

Out of the box, Docker creates three networks:

- bridge - An automatically generated network with a subnet and a gateway within a host
- host - Allows a container to attach to the host's network
- none - A container-specific network stack that lacks a network interface.

Docker connects to the bridge network by default; this allows deployed containers to be seen on your network. Let's see how we can manage those networks, create a new network, and then deploy a container on our new network.

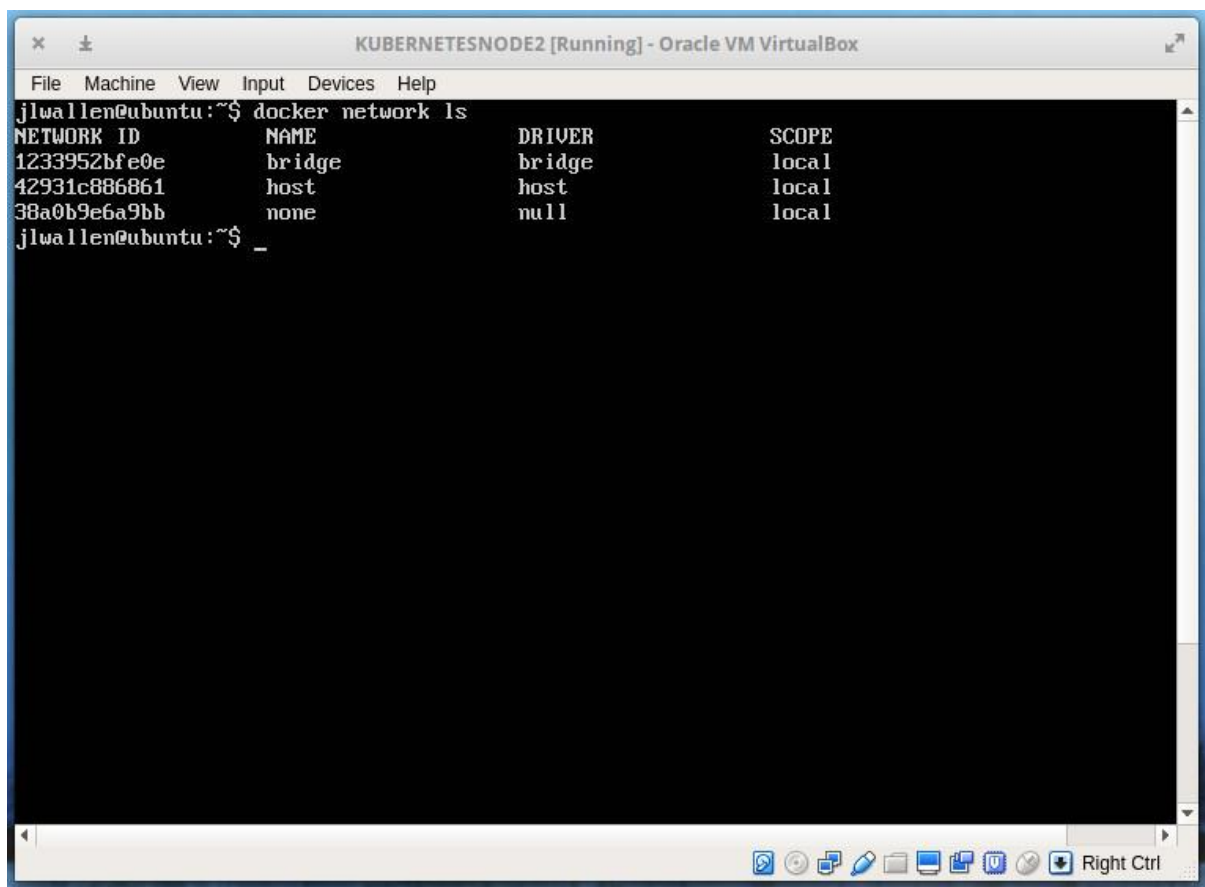
## Viewing networks

To view the current list of Docker networks, issue the command:

```
docker network ls
```

The above command will list out the Docker networks (**Figure A**).

