# Overview

For containers to communicate, they must be attached to a network. In this guided practice, you will explore the common ways in which containers are networked to each other and the external world.

# Objectives

* Create containers
  + Manage containers with docker
    - Manage container storage and networking

## Skills Reviewed

* Creating, running, and removing containers
* Starting and stopping containers
* Attaching to and detaching from containers

## New Skills

* Viewing docker networks and network settings
* Creating docker networks
* Attaching a container to a network
* Connecting and disconnecting containers to and from a docker network
* Publishing ports from a docker container

## References

* Windows Container Networking - <https://docs.microsoft.com/en-us/virtualization/windowscontainers/container-networking/architecture>

# Initial Conditions

* Docker Enterprise installed on a Windows Server 2019 virtual machine
* The **iis-core-template** image created
  + A Windows Server Core container with IIS installed
* An external virtual switch named **VS-EXT-01**

# Final Conditions

A docker container named **Net-Test** and **Port-Test** created

# Instructions

## Setup

**Remove** any **existing** **containers**.

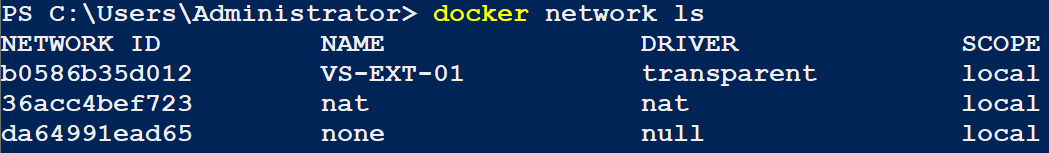
## The default Container Switch

When Docker Enterprise edition is installed on Server 2019 it creates a network named **nat** that is used by any containers that do not specify a network when run.

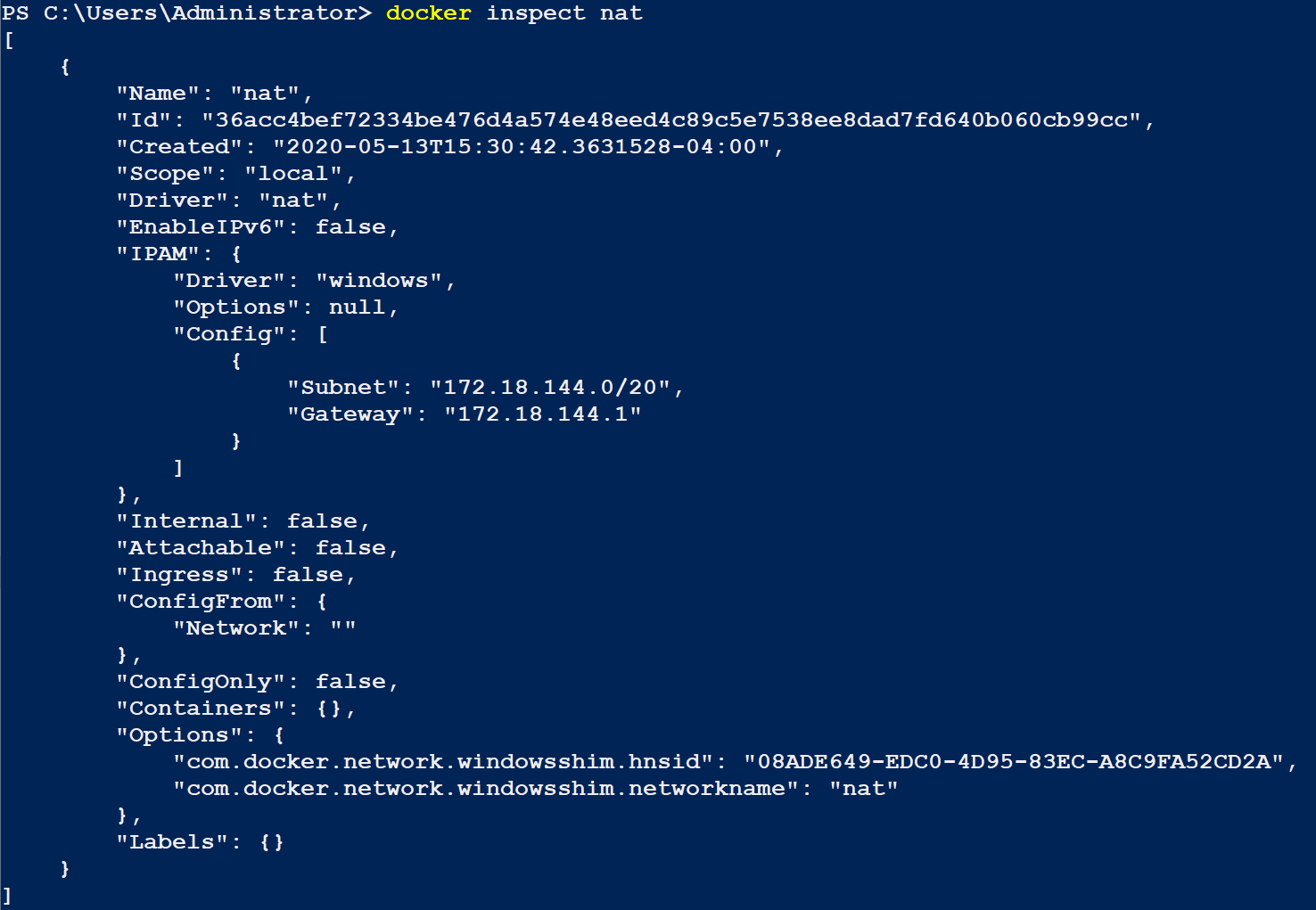
To view the docker networks on your system, perform the following:

