# Docker Network Examples on EC2

## 1. Bridge Network

A bridge network is the default network driver. It allows containers connected to the same bridge network to communicate with each other, while isolating them from external networks.  
  
Example:  
  
# Create a bridge network  
docker network create my\_bridge\_network  
  
# Run two containers on the bridge network  
docker run -d --name container1 --network my\_bridge\_network busybox sleep 3600  
docker run -d --name container2 --network my\_bridge\_network busybox sleep 3600  
  
# Containers can communicate using container names  
docker exec container1 ping container2

## 2. Host Network

A host network removes the network isolation between the Docker container and the Docker host, using the host’s networking directly.  
  
Example:  
  
# Run a container with the host network  
docker run -d --name container\_host --network host nginx  
  
In this example, the `nginx` container will use the same network interfaces as the host EC2 instance.

## 3. None Network

The none network disables all networking for a container. It can be useful for security or for containers that do not need network access.  
  
Example:  
  
# Run a container with the none network  
docker run -d --name container\_none --network none busybox sleep 3600

## 4. Custom Bridge Network

A custom bridge network is similar to the default bridge network but allows for more configuration options.  
  
Example:  
  
# Create a custom bridge network  
docker network create --driver bridge my\_custom\_bridge  
  
# Run containers on the custom bridge network  
docker run -d --name container1 --network my\_custom\_bridge busybox sleep 3600  
docker run -d --name container2 --network my\_custom\_bridge busybox sleep 3600  
  
# Containers can communicate using container names  
docker exec container1 ping container2

## 5. Overlay Network

An overlay network is used for multi-host networking. It allows containers running on different Docker hosts to communicate with each other. This requires Docker Swarm or another orchestration tool.  
  
Example:  
First, initialize Docker Swarm on your EC2 instance:  
  
# Initialize Docker Swarm  
docker swarm init --advertise-addr <EC2-Instance-IP>  
  
  
Then create an overlay network:  
  
# Create an overlay network  
docker network create -d overlay my\_overlay\_network  
  
  
Run services on the overlay network:  
  
# Run services on the overlay network  
docker service create --name service1 --network my\_overlay\_network nginx  
docker service create --name service2 --network my\_overlay\_network busybox sleep 3600

## 6. Macvlan Network

A macvlan network assigns a MAC address to each container, making it appear as a physical device on the network.  
  
Example:  
  
# Create a macvlan network  
docker network create -d macvlan \  
 --subnet=192.168.1.0/24 \  
 --gateway=192.168.1.1 \  
 -o parent=eth0 my\_macvlan\_network  
  
# Run a container on the macvlan network  
docker run -d --name container\_macvlan --network my\_macvlan\_network busybox sleep 3600

## 7. IPVLAN Network

The IPVLAN network driver gives users total control over both IPv4 and IPv6 addressing.  
  
Example:  
  
# Create an ipvlan network  
docker network create -d ipvlan \  
 --subnet=192.168.1.0/24 \  
 --gateway=192.168.1.1 \  
 -o parent=eth0 my\_ipvlan\_network  
  
# Run a container on the ipvlan network  
docker run -d --name container\_ipvlan --network my\_ipvlan\_network busybox sleep 3600

## 8. Custom Network Plugins

Docker supports third-party network plugins which can be used for advanced networking setups.  
  
Example:  
  
# Install a network plugin (e.g., Weave, Calico, etc.)  
docker plugin install weaveworks/net-plugin:latest\_release  
  
# Create a network using the plugin  
docker network create -d weave my\_weave\_network  
  
# Run containers on the plugin network  
docker run -d --name container\_weave --network my\_weave\_network busybox sleep 3600