docker network create

Estimated reading time: 9 minutes

Description

Create a network

Usage

```
docker network create [OPTIONS] NETWORK
```

Options

Name, shorthand	Default	Description
`--attachable`		API 1.25+ (https://docs.docker.com/engine/api/v1.25/) Enable manual container attachment
`--aux-address`		Auxiliary IPv4 or IPv6 addresses used by Network driver
`--config-from`		API 1.30+ (https://docs.docker.com/engine/api/v1.30/) The network from which copying the configuration
`--config-only`		API 1.30+ (https://docs.docker.com/engine/api/v1.30/) Create a configuration only network
`--driver , -d`	`bridge`	Driver to manage the Network
`--gateway`		IPv4 or IPv6 Gateway for the master subnet
`--ingress`		API 1.29+ (https://docs.docker.com/engine/api/v1.29/) Create swarm routing-mesh network
`--internal`		Restrict external access to the network
`--ip-range`		Allocate container ip from a sub-range
`--ipam-driver`		IP Address Management Driver
`--ipam-opt`		Set IPAM driver specific options
`--ipv6`		Enable IPv6 networking
`--label`		Set metadata on a network

Name, shorthand	Default	Description
`--opt , -o`		Set driver specific options
`--scope`		API 1.30+ (https://docs.docker.com/engine/api/v1.30/) Control the network's scope
`--subnet`		Subnet in CIDR format that represents a network segment

Parent command

Command	Description
docker network (https://docs.docker.com/engine/reference/commandline/network)	Manage networks

Related commands

Command	Description
docker network connect (https://docs.docker.com/engine/reference/commandline/network_connect/)	Connect a container to a network
docker network create (https://docs.docker.com/engine/reference/commandline/network_create/)	Create a network
docker network disconnect (https://docs.docker.com/engine/reference/commandline/network_disconnect/)	Disconnect a container from a network
docker network inspect (https://docs.docker.com/engine/reference/commandline/network_inspect/)	Display detailed information on one or more networks
docker network ls (https://docs.docker.com/engine/reference/commandline/network_ls/)	List networks
docker network prune (https://docs.docker.com/engine/reference/commandline/network_prune/)	Remove all unused networks
docker network rm (https://docs.docker.com/engine/reference/commandline/network_rm/)	Remove one or more networks

Extended 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).

While not required, it is a good idea to install Docker Swarm to manage the cluster that makes up your network. Swarm provides sophisticated discovery and server management tools 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.

Overlay network limitations

You should create overlay networks with `/24` blocks (the default), which limits you to 256 IP addresses, when you create networks using the default VIP-based endpoint-mode. This recommendation addresses limitations with swarm mode (https://github.com/moby/moby/issues/30820). If you need more than 256 IP addresses, do not

increase the IP block size. You can either use `dnsrr` endpoint mode with an external load balancer, or use multiple smaller overlay networks. See Configure service discovery (https://docs.docker.com/engine/swarm/networking/#configure-service-discovery) for more information about different endpoint modes.

Examples

Connect containers

When you start a container, use the `--network` flag to connect it to a network. This example 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.

Specify 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 --driver=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. This example uses two `/25` subnet mask to adhere to the current guidance of not having more than 256 IPs in a single overlay network. Each of the subnetworks has 126 usable addresses.

```
$ docker network create -d overlay \
  --subnet=192.168.1.0/25 \
  --subnet=192.170.2.0/25 \
  --gateway=192.168.1.100 \
  --gateway=192.170.2.100 \
  --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.

Bridge driver options

When creating a custom network, the default network driver (i.e. `bridge`) has additional options that can be passed. The following are those options and the equivalent docker daemon flags used for docker0 bridge:

Option	Equivalent	Description
`com.docker.network.bridge.name`	-	bridge name to be used when creating the Linux bridge
`com.docker.network.bridge.enable_ip_masquerade`	`--ip-masq`	Enable IP masquerading
`com.docker.network.bridge.enable_icc`	`--icc`	Enable or Disable Inter Container Connectivity
`com.docker.network.bridge.host_binding_ipv4`	`--ip`	Default IP when binding container ports
`com.docker.network.driver.mtu`	`--mtu`	Set the containers network MTU

The following arguments can be passed to `docker network create` for any network driver, again with their approximate equivalents to `docker daemon` .

Argument	Equivalent	Description
`--gateway`	-	IPv4 or IPv6 Gateway for the master subnet
`--ip-range`	`--fixed-cidr`	Allocate IPs from a range

Argument	Equivalent	Description
`--internal`	-	Restrict external access to the network
`--ipv6`	`--ipv6`	Enable IPv6 networking
`--subnet`	`--bip`	Subnet for network

For example, let's use `-o` or `--opt` options to specify an IP address binding when publishing ports:

```
$ docker network create \
    -o "com.docker.network.bridge.host_binding_ipv4"="172.19.0.1" \
    simple-network
```

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.driver.mtu=9216 \
  --opt encrypted=true \
  my-ingress-network
```