1. Open an **elevated** **PowerShell** session on the **Host** virtual machine.
2. Type the following **command** to display all the networks available to docker connections:

docker network ls

1. You should see output like the figure below.
2. To see the details for a specific network, **type** the following command:

docker inspect nat

1. The output should look like the screenshot below. The information is presented in json format.
2. Note the subnet and gateway shown in the image above. When no network settings are specified the container will be given an IP address in the range specified and a default gateway specified in network configuation.

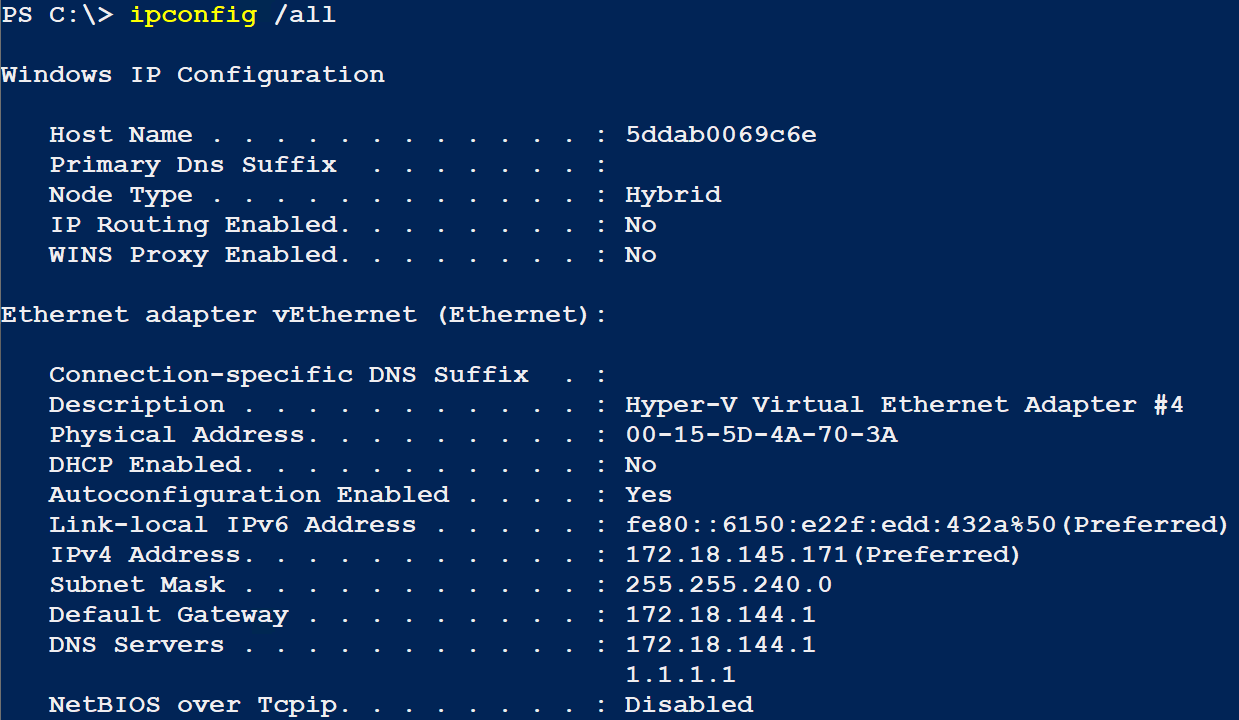
## Exploring the Default Networking Settings for a Container

In this step, you will start a container and inspect its configuration from inside the container. Then you will start an additional container and test for connectivity.

