

Contents

[Network Watcher Documentation](#)

[Overview](#)

[About Network Watcher](#)

[Quickstarts](#)

[Connection Monitor \(Preview\)](#)

[Diagnose VM traffic filter problem - Portal](#)

[Diagnose VM traffic filter problem - PowerShell](#)

[Diagnose VM traffic filter problem - Azure CLI](#)

[Tutorials](#)

[Diagnose a VM routing problem](#)

[Monitor communication between VMs](#)

[Diagnose a communication problem between networks](#)

[Log VM network traffic](#)

[Concepts](#)

[Diagnose VM network traffic filter problems](#)

[Diagnose VM routing problems](#)

[Diagnose outbound VM communication problems](#)

[Troubleshoot VPN connectivity problems](#)

[Variable packet capture](#)

[Network security group flow logging](#)

[Network security group view](#)

[Role-based access control permissions](#)

[How-to guides](#)

[Configure Network Watcher](#)

[Diagnose VM network problems](#)

[Install VM extension](#)

[Windows](#)

[Linux](#)

[Diagnose a routing problem](#)

- Azure PowerShell
 - Azure CLI
- Diagnose outbound connection problems
 - Azure portal
 - Azure PowerShell
 - Azure CLI
 - REST
- Capture and analyze packets
 - Manage a packet capture
 - Azure portal
 - Azure PowerShell
 - Azure CLI
 - REST
 - Analyze a packet capture
 - Find anomalies
 - Proactive network monitoring with Azure Functions
 - Perform intrusion detection using open source tools
 - Visualize network traffic patterns using open source tools
- Work with network security groups
 - Configure NSG flow logs
 - Azure PowerShell
 - Azure CLI
 - REST
 - Azure Resource Manager
 - Delete NSG flow log storage blobs
 - Analyze NSG flow logs
 - Read NSG flow logs
 - Use traffic analytics
 - Frequently asked questions
 - Schema and Data Aggregation
 - Use Power BI
 - Use Elastic Stack

[Use Grafana](#)

[Use Graylog](#)

[View network security groups](#)

[Azure PowerShell](#)

[Azure CLI](#)

[REST](#)

[Perform compliance and audit your network](#)

[Diagnose VPN gateway and connections](#)

[Troubleshoot](#)

[Azure PowerShell](#)

[Azure CLI](#)

[REST](#)

[Monitor VPN gateway with Azure Automation](#)

[Diagnose on-premises connectivity via VPN gateway](#)

[View network topology](#)

[Determine relative latency between a location and Azure region](#)

[Monitor network performance](#)

[Reference](#)

[Azure CLI](#)

[Azure PowerShell](#)

[Java](#)

[Ruby](#)

[Python](#)

[.NET](#)

[Node.js](#)

[REST](#)

[Resources](#)

[Network Watcher FAQs](#)

[Azure Roadmap](#)

[MSDN forum](#)

[Pricing](#)

[Service updates](#)

SLA

Stack Overflow

What is Azure Network Watcher?

1/28/2020 • 8 minutes to read • [Edit Online](#)

Azure Network Watcher provides tools to monitor, diagnose, view metrics, and enable or disable logs for resources in an Azure virtual network. Network Watcher is designed to monitor and repair the network health of IaaS (Infrastructure-as-a-Service) products which includes Virtual Machines, Virtual Networks, Application Gateways, Load balancers, etc. Note: It is not intended for and will not work for PaaS monitoring or Web analytics.

Monitoring

Monitor communication between a virtual machine and an endpoint

Endpoints can be another virtual machine (VM), a fully qualified domain name (FQDN), a uniform resource identifier (URI), or IPv4 address. The *connection monitor* capability monitors communication at a regular interval and informs you of reachability, latency, and network topology changes between the VM and the endpoint. For example, you might have a web server VM that communicates with a database server VM. Someone in your organization may, unknown to you, apply a custom route or network security rule to the web server or database server VM or subnet.

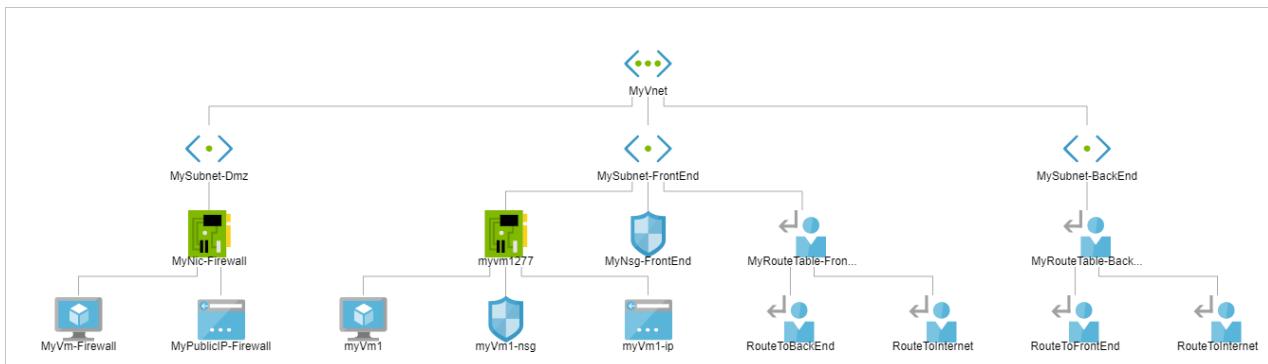
If an endpoint becomes unreachable, connection troubleshoot informs you of the reason. Potential reasons are a DNS name resolution problem, the CPU, memory, or firewall within the operating system of a VM, or the hop type of a custom route, or security rule for the VM or subnet of the outbound connection. Learn more about [security rules](#) and [route hop types](#) in Azure.

Connection monitor also provides the minimum, average, and maximum latency observed over time. After learning the latency for a connection, you may find that you're able to decrease the latency by moving your Azure resources to different Azure regions. Learn more about determining [relative latencies between Azure regions and internet service providers](#) and how to monitor communication between a VM and an endpoint with [connection monitor](#). If you'd rather test a connection at a point in time, rather than monitor the connection over time, like you do with connection monitor, use the [connection troubleshoot](#) capability.