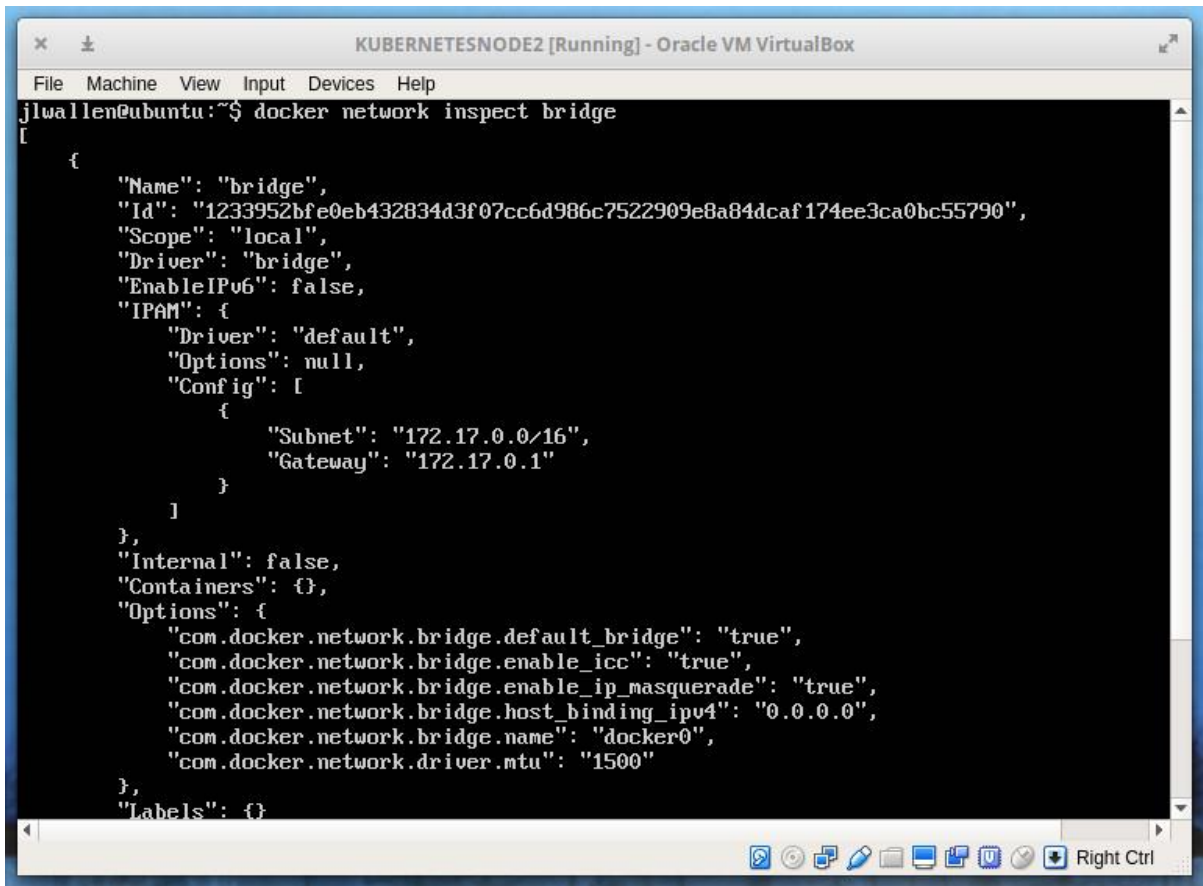
**Figure A**



Our default Docker networks.

You can get more information on a particular network, by issuing the command `docker network inspect NAME` (Where NAME is the name of the network you want to view). If you want to view details on the bridge network, that command would be `docker network inspect bridge`. The output of that command would give you all the information you need about that network (**Figure B**).

