## NAME

docker-network-create - Create a network

## SYNOPSIS

**docker network create [OPTIONS] NETWORK**

## DESCRIPTION

Creates a new network. The `DRIVER` accepts `bridge` or `overlay` which are the built-in network drivers. If you have installed a third party or your own custom network driver you can specify that `DRIVER` here also. If you don't specify the `--driver` option, the command automatically creates a `bridge` network for you. When you install Docker Engine it creates a `bridge` network automatically. This network corresponds to the `docker0` bridge that Engine has traditionally relied on. When you launch a new container with `docker run` it automatically connects to this bridge network. You cannot remove this default bridge network but you can create new ones using the `network create` command.

```
$ docker network create -d bridge my-bridge-network
```

Bridge networks are isolated networks on a single Engine installation. If you want to create a network that spans multiple Docker hosts each running an Engine, you must create an `overlay` network. Unlike `bridge` networks overlay networks require some pre-existing conditions before you can create one. These conditions are:

- Access to a key-value store. Engine supports Consul, Etcd, and Zookeeper (Distributed store) key-value stores.
- A cluster of hosts with connectivity to the key-value store.
- A properly configured Engine `daemon` on each host in the cluster.

The `dockerd` options that support the `overlay` network are:

- `--cluster-store`
- `--cluster-store-opt`
- `--cluster-advertise`

To read more about these options and how to configure them, see "*Get started with multi-host network*" ⟨https://docs.docker.com/engine/userguide/networking/get-started-overlay/⟩.

It is also a good idea, though not required, that you install Docker Swarm on to manage the cluster that makes up your network. Swarm provides sophisticated discovery and server management that can assist your implementation.

Once you have prepared the `overlay` network prerequisites you simply choose a Docker host in the cluster and issue the following to create the network:

```
$ docker network create -d overlay my-multihost-network
```

Network names must be unique. The Docker daemon attempts to identify naming conflicts but this is not guaranteed. It is the user's responsibility to avoid name conflicts.

## Connect containers

When you start a container use the `--network` flag to connect it to a network. This adds the `busybox` container to the `mynet` network.

```
$ docker run -itd --network=mynet busybox
```

If you want to add a container to a network after the container is already running use the `docker network connect` subcommand.

You can connect multiple containers to the same network. Once connected, the containers can communicate using only another container's IP address or name. For `overlay` networks or custom plugins that support multi-host connectivity, containers connected to the same multi-host network but launched from different Engines can also communicate in this way.

You can disconnect a container from a network using the `docker network disconnect` command.

## Specifying advanced options

When you create a network, Engine creates a non-overlapping subnetwork for the network by default. This subnetwork is not a subdivision of an existing network. It is purely for ip-addressing purposes. You can override this default and specify subnetwork values directly using the `--subnet` option. On a `bridge` network you can only create a single subnet:

```
$ docker network create -d bridge --subnet=192.168.0.0/16 br0
```

Additionally, you also specify the `--gateway` `--ip-range` and `--aux-address` options.

```
$ docker network create \
  --driver=bridge \
  --subnet=172.28.0.0/16 \
  --ip-range=172.28.5.0/24 \
  --gateway=172.28.5.254 \
  br0
```

If you omit the `--gateway` flag the Engine selects one for you from inside a preferred pool. For `overlay` networks and for network driver plugins that support it you can create multiple subnetworks.

```
$ docker network create -d overlay \
  --subnet=192.168.0.0/16 \
  --subnet=192.170.0.0/16 \
  --gateway=192.168.0.100 \
  --gateway=192.170.0.100 \
  --ip-range=192.168.1.0/24 \
  --aux-address="my-router=192.168.1.5" --aux-address="my-switch=192.168.1.6" \
  --aux-address="my-printer=192.170.1.5" --aux-address="my-nas=192.170.1.6" \
  my-multihost-network
```

Be sure that your subnetworks do not overlap. If they do, the network create fails and Engine returns an error.

**Network internal mode**

By default, when you connect a container to an `overlay` network, Docker also connects a bridge network to it to provide external connectivity. If you want to create an externally isolated `overlay` network, you can specify the `--internal` option.

**Network ingress mode**

You can create the network which will be used to provide the routing-mesh in the swarm cluster. You do so by specifying `--ingress` when creating the network. Only one ingress network can be created at the time. The network can be removed only if no services depend on it. Any option available when creating an overlay network is also available when creating the ingress network, besides the `--attachable` option.

```
$ docker network create -d overlay \
  --subnet=10.11.0.0/16 \
  --ingress \
  --opt com.docker.network.mtu=9216 \
  --opt encrypted=true \
  my-ingress-network
```

**Run services on predefined networks**

You can create services on the predefined docker networks `bridge` and `host`.

```
$ docker service create --name my-service \
  --network host \
  --replicas 2 \
  busybox top
```

**Swarm networks with local scope drivers**

You can create a swarm network with local scope network drivers. You do so by promoting the network scope to `swarm` during the creation of the network. You will then be able to use this network when creating services.

```
$ docker network create -d bridge \
  --scope swarm \
  --attachable \
  swarm-network
```

For network drivers which provide connectivity across hosts (ex. macvlan), if node specific configurations are needed in order to plumb the network on each host, you will supply that configuration via a configuration only network. When you create the swarm scoped network, you will then specify the name of the network which contains the configuration.

```
node1$ docker network create --config-only --subnet 192.168.100.0/24 --gateway 192.168.100.115 mv-config
node2$ docker network create --config-only --subnet 192.168.200.0/24 --gateway 192.168.200.202 mv-config
node1$ docker network create -d macvlan --scope swarm --config-from mv-config --attachable swarm-network
```

## OPTIONS

**--attachable**[=false] Enable manual container attachment

**--aux-address**=map[] Auxiliary IPv4 or IPv6 addresses used by Network driver

**--config-from**="" The network from which to copy the configuration

**--config-only**[=false] Create a configuration only network

**-d**, **--driver**="bridge" Driver to manage the Network

**--gateway**=[] IPv4 or IPv6 Gateway for the master subnet

**-h**, **--help**[=false] help for create

**--ingress**[=false] Create swarm routing-mesh network

**--internal**[=false] Restrict external access to the network

**--ip-range**=[] Allocate container ip from a sub-range

**--ipam-driver**="default" IP Address Management Driver

**--ipam-opt**=map[] Set IPAM driver specific options

**--ipv6**[=false] Enable IPv6 networking

**--label=** Set metadata on a network

**-o**, **--opt**=map[] Set driver specific options

**--scope**="" Control the network's scope

**--subnet**=[] Subnet in CIDR format that represents a network segment

## SEE ALSO

**docker-network(1)**