Docker networking

Network drivers

Docker networks subsystem is pluggable using network drivers.

A bridge may be a hardware device or a software device present on host system. It forwards traffic between network segments. In Docker it is a **software bridge** container to communicate to each other if they are preset on the same bridge. Bride network apply to container running on the same **Docker daemon host.**For container present on different docker daemon host communication happens using network overlay or routing at os level. When you start using docker default bridge network is automatically created and newly start container connect to it unless an until specified.

User defined bridge Vs Default Bridge.

User define bridge provide better isolation and interoperability between the containers.Conatiners present on user-defined bridge expose **all port to each other and no port to outside world.**

Case For a application containing web front-end and database backend, where you need to provide access to only front end and not not back end user defined network is used and only web ports are opened to outside world. Whereas for default network both front end and backend ports are exposed.

User defined bridge provides **automatic DNS resolution** between conatainers.

Conatiners on default bridge can only be accessed by IP Address unless -- link option is specified. On user defined bridge container can resolve each other with name or alias.

Case Consider the scenario of web front end and database backend having container name web and db respectively The Web container can connect to database container at db no matter which docker host the application is running on. If this was present on default network we need to manually create links between container using -- link ,This links need to be created for both the direction has complexcity start increasing as number of containers start increasing.

Container can be  **attached** or  **detached** from user defined container on fly. To remove a container from default bridge network. You need to stop the container and recreate it with default bridge options.

User Defined bridges are easily **configurable.** Default bridge network can be configured but the containers will uses the same setting such as MTU IP tables etc. Additionally configuration of default network drive happens outside bridge network and it requires a docker restart.

**Linked containers**  present on **default** bridge network share environmental variable.This type of variable sharing is not possible for user defined networks.

Following are better option to share environment variable for bridge network.

Multiple network can mount a file or directory containing shared information using docker volumes.

Multple conatainer can be started together using docker compose and can have a shared environmental variables defined.

Swarm service can be used to to take advantage of secrets or configs.

Creating user defined network

The network created can be have customized ip address range gateway etc.

docker network create my\_net.

Docker container rm my\_net can be used to remove a user defined bridge.

--network option is used to connect your container to user-defined bridge network.

Docker network connect

docker network disconnect

These command can be used to connect and disconnect containers to user defined networks

Host: for standalone containers it is use to remove network isolation and used networks host directly.

If host network mode is used for a container the containers network stack is not isolated from the docker host container (containers share host networking space) and the container does not gets it own ip address. If you run a container which binds to port 80 and use host networking mode. It is available on port 80 of host. For host networking port mapping does not take place .Host networking mode is only supported for  **linux docker host** . it is not supported for windows or mac addressing. Host network can also be used for swarm services by --network host with docker service create command.

Overlay network:

They connect multiple docker daemons together and enable swarm service to communicate with each others. Overlay network can be also used for swarm services to communicate over standalone containers or between two standalone containers having a different docker daemon. This stagery general removes os level routing.

Overlay network creates a distributed network among multiple docker daemon host.This network sits on top of the host specified network(i.e overlays) allowing the containers to communicate securely . There is network transparency for routing a packet to and from docker daemon host.

When a docker swarm is initialized or new host is joined to swarm two networks are created on that docker host.

An overlay network called ingress , it handles or controls data traffic related to swarm services. Swarm service are connected to ingress overlay network if nothing is specified explicitly.

Second network which is created is bridge network called docker\_gwbridge. It connects individual docker daemon to other participating docker daemon.

Creating a overlay network.

Firewall rule for docker daemon using overlay network. Following port must be pen to allow traffic to flow through the each docker host participating in overlay network. TCP port 2377 for cluster management communication. TCP and udp port 7946 for communication between nodes. Tcp port 4789 for overlay network traffic.

Docker host should either start a swarm or join a swarm to uses overlay network. This create/s a default ingress network which swarm uses by default.

docker network create – do overlay my-overlay

to create a overlay network for which can be used by swarm service or other standalone container located on different docker daemon host add --attachable flag.

docker network create my-overlay –attachable my-attachable overlay IP address range subnet and Gateway.

All swarm service management traffic through swarm is encrypted by default using AES algorithm in GCM mode. Manger node in swarm rotates the key to encrypt data ever 12 hours. To encrypt data also use --opt encrypted when i=creating a overlay network . these enable IPSEC encryption at vxlan .Dokcer is responsible for creating IPSEC tunneling between all nodes. This tunnel also uses AES encryption alogo rthim with automatic key rotation.

**Windows node are not suitable for overlay network**

Customizing default ingress network.

Costomizing the ingress network involves removing and recreating it. This is usally done before you create any swarm service. If you hae any service running on ingress network those service needs to be removed to remove the ingress network.

During time no ingress network is present , existing service which **do not publish port** continue to function but are not load balanced. Service which pulishs port gets affected by this action.

Inspect the default ingress network with “ docker network inspect ingress “ command.

Remove existing ingress network “ docker network rm ingress “

Create a new ingress network using -- ingress flag along with all the customized option which you want to configure. Example set MTU to 1200 subnet to 10.11.0.0/16 and set gateway to 10.11.0.2

Example

docker network create –driver overlay –ingress --subnet 10.11.0.0/16 --gateway 10.11.0.2 --opt com.docker.network drivermtu=1200 my-ingress.

Note ingress network can be named ingress but only one ingress network should be present.

Customizing docker gw\_bridge interface.

Docker gw\_bridge is a virtual bridge that connect overlays network ( including ingress network) to docker daemon’s physical network. docker creates it when host initailze or join a swarm but it is not a docker device. It exist in kernel of the host If you want to do any customization you shoul do before the docker host is part of any swarm activities.

docker stop

Delete the existing gw\_bridge interface

sudo ip link set docker\_gwbridge down

sudo ip link del dev docker\_gwbridge

start docker

create or recreate docker\_gw interface with netw customized option using docker network create

Example

docker network create --subnet 10.11.0.0/16 –opt com.docker.network.bridge.name=docker\_gwbridge --opt com.docker.network.bridge.enable\_icc=false --opt com.docker.network.bridg.enable\_ip\_masquerade= true docker\_gwbridge.

Part take in swarm activities as bridge already exist .

Publish port on overlay network

Swarm service connected to overlay network effectively exposes all the ports to each other.For a port to be accessible outside the network the port must be published with –p or –publish command. With docker service create or docker service updated. Both legacy separated syntax and new comma separated syntax are supported.

By default swarm which publishes the port do this using routing mesh. When you connect to publish port of any node present in swarm whether it has the given service running or not you are always redirected to a worker which has that service running.That is docker acts as a load balancer for your service. Services using routing mesh are running in virtual ip (vip) mode. Even service running on each node uses routing mesh. When using routing mesh there is no gaurentee which docker swarm node will serve the client request. To bypass the routing mesh , you can start a **DNS round Robin** mode using --endpoint-mode flag to dnsrr . For this you must run your own load balancer in front of that service. A DNS query for the service name on the docker host returns a list of IP address for for thr nodes running in the service.

MACVLAN : macvlan networks allows you to assign mac address to your container. Making it appear as a physical device on your network. The docker daemon routes traffic to container based on their mac address. Macvlan network is general used for legacy application which need to be directly connected to physical network. rather than routing through host network stack.

To use macvlan networking mode you need a physical interface on your docker host along with subnet and gateway of macvlan.

Non it is used to disable network for your container you can uses –network none flag when starting your container. Hence only a loop back device is created within the container.

Create container

Docker container run --rm -dit --network none --name my\_container alpine:latest

If you see inside the container no etho is created.