Docker networking is a crucial part of managing containerized applications, allowing containers to communicate with each other, the host system, and external networks. Below is a list of important Docker networking commands along with explanations:

**1. docker network ls**

* **Purpose**: Lists all the networks available in Docker.
* **Explanation**: This command shows the networks created by Docker, including the default ones (bridge, host, none) and any user-defined networks.

bash

Copy code

docker network ls

**2. docker network inspect <network\_name>**

* **Purpose**: Provides detailed information about a specific network.
* **Explanation**: This command displays configuration details, connected containers, and settings of the specified network.

bash

Copy code

docker network inspect bridge

**3. docker network create <network\_name>**

* **Purpose**: Creates a new user-defined network.
* **Explanation**: You can create different types of networks (e.g., bridge, overlay, macvlan). The default type is bridge.

bash

Copy code

docker network create my\_bridge\_network

**4. docker network connect <network\_name> <container\_name>**

* **Purpose**: Connects a container to an existing network.
* **Explanation**: This command attaches a running container to an additional network, enabling it to communicate with other containers on that network.

bash

Copy code

docker network connect my\_bridge\_network my\_container

**5. docker network disconnect <network\_name> <container\_name>**

* **Purpose**: Disconnects a container from a network.
* **Explanation**: Removes the container from the specified network, stopping its communication with other containers on that network.

bash

Copy code

docker network disconnect my\_bridge\_network my\_container

**6. docker run --network <network\_name>**

* **Purpose**: Specifies a network for a container at runtime.
* **Explanation**: When starting a new container, you can specify which network it should join using this option.

bash

Copy code

docker run --network my\_bridge\_network -d nginx

**7. docker network rm <network\_name>**

* **Purpose**: Removes a user-defined network.
* **Explanation**: Deletes the specified network. Containers connected to this network will be disconnected, but the containers themselves will not be stopped.

bash

Copy code

docker network rm my\_bridge\_network

**8. docker network prune**

* **Purpose**: Removes all unused networks.
* **Explanation**: This command deletes all networks that are not being used by at least one container, freeing up resources.

bash

Copy code

docker network prune

**9. docker run --network="host"**

* **Purpose**: Runs a container with the host network.
* **Explanation**: The container shares the network stack with the host, which means it has the same IP address as the host system. Useful for performance-critical applications, but be cautious with security.

bash

Copy code

docker run --network="host" -d nginx

**10. docker run --network="none"**

* **Purpose**: Runs a container with no network access.
* **Explanation**: The container is isolated from all networks. This is useful for security reasons when a container doesn’t need any network connectivity.

bash

Copy code

docker run --network="none" -d nginx

**11. docker network create --driver <driver\_name> <network\_name>**

* **Purpose**: Creates a network with a specific driver.
* **Explanation**: Docker supports different network drivers (bridge, overlay, macvlan, etc.). This command allows you to create a network with a custom driver.

bash

Copy code

docker network create --driver overlay my\_overlay\_network

**12. docker network create --subnet <subnet> --gateway <gateway> <network\_name>**

* **Purpose**: Creates a network with a custom IP subnet and gateway.
* **Explanation**: This is useful when you want to control the IP range and gateway that Docker assigns to containers on this network.

bash

Copy code

docker network create --subnet 192.168.1.0/24 --gateway 192.168.1.1 my\_custom\_network

**13. docker network create --ip-range <ip-range> <network\_name>**

* **Purpose**: Specifies an IP range for container IPs within the subnet.
* **Explanation**: Limits the IP addresses that Docker can assign to containers in a network to a specific range within the network’s subnet.

bash

Copy code

docker network create --subnet 192.168.1.0/24 --ip-range 192.168.1.128/25 my\_custom\_network

**14. docker-compose up with networks**

* **Purpose**: Defines and starts containers with custom networks in a Docker Compose file.
* **Explanation**: You can specify networks in the docker-compose.yml file to ensure containers are attached to specific networks when started.

yaml

Copy code

version: '3'

services:

web:

image: nginx

networks:

- my\_network

networks:

my\_network:

driver: bridge

These commands provide you with control over Docker's networking features, allowing for effective management and customization of container communication.