": 80,
                        "PublishedPort": 80,
                        "PublishMode": "ingress"
                    }
                ]
            }
        },
        "Endpoint": {
            "Spec": {
```

```
                "Mode": "vip",
                "Ports": [
                    {
                        "Protocol": "tcp",
                        "TargetPort": 80,
                        "PublishedPort": 80,
                        "PublishMode": "ingress"
                    }
                ]
            },
            "Ports": [
                {
                    "Protocol": "tcp",
                    "TargetPort": 80,
                    "PublishedPort": 80,
                    "PublishMode": "ingress"
                }
            ],
            "VirtualIPs": [
                {
                    "NetworkID": "r8htcvc8oxmzy896xvwvv87k5",
                    "Addr": "10.255.0.5/16"
                },
                {
                    "NetworkID": "j57jhtug4kjxp22ai1y664lqr",
                    "Addr": "10.0.0.2/24"
                }
            ]
        }
    }
]
```


Important: Note here information about the ports and Endpoints used by

the service.

1.3 - Move the Service to another Network overlay

Check the overlay network **nginx-net** on the three nodes:

```
root@manager:~# docker inspect nginx-net
[
    {
        "Name": "nginx-net",
        "Id": "j57jhtug4kjxp22ai1y664lqr",
        "Created": "2019-10-28T07:23:29.492986337+01:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
```

```
        "ConfigOnly": false,
        "Containers": {
            "b2e882e530b10f8fd0b2481f851007f864ce1495bc9fdedcf51a475c0fc03aeb": {
                "Name": "my-nginx.2.bo4q3us1f6m0uwxhqgtau1yg5",
                "EndpointID": "f6f82bcb81ba82191f3988702b0e91f7f5f139c5c88899ad7c95e12ab189e055",
                "MacAddress": "02:42:0a:00:00:04",
                "IPv4Address": "10.0.0.4/24",
                "IPv6Address": ""
            },
            "c0a76b54dad58b0faf80d2f915a10072aa7d726c46036caa3157d22c30dba843": {
                "Name": "my-nginx.4.aqj5vafpqtkc8f4rn4v04x4kn",
                "EndpointID": "813bef65edc4de42d5ec4357013f5b711cd21ce7d1a1c8361c1d989d0d709071",
                "MacAddress": "02:42:0a:00:00:06",
                "IPv4Address": "10.0.0.6/24",
                "IPv6Address": ""
            },
            "lb-nginx-net": {
                "Name": "nginx-net-endpoint",
                "EndpointID": "d087f5fe91481b12ca0b966d01584d143b25c746952bb517441cfad6beba90de",
                "MacAddress": "02:42:0a:00:00:08",
                "IPv4Address": "10.0.0.8/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "1199cab4a6dd",
                "IP": "10.0.2.62"
            },
            {
```

```
                "Name": "69676ae46ab9",
                "IP": "10.0.2.63"
            },
            {
                "Name": "d058d363197d",
                "IP": "10.0.2.64"
            }
        ]
    }
]
```

```
root@worker1:~# docker inspect nginx-net
[
    {
        "Name": "nginx-net",
        "Id": "j57jhtug4kjxp22ai1y664lqr",
        "Created": "2019-10-28T07:23:29.561068917+01:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
```

```
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "50b205e2ed4ccaaad5adc06c508af235557c89c116c819e367a1d925e9c2b564": {
                "Name": "my-nginx.1.gcz867ezj0y46tsdgoz8j3jz2",
                "EndpointID": "a48a43da98acef2748f42ffa992ba302863ed3c417fa3289cbd3aed0e33e97fa",
                "MacAddress": "02:42:0a:00:00:03",
                "IPv4Address": "10.0.0.3/24",
                "IPv6Address": ""
            },
            "lb-nginx-net": {
                "Name": "nginx-net-endpoint",
                "EndpointID": "54ed15511cdd574cb60d37d39257cbf7b30331b24bb069aadb33b457b2864789",
                "MacAddress": "02:42:0a:00:00:0a",
                "IPv4Address": "10.0.0.10/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "69676ae46ab9",
                "IP": "10.0.2.63"
            },
            {
                "Name": "d058d363197d",
                "IP": "10.0.2.64"
            },
            {
                "Name": "1199cab4a6dd",
```

```
                "IP": "10.0.2.62"
            }
        ]
    }
]
```

```
root@worker2:~# docker inspect nginx-net
[
    {
        "Name": "nginx-net",
        "Id": "j57jhtug4kjxp22ai1y664lqr",
        "Created": "2019-10-28T07:23:29.562818383+01:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "31bcb5e553886cd9b3a6b8e70fe0c2bed92fe081bd0def0c94864631a940cbd6": {
```

```
                "Name": "my-nginx.5.t3be85jtp2qlhpmvsl4866s5m",
                "EndpointID": "ffa92f5f3bb7fd2665a8be336ef1e4e2d786790852eb152dac1a2c45f18518ba",
                "MacAddress": "02:42:0a:00:00:07",
                "IPv4Address": "10.0.0.7/24",
                "IPv6Address": ""
            },
            "8e2ce40a6e0d9fb2bc64c264b92164b6ea241a2369d8e6844d00b8952f5729a7": {
                "Name": "my-nginx.3.dma616z2rkbted13zd824fyo2",
                "EndpointID": "99cfb31ce34ccd9b6b15f71c87eddb5f39a84512ec76d215d54bdaaf851d5129",
                "MacAddress": "02:42:0a:00:00:05",
                "IPv4Address": "10.0.0.5/24",
                "IPv6Address": ""
            },
            "lb-nginx-net": {
                "Name": "nginx-net-endpoint",
                "EndpointID": "c0816f6f1e5c046ac1deb8163c5a8cf40765a126bf76b6f10bf1bb708a51dfa1",
                "MacAddress": "02:42:0a:00:00:09",
                "IPv4Address": "10.0.0.9/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "d058d363197d",
                "IP": "10.0.2.64"
            },
            {
                "Name": "69676ae46ab9",
                "IP": "10.0.2.63"
            },
```