Network performance monitor is a cloud-based hybrid network monitoring solution that helps you monitor network performance between various points in your network infrastructure. It also helps you monitor network connectivity to service and application endpoints and monitor the performance of Azure ExpressRoute. Network performance monitor detects network issues like traffic blackholing, routing errors, and issues that conventional network monitoring methods aren't able to detect. The solution generates alerts and notifies you when a threshold is breached for a network link. It also ensures timely detection of network performance issues and localizes the source of the problem to a particular network segment or device. Learn more about [network performance monitor](#).

View resources in a virtual network and their relationships

As resources are added to a virtual network, it can become difficult to understand what resources are in a virtual network and how they relate to each other. The *topology* capability enables you to generate a visual diagram of the resources in a virtual network, and the relationships between the resources. The following picture shows an example topology diagram for a virtual network that has three subnets, two VMs, network interfaces, public IP addresses, network security groups, route tables, and the relationships between the resources:



You can download an editable version of the picture in [svg](#) format. Learn more about [topology view](#).

Diagnostics

Diagnose network traffic filtering problems to or from a VM

When you deploy a VM, Azure applies several default security rules to the VM that allow or deny traffic to or from the VM. You might override Azure's default rules, or create additional rules. At some point, a VM may become unable to communicate with other resources, because of a security rule. The *IP flow verify* capability enables you to specify a source and destination IPv4 address, port, protocol (TCP or UDP), and traffic direction (inbound or outbound). IP flow verify then tests the communication and informs you if the connection succeeds or fails. If the connection fails, IP flow verify tells you which security rule allowed or denied the communication, so that you can resolve the problem. Learn more about IP flow verify by completing the [Diagnose a virtual machine network traffic filter problem](#) tutorial.

Diagnose network routing problems from a VM

When you create a virtual network, Azure creates several default outbound routes for network traffic. The outbound traffic from all resources, such as VMs, deployed in a virtual network, are routed based on Azure's default routes. You might override Azure's default routes, or create additional routes. You may find that a VM can no longer communicate with other resources because of a specific route. The *next hop* capability enables you to specify a source and destination IPv4 address. Next hop then tests the communication and informs you what type of next hop is used to route the traffic. You can then remove, change, or add a route, to resolve a routing problem. Learn more about the [next hop](#) capability.

Diagnose outbound connections from a VM

The *connection troubleshoot* capability enables you to test a connection between a VM and another VM, an FQDN, a URI, or an IPv4 address. The test returns similar information returned when using the [connection monitor](#) capability, but tests the connection at a point in time, rather than monitoring it over time, as connection monitor does. Learn more about how to troubleshoot connections using [connection-troubleshoot](#).

Capture packets to and from a VM

Advanced filtering options and fine-tuned controls, such as the ability to set time and size limitations, provide versatility. The capture can be stored in Azure Storage, on the VM's disk, or both. You can then analyze the capture file using several standard network capture analysis tools. Learn more about [packet capture](#).

Diagnose problems with an Azure Virtual network gateway and connections

Virtual network gateways provide connectivity between on-premises resources and Azure virtual networks. Monitoring gateways and their connections are critical to ensuring communication is not broken. The *VPN diagnostics* capability provides the ability to diagnose gateways and connections. VPN diagnostics diagnoses the health of the gateway, or gateway connection, and informs you whether a gateway and gateway connections, are available. If the gateway or connection is not available, VPN diagnostics tells you why, so you can resolve the problem. Learn more about VPN diagnostics by completing the [Diagnose a communication problem between networks](#) tutorial.

Determine relative latencies between Azure regions and internet service providers

You can query Network Watcher for latency information between Azure regions and across internet service providers. When you know latencies between Azure regions and across Internet service providers, you can deploy Azure resources to optimize network response time. Learn more about [relative latencies](#).

View security rules for a network interface

The effective security rules for a network interface are a combination of all security rules applied to the network interface, and the subnet the network interface is in. The *security group view* capability shows you all security rules applied to the network interface, the subnet the network interface is in, and the aggregate of both. With an understanding of which rules are applied to a network interface, you can add, remove, or change rules, if they're allowing or denying traffic that you want to change. Learn more about [security group view](#).

Metrics

There are [limits](#) to the number of network resources that you can create within an Azure subscription and region. If you meet the limits, you're unable to create more resources within the subscription or region. The *network subscription limit* capability provides a summary of how many of each network resource you have deployed in a subscription and region, and what the limit is for the resource. The following picture shows the partial output for network resources deployed in the East US region for an example subscription:

Subscription	Location	
	East US	
NAME	CURRENT LIMIT	USAGE
VirtualNetworks	50	2
StaticPublicIPAddresses	20	0
NetworkSecurityGroups	100	3
PublicIPAddresses	60	5
NetworkInterfaces	24000	4
LoadBalancers	100	0
ApplicationGateways	50	0
RouteTables	100	2

The information is helpful when planning future resource deployments.

Logs

Analyze traffic to or from a network security group

Network security groups (NSG) allow or deny inbound or outbound traffic to a network interface in a VM. The *NSG flow log* capability allows you to log the source and destination IP address, port, protocol, and whether traffic was allowed or denied by an NSG. You can analyze logs using a variety of tools, such as PowerBI and the *traffic analytics* capability. Traffic analytics provides rich visualizations of data written to NSG flow logs. The following picture shows some of the information and visualizations that traffic analytics presents from NSG flow log data:

Azure data centers with... 1	Active ports on inte... 1	Hosts active on inte... 1	Malicious flows 1	
7	6	26	2.87k	Allowed 1 103 Blocked 2.77k
out of 36				

Traffic distribution across Azure data centers 1

Active DCs	3	Traffic analytics enabl... 1
Inactive DCs	4	Total 6
Allowed malicious... 1		Active countries 1

1/7

DCs affected

[Click here to see live geomap](#)



Virtual network distribu1

VNet traffic 1	Ex
Total 89	Or
Active 14	Az
Inactive 75	Pu

[Click here to see VNet traffic topol](#)

Hosts with most traffic

4.5M

Total flows

IP	%	DISTRIBU...
10.156.1.4	9.02	<div style="width: 9.02%; background-color: #0072bc;"></div>
10.186.1.4	7.4	<div style="width: 7.4%; background-color: #0072bc;"></div>

Most frequent conversations

4.5M

Total flows

SOURCE	DESTI...	%	DISTRI...
10.186...	10.15...	3.33	<div style="width: 3.33%; background-color: #0072bc;"></div>
westus2	10.4.1.4	1.46	<div style="width: 1.46%; background-color: #0072bc;"></div>

Top application protocols

4.5M

Total flows

L7 PROTO...	%	DISTRIBU...
http	6.06	<div style="width: 6.06%; background-color: #0072bc;"></div>
ldap	1.43	<div style="width: 1.43%; background-color: #0072bc;"></div>

Learn more about NSG flow logs by completing the [Log network traffic to and from a virtual machine](#) tutorial and how to implement [traffic analytics](#).