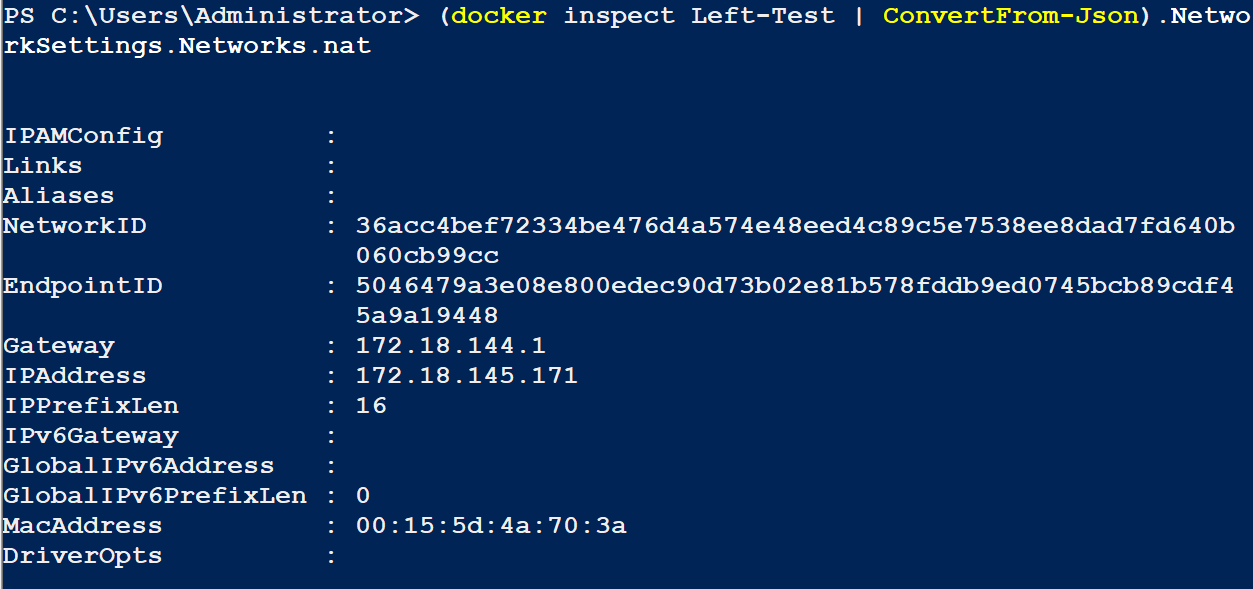
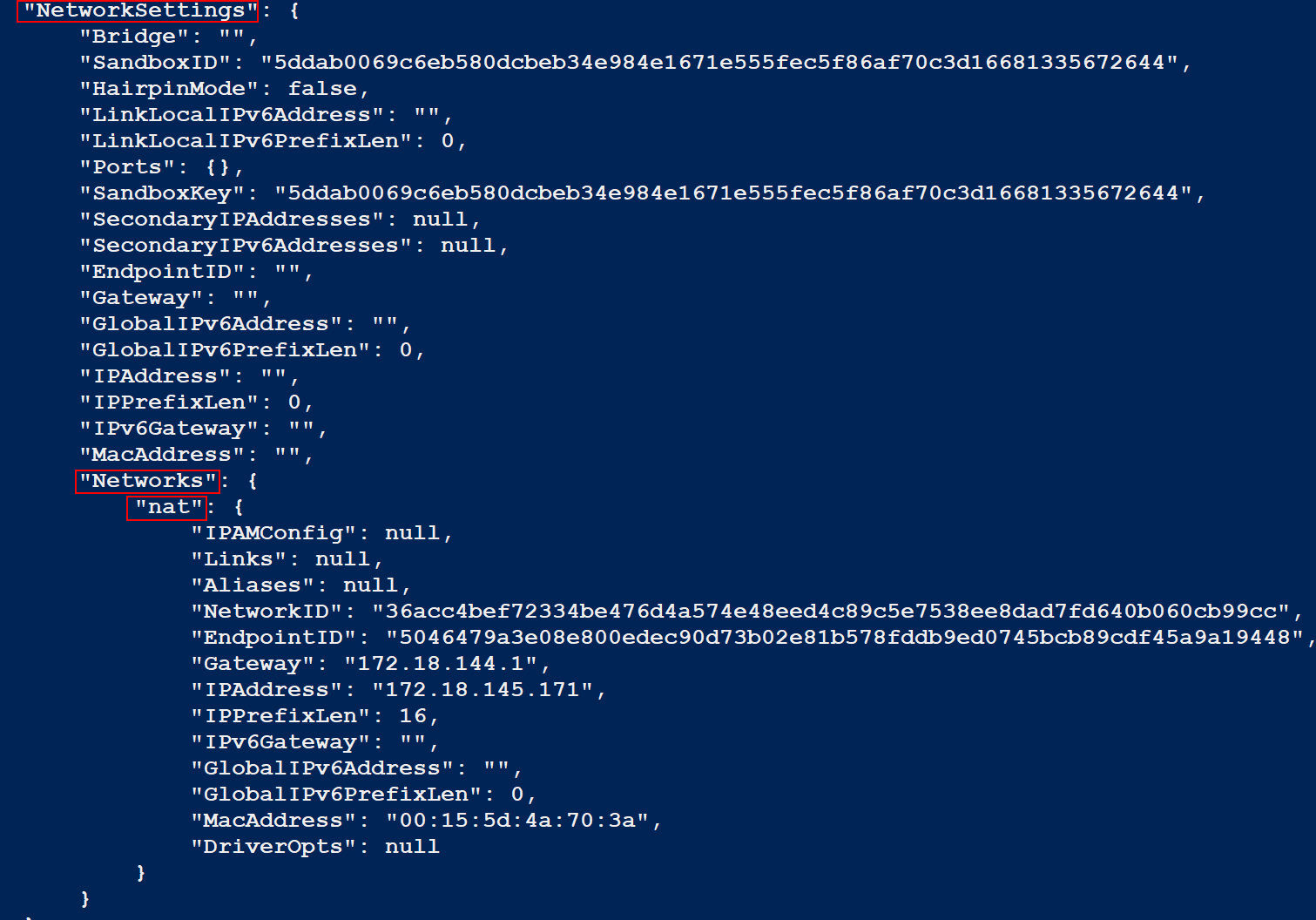
To view and test the network settings for a container, perform the following:

1. Open an **elevated** **PowerShell** session.
2. Create a **container** using the following settings:
   1. Name: **Left-Test**
   2. Image: **iis-core-template**
   3. Command: **PowerShell**
   4. Options: **interactive** **terminal**
3. You should now have a container with an interactive PowerShell session.
4. **Type** the following **command** to show the TCP/IP settings for your container.

ipconfig /all

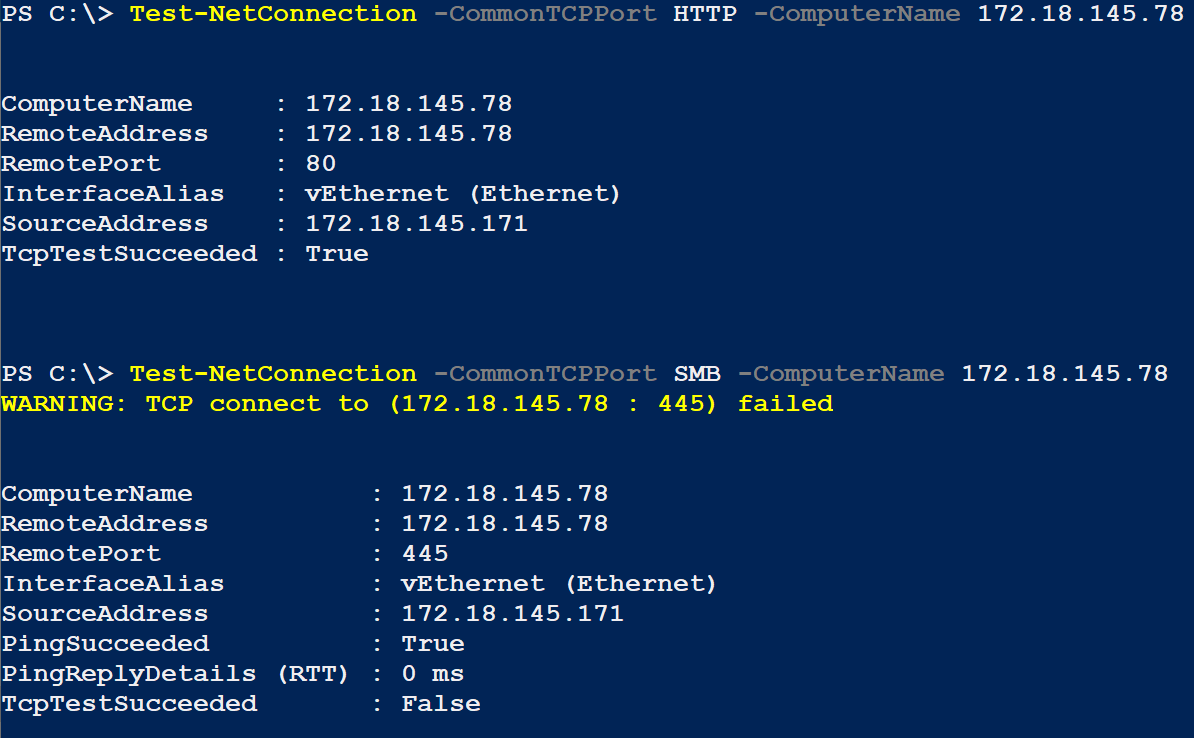
1. You should see settings like those shown below.
2. Notice that you have been assigned an address and gateway specified by the docker **nat** network and your DNS client has been configured to use the same address as the default gateway and the DNS server address configured on the NIC for the host machine. These configurations will allow you to reach the Internet from your container.
3. Open another **elevated** **PowerShell** session.
4. Type the following **command** to view the network settings for the **docker** container:

(docker inspect Left-Test | ConvertFrom-Json).NetworkSettings.Networks.nat

1. You should see the output like that shown below.
2. This command takes the output of the docker inspect command and converts it from JSON to a PowerShell object and allows you to use the dot object notation to extract just a portion of the output. The image below shows the output from the docker inspect and how it was used to create the command above.
3. In the **PowerShell** **session** you just opened, **create** another **container** using the **same** settings as the previous container except name it **Right-Test**.
4. Arrange the PowerShell sessions so that **Left-Test** is on the **left** and **Right-Test** is on the **right**.
5. Verify that you can **ping** **between** the **containers**.
6. Verify that the host **firewall** is still operational, by **typing** the following **commands** from one of the containers (You will need to change the IP addresses to match your configuration. Whichever container you are in the IP address used in the command should be the IP address of the other container):

