NNM Deployment Guide

Last Updated: July 11, 2023

Table of Contents

Welcome to the NNM Deployment Guide 5

Cloud Deployments 6

Cloud Deployment Software Requirements 7

Cloud Deployment Hardware Requirements 12

Introduction to Amazon Web Services 14

AWS VPC Traffic Mirroring Deployment 15

AWS NAT Gateway Deployment 16

Set up a NAT Gateway 17

Install NNM on the NAT Gateway 23

Example Deployment 24

Introduction to Google Cloud Platform Compute Engine 26

GCP VPC Packet Mirroring Deployment 27

GCP NAT Gateway Deployment 28

Set up a Google Cloud Platform Project 29

Set up a NAT Gateway 30

Install NNM on the NAT Gateway 34

Example Deployment 35

Introduction to Microsoft Azure 37

Azure Virtual Network TAP Deployment 38

Azure NAT Gateway Deployment 39

Set up a NAT Gateway 40

Install NNM on the NAT Gateway 52

Example Deployment 53

Example Load Balancing Deployment 55
Set up a NAT Gateway with Load Balancing 57
Introduction to Docker 74
Docker Software Requirements 75
Configure NNM in a Docker Container 76
Monitored Interfaces 77
Monitored Interfaces Examples 79
Configure NNM on the Docker Host 84
Introduction to Gigamon 85
SSL Decryption with NNM 86
Configure SSL Server 88
Configure Gigamon 89
Configure the Tool Port 90
Configure the GS Group 92
Configure the GS Operation 95
Configure the SSL Keychain Password 97
Upload the Private Key 99
Create an SSL Service 101
Create a Map 103
Example Gigamon Deployment 106
Introduction to Waterfall 107
Waterfall Integration Test 108
Test Goals 109
Test Method 110

Test Results 111
Waterfall Architecture and Data Flow 112
Install Waterfall 113
Tenable Channel 114
Configure the TX Agent 115
Configure the RX Agent 116
Additional Waterfall Support 117
Introduction to Splunk **118**
DHCP Setup and Configuration 120
Install the Splunk Universal Log Forwarder 122
SIEM Pull Service Queries 125
VMWare ERSPAN **135**
Virtual Switches for Use with NNM **137**
Basic NNM VM Configuration 138
Platforms 139
VMWare ESXi - Desktop Client 140
VMWare vSphere - Flash Web User Interface 148
VMWare vSphere - HTML5 Web User Interface 154
Microsoft Hyper-V 165
Pass Data from an External Source 171
Pass External Data through Microsoft Hyper-V 172

Welcome to the NNM Deployment Guide

This guide includes the following deployment information:

- Amazon Web Services
- Docker
- Gigamon
- Google Cloud Platform
- Microsoft Azure
- Splunk
- Waterfall
- VMWare ERSPAN
- Virtual Switches

Cloud Deployments

To deploy Tenable Nessus Network Monitor in a cloud deployment, see the following:

- Cloud Deployment Software Requirements
- Cloud Deployment Hardware Requirements

You can deploy Tenable Nessus Network Monitor on the following cloud platforms:

- AWS
- Google Cloud Platform
- Microsoft Azure

Cloud Deployment Software Requirements

Note: Standard support for Tenable Nessus Network Monitor 5.12 ends 09/30/2022. Tenable recommends updating to Tenable Nessus Network Monitor 6.0.0 or later. Otherwise, you will not be able to report issues and bugs. Users that connect to Tenable Vulnerability Management using a web proxy need to upgrade to Tenable Nessus Network Monitor 6.1.1.

Tenable Nessus Network Monitor is available for the following platforms:

Version	Software Requirements
6.2.x	• Red Hat Linux ES 7 / CentOS 7 (through 7.9) 64-bit • Red Hat Linux ES 8 / CentOS 8 (through 8.7) 64-bit • Red Hat Linux ES 9 64-bit **Note:** For all versions of Red Hat Linux ES and CentOS, Tenable Nessus Network Monitor requires that you have `systemd` and `firewalld` on your system. • Microsoft Windows 7, 8, 10, Server 2012, Server 2016, and Server 2019 64-bit **Note:** Tenable Nessus Network Monitor requires Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019. You must download the specific package `vc_redist.x64.exe` from the Microsoft downloads site. **High Performance mode only available on:** • RH7/CentOS7 (RH 7.0 through RH 7.9) : 3.10.0-1160 • RH8/CentOS8 (RH 8.0 through 8.5): 4.18.0-348 • RH8 (RH 8.6-8.7): 4.18.0-425
6.1.x	• Red Hat Linux ES 7 / CentOS 7 (through 7.9) 64-bit **Note:** For this version, Tenable Nessus Network Monitor requires that you have `systemd` and `firewalld` on your system. • Red Hat Linux ES 8 / CentOS 8 (through 8.5) 64-bit

	Note: For this version, Tenable Nessus Network Monitor requires that you have `systemd` and `firewalld` on your system. • Microsoft Windows 7, 8, 10, Server 2008, Server 2012, Server 2016, and Server 2019 64-bit **Note:** Tenable Nessus Network Monitor requires Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019. You must download the specific package `vc_redist.x64.exe` from the Microsoft downloads site. **High Performance mode only available on:** • RH7/CentOS7 (RH 7.0 through RH7.4) : 3.10.0-693 • RH7/CentOS7 (RH 7.5): 3.10.0-862 • RH7/CentOS7 (RH 7.6): 3.10.0-957 • RH7/CentOS7 (RH 7.7): 3.10.0-1062 • RH7/CentOS7 (RH 7.8): 3.10.0-1127 • RH7/CentOS7 (RH 7.9): 3.10-1160 • RH8/CentOS8 (RH 8.0 through 8.5): 4.18.0-348
6.0.x	• Red Hat Linux ES 7 / CentOS 7 (through 7.9) 64-bit **Note:** For this version, Tenable Nessus Network Monitor requires that you have `systemd` and `firewalld` on your system. • Red Hat Linux ES 8 / CentOS 8 (through 8.5) 64-bit **Note:** For this version, Tenable Nessus Network Monitor requires that you have `systemd` and `firewalld` on your system. • macOS 10.9-10.13 64-bit • Microsoft Windows 7, 8, 10, Server 2008, Server 2012, Server 2016, and Server 2019 64-bit **Note:** Tenable Nessus Network Monitor requires Microsoft Visual C++ Redis-

	tributable for Visual Studio 2015, 2017 and 2019. You must download the specific package `vc_redist.x64.exe` from the Microsoft downloads site. **High Performance mode only available on:** • RH7/CentOS7 (RH 7.0 through RH7.4) : 3.10.0-693 • RH7/CentOS7 (RH 7.5): 3.10.0-862 • RH7/CentOS7 (RH 7.6): 3.10.0-957 • RH7/CentOS7 (RH 7.7): 3.10.0-1062 • RH7/CentOS7 (RH 7.8): 3.10.0-1127 • RH7/CentOS7 (RH 7.9): 3.10-1160 • RH8/CentOS8 (RH 8.0 through 8.5): 4.18.0-348
Previous Versions	
5.13.x	• Red Hat Linux ES/ CentOS 64-bit • Red Hat Linux ES 7 / CentOS 7 (through 7.9) 64-bit **Note:** For this version, Tenable Nessus Network Monitor requires that you have `systemd` and `firewalld` on your system. • Red Hat Linux ES 8 (through 8.3) 64-bit **Note:** For this version, Tenable Nessus Network Monitor requires that you have `systemd` and `firewalld` on your system. • macOS 10.9-10.13 64-bit • Microsoft Windows 7, 8, 10, Server 2008, Server 2012, Server 2016, and Server 2019 64-bit **Note:** Tenable Nessus Network Monitor requires Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019. You must download the specific package `vc_redist.x64.exe` from the Microsoft downloads site.

	High Performance mode only available on: • RH6/CentOS6 (RH 6.0 through RH6.9) : 2.6.32-696 • RH7/CentOS7 (RH 7.0 through RH7.4) : 3.10.0-693 • RH7/CentOS7 (RH 7.5): 3.10.0-862 • RH7/CentOS7 (RH 7.6): 3.10.0-957 • RH7/CentOS7 (RH 7.7): 3.10.0-1062 • RH7/CentOS7 (RH 7.8): 3.10.0-1127 • RH7/CentOS7 (RH 7.9): 3.10-1160 • RH8/CentOS8 (RH 8.0 through 8.5): 4.18.0-240
5.12.x	• Red Hat Linux ES 5 • Red Hat Linux ES 6 / CentOS 6 64-bit • Red Hat Linux ES 7 / CentOS 7 64-bit (through 7.8) **Note:** For this version, Tenable Nessus Network Monitor requires that you have `systemd` and `firewalld` on your system. • macOS 10.9-10.13 64-bit • Microsoft Windows 7, 8, 10, Server 2008, Server 2012, and Server 2016 64-bit OS • Microsoft Visual C++ 2010 Redistributable Package **High Performance mode only available on:** • RH6/CentOS6 (RH6.0 thru RH6.9) : 2.6.32-696 • RH7/CentOS7 (RH7.0 thru RH7.4) : 3.10.0-693 • RH7/CentOS7 (RH7.5): 3.10.0-862 • RH7/CentOS7 (RH 7.6): 3.10.0-957

	• RH7/CentOS7 (RH 7.7): 3.10.0-1062 • RH7/CentOS7 (RH 7.8): 3.10.0-1127

You can use ERSPAN to mirror traffic from one or more source ports on a virtual switch, physical switch, or router and send the traffic to a destination IP host running Tenable Nessus Network Monitor. Tenable Nessus Network Monitor supports the following ERSPAN virtual environments:

- VMware ERSPAN (Transparent Ethernet Bridging)
- Cisco ERSPAN (ERSPAN Type II)

Tip: Refer to the Configuring Virtual Switches for Use with Tenable Nessus Network Monitor document for details on configuring your virtual environment.

High Performance Mode

To run Tenable Nessus Network Monitor in High Performance mode, you must enable HugePages support. HugePages is a performance feature of the Linux kernel and is necessary for the large memory pool allocation used for packet buffers. If your Linux kernel does not have HugePages configured, Tenable Nessus Network Monitor automatically configures HugePages per the appropriate settings. Otherwise, if your Linux kernel has defined HugePages, refer to the Configuring HugePages instructions in the Linux Command Line Operations section.

Cloud Deployment Hardware Requirements

Enterprise networks can vary in performance, capacity, protocols, and overall activity. Resource requirements to consider for Tenable Nessus Network Monitor deployments include raw network speed, the size of the network being monitored, and the configuration of Tenable Nessus Network Monitor.

The following chart outlines some basic hardware requirements for operating Tenable Nessus Network Monitor:

Version	Installation scenario	RAM	Processor	Hard Disk
All Versions	Tenable Nessus Network Monitor managing up to 50,000 hosts * (**)	2 GB RAM (4 GB RAM recommended)	2 2GHz cores	20 GB HDD minimum
	Tenable Nessus Network Monitor managing more than 50,000 hosts **	4 GB RAM (8 GB RAM recommended)	4 2GHz cores	20 GB HDD minimum
	Tenable Nessus Network Monitor running in High Performance mode	16 GB RAM (HugePages memory: 2 GB)	10 2GHz cores with hyper-threading enabled	20 GB HDD minimum

*The ability to monitor a given number of hosts depends on the bandwidth, memory, and processing power available to the system running Tenable Nessus Network Monitor.

**For optimal data collection, Tenable Nessus Network Monitor must be connected to the network segment via a hub, spanned port, or network tap to have a full, continuous view of network traffic.

Note: Research your VM software vendor for comparative recommendations, as VMs typically see up to a 30% loss in efficiency compared to dedicated servers. Tenable Nessus Network Monitor supports VMware's vmxnet3 driver.

High Performance Mode

To run Tenable Nessus Network Monitor in High Performance mode, a minimum of two of the following types of Intel NICs are required; one as a management interface and at least one as a monitoring interface:

- e1000 (82540, 82545, 82546)
- e1000e (82571, 82574, 82583, ICH8.ICH10, PCH.PCH2)
- igb (82575, 82576, 82580, I210, I211, I350, I354, DH89xx)
- ixgbe (82598, 82599, X540, X550)
- i40e (X710, XL710)
- NT40A01-4x1

Introduction to Amazon Web Services

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. The Nessus Network Monitor (NNM) can be installed on a VM running on AWS infrastructure.

Before you deploy Tenable Nessus Network Monitor, ensure you meet the following requirements:

- Cloud Deployment Software Requirements
- Cloud Deployment Hardware Requirements

You can deploy Tenable Nessus Network Monitor on AWS using one of the following methods:

- AWS VPC Traffic Mirroring Deployment
- AWS NAT Gateway Deployment

AWS VPC Traffic Mirroring Deployment

Tenable Nessus Network Monitor can be installed on a VM running on AWS infrastructure using VPC Traffic Mirroring.

To deploy Tenable Nessus Network Monitor on AWS using VPC Traffic Mirroring, see the AWS documentation.

AWS NAT Gateway Deployment

Tenable Nessus Network Monitor can be installed on a VM running on AWS infrastructure as a NAT gateway deployment.

For more information, see the following:

- Set up a NAT Gateway
- Install NNM on the NAT Gateway
- Example Deployment

Set up a NAT Gateway

Introduction

In order for NNM to monitor virtual machine instances in an Amazon Web Services Virtual Private Cloud (VPC), NNM must run on a virtual machine instance that functions as a network address translation (NAT) gateway. A NAT gateway instance routes traffic from internal-only virtual machine instances to the Internet. A NNM installed on a NAT gateway has visibility into the hostnames and private IP addresses of the internal virtual machine instances before the NAT gateway masquerades the source IP address of incoming packets to forward them to the Internet.

This guide shows setting up a NAT gateway in an Amazon Web Services Virtual Private Cloud.

Before You Begin

Follow the Amazon EC2 Setup Instructions. You can skip the steps for creating a VPC and creating a Security Group as they will be covered in the steps below.

Steps

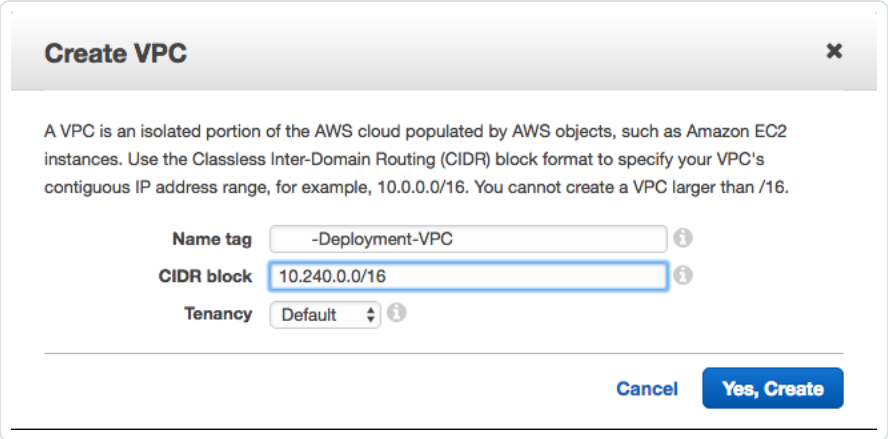
1. From the EC2 management console, click Launch Instance to create a new virtual machine which will be used as the NAT gateway. In this example, a CentOS 6 Amazon Machine Image (AMI) is used.

   > **Note:** If you select a different AMI to install on your NAT gateway virtual machine, ensure that it is a platform that NNM supports.

2. In the **Configure Instance** section of the **Launch Instance** wizard, click the **Create new VPC** button to configure the instance's network.

   The **Create VPC** window appears.

3. Enter the details for the new VPC as shown in the following image.



4. Click **Yes, Create**.

   The window closes and the **Configure Instance** section appears.

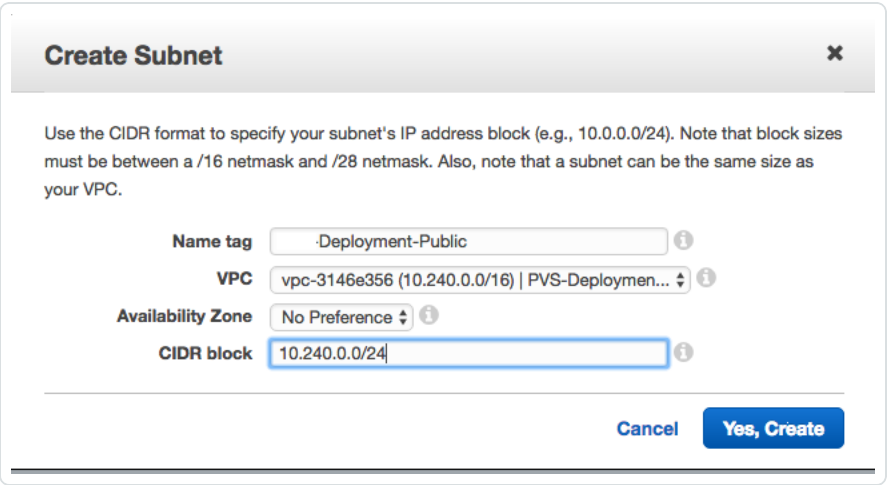
5. Select the VPC you just created in the **Network** drop-down box.

   > **Tip:** You may need to click the **Refresh** button next to the **Create new VPC** button to force the new VPC to appear in the drop-down box.

6. Click the **Create new subnet** button to configure the instance's subnet.

   The **Create Subnet** window appears.

7. Enter the details for the new subnet as shown in the following image. The **VPC** box displays the VPC you created in step 3.



8. Click **Yes, Create**.

The window closes and the **Configure Instance** section appears.

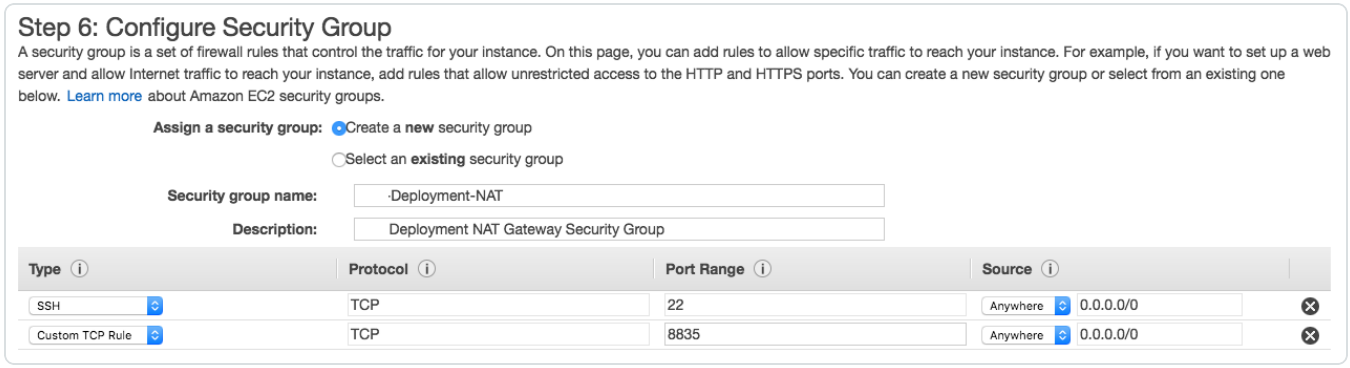
9. Select the subnet you just created in the **Subnet** drop-down box.

   > **Tip:** You may need to click the **Refresh** button next to the **Create new subnet** button to force the new subnet to appear in the drop-down box.

10. In the **Auto-assign Public IP** drop-down box, select **Enable**.
11. In the **Tag Instance** section of the **Launch Instance** wizard, assign a name to the instance.

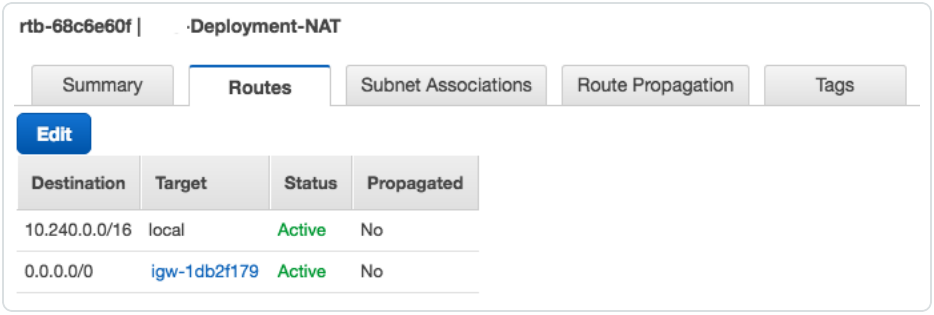
    In this example, the name **NNM-Deployment-NAT** is used.

12. In the **Configure Security Group** section of the **Launch Instance** wizard, create a new security group that allows incoming SSH and TCP port 8835 (the default port for the NNM Web server) connections from anywhere.



13. Proceed through the rest of the virtual machine instance setup and then launch the virtual machine.
14. Create an Internet Gateway to provide the NAT gateway with internet access and attach it to the VPC created in Step 3. In this example, the created Internet Gateway is attached to **NNM-Deployment-VPC**.
15. Create a route table for the VPC created in Step 3. Then, create a default route using the Internet Gateway created in Step 12 as the target. In this example, the Internet Gateway is **igw-1db2f179**.

    Associate the route table with the subnet created in Step 7. In this example, the route table is created for **NNM-Deployment-VPC** and is associated with the **NNM-Deployment-Public** subnet.



16. Connect to the new NAT gateway instance using the public IP that was automatically assigned.

17. Once logged into your NAT gateway instance, configure iptables and IP forwarding.

```
user@nat-gateway:~$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

> **Note:** Some CentOS instances have existing iptables rules that should be flushed before executing the iptables command below to avoid conflicting rules. Execute the following to flush iptables rules: **sudo iptables -F**

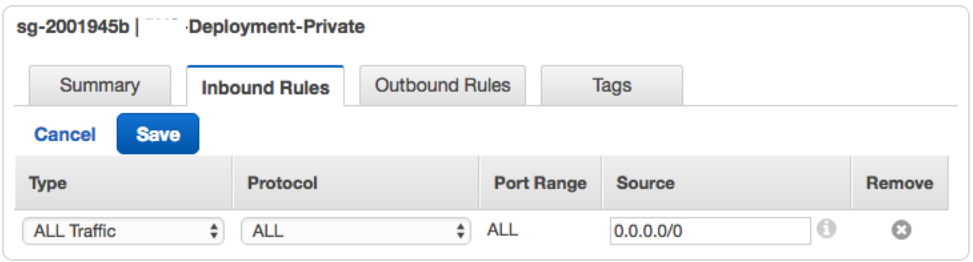
```
user@nat-gateway:~$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

The first **sudo** command tells the kernel to allow IP forwarding. The second **sudo** command masquerades packets received from internal instances as if they originated from the NAT gateway instance.

> **Tip:** Consider saving these commands in a startup script, because these settings will not persist if the instance is rebooted.

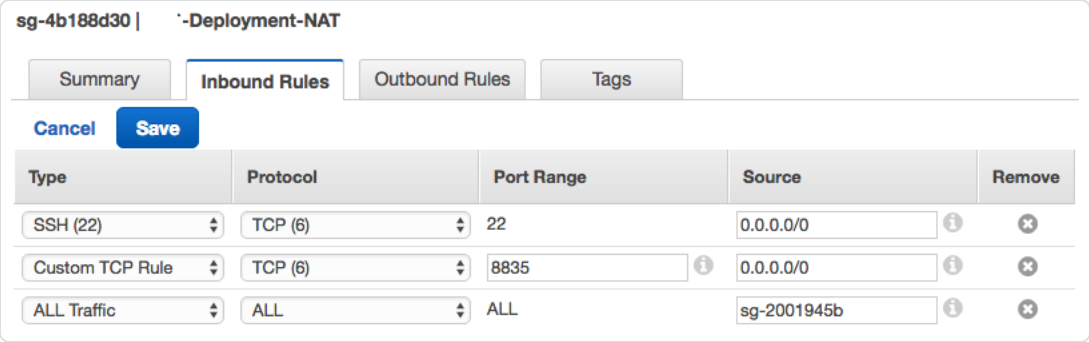
18. Create a security group for the private subnet and associate it with the VPC created in Step 3.

19. Create an inbound rule for the new security group to allow all traffic.

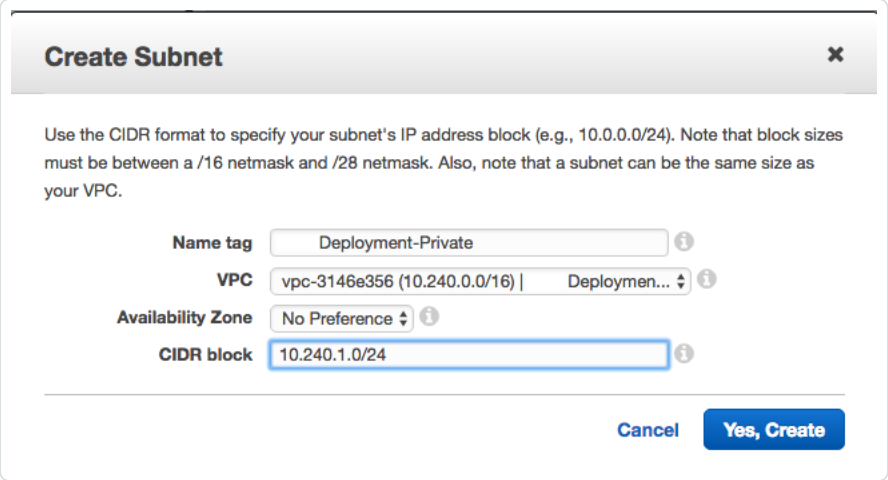


20. Update the security group that was created in Step 18 to allow all traffic from the new private network security group.

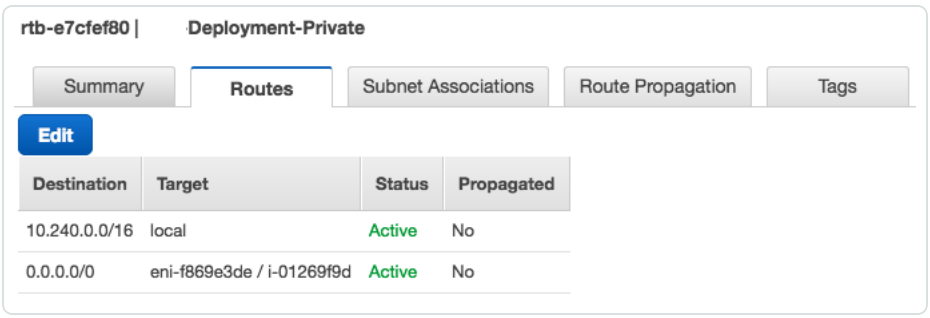
    In this example, the **NNM-Deployment-NAT** security group is updated to allow all traffic from the **NNM-Deployment-Private** security group using its ID **sg-2001945b** as the source.



21. Create a private subnet for the hosts that will not have public IP addresses and will access the internet through the NAT gateway.



22. Create a route table for the VPC created in Step 2. Then create a default route using the NNM-Deployment-NAT instance as the target. In this example the NNM-Deployment-NAT instance is **i-01269f9d**. Associate the route table with the subnet created in Step 21. In this example, the route table is created for **NNM-Deployment-VPC** and is associated with the **NNM-Deployment-Private** subnet.



23. Disable source/destination checks on the NAT gateway instance.

24. Launch example instances into the private subnet. Use the VPC created in Step 3 as the network and the subnet created in Step 21 for the subnet. In this example, the **NNM-Deployment-VPC** is used as the network and the **NNM-Deployment-Private** subnet is used as the subnet.

    For the **Auto-assign Public IP** setting, select **Use subnet setting** from the drop-down. For the security group, select the security group that was created in Step 18. In this example, the **NNM-Deployment-Private** security group is used.

Install NNM on the NAT Gateway

Before You Begin

These steps assume that you have set up a NAT gateway in an Amazon Web Services Virtual Private Cloud.

The NNM installer package for your NAT gateway instance's platform can be downloaded from the Tenable Downloads page.

Steps

1. Copy the NNM installer package to the home directory in your NAT gateway instance.
2. Log in to your NAT gateway instance.
3. Once logged into your NAT gateway instance, install NNM.

   Once NNM is installed and running on the NAT gateway, you can access the NNM web front end by navigating to **https://<external IP address of nat-gateway>:8835** in your web browser.

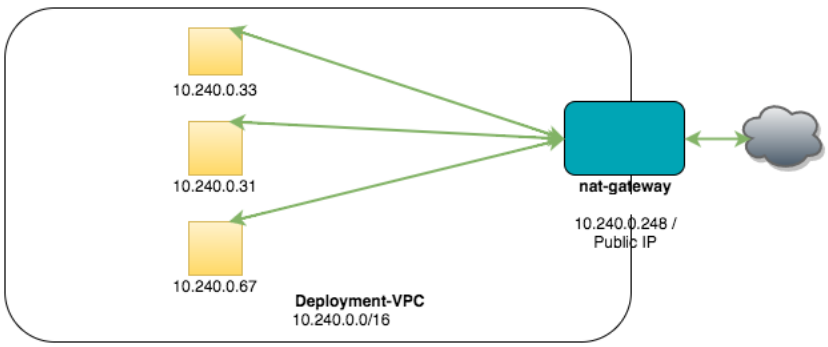
Example Deployment

This section demonstrates an example of NNM running on a virtual machine functioning as a NAT gateway instance within an Amazon Web Services Virtual Private Cloud (VPC).

In the examples used in the instructions for setting up a NAT gateway, the VPC **NNM-Deployment-VPC** was created, which has the network range **10.240.0.0/16**. Additionally, the virtual machine instance **NNM-Deployment-NAT** was created in the **NNM-Deployment-Public** subnet to function as the NAT gateway. In this example, three other virtual machine instances were created within the **NNM-Deployment-Private** subnet. None of the virtual machine instances in **NNM-Deployment-Private** are assigned an external IP address and all outgoing traffic is routed through **NNM-Deployment-NAT**.

In this example, there are four virtual machine instances within **NNM-Deployment-VPC**:

VM Instance Name	Internal IP	Has External IP?
NNM-Deployment-NAT	10.240.0.248	Yes
example-instance	10.240.1.33	No
example-instance2	10.240.1.67	No
example-instance3	10.240.1.31	No



NNM is running on **NNM-Deployment-NAT** and has the following configuration:

Configuration Parameter	Value
Monitored Network Interfaces	eth0
Monitored Network IP Addresses and Ranges	10.240.0.0/16

With this configuration, NNM will monitor traffic

- from the internal virtual machine instances to the Internet,
- between **NNM-Deployment-NAT** and the internal virtual machine instances,
- from the Internet to internal virtual machine instances if you have enabled port forwarding on the NAT gateway to make them Internet accessible,
- and between **NNM-Deployment-NAT** and the Internet.

> **Note:** Due to the design of the hypervisor used by Amazon for running all virtual instances, traffic not addressed to a virtual instance can't be sniffed by the virtual instance. As a result, NNM can't monitor traffic between other virtual instances.

Introduction to Google Cloud Platform Compute Engine

Google Cloud Platform's Compute Engine lets you create and run virtual machines on Google infrastructure. With Google Compute Engine, you can run thousands of virtual CPUs on a system that has been designed to be fast, and to offer strong consistency of performance.

The Nessus Network Monitor (NNM) can be installed on a VM running on Google infrastructure.

Before you deploy Tenable Nessus Network Monitor, ensure you meet the following requirements:

- Cloud Deployment Software Requirements
- Cloud Deployment Hardware Requirements

You can deploy Tenable Nessus Network Monitor on GCP using one of the following methods:

- GCP VPC Packet Mirroring Deployment
- GCP NAT Gateway Deployment

GCP VPC Packet Mirroring Deployment

Tenable Nessus Network Monitor can be installed on a VM running on GCP infrastructure using VPC Packet Mirroring.

To deploy Tenable Nessus Network Monitor on AWS using VPC Packet Mirroring, see the GCP Virtual Private Cloud documentation.

GCP NAT Gateway Deployment

Tenable Nessus Network Monitor can be installed on a VM running on GCP infrastructure as a NAT gateway deployment.

For more information, see the following:

- Set up a Google Cloud Platform Project
- Set up a NAT Gateway
- Install NNM on the NAT Gateway
- Example Deployment

Set up a Google Cloud Platform Project

Before You Begin

Python 2.7 must be installed on your local machine to use the Google Cloud SDK tools. The gcloud tool is used in this guide to interact with the Compute Engine API.

Create a Google Account and log in to the Google Cloud Platform Console.

Steps

1. Create a new project in the Google Cloud Platform Console.
2. In the upper left corner of the Console, click  , and then select **API Manager**.
3. In the **Google Cloud APIs** section, select **Compute Engine API**.
4. Click the **Enable** button.

   Caution: In order to use Google Compute Engine, you must enter billing information. Refer to Google's Compute Engine Pricing article for more information.

5. Install and initialize the Google Cloud SDK for your local machine's platform by referencing the **Before you Begin** and **Initialize the SDK** sections in one of the following links:

   - Instructions for Linux
   - Instructions for Debian and Ubuntu
   - Instructions for Mac OS X
   - Instructions for Windows

   Note: Make note of which compute zone you choose during the initialization of Google Cloud SDK. The examples in this guide use the zone **us-east1-b**. Make sure to substitute with the zone you chose if it's different than the one used in this guide.

Set up a NAT Gateway

Introduction

In order for NNM to monitor virtual machine instances in a Google Compute Engine network, NNM must run on a virtual machine instance that functions as a network address translation (NAT) gateway. A NAT gateway instance routes traffic from internal-only virtual machine instances to the Internet. A NNM installed on a NAT gateway has visibility into the hostnames and private IP addresses of the internal virtual machine instances before the NAT gateway masquerades the source IP address of incoming packets to forward them to the Internet.

This guide shows setting up a NAT gateway in a Google Compute Engine legacy network. Network ranges must be adjusted if you're using a subnetwork.

Before You Begin

Follow the instructions on setting up a Google Cloud Platform project.

Steps

1. Create a Compute Engine network to host your virtual machine instances. In this example, the legacy network range used is **10.240.0.0/16** with a gateway of **10.240.0.1**. You can select your own IPv4 range and gateway addresses as needed. You can also create a subnetwork instead.

   > If you want to use the default network, you can skip this step and replace **gce-network** in the examples below with **default**.

   ```
   $ gcloud compute networks create gce-network --range 10.240.0.0/16 --mode=legacy

   Created [https://www.googleapis.com/compute/v1/projects/nnm-example-
   project/global/networks/gce-network].
   NAME        MODE    IPV4_RANGE    GATEWAY_IPV4
   gce-network legacy 10.240.0.0/16 10.240.0.1
   ```

```
Instances on this network will not be reachable until firewall rules are created.
As an example, you can allow all internal traffic between instances as well as
SSH, RDP, and ICMP by running:
$ gcloud compute firewall-rules create <FIREWALL_NAME> --network gce-network --
allow tcp,udp,icmp --source-ranges <IP_RANGE>
$ gcloud compute firewall-rules create <FIREWALL_NAME> --network gce-network --
allow tcp:22,tcp:3389,icmp
```

2. Create firewall rules to allow SSH connections in the new network you just created.

```
$ gcloud compute firewall-rules create gce-network-allow-ssh --allow tcp:22 --
network gce-network

Created [https://www.googleapis.com/compute/v1/projects/nnm-example-
project/global/firewalls/gce-network-allow-ssh].
NAME                  NETWORK     SRC_RANGES RULES   SRC_TAGS TARGET_TAGS
gce-network-allow-ssh gce-network 0.0.0.0/0  tcp:22
```

3. Create firewall rules to allow TCP, UDP, and ICMP traffic within the new network you just created.

```
$ gcloud compute firewall-rules create gce-network-allow-internal --allow tcp:1-
65535,udp:1-65535,icmp --source-ranges 10.240.0.0/16 --network gce-network

Created [https://www.googleapis.com/compute/v1/projects/nnm-example-
project/global/firewalls/gce-network-allow-internal].
NAME                        NETWORK     SRC_
RANGES    RULES                         SRC_TAGS TARGET_TAGS
gce-network-allow-internal gce-network 10.240.0.0/16 tcp:1-65535,udp:1-65535,icmp
```

4. Create a virtual machine instance to act as a NAT gateway on the **gce-network** or the **default** network. In this example, a CentOS 6 virtual machine is created.

> **Note:** If you choose a different image to install on your NAT gateway virtual machine, make sure that it's a platform that NNM supports.

For the following examples, use the zone name that was chosen when setting up the Google Cloud Platform project.

```
$ gcloud compute instances create nat-gateway --network gce-network --can-ip-
forward --zone us-east1-b --image centos-6 --tags nat

Created [https://www.googleapis.com/compute/v1/projects/nnm-example-
project/zones/us-east1-b/instances/nat-gateway].
NAME        ZONE       MACHINE_TYPE  PREEMPTIBLE INTERNAL_IP    EXTERNAL_
IP     STATUS
nat-gateway us-east1-b n1-standard-1             10.240.0.2     104.xxx.xxx.xxx
RUNNING
```

5. Tag any virtual machine instances without an external IP address that will use the gateway instance with the tag **no-ip**, or create a new virtual machine without an external IP address and tag the instance with the **no-ip** tag.

```
# Add tags to an existing instance ...
$ gcloud compute instances add-tags existing-instance --tags no-ip

Updated [https://www.googleapis.com/compute/v1/projects/nnm-example-
project/zones/us-east1-b/instances/existing-instance].
```

```
# Or create a new virtual machine without an external IP address
$ gcloud compute instances create example-instance --network gce-network --no-
address --zone us-east1-b --image centos-6 --tags no-ip

Created [https://www.googleapis.com/compute/v1/projects/nnm-example-
project/zones/us-east1-b/instances/example-instance].
NAME             ZONE       MACHINE_TYPE  PREEMPTIBLE INTERNAL_IP EXTERNAL_IP
STATUS
example-instance us-east1-b n1-standard-
1             10.240.0.3              RUNNING
```

6. Create a route to send traffic destined to the Internet through your gateway instance.

```
$ gcloud compute routes create no-ip-internet-route --network gce-network --
destination-range 0.0.0.0/0 --next-hop-instance nat-gateway --next-hop-instance-
zone us-east1-b --tags no-ip --priority 800

Created [https://www.googleapis.com/compute/v1/projects/nnm-example-
project/global/routes/no-ip-internet-route].
NAME                 NETWORK     DEST_RANGE NEXT_
HOP                          PRIORITY
no-ip-internet-route gce-network 0.0.0.0/0  us-east1-b/instances/nat-gateway 800
```

Setting the priority of this route ensures that this route takes precedence if there are any other conflicting routes. 1000 is the default priority and a value lower than 1000 takes precedent.