View diagnostic logs for network resources

You can enable diagnostic logging for Azure networking resources such as network security groups, public IP addresses, load balancers, virtual network gateways, and application gateways. The *Diagnostic logs* capability provides a single interface to enable and disable network resource diagnostic logs for any existing network resource that generates a diagnostic log. You can view diagnostic logs using tools such as Microsoft Power BI and Azure Monitor logs. To learn more about analyzing Azure network diagnostic logs, see [Azure network solutions in Azure Monitor logs](#).

Network Watcher automatic enablement

When you create or update a virtual network in your subscription, Network Watcher will be enabled automatically in your Virtual Network's region. There is no impact to your resources or associated charge for automatically enabling Network Watcher. For more information, see [Network Watcher create](#).

Next steps

You now have an overview of Azure Network Watcher. To get started using Network Watcher, diagnose a common communication problem to and from a virtual machine using IP flow verify. To learn how, see the [Diagnose a virtual machine network traffic filter problem](#) quickstart.

Network Connectivity Monitoring with Connection Monitor (Preview)

2/25/2020 • 21 minutes to read • [Edit Online](#)

Connection Monitor (Preview) provides unified end-to-end connection monitoring in Azure Network Watcher. The Connection Monitor (Preview) feature supports hybrid and Azure cloud deployments. Network Watcher provides tools to monitor, diagnose, and view connectivity-related metrics for your Azure deployments.

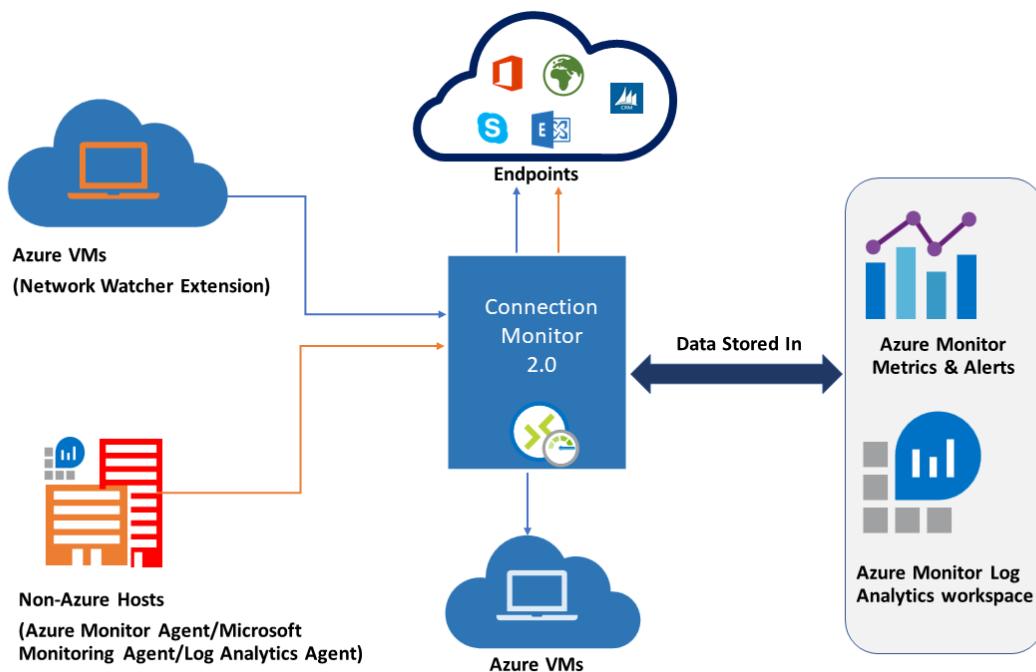
Here are some use cases for Connection Monitor (Preview):

- Your front-end web server VM communicates with a database server VM in a multi-tier application. You want to check network connectivity between the two VMs.
- You want VMs in the East US region to ping VMs in the Central US region, and you want to compare cross-region network latencies.
- You have multiple on-premises office sites in Seattle, Washington, and in Ashburn, Virginia. Your office sites connect to Office 365 URLs. For your users of Office 365 URLs, compare the latencies between Seattle and Ashburn.
- Your hybrid application needs connectivity to an Azure Storage endpoint. Your on-premises site and your Azure application connect to the same Azure Storage endpoint. You want to compare the latencies of the on-premises site to the latencies of the Azure application.
- You want to check the connectivity between your on-premises setups and the Azure VMs that host your cloud application.

In its preview phase, Connection Monitor combines the best of two features: the Network Watcher [Connection Monitor](#) feature and the Network Performance Monitor (NPM) [Service Connectivity Monitor](#) feature.

Here are some benefits of Connection Monitor (Preview):

- Unified, intuitive experience for Azure and hybrid monitoring needs
- Cross-region, cross-workspace connectivity monitoring
- Higher probing frequencies and better visibility into network performance
- Faster alerting for your hybrid deployments
- Support for connectivity checks that are based on HTTP, TCP, and ICMP
- Metrics and Log Analytics support for both Azure and non-Azure test setups



To start using Connection Monitor (Preview) for monitoring, follow these steps:

1. Install monitoring agents.
2. Enable Network Watcher on your subscription.
3. Create a connection monitor.
4. Set up data analysis and alerts.
5. Diagnose issues in your network.

The following sections provide details for these steps.

Install monitoring agents

Connection Monitor relies on lightweight executable files to run connectivity checks. It supports connectivity checks from both Azure environments and on-premises environments. The executable file that you use depends on whether your VM is hosted on Azure or on-premises.