```
            {
                "Name": "1199cab4a6dd",
                "IP": "10.0.2.62"
            }
        ]
    }
]
```

> **Important**: Note that the **nginx-net** network has been created automatically on both Workers. Also note the contents of the **Peers** section, which lists the nodes, as well as the **Containers** section, which lists the containers on each node that are connected to the overlay network.

Now create a second overlay network, called **nginx-net-2** :

```
root@manager:~# docker network create -d overlay nginx-net-2
aez5huut9hd472qmldzf2tsud
```

Move the **my-nginx** service to the **nginx-net-2** network:

```
root@manager:~# docker service update --network-add nginx-net-2 --network-rm nginx-net my-nginx
my-nginx
overall progress: 5 out of 5 tasks
1/5: running [==================================================>]
2/5: running [==================================================>]
3/5: running [==================================================>]
4/5: running [==================================================>]
5/5: running [==================================================>]
verify: Service converged
```

Check that the service is running before continuing:

```
root@manager:~# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
fpydgix3e1rc        my-nginx            replicated          5/5                 nginx:latest        *:80->80/tcp
```

Check that there are no containers in the **nginx-net** network:

```
root@manager:~# docker network inspect nginx-net
[
    {
        "Name": "nginx-net",
        "Id": "j57jhtug4kjxp22ai1y664lqr",
        "Created": "2019-10-28T06:21:18.337578134Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
```

```
        "Containers": null,
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": null
    }
]
```

Now check that the containers are in the **nginx-net-2** network :

```
root@manager:~# docker network inspect nginx-net-2
[
    {
        "Name": "nginx-net-2",
        "Id": "aez5huut9hd472qmldzf2tsud",
        "Created": "2019-10-28T10:09:54.465105557+01:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.1.0/24",
                    "Gateway": "10.0.1.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
```

```
        },
        "ConfigOnly": false,
        "Containers": {
            "0bf159064e30d5e788a12baca53ee8e9504a2d7300017fb268cb9e90caaea27a": {
                "Name": "my-nginx.2.81pveqac42zesvuulpbiho7k6",
                "EndpointID": "25c9587e76cfca10d17b10fa967186bc73ca6b444cc2689e43a7243f5d1795b2",
                "MacAddress": "02:42:0a:00:01:05",
                "IPv4Address": "10.0.1.5/24",
                "IPv6Address": ""
            },
            "74e656da8c670fca23270078565af164c4d42415f012ff51ccb02395c6d121e9": {
                "Name": "my-nginx.3.mjj1bsguaaewk61dw7yxxjdlu",
                "EndpointID": "2be3c3e0286d3afb5ba47bbd903151a4d337a45743cb30c46595160223e02fba",
                "MacAddress": "02:42:0a:00:01:07",
                "IPv4Address": "10.0.1.7/24",
                "IPv6Address": ""
            },
            "lb-nginx-net-2": {
                "Name": "nginx-net-2-endpoint",
                "EndpointID": "768a4cc926b5c94a20904e5db500dc62b40a063077a49769ccccc007a6cb61ac",
                "MacAddress": "02:42:0a:00:01:06",
                "IPv4Address": "10.0.1.6/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4098"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "69676ae46ab9",
                "IP": "10.0.2.63"
            },
```

```
            {
                "Name": "1199cab4a6dd",
                "IP": "10.0.2.62"
            },
            {
                "Name": "d058d363197d",
                "IP": "10.0.2.64"
            }
        ]
    }
]
```

Now remove the **my-nginx** service as well as the two overlay networks **nginx-net** and **nginx-net-2** :

```
root@manager:~# docker service rm my-nginx
my-nginx

root@manager:~# docker network rm nginx-net nginx-net-2
nginx-net
nginx-net-2
```

1.4 - DNS container discovery

The Docker daemon runs an embedded DNS server at address 127.0.0.11 that enables name resolution in a custom network. If this server is unable to perform the resolution, it transfers the request to any external server defined in the container.

For **DNS container discovery** to work, the following ports must be open on the nodes:

- 2377/tcp
- 7946/tcp
- 7946/udp
- 4789/udp

Now create the overlay network **test-net** :

```
root@manager:~# docker network create --driver=overlay --attachable test-net
hrs25w4l951kkickhj6262mjg
```

> **Important**: Note that the **NETWORK-ID** here is **hrs25w4l951kkickhj6262mjg**.

On the Manager, start an interactive container called **alpine1** that connects to the **test-net** network:

```
root@manager:~# docker run -it --name alpine1 --network test-net alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
89d9c30c1d48: Pull complete
Digest: sha256:c19173c5ada610a5989151111163d28a67368362762534d8a8121ce95cf2bd5a
Status: Downloaded newer image for alpine:latest
/ #
```

List the networks available on **Worker1** :

```
root@worker1:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
3fe43b514f9d        bridge              bridge              local
ee22b3e623ca        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
r8htcvc8oxmz        ingress             overlay             swarm
de563e30d473        none                null                local
```

> **Important**: Note that the **test-net** network has not been created.

Now start a **alpine2** container on **Worker1** :

```
root@worker1:~# docker run -dit --name alpine2 --network test-net alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
89d9c30c1d48: Pull complete
Digest: sha256:c19173c5ada610a5989151111163d28a67368362762534d8a8121ce95cf2bd5a
Status: Downloaded newer image for alpine:latest
5734e84cd460cdd33ce90970d98a96837a0305832a86fc4d86be38aecf51b23b
```

Enter the **docker network ls** command on **Worker1** :

```
root@worker1:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
3fe43b514f9d        bridge              bridge              local
ee22b3e623ca        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
r8htcvc8oxmz        ingress             overlay             swarm
de563e30d473        none                null                local
hrs25w4l951k        test-net            overlay             swarm
```

Important: Note that the **test-net** network, having the same **NETWORK ID**, was automatically created when the **alpine2** container was created.

List the networks available on **Worker2**:

```
root@worker2:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
ff7308310f60        bridge              bridge              local
0ce1d8369c29        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
```

```
r8htcvc8oxmz        ingress             overlay             swarm
de563e30d473        none                null                local
```

Important: Note that the **test-net** network has not been created.

Attach to the **alpine2** container on **Worker1** and try to contact the **alpine1** container :

```
root@worker1:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
NAMES
ce9097b864dc        alpine              "/bin/sh"           23 minutes ago      Up 23 minutes
alpine2

root@worker1:~# docker attach alpine2
/ #

/ # ping -c 2 alpine1
PING alpine1 (10.0.2.2): 56 data bytes
64 bytes from 10.0.2.2: seq=0 ttl=64 time=1.874 ms
64 bytes from 10.0.2.2: seq=1 ttl=64 time=1.669 ms

--- alpine1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 1.669/1.771/1.874 ms
/ #
```

Return to the **Manager** VM and try to contact the **alpine2** container from the **alpine1** container:

```
root@manager:~# docker attach alpine1
/ #
```

```
/ # ping -c 2 alpine2
PING alpine2 (10.0.0.4): 56 data bytes
64 bytes from 10.0.0.4: seq=0 ttl=64 time=0.666 ms
64 bytes from 10.0.0.4: seq=1 ttl=64 time=1.239 ms

--- alpine2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.666/0.952/1.239 ms
/ #
```

Then create the **alpine3** container on the **Worker2** try to contact the **alpine1** container :

```
root@worker2:~# docker run -it --rm --name alpine3 --network test-net alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
c9b1b535fdd9: Pull complete
Digest: sha256:ab00606a42621fb68f2ed6ad3c88be54397f981a7b70a79db3d1172b11c4367d
Status: Downloaded newer image for alpine:latest
/ #

/ # ping -c 2 alpine1
PING alpine1 (10.0.2.2): 56 data bytes
64 bytes from 10.0.2.2: seq=0 ttl=64 time=0.642 ms
64 bytes from 10.0.2.2: seq=1 ttl=64 time=1.684 ms

--- alpine1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.642/1.163/1.684 ms
/ # exit
```

Now stop the **alpine2** container on **Worker1** :

```
root@worker1:~# docker container stop alpine2
```

```
alpine2
```

Enter the **docker network ls** command:

```
root@worker1:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
3bb80f391804        bridge              bridge              local
ee22b3e623ca        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
r8htcvc8oxmz        ingress             overlay             swarm
de563e30d473        none                null                local
```

Important: Note that the **test-net** network has been removed.

Delete the **alpine2** container:

```
root@worker1:~# docker container rm alpine2
alpine2
```

Stop the **alpine1** container and delete the **test-net** network on **Manager**:

```
/ # exit

root@manager:~# docker container stop alpine1
alpine1

root@manager:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
a604e7db6f95        bridge              bridge              local
d4c9b0c9437a        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
```

```
jxu667wzmj2u        ingress             overlay             swarm
de563e30d473        none                null                local
518l09lcjhsp        test-net            overlay             swarm

root@manager:~# docker network rm test-net
test-net
```

1.5 - Creating a Custom Network overlay

It is possible to create a custom overlay network. In this case, the existing ingress network must be deleted:

```
root@manager:~# docker network rm ingress
WARNING! Before removing the routing-mesh network, make sure all the nodes in your swarm run the same docker
engine version. Otherwise, removal may not be effective and functionality of newly create ingress networks will
be impaired.
Are you sure you want to continue? [y/N] y
ingress
```

Next, create your custom network:

```
root@manager:~# docker network create --driver overlay --ingress --subnet=10.11.0.0/16 --gateway=10.11.0.2 --opt
com.docker.network.driver.mtu=1200 my-ingress
44ozn3vtg23zkksrvloxuulcl

root@manager:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
24be8a0f0ef5        bridge              bridge              local
d4c9b0c9437a        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
44ozn3vtg23z        my-ingress          overlay             swarm
de563e30d473        none                null                local
```

Create the **my-nginx** service again:

```
root@manager:~# docker service create --name my-nginx --publish target=80,published=80 --replicas=5 nginx
gp1iozmbi25dx3skn00m6suoz
overall progress: 5 out of 5 tasks
1/5: running   [==================================================>]
2/5: running   [==================================================>]
3/5: running   [==================================================>]
4/5: running   [==================================================>]
5/5: running   [==================================================>]
verify: Service converged

root@manager:~# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
gp1iozmbi25d        my-nginx            replicated          5/5                 nginx:latest        *:80->80/tcp

root@manager:~# docker service ps my-nginx
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE
ERROR               PORTS
upmbwmtr76cm        my-nginx.1          nginx:latest        worker1.i2tch.loc   Running             Running about
a minute ago
qz6p1li7zmef        my-nginx.2          nginx:latest        worker2.i2tch.loc   Running             Running about
a minute ago
me50mkhd11yk        my-nginx.3          nginx:latest        manager.i2tch.loc   Running             Running about
a minute ago
sctjud70ihkl        my-nginx.4          nginx:latest        worker1.i2tch.loc   Running             Running about
a minute ago
kql9qx3phb73        my-nginx.5          nginx:latest        worker2.i2tch.loc   Running             Running about
a minute ago
```

View information about the **my-nginx** service:

```
root@manager:~# docker service inspect my-nginx
[
```

```
    {
        "ID": "gp1iozmbi25dx3skn00m6suoz",
        "Version": {
            "Index": 230
        },
        "CreatedAt": "2019-10-28T14:49:33.6719228Z",
        "UpdatedAt": "2019-10-28T14:49:33.679624758Z",
        "Spec": {
            "Name": "my-nginx",
            "Labels": {},
            "TaskTemplate": {
                "ContainerSpec": {
                    "Image":
"nginx:latest@sha256:922c815aa4df050d4df476e92daed4231f466acc8ee90e0e774951b0fd7195a4",
                    "Init": false,
                    "StopGracePeriod": 10000000000,
                    "DNSConfig": {},
                    "Isolation": "default"
                },
                "Resources": {
                    "Limits": {},
                    "Reservations": {}
                },
                "RestartPolicy": {
                    "Condition": "any",
                    "Delay": 5000000000,
                    "MaxAttempts": 0
                },
                "Placement": {
                    "Platforms": [
                        {
                            "Architecture": "amd64",
                            "OS": "linux"
                        },
```

```
                    {
                        "OS": "linux"
                    },
                    {
                        "Architecture": "arm64",
                        "OS": "linux"
                    },
                    {
                        "Architecture": "386",
                        "OS": "linux"
                    },
                    {
                        "Architecture": "ppc64le",
                        "OS": "linux"
                    },
                    {
                        "Architecture": "s390x",
                        "OS": "linux"
                    }
                ]
            },
            "ForceUpdate": 0,
            "Runtime": "container"
        },
        "Mode": {
            "Replicated": {
                "Replicas": 5
            }
        },
        "UpdateConfig": {
            "Parallelism": 1,
            "FailureAction": "pause",
            "Monitor": 5000000000,
            "MaxFailureRatio": 0,
```

```
                "Order": "stop-first"
            },
            "RollbackConfig": {
                "Parallelism": 1,
                "FailureAction": "pause",
                "Monitor": 5000000000,
                "MaxFailureRatio": 0,
                "Order": "stop-first"
            },
            "EndpointSpec": {
                "Mode": "vip",
                "Ports": [
                    {
                        "Protocol": "tcp",
                        "TargetPort": 80,
                        "PublishedPort": 80,
                        "PublishMode": "ingress"
                    }
                ]
            }
        },
        "endpoint": {
            "Spec": {
                "Mode": "vip",
                "Ports": [
                    {
                        "Protocol": "tcp",
                        "TargetPort": 80,
                        "PublishedPort": 80,
                        "PublishMode": "ingress"
                    }
                ]
            },
            "Ports": [
```

```
                {
                    "Protocol": "tcp",
                    "TargetPort": 80,
                    "PublishedPort": 80,
                    "PublishMode": "ingress"
                }
            ],
            "VirtualIPs": [
                {
                    "NetworkID": "44ozn3vtg23zkksrvloxuulcl",
                    "Addr": "10.11.0.1/16"
                }
            ]
        }
    }
]
```

Now check that the containers are in the **my-ingress** network:

```
root@manager:~# docker inspect my-ingress
[
    {
        "Name": "my-ingress",
        "Id": "l11ucu5ufjfwwz6e0umtygdqy",
        "Created": "2020-03-10T11:02:38.278429829+01:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.11.0.0/16",
```

```
                "Gateway": "10.11.0.2"
            }
        ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": true,
    "ConfigFrom": {
        "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
        "6f0168ff5153b899af31098740de34997b12417ef7c0f3824938edf79b2bca7f": {
            "Name": "my-nginx.3.me50mkhd11ykwz7aj07znloh1",
            "EndpointID": "41531d43496f4723cb62cad1d57c5a088faebe79c430d04a1765022e31d8ae17",
            "MacAddress": "02:42:0a:0b:00:05",
            "IPv4Address": "10.11.0.5/16",
            "IPv6Address": ""
        },
        "my-ingress-sbox": {
            "Name": "my-ingress-endpoint",
            "EndpointID": "0205796eeb005ef77b3ea382fd1e72c312a58fd717b5a79ca6cacc7e090068e6",
            "MacAddress": "02:42:0a:0b:00:0a",
            "IPv4Address": "10.11.0.10/16",
            "IPv6Address": ""
        }
    },
    "Options": {
        "com.docker.network.driver.mtu": "1200",
        "com.docker.network.driver.overlay.vxlanid_list": "4100"
    },
    "Labels": {},
    "Peers": [
        {
```