7. Log in to your NAT gateway instance.

```
$ gcloud compute ssh nat-gateway --zone us-east1-b
```

8. Once logged into your NAT gateway instance, configure iptables.

```
user@nat-gateway:~$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
user@nat-gateway:~$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

The first **sudo** command tells the kernel to allow IP forwarding. The second **sudo** command masquerades packets received from internal instances as if they originated from the NAT gateway instance.

> **Tip:** Consider saving these commands in a startup script, because these settings will not persist if the instance is rebooted.

Install NNM on the NAT Gateway

Before You Begin

Follow the instructions on setting up a NAT gateway in a Google Compute Engine legacy network.

The NNM installer package for your NAT gateway instance's platform can be downloaded from the Tenable Downloads page.

Steps

1. Copy the NNM installer package to the home directory in your NAT gateway instance.

   ```
   $ gcloud compute copy-files /path/to/nnm-installer nat-gateway:~ --zone us-east1-b
   ```

2. Create a firewall rule to allow incoming connections to the NNM Web server. By default, the NNM Web server listens on port 8835.

   ```
   $ gcloud compute firewall-rules create gce-network-allow-nnm-www --allow tcp:8835
   --network gce-network

   Created [https://www.googleapis.com/compute/v1/projects/nnm-example-
   project/global/firewalls/gce-network-allow-nnm-www].
   NAME                       NETWORK     SRC_RANGES RULES    SRC_TAGS TARGET_TAGS
   gce-network-allow-nnm-www gce-network 0.0.0.0/0  tcp:8835
   ```

3. Log in to your NAT gateway instance.

   ```
   $ gcloud compute ssh nat-gateway --zone us-east1-b
   ```

4. Once logged into your NAT gateway instance, install NNM.

   Once NNM is installed and running on the NAT gateway, you may access the NNM web front end by navigating to **https://<external IP address of nat-gateway>:8835** in your Web browser. The external IP address of **nat-gateway** can be found by executing **gcloud compute instances describe nat-gateway** and looking for **networkInterfaces > accessConfigs > natIP**.

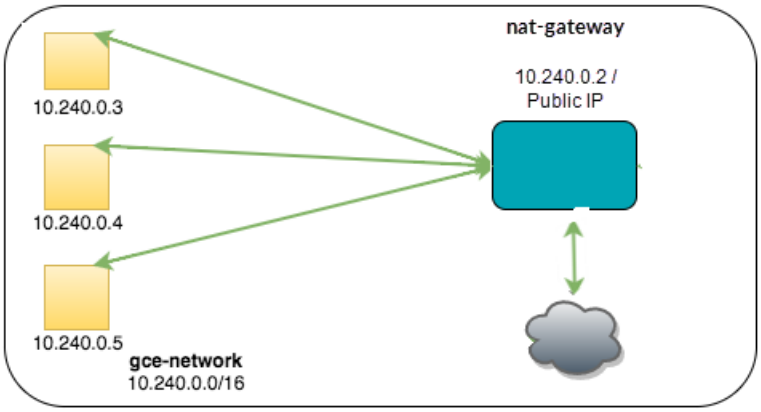
Example Deployment

This section demonstrates an example of NNM running on a virtual machine functioning as a NAT gateway instance within a Google Cloud Platform Compute Engine legacy network.

In the examples used in the instructions for setting up a NAT gateway, the Compute Engine legacy network **gce-network** was created, which has the network range **10.240.0.0/16**. Additionally, the virtual machine instance **nat-gateway** was created to function as the NAT gateway in **gce-network**. In this example, three other virtual machine instances were created with the **--no-address** flag and bound to the tag **no-ip**, so none of the virtual machine instances are assigned an external IP address and all outgoing traffic is routed to **nat-gateway**. as a result of the **no-ip-internet-route** rule that was created.

In this example, there are four virtual machine instances within **gce-network**:

VM Instance Name	Internal IP	Has External IP?
nat-gateway	10.240.0.2	Yes
example-instance	10.240.0.3	No
centos-instance	10.240.0.4	No
windows-instance	10.240.0.5	No



NNM is running on **nat-gateway** and has the following configuration:

Configuration Parameter	Value
Monitored Network Interfaces	eth0

Monitored Network IP Addresses and Ranges	10.240.0.0/16

With this configuration, NNM monitors traffic:

- from the internal virtual machine instances to the Internet,
- between **nat-gateway** and the internal virtual machine instances,
- and between **nat-gateway** and the Internet.

Note: The routing of packets destined for the **gce-network** legacy network cannot be changed. As a result, there is no way to configure forwarding of traffic between two internal virtual machine instances through **nat-gateway**.

Introduction to Microsoft Azure

Microsoft Azure is a collection of integrated cloud services for building, deploying, and managing applications and services. Nessus Network Monitor (NNM) can be installed on a VM running on Azure infrastructure.

Before you deploy Tenable Nessus Network Monitor, ensure you meet the following requirements:

- Cloud Deployment Software Requirements
- Cloud Deployment Hardware Requirements

You can deploy Tenable Nessus Network Monitor on Azure using one of the following methods:

- Azure Virtual Network TAP Deployment
- Azure NAT Gateway Deployment

Azure Virtual Network TAP Deployment

You can deploy Tenable Nessus Network Monitor on a VM running on Azure infrastructure using a virtual network Terminal Access Point (TAP).

To deploy Tenable Nessus Network Monitor on Azure using a virtual network TAP, see the Microsoft Azure documentation.

Azure NAT Gateway Deployment

Tenable Nessus Network Monitor can be installed on a VM running on Microsoft Azure infrastructure as a NAT gateway deployment.

For more information, see the following:

- Set up a NAT Gateway
- Install NNM on the NAT Gateway
- Example Deployment
- Example Load Balancing Deployment
- Set up a NAT Gateway with Load Balancing

Set up a NAT Gateway

Introduction

In order for NNM to monitor virtual machine instances in a Microsoft Azure Virtual Network, NNM must run on a virtual machine instance that functions as a network address translation (NAT) gateway. A NAT gateway instance routes traffic from internal-only virtual machine instances to the Internet. A NNM installed on a NAT gateway has visibility into the hostnames and private IP addresses of the internal virtual machine instances before the NAT gateway masquerades the source IP address of incoming packets to forward them to the Internet.

This guide shows setting up a NAT gateway in a Microsoft Azure Virtual Network.

Before You Begin

Follow the Azure CLI Installation Instructions. Then connect to your subscription from the CLI.

> **Tip:** If you encounter an error in the Azure CLI about the your subscription not being registered to use a namespace, see this section on the common deployment errors page.

Steps

1. Enable **Azure CLI Resource Manager** commands.

   ```
   azure config mode arm
   ```

2. Create a resource group.

   In this example, the resource group **azureNNM** is created.

   ```
   azure group create azureNNM eastus
   info:    Executing command group create
   + Getting resource group azureNNM
   + Creating resource group azureNNM
   info:    Created resource group azureNNM
   data:    Name:                azureNNM
   data:    Location:            eastus
   ```

```
data:    Provisioning State:  Succeeded
data:    Tags: null
data:
info:    group create command OK
```

3. Create a storage account in the resource group **azureNNM**.

   In this example, the storage group **nnmstore** is created.

```
azure storage account create --location eastus --resource-group azureNNM  --kind
Storage --sku-name GRS nnmstore
info:    Executing command storage account create
+ Checking availability of the storage account name
+ Creating storage account
info:    storage account create command OK
```

4. Create a Virtual Network in the resource group **azureNNM**.

   In this example, the Virtual Network is **nnmVNet** and has the network range **10.240.0.0/16**.

```
azure network vnet create -g azureNNM -n nnmVNet -a 10.240.0.0/16 -l eastus
info:    Executing command network vnet create
+ Looking up the virtual network "nnmVNet"
+ Creating virtual network "nnmVNet"
data:    Name                           : nnmVNet
data:    Type                           : Microsoft.Network/virtualNetworks
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
data:    Address prefixes:
data:      10.240.0.0/16
info:    network vnet create command OK
```

5. Create a public subnet for the NAT gateway.

   In this example, the public subnet is **nnmPublic** and has the network range **10.240.0.0/24**.

```
azure network vnet subnet create -g azureNNM -e nnmVNet -n nnmPublic -a
10.240.0.0/24
info:    Executing command network vnet subnet create
```

```
+ Looking up the virtual network "nnmVNet"
+ Looking up the subnet "nnmPublic"
+ Creating subnet "nnmPublic"
data:    Name                           : nnmPublic
data:    Provisioning state             : Succeeded
data:    Address prefix                 : 10.240.0.0/24
info:    network vnet subnet create command OK
```

6. Create a public IP and sub domain name for the NAT gateway.

   In this example, the sub domain name is **examplesubdomain** and the public IP is **nnmPIP**.

```
azure network public-ip create -d examplesubdomain azureNNM nnmPIP eastus
info:    Executing command network public-ip create
warn:    Using default --idle-timeout 4
warn:    Using default --allocation-method Dynamic
warn:    Using default --ip-version IPv4
+ Looking up the public ip "nnmPIP"
+ Creating public ip address "nnmPIP"
data:    Name                           : nnmPIP
data:    Type                           : Microsoft.Network/publicIPAddresses
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
data:    Allocation method              : Dynamic
data:    IP version                     : IPv4
data:    Idle timeout in minutes        : 4
data:    Domain name label              : examplesubdomain
data:    FQDN                           :
examplesubdomain.eastus.cloudapp.azure.com
info:    network public-ip create command OK
```

7. Create a NIC for the NAT gateway and associate it with the public IP **nnmPIP** and public subnet **nnmPublic**.

   In this example, the new NIC is **nnmNatNic**.

```
azure network nic create --public-ip-name nnmPIP --subnet-name nnmPublic --subnet-
vnet-name nnmVNet  azureNNM nnmNatNic eastus
info:    Executing command network nic create
```

```
+ Looking up the network interface "nnmNatNic"
+ Looking up the subnet "nnmPublic"
+ Looking up the public ip "nnmPIP"
+ Creating network interface "nnmNatNic"
data:    Name                              : nnmNatNic
data:    Type                              : Microsoft.Network/networkInterfaces
data:    Location                          : eastus
data:    Provisioning state                : Succeeded
data:    Internal domain name suffix       :
gqhqyfrlprbu3jyndjoq4ap5se.bx.internal.cloudapp.net
data:    Enable IP forwarding              : false
data:    IP configurations:
data:      Name                            : default-ip-config
data:      Provisioning state              : Succeeded
data:      Private IP address              : 10.240.0.4
data:      Private IP version              : IPv4
data:      Private IP allocation method    : Dynamic
data:
info:    network nic create command OK
```

8. Enable IP forwarding on the new interface **nnmNatNic**.

```
azure network nic set -g azureNNM -n nnmNatNic -f true
info:    Executing command network nic set
+ Looking up the network interface "nnmNatNic"
+ Updating network interface "nnmNatNic"
data:    Name                              : nnmNatNic
data:    Type                              : Microsoft.Network/networkInterfaces
data:    Location                          : eastus
data:    Provisioning state                : Succeeded
data:    MAC address                       : 00-0D-3A-13-27-48
data:    Internal domain name suffix       :
gqhqyfrlprbu3jyndjoq4ap5se.bx.internal.cloudapp.net
data:    Enable IP forwarding              : true
data:    IP configurations:
data:      Name                            : default-ip-config
data:      Provisioning state              : Succeeded
data:      Private IP address              : 10.240.0.4
data:      Private IP version              : IPv4
```

```
data:      Private IP allocation method  : Dynamic
data:
info:    network nic set command OK
```

9. Create a private subnet for the instances that will not have a public IP address.

   In this example, the private subnet is **nnmPrivate**.

```
azure network vnet subnet create -g azureNNM -e nnmVNet -n nnmPrivate -a
10.240.1.0/24
info:    Executing command network vnet subnet create
+ Looking up the virtual network "nnmVNet"
+ Looking up the subnet "nnmPrivate"
+ Creating subnet "nnmPrivate"
data:    Name                           : nnmPrivate
data:    Provisioning state             : Succeeded
data:    Address prefix                 : 10.240.1.0/24
info:    network vnet subnet create command OK
```

10. Create a security group for the NAT gateway.

    In this example, the security group is **nnmPublicNSG**.

```
azure network nsg create azureNNM nnmPublicNSG eastus
info:    Executing command network nsg create
+ Looking up the network security group "nnmPublicNSG"
+ Creating a network security group "nnmPublicNSG"
data:    Name                           : nnmPublicNSG
data:    Type                           : Microsoft.Network/networkSecurityGroups
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
data:    Security rules:
data:    Name                          Source IP          Source Port
Destination IP  Destination Port  Protocol  Direction  Access  Priority
data:    ----------------------------  -----------------  -----------  ----------
----  ---------------  --------  ---------  ------  --------
data:    AllowVnetInBound              VirtualNetwork     *
VirtualNetwork  *                *         Inbound    Allow   65000
```

```
data:    AllowAzureLoadBalancerInBound  AzureLoadBalancer  *           *
     *                *         Inbound    Allow   65001
data:    DenyAllInBound                 *                  *           *
     *                *         Inbound    Deny    65500
data:    AllowVnetOutBound              VirtualNetwork     *
VirtualNetwork  *                *         Outbound   Allow   65000
data:    AllowInternetOutBound          *                  *           Internet
     *                *         Outbound   Allow   65001
data:    DenyAllOutBound                *                  *           *
     *                *         Outbound   Deny    65500
info:    network nsg create command OK
```

11. Create a rule in the **nnmPublicNSG** to allow SSH to the NAT gateway.

    In this example, the new rule is called **SSHRule** and the rule has a priority of 1000. This gives it precedence over the existing rules seen in the previous step.

```
azure network nsg rule create --protocol tcp --direction inbound --priority 1000 -
-destination-port-range 22 --access allow azureNNM nnmPublicNSG SSHRule
info:    Executing command network nsg rule create
warn:    Using default --source-port-range *
warn:    Using default --source-address-prefix *
warn:    Using default --destination-address-prefix *
+ Looking up the network security group "nnmPublicNSG"
+ Looking up the network security rule "SSHRule"
+ Creating a network security rule "SSHRule"
data:    Name                           : SSHRule
data:    Type                           :
Microsoft.Network/networkSecurityGroups/securityRules
data:    Provisioning state             : Succeeded
data:    Source IP                      : *
data:    Source Port                    : *
data:    Destination IP                 : *
data:    Destination Port               : 22
data:    Protocol                       : Tcp
data:    Direction                      : Inbound
data:    Access                         : Allow
data:    Priority                       : 1000
info:    network nsg rule create command OK
```

12. Create a rule in the **nnmPublicNSG**to allow all traffic to the NAT gateway from within the virtual network.

    In this example, the new rule is called **PrivateToPublicRule**and the rule has a priority of 1001. This gives it precedence over the existing rules that disallow traffic.

```
azure network nsg rule create --direction inbound --priority 1001  --source-
address-prefix VirtualNetwork --destination-port-range 0-65535 --access allow
azureNNM nnmPublicNSG PrivateToPublicRule
info:    Executing command network nsg rule create
warn:    Using default --protocol *
warn:    Using default --source-port-range *
warn:    Using default --destination-address-prefix *
+ Looking up the network security group "nnmPublicNSG"
+ Looking up the network security rule "PrivateToPublicRule"
+ Creating a network security rule "PrivateToPublicRule"
data:    Name                           : PrivateToPublicRule
data:    Type                           :
Microsoft.Network/networkSecurityGroups/securityRules
data:    Provisioning state             : Succeeded
data:    Source IP                      : VirtualNetwork
data:    Source Port                    : *
data:    Destination IP                 : *
data:    Destination Port               : 0-65535
data:    Protocol                       : *
data:    Direction                      : Inbound
data:    Access                         : Allow
data:    Priority                       : 1001
info:    network nsg rule create command OK
```

13. Create a rule in the **nnmPublicNSG** to allow traffic to the NNM web server from the Internet. The default port is 8835.

    In this example, the new rule is called **NNMWebRule** and the rule has a priority of 1002. This gives it precedence over the existing rules that disallow traffic.

```
azure network nsg rule create --direction inbound --priority 1002 --protocol tcp
--source-address-prefix Internet --destination-port-range 8835 --access allow
azureNNM nnmPublicNSG NnmWebRule
```

```
info:    Executing command network nsg rule create
warn:    Using default --source-port-range *
warn:    Using default --destination-address-prefix *
+ Looking up the network security group "nnmPublicNSG"
+ Looking up the network security rule "NnmWebRule"
+ Creating a network security rule "NnmWebRule"
data:    Name                           : NnmWebRule
data:    Type                           :
Microsoft.Network/networkSecurityGroups/securityRules
data:    Provisioning state             : Succeeded
data:    Source IP                      : Internet
data:    Source Port                    : *
data:    Destination IP                 : *
data:    Destination Port               : 8835
data:    Protocol                       : Tcp
data:    Direction                      : Inbound
data:    Access                         : Allow
data:    Priority                       : 1002
info:    network nsg rule create command OK
```

14. Assign the security group **nnmPublicNSG** to the **nnmNatNic**, which will be used as the interface of the NAT gateway when it is launched.

```
azure network nic set -g azureNNM -n nnmNatNic -o nnmPublicNSG
info:    Executing command network nic set
+ Looking up the network interface "nnmNatNic"
+ Looking up the network security group "nnmPublicNSG"
+ Updating network interface "nnmNatNic"
data:    Name                           : nnmNatNic
data:    Type                           : Microsoft.Network/networkInterfaces
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
data:    Internal domain name suffix    :
gqhqyfrlprbu3jyndjoq4ap5se.bx.internal.cloudapp.net
data:    Enable IP forwarding           : false
data:    IP configurations:
data:      Name                         : default-ip-config
data:      Provisioning state           : Succeeded
data:      Private IP address           : 10.240.0.4
```

```
data:      Private IP version            : IPv4
data:      Private IP allocation method  : Dynamic
data:
info:    network nic set command OK
```

15. Launch the NAT gateway instance.

    In this example, CentOS 7 and the SSH key **azureNNM_id_rsa** are used. If you do not have an SSH key, refer to the Azure documentation for instructions on how to generate a key.

    > **Note:** If you select a different image to install on your NAT gateway virtual machine, ensure that it is a platform that NNM supports.

```
azure vm create --resource-group azureNNM --name nnmNatGateway --location eastus --os-type linux  --nic-name nnmNatNic --vnet-name nnmVNet --vnet-subnet-name nnmPublic --storage-account-name nnmstore --image-urn CentOS --ssh-publickey-file ~/.ssh/azureNNM_id_rsa.pub --admin-username centos
info:    Executing command vm create
+ Looking up the VM "nnmNatGateway"
info:    Verifying the public key SSH file: ~/.ssh/azureNNM_id_rsa.pub
info:    Using the VM Size "Standard_DS1"
info:    The [OS, Data] Disk or image configuration requires storage account
+ Looking up the storage account nnmstore
+ Looking up the NIC "nnmNatNic"
info:    Found an existing NIC "nnmNatNic"
info:    The storage URI 'https://nnmstore.blob.core.windows.net/' will be used for boot diagnostics settings, and it can be overwritten by the parameter input of '--boot-diagnostics-storage-uri'.
+ Creating VM "nnmNatGateway"
info:    vm create command OK
```

16. Connect to the new NAT gateway instance using the public DNS name that was created when your public IP was created.

```
ssh -i ~/.ssh/azureNNM_id_rsa.pub centos@example-subdomain.eastus.cloudapp.azure.com
```

17. Once logged into your NAT gateway instance, configure iptables and IP forwarding.

```
user@nat-gateway:~$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

```
user@nat-gateway:~$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

The first **sudo** command tells the kernel to allow IP forwarding. The second **sudo** command masquerades packets received from internal instances as if they originated from the NAT gateway instance.

> **Tip:** Consider saving these commands in a startup script, because these settings will not persist if the instance is rebooted.

18. Create a route table for the private subnet.

    In this example, the route table is **nnmPrivateUDR**.

```
azure network route-table create -g azureNNM -n nnmPrivateUDR -l eastus
info:    Executing command network route-table create
+ Looking up Route Table "nnmPrivateUDR"
+ Creating Route Table "nnmPrivateUDR"
data:    Name                           : nnmPrivateUDR
data:    Type                           : Microsoft.Network/routeTables
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
info:    network route-table create command OK
```

19. Create a route to the internet using the NAT gateway as the next hop for instances in the private subnet.

    In this example, the private IP address of the NAT gateway is **10.240.0.4**.

```
azure network route-table route create -g azureNNM -r nnmPrivateUDR -n
RouteToInternet -a 0.0.0.0/0 -y VirtualAppliance -p 10.240.0.4
info:    Executing command network route-table route create
+ Looking up Route Table "nnmPrivateUDR"
+ Looking up route "RouteToInternet" in route table "nnmPrivateUDR"
+ Creating route "RouteToInternet" in a route table "nnmPrivateUDR"
data:    Name                           : RouteToInternet
```

```
data:    Provisioning state             : Succeeded
data:    Next hop type                  : VirtualAppliance
data:    Next hop IP address            : 10.240.0.4
data:    Address prefix                 : 0.0.0.0/0
info:    network route-table route create command OK
```

20. Associate the route table **nnmPrivateUDR** with the private subnet **nnmPrivate**.

```
azure network vnet subnet set -g azureNNM -e nnmVNet -n nnmPrivate -r
nnmPrivateUDR
info:    Executing command network vnet subnet set
+ Looking up the virtual network "nnmVNet"
+ Looking up the subnet "nnmPrivate"
+ Looking up Route Table "nnmPrivateUDR"
+ Updating subnet "nnmPrivate"
data:    Name                           : nnmPrivate
data:    Provisioning state             : Succeeded
data:    Address prefix                 : 10.240.1.0/24
info:    network vnet subnet set command OK
```

21. Create a NIC for an example instance in the private subnet. You will need to create a new NIC for every additional instance you create.

    In this example, the new NIC is named **nnmPrivateNic**.

```
azure network nic create --subnet-name nnmPrivate --subnet-vnet-name nnmVNet
azureNNM nnmPrivateNic eastus
info:    Executing command network nic create
+ Looking up the network interface "nnmPrivateNic"
+ Looking up the subnet "nnmPrivate"
+ Creating network interface "nnmPrivateNic"
data:    Name                           : nnmPrivateNic
data:    Type                           : Microsoft.Network/networkInterfaces
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
data:    Internal domain name suffix    :
gqhqyfrlprbu3jyndjoq4ap5se.bx.internal.cloudapp.net
data:    Enable IP forwarding           : false
```

```
data:    IP configurations:
data:      Name                          : default-ip-config
data:      Provisioning state            : Succeeded
data:      Private IP address            : 10.240.1.4
data:      Private IP version            : IPv4
data:      Private IP allocation method  : Dynamic
data:
info:    network nic create command OK
```

22. Launch an example instance into the private subnet **nnmPrivate** using the **nnmPrivateNic** as the NIC.

```
azure vm create --resource-group azureNNM --name exampleInstance --location eastus
--os-type linux  --nic-name nnmPrivateNic --vnet-name nnmVNet --vnet-subnet-name
nnmPrivate --storage-account-name nnmstore --image-urn CentOS --ssh-publickey-file
~/.ssh/azureNNM_id_rsa.pub --admin-username centos
info:    Executing command vm create
+ Looking up the VM "exampleInstance"
info:    Verifying the public key SSH file: ~/.ssh/azureNNM_id_rsa.pub
info:    Using the VM Size "Standard_DS1"
info:    The [OS, Data] Disk or image configuration requires storage account
+ Looking up the storage account nnmstore
+ Looking up the NIC "nnmPrivateNic"
info:    Found an existing NIC "nnmPrivateNic"
info:    This is an NIC without publicIP configured
info:    The storage URI 'https://nnmstore.blob.core.windows.net/' will be used
for boot diagnostics settings, and it can be overwritten by the parameter input of
'--boot-diagnostics-storage-uri'.
+ Creating VM "exampleInstance"
info:    vm create command OK
```

Install NNM on the NAT Gateway

Before You Begin

Follow the instructions on setting up a NAT gateway in a Microsoft Azure Virtual Network.

The NNM installer package for your NAT gateway instance's platform can be downloaded from the Tenable Downloads page.

Steps

1. Follow the instructions for using SSH to connect to an Azure Linux instance. Once you can connect to your instance using SSH, you can use scp to copy the NNM installer package to the home directory in your NAT gateway instance.
2. Log in to your NAT gateway instance.
3. Once logged into your NAT gateway instance, install NNM.

   After NNM is installed and running on the NAT gateway, you may access the NNM web front end by navigating to **https://<external IP address of nat-gateway>:8835** in your Web browser.

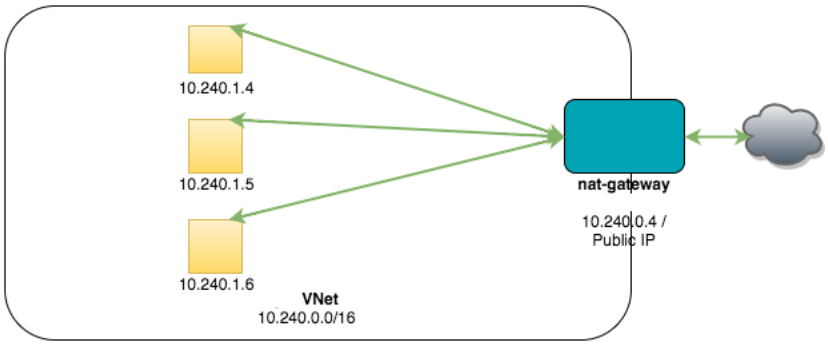
Example Deployment

This section demonstrates an example of NNM running on a virtual machine functioning as a NAT gateway instance within a Microsoft Azure Virtual Network.

In the examples used in the instructions for setting up a NAT gateway, the Virtual Network **pvsVNet** was created, which has the network range **10.240.0.0/16**. Additionally, the virtual machine instance **pvsNatGateway** was created in the **pvsPublic** subnet to function as the NAT gateway. In this example, three other virtual machine instances were created within the **pvsPrivate** subnet. None of the virtual machine instances in **pvsPrivate** are assigned an external IP address and all outgoing traffic is routed through **pvsNatGateway** .

In this example, there are four virtual machine instances within **pvsVNet**:

VM Instance Name	Internal IP	Has External IP?
pvsNatGateway	10.240.0.4	Yes
exampleInstance	10.240.1.4	No
exampleInstance2	10.240.1.5	No
exampleInstance3	10.240.1.6	No



NNM is running on **pvsNatGateway** and has the following configuration:

Configuration Parameter	Value
Monitored Network Interfaces	eth0
Monitored Network IP Addresses and Ranges	10.240.0.0/16

With this configuration, NNM will monitor traffic

- from the internal virtual machine instances to the Internet,
- between **pvsNatGateway** and the internal virtual machine instances,
- from the Internet to internal virtual machine instances if you have enabled port forwarding on the NAT gateway to make them Internet accessible,
- and between **pvsNatGateway** and the Internet.

> **Note:** Azure policy prevents interfaces from operating in promiscuous mode. As a result, NNM can't monitor traffic between other virtual instances.

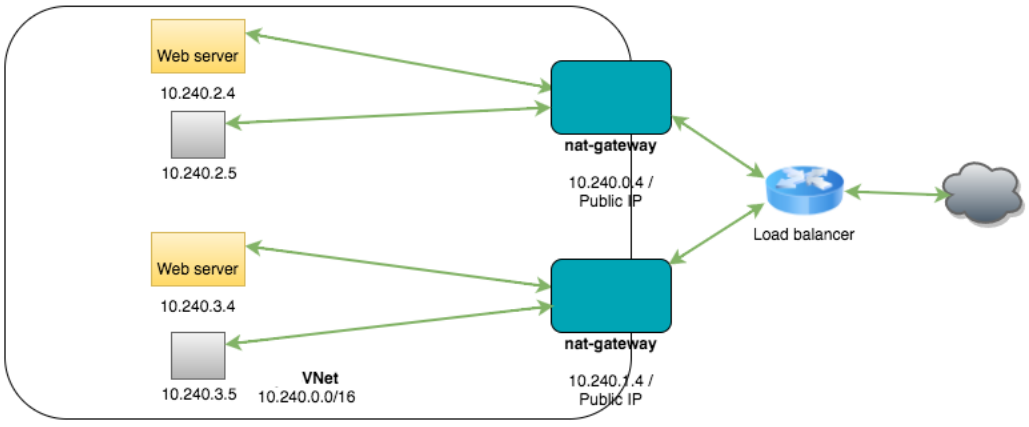
Example Load Balancing Deployment

This section demonstrates an example of NNM running on two virtual machines functioning as NAT gateways within a Microsoft Azure Virtual Network using a Load Balancer. The NAT gateways also serve as the back end pool for the load balancer.

In the examples used in the instructions for setting up a NAT gateway with load balancing, the Virtual Network **pvsVNet** was created, which has the network range **10.240.0.0/16**. Additionally, two virtual machine instances were created to function as NAT gateways. The NAT gateways provide a connection to the Internet for hosts behind the NAT. They also function as the backend pool for the load balancer. When a web request is made, the load balancer will distribute the traffic between the two NAT gateways. The NAT gateways have IP tables rules to forward the web request to the web server behind them.

In this example, there are four virtual machine instances within **pvsVNet**. The diagram shows two additional instances to indicate that normally there would be additional hosts behind the NAT:

VM Instance Name	Internal IP	Has External IP?
pvsNatGateway	10.240.0.4	Yes
pvsNatGateway2	10.240.1.4	Yes
Web server	10.240.2.4	No
Web server 2	10.240.3.4	No



NNM is running on the NAT gateways at **10.240.0.4** and **10.240.1.4** and has the following configuration:

Configuration Parameter	Value
Monitored Network Interfaces	eth0
Monitored Network IP Addresses and Ranges	10.240.0.0/16

With this configuration, NNM will monitor traffic

- from the internal virtual machine instances to the Internet,
- between the NAT gateway and the internal virtual machine instances,
- from the Internet to internal virtual machine instances if you have enabled port forwarding on the NAT gateway to make them Internet accessible,
- and between the NAT gateway and the Internet.

> **Note:** Azure policy prevents interfaces from operating in promiscuous mode. As a result, NNM can't monitor traffic between other virtual instances.

Set up a NAT Gateway with Load Balancing

Introduction

In order for NNM to monitor virtual machine instances in a Microsoft Azure Virtual Network, NNM must run on a virtual machine instance that functions as a network address translation (NAT) gateway. A NAT gateway instance routes traffic from internal-only virtual machine instances to the Internet. A NNM installed on a NAT gateway has visibility into the hostnames and private IP addresses of the internal virtual machine instances before the NAT gateway masquerades the source IP address of incoming packets to forward them to the Internet. Microsoft Azure provides a load balancing service that distributes traffic between multiple servers.

This guide shows setting up a NAT gateway in a Microsoft Azure Virtual Network using load balancing.

Before You Begin

Follow the Azure CLI Installation Instructions. Then connect to your subscription from the CLI.

> **Tip:** If you encounter an error in the Azure CLI about the your subscription not being registered to use a namespace, see this section on the common deployment errors page.

Steps

1. Enable Azure CLI Resource Manager commands.

   ```
   azure config mode arm
   ```

2. Create a resource group.

   In this example, the resource group **pvsLbRg** is created.

   ```
   azure group create pvsLbRg eastus
   info:    Executing command group create
   + Getting resource group pvsLbRg
   + Creating resource group pvsLbRg
   info:    Created resource group pvsLbRg
   ```

```
data:    Name:                pvsLbRg
data:    Location:            eastus
data:    Provisioning State:  Succeeded
data:    Tags: null
data:
info:    group create command OK
```

3. Create a storage account in the resource group **pvsLbRg** .

   In this example, the storage group **pvslbstore** is created.

```
azure storage account create --location eastus --resource-group pvsLbRg  --kind
Storage --sku-name GRS pvslbstore
info:    Executing command storage account create
+ Checking availability of the storage account name
+ Creating storage account
info:    storage account create command OK
```

4. Create a Virtual Network in the resource group **pvsLbRg** .

   In this example, the Virtual Network is **pvsVNet** and has the network range **10.240.0.0/16**.

```
azure network vnet create -g pvsLbRg -n pvsVNet -a 10.240.0.0/16 -l eastus
info:    Executing command network vnet create
+ Looking up the virtual network "pvsVNet"
+ Creating virtual network "pvsVNet"
data:    Name                            : pvsVNet
data:    Type                            : Microsoft.Network/virtualNetworks
data:    Location                        : eastus
data:    Provisioning state              : Succeeded
data:    Address prefixes:
data:      10.240.0.0/16
info:    network vnet create command OK
```

5. Create a Load Balancer.

   In this example, the Load Balancer is **pvsLb**.

```
azure network lb create pvsLbRg pvsLb eastus
```

```
info:    Executing command network lb create
+ Looking up the load balancer "pvsLb"
+ Creating load balancer "pvsLb"
data:    Name                           : pvsLb
data:    Type                           : Microsoft.Network/loadBalancers
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
info:    network lb create command OK
```

6. Create a public IP and sub domain name for the Load Balancer front end pool.

   In this example, the sub domain name is **examplelbsubdomain** and the public IP is **pvsPIP**.

```
azure network public-ip create -d examplelbsubdomain pvsLbRg  pvsLbPIP eastus
info:    Executing command network public-ip create
warn:    Using default --idle-timeout 4
warn:    Using default --allocation-method Dynamic
warn:    Using default --ip-version IPv4
+ Looking up the public ip "pvsLbPIP "
+ Creating public ip address "pvsLbPIP "
data:    Name                           : pvsLbPIP
data:    Type                           : Microsoft.Network/publicIPAddresses
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
data:    Allocation method              : Dynamic
data:    IP version                     : IPv4
data:    Idle timeout in minutes        : 4
data:    Domain name label              : examplelbsubdomain
data:    FQDN                           :
examplelbsubdomain.eastus.cloudapp.azure.com
info:    network public-ip create command OK
```

7. Create a front end IP pool and associate it with the public IP **pvsLbPIP**.

   In this example, the frontend pool is named **pvsFrontEndPool**.

```
azure network lb frontend-ip create pvsLbRg pvsLb pvsFrontendPool -i pvsLbPIP
info:    Executing command network lb frontend-ip create
+ Looking up the load balancer "pvsLb"
```

```
+ Looking up the public ip "pvsLbPIP"
+ Updating load balancer "pvsLb"
data:    Name                            : pvsFrontendPool
data:    Provisioning state              : Succeeded
data:    Private IP allocation method    : Dynamic
info:    network lb frontend-ip create command OK
```

8. Create a back end IP pool.

   In this example, the back end pool is **pvsLbBackendPool**.

```
azure network lb address-pool create pvsLbRg pvsLb pvsLbBackendPool
info:    Executing command network lb address-pool create
+ Looking up the load balancer "pvsLb"
+ Updating load balancer "pvsLb"
data:    Name                            : pvsLbBackendPool
data:    Provisioning state              : Succeeded
info:    network lb address-pool create command OK
```

9. Create a load balancer rule to balance all incoming traffic on port 80 to port 80 on the addresses in the back end pool.

```
azure network lb rule create pvsLbRg pvsLb webLbRule -p tcp -f 80 -b 80 -t
pvsFrontendPool -o pvsLbBackendPool
info:    Executing command network lb rule create
+ Looking up the load balancer "pvsLb"
warn:    Using default --idle-timeout 4
warn:    Using default --enable-floating-ip false
warn:    Using default --load-distribution Default
+ Updating load balancer "pvsLb"
data:    Name                            : webLbRule
data:    Provisioning state              : Succeeded
data:    Protocol                        : Tcp
data:    Frontend port                   : 80
data:    Backend port                    : 80
data:    Enable floating IP              : false
data:    Load distribution               : Default
data:    Idle timeout in minutes         : 4
```

```
info:    network lb rule create command OK
```

10. Create a public subnet for the first NAT gateway so that it is accessible over SSH and the NNM web server port.

    In this example, the public subnet is **pvsPublic** and has the network range **10.240.0.0/24**.

```
azure network vnet subnet create -g pvsLbRg -e pvsVNet -n pvsPublic -a
10.240.0.0/24
info:    Executing command network vnet subnet create
+ Looking up the virtual network "pvsVNet"
+ Looking up the subnet "pvsPublic"
+ Creating subnet "pvsPublic"
data:    Name                           : pvsPublic
data:    Provisioning state             : Succeeded
data:    Address prefix                 : 10.240.0.0/24
info:    network vnet subnet create command OK
```

11. Create a public IP and sub domain name for the NAT gateway.

    In this example, the sub domain name is **exampleNatsubdomain** and the public IP is **pvsNatPIP**.

```
azure network public-ip create -d exampleNatsubdomain pvsLbRg  pvsNatPIP eastus
info:    Executing command network public-ip create
warn:    Using default --idle-timeout 4
warn:    Using default --allocation-method Dynamic
warn:    Using default --ip-version IPv4
+ Looking up the public ip "pvsNatPIP"
+ Creating public ip address "pvsNatPIP"
data:    Name                           : pvsNatPIP
data:    Type                           : Microsoft.Network/publicIPAddresses
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
data:    Allocation method              : Dynamic
data:    IP version                     : IPv4
data:    Idle timeout in minutes        : 4
data:    Domain name label              : exampleNatsubdomain
```

```
data:    FQDN                            :
exampleNatsubdomain.eastus.cloudapp.azure.com
info:    network public-ip create command OK
```

12. Create a NIC for the NAT gateway and associate it with the public IP **pvsNatPIP**, public subnet **pvsPublic**, and back end pool **pvsLbBackendPool**.

    In this example, the new NIC is **pvsNatNic**.

```
azure network nic create --public-ip-name pvsNatPIP--subnet-name pvsPublic --
subnet-vnet-name pvsVNet  pvsLbRg  pvsNatNic -d /subscriptions/x-x-x-x-
x/resourceGroups/pvsLbRg/providers/Microsoft.Network/loadBalancers/pvsLb/backendAd-
dressPools/pvsLbBackendPool eastus
info:    Executing command network nic create
+ Looking up the network interface "pvsNatNic"
+ Looking up the subnet "pvsPublic"
+ Looking up the public ip "pvsPIP"
+ Creating network interface "pvsNatNic"
data:    Name                            : pvsNatNic
data:    Type                            : Microsoft.Network/networkInterfaces
data:    Location                        : eastus
data:    Provisioning state              : Succeeded
data:    Internal domain name suffix     :
gqhqyfrlprbu3jyndjoq4ap5se.bx.internal.cloudapp.net
data:    Enable IP forwarding            : false
data:    IP configurations:
data:      Name                          : default-ip-config
data:      Provisioning state            : Succeeded
data:      Private IP address            : 10.240.0.4
data:      Private IP version            : IPv4
data:      Private IP allocation method  : Dynamic
data:
info:    network nic create command OK
```

13. Enable IP forwarding on the new interface **pvsNatNic**.

```
azure network nic set -g pvsLbRg  -n pvsNatNic-f true
info:    Executing command network nic set
```

```
+ Looking up the network interface "pvsNatNic"
+ Updating network interface "pvsNatNic"
data:    Name                              : pvsNatNic
data:    Type                              : Microsoft.Network/networkInterfaces
data:    Location                          : eastus
data:    Provisioning state                : Succeeded
data:    MAC address                       : 00-0D-3A-13-27-48
data:    Internal domain name suffix       :
gqhqyfrlprbu3jyndjoq4ap5se.bx.internal.cloudapp.net
data:    Enable IP forwarding              : true
data:    IP configurations:
data:      Name                            : default-ip-config
data:      Provisioning state              : Succeeded
data:      Private IP address              : 10.240.0.4
data:      Private IP version              : IPv4
data:      Private IP allocation method    : Dynamic
data:
info:    network nic set command OK
```

14. Repeat the previous steps to create a public subnet (**pvsPublic2**) with network range **10.240.1.0/24**, create a public IP, create a NIC (**pvsNatNic2**) and add it to the back end pool, and enable IP forwarding on the new NIC.

    Repeat this step for other NAT gateway instances that you want to use.

```
azure network vnet subnet create -g pvsLbRg -e pvsVNet -n pvsPublic2 -a
10.240.1.0/24
azure network public-ip create -d examplenat2subdomain pvsLbRg  pvsNatPIP2 eastus
azure network nic create --public-ip-name pvsNatPIP2 --subnet-name pvsPublic2 --
subnet-vnet-name pvsVNet  pvsLbRg  pvsNatNic2 -d subscriptions/x-x-x-
x/resourceGroups/pvsLbRg/providers/Microsoft.Network/loadBalancers/pvsLb/backendAd-
dressPools/pvsLbBackendPool eastus
azure network nic set -g pvsLbRg  -n pvsNatNic2 -f true
```

15. Create a security group for the NAT gateways.

    In this example, the security group is **pvsPublicNSG**.

```
azure network nsg create pvsLbRg  pvsPublicNSG eastus
info:    Executing command network nsg create
+ Looking up the network security group "pvsPublicNSG"
+ Creating a network security group "pvsPublicNSG"
data:    Name                           : pvsPublicNSG
data:    Type                           : Microsoft.Network/networkSecurityGroups
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
data:    Security rules:
data:    Name                           Source IP          Source Port
Destination IP  Destination Port  Protocol  Direction  Access  Priority
data:    -----------------------------  -----------------  -----------  ----------
----  ---------------  --------  ---------  ------  --------
data:    AllowVnetInBound               VirtualNetwork     *
VirtualNetwork  *                *         Inbound    Allow   65000
data:    AllowAzureLoadBalancerInBound  AzureLoadBalancer  *            *
     *                *         Inbound    Allow   65001
data:    DenyAllInBound                 *                  *            *
     *                *         Inbound    Deny    65500
data:    AllowVnetOutBound              VirtualNetwork     *
VirtualNetwork  *                *         Outbound   Allow   65000
data:    AllowInternetOutBound          *                  *            Internet
     *                *         Outbound   Allow   65001
data:    DenyAllOutBound                *                  *            *
     *                *         Outbound   Deny    65500
info:    network nsg create command OK
```

16. Create a rule in the **pvsPublicNSG** to allow SSH to the NAT gateway.

    In this example, the new rule is called **SSHRule** and the rule has a priority of 1000. This gives it precedence over the existing rules seen in the previous step.

