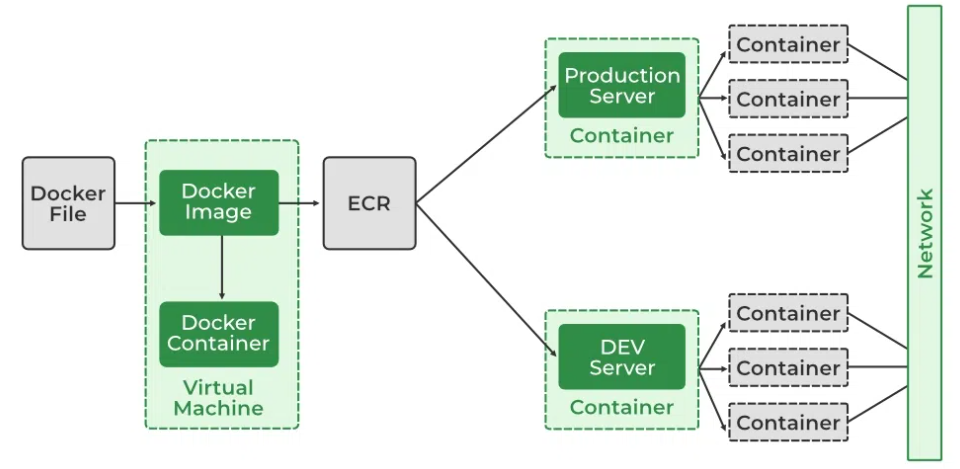
# Docker Networking

Docker Networking allows you to create a Network of Docker Containers managed by a master node called the manager. Containers inside the Docker Network can talk to each other by sharing packets of information. In this article, we will discuss some basic commands that would help you get started with Docker Networking.

## Docker Networking

A network is a group of two or more devices that can communicate with each other either physically or virtually. The Docker network is a virtual network created by Docker to enable communication between [Docker containers](https://www.geeksforgeeks.org/containerization-using-docker/). If two containers are running on the same host they can communicate with each other without the need for ports to be exposed to the host machine. You may use Docker to manage your Docker hosts in any platform manner, regardless of whether they run [Windows, Linux,](https://www.geeksforgeeks.org/operating-systems/) or a combination of the two.

**

## Network Drivers

There are several default network drivers available in Docker and some can be installed with the help of plugins, Command to see the list of containers in Docker mentioned below.

docker network ls

### Types of Network Drivers

1. **bridge:**If you build a container without specifying the kind of driver, the container will only be created in the bridge network, which is the default network.
2. **host:**Containers will not have any IP address they will be directly created in the system network which will remove isolation between the docker host and containers.
3. **none:**IP addresses won’t be assigned to containers. These containments are not accessible to us from the outside or from any other container.
4. **overlay:**overlay network will enable the connection between multiple Docker demons and make different Docker swarm services communicate with each other.
5. **ipvlan:**Users have complete control over both IPv4 and IPv6 addressing by using the IPvlan driver.
6. **macvlan:**macvlan driver makes it possible to assign MAC addresses to a container.

*A screenshot of a computer

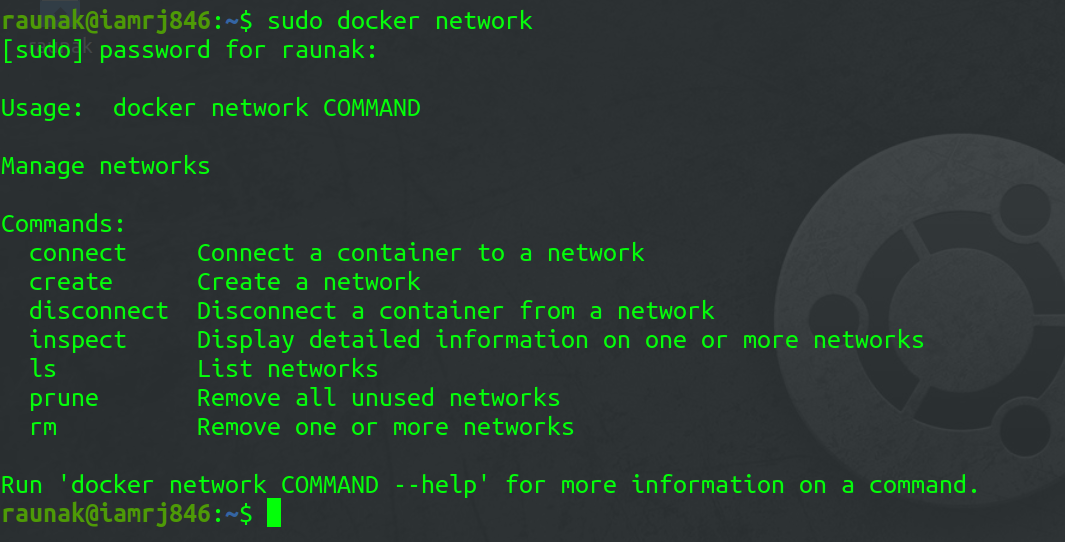
Description automatically generated with low confidence*