```
                "Name": "9a00e8bc72fe",
                "IP": "10.0.2.62"
            },
            {
                "Name": "3ea669d48ca2",
                "IP": "10.0.2.64"
            },
            {
                "Name": "f30e39df1704",
                "IP": "10.0.2.63"
            }
        ]
    }
]
```

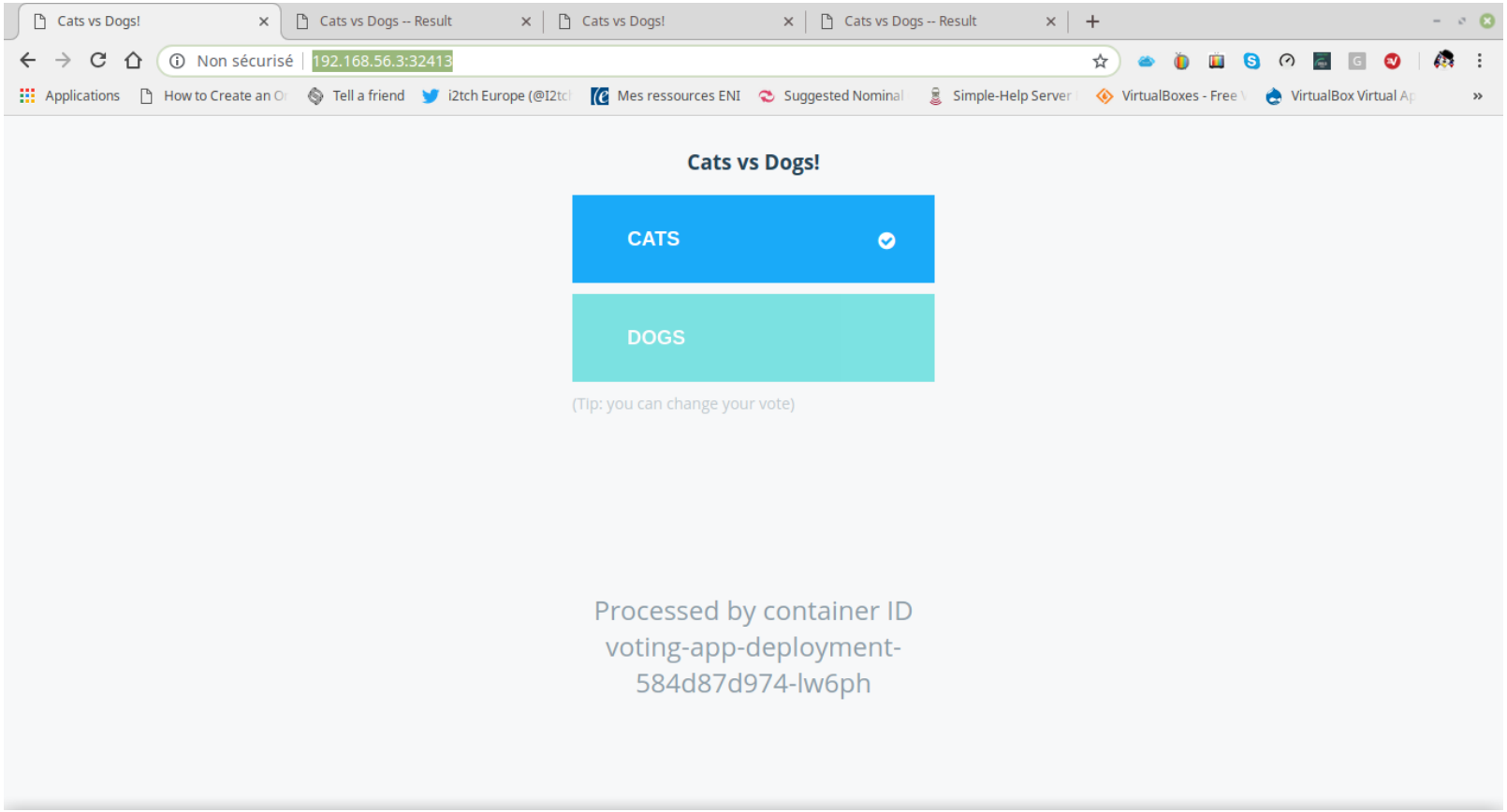
Now remove the **my-nginx** service:

```
root@manager:~# docker service rm my-nginx
my-nginx
```

LAB #2 - Managing a Microservices Architecture

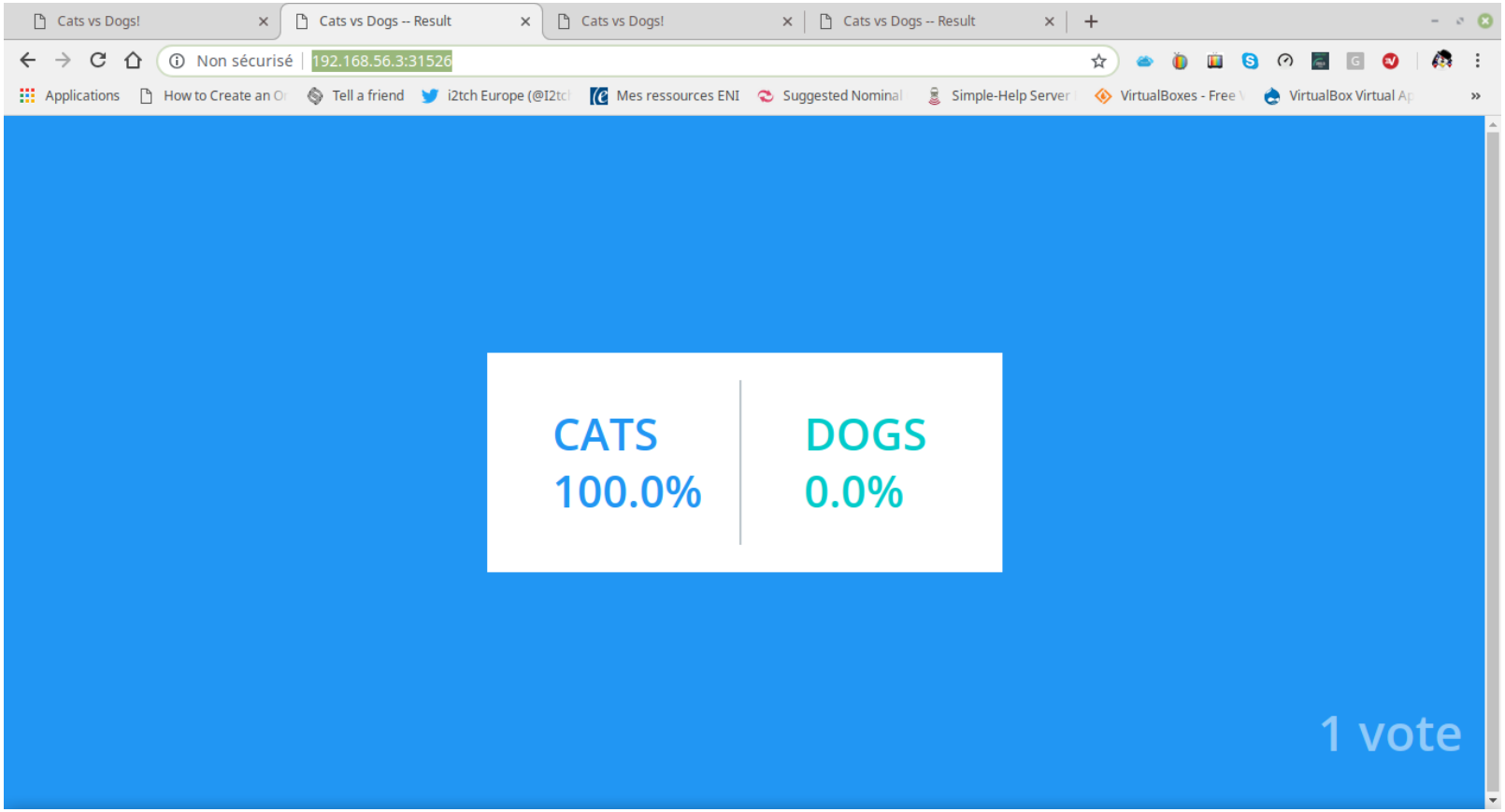
You are going to set up a simple application, called **demo-voting-app** and developed by Docker, in the form of microservices:

In this application, the **voting-app** container allows you to vote for **cats** or **dogs**. This application runs under Python and provides an HTML interface:



During the vote, the result of the vote is stored in **Redis** in an in-memory database. The result is then passed to the **Worker** container, which runs under .NET and updates the persistent database in the **db** container, which runs under PostgreSQL.

The **result-app** application running in NodeJS then reads the table from the PostgreSQL database and displays the result in HTML form:



}

2.1 - Setting up with Docker Swarm using Overlay networks

This application can be set up using docker swarm with the **docker stack** command. A **stack** is a group of services. First, check the state of the Swarm:

```
root@manager:~# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
ENGINE VERSION
b85hxlixbr1mh1txd1hrfe4us *   manager.i2tch.loc   Ready               Active              Leader
```

```
19.03.4
4sui75vvdhmet4qvt0zbvzlzl     worker1.i2tch.loc   Ready               Active
19.03.4
lbjtg5o9kw3x6xg7frm07jfuw     worker2.i2tch.loc   Ready               Active
19.03.4
```

Now create the **docker-stack.yml** file:

```
root@manager:~# vi docker-stack.yml

root@manager:~# cat docker-stack.yml
version: "3"
services:

  redis:
    image: redis:alpine
    networks:
      - frontend
    deploy:
      replicas: 1
      update_config:
        parallelism: 2
        delay: 10s
      restart_policy:
        condition: on-failure
  db:
    image: postgres:9.4
    environment:
      POSTGRES_USER: "postgres"
      POSTGRES_PASSWORD: "postgres"
    volumes:
      - db-data:/var/lib/postgresql/data
    networks:
      - backend
```

```
    deploy:
      placement:
        constraints: [node.role == manager]
  vote:
    image: dockersamples/examplevotingapp_vote:before
    ports:
      - 5000:80
    networks:
      - frontend
    depends_on:
      - redis
    deploy:
      replicas: 2
      update_config:
        parallelism: 2
      restart_policy:
        condition: on-failure
  result:
    image: dockersamples/examplevotingapp_result:before
    ports:
      - 5001:80
    networks:
      - backend
    depends_on:
      - db
    deploy:
      replicas: 1
      update_config:
        parallelism: 2
        delay: 10s
      restart_policy:
        condition: on-failure

  worker:
```

```
    image: dockersamples/examplevotingapp_worker
    networks:
      - frontend
      - backend
    depends_on:
      - db
      - redis
    deploy:
      mode: replicated
      replicas: 1
      labels: [APP=VOTING]
      restart_policy:
        condition: on-failure
        delay: 10s
        max_attempts: 3
        window: 120s
      placement:
        constraints: [node.role == manager]

  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    stop_grace_period: 1m30s
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]

networks:
  frontend:
  backend:
```

```
volumes:
  db-data:
```

In this file we can see 6 services, **redis**, **db**, **vote**, **result**, **worker** and **visualizer**. The first 5 services together form the application, while the **visualizer** service will allow us to see how the application has been set up.

First, look at the **deploy** key for the **worker** service:

```
...
    deploy:
      mode: replicated
      replicas: 1
      labels: [APP=VOTING]
      restart_policy:
        condition: on-failure
        delay: 10s
        max_attempts: 3
        window: 120s
      placement:
        constraints: [node.role == manager]
...
```

The **deploy** key is used to specify options when deploying the service :

- **mode** - There are two types of service. **Replicated** where we specify the number of instances that Docker should set up on **available** hosts depending on the value of **replicas** and **Global** which implies that Docker will start an instance of the service on each host each time a host becomes available.

- **replicas** - specifies the number of replicas
- **restart_policy** specifies what happens if the service is stopped. In the case above, docker will try to restart the service **3** times (**max_attempts**) at **10** second intervals (**delay**), waiting **120** seconds (**window**) each time to see if the service has actually restarted,
- **placement** - specifies where the service should be started.

Now deploy the stack:

```
root@manager:~# docker stack deploy -c docker-stack.yml app
Creating network app_backend
Creating network app_default
Creating network app_frontend
Creating service app_worker
Creating service app_visualizer
Creating service app_redis
Creating service app_db
Creating service app_vote
Creating service app_result
```

Important - Note that each network and service has the application name **app** as its prefix.

Now check the status of the stack:

```
root@manager:~# docker stack ls
NAME                SERVICES            ORCHESTRATOR
app                 6                   Swarm
```

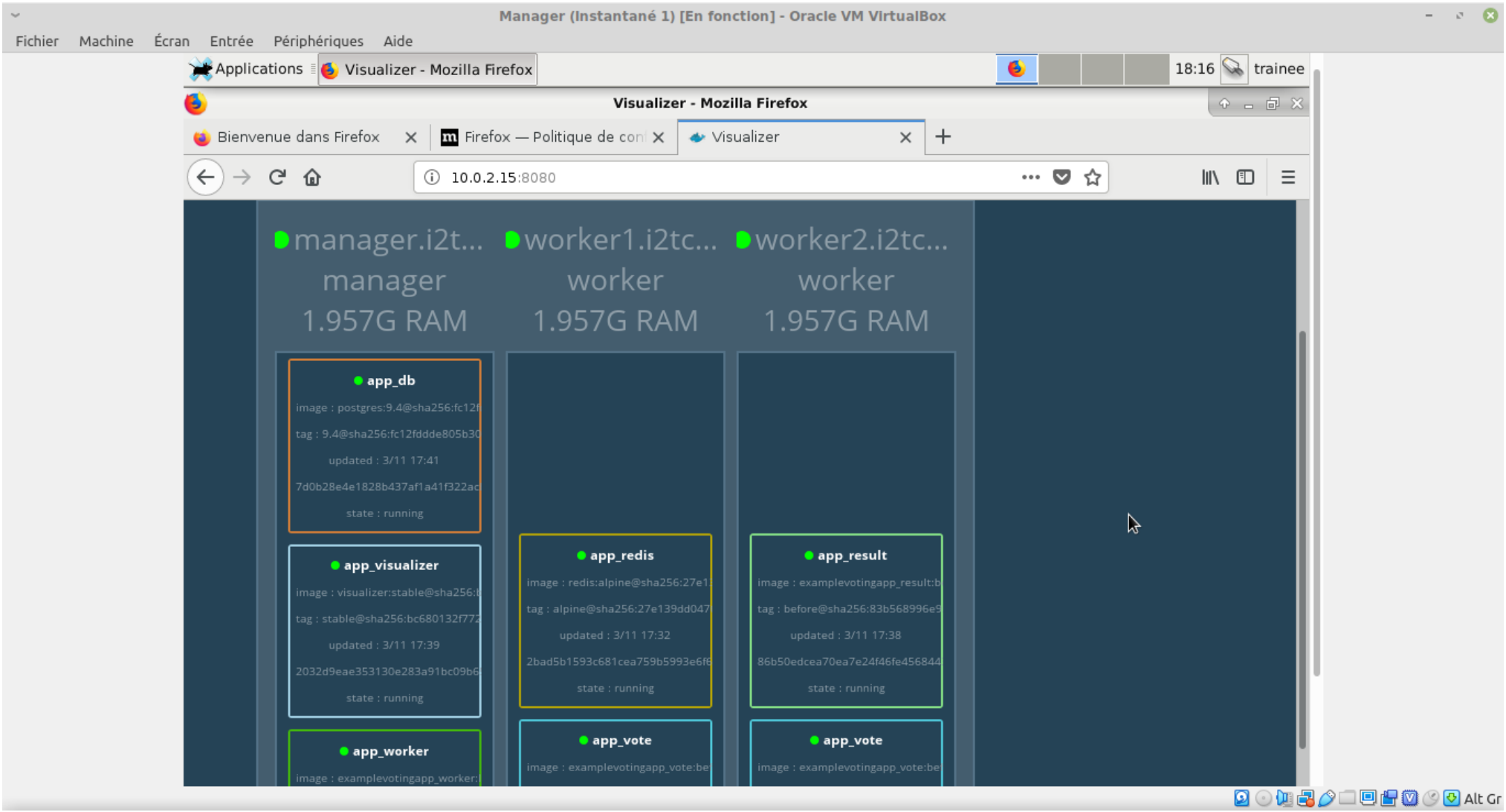
Then check the status of the services:

```
root@manager:~# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE
PORTS
d0i4ac4fshw0        app_db              replicated          1/1                 postgres:9.4
funp5kboyip1        app_redis           replicated          1/1                 redis:alpine
dpdkc49oj671        app_result          replicated          1/1
dockersamples/examplevotingapp_result:before   *:5001->80/tcp
```

```
vrkahv38v5mn        app_visualizer      replicated          1/1                 dockersamples/visualizer:stable
*:8080->8080/tcp
t4u16cpdrx21        app_vote            replicated          2/2
dockersamples/examplevotingapp_vote:before     *:5000->80/tcp
so40eljbcviy        app_worker          replicated          1/1
dockersamples/examplevotingapp_worker:latest
```

> **Important**: Note that the **visualizer** service configuration has exposed port **8080**. This way, this service is available on port 8080 of every node in the swarm.

Return to the Apache Guacamole window in **your** computer's browser. Click on the **Debian11_10.0.2.46_VNC** connection. Then launch an Internet browser. Go to the URL http://10.0.2.62:8080 and consult the **visualizer** service:



As you can see, according to the **docker-stack.yml** file, the three containers **db**, **worker** and **visualizer** have been started on the manager node.

Go back to your SSH connection and check the status of the networks in the three nodes:

```
root@manager:~# docker network ls
```

```
NETWORK ID          NAME                DRIVER              SCOPE
sw489bb290zb        app_backend         overlay             swarm
smuxoglyudpo        app_default         overlay             swarm
lfizui95od90        app_frontend        overlay             swarm
24be8a0f0ef5        bridge              bridge              local
d4c9b0c9437a        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
x7l4mk4ldb75        my-ingress          overlay             swarm
de563e30d473        none                null                local
```

Important: Note that the three networks created are of type **overlay**.

```
root@worker1:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
qhysvpoolsw0        app_frontend        overlay             swarm
f9a69d02de3b        bridge              bridge              local
ee22b3e623ca        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
x7l4mk4ldb75        my-ingress          overlay             swarm
de563e30d473        none                null                local
```

Important: Note that only the **app_frontend** network has been created in **worker1**.