Test-NetConnection -CommonTCPPort HTTP -ComputerName 172.18.145.78

Test-NetConnection -CommonTCPPort SMB -ComputerName 172.18.145.78

1. You should see the output like that shown below:
2. This is what you should expect given the current configuration of the firewall.
3. Open a browser on your host and browse to the **IP address** of one of your **containers**. You should be able to connect to the website. This is because the **nat** virtual switch is an internal switch that will give the host an adapter attached to the same network. This allows the host to be on the default network with any containers.
4. Exit and remove the containers you just created.

## Creating a Docker Network

### Creating a Network Using the NAT Driver

There are times when you will want to create a network that has different settings from the default network, or you need to isolate containers.

To create a new network using the **nat** driver, perform the following:

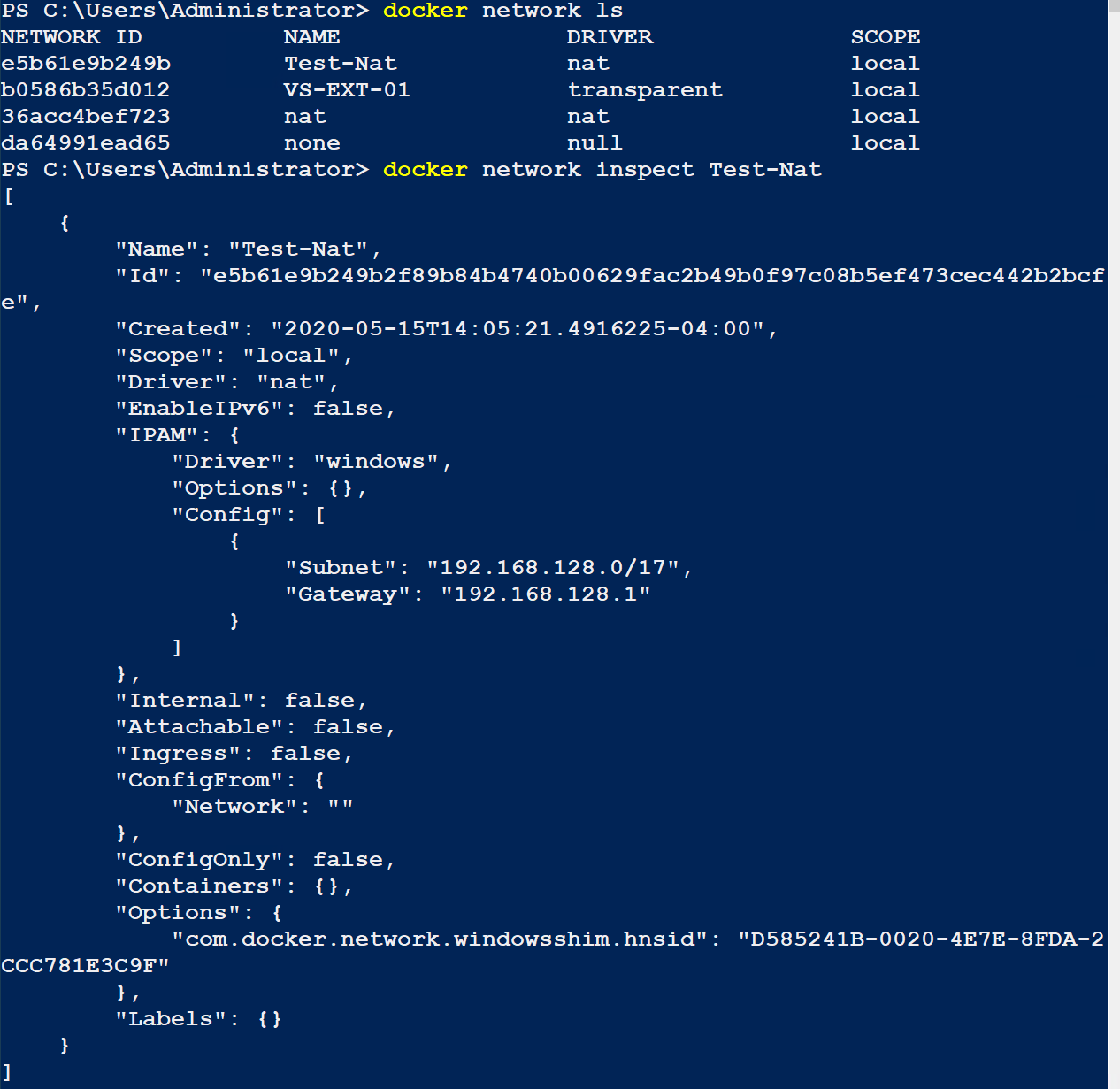
1. Open an **elevated** **PowerShell** session.
2. Type the following **command**:

docker network create --driver nat --subnet 192.168.128.0/17 Test-Nat

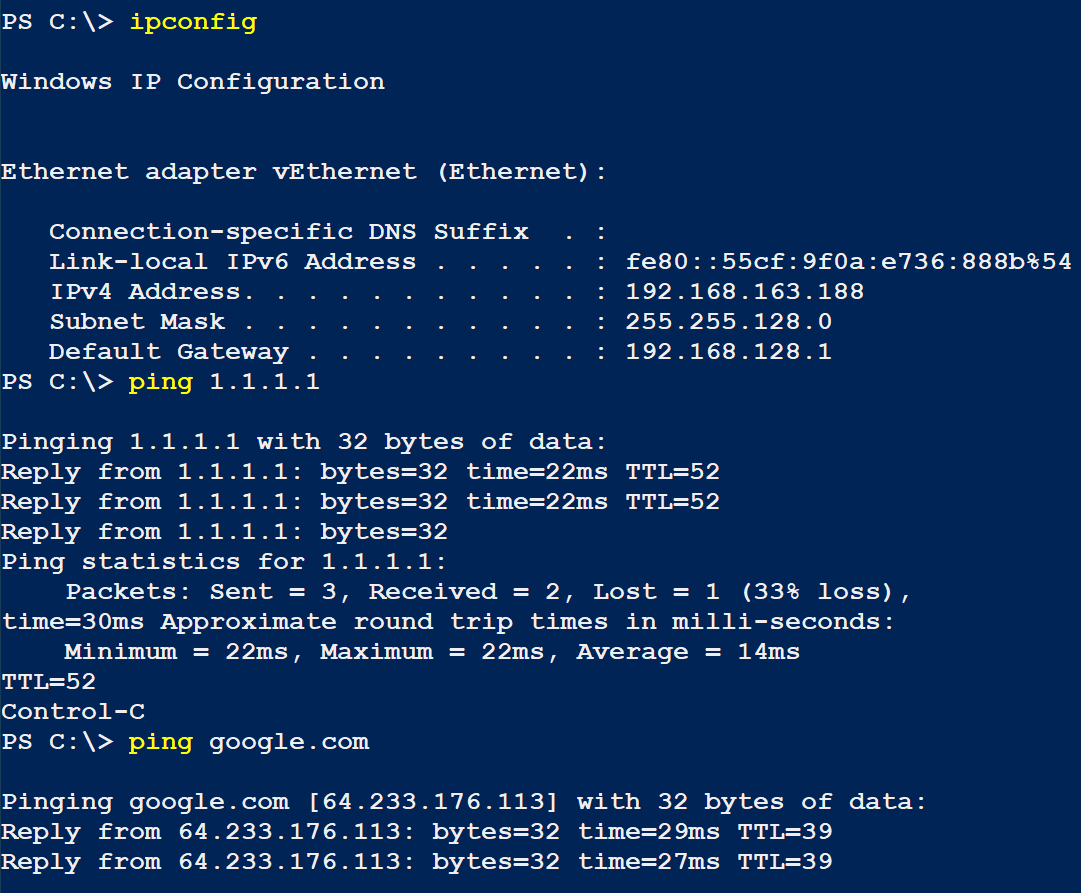
1. The command will output the ID for the network.
2. You can view the network using the following commands:

docker network ls

docker network inspect Test-Nat

1. You should see output like the image below.
2. To test the network, create a container and attach it to the **Test-Nat** network, by running the following command:

docker run -it --name Net-Test --network Test-Nat iis-core-template PowerShell

1. From the PowerShell session, you can test the network settings as shown in the figure below.
2. Stop the **Net-Test** container.

## Connecting to an External Network Using the Transparent Driver

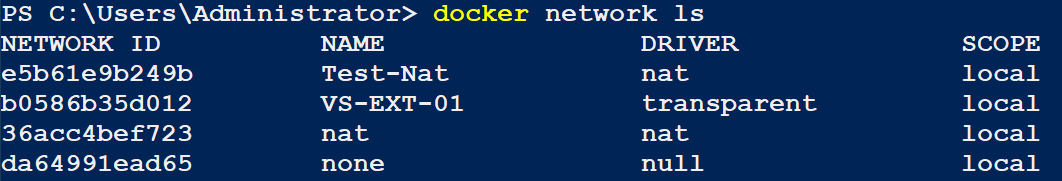
Windows containers have several different network driver types. The type chosen will depend on your connectivity requirements and your virtualization setup. The two most used drivers are **nat** (which you have been using) and transparent. The transparent driver allows you to create a network that is directly connected to a physical network using an external virtual switch.

In this section, you will view the networks and connect your running container to the **VS-EXT-01** network.