## Launch a Container on the Default Network

### 1. Understanding the Docker Network Command

The Docker **Network** command is the main command that would allow you to create, manage, and configure your Docker Network. Let’s see what the sub-commands can be used with the Docker Network command. to know more about [Creating a Network in Docker and Connecting a Container to That Network.](https://www.geeksforgeeks.org/creating-a-network-in-docker-and-connecting-a-container-to-that-network/)

sudo docker network

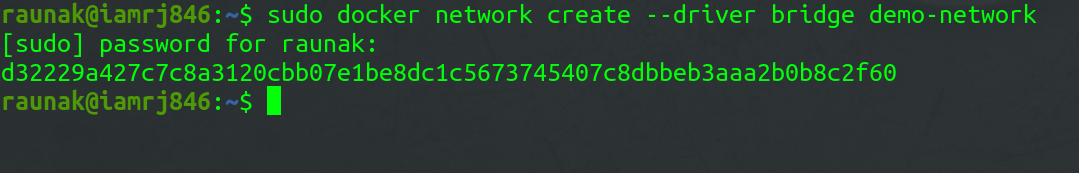


We will see all the Network sub-commands one by one.

### 2. Using Docker Network Create command

With the help of the **“Create”**command, we can create our own docker network and can deploy our containers in **it.**

sudo docker network create --driver <driver-name> <bridge-name>

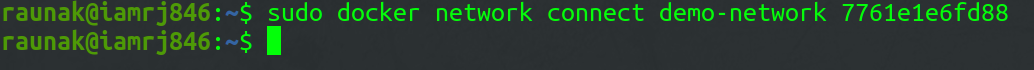


### 3. Using the Docker Network Connect command

Using the **“Connect”** command, you can connect a running Docker Container to an existing Network.

sudo docker network connect <network-name> <container-name or id>

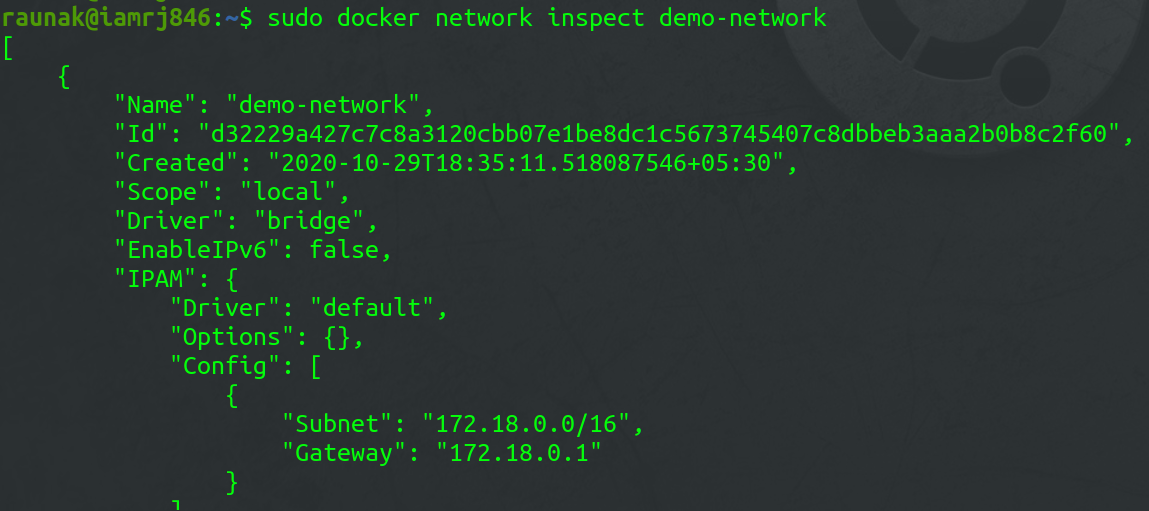
In this example, we will connect an Ubuntu Container to the Bridge Network we created in the last step.