```
azure network nsg rule create --protocol tcp --direction inbound --priority 1000 -
-destination-port-range 22 --access allow pvsLbRg  pvsPublicNSG SSHRule
info:    Executing command network nsg rule create
warn:    Using default --source-port-range *
warn:    Using default --source-address-prefix *
warn:    Using default --destination-address-prefix *
```

```
+ Looking up the network security group "pvsPublicNSG"
+ Looking up the network security rule "SSHRule"
+ Creating a network security rule "SSHRule"
data:    Name                           : SSHRule
data:    Type                           :
Microsoft.Network/networkSecurityGroups/securityRules
data:    Provisioning state             : Succeeded
data:    Source IP                      : *
data:    Source Port                    : *
data:    Destination IP                 : *
data:    Destination Port               : 22
data:    Protocol                       : Tcp
data:    Direction                      : Inbound
data:    Access                         : Allow
data:    Priority                       : 1000
info:    network nsg rule create command OK
```

17. Create a rule in the **pvsPublicNSG** to allow all traffic to the NAT gateway from within the virtual network.

    In this example, the new rule is called **PrivateToPublicRule** and the rule has a priority of 1001. This gives it precedence over the existing rules that disallow traffic.

```
azure network nsg rule create --direction inbound --priority 1001  --source-
address-prefix VirtualNetwork --destination-port-range 0-65535 --access allow
pvsLbRg  pvsPublicNSG PrivateToPublicRule
info:    Executing command network nsg rule create
warn:    Using default --protocol *
warn:    Using default --source-port-range *
warn:    Using default --destination-address-prefix *
+ Looking up the network security group "pvsPublicNSG"
+ Looking up the network security rule "PrivateToPublicRule"
+ Creating a network security rule "PrivateToPublicRule"
data:    Name                           : PrivateToPublicRule
data:    Type                           :
Microsoft.Network/networkSecurityGroups/securityRules
data:    Provisioning state             : Succeeded
data:    Source IP                      : VirtualNetwork
data:    Source Port                    : *
```

```
data:    Destination IP                  : *
data:    Destination Port                : 0-65535
data:    Protocol                        : *
data:    Direction                       : Inbound
data:    Access                          : Allow
data:    Priority                        : 1001
info:    network nsg rule create command OK
```

18. Create a rule in the **pvsPublicNSG** to allow traffic to the NNM webserver from the Internet. The default port is 8835.

    In this example, the new rule is called **PVSWebRule** and the rule has a priority of 1002. This gives it precedence over the existing rules that disallow traffic.

```
azure network nsg rule create --direction inbound --priority 1002 --protocol tcp
--source-address-prefix Internet --destination-port-range 8835 --access allow
pvsLbRg  pvsPublicNSG PvsWebRule
info:    Executing command network nsg rule create
warn:    Using default --source-port-range *
warn:    Using default --destination-address-prefix *
+ Looking up the network security group "pvsPublicNSG"
+ Looking up the network security rule "PvsWebRule"
+ Creating a network security rule "PvsWebRule"
data:    Name                            : PvsWebRule
data:    Type                            :
Microsoft.Network/networkSecurityGroups/securityRules
data:    Provisioning state              : Succeeded
data:    Source IP                       : Internet
data:    Source Port                     : *
data:    Destination IP                  : *
data:    Destination Port                : 8835
data:    Protocol                        : Tcp
data:    Direction                       : Inbound
data:    Access                          : Allow
data:    Priority                        : 1002
info:    network nsg rule create command OK
```

19. Create a rule in the **pvsPublicNSG** to allow HTTP traffic to the NAT gateway so it can be forwarded to the web servers being load balanced.

    In this example, the new rule is called **AllWebRule** and the rule has a priority of 1003. This gives it precedence over the existing rules that disallow traffic.

```
azure network nsg rule create --direction inbound --priority 1003 --protocol tcp
--source-address-prefix Internet --destination-port-range 80 --access allow
pvsLbRg  pvsPublicNSG AllWebRule
```

20. Assign the security group **pvsPublicNSG** to the **pvsNatNic**, which will be used as the interface of the NAT gateway when it is launched.

```
azure network nic set -g pvsLbRg  -n pvsNatNic -o pvsPublicNSG
info:    Executing command network nic set
+ Looking up the network interface "pvsNatNic"
+ Looking up the network security group "pvsPublicNSG"
+ Updating network interface "pvsNatNic"
data:    Name                           : pvsNatNic
data:    Type                           : Microsoft.Network/networkInterfaces
data:    Location                       : eastus
data:    Provisioning state             : Succeeded
data:    Internal domain name suffix    :
gqhqyfrlprbu3jyndjoq4ap5se.bx.internal.cloudapp.net
data:    Enable IP forwarding           : false
data:    IP configurations:
data:      Name                         : default-ip-config
data:      Provisioning state           : Succeeded
data:      Private IP address           : 10.240.0.4
data:      Private IP version           : IPv4
data:      Private IP allocation method : Dynamic
data:
info:    network nic set command OK
```

21. Repeat the previous step for any other NAT gateway NICs you have created.

```
azure network nic set -g pvsLbRg  -n pvsNatNic2 -o pvsPublicNSG
```

22. Create an availability set for all the VM instances that will be created. This is required for having more than one VM attached to the load balancer.

```
azure availset create pvsLbRg pvsLbAs eastus
info:    Executing command availset create
+ Looking up the availability set "pvsLbAs"
+ Creating availability set "pvsLbAs"
info:    availset create command OK
```

23. Launch the NAT gateway instance. In this example CentOS 7 is used.

    In this example, the SSH key **azurePVS_id_rsa.pub** is used. If you do not have an SSH key, refer to the Azure documentation for instructions on how to generate a key.

    Note: If you select a different image to install on your NAT gateway virtual machine, ensure that it is a platform that NNM supports.

```
azure vm create --resource-group pvsLbRg  --name pvsNatGateway --location eastus --os-type linux  --nic-name pvsNatNic --vnet-name pvsVNet --vnet-subnet-name pvsPublic --storage-account-name pvslbstore --image-urn CentOS -r pvsLbAs --ssh-publickey-file ~/.ssh/azurePVS_id_rsa.pub --admin-username centos
info:    Executing command vm create
+ Looking up the VM "pvsNatGateway"
info:    Verifying the public key SSH file: ~/.ssh/azurePVS_id_rsa.pub
info:    Using the VM Size "Standard_DS1"
info:    The [OS, Data] Disk or image configuration requires storage account
+ Looking up the storage account pvsstore
+ Looking up the NIC "pvsNatNic"
info:    Found an existing NIC "pvsNatNic"
info:    The storage URI 'https://pvslbstore.blob.core.windows.net/' will be used for boot diagnostics settings, and it can be overwritten by the parameter input of '--boot-diagnostics-storage-uri'.
+ Creating VM "pvsNatGateway"
info:    vm create command OK
```

24. Launch any other NAT gateway instances.

```
azure vm create --resource-group pvsLbRg  --name pvsNatGateway2 --location eastus
--os-type linux  --nic-name pvsNatNic2 --vnet-name pvsVNet --vnet-subnet-name
pvsPublic2 --storage-account-name pvslbstore --image-urn CentOS -r pvsLbAs --ssh-
publickey-file ~/.ssh/azurePVS_id_rsa.pub --admin-username centos
```

25. Create a private subnet for the instances that won't have a public IP address.

    In this example, the private subnet is **pvsPrivate**.

```
azure network vnet subnet create -g pvsLbRg  -e pvsVNet -n pvsPrivate -a
10.240.2.0/24
info:    Executing command network vnet subnet create
+ Looking up the virtual network "pvsVNet"
+ Looking up the subnet "pvsPrivate"
+ Creating subnet "pvsPrivate"
data:    Name                          : pvsPrivate
data:    Provisioning state            : Succeeded
data:    Address prefix                : 10.240.2.0/24
info:    network vnet subnet create command OK
```

26. Create a route table for the private subnet.

    In this example, the route table is **pvsPrivateUDR**.

```
azure network route-table create -g pvsLbRg  -n pvsPrivateUDR -l eastus
info:    Executing command network route-table create
+ Looking up Route Table "pvsPrivateUDR"
+ Creating Route Table "pvsPrivateUDR"
data:    Name                          : pvsPrivateUDR
data:    Type                          : Microsoft.Network/routeTables
data:    Location                      : eastus
data:    Provisioning state            : Succeeded
info:    network route-table create command OK
```

27. Create a route to the internet using the NAT gateway as the next hop for instances in the private subnet.

    In this example, the private IP address of the NAT gateway is **10.240.0.4**.

```
azure network route-table route create -g pvsLbRg  -r pvsPrivateUDR -n
RouteToInternet -a 0.0.0.0/0 -y VirtualAppliance -p 10.240.0.4
info:    Executing command network route-table route create
+ Looking up Route Table "pvsPrivateUDR"
+ Looking up route "RouteToInternet" in route table "pvsPrivateUDR"
+ Creating route "RouteToInternet" in a route table "pvsPrivateUDR"
data:    Name                           : RouteToInternet
data:    Provisioning state             : Succeeded
data:    Next hop type                  : VirtualAppliance
data:    Next hop IP address            : 10.240.0.4
data:    Address prefix                 : 0.0.0.0/0
info:    network route-table route create command OK
```

28. Associate the route table **pvsPrivateUDR** with the private subnet **pvsPrivate**.

```
azure network vnet subnet set -g pvsLbRg  -e pvsVNet -n pvsPrivate -r
pvsPrivateUDR
info:    Executing command network vnet subnet set
+ Looking up the virtual network "pvsVNet"
+ Looking up the subnet "pvsPrivate"
+ Looking up Route Table "pvsPrivateUDR"
+ Updating subnet "pvsPrivate"
data:    Name                           : pvsPrivate
data:    Provisioning state             : Succeeded
data:    Address prefix                 : 10.240.2.0/24
info:    network vnet subnet set command OK
```

29. Create a NIC for an example instance in the private subnet. You will need to create a new NIC for every additional instance you create.

    In this example, the new NIC is named **pvsPrivateNic**.

```
azure network nic create --subnet-name pvsPrivate --subnet-vnet-name pvsVNet
pvsLbRg  pvsPrivateNic eastus
info:    Executing command network nic create
+ Looking up the network interface "pvsPrivateNic"
+ Looking up the subnet "pvsPrivate"
+ Creating network interface "pvsPrivateNic"
```

```
data:    Name                             : pvsPrivateNic
data:    Type                             : Microsoft.Network/networkInterfaces
data:    Location                         : eastus
data:    Provisioning state               : Succeeded
data:    Internal domain name suffix      :
gqhqyfrlprbu3jyndjoq4ap5se.bx.internal.cloudapp.net
data:    Enable IP forwarding             : false
data:    IP configurations:
data:      Name                           : default-ip-config
data:      Provisioning state             : Succeeded
data:      Private IP address             : 10.240.2.4
data:      Private IP version             : IPv4
data:      Private IP allocation method   : Dynamic
data:
info:    network nic create command OK
```

30. Launch an example instance into the private subnet **pvsPrivate** using the **pvsPrivateNic** as the NIC.

```
azure vm create --resource-group pvsLbRg  --name exampleInstance --location eastus
--os-type linux  --nic-name pvsPrivateNic --vnet-name pvsVNet --vnet-subnet-name
pvsPrivate --storage-account-name pvslbstore --image-urn CentOS --ssh-publickey-
file ~/.ssh/azurePVS_id_rsa.pub --admin-username centos
info:    Executing command vm create
+ Looking up the VM "exampleInstance"
info:    Verifying the public key SSH file: ~/.ssh/azurePVS_id_rsa.pub
info:    Using the VM Size "Standard_DS1"
info:    The [OS, Data] Disk or image configuration requires storage account
+ Looking up the storage account pvsstore
+ Looking up the NIC "pvsPrivateNic"
info:    Found an existing NIC "pvsPrivateNic"
info:    This is an NIC without publicIP configured
info:    The storage URI 'https://pvsstore.blob.core.windows.net/' will be used
for boot diagnostics settings, and it can be overwritten by the parameter input of
'--boot-diagnostics-storage-uri'.
+ Creating VM "exampleInstance"
info:    vm create command OK
```

31. Repeat the previous steps to create any additional private instances if they are going into a new subnet. To create a new instance in an existing subnet, simply repeat the launch into the subnet step.

```
azure network vnet subnet create -g pvsLbRg  -e pvsVNet -n pvsPrivate2 -a
10.240.3.0/24
```

```
azure network route-table create -g pvsLbRg  -n pvsPrivateUDR2 -l eastus
```

```
azure network route-table route create -g pvsLbRg  -r pvsPrivateUDR2 -n
RouteToInternet -a 0.0.0.0/0 -y VirtualAppliance -p 10.240.1.4
```

```
azure network vnet subnet set -g pvsLbRg  -e pvsVNet -n pvsPrivate2 -r
pvsPrivateUDR2
```

```
azure network nic create --subnet-name pvsPrivate2 --subnet-vnet-name pvsVNet
pvsLbRg  pvsPrivateNic2 eastus
```

```
azure vm create --resource-group pvsLbRg  --name exampleInstance2 --location
eastus --os-type linux  --nic-name pvsPrivateNic2 --vnet-name pvsVNet --vnet-
subnet-name pvsPrivate2 --storage-account-name pvslbstore --image-urn CentOS --
ssh-publickey-file ~/.ssh/azurePVS_id_rsa.pub --admin-username centos
```

32. Connect to the new NAT gateway instances using the public DNS name that was created when your public IP was created.

```
ssh -i ~/.ssh/azurePVS_id_rsa.pub
centos@examplesubdomain.eastus.cloudapp.azure.com
```

33. Once logged in, configure iptables and IP forwarding.

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
sudo iptables -t nat -A PREROUTING -i eth0 ! -s 10.240.0.0/16 -p tcp --dport 80 -j
DNAT --to-destination 10.240.2.4:80
```

The first **sudo** command tells the kernel to allow IP forwarding. The second **sudo** command masquerades packets received from internal instances as if they originated from the NAT gateway instance. The last command enables port forwarding so traffic to the NAT gateway on port 80 gets forwarded to port 80 on 10.240.2.4. In this example 10.240.2.4 is one the web servers being load balanced.

> **Tip:** Consider saving these commands in a startup script, because these settings will not persist if the instance is rebooted.

Introduction to Docker

Docker is an open source program that you can use to package software in containers that include everything the software needs to run. Unlike virtual machines, Docker uses container-based virtualization that isolates applications on a shared operating system and reduces the overhead involved with having each guest running a completely installed operating system.

The Nessus Network Monitor (NNM) can be installed either on the host machine or in a Docker container and can be configured to sniff traffic on a Docker network, from one or more Docker containers, or from the host itself.

Note: To use Docker with NNM, NNM must be run in Standard mode.

Helpful Docker Commands

Command	Description
Ctrl-p + Ctrl-q	Disconnects from current interactive container.
`docker ps`	Lists the current running containers.
`docker images`	Lists available containers.
`docker stop <container>`	Stops the specified container.
`docker port <container>`	Lists the ports exposed on the specified container.
`docker network inspect <container>`	Displays the net information for the specified container.
`docker network ls`	Lists the available networks. The Network ID corresponds to the bridge interface on the host.

Docker Software Requirements

NNM running directly on a Docker host is available for the following platforms:

- Red Hat Linux ES 7 64-bit
- CentOS 7 64-bit

Configure NNM in a Docker Container

Before You Begin

Install an instance of NNM in a Docker container.

Dockerfile

```
FROM            centos:7
EXPOSE          8835
ADD             /nnm-5.0.0-es7.x86_64.rpm /nnm-5.0.0-es7.x86_64.rpm
RUN             rpm  -i nnm-5.0.0-es7.x86_64.rpm;
ENV             PATH /opt/nnm/bin:$PATH
CMD             /opt/nnm/bin/nnm && /opt/nnm/bin/nnm-proxy
```

Steps

1. Copy the text of the **`Dockerfile`** section above and paste them in a file named **`Dockerfile`**.
2. Copy the NNM **`*.rpm`** and paste it in the directory where you pasted the **`Dockerfile`**.
3. In a Linux shell, run the **`cd`** command to navigate to the directory that contains **`Dockerfile`** and the NNM **`*.rpm`** and run the following command:

   `docker build -t centos/nnm .`

4. Run the Docker container that contains NNM using the following command:

   `docker run --net=host -d -p 8835:8835 centos/nnm`

   > **Tip:** If you need to interact with NNM from the shell, you can run the following command instead: **`docker run --net=host -t -i -p 8835:8835 centos/nnm`**

5. Navigate to **`https://<IP address or hostname>:8835`**, which will display the NNM web front end to log in.

   Refer to the Configure NNM section of the NNM user guide for configuration instructions.

6. In step 5 of the Configure NNM instructions, configure the monitored network interfaces depending on your needs, outlined in the Monitored Interfaces section of this guide.

Monitored Interfaces

This topic describes guidelines for configuring the monitored network interfaces.

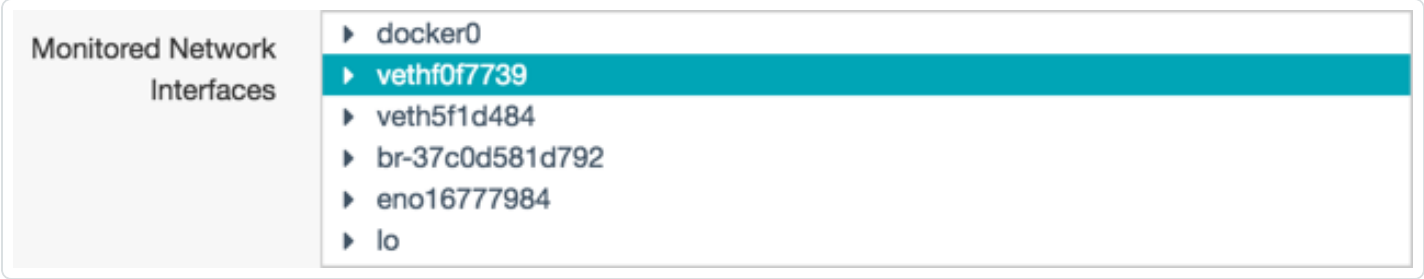
Available Monitored Network Interfaces

Monitored Network Interface	Description
docker0	An interface that bridges all virtual interfaces. Monitoring this interface will sniff traffic in all containers.
veth*	An interface that is associated with a container. Each container has one veth* interface.
Host interface	A host interface. Monitoring this interface will sniff all network traffic from the host, including traffic in Docker containers.
User-created Docker networks	An interface created with the **`docker network create`** parameter. This interface is discoverable using the **`docker network inspect <networkname>`** parameter.

Container to Network

To monitor traffic from containers to the network, set the monitored network interfaces to `docker0`, a host interface, or the virtual interface (`veth*`) assigned to that container. When monitoring only the host interface, connections made from the container will be reported by NNM as having the IP of the host rather than the container's private IP address.

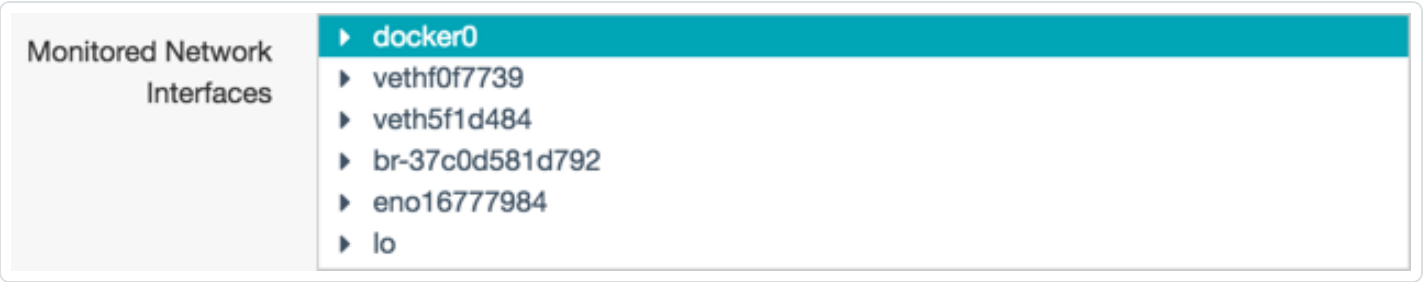
The following image shows an example monitoring configuration where NNM is running on the host and one container interface is selected.



Container to Container

To monitor traffic between all containers from the host, set the monitored network interface to `docker0`. You can also select the veth* of just the container(s) that you would like to monitor.

The following image shows an example monitoring configuration where `docker0` is selected, which will discover traffic from all containers.

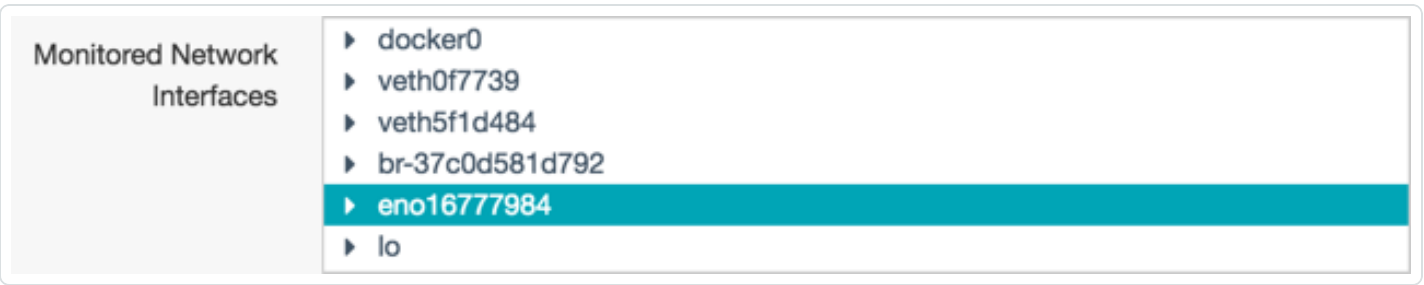


Monitoring these interfaces will detect traffic between docker containers, but not over custom bridges. If you have containers configured to use a custom bridge, configure NNM to monitor that interface. If you want the container to use an existing container's network stack, run the container with the **`--net=container:NAME_or_ID`** option (e.g., **`docker run --net=container:<NAME_or_ID> <container>`**).

Host to Network

To monitor traffic from the host to the network, set the monitored network interface to a host interface. This will also detect all traffic between containers and the external network.

The following image shows an example monitoring configuration where only the host interface is selected.



Note: Monitoring only on the host interface will not detect traffic between containers.

Monitored Interfaces Examples

Running a Docker container with the **`--net=host`** option allows the container to see all interfaces that are available to the host, but will prevent the container from creating its own local interface. To run a docker container using the **`--net=host`** option use the following command:

```
docker run --net=host <container>
```

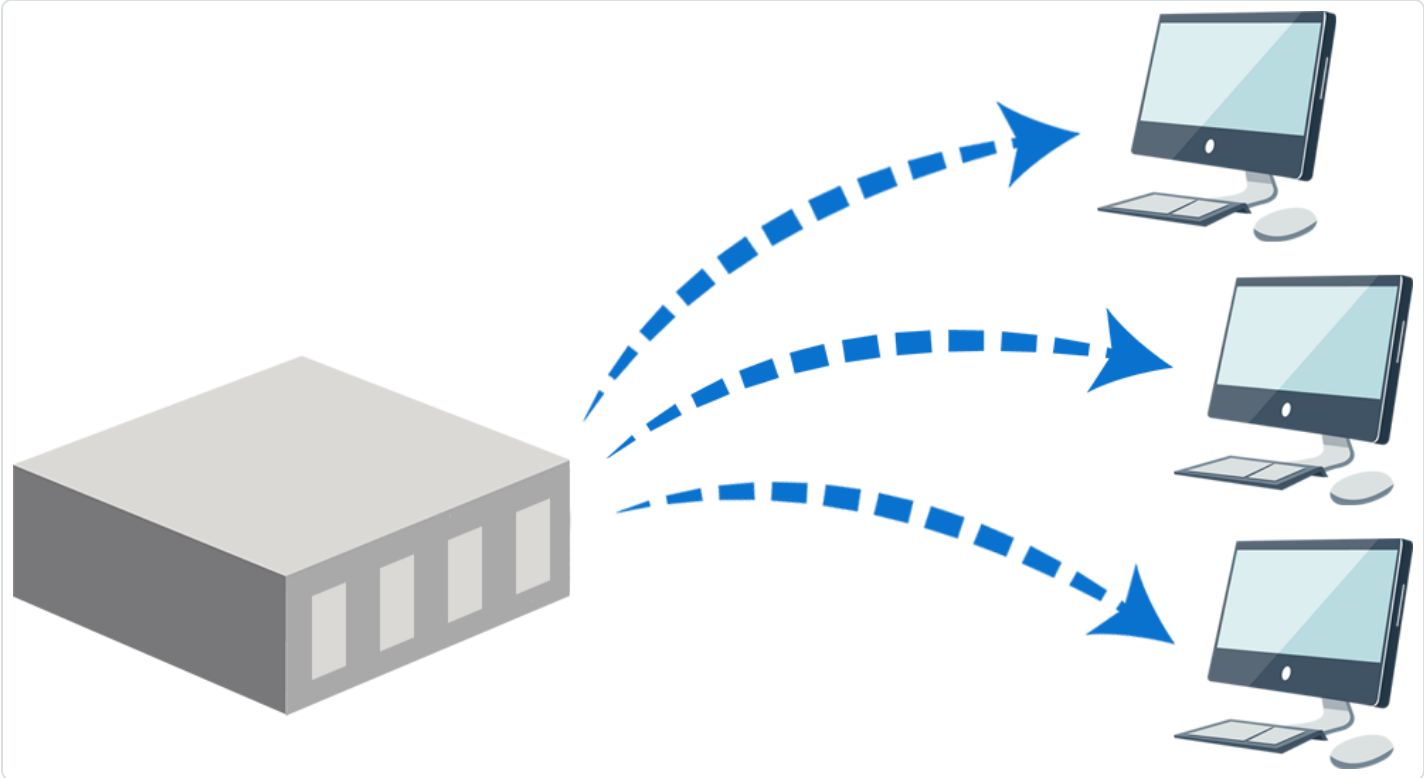
For the following examples, assume the following IP/host/container combinations:

Host - 192.0.2.1

Container 1 - veth1 - 192.0.2.2

Container 2 - veth2 - 192.0.2.3

Container to Network



Container 1 running with **`--net=host`** option.

Container 2 running without **`--net=host`** option.

From Host

Host interface

Traffic from the host will be reported with an IP of 192.0.2.1

Traffic from Container 1 will be reported with an IP of 192.0.2.1

Traffic from Container 2 will be reported with an IP of 192.0.2.1

Docker0

Traffic from the host will not be reported.

Traffic from Container 1 will be reported as coming from 192.0.2.2

Traffic from Container 2 will be reported as coming from 192.0.2.3

veth1

Will not exist due to running with **`--net=host`** option.

veth2

Traffic from the host will not be reported.

Traffic from Container 1 will not be reported.

Traffic from Container 2 will be reported as coming from 192.0.2.3

From Container 1

Host interface

Traffic from the host will be reported with an IP of 192.0.2.1

Traffic from Container 1 will be reported with an IP of 192.0.2.1

Traffic from Container 2 will be reported with an IP of 192.0.2.1

Docker0

Traffic from the host will not be reported.

Traffic from Container 1 will be reported as coming from 192.0.2.2

Traffic from Container 2 will be reported as coming from 192.0.2.3

Container 1 local interface

will not exist due to running with **`--net=host`**.

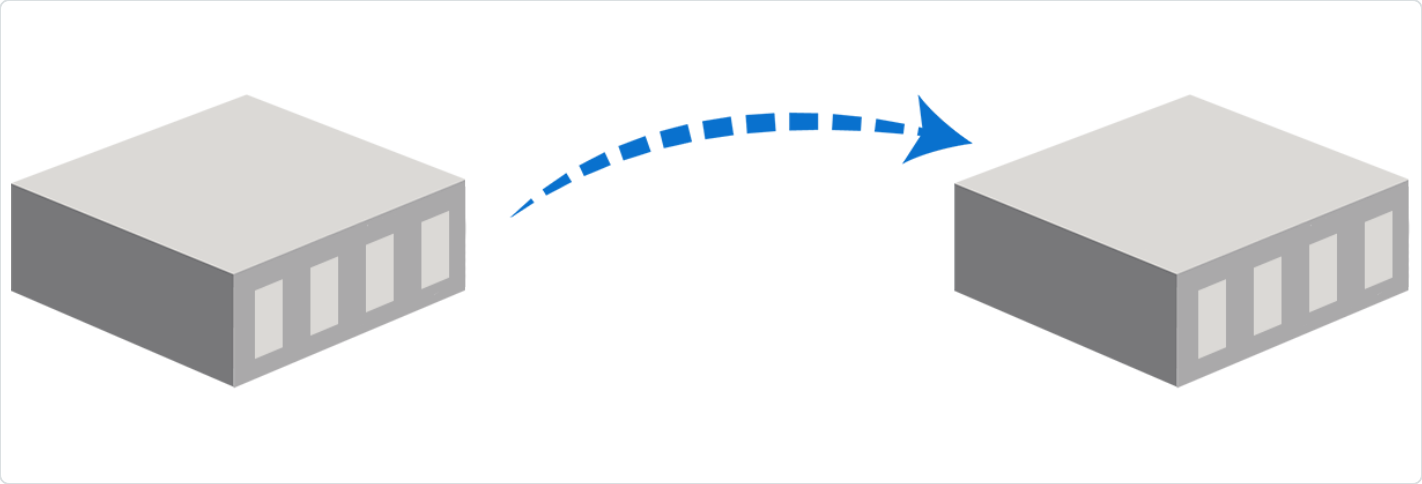
veth2

Traffic from the host will not be reported.

Traffic from Container 1 will not be reported.

Traffic from Container 2 will be reported as coming from 192.0.2.3

Container to Container



To monitor traffic between all containers from the host you must use the docker0 interface. You could also select the veth* of just the container(s) that you would like to monitor.

From Host

Container 1 running without **`--net=host`** option.

Container 2 running without **`--net=host`** option.

Docker0

Traffic from the host will not be reported.

Traffic from Container 1 will be reported as coming from 192.0.2.2

Traffic from Container 2 will be reported as coming from 192.0.2.3

veth1

Traffic from Container 1 will be reported as coming from 192.0.2.2

veth1 && veth2

Traffic from Container 1 will be reported as coming from 192.0.2.2

Traffic from Container 2 will be reported as coming from 192.0.2.3

From Container 1

Container 1 running with **`--net=host`** option.

Container 2 running without **`--net=host`** option.

Docker0

Traffic from the host will not be reported.

Traffic from Container 1 will be reported as coming from 192.0.2.2

Traffic from Container 2 will be reported as coming from 192.0.2.3

Container 1 local interface

will not exist due to running with **`--net=host`**.

veth2

Traffic from Container 2 will be reported as coming from 192.0.2.3

Host to Network

Container 1 run with `--net=host` option.

Container 2 run without `--net=host` option.

From Host

Host interface

Traffic from the host will be reported with an IP of 192.0.2.1

Traffic from Container 1 will be reported with an IP of 192.0.2.1

Traffic from Container 2 will be reported with an IP of 192.0.2.1

From Container

Host interface

Traffic from the host will be reported with an IP of 192.0.2.1

Traffic from Container 1 will be reported with an IP of 192.0.2.1

Traffic from Container 2 will be reported with an IP of 192.0.2.1

Configure NNM on the Docker Host

Before You Begin

Install an instance of NNM on the Docker host.

Steps

1. Navigate to **`https://<IP address or hostname>:8835`**, which will display the NNM web front end to log in.

   Refer to the Configure NNM section of the NNM user guide for configuration instructions.

2. In step 5 of the Configure NNM instructions, configure the monitored network interfaces depending on your needs, outlined in the Monitored Interfaces section of this guide.

Introduction to Gigamon

Recognized by Gartner and others as the market leading solution, Gigamon provides active visibility into physical and virtual network traffic, enabling stronger security and superior performance. Gigamon's Visibility FabricTM and GigaSECURE®, the industry's first Security Delivery Platform, deliver advanced intelligence so that security, network and application performance management solutions in enterprise, government and service provider networks operate more efficiently and effectively.

Gigamon's Visibility Fabric™ has access to bidirectional network traffic so it has the ability to observe the exchange of public keys at the start of any transaction. Private keys are securely stored on the system. The power of the GigaSMART® traffic intelligence engine can decrypt the traffic and forward it to tools like NNM for analysis. Each GigaSMART module contains high-performance compute engines that have hardware performance accelerators to handle SSL traffic.

SSL Decryption is not limited to specific ingress ports or where the GigaSMART engine is located within the Visibility Fabric. Any traffic received on any network port in the cluster of Gigamon visibility nodes can take advantage of SSL Decryption. And that traffic can be sent to any tool ports in the cluster. This is an important attribute because not every node in the cluster needs to have the SSL Decryption capability. Furthermore, additional SSL Decryption throughput can be achieved by adding more GigaSMART modules to the cluster, allowing inspection to grow as SSL processing needs increase.

GigaSMART SSL-decryption functionality can be provided for NNM tools within the following Gigamon devices:

- GigaVUE-HD4/HD8
- GigaVUE-HC2
- GigaVUE-HB1

This section describes the steps to integrate NNM with Gigamon's solution, as well as an example deployment strategy.

SSL Decryption with NNM

SSL Overview

If an attacker is able to intercept all data being sent between a browser and a web server, they can see and use that information. Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), provide privacy and data integrity allowing secure transmission of sensitive information such as credit card numbers, social security numbers, and login credentials. SSL decryption uses keys to decode the traffic between the client and server so you are only going to be able to decrypt traffic if you have access to the private key used to encrypt it.

NNM and SSL Encrypted Traffic

As websites and services begin to default to encrypted connections, you can use a decryption appliance with NNM to improve visibility to your network infrastructure by decrypting encrypted traffic and eliminating blind spots.

In order for NNM to successfully detect threats and vulnerabilities within encrypted traffic, a decryption appliance must be employed which will decrypt the SSL traffic and enable NNM to successfully process these packets.

Decryption Limitations

A decryption appliance will provide NNM the ability to successfully process encrypted traffic, however, additional technologies also exist that could still prevent NNM from being able to process packets from some sessions. The following are two of the most common ways that sessions are further secured that will prevent traffic from being able to be processed by NNM.

HTTP Strict Transport Security (HSTS)

HSTS is a web security policy mechanism which allows web servers to require clients to communicate via encrypted channels. HSTS is used in order to prevent SSL stripping attacks which convert a secure HTTPS connection into a plain HTTP connection.

HSTS Preloading and Public Key Pinning

When connecting to an HSTS host for the first time, the browser will not know whether or not to use a secure connection. Consequently, an attacker could prevent the browser from ever connecting

securely. To mitigate this attack, browsers include a preloaded list of websites that want HSTS enforced by default, like Google, Dropbox, and Facebook, which can prevent detection by NNM. Also, browsers include a variation of certificate pinning using the HSTS mechanism. A preloaded set of public key hashes in the HSTS configuration limits the valid certificates to only those which indicate the specified public key.

Configure SSL Server

Prior to configuring the Gigamon appliance you need to obtain the Private Key(s) from any local web servers you wish to monitor with NNM. The Private Key(s) are necessary for Gigamon to perform SSL decryption.

Configure Gigamon

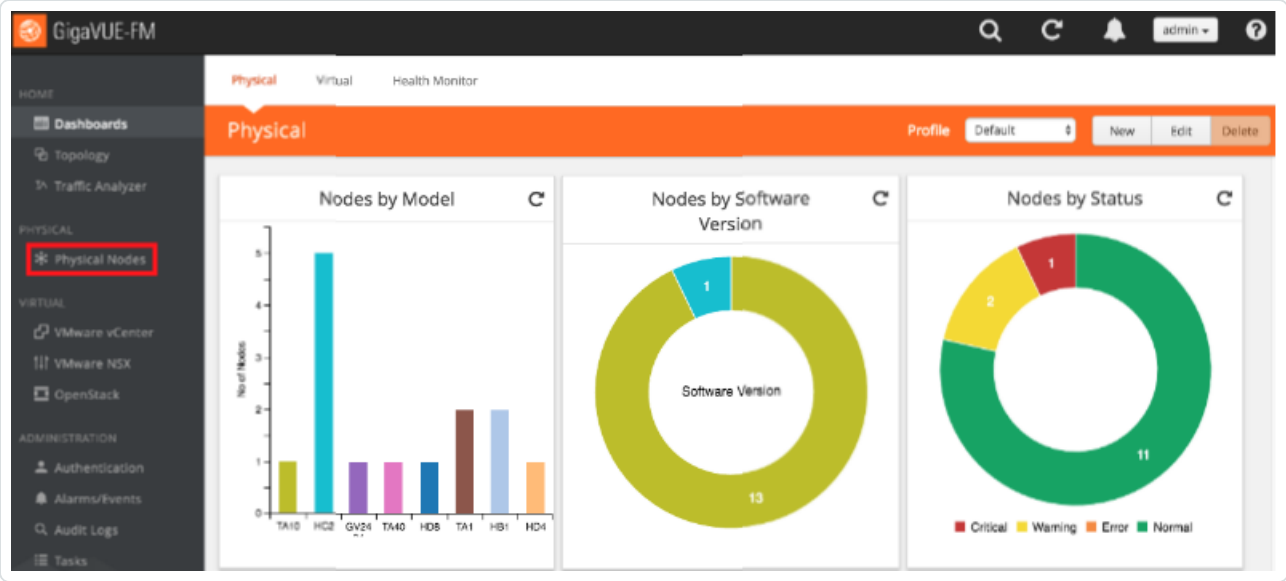
This section contains the following instructions for configuring the Gigamon decryption appliance for use with NNM:

- Configure the Tool Port
- Configure the GS Group
- Configure the GS Operation
- Configure the SSL Keychain Password
- Upload the Private Key
- Create an SSL Service
- Create a Map

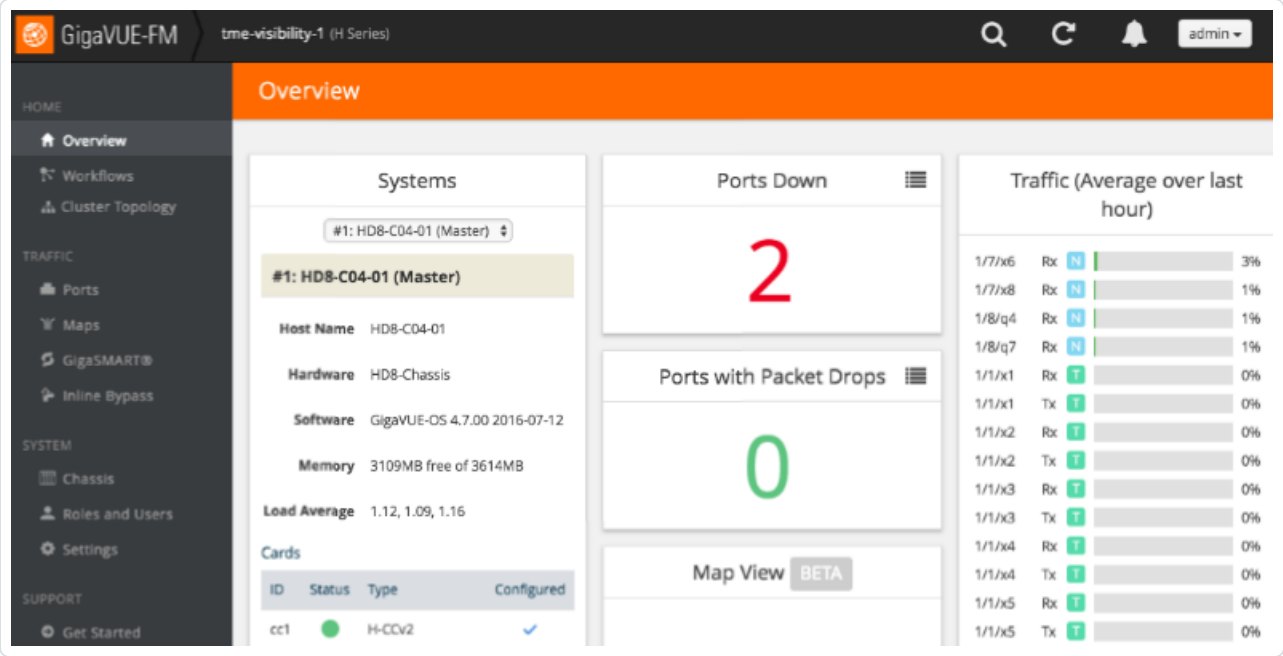
Configure the Tool Port

Steps

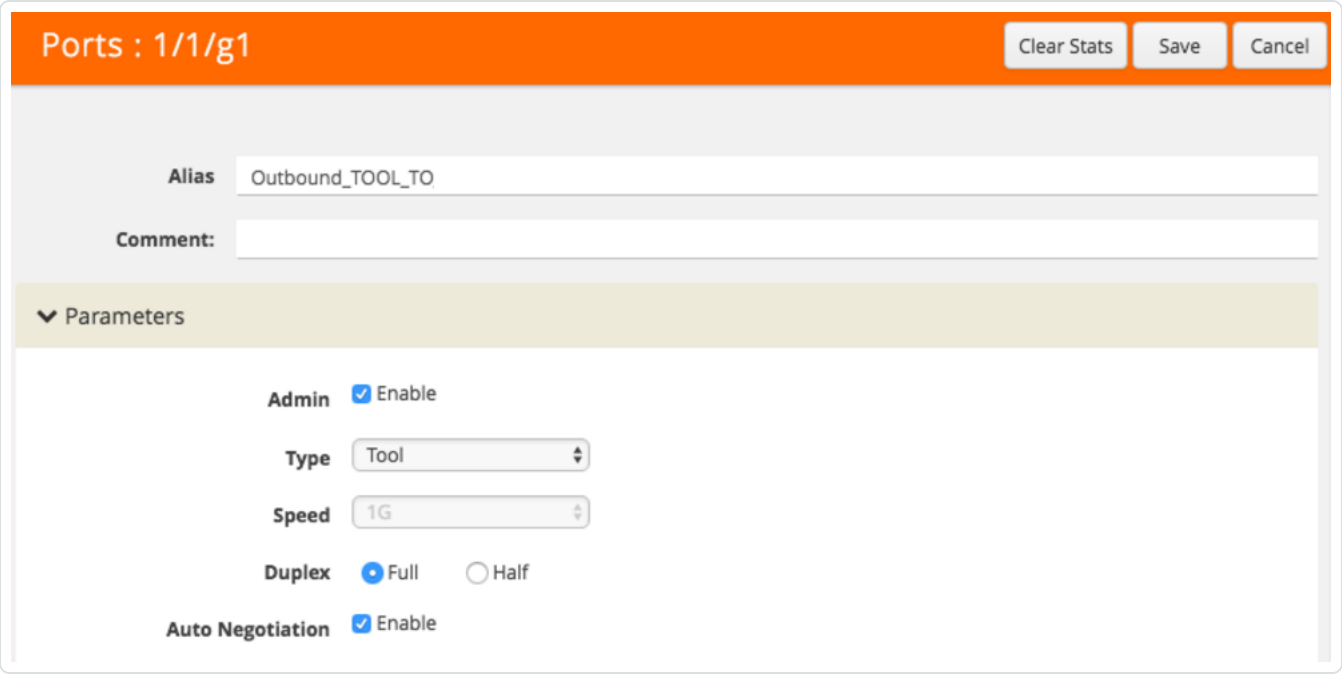
1. In the **GigaVUE-OS** interface, on the left side bar, select **Physical Nodes**. The **Physical Nodes** screen appears.