To connect a running container to a network, perform the following:

1. Start the **Net-Test** container and then **attach** to it interactively using a PowerShell session.
2. In an **elevated** **PowerShell** on the **Host** machine, **type** the following **command** to display the docker networks:

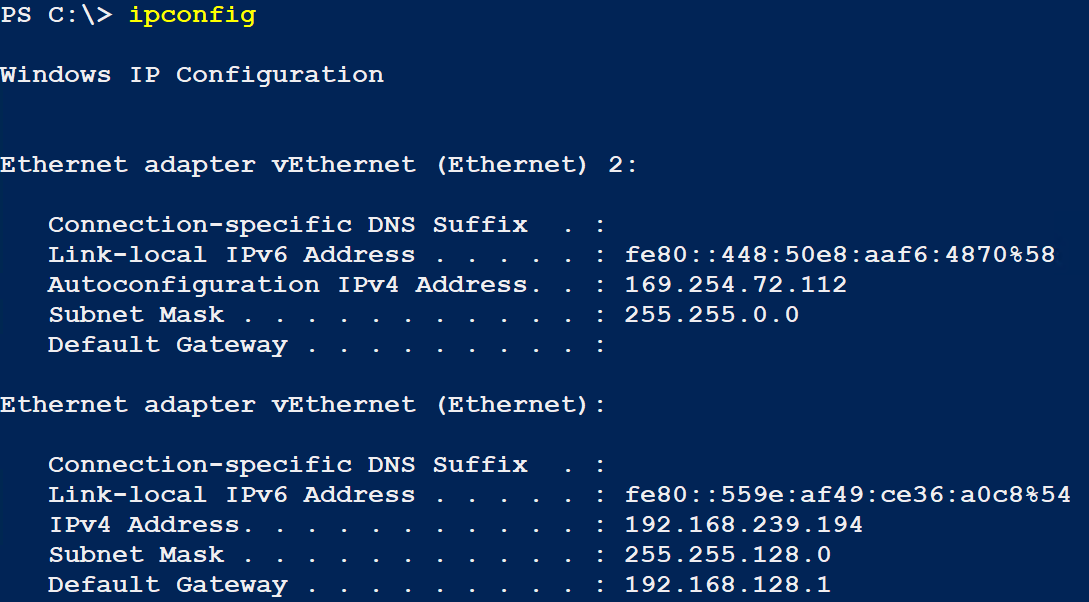
docker network ls

1. You should see the output shown below.
2. This output shows that the **VS-EXT-01** network is connected using the transparent driver, so you know that this network is connected to a physical network adapter.
3. To connect the **Net-Test** container to the **VS-EXT-01** network, type the following command:

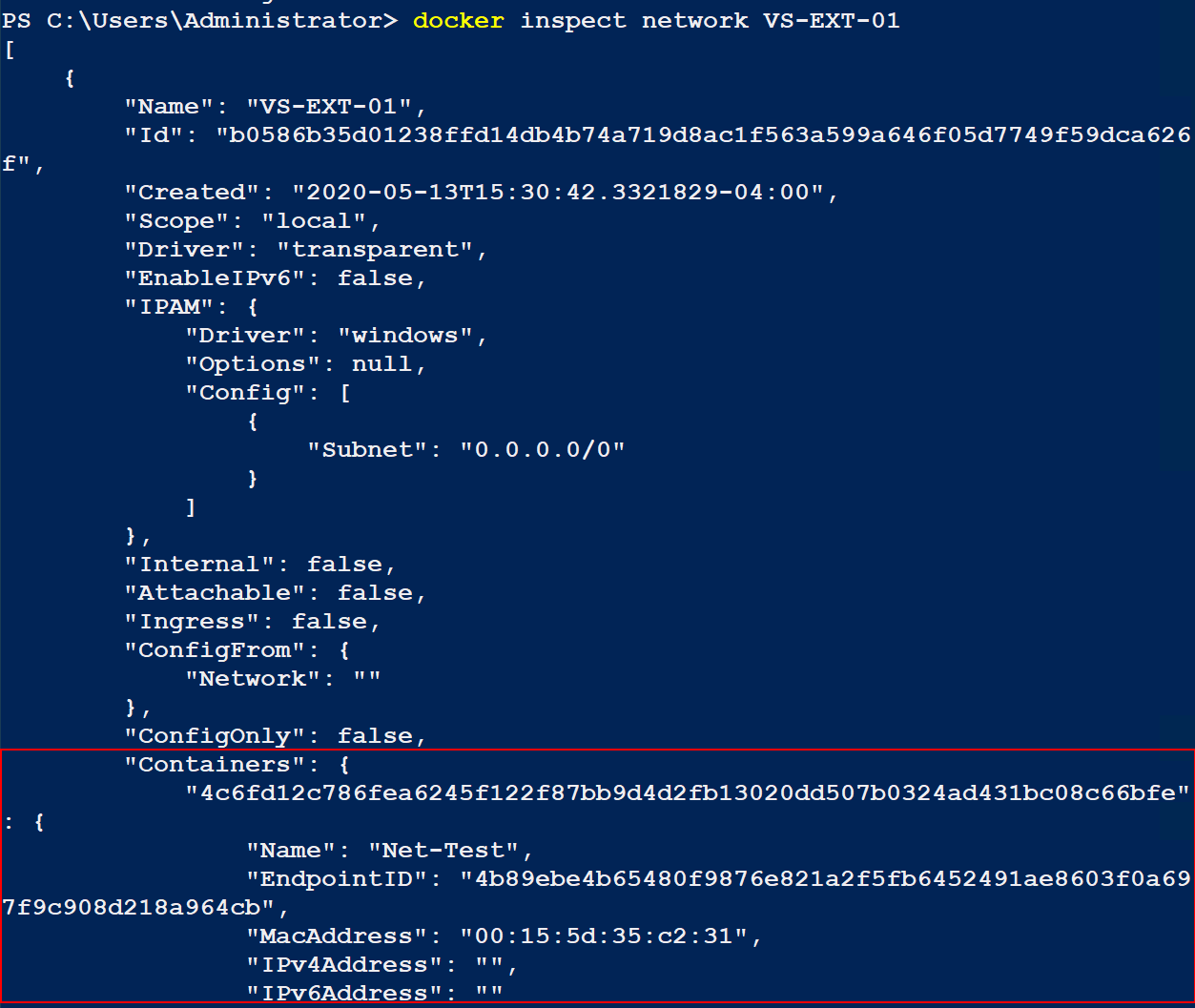
docker network connect VS-EXT-01 Net-Test