**Figure B**

A screenshot of a terminal window titled "KUBERNETESNODE2 [Running] - Oracle VM VirtualBox". The terminal shows the command `jlwallen@ubuntu:~$ docker network inspect bridge` and its output, which is a JSON object describing the 'bridge' network. The output includes details like Name, Id, Scope, Driver, EnableIPv6, IPAM configuration (Subnet: 172.17.0.0/16, Gateway: 172.17.0.1), Internal status, Containers, Options, and Labels.

```
j1wallen@ubuntu:~$ docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "1233952bfe0eb432834d3f07cc6d986c7522909e8a84dcaf174ee3ca0bc55790",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
    "Containers": {},
    "Options": {
      "com.docker.network.bridge.default_bridge": "true",
      "com.docker.network.bridge.enable_icc": "true",
      "com.docker.network.bridge.enable_ip_masquerade": "true",
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
      "com.docker.network.bridge.name": "docker0",
      "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {}
  }
]
```

The bridge network information listing.

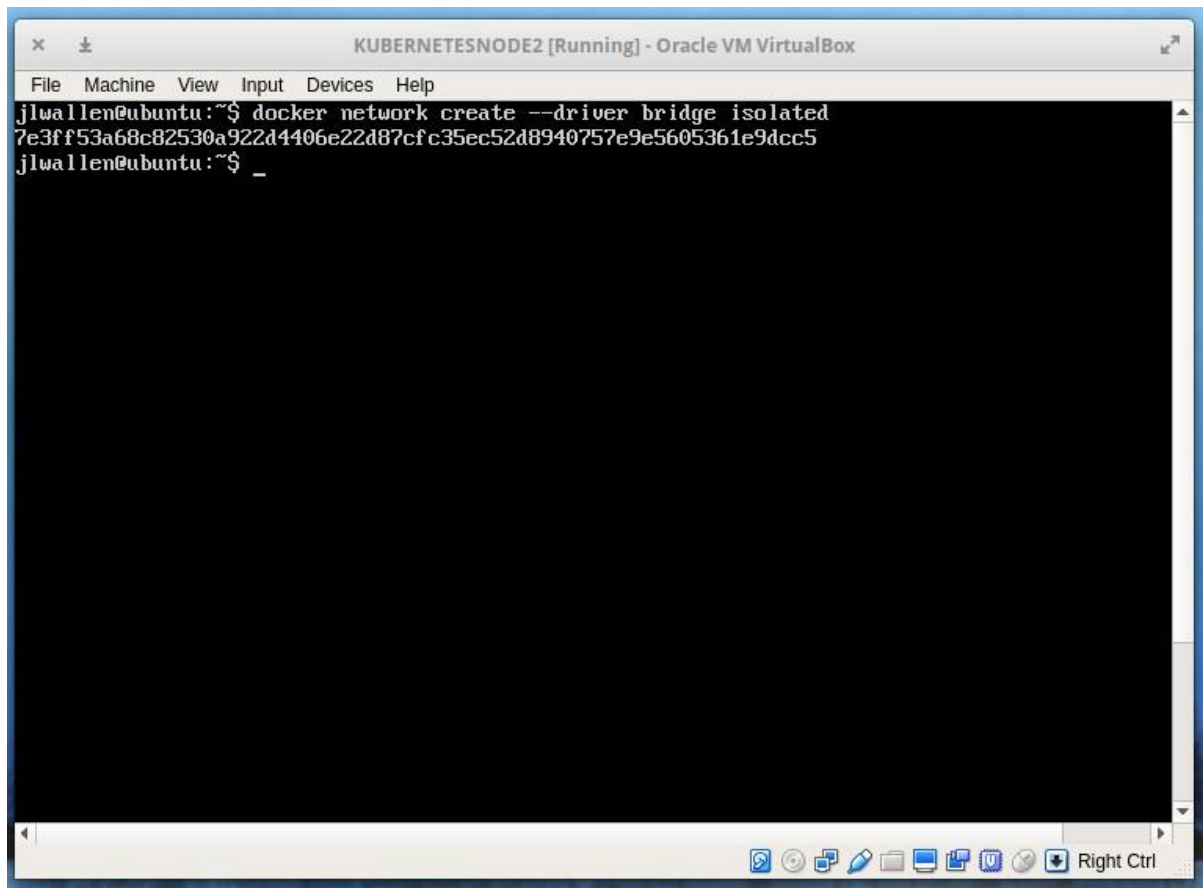
## Creating a new network

I'm going to demonstrate how to create a bridge network and then show you how to deploy a container on that network. We'll create a network called `isolated`. The creation of this network can be achieved with a single command:

```
docker network create --driver bridge isolated
```

The output of that command will be a long string of characters that represents the ID of that newly-created network (**Figure C**).