2. Select the **Physical Node** you wish to configure. The **Overview** screen appears.



3. On the left side bar, select **Ports**. The **Ports** tab appears.
4. On the **Ports** tab, in the list, select the port that is connected to NNM. The **Ports** window for the selected port appears.
5. In the upper right corner, click the **Edit** button. The **Ports** page for the selected port appears.

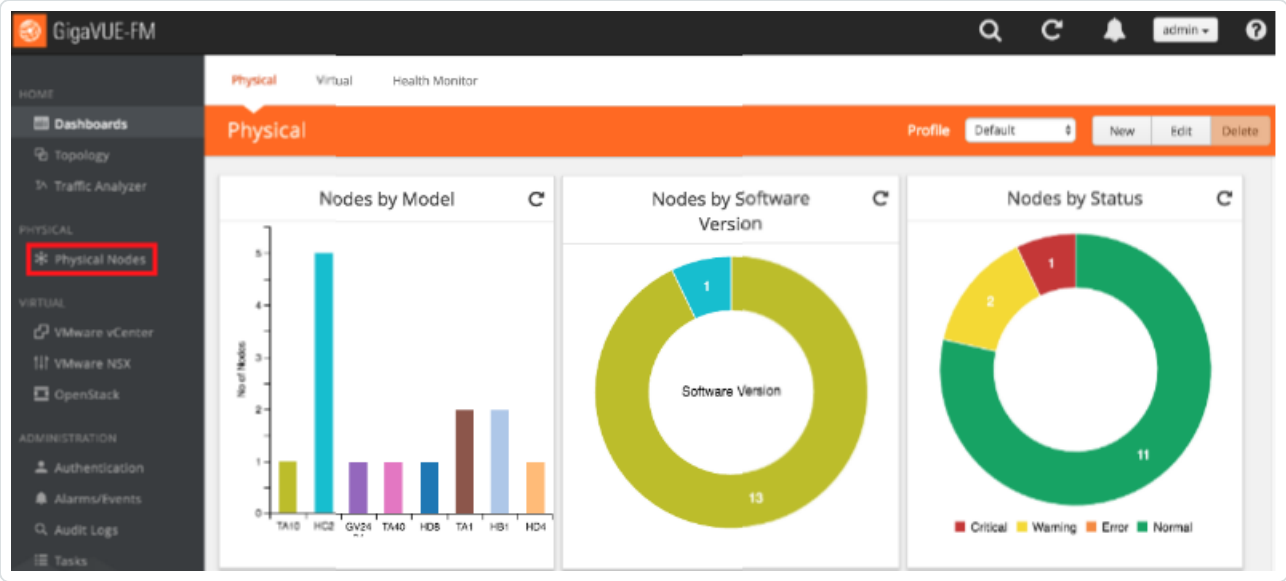


6. In the **Alias** box, enter a name for the port (e.g., Outbound_TOOL_TO_PVS).
7. In the **Admin** row, select the **Enable** check box.
8. In the **Type** drop-down box, select **Tool**.
9. In the **Duplex** row, select **Full**.
10. In the **Auto Negotiation** row, select the **Enable** check box.
11. In the upper right corner of the page, click **Save**.

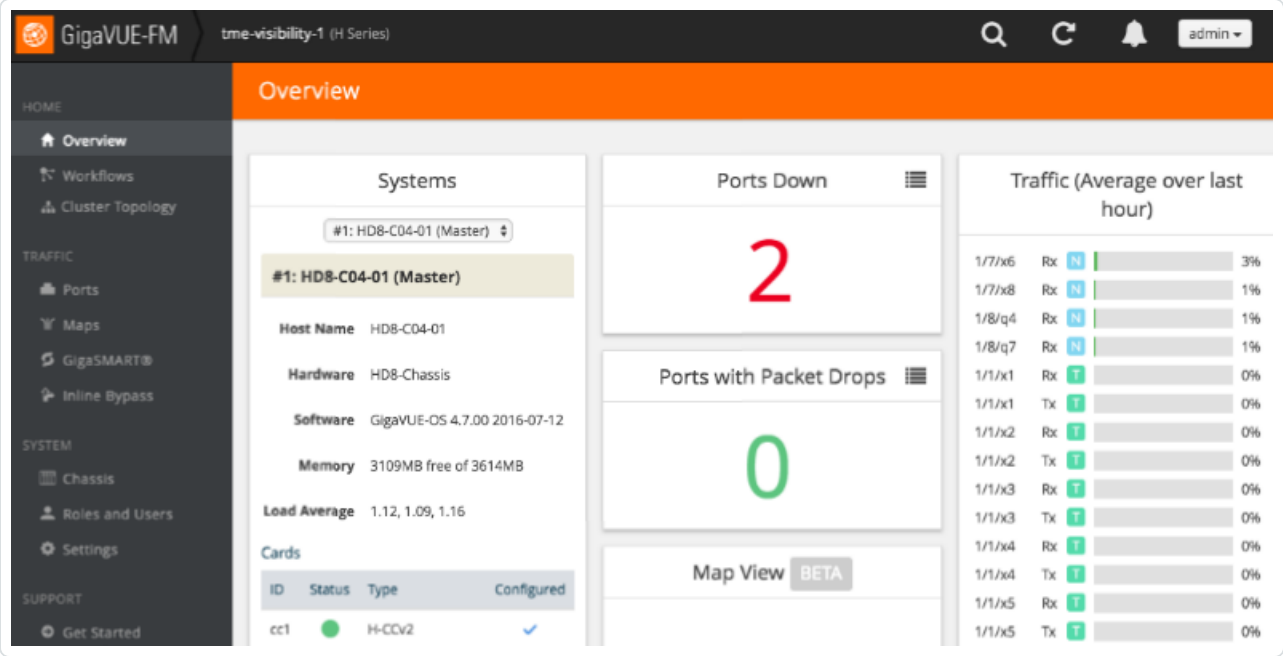
Configure the GS Group

Steps

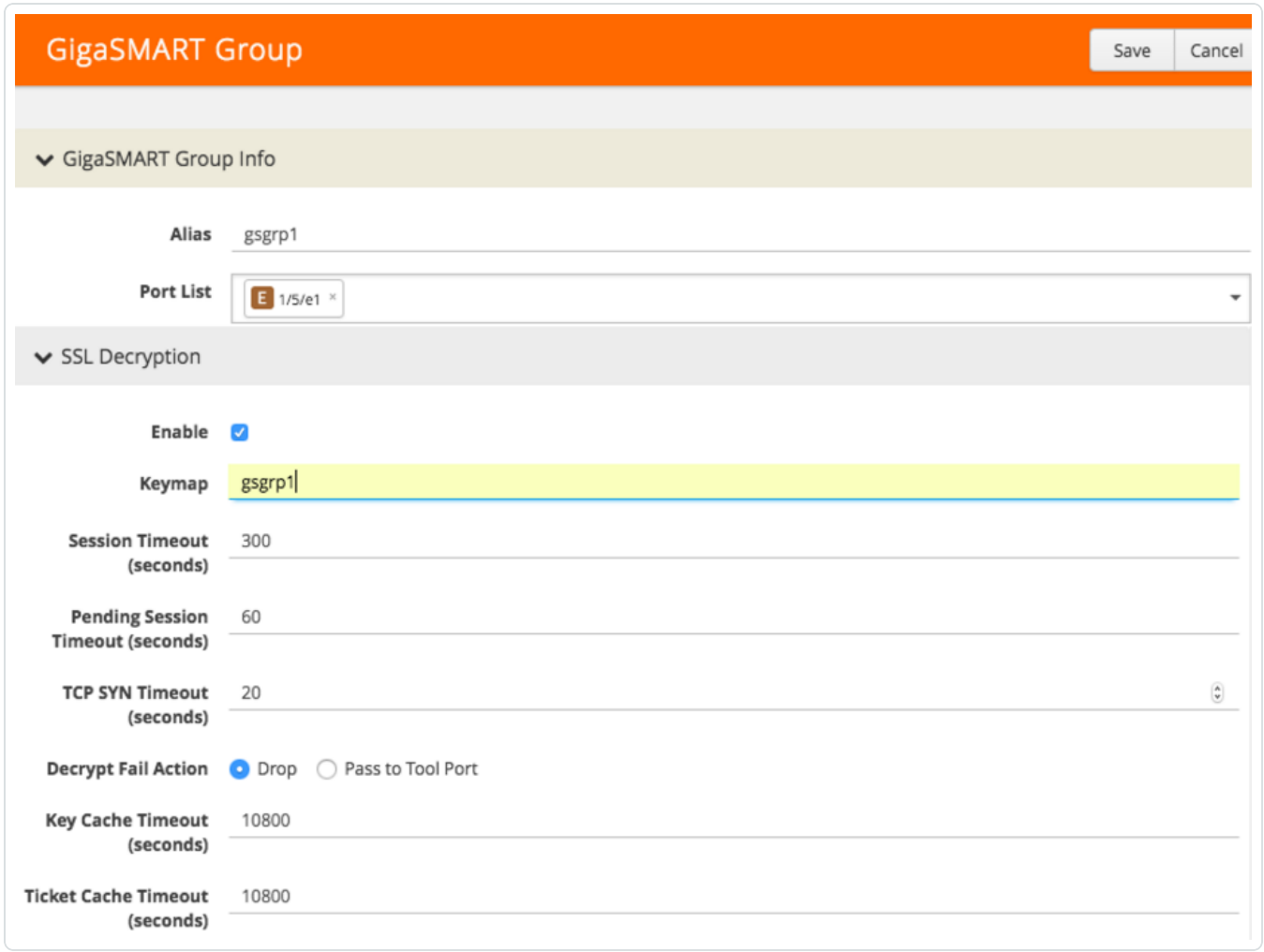
1. In the **GigaVUE-OS** interface, on the left side bar, select **Physical Nodes**. The **Physical Nodes** screen appears.



2. Select the **Physical Node** you wish to configure. The **Overview** screen appears.



3. On the left bar, select **GigaSMART®**. The **GS Operations** tab appears.
4. On the top navigation bar, select **GS Groups**. The **GS Groups** tab appears.
5. In the upper right corner, click the **New** button. The **GS Group** page appears.



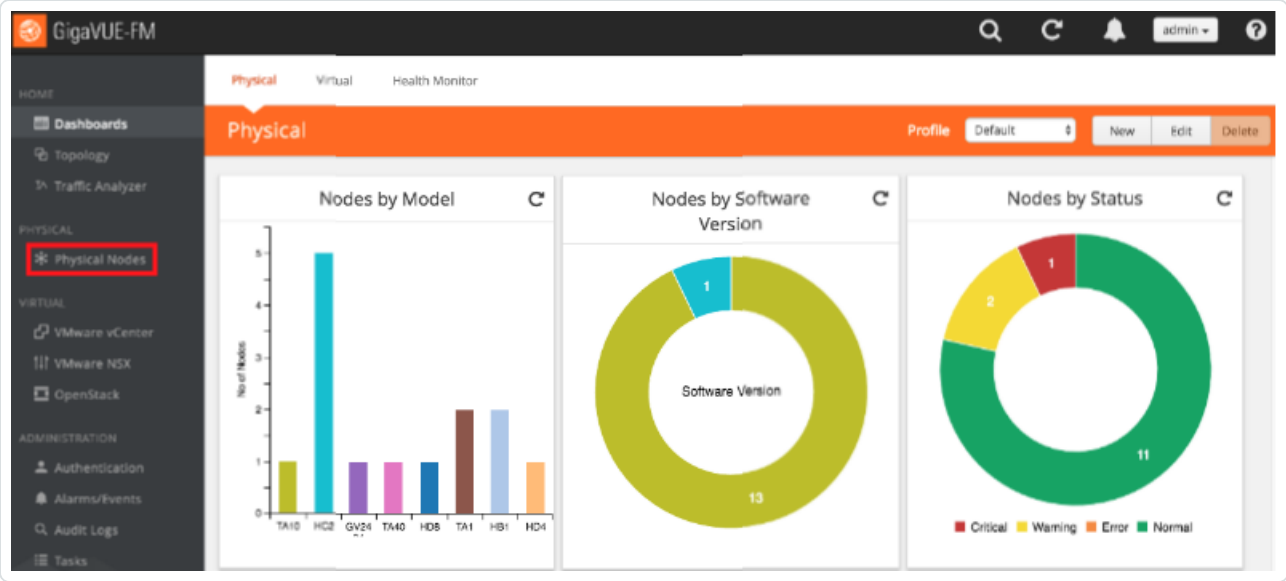
6. In the **GS Group Info** section, in the **Alias** box, enter a name for the group (e.g., gsgrp1).
7. In the **Port List** drop-down box, select the tool port you created in Configure the Tool Port.
8. In the **SSL Decryption** section, select the **Enable** check box.
9. In the **Keymap** box, enter the alias you set in step 2.
10. In the **Decrypt Fail Action** row, select either **Drop** or **Pass to Tool Port**.
11. In the **Non SSL Traffic** row, select either **Drop** or **Pass to Tool Port**.

12. In the upper right corner of the page, click **Save**.

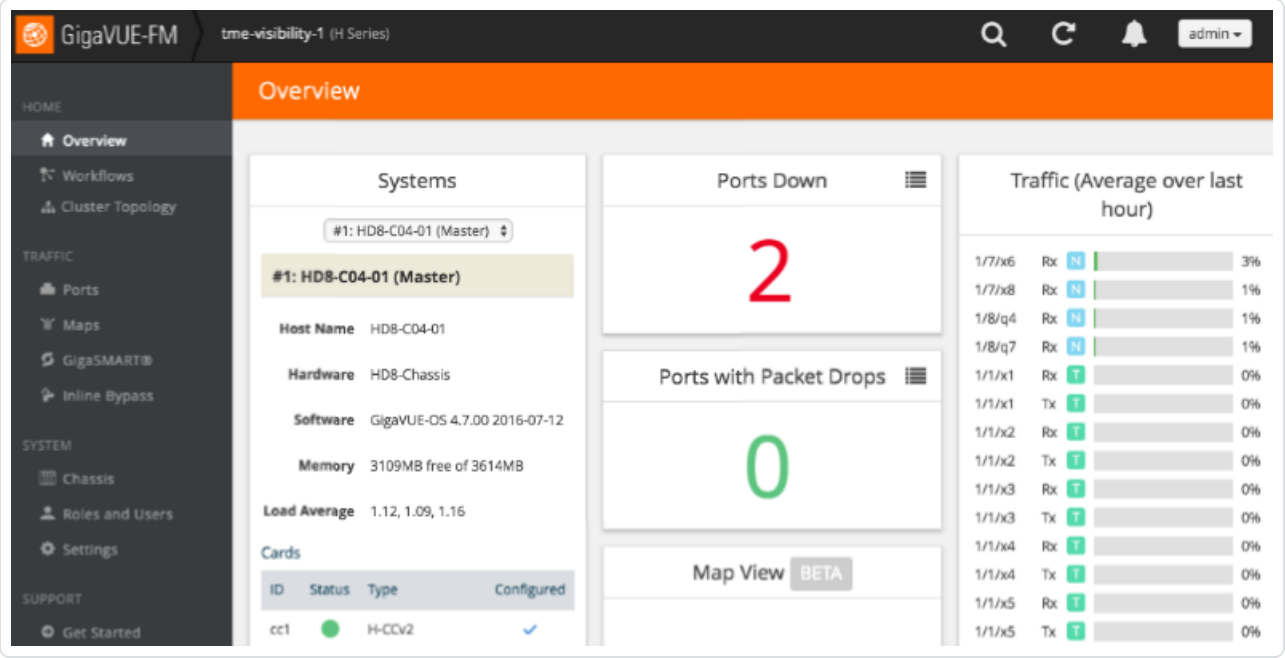
Configure the GS Operation

Steps

1. In the **GigaVUE-OS** interface, on the left side bar, select **Physical Nodes**. The **Physical Nodes** screen appears.



2. Select the **Physical Node** you wish to configure. The **Overview** screen appears.



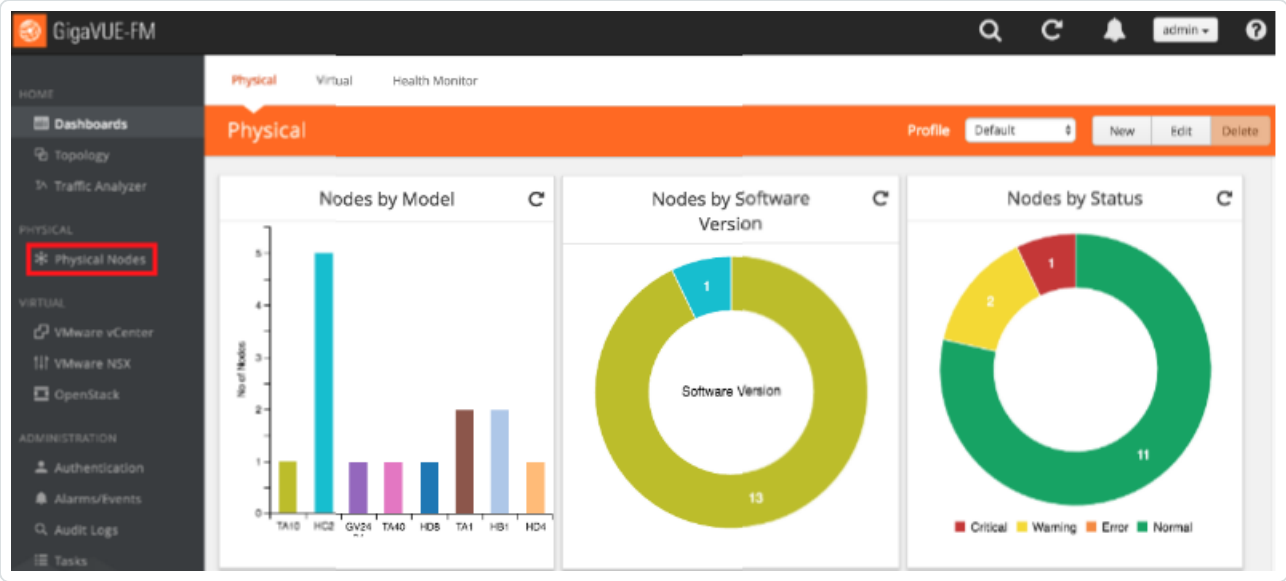
3. On the left bar, select **GigaSMART®**. The **GS Operations** tab appears.
4. In the upper right corner, click the **New** button. The **GS Operations** page appears.

5. In the **Alias** box, enter a name for the operation (e.g., SSL_Decryption).
6. In the **GS Groups** drop-down box, select the group you created in Configure the GS Group.
7. In the **GS Operations** drop-down box, select **SSL Decryption**. The **SSL Decryption** section appears.
8. In the **In Port** box, enter **any**.
9. In the **Out Port** box, enter **auto**.
10. In the upper right corner of the page, click **Save**.

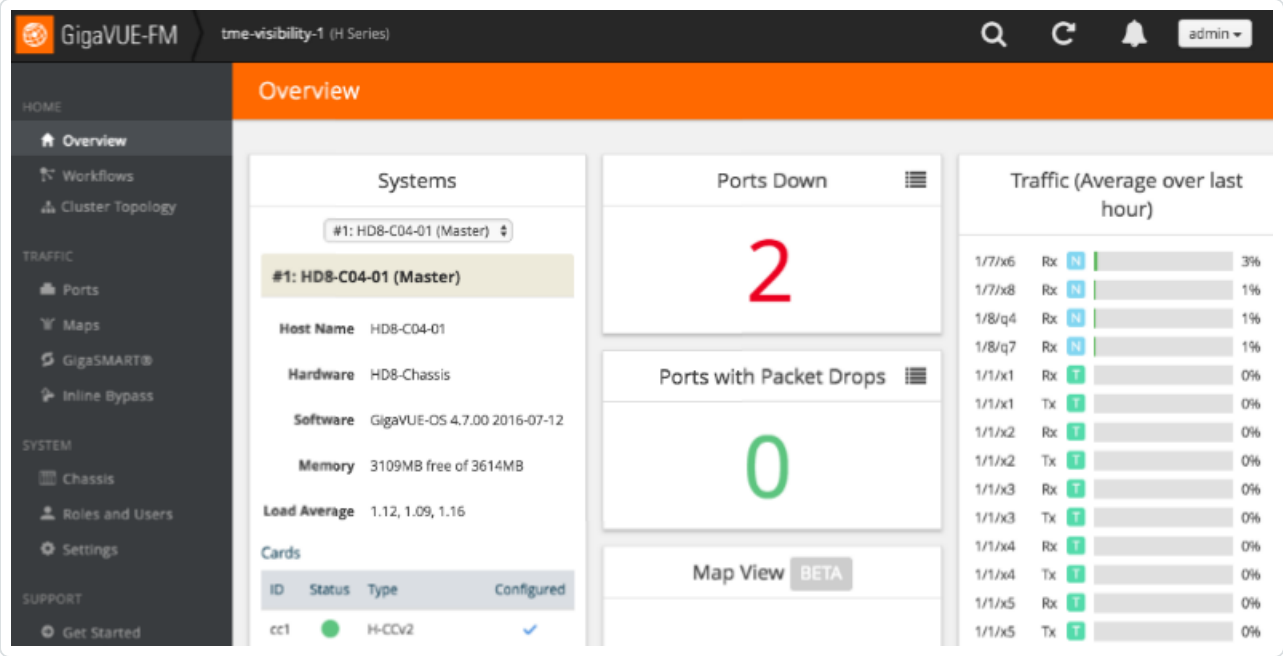
Configure the SSL Keychain Password

Steps

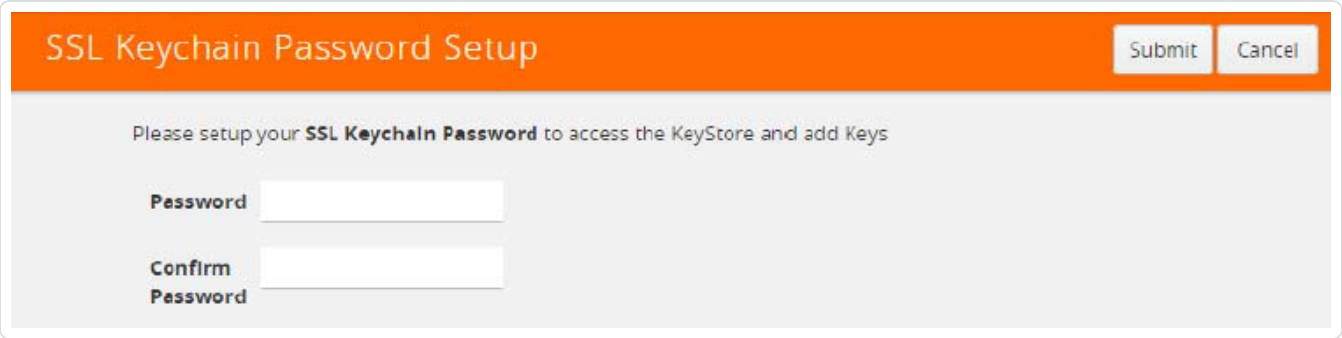
1. In the **GigaVUE-OS** interface, on the left side bar, select **Physical Nodes**. The **Physical Nodes** screen appears.



2. Select the **Physical Node** you wish to configure. The **Overview** screen appears.



3. On the left bar, select **GigaSMART®**. The **GS Operations** tab appears.
4. On the top navigation bar, select **SSL Decryption**. The **SSL KEYS** tab appears.
5. In the upper right corner, click the **Password** button. The **SSL Keychain Password Setup** page appears.



6. In the **Password** and **Confirm Password** boxes, enter a password.
7. In the upper right corner of the page, click **Submit**.

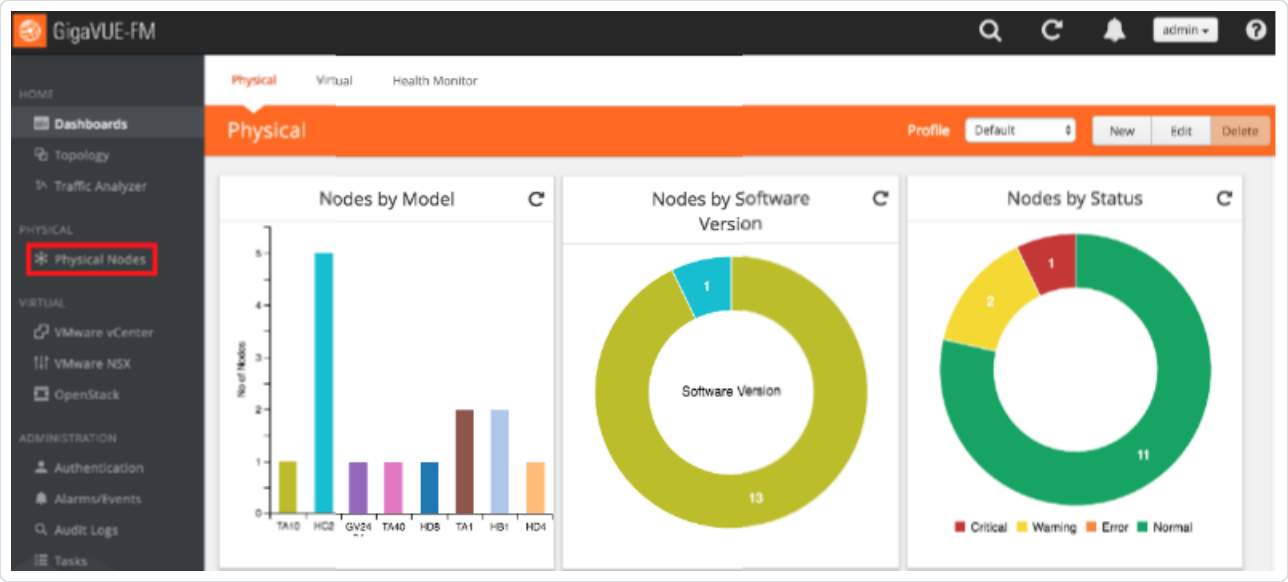
Upload the Private Key

Before You Begin

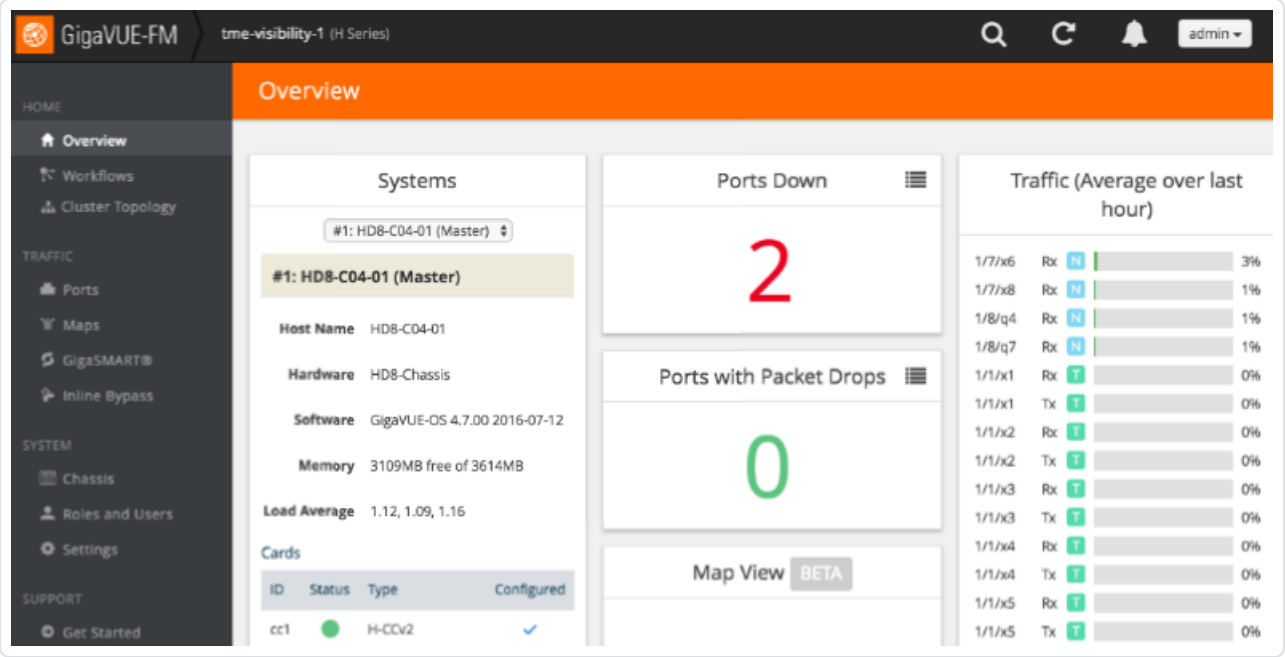
To send traffic through the proxy and have it decrypted by the Gigamon appliance, set the browser, device, or application proxy settings to allow internet access by the proxy's IP and port number.

Steps

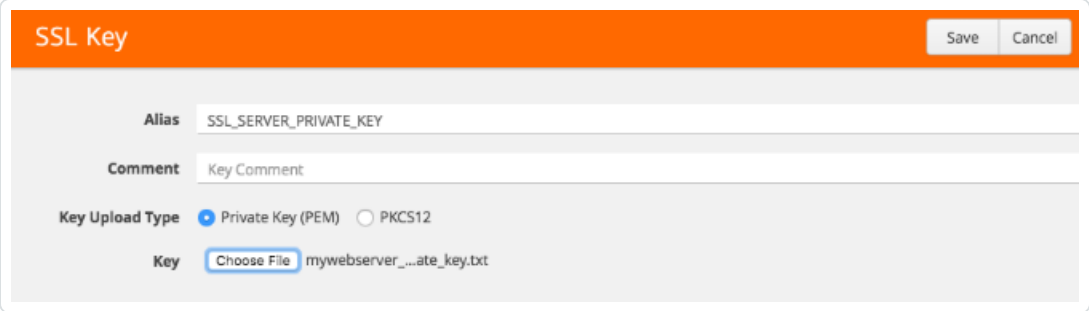
1. In the **GigaVUE-OS** interface, on the left side bar, select **Physical Nodes**. The **Physical Nodes** screen appears.



2. Select the **Physical Node** you wish to configure. The **Overview** screen appears.



3. On the left bar, select **GigaSMART®**. The **GS Operations** tab appears.
4. On the top navigation bar, select **SSL Decryption**. The **SSL KEYS** tab appears.
5. In the upper right corner, click the **Install** button. The **SSL Key** page appears.

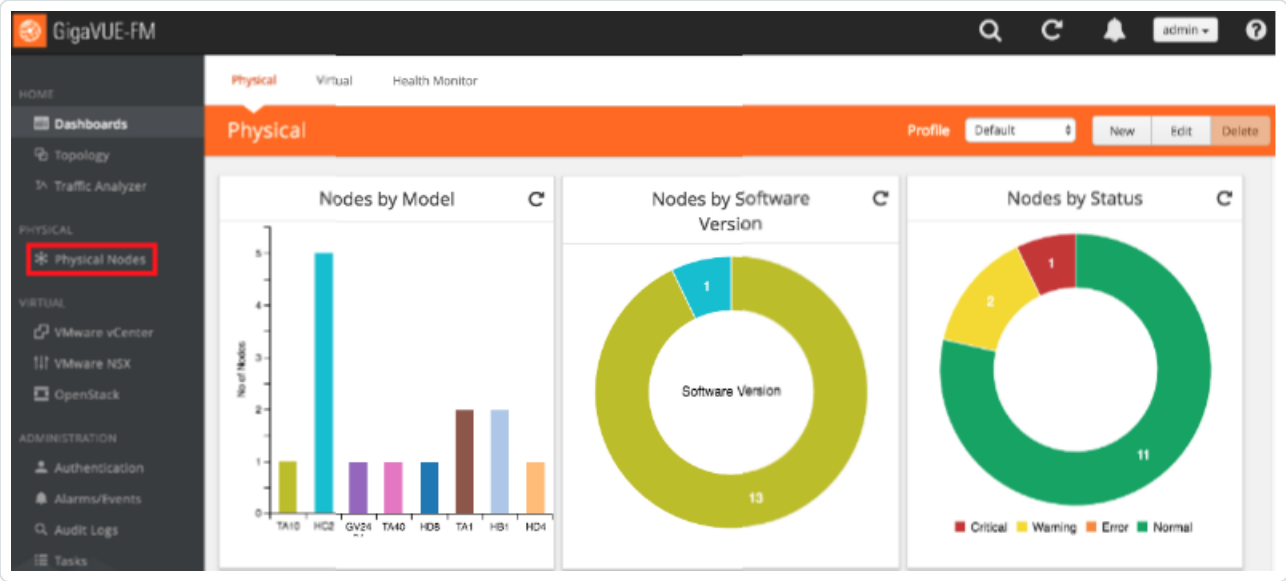


6. In the **Alias** box, enter a name for the SSL key.
7. In the **Key Upload Type** row, select **Private Key**.
8. In the **Key** row, select **Choose File**. Navigate to and upload the SSL key file.
9. In the upper right corner of the page, click **Save**.

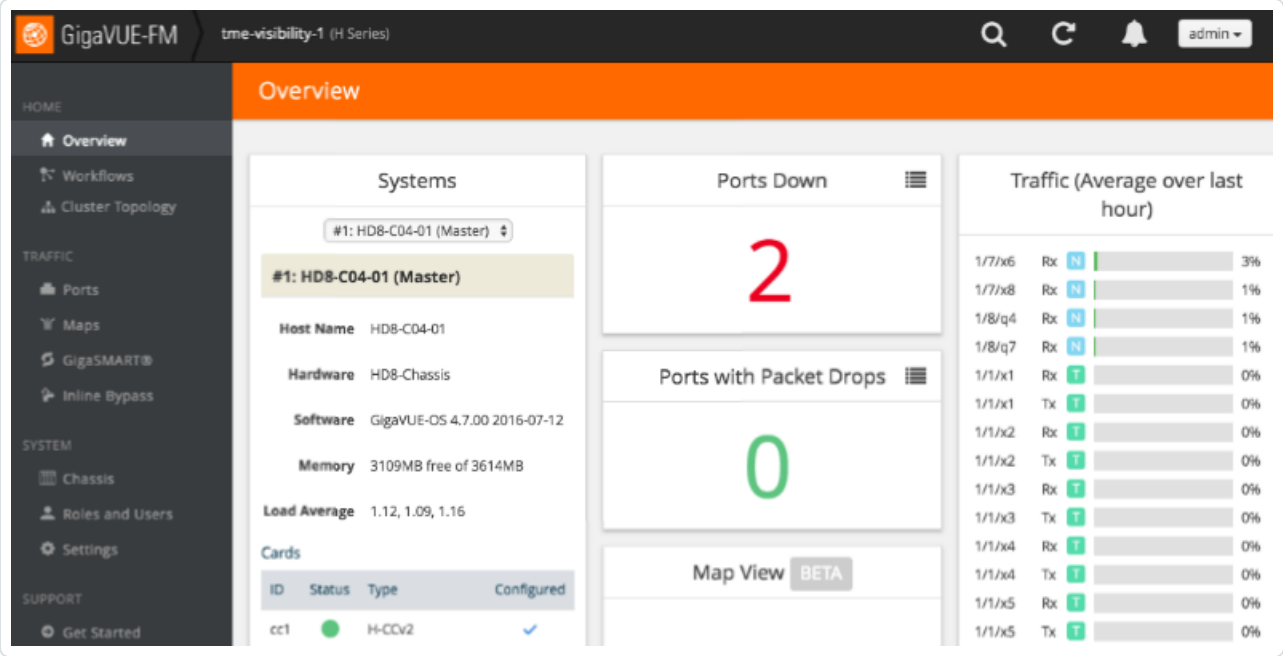
Create an SSL Service

Steps

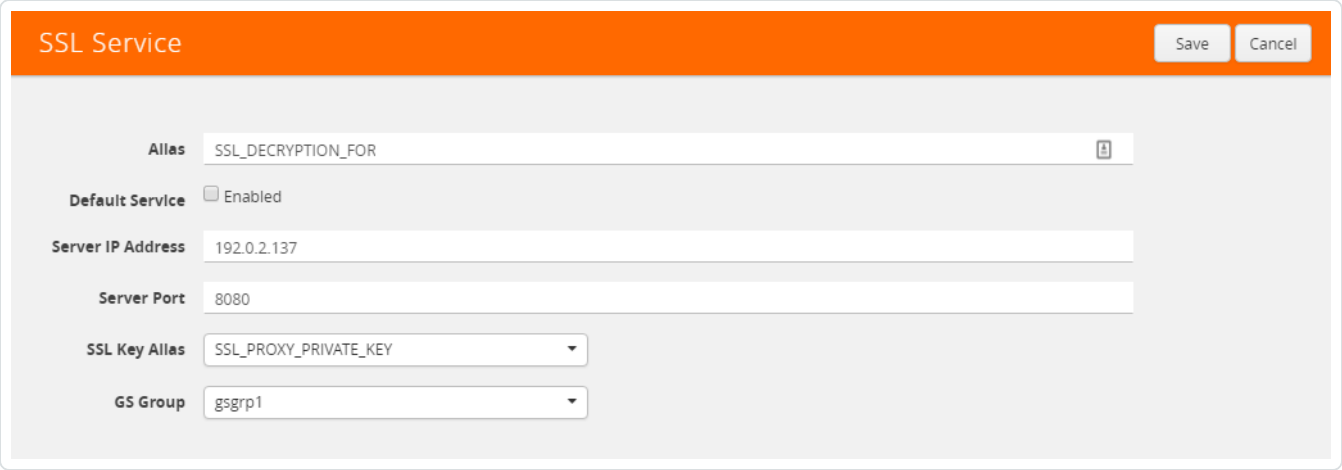
1. In the **GigaVUE-OS** interface, on the left side bar, select **Physical Nodes**. The **Physical Nodes** screen appears.