1. There is no output from this command. To verify that the network was connected perform, type the following command in the **Net-Test** container session:

ipconfig

1. You should see output like the screenshot below. Notice you have two adapters now. Since there is no DHCP server on the network the adapter has been assigned an APIPA address
2. You can also use the docker inspect command to check if the network adapter is connected. To do this, type the following on the host machine:

docker inspect network VS-EXT-01

1. Note the output below showing the container connected to the network.
2. **Disconnect** the **adapter** **from** the **container;** **type** the following **command**:

docker network disconnect VS-EXT-01 Net-Test

1. You can learn more about connecting a container to a network by running the following command:

docker network connect --help

1. **Stop** the **Net-Test** container.

## Exposing Container Ports on the Host

For testing or security purposes, you may want to only make individual services available to the physical network. This is called publishing ports. When publishing ports, you map an internal port on a container to an external port on the host.

To publish a port on a container, perform the following:

1. **Open** an **elevated PowerShell** session and **type** the following **command**:

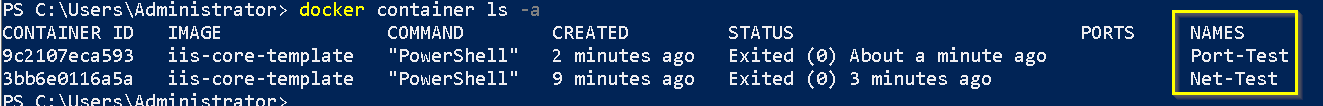
docker run -it --name Port-Test -p 12345:80/tcp iis-core-template PowerShell

1. The command above exposes TCP port 80 on the container to TCP port 12345 on the host. If you are connected to a transparent network, you can also choose the host IP address to bind to i.e. **-p 192.168.1.1:12345:80/tcp**.
2. You can test the port is exposed by using your web browser to browse to <http://localhost:12345>
3. **Exit** **PowerShell** to stop the container.

# Submission Requirements

1. Verify that you have two containers: **Port-Test** & **Net-Test**. If both containers do not exit, the grading script will fail with ALL red.

docker container ls -a



1. **Download** the **grading** **script** from the assignment page to the **C:\Scripts** folder.
2. Check your lab by running the following command:

Invoke-Pester -Path C:\Scripts\GP26-Managing\_Container\_ Networking.Test.ps1

**Note**: You will see a security warning when running the script. Enter **R** to run the script.

If you want to see more detail, add **-Output Detailed** to the command. This may assist you with troubleshooting

Invoke-Pester -Path C:\Scripts\GP26-Managing\_Container\_ Networking.Test.ps1 -Output Detailed

1. You should not see any red in the output. Red in the PowerShell way of telling you that an error condition exists. Most of the time, the output will tell you what is wrong. If it is not obvious, contact your teacher and ask for assistance. You will be learning PowerShell during this term. **Correct** any **errors** you may have and run the script until all the output has no red. You should see the output like the images belowText

   Description automatically generated
2. Capture a snippet that shows the PowerShell Command and all its output. If you must use more than one snippet to capture the output, you must have at least **one line of overlap** in the snippets. The text in the snippets **must be legible** when pasted into the Word document. Paste the snippet(s) into a **new** **Word** **document**
3. **Fill** **in** the **information** in the following table. Copy the following table into the **Word** **document** and fill in the information about all the **new** commands used in this lab (the example provided is not a new command and should be deleted):

|  |  |  |
| --- | --- | --- |
| Docker Networking Commands | | |
| Command | Example | Description |
| docker run | docker run -it --rm iis-core-template PowerShell | Creates a container using the iis-core-template …. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. **Upload** the **document** in the submission area of the assignment.