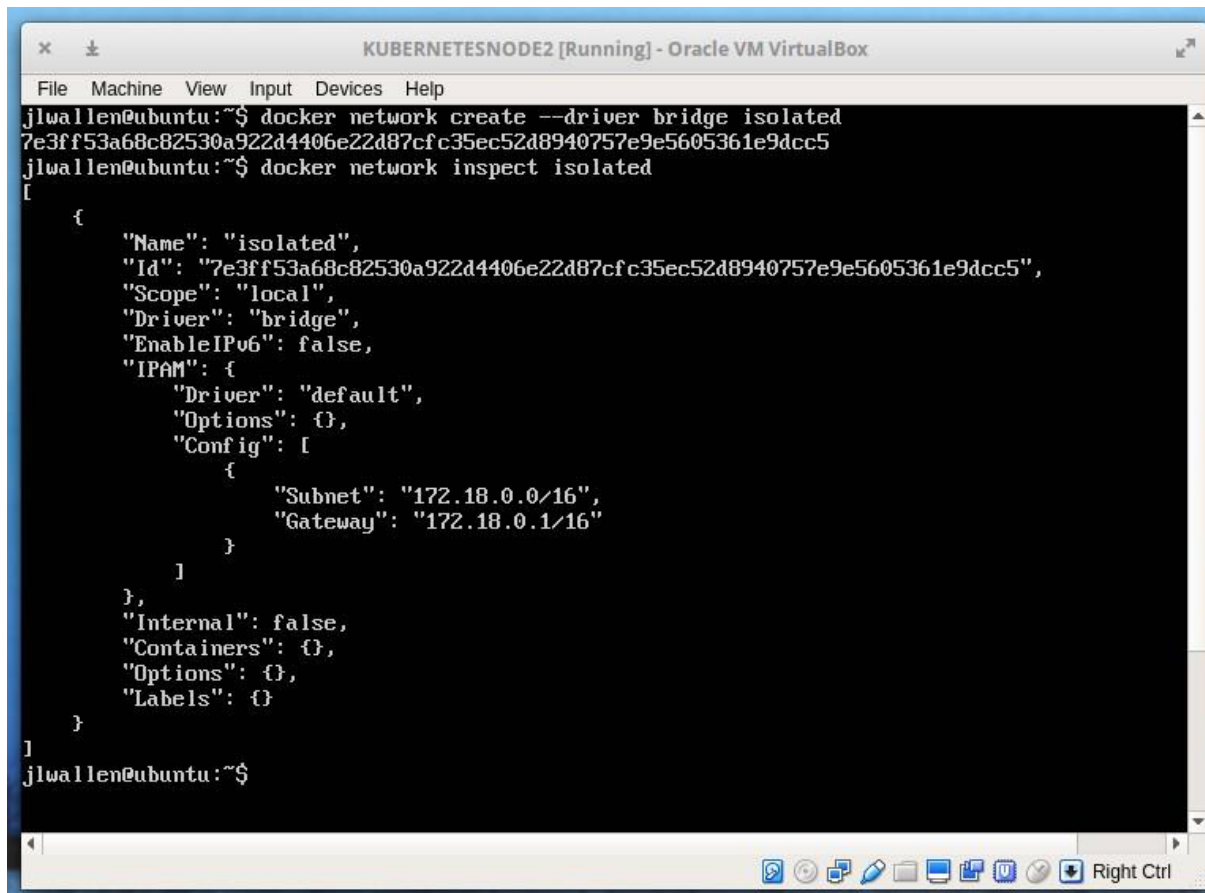
**Figure C**



A newly created network.