2. Select the **Physical Node** you wish to configure. The **Overview** screen appears.



3. On the left bar, select **GigaSMART®**. The **GS Operations** tab appears.
4. On the top navigation bar, select **SSL Decryption**. The **SSL Services** tab appears.
5. In the upper right corner, click the **New** button. The **SSL Service** page appears.

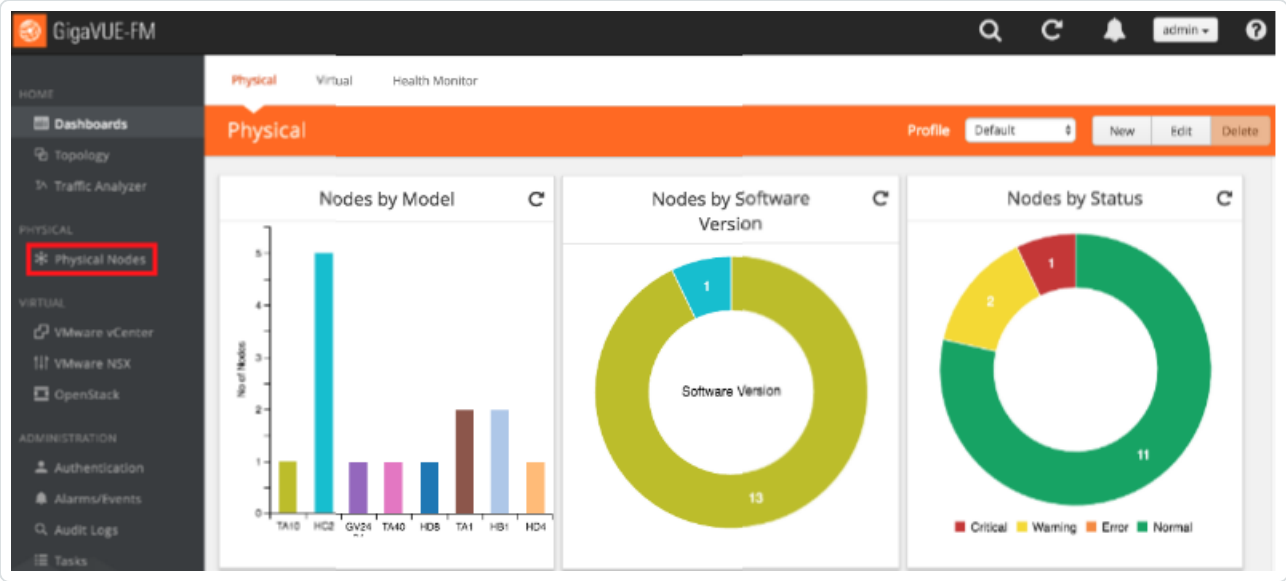


6. In the **Alias** box, enter a name for the SSL service.
7. In the **Server IP Address** box, enter the IP address of the server.
8. In the **Server Port** box, enter the server SSL port.
9. In the **SSL Key Alias** drop-down box, select the private key you uploaded in Upload the Private Key.
10. In the **GS Group** drop-down box, select the group you created in Configure the GS Group.
11. In the upper right corner of the page, click **Save**.

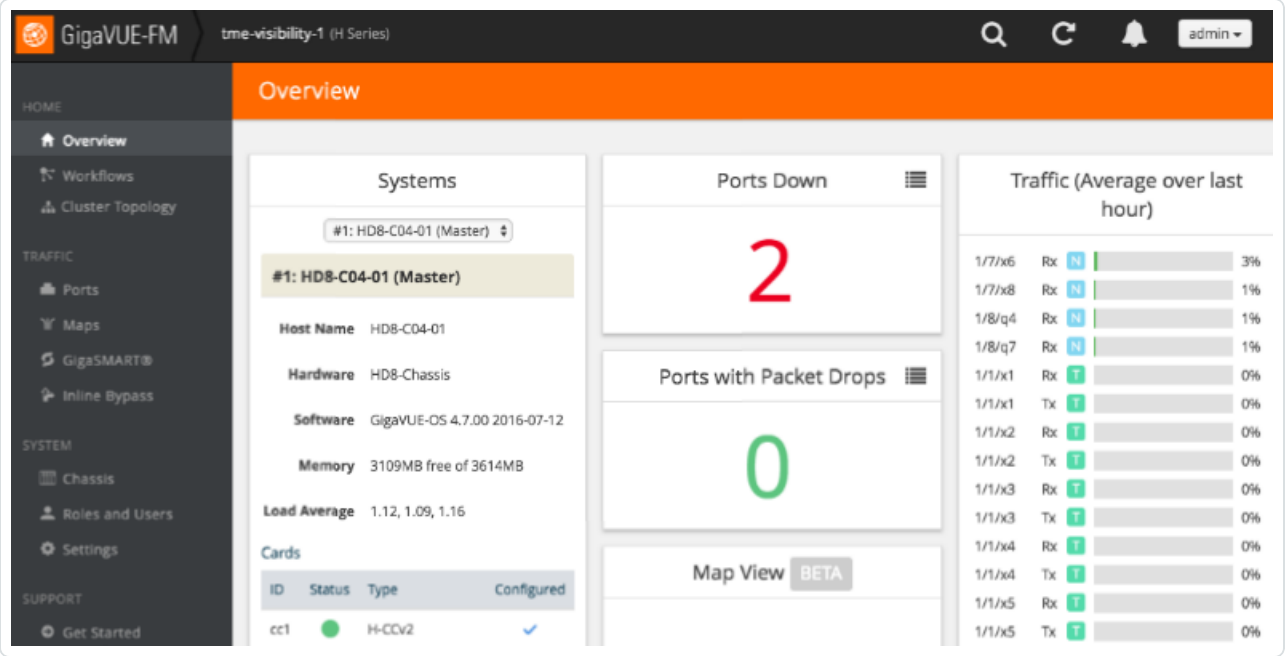
Create a Map

Steps

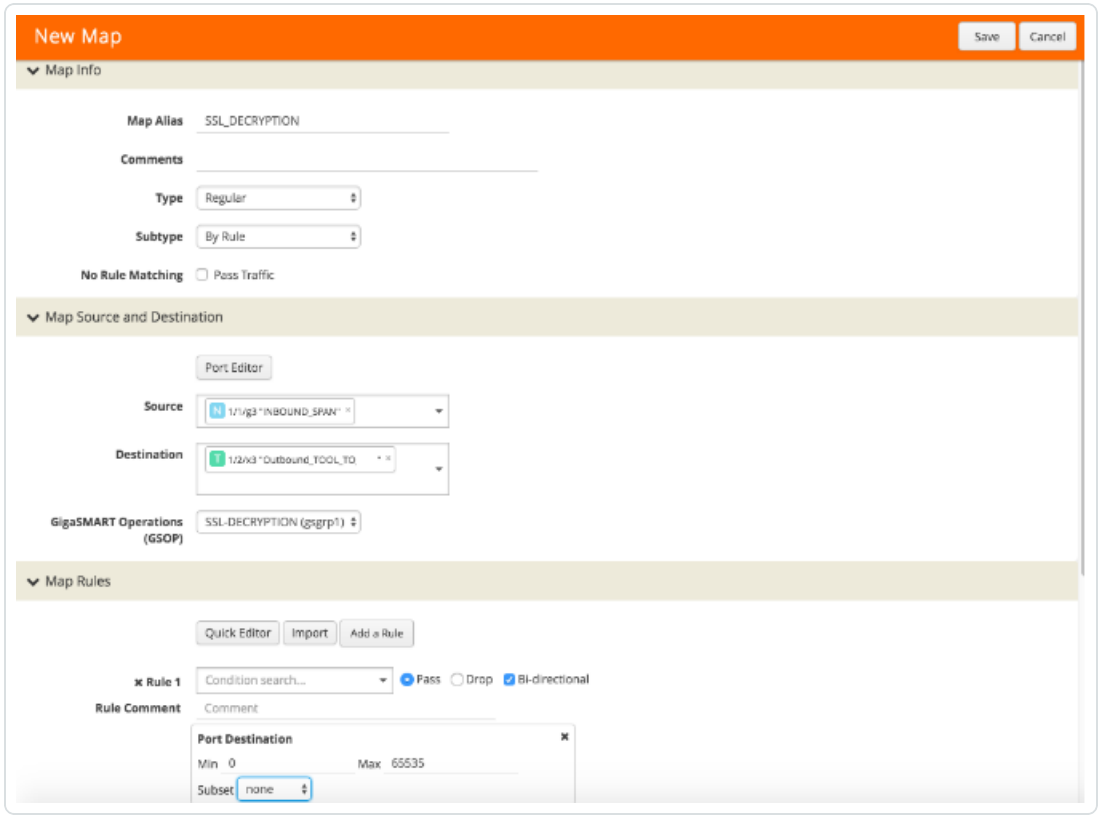
1. In the **GigaVUE-OS** interface, on the left side bar, select **Physical Nodes**. The **Physical Nodes** screen appears.



2. Select the **Physical Node** you wish to configure. The **Overview** screen appears.



3. In the **GigaVUE-OS** interface, on the left bar, select **Maps**. The **Maps** tab appears.
4. In the upper right corner, click **New**. The **New Map** page appears.



5. In the **Map Info** section, in the **Map Alias** box, enter a name for the map.
6. In the **Type** drop-down box, select **Regular**.
7. In the **Sub Type** drop-down box, select **By Rule**.
8. In the **Map Source and Destination** section, in the **Source** drop-down box, select one or more source ports.
9. In the **Destination** drop-down box, select the tool port you created in Configure the Tool Port.
10. In the **GSOP** drop-down box, select the operation you created in Configure the GS Operation.
11. In the **Map Rules** section, click the **Add a Rule** button.
12. In the **x Rule 1** row, select the **Pass** option and the **Bi Directional** check box.
13. In the **x Rule 1** drop-down box, select **Port Destination**. The **Port Destination** sub-section appears.

14. In the **Min** box, enter 0.
15. In the **Max** box, enter 66535.
16. In the upper right corner of the page, click **Save**.

Example Gigamon Deployment

For additional deployment use cases refer to the Gigamon Deployment Guides on www.-gigamon.com under **Resources**.

Virtual Machines

Virtual Machine	IP	Function
Cent6-NNM	192.0.2.46	NNM / Traffic capture

Introduction to Waterfall

Tenable Network Security's Nessus Network Monitor (NNM) provides continuous visibility into the systems and services running on your networks, for unmatched asset insight. From legacy assets to the latest technologies, and from IT to OT, it illuminates blind spots so you can see and protect your entire environment. The Nessus Network Monitor offers enhanced OT support, including asset discovery and protocol detection, for passively monitoring industrial control systems (ICS), SCADA systems, and other operational technology. This gives security teams a safe and non-intrusive way to discover and monitor sensitive critical infrastructure systems. The Nessus Network Monitor also detects new and unmanaged assets – spanning operating systems, network devices, hypervisors, databases, mobile devices, web servers, cloud applications, IoT devices, critical infrastructure such as ICS / SCADA, and more.

Waterfall Security Solutions produces a wide spectrum of solutions based on and complementing Waterfall's world-leading Unidirectional Security Gateways. Waterfall Unidirectional Security Gateways replace firewalls in industrial network environments, providing absolute protection to control systems and operations networks from attacks originating on external networks. Unidirectional Gateway solutions come in pairs: the TX appliance contains a laser, and the RX appliance contains an optical receiver. The Gateway pair transmits information out of an operations network, but is incapable of propagating any virus, DOS attack, human error or any information at all back into the protected network. By seamlessly connecting to the data source and replicating it on the other side, Waterfall's Unidirectional Security Gateways provide access to real-time data, without introducing threats to operations networks which accompany firewall connections.

By integrating Waterfall's Unidirectional Security Gateway with Tenable Network Security's Nessus Network Monitor, customers can deploy their vulnerability management solution on the IT side of their infrastructure and get unparalleled visibility into both their IT and OT infrastructure.

Waterfall Integration Test

Test Goals

The test conducted during an on-site POC came to prove that any data flow going from a data-source, through the Waterfall Unidirectional channel to Tenable NNM, arrived at NNM as expected, identical to the results that were achieved without the Waterfall channel; and hence, proving the integration between Waterfall USG and Tenable NNM as fully seamless for end users.

Test Method

A Unidirectional Security Gateway and NNM server were installed at Tenable's lab.

- The TX side of the USG was connected to a span port of the switch on the OT side, simulating a general use-case of generic traffic channeling.
- The RX side of the USG was connected to the NNM server, on the IT side.
- Modbus traffic was generated on the OT side, flowing through the USG and arriving at NNM on the IT side.
- The data on the NNM was compared to the expected results, based on the results of a direct connection between the data source and the NNM (without any intermediate channel).

Test Results

The traffic arriving at the NNM from the USG USG RX Agent was compared to the expected results, and was confirmed as identical. Therefore, the test had concluded as successful, and the technical aspect of the integration complete.

Waterfall Architecture and Data Flow

1. Waterfall Unidirectional Security Gateway is implemented on customer's site.
2. The TX Agent is connected to the Industrial network, while the RX Agent is connected to the lower-trust network, either a corporate network or the Internet (both connections are with a standard RJ45 Ethernet copper cable).
3. The channel installed on both Waterfall Agents is depending on customer's needs and architecture:
    - If the source of the data is a specific protocol or industrial system, a corresponding channel is installed, and the TX agent will be connected directly to the system or data source.
    - If multiple or unspecific sources are required, then the Waterfall Channel for Tenable is installed, and the TX Agent will be connected to the span port of the switch, in which the traffic flows.

    Caution: The Waterfall Channel for Tenable forwards all traffic going through the span port. To prevent overflow of unnecessary information, the data needs to be filtered, either before or at the Waterfall TX Agent.
4. The collected data is forwarded through the unidirectional channel, from the TX side to the RX side.
5. The RX Agent forwards the data to the Tenable NNM server, whether locally installed or on the Cloud.
6. Users and operators will access the data at NNM, without any traffic getting back into the industrial network.

Install Waterfall

Waterfall provides on-site installation and configuration, please contact Waterfall Support Center for further assistance.

Below is a basic configuration for a Waterfall Ethernet Multicast channel.

> **Note:** According to customer's needs and architecture, different channels may be required. Different channels may require different installation and/or configuration steps. Please contact Waterfall Support Center for any assistance.

Tenable Channel

Configure the TX Agent

1. Click the **Waterfall Configuration** icon to open the **Waterfall Configuration** tool.
2. Under the **Stream** section in the navigation tree on the left, choose the **Ethernet Spoofing** channel.
3. Copy the **Channel** name: the same name must match on both TX and RX Agents.
4. Under **Adapter**, from the **Name** drop-down menu, select the **Network NIC**.

   > **Note:** This assumes the Network NICs have been assigned a valid IP for both the TX and RX hosts. This is normally configured during the initial Waterfall installation.

5. To add protocol or port filtering, add the required protocol and/or port into the **Filter expression** field. For example:
   - Single filter: tcp port 80
   - Multiple filters: tcp port 80 or tcp port 23
6. Click **Apply**.

Configure the RX Agent

1. Click the **Waterfall Configuration** icon to open the **Waterfall Configuration** tool.
2. Under the **Stream** section in the navigation tree on the left, choose the **Ethernet Spoofing** channel.
3. Ensure the **Channel** name matches the TX Agent name.
4. Under **Adapter**, from the **Name** drop-down menu, select the **Network NIC**.

   > **Note:** This assumes the Network NICs have been assigned a valid IP for both the TX and RX hosts. This is normally configured during the initial Waterfall installation.

5. Under **Target addresses**, check the following boxes:
   - **Keep MAC**
   - **Keep IP**
   - **Keep Port**
6. Click **Apply**.

Additional Waterfall Support

For assistance, contact Waterfall's Technical Assistance Support Team at support@waterfall-security.com.

Introduction to Splunk

Splunk is a Security Information and Event Management (SIEM) application used by Tenable customers to collect and store events from assets within the organization. Tenable Nessus Network Monitor provides the SIEM Pull Service to enhance the vulnerability management process through event collection and analysis. The SIEM Pull Service looks for *risk-altering events* in collected data and send the data to Tenable Vulnerability Management or Tenable Security Center for use in the Risk Based Vulnerability Management (RBVM) program. A risk-altering event is an event that changes an asset's risk posture (for example, starting or stopping a service) . When these events occur, and the event matches the core query provided with plugins, the SIEM Pull Service sends the data to Tenable Nessus Network Monitor, then to Tenable Vulnerability Management or Tenable Security Center.

The SIEM Pull Service monitors for the following four risk-altering event types:

Event Type	Description
Assets Discovery	Instances where assets are discovered using DHCP events.
User Account Activity	Instances where a user account on an asset is modified in one of the following ways: • Account is created or deleted • Account is added or removed to/from a group • Account password modified • Policy that affects user accounts is modified (i.e. password policy, lockout policy)
Software Detection	Instances where software is added or removed by a user or the software management system. For example: • RPM installations • Software added via YUM • Installations on Windows using standard install tools

	Note: This type does not include instances where binaries are copied on the system and run without execution.
Service Modification	Instances where the software service is modified in one of the following ways: • Service starts or stops • Service fails to start • Service reboots • Service is installed or uninstalled

For more information, see the following topics:

- DHCP Setup and Configuration
- Install the Splunk Universal Log Forwarder
- SIEM Pull Service Queries

DHCP Setup and Configuration

To monitor DHCP events, Tenable provides two examples using Windows Server 2019 and CentOS7 as DHCP servers. These configurations use many of the default settings. However, Tenable does not provide support for configuring DHCP services on a customer's network.

Windows DHCP

DHCP server logs can be tracked in the same way as any other log type. You can add the Windows DHCP server logs during installation.

To add new log sources after installation, add entries to the `inputs.conf` file (see the configuration example below):

```
C:\Program Files\SplunkUniversalForwarder\etc\system\local\inputs.conf
```

Configuration example

Configuration file - `[monitor://C:/Windows\System32\dhcp]sourcetype = dhcp`

Settings -

- **crcSalt** - `<SOURCE>`
- **alwaysOpenFile** - `1`
- **disabled** - `false`
- **whitelist** - `Dhcp.+\.log`

Note: If you are not using the default install path for your Windows server, you can find the path in your server settings:

1. Open the DHCP Microsoft Management Console.
2. Right-click your server.
3. Click **Properties**.
4. Open the **Advanced** tab.

   The **Audit log file path** is the installation path.

Linux DHCP

For Linux based DHCP servers, tail the DHCP log to add it to the `input.conf` file:

```
> /opt/splunkforwarder/bin/splunk add monitor /var/log/dhcpd.log
```

Install the Splunk Universal Log Forwarder

To set up the Splunk Universal Log Forwarder, download the version for your operating system from https://www.splunk.com/en_us/download/universal-forwarder.html, then follow the steps below.

> **Note:** If you don't already have an account, you need to create a free Splunk account to download the Universal Forwarder installation package(s).

- To install on Windows:
  1. Open the Windows Splunk installation package (e.g. `splunkforwarder-8.2.2.1-ae6821b7c64b-x64-release.msi`).
  2. Accept the **License Agreement**.
  3. Choose **Splunk Enterprise** (on-premises) or **Splunk Cloud**.
  4. Click **Customize Options**.
  5. Choose the file path to install the Universal Forwarder to.
  6. Click **Next**.
  7. (Optional) Enter the SSL certificate, certificate password, and SSL root CA.
  8. Click **Next**.
  9. Choose the Splunk Account type that the installation will create.
  10. Click **Next**.
  11. Select all the checkboxes under **Windows Event Logs**, **Performance Monitor**, and **Active Directory Monitoring** to ensure that all logs will be monitored.

      > **Tip:** To monitor DHCP logs, enter **C:\Windows\System32\DHCP** in the **Path to monitor** box.

  12. Click **Next**.
  13. Enter a **Username** and **Password** for the new Splunk account.
  14. Click **Next**.
  15. (Optional) Under **Deployment Server**, enter your deployment server IP and port.

16. Click **Next**.
17. Under **Receiving Indexer**, enter the receiving indexer IP and port.
18. Click **Next**.
19. Click **Finish**.

- To install on Linux:

1. Install the Linux installation package based on your distribution.
2. Refer to the Splunk Admin Manual to begin forwarding logs to your Splunk server.

Example installation:

1. Open CentOS.
2. Install the downloaded Universal Forwarder package. Enter the following command:

```
> rpm -Uvh /[PATH_TO_FILE]/splunkforwarder-[VERSION_NUMBER].rpm
```

3. Add the forwarding server. Enter the following command:

```
/opt/splunkforwarder/bin/splunk add forward-server X.X.X.X:9997
```

> **Note:** If this is the first time you are running a command using the Splunk client, you must accept the user agreement and a create a Splunk username and password. This user account will be used for Splunk management.

4. Install the audit service. If the audit service came pre-installed with your distribution, skip this step.
    a. Run the following commands in the listed order:

```
> sudo yum install audit
```

```
> sudo auditctl restart
```

```
> /opt/splunkforwarder/bin/splunk add monitor /var/log/audit/audit.log
```

```
> /opt/splunkforwarder/bin/splunk add monitor /var/log/messages
```

```
> /opt/splunkforwarder/bin/splunk add monitor /var/log
```

Note: To ensure the proper host is attributed in the Splunk query, Tenable Nessus Network Monitor maintains a name-to-IP address cache. This cache is directed from Splunk queries or Tenable Nessus Network Monitor's passively-collected data. In some cases, the log forwarder may have a misconfigured name, and therefore the Splunk **IP to Name** mapping may be inaccurate. To correct this issue, review the **serverName** setting in the **$SPLUNK_HOME/etc/system/local/server.conf**. This name must match the name returned in the query `index=_internal sourcetype=splunkd group=tcpin_connections | stats latest(sourceIp) by hostname`. If the names do not match, the Splunk **IP to Name** mapping will be incorrect and the SIEM Pull Service will not provide data to Tenable Nessus Network Monitor.

Example:

```
cat /opt/splunkforwarder/etc/system/local/server.conf
[general]
serverName = dhcpc7
```

Once installed, the Universal Log Forwarder sends the logs to Splunk that Tenable Nessus Network Monitor needs to query and list events.

Tenable Nessus Network Monitor uses the following Splunk query to generate events (using DHCP as an example):

```
source="/var/log/messages" *dhcpd*dhcp*
```

This query generates an output of all DHCP events. For example:

```
[DHCP_HOST] dhcpd: DHCPACK on 127.0.0.1 to 00:11:aa:bb:22:ff (QUERYING_HOST) via ens192
```

For more information on the Splunk Universal Log Forwarder, see the following topics:

- DHCP Setup and Configuration
- SIEM Pull Service Queries

SIEM Pull Service Queries

The SIEM Pull Service is a daemon that connects to Splunk and queries for specific risk-related event types, or risk altering events. The SIEM Pull Service is configured for four types of risk-altering event: *Asset Discovery*, *Service Modification*, *Software Detection*, and *User Account Activity*. These events are most likely to alter the risk profile of an asset, and therefore, Tenable Security Center or Tenable Vulnerability Management should re-scan the affected asset. The risk-altering event types, initial pull service queries to Splunk, and respective plugins are listed below.

Asset Discovery

The SIEM Pull Services uses DHCP logs to detect when assets connect to the network and provide an IP address. The discovered assets can then be used to target in scanning to collect the vulnerability data and establish a risk profile for the asset.

- DHCP Address Assignments

Operating System	Plugins
Linux	• `Linux Assets Discovery Linux DHCP Lease (via Splunk) [710023]` • `Linux Assets Discovery Linux DHCP Expire (via Splunk) [710027]` • `Linux Assets Discovery Linux DHCP Renew (via Splunk) [710024]`
Windows	• `Windows Assets Discovery Windows DHCP Lease (via Splunk) [710025]` • `Windows Assets Discovery Windows DHCP Expire (via Splunk) [710028]` • `Windows Assets Discovery Windows DHCP Renew (via Splunk) [710049]`

Core query:

```
(sourcetype=*dhcp* OR *dhcpd*DHCP* OR source="*dhcp*") AND NOT ("*DNS Update*" OR
"*DNS record*")
```

Service Modification

When services are changed, (added, removed, stopped or started) the risks of an asset are impacted. The assets should be scanned immediately to determine the impact to the risk profile.

- Service Start

Operating System	Plugins
Linux	• Linux Service Modification service start (via Splunk): audit (SERVICE_START) [710020] • Linux Service Modification service start (via Splunk): dbus [710038] • Linux Service Modification service start (via Splunk): systemd (Starting) [710043] • Linux Service Modification service start (via Splunk): systemd (executable .service file) [710044] Core query: `(type=DAEMON_START" OR type="SERVICE_START" OR "systemd: Start*" OR "Successfully activated service" OR "service is marked executable")"`
Windows	• Windows Service Modification service start (via Splunk): code 7036 [710009] • Windows Service Modification service start (via Splunk): code 902 [710036] Core query: `source=WinEventLog:*" AND (Message="*service*running state*" OR Message="*service*start*")"`

- Service Stop

Operating System	Plugins

Linux	• Linux Service Modification service stop (via Splunk) [710021] Core query: `(type=DAEMON_END OR type="SERVICE_STOP" OR "systemd: Stop*" OR "normal halt")`
Windows	• Windows Service Modification service stop (via Splunk): code 7036 [710010] • Windows Service Modification service stop (via Splunk): code 7042 [710042] • Windows Service Modification service stop (via Splunk): code 903 [710048] Core query: `source=WinEventLog:* AND (Message="*service*stopped state*" OR Message="*service*stop*")`

Software Detection

Software Detection events are the result of software installations or removals using common tools such as MSI files, YUM, and DPKG. When software is added to a system, the risk is altered and the system should be scanned using credentials to properly assess the change in risk.

> **Note:** If a binary was manually copied to the system, the event will not be captured.

- Application Removal

Operating System	Plugins
Linux	• Linux Software Detection Application removal (via Splunk): non-yum [710018] • Linux Software Detection Application removal (via Splunk): yum [710035] Core query: `(sourcetype=dpkg* OR sourcetype="syslog*" OR sourcetype="yum*") AND (remove OR "yum* *rase*") AND NOT (systemd OR startup)`
Windows	• Windows Software Detection Application removal (via Splunk): code 1001 [710007] • Windows Software Detection Application removal (via Splunk): code 1034 [710046] Core query: `((sourcetype=WinEventLog:System OR sourcetype="WinEventLog:Application") AND remove*)`

- Application Install

Operating System	Plugins
Linux	• Linux Software Detection application install (via Splunk) [710017] Core query:

	`(type=SOFTWARE_UPDATE OR sourcetype="dpkg*" OR sourcetype="syslog*" OR sourcetype="yum*") AND (Install* OR instal* OR rpm)`
Windows	• `Windows Software Detection application install (via Splunk): code 1033 [710006]` • `Windows Software Detection application install (via Splunk): code 11707 [710041]` • `Windows Software Detection application install (via Splunk): code 7045 [710047]` Core query: `(sourcetype=WinEventLog:System OR sourcetype="WinEventLog:Application") AND install*)`

- Application Update

Operating System	Plugins
Linux	• `Linux Software Detection application update (via Splunk) [710040]` Core query: `(type=SOFTWARE_UPDATE OR sourcetype="dpkg*" OR sourcetype="syslog*" OR sourcetype="yum*") AND (Install* OR instal* OR rpm)`

User Account Activity

User Account Activity events are related to system's user accounts. Each time an account is modified, the impact of that change is worth noting. Many compliance reports require the tracking of password changes, group memberships, and similar activities.

- Add User

Operating System	Plugins
Linux	• Linux User Account Activity Add user (via Splunk): useradd, plain [710012] • Linux User Account Activity Add user (via Splunk): audit (ADD_USER) [710037] • Linux User Account Activity Add user (via Splunk): audit (USER_MGMT) [710045] Core query: `(sourcetype=linux_audit OR sourcetype=linux_secure) AND (new* OR ADD) AND (user OR USER)`
Windows	• Windows User Account Activity Add user (via Splunk) [710001] Core query: `(sourcetype=WinEventLog:Security Message="*user*created*")`

- Add User to Group

Operating System	Plugins
Linux	• Linux User Account Activity Add User to a Group (via Splunk): audit (USER_MGMT) [710013] • Linux User Account Activity Add User to a Group (via Splunk): audit (USER_CHAUTHTOK) [710029] Core query:

	`(sourcetype=linux_audit OR sourcetype=linux_secure) AND (op=adding user to group OR add-user-to-group)`
Windows	• Windows User Account Activity Add User to a Group (via Splunk) [710002] Core query: `(sourcetype=WinEventLog:Security Message="A member was added*group*")`

- Modify Password

Operating System	Plugins
Linux	• Linux User Account Activity Modify Password (via Splunk): usermod, plain [710015] • Linux User Account Activity Modify Password (via Splunk): audit (changing password) [710031] • Linux User Account Activity Modify Password (via Splunk): audit (updating password) [710032] Core query: `(sourcetype=linux_audit OR sourcetype=linux_secure OR type=USER_CHAUTHTOK) AND (updat* OR chang*) AND (password)`
Windows	• Windows User Account Activity Modify Password (via Splunk) [710004] Core query: `(sourcetype=WinEventLog:Security AND "Message=A user account was changed") | regex "`

	`(Password\sLast\sSet:\s+\d+\/\d+\/\d+\s\d+\:\d+\:\d+\s)(AM|PM)"`

- Remove User

Operating System	Plugins
Linux	• `Linux User Account Activity remove user (via Splunk): userdel, plain [710016]` • `Linux User Account Activity remove user (via Splunk): audit (DEL_USER, plain) [710033]` • `Linux User Account Activity remove user (via Splunk): audit (DEL_USER, not found) [710034]` • `Linux User Account Activity remove user (via Splunk): audit (DEL_USER, deleting user entries) [710039]` Core query: `(sourcetype=linux_audit OR sourcetype=linux_secure) AND (op=delete-user OR delet* user)`
Windows	• `Windows User Account Activity remove user (via Splunk) [710005]` Core query: `(sourcetype=WinEventLog:Security AND Message="A user account was deleted*")`

- Remove User from Group

Operating System	Plugins
Linux	• `Linux User Account Activity Remove User from a group`

	(via Splunk): audit (USER_MGMT) [710014] • Linux User Account Activity Remove User from a group (via Splunk): audit (USER_ACCT) [710030] Core query: `(sourcetype=linux_audit OR sourcetype=linux_secure) AND (op=user * removed by * from group OR op=delete-user-from-group)`
Windows	• Windows User Account Activity Remove User from a Group (via Splunk) [710003] Core query: `(sourcetype=WinEventLog:Security Message="A member was removed*group*")`

VMWare ERSPAN

To monitor virtual machines in a VMware vSphere environment, VMware vSphere Distributed Switch (VDS) supports industry standard features such as port mirroring and NetFlow. These features were introduced with the release of vSphere 5.0. You can use ERSPAN to mirror traffic from one or more source ports on a virtual switch, physical switch, or router and send the traffic to a destination IP host running NNM. The following ERSPAN virtual environments are supported for NNM:

- VMware ERSPAN (Transparent Ethernet Bridging)
- Cisco ERSPAN (ERSPAN Type II)

Note: Tenable Nessus Network Monitor does not support ERSPAN Type III. As a workaround, you can create a new port mirroring session using GRE tunneling.

For VSphere 5.1:

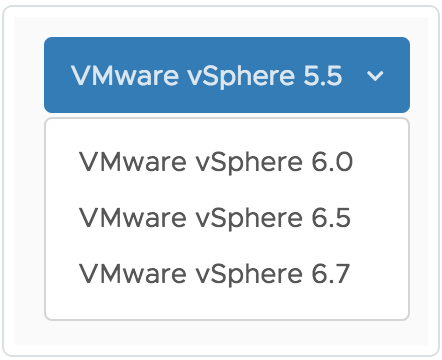
To monitor virtual machines with NNM residing on the same ESX host as the virtual machines, see the following link:
https://blogs.vmware.com/vsphere/2013/01/vsphere-5-1-vds-feature-enhancements-port-mirroring-part-1.html

To monitor virtual machines with NNM residing on external hardware, see the following link:
https://blogs.vmware.com/vsphere/2013/02/vsphere-5-1-vds-feature-enhancements-port-mirroring-part-2.html

To monitor virtual machines with NNM residing on an ESX host other than where the virtual machines reside, see the following link:
https://blogs.vmware.com/vsphere/2013/02/vsphere-5-1-vds-feature-enhancements-port-mirroring-part-3.html

For VSphere versions above 5.1:

1. In your browser, navigate to https://docs.vmware.com/en/VMware-vSphere/5.5/-com.vmware.vsphere.networking.doc/GUID-68B5DD45-DD3F-4E9B-A6CD-BE97026A846A.html.
2. In the top right corner of the screen, select the version of VSphere for which you wish to view configuration instructions.



The instructions include information regarding all 3 relevant configurations for Tenable Nessus Network Monitor.

Virtual Switches for Use with NNM

The Tenable NNM monitors network traffic at the packet layer to determine topology and identify services, security vulnerabilities, suspicious network relationships, and compliance violations.

NNM provides visibility into both server and client-side vulnerabilities, discovers the use of common protocols and services (e.g., HTTP, SQL, file sharing), and performs full asset discovery for both IPv4 and IPv6, and even on hybrid networks.

Virtualization of server rooms provides an added challenge to monitoring the network. Communication between VMs within the virtual switch is not monitored by the standard monitoring tools on the physical network since traffic between VMs does not route to the physical switch. NNM provides the ability to passively scan virtual network traffic between VMs that are in the same virtual switch as a deployed NNM VM.

This section provides an overview of the standard methods to configure the virtual switches in various systems to provide NNM with a SPAN or mirror port to gather data from inside the virtual network between VMs. While some platforms provide the ability to send monitored traffic to a remote host, the guidance provided in this document describes an environment where NNM is configured on a VM within the virtual switch cluster. The exact desired options may vary based on local monitoring requirements. The platform use to generate the technical steps in this document was configured with the most recent versions of the software. If you are using older or newer software revisions, some of these steps may vary.

Basic NNM VM Configuration

The first step in the process is to install a NNM VM that is attached to the virtual switch's span port. Tenable Core can be used for this purpose. Tenable Core and its documentation can be downloaded from the Tenable Downloads page and installed as many times as your license allows. During configuration, ensure that the configured networking ports include the monitoring port(s) of the virtual switch. Under the NNM configuration, confirm that the monitored port(s) include the ports configured for mirroring.

Platforms

Tenable Nessus Network Monitor can be configured with the following platforms:

- VMWare ESXi - Desktop Client
- VMWare vSphere - Flash Web UI
- VMware vSphere - HTML Web UI
- Microsoft Hyper-V

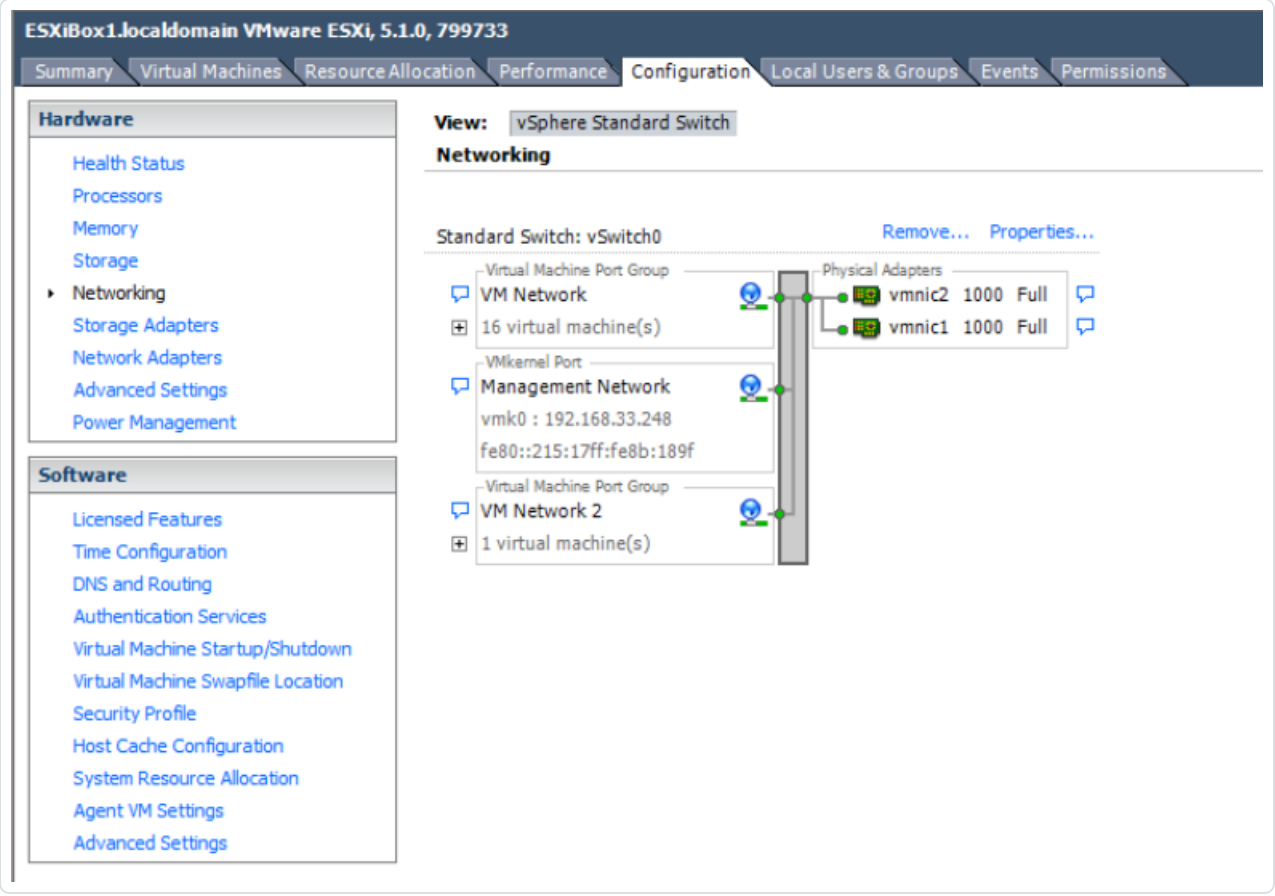
VMWare ESXi - Desktop Client

Configuring the virtual switch provided with VMware ESXi for monitoring uses a port group set for promiscuous mode. Only attach VMs to this port group that will be used to monitor the traffic. Any VM using this port group has the ability to monitor all traffic.