Agents for Azure virtual machines

To make Connection Monitor recognize your Azure VMs as monitoring sources, install the Network Watcher Agent virtual machine extension on them. This extension is also known as the *Network Watcher extension*. Azure virtual machines require the extension to trigger end-to-end monitoring and other advanced functionality.

You can install the Network Watcher extension when you [create a VM](#). You can also separately install, configure, and troubleshoot the Network Watcher extension for [Linux](#) and [Windows](#).

Rules for a network security group (NSG) or firewall can block communication between the source and destination. Connection Monitor detects this issue and shows it as a diagnostic message in the topology. To enable connection monitoring, ensure that NSG and firewall rules allow packets over TCP or ICMP between the source and destination.

Agents for on-premises machines

To make Connection Monitor recognize your on-premises machines as sources for monitoring, install the Log Analytics agent on the machines. Then enable the Network Performance Monitor solution. These agents are linked to Log Analytics workspaces, so you need to set up the workspace ID and primary key before the agents can start monitoring.

To install the Log Analytics agent for Windows machines, see [Azure Monitor virtual machine extension for Windows](#).

If the path includes firewalls or network virtual appliances (NVAs), then make sure that the destination is reachable.

Enable Network Watcher on your subscription

All subscriptions that have a virtual network are enabled with Network Watcher. When you create a virtual network in your subscription, Network Watcher is automatically enabled in the virtual network's region and subscription. This automatic enabling doesn't affect your resources or incur a charge. Ensure that Network Watcher isn't explicitly disabled on your subscription.

For more information, see [Enable Network Watcher](#).

Create a connection monitor

Connection Monitor monitors communication at regular intervals. It informs you of changes in reachability and latency. You can also check the current and historical network topology between source agents and destination endpoints.

Sources can be Azure VMs or on-premises machines that have an installed monitoring agent. Destination endpoints can be Office 365 URLs, Dynamics 365 URLs, custom URLs, Azure VM resource IDs, IPv4, IPv6, FQDN, or any domain name.

Access Connection Monitor (Preview)

1. On the Azure portal home page, go to **Network Watcher**.
2. On the left, in the **Monitoring** section, select **Connection Monitor (Preview)**.
3. You see all of the connection monitors that were created in Connection Monitor (Preview). To see the connection monitors that were created in the classic experience of Connection Monitor, go to the **Connection Monitor** tab.

The screenshot shows the Azure portal's Connection Monitor (Preview) dashboard. At the top, there are filters for Subscriptions (2 of 34 selected), Locations (2 locations), Sources (All sources), Destinations (All destinations), and Time (Current Time: 9/23/2019, 11:32:30). Below this is a summary bar with counts for Fail (2), Warning (5), Indeterminate (0), Not Running (5), and Pass (1). A table lists connection monitors under the 'DemoNetworkMonitoring' group:

CONNECTION MONITOR	TEST CONFIGURATIONS	STATUS	LAST POLLED
Connectivity_To_Outlook	HTTP_TC +1 more	⚠️	9/23/2019 11:31:00 AM
CMPreviewVM(CMPreviewTest) -> outlook.office365.com	HTTP_TC	✗	9/23/2019 11:30:43 AM
WIN-BLGH868DJHC -> outlook.office365.com	HTTP_TC_networkTestConfig	✓	9/23/2019 11:31:00 AM
WIN-GTV0G9E2LCF -> outlook.office365.com	HTTP_TC_networkTestConfig	✓	9/23/2019 11:30:00 AM
WIN-BLGH868DJHC -> outlook.office365.com	HTTP_TC	✓	9/23/2019 11:31:00 AM
WIN-GTV0G9E2LCF -> outlook.office365.com	HTTP_TC	✓	9/23/2019 11:30:00 AM
CMPreviewVM(CMPreviewTest) -> outlook.office365.com	HTTP_TC_networkTestConfig	✓	9/23/2019 11:30:43 AM
EastUS_To_CentralUS	HTTP_TC_ForAzureOnly_networkTestConfig +1 more	✗	9/23/2019 11:31:04 AM
CMPreviewVM(CMPreviewTest) -> CMPreviewCentralEUAF	HTTP_TC_ForAzureOnly_networkTestConfig	✗	9/23/2019 11:30:44 AM
CMPreviewVM(CMPreviewTest) -> CMPreviewCentralEUAF	HTTP_TC_ForAzureOnly	✗	9/23/2019 11:31:04 AM
NetworkMonitoring		⚠️	9/23/2019 11:31:04 AM

Create a connection monitor

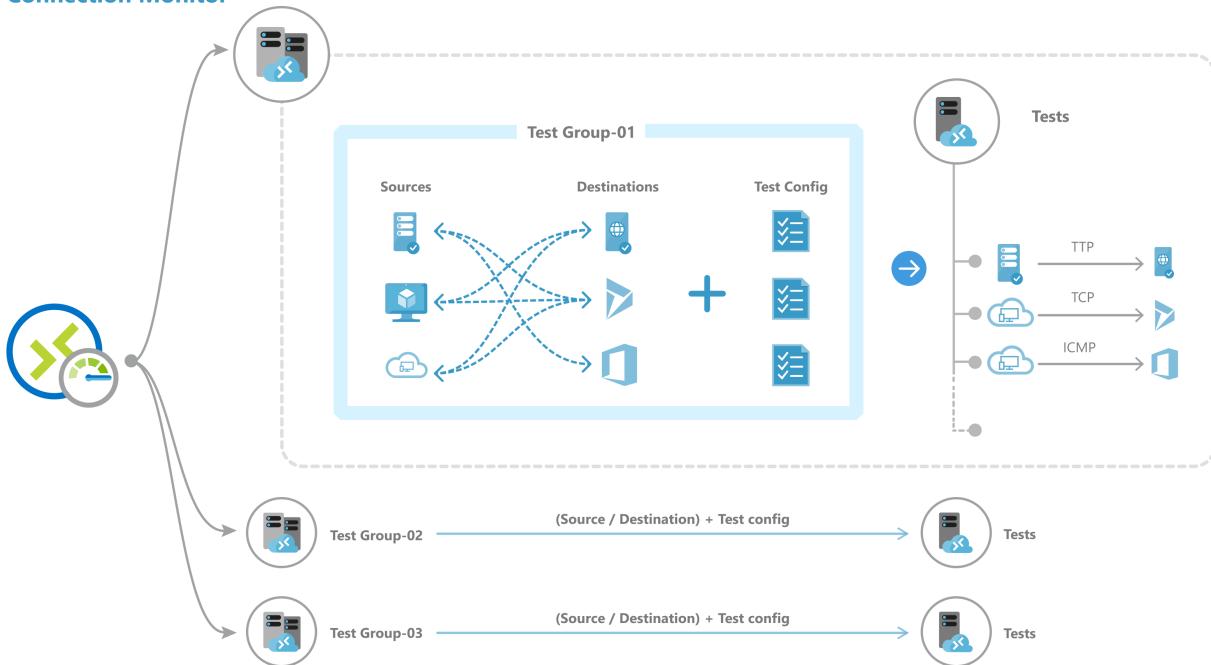
In connection monitors that you create in Connection Monitor (Preview), you can add both on-premises machines and Azure VMs as sources. These connection monitors can also monitor connectivity to endpoints. The endpoints