Run the inspect command on that newly created network with the command `docker network inspect isolated` to see that our new network has been automatically given its own subnet and gateway (**Figure D**).

**Figure D**

A screenshot of a terminal window titled "KUBERNETESNODE2 [Running] - Oracle VM VirtualBox". The terminal shows a user at the "jllwallen@ubuntu" prompt. The user enters the command "docker network create --driver bridge isolated" followed by its long ID. Then, they enter "docker network inspect isolated", which returns a JSON object describing the network configuration. The JSON shows the network name as "isolated", scope as "local", driver as "bridge", and IPAM configuration with a subnet of "172.18.0.0/16" and gateway of "172.18.0.1/16". The terminal window has a menu bar (File, Machine, View, Input, Devices, Help) and a toolbar at the bottom with icons for various actions and a "Right Ctrl" button.

```
jllwallen@ubuntu:~$ docker network create --driver bridge isolated
7e3ff53a68c82530a922d4406e22d87cfc35ec52d8940757e9e5605361e9dcc5
jllwallen@ubuntu:~$ docker network inspect isolated
[
  {
    "Name": "isolated",
    "Id": "7e3ff53a68c82530a922d4406e22d87cfc35ec52d8940757e9e5605361e9dcc5",
    "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": {}
  }
]
jllwallen@ubuntu:~$
```

Our new network, named isolated.

But what if you want to create a network with a specific subnet and gateway? That's possible as well. Let's say you want to create a network with a subnet of 192.168.2.0/24, a gateway of 192.168.2.10, and the name `new_subnet`. The command for this would be:

```
docker network create --driver=bridge --subnet=192.168.2.0/24 --gateway=192.168.2.10 new_subnet
```

Once created, inspect the network, with the command `docker network inspect new_subnet` to see the results (**Figure E**).

**Figure E**

```
j1wallen@ubuntu:~$ docker network create --driver bridge --subnet=192.168.2.0/24 --gateway=192.168.2.10 new_subnet
b19e0fc5e08cb8dc58a8c5db8c5224fc04719a243bbf1a694cb11bb3958797b4
j1wallen@ubuntu:~$ docker network inspect new_subnet
[
  {
    "Name": "new_subnet",
    "Id": "b19e0fc5e08cb8dc58a8c5db8c5224fc04719a243bbf1a694cb11bb3958797b4",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "192.168.2.0/24",
          "Gateway": "192.168.2.10"
        }
      ]
    },
    "Internal": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]
```

Our new network with a specific subnet and gateway.

## Attaching a container to a networking

Let's attach a container to our newly created network. Say you've already pulled down the nginx image and want to launch a container, named docker-nginx, attached to the isolated network. To do this, the command would look like:

```
docker run --network=isolated -itd --name=docker-nginx nginx
```

If you now run the command `docker network inspect isolated`, you'll see that the container has been attached (**Figure F**).

**Figure F**

```

Digest: sha256:788fa27763db6d69ad3444e8ba72f947df9e7e163bad7c1f5614f8fd27a311c3
Status: Downloaded newer image for nginx:latest
jlwallen@ubuntu:~$ docker run --network=isolated -itd --name=docker-nginx nginx
d610914dca023b4c56f80e3e73780711018209fa7580d211061034aced85d947
jlwallen@ubuntu:~$ docker network inspect isolated
[
  {
    "Name": "isolated",
    "Id": "7e3ff53a68c82530a922d4406e22d87cfc35ec52d8940757e9e5605361e9dcc5",
    "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": {
      "d610914dca023b4c56f80e3e73780711018209fa7580d211061034aced85d947": {
        "Name": "docker-nginx",
        "EndpointID": "01dd6ab77298f799716b9cecf9a971c175d7c4569789a4ee5cc23e9ce2600d4cb",
        "MacAddress": "02:42:ac:12:00:02",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
jlwallen@ubuntu:~$ _

```

Our container has been attached.

Any other container you create on this network would be able to automatically connect to one another. So if you create a database container on isolated, it would be available for any other container on the same network.