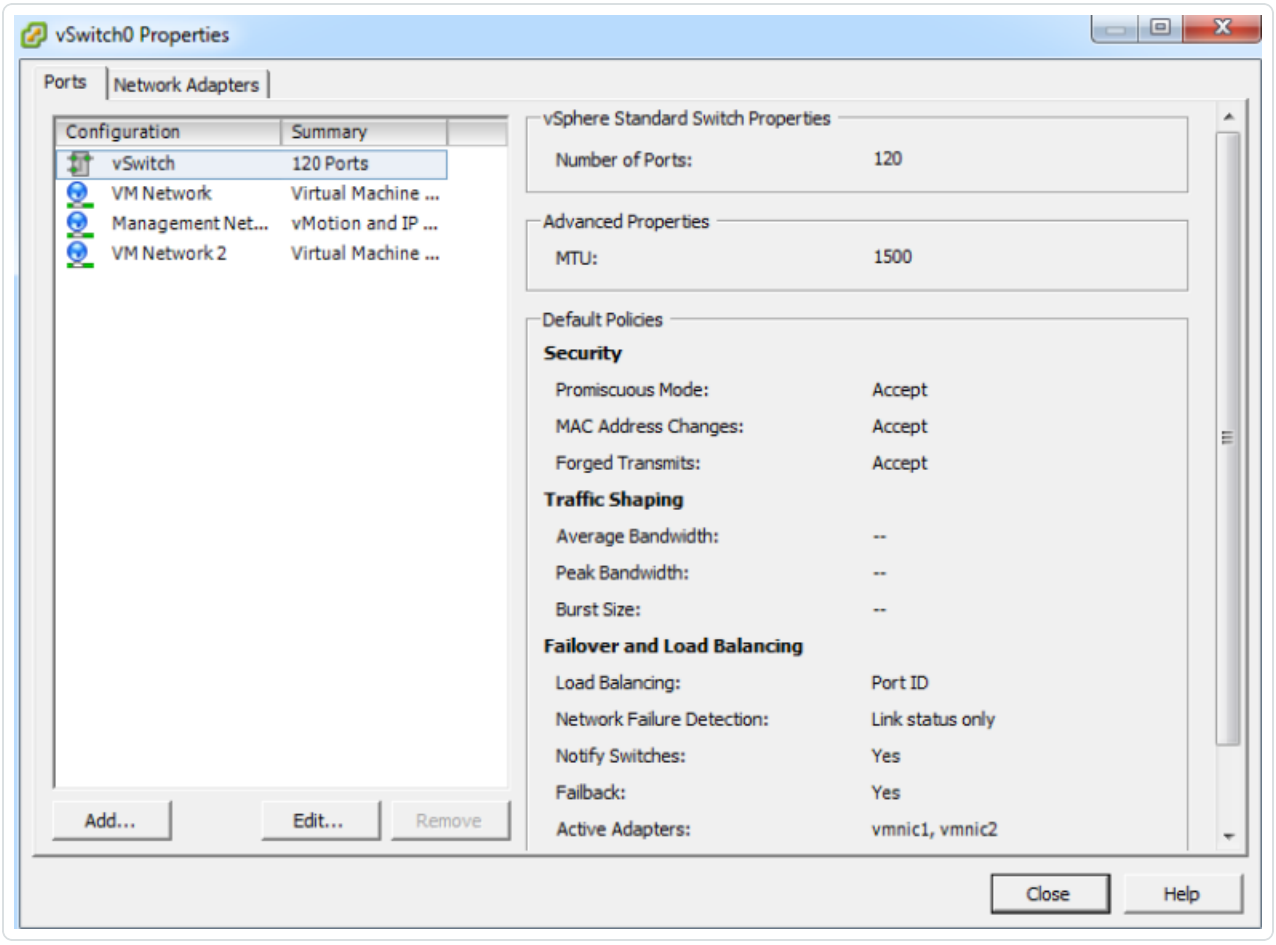
Configure the ESX Management Portal

The following steps are performed on the ESX Management Portal.

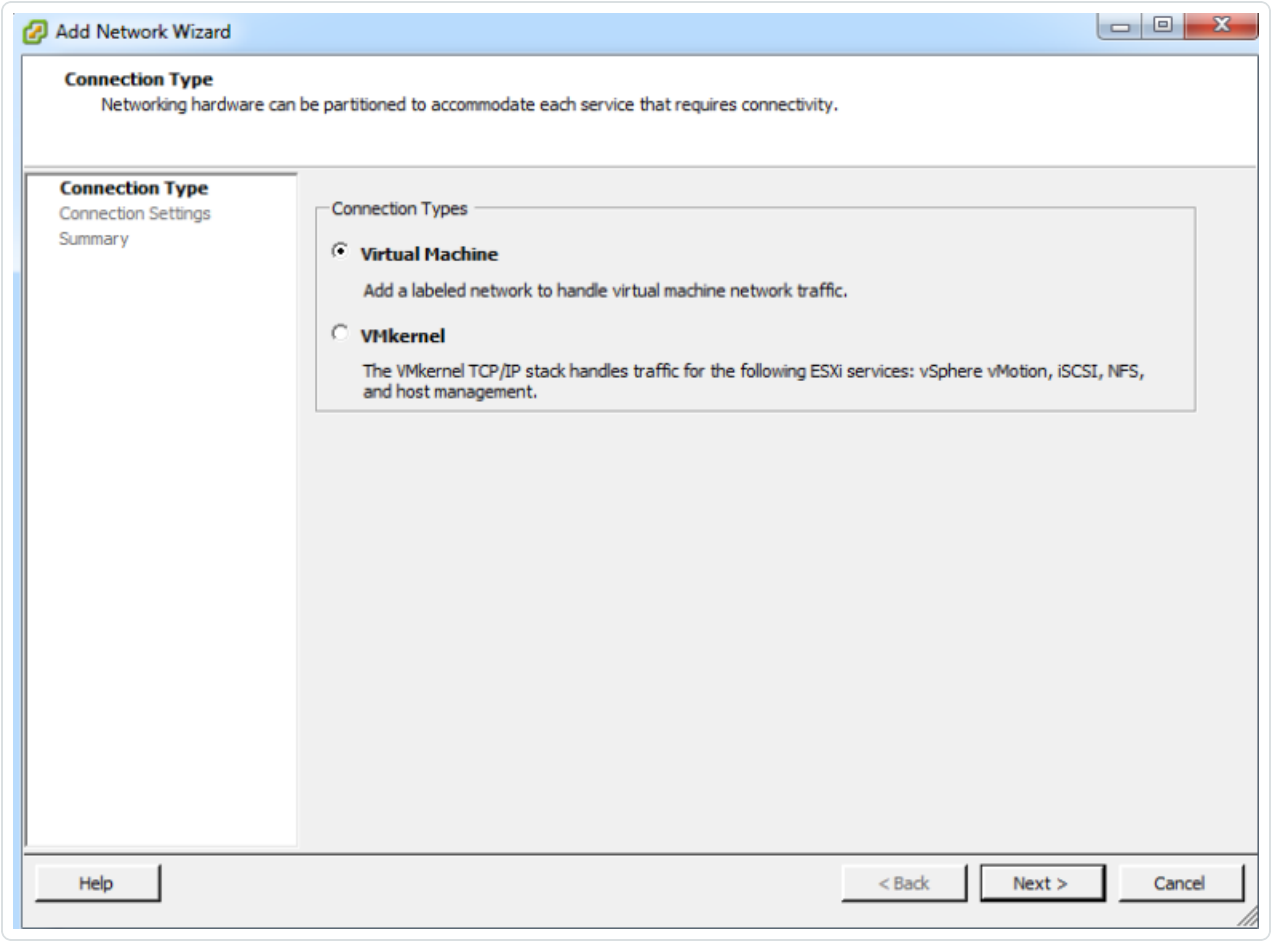
1. Log in to the ESX management portal and navigate to the **Configuration** tab for the ESXi host.
2. From the **Hardware** list, select **Networking**. Click **Properties**.



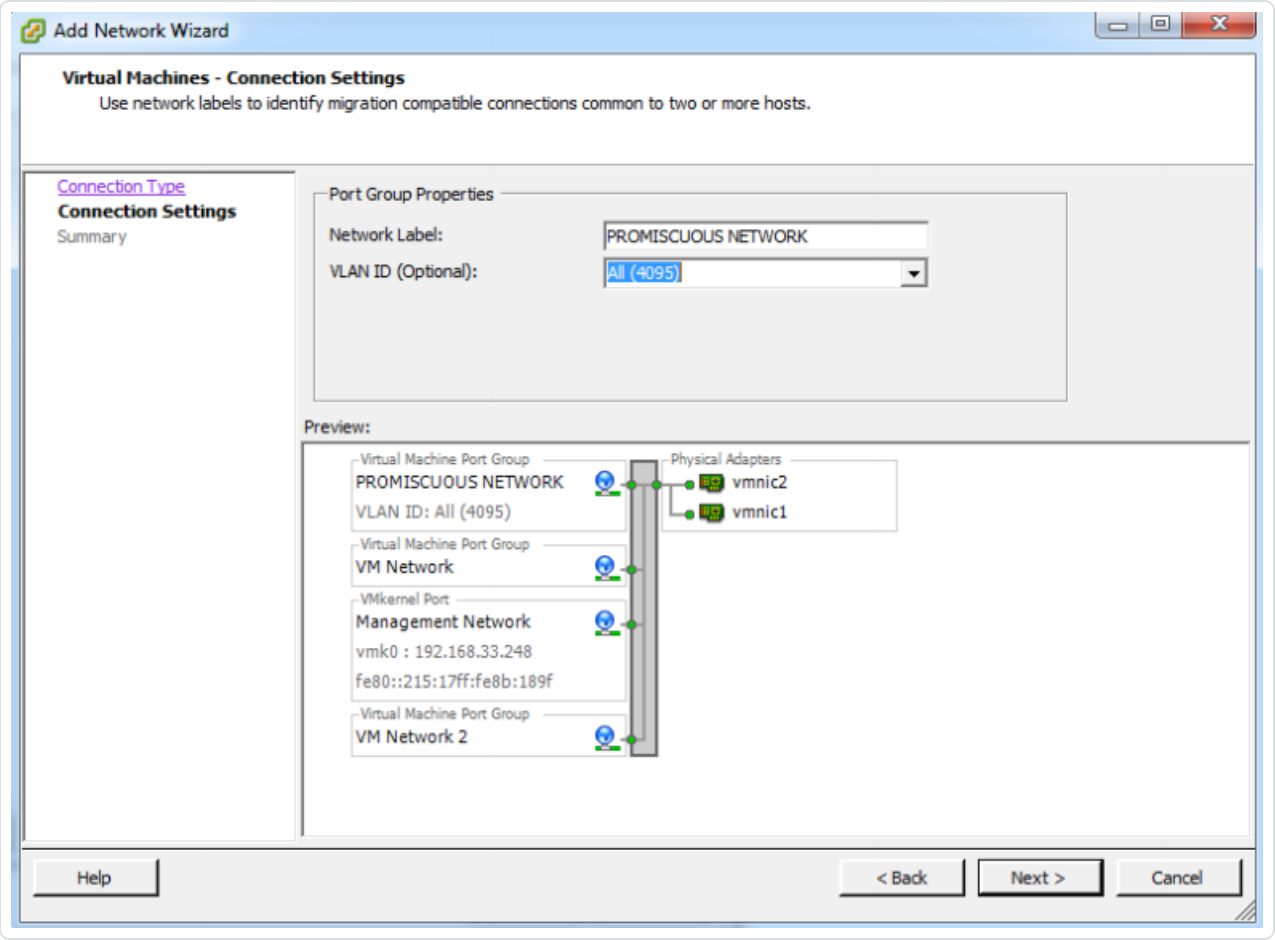
3. Under the **Ports** tab, click **Add** to create a new port group.



4. Select **Virtual Machine**.

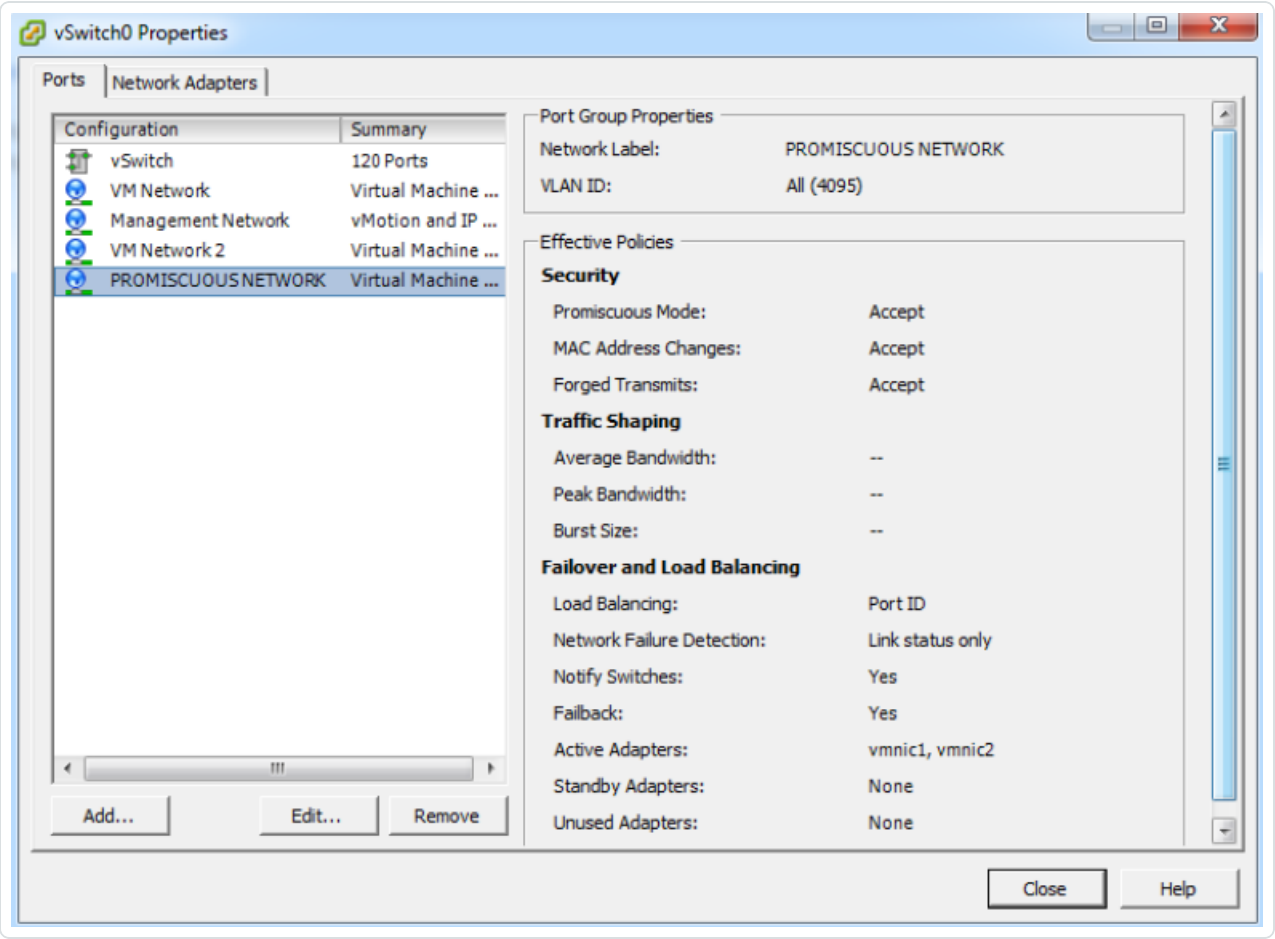


5. Click **Next**.
6. Set a descriptive name for the new port group and a VLAN ID, if desired. Setting the VLAN ID to 4095 utilizes the special VMware VLAN to monitor all other VLANs.



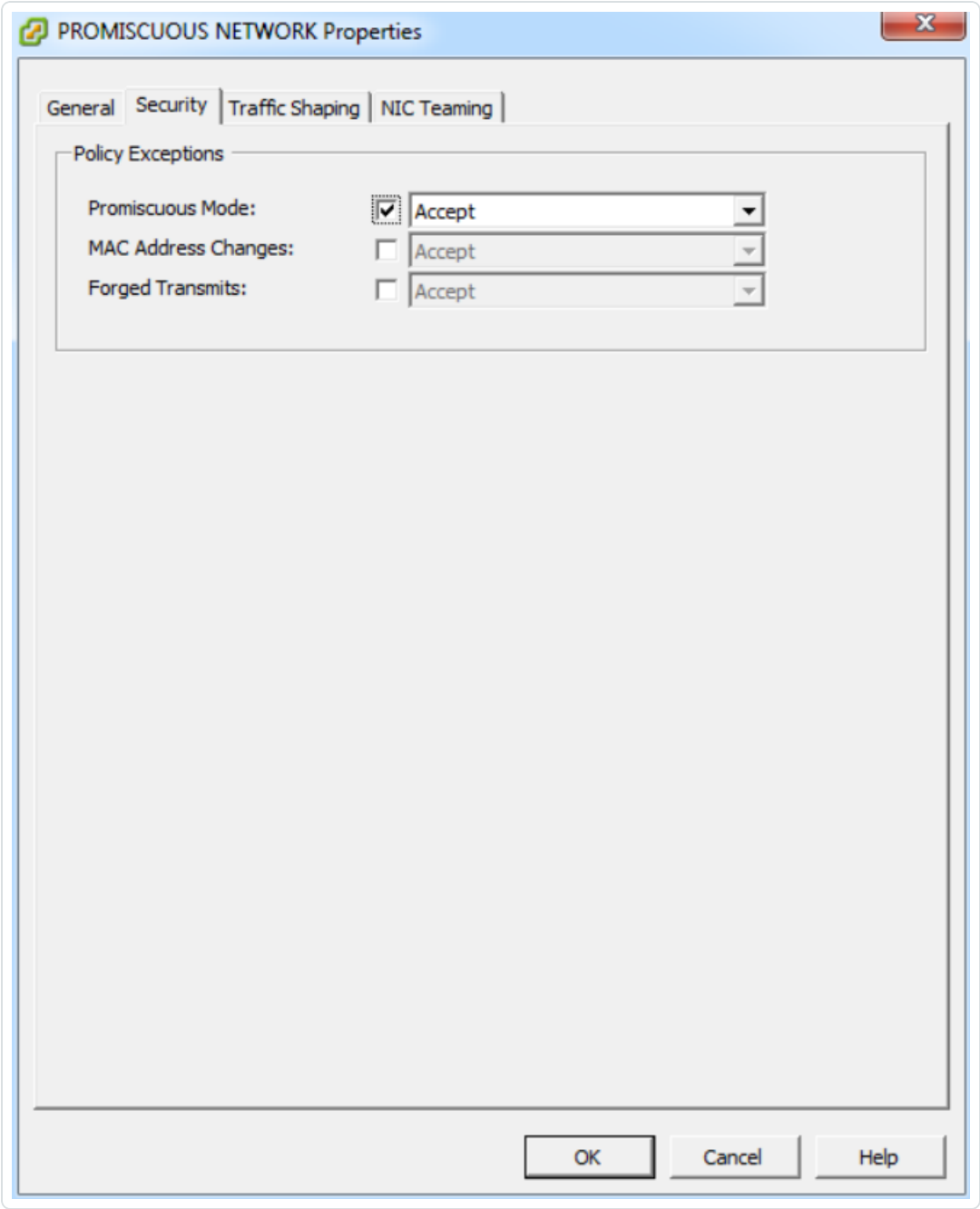
7. Click **Next** and then **Finish**. You return to the **Properties** page.

8. Select your new port group and click **Edit**.



9. On the port group properties page, select the **Security** tab and click on the checkbox next to **Promiscuous Mode**.

10. From the drop-down menu select **Accept**.

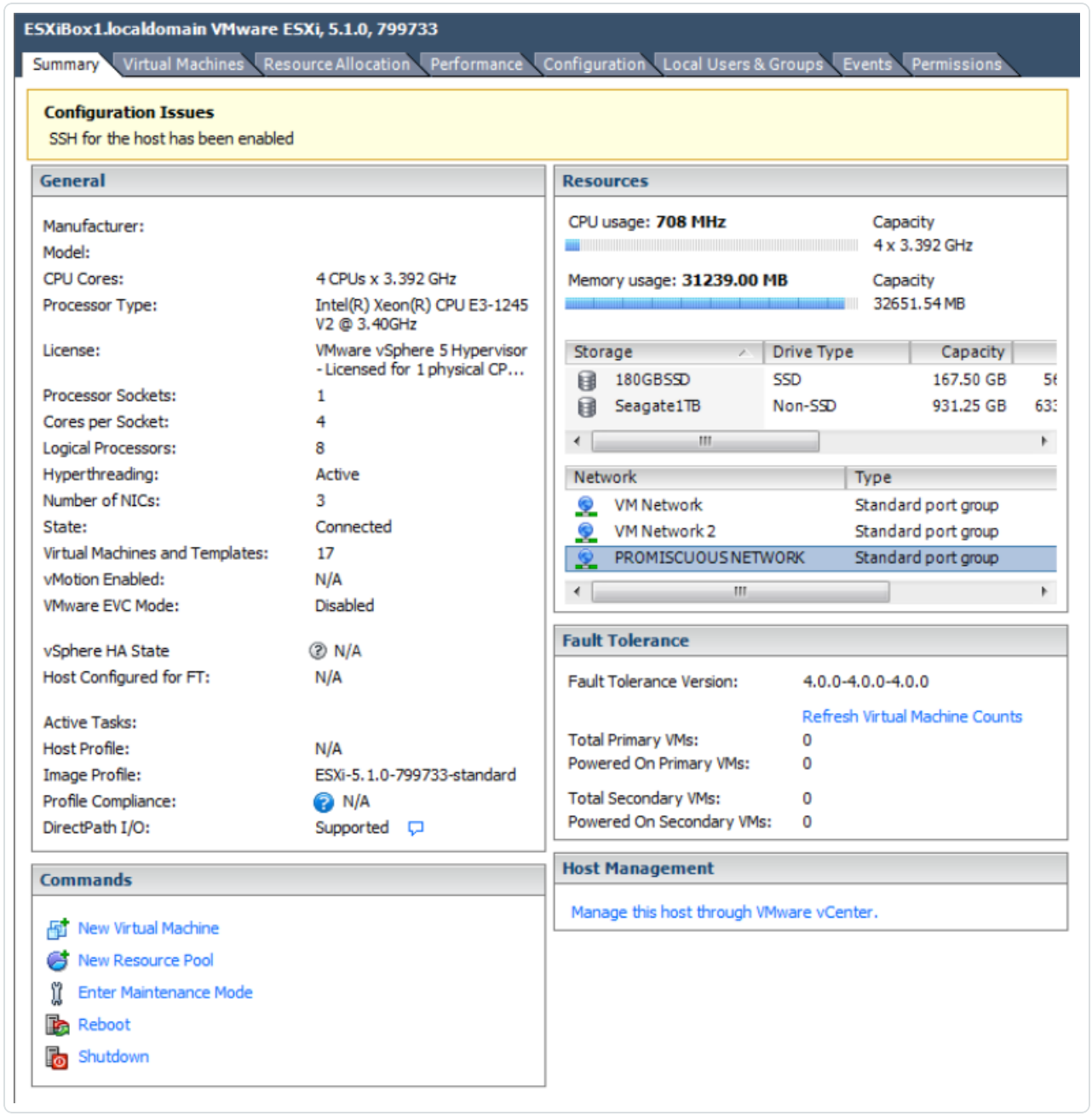


11. Click **OK**.

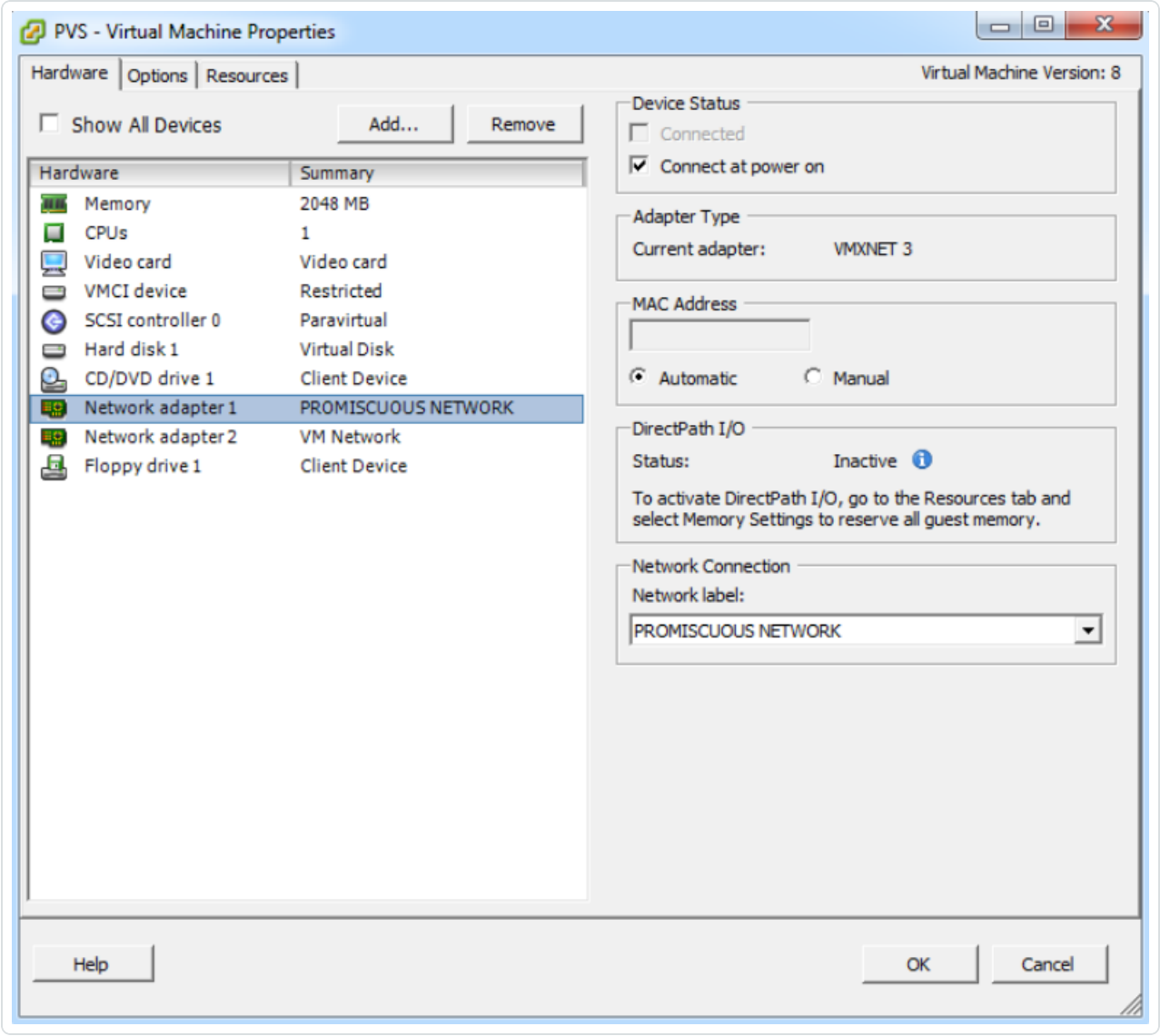
Configure the NNM VM

The following steps are performed on the **Properties** tab of the NNM VM within the VM platform. For further guidance on configuring NNM please refer to the NNM User Guide available on Tenable NNM Docs page.

1. Navigate to the **Properties** tab of the NNM VM within the VM Platform.



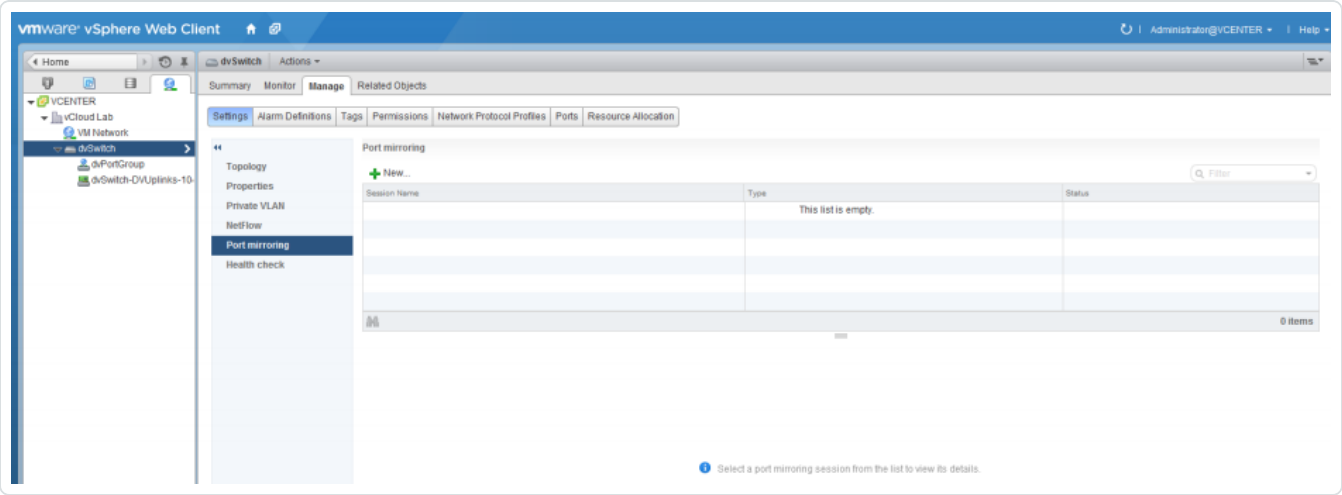
2. In the **Properties** area of the adapter settings, set the network connection's **Network Label** field to the newly created port group.



3. Click **OK**.
4. Start the NNM VM and configure the NNM to use the promiscuous network adapter for monitoring.
5. Start (or restart) the NNM service with the new settings. Network traffic on the virtual switch is now collected by the NNM.

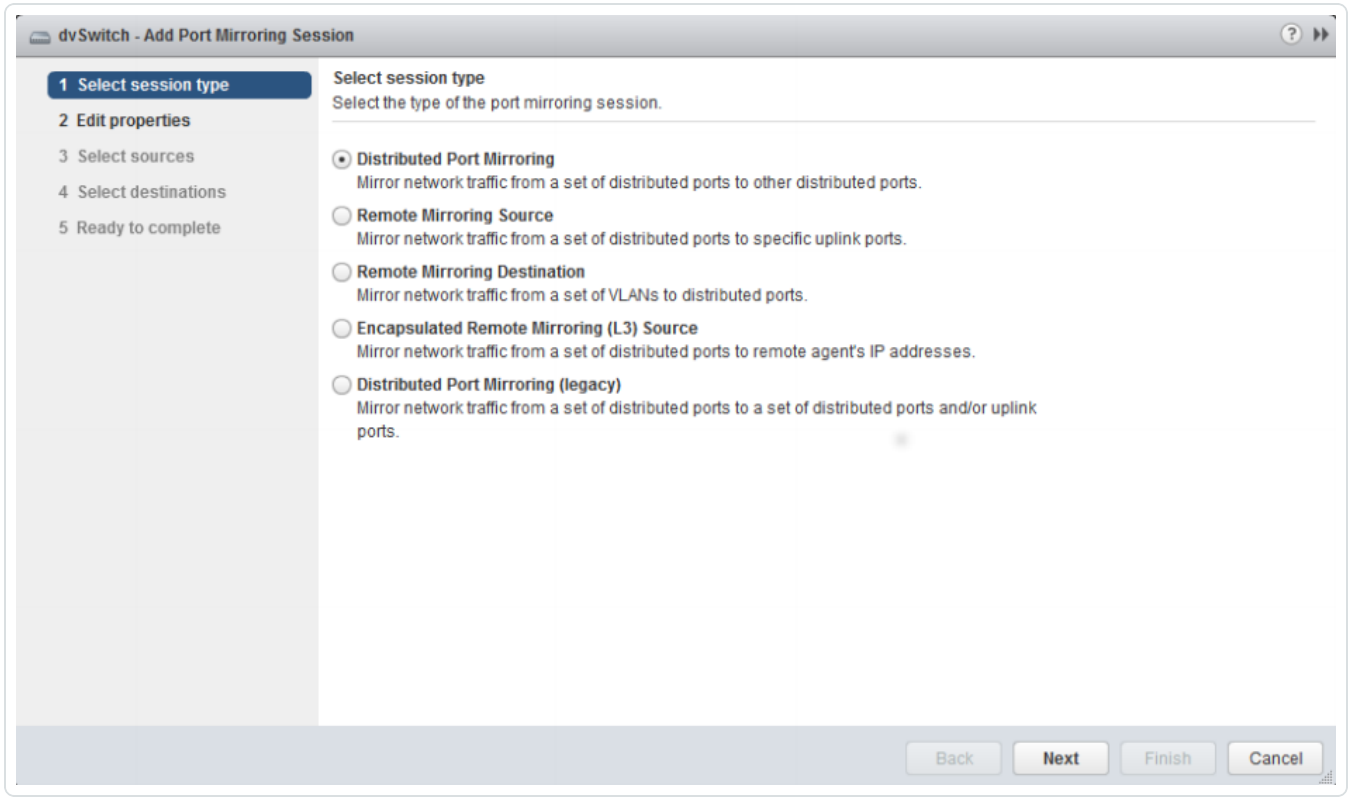
Configure VDS

1. Select the VDS to configure for port mirroring from the list.
2. Navigate to the **Manage** tab and, in the Settings section, select **Port Mirroring**.



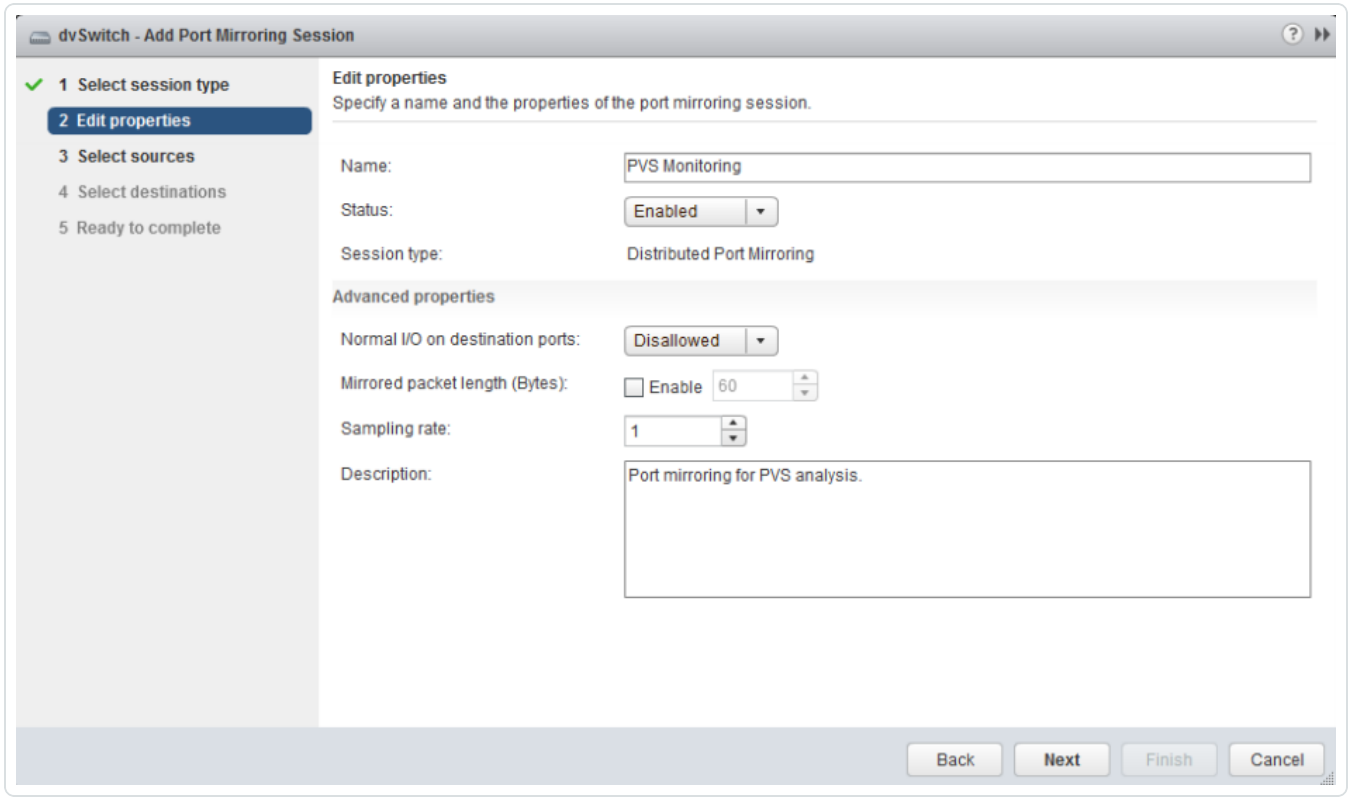
3. Click **New** to begin creating a new port mirror configuration. A wizard appears.

4. Select a session type, such as **Distributed Port Mirroring**.

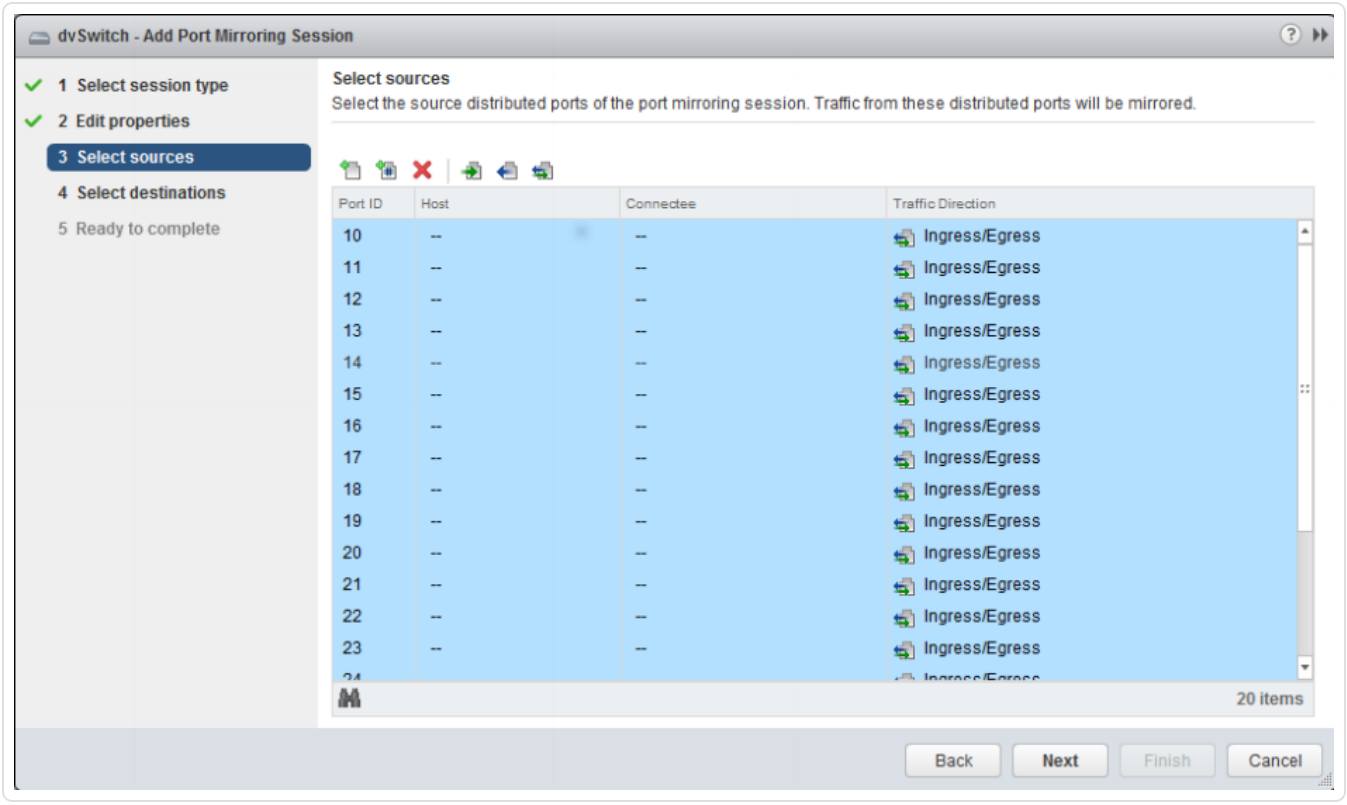


5. In the **Edit Properties** section, in the **Name** field, type a name for the port mirror set.
6. Ensure the **Status** setting is **Enabled**.
7. Because the port is only used to monitor traffic, set the **Normal I/O** on destination ports field to **Disallowed**.
8. Adjust the **Mirrored packet length** setting as needed for the environment.