```
root@worker2:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
s4gbgi4isp1i        app_backend         overlay             swarm
qhysvpoolsw0        app_frontend        overlay             swarm
```

```
0e6c118bf3fd        bridge              bridge              local
0ce1d8369c29        docker_gwbridge     bridge              local
f3cb3bc3c581        host                host                local
x7l4mk4ldb75        my-ingress          overlay             swarm
de563e30d473        none                null                local
```

Important: Note that the two networks **app_frontend** and **app_backend** were created in **worker2**.

View information about the **app_backend** network:

```
root@manager:~# docker inspect app_backend
[
    {
        "Name": "app_backend",
        "Id": "s4gbgi4isp1i5wjpgnf4uci2a",
        "Created": "2019-11-03T17:30:56.822222239+01:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.3.0/24",
                    "Gateway": "10.0.3.1"
                }
            ]
        },
        "Internal": false,
```

```
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "7d0b28e4e1828b437af1a41f322acb5cf19afc25c42303986dd2c7b4d5aea568": {
                "Name": "app_db.1.s6g6w47k532rvaeoyske8as9i",
                "EndpointID": "c26795c837f6dc736a3f9be34525ae505e9db6381a2144bb62087b3ee6c7ff25",
                "MacAddress": "02:42:0a:00:03:03",
                "IPv4Address": "10.0.3.3/24",
                "IPv6Address": ""
            },
            "ef7227281d297b001bb0f60ac81a0c9926e8fb663a7f43eb201cced632dc5564": {
                "Name": "app_worker.1.38kniuqoe1vfyonwdcytlhpqo",
                "EndpointID": "990065eec5062ff159e82bc1e4666fd098d5597439221995af4f01040ab24599",
                "MacAddress": "02:42:0a:00:03:09",
                "IPv4Address": "10.0.3.9/24",
                "IPv6Address": ""
            },
            "lb-app_backend": {
                "Name": "app_backend-endpoint",
                "EndpointID": "913845cbab9a6c3011eaaa87fcc66f10268b5e11554be9f1a20b1078f7b9b8a4",
                "MacAddress": "02:42:0a:00:03:04",
                "IPv4Address": "10.0.3.4/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4101"
        },
        "Labels": {
            "com.docker.stack.namespace": "app"
```

```
        },
        "Peers": [
            {
                "Name": "377986fb7d5a",
                "IP": "10.0.2.62"
            },
            {
                "Name": "5cc4b863da9f",
                "IP": "10.0.2.64"
            }
        ]
    }
]
```

Important: Note that the network is **10.0.3.0/24** and the gateway **10.0.3.1**.

View information about the **app_frontend** network:

```
root@manager:~# docker inspect app_frontend
[
    {
        "Name": "app_frontend",
        "Id": "qhysvpoolsw0318gsubbvd3rx",
        "Created": "2019-11-03T17:31:27.354293132+01:00",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
```

```
            "Config": [
                {
                    "Subnet": "10.0.2.0/24",
                    "Gateway": "10.0.2.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "ef7227281d297b001bb0f60ac81a0c9926e8fb663a7f43eb201cced632dc5564": {
                "Name": "app_worker.1.38kniuqoe1vfyonwdcytlhpqo",
                "EndpointID": "3fad9773920412464b6aeee59f8d9ffc5aac2e937b88dc384268591cf7d21fb9",
                "MacAddress": "02:42:0a:00:02:0a",
                "IPv4Address": "10.0.2.10/24",
                "IPv6Address": ""
            },
            "lb-app_frontend": {
                "Name": "app_frontend-endpoint",
                "EndpointID": "343887373c1f92ac08b271ee52dd160089eeed7cad13b7924e438919254b6442",
                "MacAddress": "02:42:0a:00:02:0b",
                "IPv4Address": "10.0.2.11/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4100"
        },
        "Labels": {
```

```
            "com.docker.stack.namespace": "app"
        },
        "Peers": [
            {
                "Name": "0e21ba1bbfab",
                "IP": "10.0.2.63"
            },
            {
                "Name": "5cc4b863da9f",
                "IP": "10.0.2.64"
            },
            {
                "Name": "377986fb7d5a",
                "IP": "10.0.2.62"
            }
        ]
    }
]
```

> **Important**: Note that the network is **10.0.2.0/24** and the gateway **10.0.2.1**.

Check the network information **app_default** :

```
root@manager:~# docker inspect app_default
[
    {
        "Name": "app_default",
        "Id": "z62t49w18wl2mrboa92tunrhq",
        "Created": "2019-10-28T17:22:44.724040846+01:00",
        "Scope": "swarm",
```

```
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.1.0/24",
                    "Gateway": "10.0.1.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "2032d9eae353130e283a91bc09b65b4a84b7e8f5602a466f4ea1bd9c64e964dc": {
                "Name": "app_visualizer.1.nbf78cn5g37dmu0fwrxt7kbrg",
                "EndpointID": "d5274ff057c9d9af0288efb7f9bfed3a5ca1b3e656e265ad343f52c0b1c161f5",
                "MacAddress": "02:42:0a:00:01:03",
                "IPv4Address": "10.0.1.3/24",
                "IPv6Address": ""
            },
            "lb-app_default": {
                "Name": "app_default-endpoint",
                "EndpointID": "6afb8909d94528633e4150054311f645790280b1ab1c686c43e865ba97ec3df9",
                "MacAddress": "02:42:0a:00:01:04",
                "IPv4Address": "10.0.1.4/24",
                "IPv6Address": ""
            }
```

```
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4099"
        },
        "Labels": {
            "com.docker.stack.namespace": "app"
        },
        "Peers": [
            {
                "Name": "377986fb7d5a",
                "IP": "10.0.2.62"
            }
        ]
    }
]
```

Important: Note that the network is **10.0.1.0/24** and the gateway **10.0.1.1**.

Schematically, the implementation of the application in the Swarm is as follows:

Lastly, delete the stack:

```
root@manager:~# docker stack ls
NAME                SERVICES            ORCHESTRATOR
app                 6                   Swarm

root@manager:~# docker stack rm app
Removing service app_db
```

```
Removing service app_redis
Removing service app_result
Removing service app_visualizer
Removing service app_vote
Removing service app_worker
Removing network app_frontend
Removing network app_backend
Removing network app_default

root@manager:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS                 NAMES
d02c6115724c        alpine              "/bin/sh"                6 days ago          Exited (0) 6 days ago
alpine1
```