can be on Azure or any other URL or IP.

Connection Monitor (Preview) includes the following entities:

- **Connection monitor resource** – A region-specific Azure resource. All of the following entities are properties of a connection monitor resource.
- **Endpoint** – A source or destination that participates in connectivity checks. Examples of endpoints include Azure VMs, on-premises agents, URLs, and IPs.
- **Test configuration** – A protocol-specific configuration for a test. Based on the protocol you chose, you can define the port, thresholds, test frequency, and other parameters.
- **Test group** – The group that contains source endpoints, destination endpoints, and test configurations. A connection monitor can contain more than one test group.
- **Test** – The combination of a source endpoint, destination endpoint, and test configuration. A test is the most granular level at which monitoring data is available. The monitoring data includes the percentage of checks that failed and the round-trip time (RTT).

Connection Monitor



Create a connection monitor from the Azure portal

To create a connection monitor from the Azure portal, follow these steps:

1. On the **Connection Monitor (Preview)** dashboard, in the upper-left corner, select **Create**.
2. On the **Basics** tab, enter information for your connection monitor:
 - **Connection Monitor Name** – Add the name of your connection monitor. Use the standard naming rules for Azure resources.
 - **Subscription** – Choose a subscription for your connection monitor.
 - **Region** – Choose a region for your connection monitor. You can select only the source VMs that are created in this region.
 - **Workspace configuration** – Your workspace holds your monitoring data. You can use a custom workspace or the default workspace.
 - To use the default workspace, select the check box.
 - To choose a custom workspace, clear the check box. Then choose the subscription and region for your custom workspace.
3. At the bottom of the tab, select **Next: Test groups**.

Connection Monitor enables you to monitor connectivity in your Azure and hybrid network. Select your preferred subscription and region from which monitoring will be performed. Use workspace configuration to store monitoring data generated by Connection Monitor tests in Log Analytics workspace. Complete the Basics tab then proceed to Test Groups tab. [Learn more](#)

Connection Monitor Name * ContosoCM

Subscription * Contoso Hotels

Don't see a subscription? [Open Directory + Subscription settings](#)

Region * East US

Workspace configuration

Use workspace created by connection monitor (default)

Subscription * 41 selected

Region * 22 selected

Workspace DefaultWorkspace-5733bcb3-7fde-4caf-8629-41dc15...
Contoso Hotels in eastus (system generated)

Next : Test groups >> Review + create Cancel

4. On the **Test groups** tab, select **+ Test group**. To set up your test groups, see [Create test groups in Connection Monitor](#).
5. At the bottom of the tab, select **Next: Review + create** to review your connection monitor.

Add test group details

Test configurations

- Name: HTTPTestGroup
- Protocol: HTTP
- Create TCP test configuration:
- Disable traceroute:
- Destination port: 80
- Test Frequency: Custom, Every 30 seconds
- Success Threshold: Checks failed %, 20
- Round trip time (ms): 50

<< Previous Next : Review + create >> Create Cancel

6. On the **Review + create** tab, review the basic information and test groups before you create the connection monitor. If you need to edit the connection monitor:
 - To edit basic details, select the pencil icon.
 - To edit a test group, select it.

NOTE

The **Review + create** tab shows the cost per month during the Connection Monitor preview stage. Currently, the **CURRENT COST** column shows no charge. When Connection Monitor becomes generally available, this column will show a monthly charge.

Even in the Connection Monitor preview stage, Log Analytics ingestion charges apply.

- When you're ready to create the connection monitor, at the bottom of the**Review + create** tab, select **Create**.

This screenshot shows the 'Create Connection Monitor' wizard in the Microsoft Azure portal. The 'Review + create' tab is selected. In the 'Primary Details' section, the status is 'Enabled', the connection monitor name is 'def', it's associated with 'Network Traffic Analytics Subscription 3', located in 'East US 2' region, and the workspace is 'DefaultWorkspace-af15e575-f948-49ac-bce0-252d028e9379-EUS'. Under 'TEST GROUPS (1)', there is one entry: 'ConnectivityToOutlook' with 'WIN-8LGH868DJHC' as the source and 'account.office.net' as the destination. A note indicates an estimated monthly cost of '\$171'. At the bottom, there are 'Create' and 'Cancel' buttons.

Connection Monitor (Preview) creates the connection monitor resource in the background.