9. Optionally, enter a **Description** to provide more information about the use of this mirrored port.

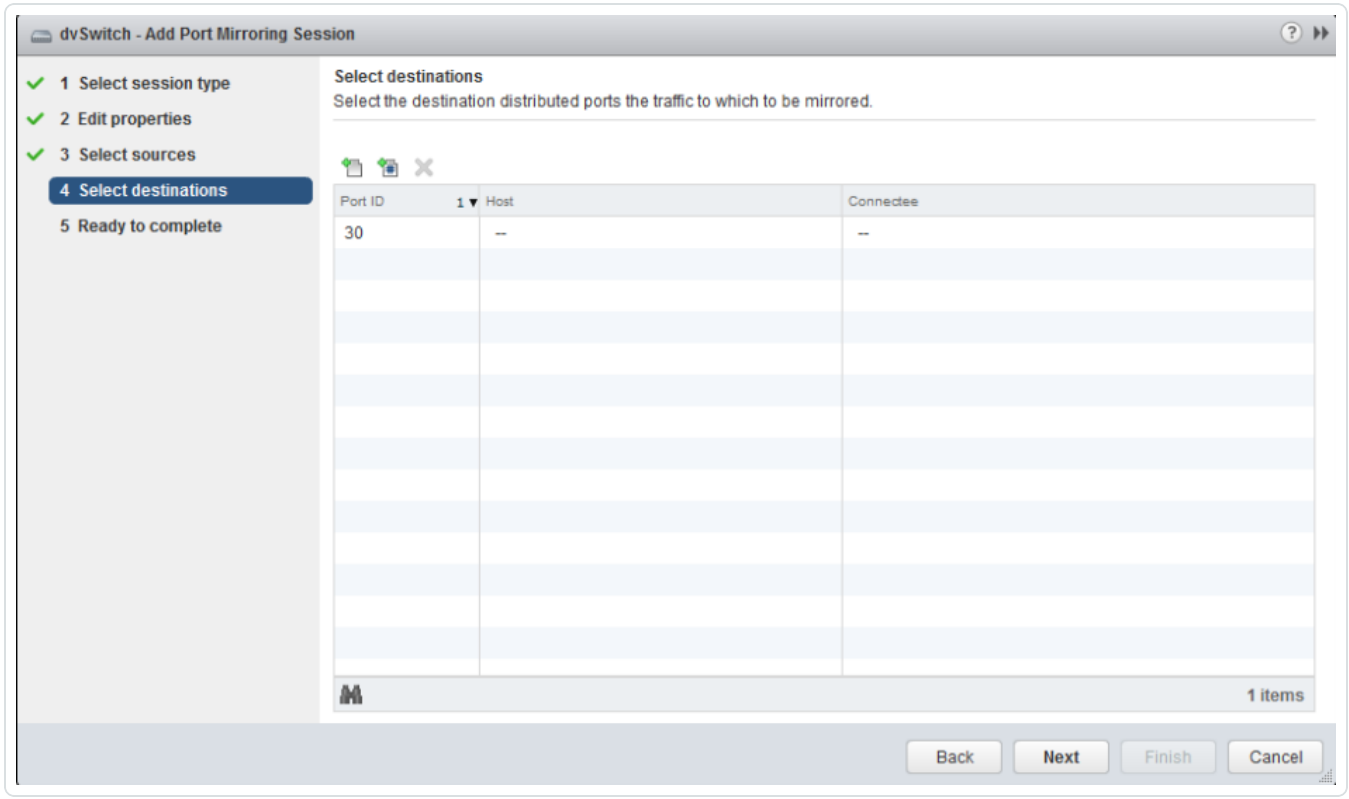


10. Click **Next**.
11. In the **Select Sources** section, select the port(s) to be mirrored for this set.
12. Click **OK**.
13. To determine which direction of traffic to monitor with this mirror, select **Ingress**, **Egress**, or **Ingress/Egress**. Monitoring both directions yields the maximum amount of information.



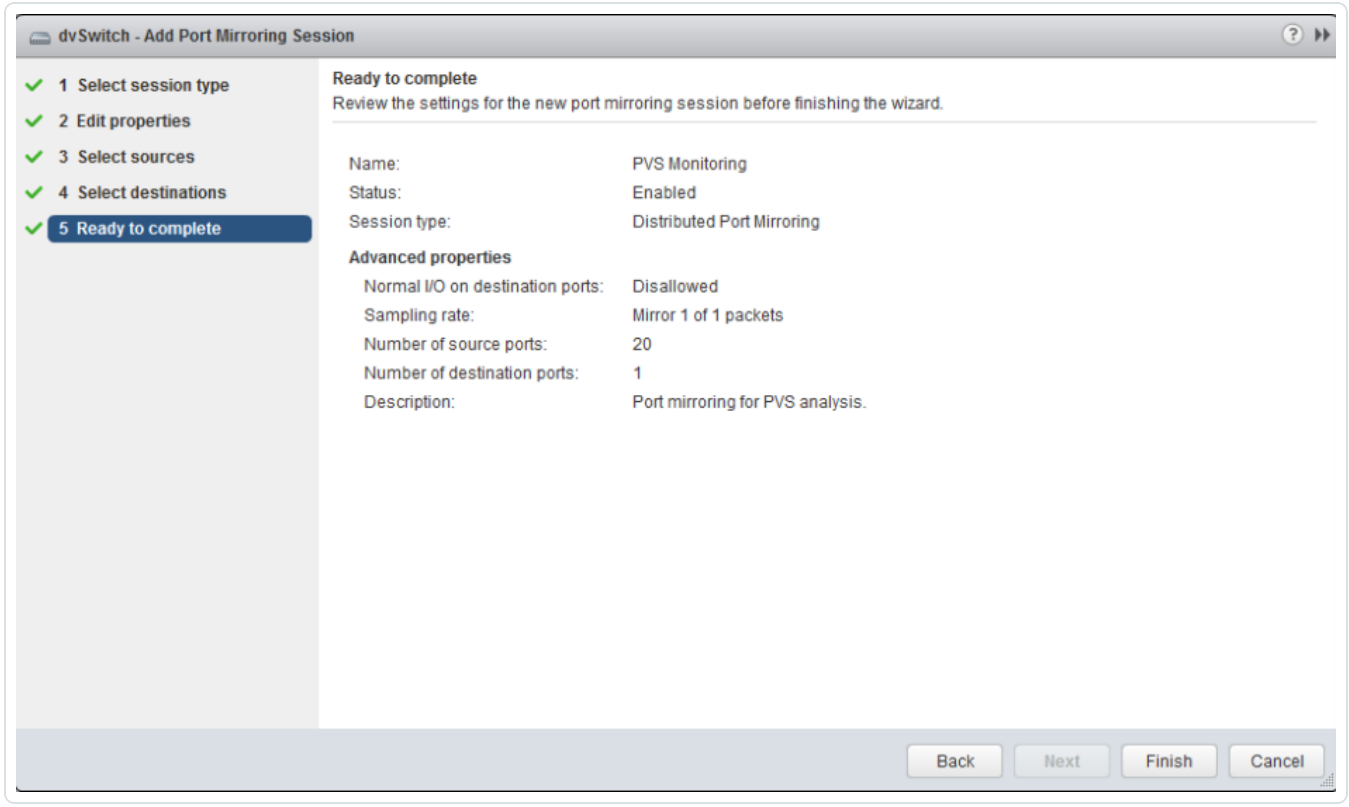
14. Click **Next**.

15. In the **Select Destinations** section, select the destination ports to receive the mirrored traffic.



16. Click **Next**.

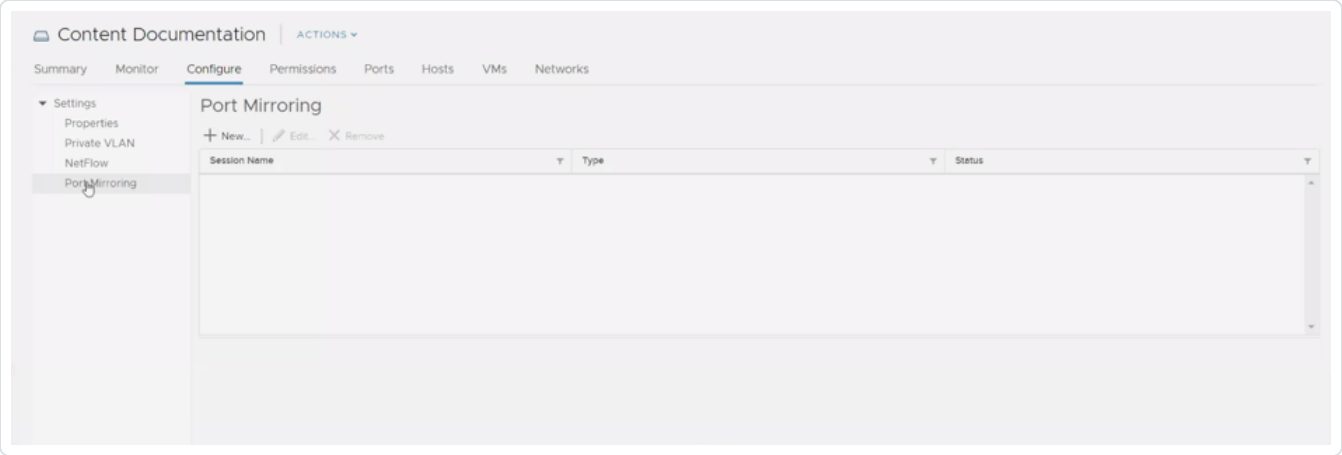
17. In the **Ready to Complete** section, review the information mirror set and click **Finish**. The settings apply and, once the NNM VM is configured correctly, NNM begins collecting data.


dvSwitch - Add Port Mirroring Session
1 Select session type
2 Edit properties
3 Select sources
4 Select destinations
5 Ready to complete
Ready to complete
Review the settings for the new port mirroring session before finishing the wizard.
Name: PVS Monitoring
Status: Enabled
Session type: Distributed Port Mirroring
Advanced properties
Normal I/O on destination ports: Disallowed
Sampling rate: Mirror 1 of 1 packets
Number of source ports: 20
Number of destination ports: 1
Description: Port mirroring for PVS analysis.
Back
Next
Finish
Cancel

VMWare vSphere - HTML5 Web User Interface

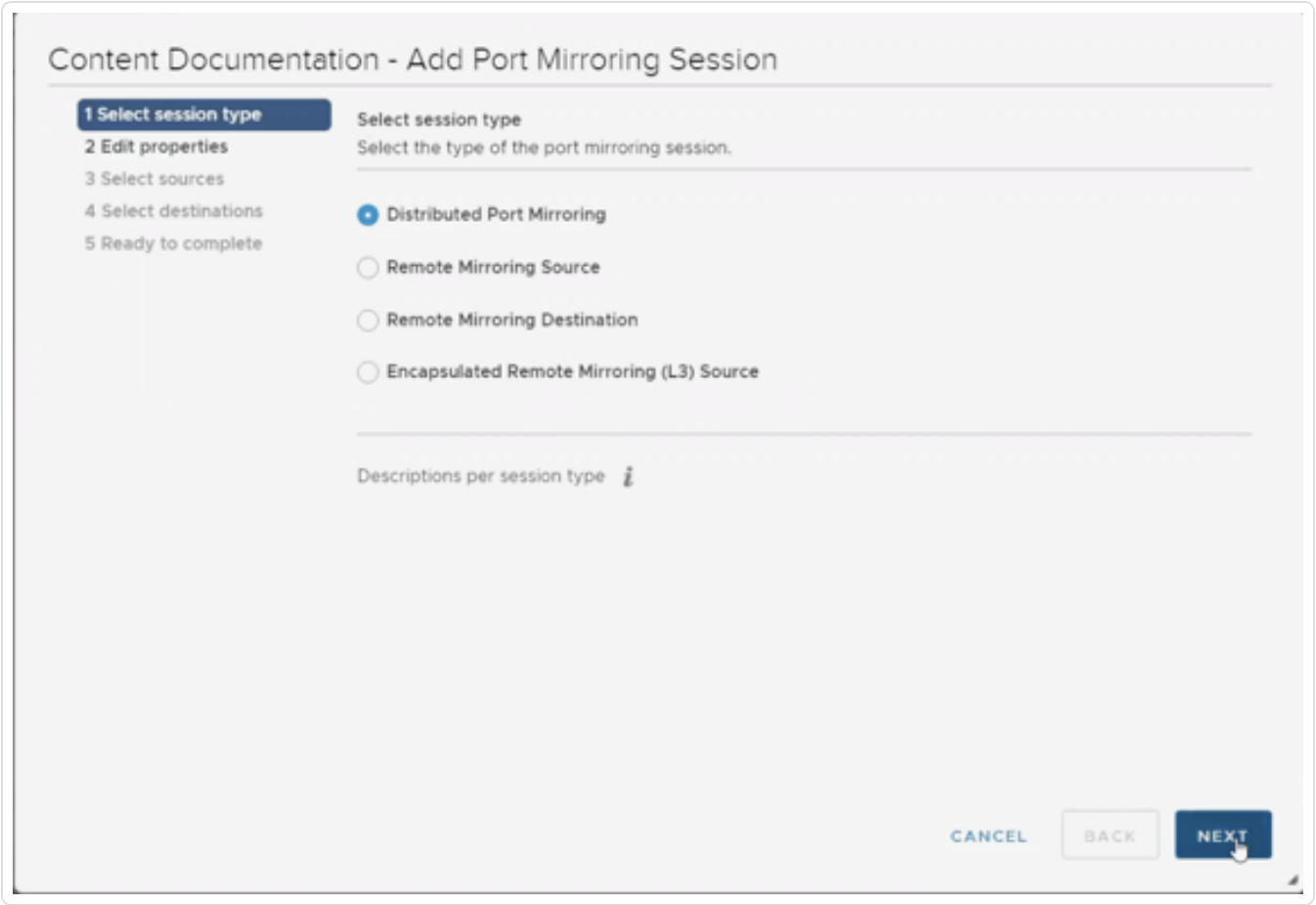
Configure Virtual Distributed Switch Port Mirroring

1. In the left panel, select the VDS for which to configure for port mirroring.
2. Click **Configure**.



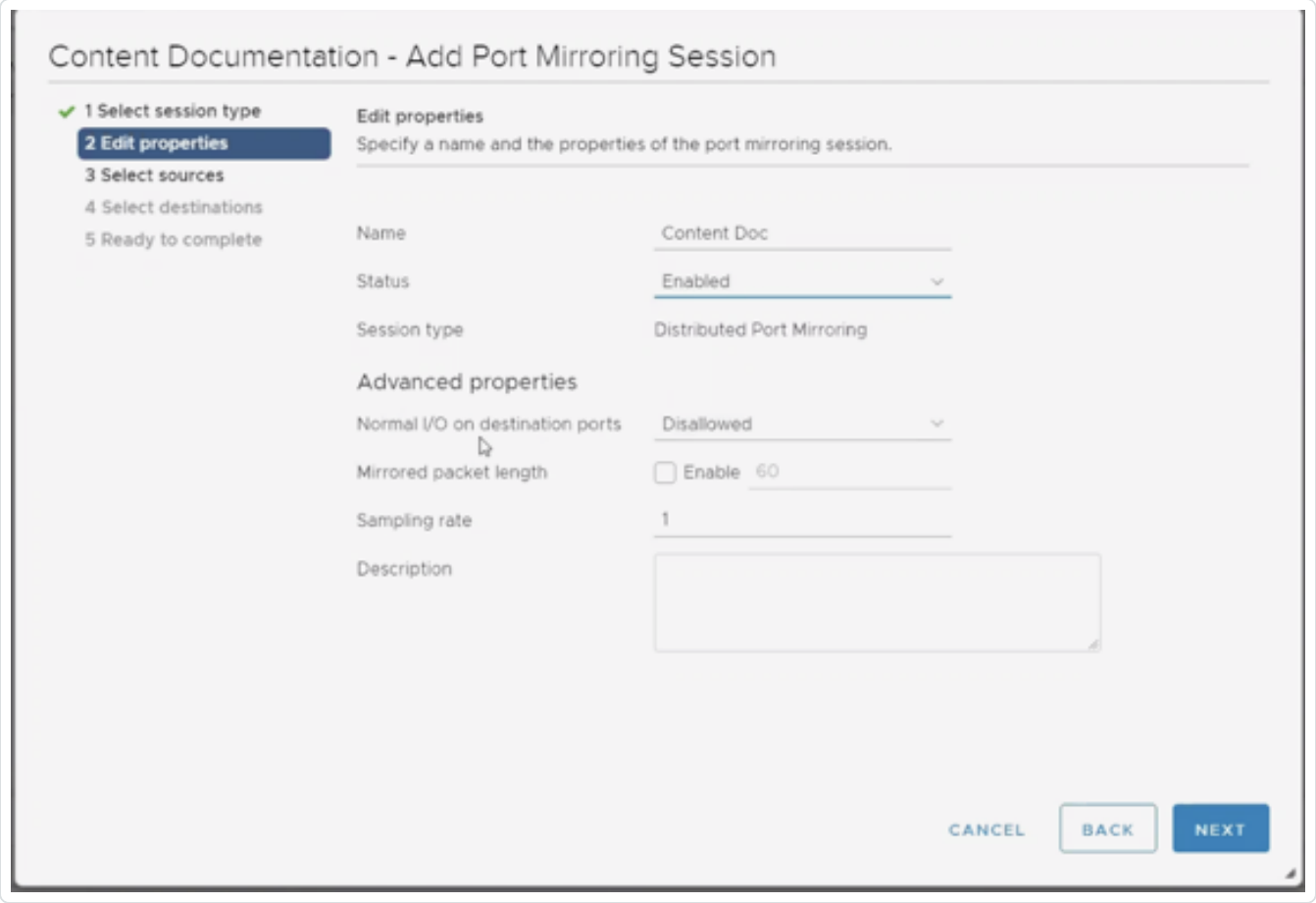
3. Click **Port Mirroring**.
4. Click **New**.

5. In the **Select Session Type** section, select **Distrubuted Port Mirroring**.



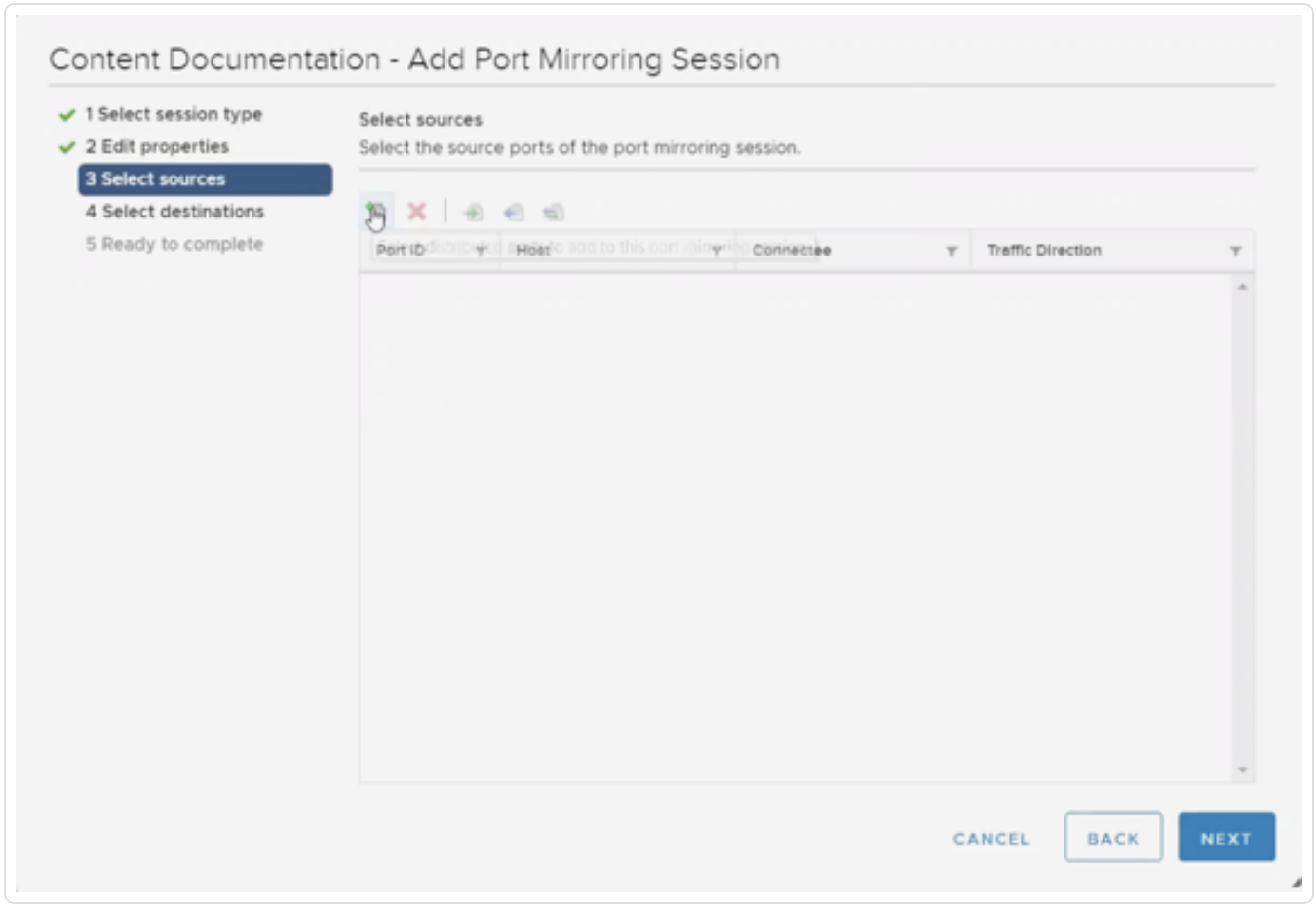
6. Click **Next**.

7. In the **Name** field, type a name for the port mirroring session.

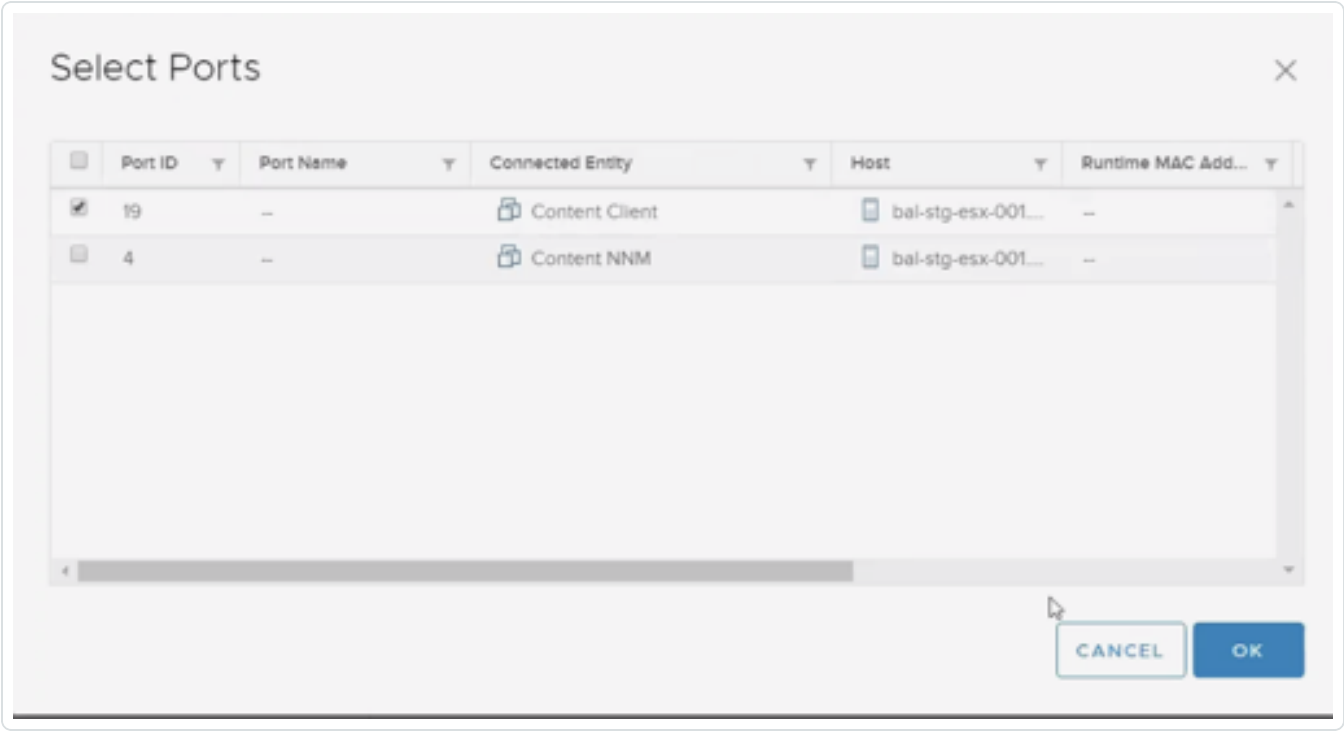


8. Change **Status** to **Enabled**.
9. (Optional) Edit the **Advanced Properties** section.
10. Click **Next**.

11. Click the **Add Source Port** button.



12. Select your Client VM(s) from the list.

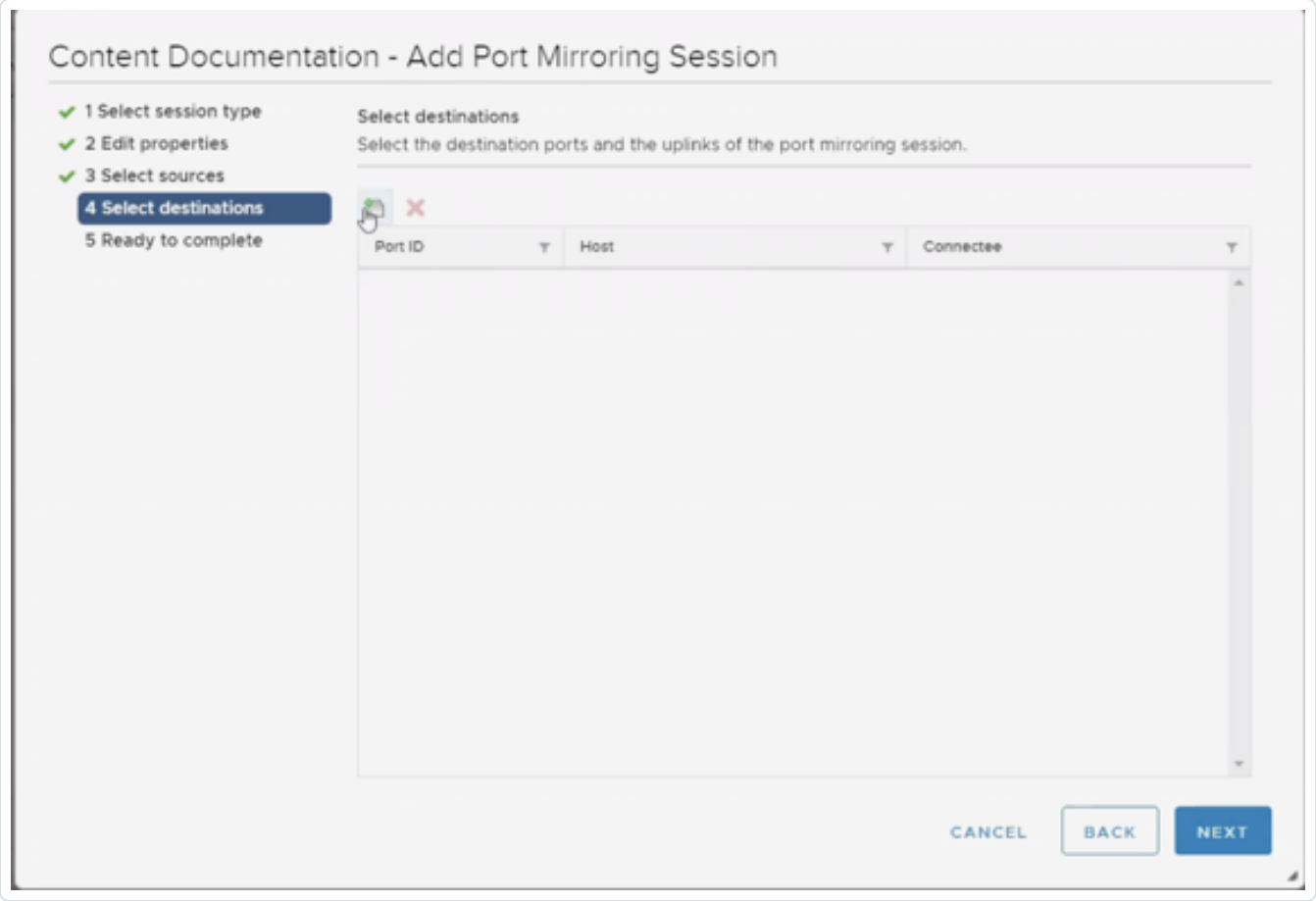


13. Click **OK**.
14. (Optional) Select **Ingress**, **Egress**, or both. By default, both are selected.

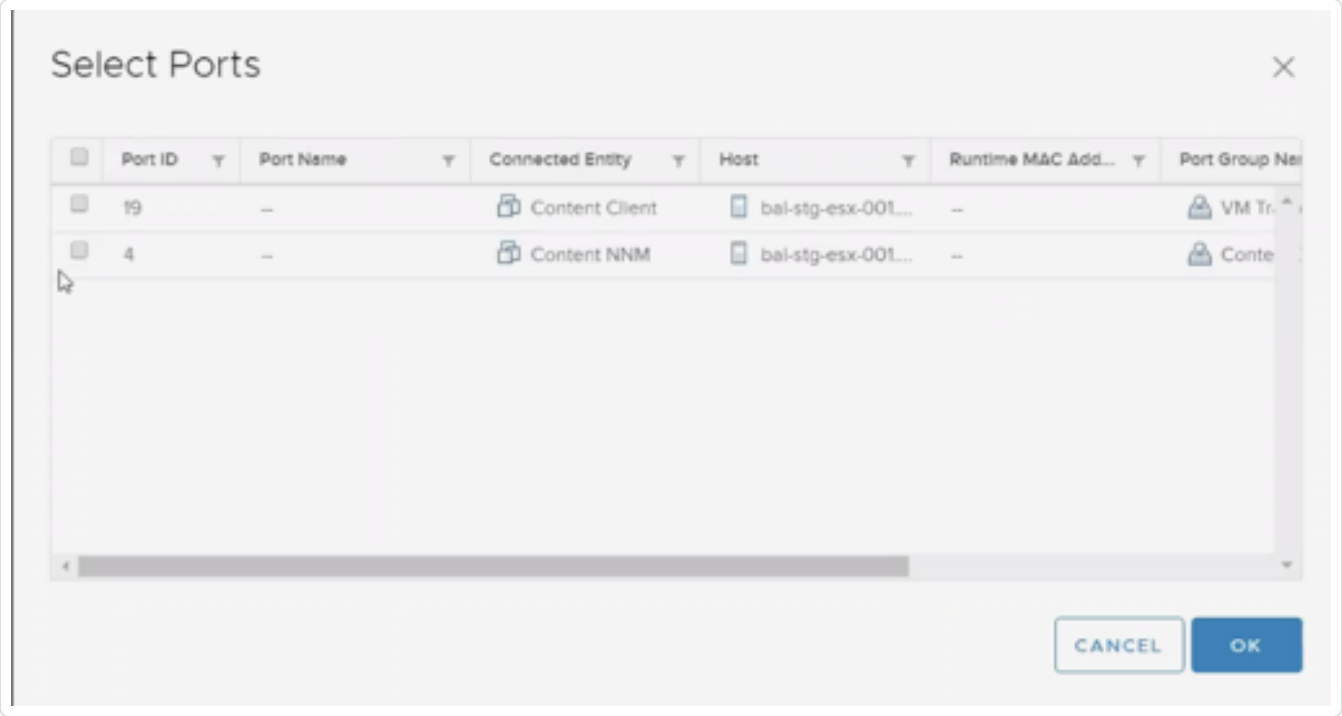
> **Tip: Ingress** and **Egress** determine the directional flow of traffic from your VMs.

15. Click **Next**.

16. Click the **Add Destination Port** button.

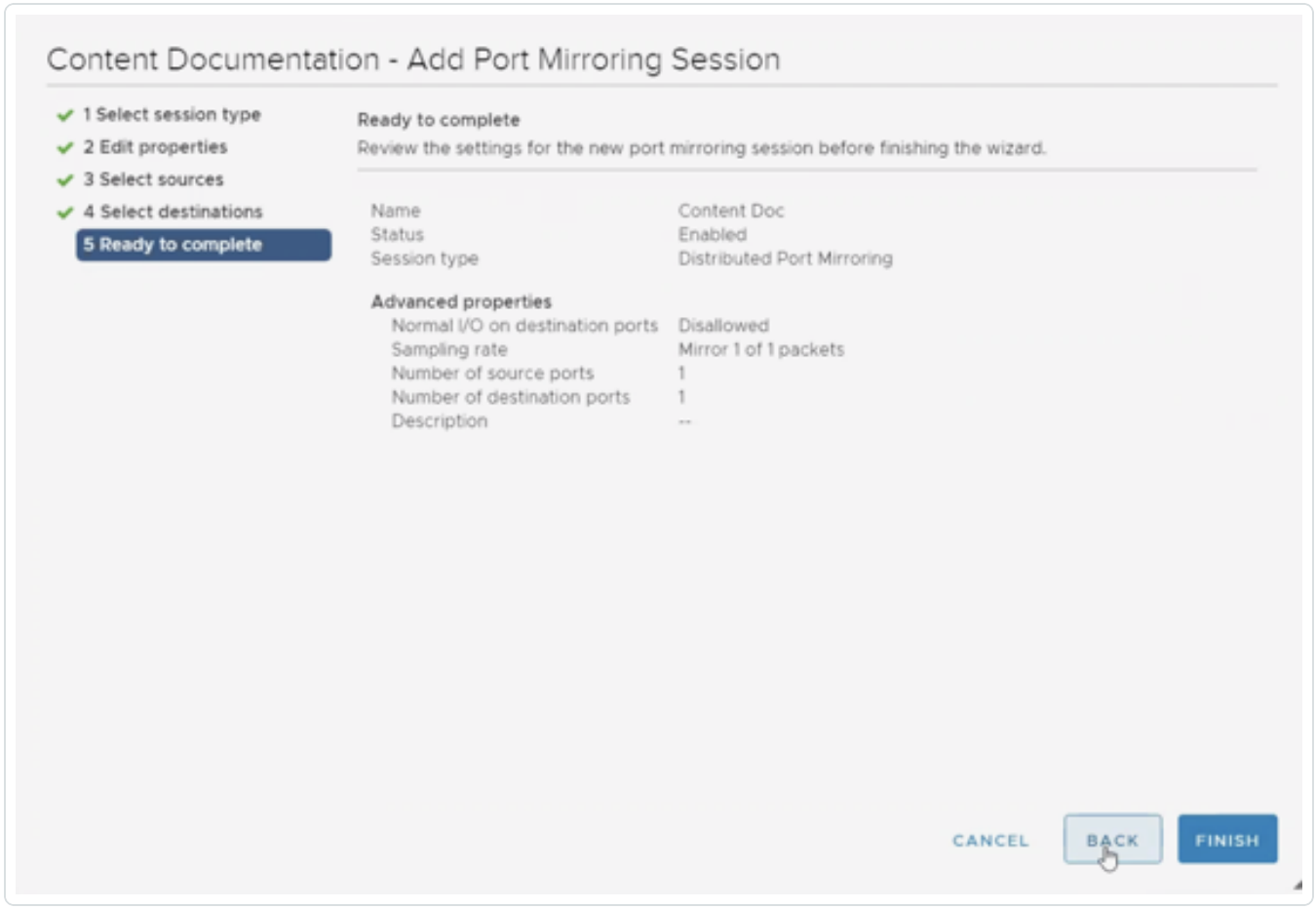


17. Select your NNM VM(s) from the list.



18. Click **OK**.
19. Click **Next**.

20. Click **Finish**.

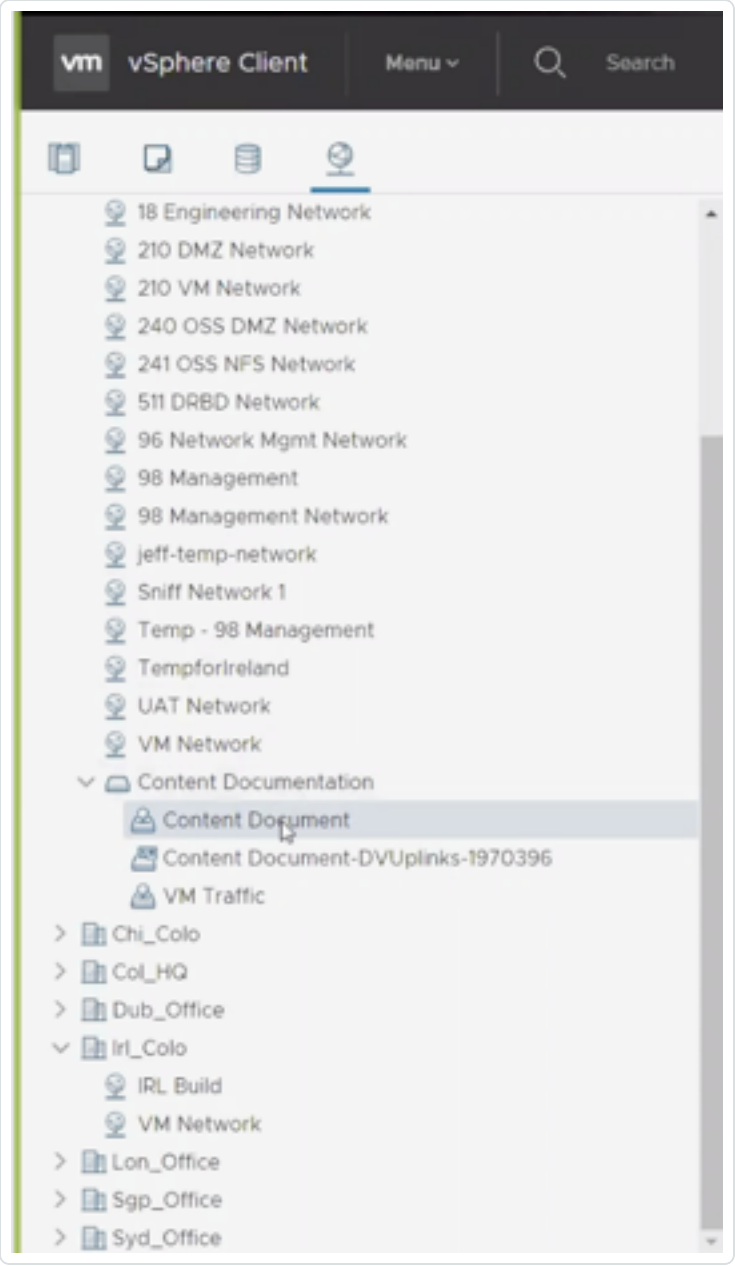


The VDS appears in the **Port Mirroring** section.