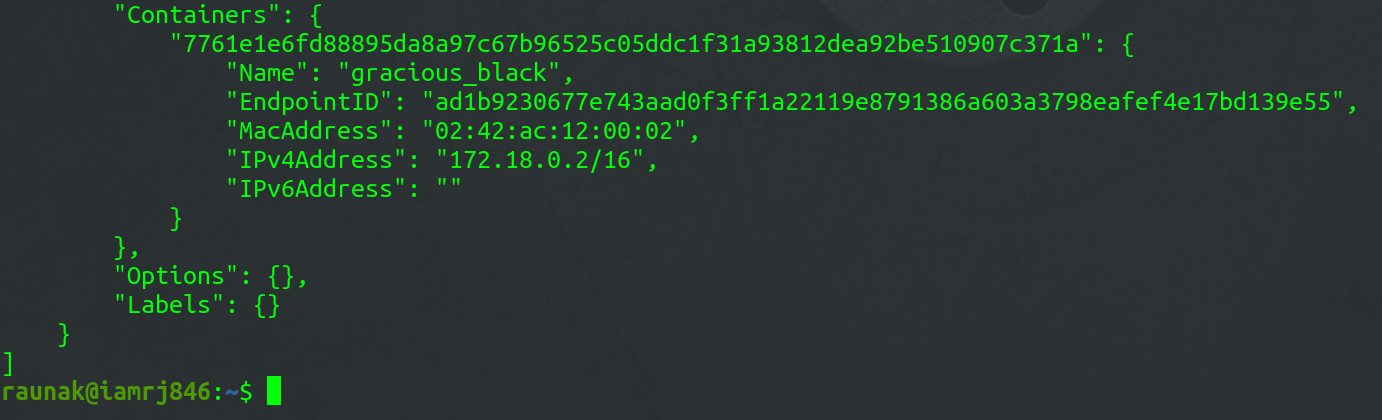
### 4. Using the Docker Network Inspect  command

Using the Network Inspect command, you can find out the details of a Docker Network.

sudo docker network inspect <network-name>



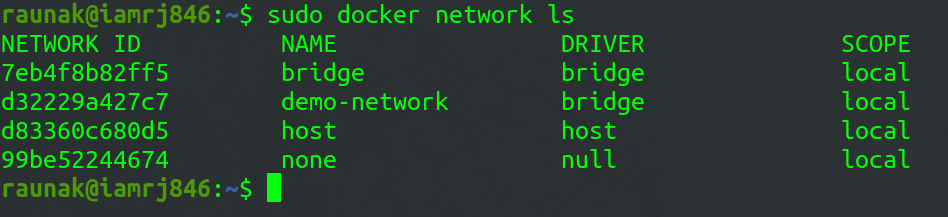
You can also find the list of Containers that are connected to the Network.



### 5. Using the Docker Network ls  command

To list all the Docker Networks, you can use the **list**command.

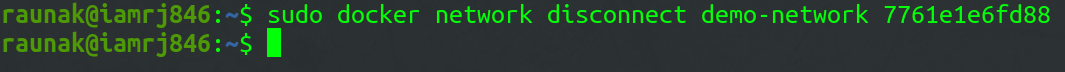
sudo docker network ls



### 6. Using the Docker Network Disconnect command

The **disconnect** command can be used to remove a Container from the Network.

sudo docker network disconnect <network-name> <container-name>

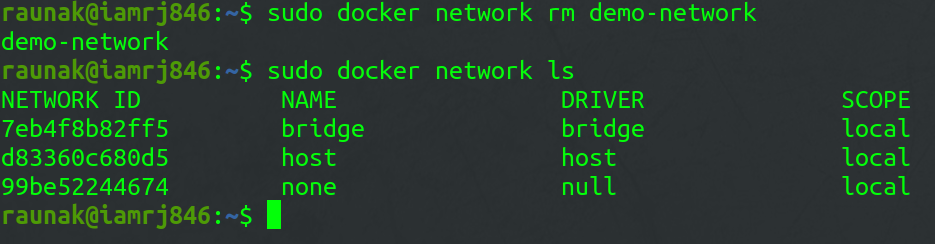


### 7. Using the Docker Network rm command

You can remove a Docker Network using the **rm** command.

sudo docker network rm <network-name>

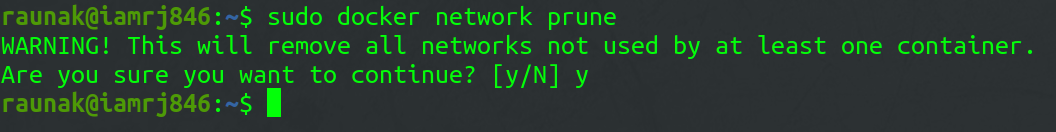
Note that if you want to remove a network, you need to make sure that no container is currently referencing the network.



### 8. Using the Docker Network prune command

To remove all the unused Docker Networks, you can use the **prune** command.

sudo docker network prune



To know more about Docker-published ports refer to this article [Docker – Managing Ports](https://www.geeksforgeeks.org/docker-managing-ports/)

## Common Operations

* **docker network inspects:** We may examine the configuration information of a specific network, such as the name of the network, the containers that have linked to this network, the type of driver used to construct this network, and other characteristics, by using the**“docker network inspect”** command.
* **docker network ls:** We can see all of the networks that are available on the current host by using **“docker network ls”**.
* **docker network creates:**Using the command **“docker network create”** and the name of the driver, such as bridge, overlay, or macvlan, we can establish a new network.
* **docker network connects:** In order to use this command, we must first confirm that the appropriate network has already been formed on the host. Then, using docker **“network connect”**, we may attach the container to the necessary network.