Create a connection monitor by using ARMClient

Use the following code to create a connection monitor by using ARMClient.

```
$connectionMonitorName = "sampleConnectionMonitor"

$ARM = "[https://](https://apac01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fbrazilus.management.azure.com&data=02%7C01%7CManasi.Sant%40microsoft.com%7Cd900da4ed7f24366842108d68022159b%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C636837281231186904&sdata=qHL8zWjkobY9MatRpAVbODwboKSQAqqEFOMnjmfyOnU%3D&reserved=0)management.azure.com"

$SUB = "subscriptions/\&lt;subscription id 1\&gt;"

$NW =
"resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher\_centraluseuap"

$body =
"{
  location: 'eastus',
  properties: {
    endpoints: [
      {
        name: 'workspace',
        type: 'LogAnalytics'
      }
    ]
  }
}

$ARM -replace "%connectionMonitorName%", $connectionMonitorName | Set-AzureRmResource -Properties $body -ResourceGroupName $NW -Scope $SUB -Type $ARM -Force
```

```
resourceId: '/subscriptions/<subscription id>/resourceGroups/<resource group>/providers/Microsoft.OperationalInsights/workspaces/sampleWorkspace',  
  
filter: {  
  
    items: [{  
  
        type: 'AgentAddress',  
  
        address: '<FQDN of your on-premises agent>'  
    }]  
}  
  
},  
  
,  
  
{  
  
name: 'vm1',  
  
resourceId: '/subscriptions/<subscription id>/resourceGroups/<resource group>/providers/Microsoft.Compute/virtualMachines/<vm-name>',  
  
,  
  
{  
  
name: 'vm2',  
  
resourceId: '/subscriptions/<subscription id>/resourceGroups/<resource group>/providers/Microsoft.Compute/virtualMachines/<vm-name>',  
  
,  
  
{  
  
name: 'azure portal',  
  
address: '<URL>',  
  
,  
  
{  
  
name: 'ip',  
  
address: '<IP>',  
  
}  
  
],  
  
testGroups: [{  
  
    name: 'Connectivity to Azure Portal and Public IP',  
  
    testConfigurations: ['http', 'https', 'tcpEnabled', 'icmpEnabled'],  
  
    sources: ['vm1', 'workspace'],  
  
    destinations: ['azure portal', 'ip'],  
  
},  
  
{  
  
    name: 'Connectivity from Azure VM 1 to Azure VM 2',
```

```
testConfigurations: ['http', 'https', 'tcpDisabled', 'icmpDisabled'],
sources: ['vm1'],
destinations: ['vm2'],
disable: true
},
],
testConfigurations: [
  {
    name: 'http',
    testFrequencySec: 60,
    protocol: 'HTTP',
    successThreshold: {
      checksFailedPercent: 50,
      roundTripTimeMs: 3.4
    }
  },
  {
    name: 'https',
    testFrequencySec: 60,
    protocol: 'HTTP',
    httpConfiguration: {
      preferHTTPS: true
    },
    successThreshold: {
      checksFailedPercent: 50,
      roundTripTimeMs: 3.4
    }
  },
  {
    name: 'tcpEnabled',
    testFrequencySec: 30,
    protocol: 'TCP',
    tcpConfiguration: {
      port: 80
    },
    successThreshold: {
      checksFailedPercent: 30,
      roundTripTimeMs: 5.2
    }
  }
]
```

```
        }

    }, {

        name: 'icmpEnabled',

        testFrequencySec: 90,

        protocol: 'ICMP',

        successThreshold: {

            checksFailedPercent: 50,

            roundTripTimeMs: 3.4

        }

    }, {

        name: 'icmpDisabled',

        testFrequencySec: 120,

        protocol: 'ICMP',

        icmpConfiguration: {

            disableTraceRoute: true

        },

        successThreshold: {

            checksFailedPercent: 50,

            roundTripTimeMs: 3.4

        }

    }, {

        name: 'tcpDisabled',

        testFrequencySec: 45,

        protocol: 'TCP',

        tcpConfiguration: {

            port: 80,

            disableTraceRoute: true

        },

        successThreshold: {

            checksFailedPercent: 30,

            roundTripTimeMs: 5.2

        }

    }

]
```

```
}
```

```
} "
```

Here's the deployment command:

```
armclient PUT $ARM/$SUB/$NW/connectionMonitors/$connectionMonitorName/?api-version=2019-07-01 $body -verbose
```

Create test groups in a connection monitor

Each test group in a connection monitor includes sources and destinations that get tested on network parameters. They're tested for the percentage of checks that fail and the RTT over test configurations.

From the Azure portal, to create a test group in a connection monitor, you specify values for the following fields:

- **Disable Test Group** – You can select this field to disable monitoring for all sources and destinations that the test group specifies. This selection is cleared by default.
- **Name** – Name your test group.
- **Sources** – You can specify both Azure VMs and on-premises machines as sources if agents are installed on them. To install an agent for your source, see [Install monitoring agents](#).
 - To choose Azure agents, select the **Azure Agents** tab. Here you see only VMs that are bound to the region that you specified when you created the connection monitor. By default, VMs are grouped into the subscription that they belong to. These groups are collapsed.

You can drill down from the Subscription level to other levels in the hierarchy:

Subscription > Resource groups > VNETs > Subnets > VMs with agents

You can also change the value of the **Group by** field to start the tree from any other level. For example, if you group by virtual network, you see the VMs that have agents in the hierarchy **VNETs > Subnets > VMs with agents**.

The screenshot shows the 'Add Sources' dialog for a connection monitor. The 'Azure Agents' tab is selected. The 'Group by' dropdown is set to 'Subscription' and the 'Region' dropdown is set to 'eastus2euap'. The search bar contains 'Search'. Below the search bar, there are two collapsed sections: 'Network Traffic Analytics Subscription 3' and 'CMPreviewTest'. Under 'CMPreviewTest', there are two expanded items: 'CMPreviewTest-vnet' and 'default'. 'default' is checked. Under 'default', there are two items: 'CMPreviewVM' (IP 10.1.2.4) and another item which is partially visible. At the bottom of the dialog are 'Done' and 'Cancel' buttons.

- To choose on-premises agents, select the **Non-Azure Agents** tab. By default, agents are grouped into workspaces by region. All of these workspaces have the Network Performance Monitor solution configured.

If you need to add Network Performance Monitor to your workspace, get it from [Azure Marketplace](#).

For information about how to add Network Performance Monitor, see [Monitoring solutions in Azure Monitor](#).

In the **Create Connection Monitor** view, on the **Basics** tab, the default region is selected. If you change the region, you can choose agents from workspaces in the new region. You can also change the value of the **Group by** field to group by subnets.