(Optional) Configure Port Group that Resides on the VDS

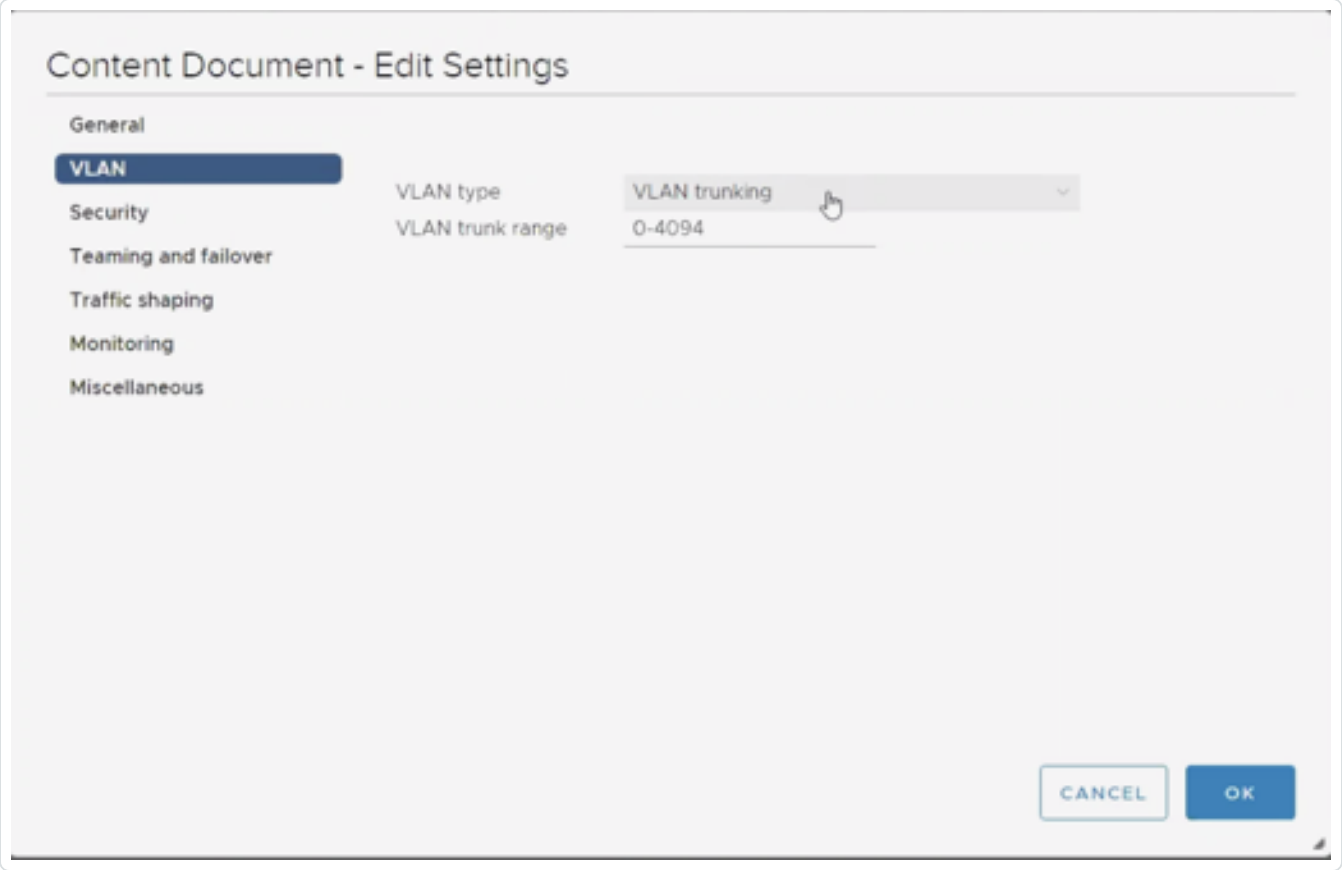
The following steps apply only if your NNM VM has a dedicated port group.

1. In the left panel, click on the VDS for which you previously configured port mirroring.
2. Below the VDS, select the port group.



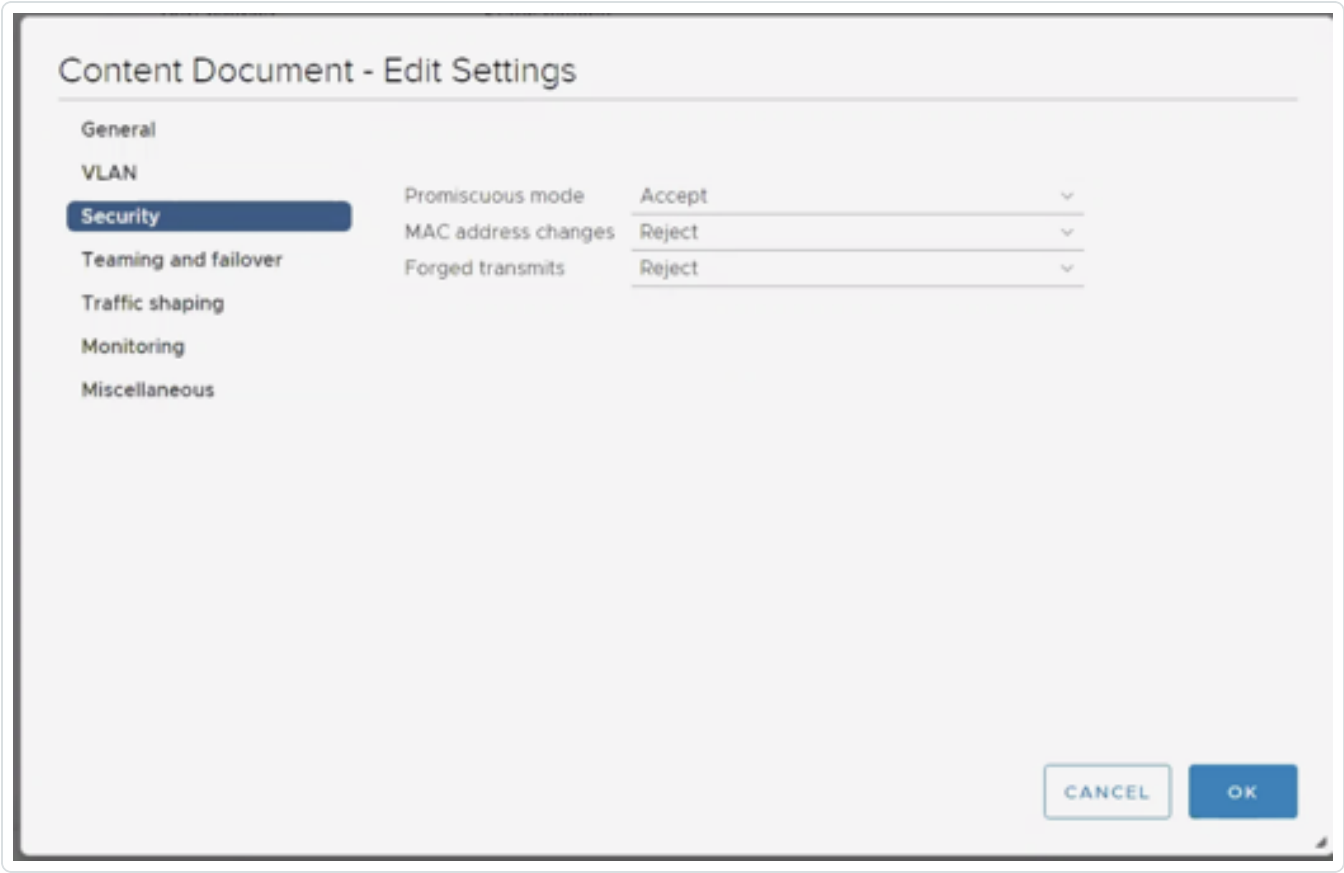
3. Click **Configure**.
4. Click **Edit**.

5. Click **VLAN**.



6. In the **VLAN Type** drop-down box, select **VLAN Trunking**.
7. In the **Range** box, set the value to **0-4094**.

8. Click **Security**.



9. Set **Promiscuous Mode** to **Accept**.
10. Click **OK**.

Power on Your VMs

> **Note:** To avoid issues, Tenable recommends ensuring that the source and traffic direction for port mirroring is correct before powering on your VMs.

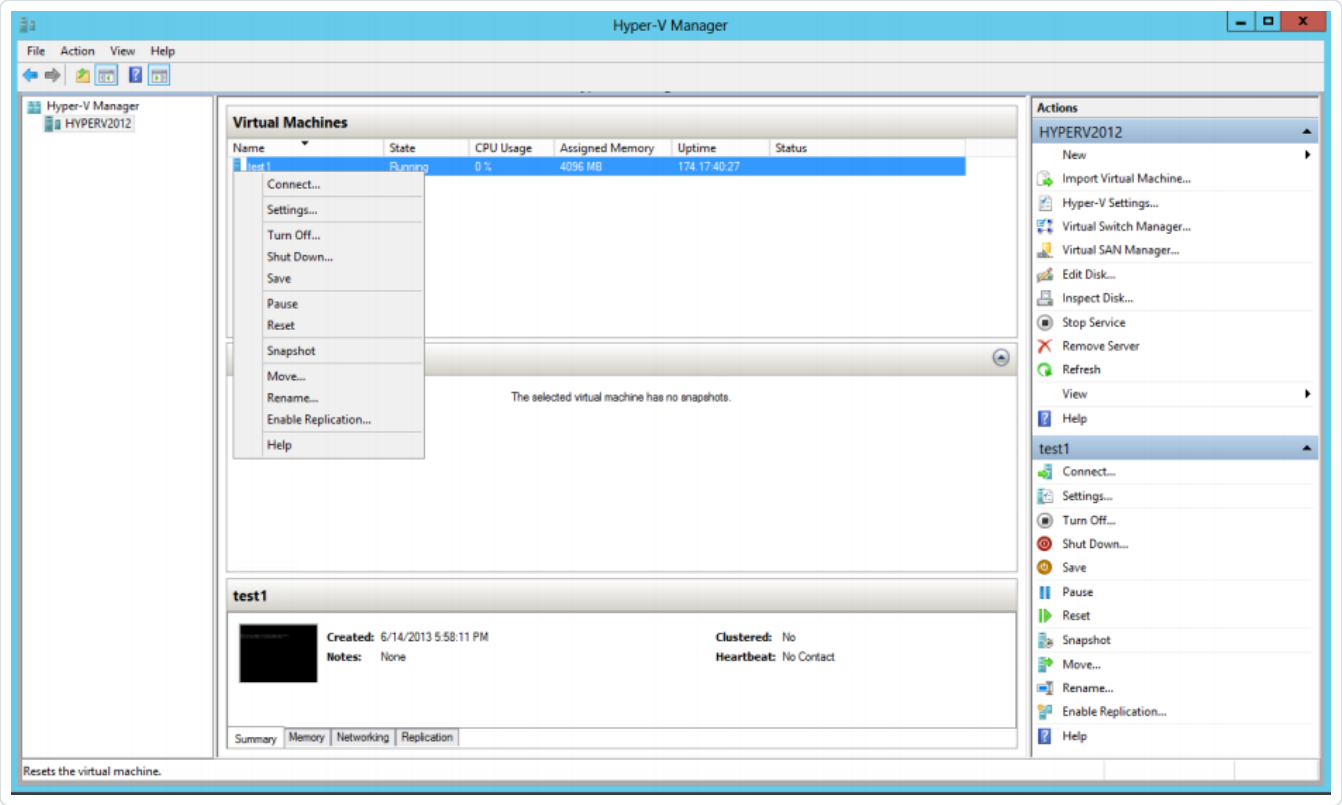
If your VMs are not already powered on, power them on. Traffic begins flowing and NNM begins collecting data.

Microsoft Hyper-V

The configuration settings have been configured using Hyper-V running on Microsoft Server 2012. Hyper-V mirroring settings are between VMs utilizing virtual ports on the same virtual switch. When adjusting the settings, the VM must be powered off. After the changes are made, power on the VM to enable the new configuration.

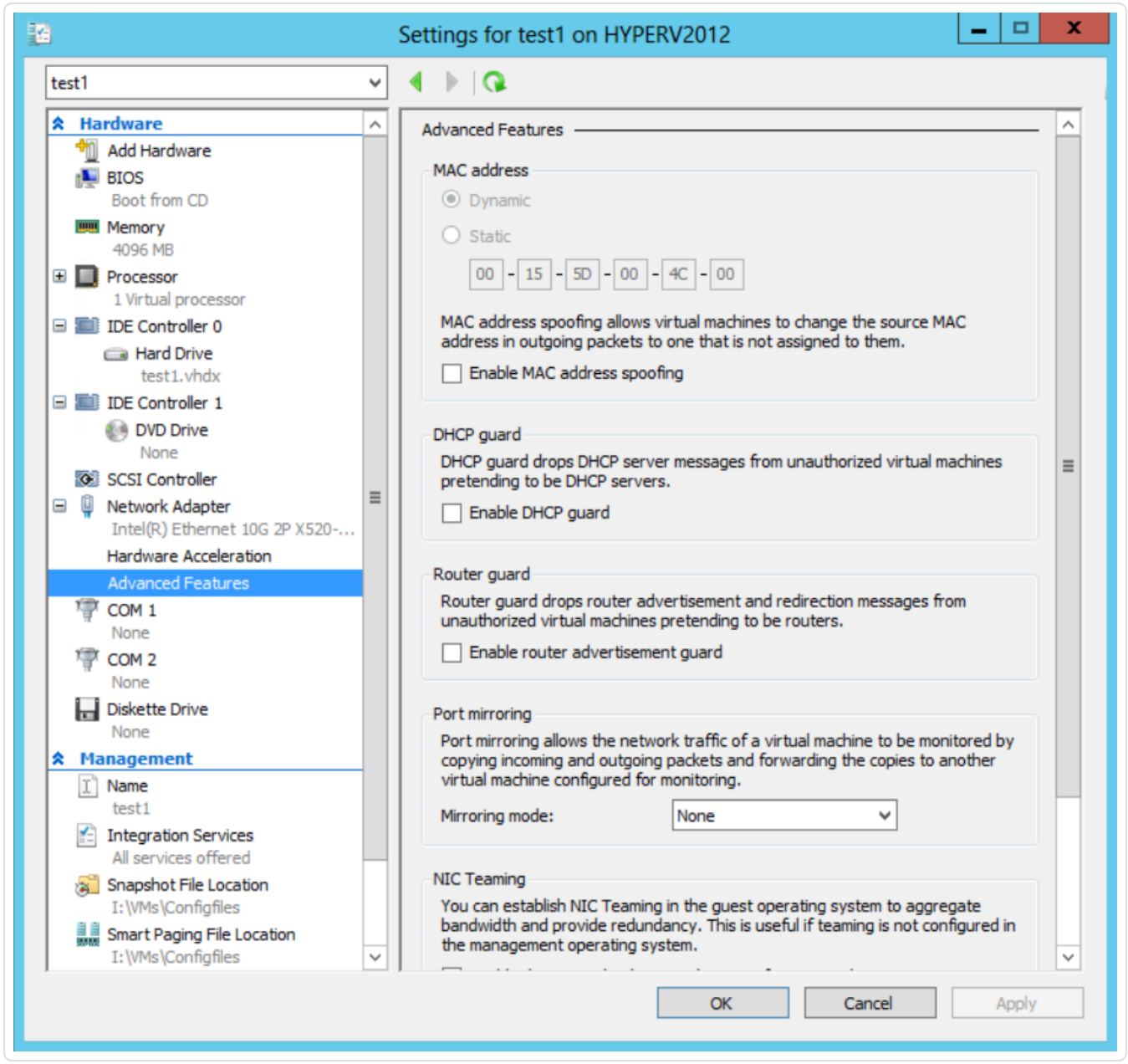
To configure the mirroring destination port on the NNM server VM:

1. Under **Actions** on the NNM VM, navigate to the **Settings** option.

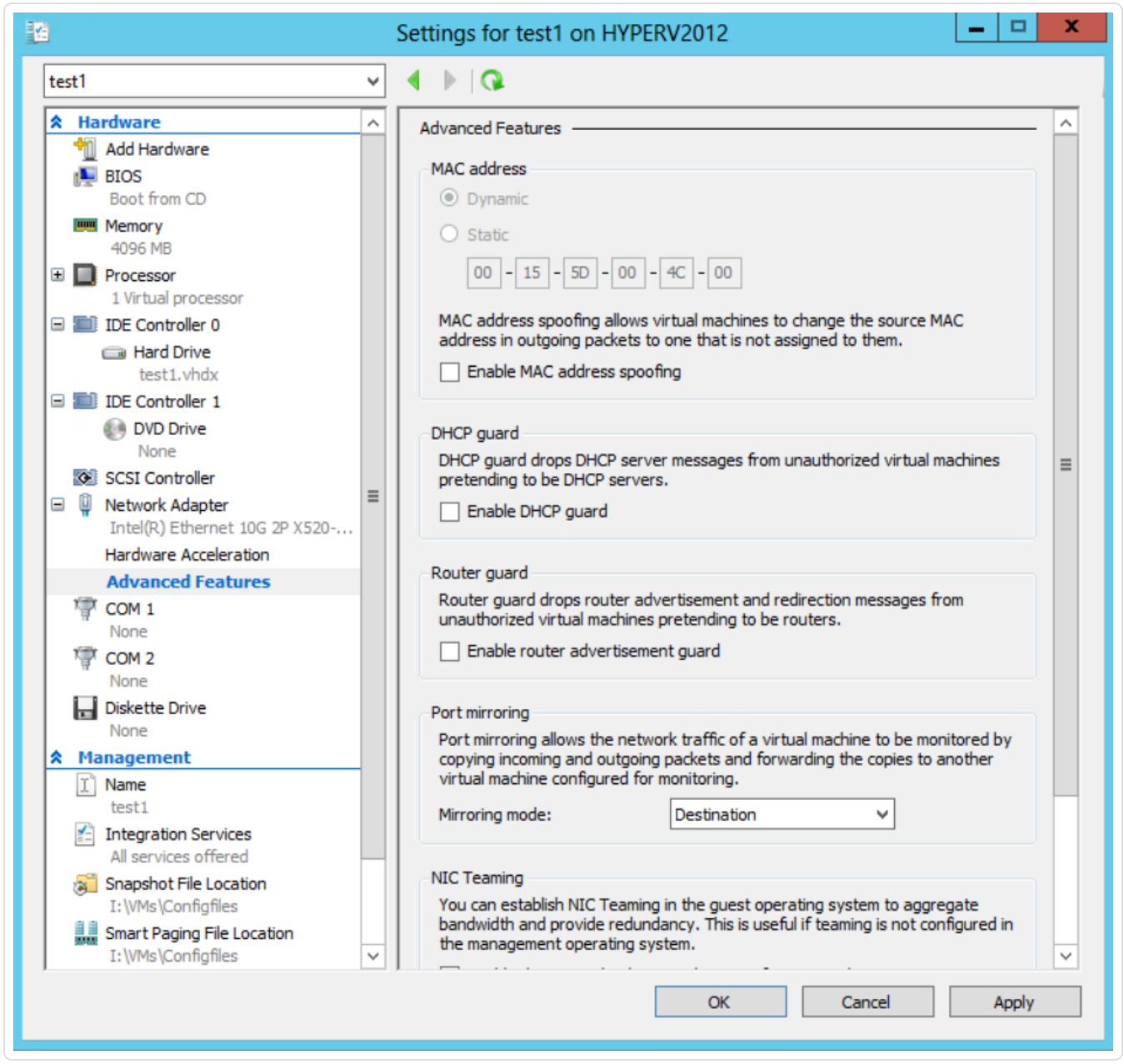


2. Select the **Advanced Features** option on the network adaptor to use to receive port mirrored

traffic from other VMs.



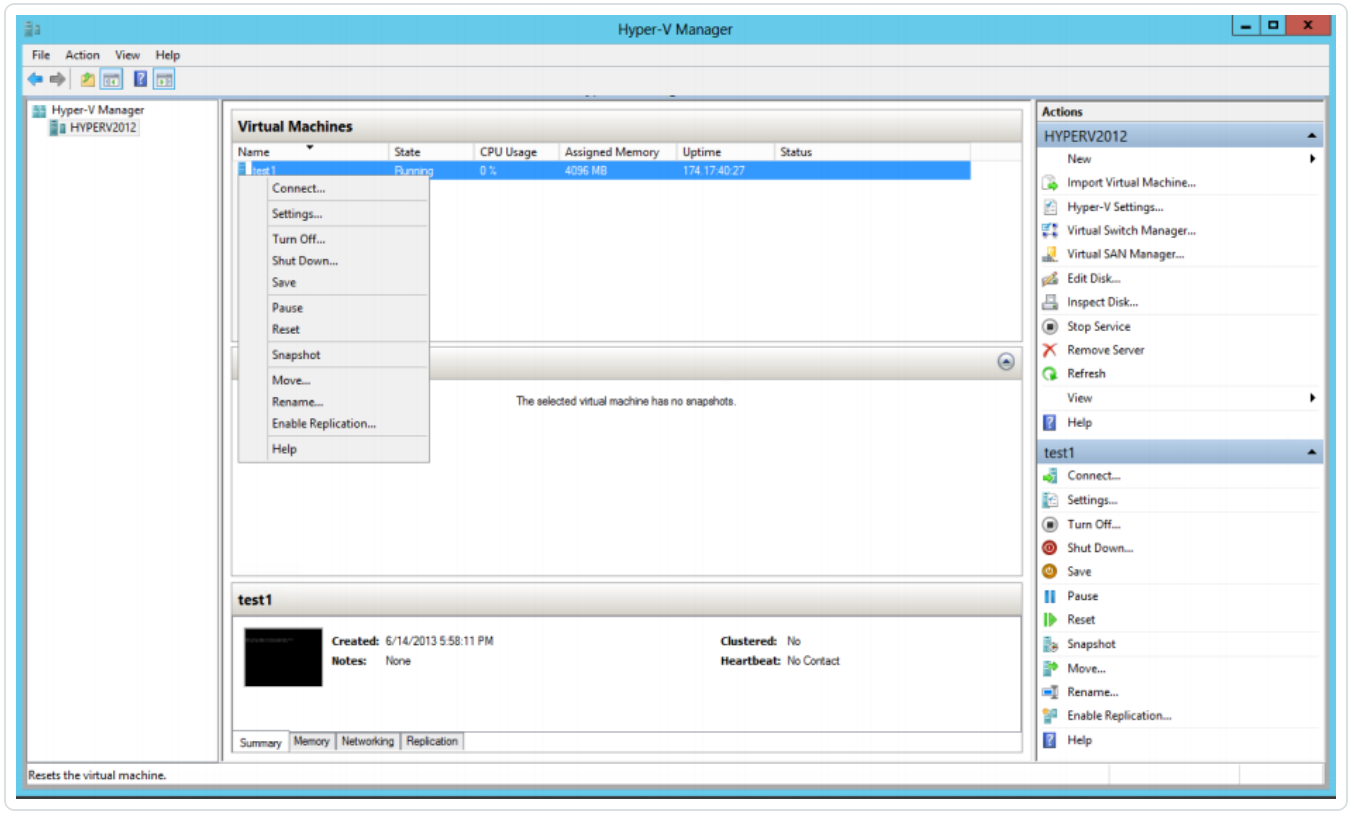
3. In the **Port Mirroring** section, from the **Mirroring Mode** drop-down menu, select **Destination**.



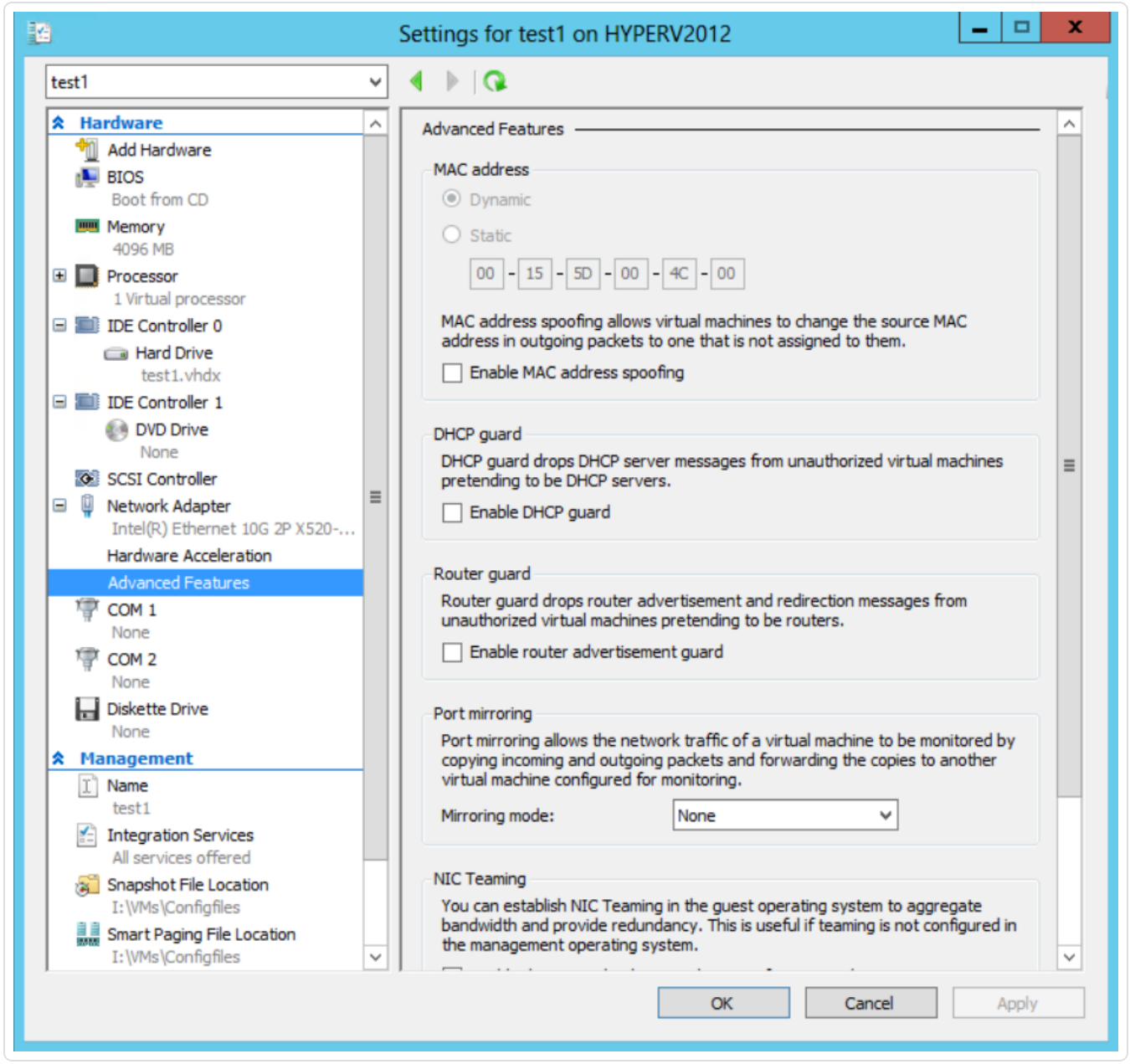
4. Click **Apply**.
5. Click **OK**.
6. Start the VM with NNM monitoring the configured port.

To configure the mirrored ports of the monitored VMs:

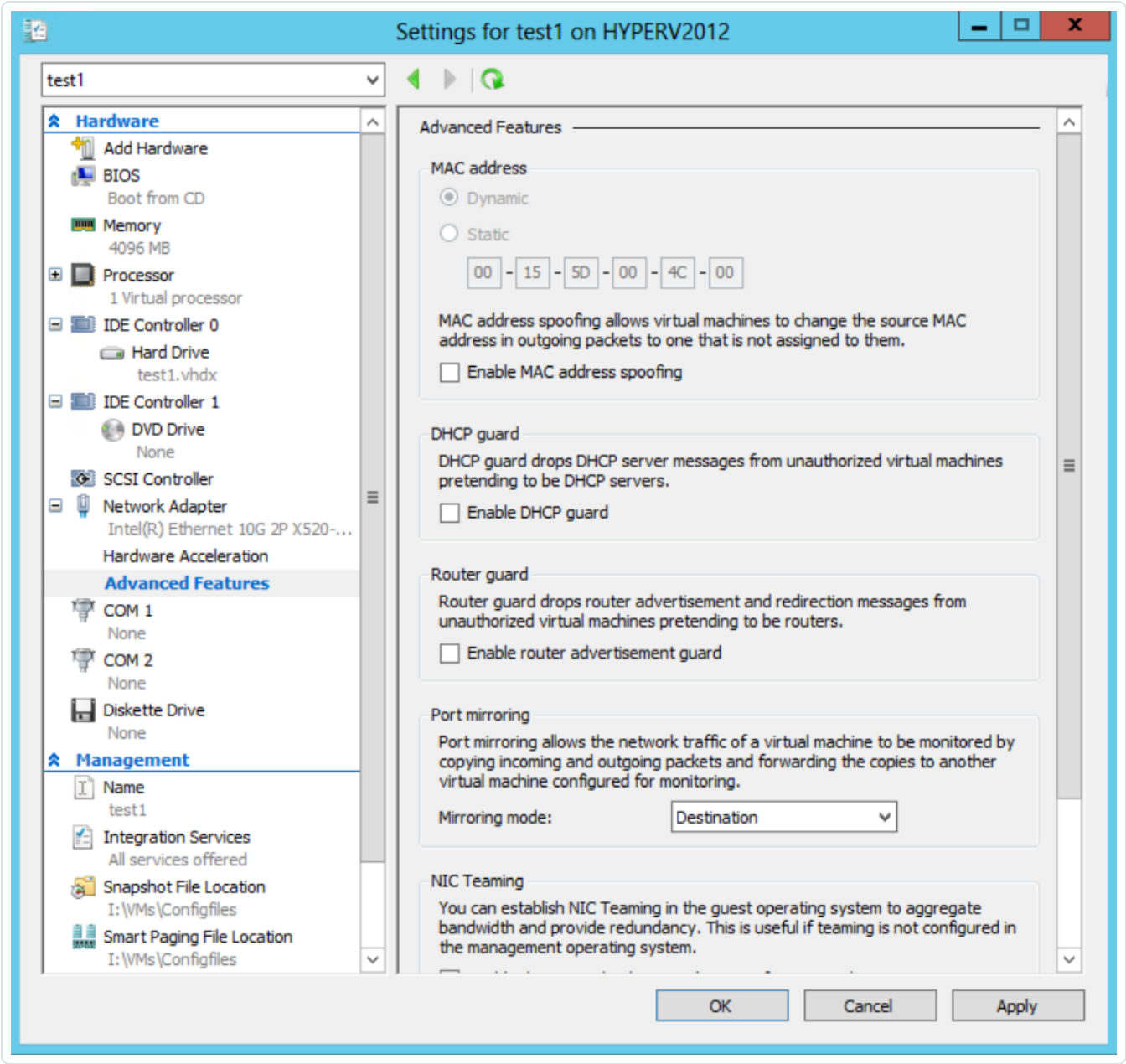
1. Under **Actions** on the monitored VM, navigate to the **Settings** option.



2. Select the **Advanced Features** option on the network adaptor(s) to use to send port mirrored traffic to the port that NNM monitors.


Settings for test1 on HYPERV2012
test1
Hardware
Add Hardware
BIOS
Boot from CD
Memory
4096 MB
Processor
1 Virtual processor
IDE Controller 0
Hard Drive
test1.vhdx
IDE Controller 1
DVD Drive
None
SCSI Controller
Network Adapter
Intel(R) Ethernet 10G 2P X520-...
Hardware Acceleration
Advanced Features
COM 1
None
COM 2
None
Diskette Drive
None
Management
Name
test1
Integration Services
All services offered
Snapshot File Location
I:\VMs\Configfiles
Smart Paging File Location
I:\VMs\Configfiles
Advanced Features
MAC address
Dynamic
Static
00 - 15 - 5D - 00 - 4C - 00
MAC address spoofing allows virtual machines to change the source MAC address in outgoing packets to one that is not assigned to them.
Enable MAC address spoofing
DHCP guard
DHCP guard drops DHCP server messages from unauthorized virtual machines pretending to be DHCP servers.
Enable DHCP guard
Router guard
Router guard drops router advertisement and redirection messages from unauthorized virtual machines pretending to be routers.
Enable router advertisement guard
Port mirroring
Port mirroring allows the network traffic of a virtual machine to be monitored by copying incoming and outgoing packets and forwarding the copies to another virtual machine configured for monitoring.
Mirroring mode:
None
NIC Teaming
You can establish NIC Teaming in the guest operating system to aggregate bandwidth and provide redundancy. This is useful if teaming is not configured in the management operating system.
OK
Cancel
Apply

3. In the **Port Mirroring** section, from the **Mirroring Mode** drop-down menu, select **Source**.



4. Click **Apply**.
5. Click **OK**.
6. Start the VM. Traffic to and from the configured port is sent to the destination port configured on the NNM server.

Pass Data from an External Source

You can also pass data from an external source, such as from a router, through Hyper-V to the NNM server. This allows you to mirror traffic from an internal VM to an internal NNM server.

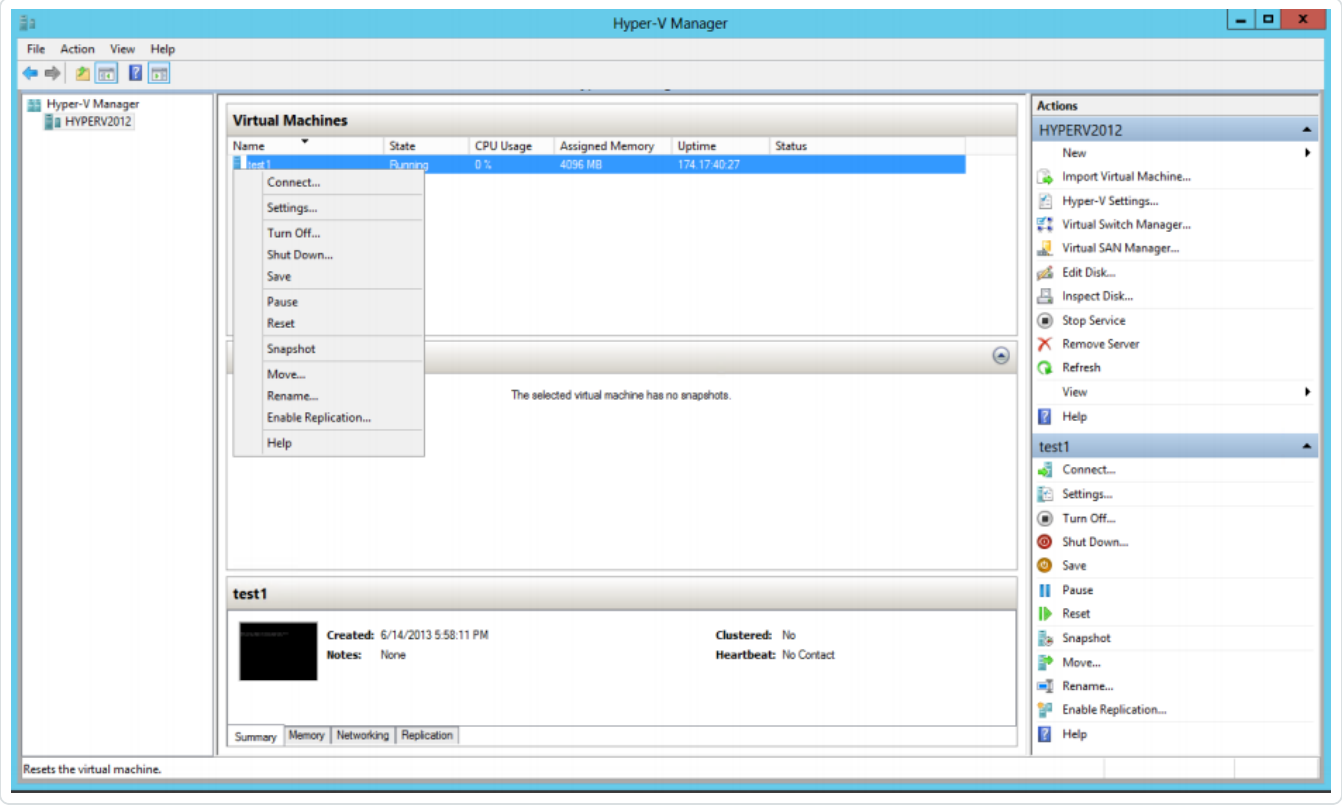
Pass External Data through Microsoft Hyper-V

You can pass data from an external source, such as a router, through Hyper-V to the Tenable Nessus Network Monitor server. This allows you to mirror traffic from an internal virtual machine to an internal instance of Tenable Nessus Network Monitor.

Note: This can only be done on Windows 2012 with a hot patch (http://support.microsoft.com/kb/2885541/en-us) or on Windows 2016. These steps do not apply to Windows 2008.

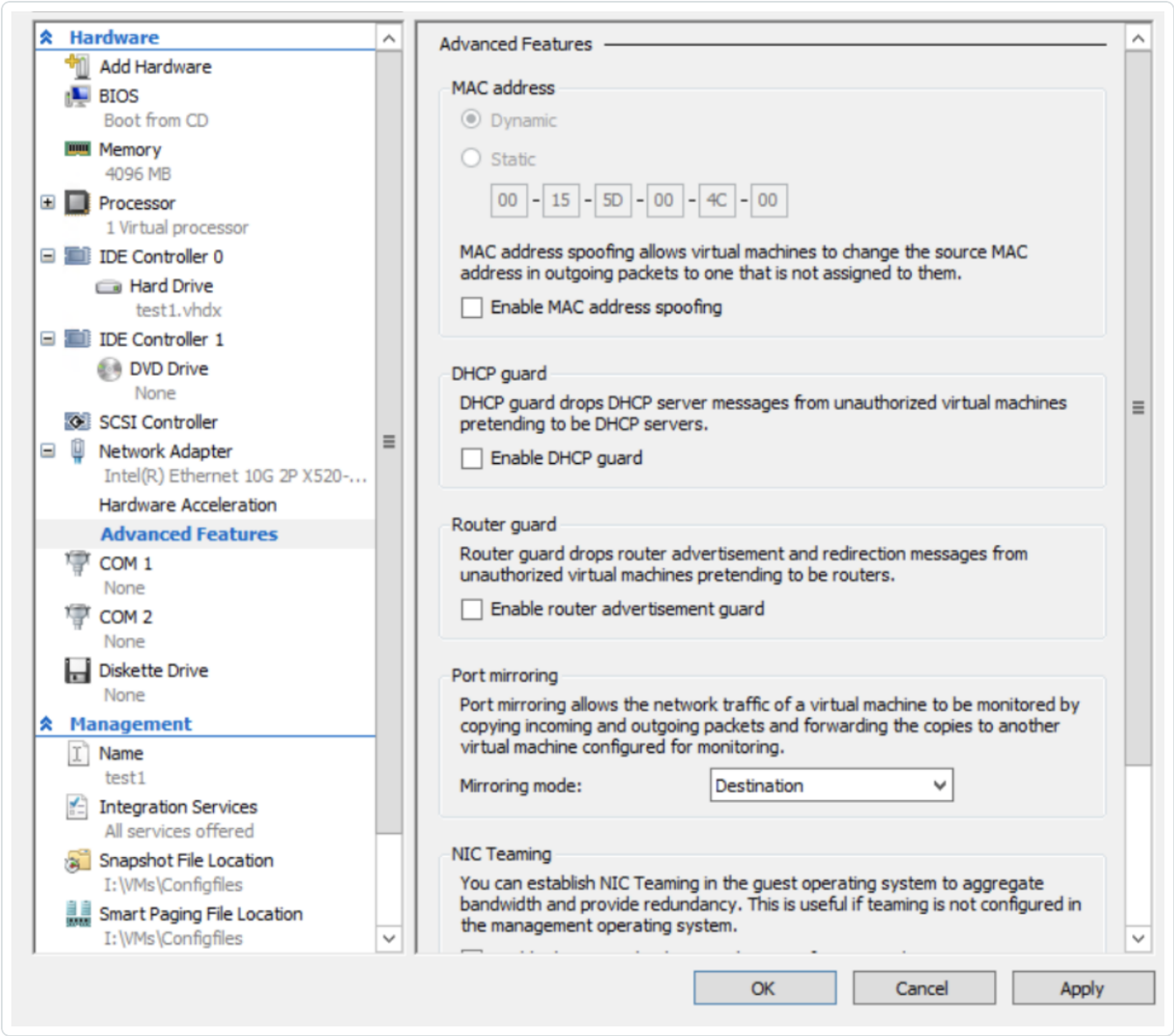
To pass external data through Hyper-V to an internal instance of Tenable Nessus Network Monitor:

1. Power down your virtual machine.
2. Under **Actions** on the monitored virtual machine, navigate to the **Settings** option.



3. Click **SPAN/Mirror NW Adapter - Advanced Features**.

4. In the **Port Mirroring** section, from the **Mirroring Mode** drop-down, select **Destination**.



5. Click **Apply**.
6. Click **OK**.
7. Start your virtual machine.
8. To enable mirror source on the external interface, run the following command:

> **Note:** Values in red must be changed to match your specific virtual machine configuration.

```
$a = Get-VMSystemSwitchExtensionPortFeature -FeatureId 776e0ba7-94a1-41c8-8f28-
951f524251b5
$a.SettingData.MonitorMode = 2
add-VMSwitchExtensionPortFeature -ExternalPort -SwitchName "<MS VSwitch Name>" -
VMSwitchExtensionFeature $a
```

9. To set all VLANs and native VLAN on the span port, run the following command:

> **Note:** Values in red must be changed to match your specific virtual machine configuration.

```
Get-VMNetworkAdapter -VMName “<VMName>” | Where-Object -Property MacAd-
dress -eq "<VM_MAC_Address>" | Set-VMNetworkAdapterVlan -Trunk -
AllowedVlanIdList "1-4094" -NativeVlanId
```