The screenshot shows the 'Add Sources' panel within the 'Edit Connection Monitor' interface. The 'Azure Agents' tab is active. On the left, there's a sidebar with various monitoring icons and a list of test groups: 'Connectivity_To_Outlook' and 'EastUS_To_CentralUS'. The main area has a search bar and dropdowns for 'Select workspace' (set to 'CMPreviewTestWS') and 'Region' (set to '8 selected'). Below these are three checked items in a list: 'WIN-GTV0G9E2LCF', 'WIN-8LGH868DJHC', and 'S3-08-04-W-10.fareast.corp.microsoft.com'. At the bottom are 'Done' and 'Cancel' buttons.

- To review the Azure and non-Azure agents that you selected, go to the **Review** tab.

The screenshot shows the 'Add Sources' panel with the 'Review' tab selected. The left sidebar remains the same. The main area is divided into two sections: 'Azure Agents' (containing 1 item: 'CMPreviewVM') and 'Non-Azure Agents' (containing 2 items: 'WIN-GTV0G9E2LCF' and 'WIN-8LGH868DJHC'). At the bottom are 'Done' and 'Cancel' buttons.

- When you finish setting up sources, at the bottom of the **Add Sources** panel, select **Done**.
- **Destinations** – You can monitor connectivity to Azure VMs or any endpoint (a public IP, URL, or FQDN) by specifying them as destinations. In a single test group, you can add Azure VMs, Office 365 URLs, Dynamics 365 URLs, and custom endpoints.
 - To choose Azure VMs as destinations, select the **Azure VMs** tab. By default, the Azure VMs are grouped into a subscription hierarchy that's in the same region that you selected in the **Create Connection Monitor** view, on the **Basics** tab. You can change the region and choose Azure VMs from the newly selected region. Then you can drill down from Subscription level to other levels in the

hierarchy, like the Azure Agents level.

NAME	SUBSCRIPTION	RESOURCE GROUP	SUBNET
DerekRGvnet691	Network Watcher Demo	DerekRG	
multiTierVNet	Network Watcher Demo	FlowLogsV2Demo	
MyCanaryFlowLo	Network Watcher Demo	MyCanaryFlowLog	
CMPreviewTest-vi	Network Traffic Analyti...	CMPreviewTest	
SourabhTestRG-v	Network Traffic Analyti...	SourabhTestRG	
juzERVnet1	Network Watcher Tea...	juz-er-networkwatcher...	
juz-er-networkwa	Network Watcher Tea...	juz-er-networkwatcher...	
juz-er-networkkwa	Network Watcher Tea...	juz-er-networkwatcher...	

NAME	IP	RESOURCE GROUP	VNET	SKU
Network Watcher Demo				
Network Traffic Analytics Subscription 3				
CMPreviewTest				
CMPreviewTest-vnet		CMPreviewTest	CMPreviewTest-vnet	
default		CMPreviewTest	CMPreviewTest-vnet	
CMPreviewVMEUSEUAP	10.1.2.5	CMPreviewTest	CMPreviewTest-vnet	
CMPreviewVM	10.1.2.4	CMPreviewTest	CMPreviewTest-vnet	
SourabhTestRG				

- To choose endpoints as destinations, select the **Endpoints** tab. The list of endpoints includes Office 365 test URLs and Dynamics 365 test URLs, grouped by name. In addition to these endpoints, you can choose an endpoint that was created in other test groups in the same connection monitor.

To add a new endpoint, in the upper-right corner, select + **Endpoints**. Then provide an endpoint name and URL, IP, or FQDN.

ENDPOINT NAME	ENDPOINTS	TYPE
EastUS_To_CentralUS		
Connectivity_To_Outlook		
Office 365-Office 365 portal and shared		
Office 365-Office 365 authentication a...		
Office 365-Office Online		
Office 365-Exchange Online		
Office 365-Skype for Business Online		
Office 365-Microsoft Teams		
teams.microsoft.com	teams.microsoft.com	Microsoft Teams
prod.registrar.skype.com	prod.registrar.skype.com	Microsoft Teams
prod.tpc.skype.com	prod.tpc.skype.com	Microsoft Teams

- To review the Azure VMs and endpoints that you chose, select the **Review** tab.
- When you finish choosing destinations, select **Done**.
- **Test configurations** – You can associate test configurations in a test group. The Azure portal allows only one test configuration per test group, but you can use ARMClient to add more.
 - **Name** – Name the test configuration.
 - **Protocol** – Choose TCP, ICMP, or HTTP. To change HTTP to HTTPS, select **HTTP** as the protocol, and select **443** as the port.
 - **Create network test configuration** – This check box appears only if you select **HTTP** in the **Protocol** field. Select this box to create another test configuration that uses the same sources and destinations that you specified elsewhere in your configuration. The newly created test configuration is named <the name of your test configuration>_networkTestConfig .
 - **Disable traceroute** – This field applies to test groups whose protocol is TCP or ICMP. Select this box to stop sources from discovering topology and hop-by-hop RTT.
 - **Destination port** – You can customize this field with a destination port of your choice.
 - **Test Frequency** – Use this field to choose how frequently sources will ping destinations on the protocol and port that you specified. You can choose 30 seconds, 1 minute, 5 minutes, 15 minutes, or 30 minutes. Sources will test connectivity to destinations based on the value that you choose. For example, if you select 30 seconds, sources will check connectivity to the destination at least once in a 30-second period.
 - **Success Threshold** – You can set thresholds on the following network parameters:
 - **Checks failed** – Set the percentage of checks that can fail when sources check connectivity to destinations by using the criteria that you specified. For TCP or ICMP protocol, the percentage of failed checks can be equated to the percentage of packet loss. For HTTP protocol, this field represents the percentage of HTTP requests that received no response.
 - **Round-trip time** – Set the RTT in milliseconds for how long sources can take to connect to the destination over the test configuration.

All sources, destinations, and test configurations that you add to a test group get broken down to individual tests. Here's an example of how sources and destinations are broken down:

- Test group: TG1
- Sources: 3 (A, B, C)
- Destinations: 2 (D, E)
- Test configurations: 2 (Config 1, Config 2)
- Total tests created: 12

TEST NUMBER	SOURCE	DESTINATION	TEST CONFIGURATION
1	A	D	Config 1
2	A	D	Config 2
3	A	E	Config 1
4	A	E	Config 2
5	B	D	Config 1
6	B	D	Config 2
7	B	E	Config 1
8	B	E	Config 2
9	C	D	Config 1
10	C	D	Config 2
11	C	E	Config 1
12	C	E	Config 2

Scale limits

Connection monitors have the following scale limits:

- Maximum connection monitors per subscription per region: 100
- Maximum test groups per connection monitor: 20
- Maximum sources and destinations per connection monitor: 100
- Maximum test configurations per connection monitor:
 - 20 via ARMClient
 - 2 via the Azure portal

Analyze monitoring data and set alerts

After you create a connection monitor, sources check connectivity to destinations based on your test configuration.

Checks in a test

Based on the protocol that you chose in the test configuration, Connection Monitor (Preview) runs a series of checks for the source-destination pair. The checks run according to the test frequency that you chose.

If you use HTTP, the service calculates the number of HTTP responses that returned a response code. The result determines the percentage of failed checks. To calculate RTT, the service measures the time between an HTTP call and the response.

If you use TCP or ICMP, the service calculates the packet-loss percentage to determine the percentage of failed checks. To calculate RTT, the service measures the time taken to receive the acknowledgment (ACK) for the packets that were sent. If you enabled traceroute data for your network tests, you can see hop-by-hop loss and latency for your on-premises network.

States of a test

Based on the data that the checks return, tests can have the following states:

- **Pass** – Actual values for the percentage of failed checks and RTT are within the specified thresholds.
- **Fail** – Actual values for the percentage of failed checks or RTT exceeded the specified thresholds. If no threshold is specified, then a test reaches the Fail state when the percentage of failed checks is 100.
- **Warning** – No criteria was specified for the percentage of failed checks. In the absence of specified criteria, Connection Monitor (Preview) automatically assigns a threshold. When that threshold is exceeded, the test status changes to Warning.

Data collection, analysis, and alerts

The data that Connection Monitor (Preview) collects is stored in the Log Analytics workspace. You set up this workspace when you created the connection monitor.

Monitoring data is also available in Azure Monitor Metrics. You can use Log Analytics to keep your monitoring data for as long as you want. Azure Monitor stores metrics for only 30 days by default.

You can [set metric-based alerts on the data](#).

Monitoring dashboards

On the monitoring dashboards, you see a list of the connection monitors that you can access for your subscriptions, regions, time stamps, sources, and destination types.

When you go to Connection Monitor (Preview) from Network Watcher, you can view data by:

- **Connection monitor** – List of all connection monitors created for your subscriptions, regions, time stamps, sources, and destination types. This view is the default.
- **Test groups** – List of all test groups created for your subscriptions, regions, time stamps, sources, and destination types. These test groups aren't filtered by connection monitors.

- **Test** – List of all tests that run for your subscriptions, regions, time stamps, sources, and destination types. These tests aren't filtered by connection monitors or test groups.

In the following image, the three data views are indicated by arrow 1.

On the dashboard, you can expand each connection monitor to see its test groups. Then you can expand each test group to see the tests that run in it.

You can filter a list based on:

- **Top-level filters** – Choose subscriptions, regions, time stamp sources, and destination types. See box 2 in the following image.
- **State-based filters** – Filter by the state of the connection monitor, test group, or test. See arrow 3 in the following image.
- **Custom filters** – Choose **Select all** to do a generic search. To search by a specific entity, select from the drop-down list. See arrow 4 in the following image.

TEST CONFIGURATIONS	STATUS	LAST POLLED
DemoNetworkMon	⚠	9/23/2019 11:31:04 AM
NetworkMonitoring	⚠	9/23/2019 11:31:04 AM
SampleInfraMonitoring	⚠	9/23/2019 11:31:03 AM
TestCM	⚠	9/23/2019 11:15:00 AM
TestCM	⚠	9/23/2019 11:31:00 AM
cmName123	✖	9/23/2019 11:30:55 AM
teapp	✖	9/23/2019 11:31:04 AM

For example, to look at all tests in Connection Monitor (Preview) where the source IP is 10.192.64.56:

1. Change the view to **Test**.
2. In the search field, type *10.192.64.56*
3. In the drop-down list, select **Sources**.

To show only failed tests in Connection Monitor (Preview) where the source IP is 10.192.64.56:

1. Change the view to **Test**.
2. For the state-based filter, select **Fail**.
3. In the search field, type *10.192.64.56*
4. In the drop-down list, select **Sources**.

To show only failed tests in Connection Monitor (Preview) where the destination is outlook.office365.com:

1. Change view to **Test**.
2. For the state-based filter, select **Fail**.

3. In the search field, enter *outlook.office365.com*

4. In the drop-down list, select **Destinations**.

The screenshot shows the Network Watcher - Connection monitor (Preview) interface. At the top, it displays 'Subscriptions: 2 of 35 selected – Don't see a subscription? Open Directory + Subscription settings'. Below this are dropdown menus for '2 subscriptions', '2 locations', 'All sources', 'All destinations', and a time range from 'Current Time (9/11/2019, 11:...)'.

The main area has tabs for 'Connection Monitor', 'Test groups', and 'Test'. The 'Test' tab is selected, showing a summary of connection status: Fail (8 out of 42), Warning (0 out of 42), Indeterminate (0 out of 42), Not Running (21 out of 42), and Pass (13 out of 42).

Below the summary is a search bar with 'outlook.office3...' and a 'Destinations' dropdown set to 'Fail'. A 'Applied filters' section shows 'Fail'.

The bottom part is a table with columns: SOURCES, DESTINATIONS, TEST CONFIGUR..., TEST GROUPS, CONNECTION M..., STATUS, LAST POLLED, ROUND TRIP TI..., and LOSS (%). It lists five entries:

SOURCES	DESTINATIONS	TEST CONFIGUR...	TEST GROUPS	CONNECTION M...	STATUS	LAST POLLED	ROUND TRIP TI...	LOSS (%)
CMPreviewVM(C...	EUS2EUAP(SOUR...	swatitest	test123	teapp	✗	9/11/2019 11:46:0...	-	100
CMPreviewVM(C...	account.office.net	swatitest	test123	teapp	✗	9/11/2019 11:45:4...	-	100
WIN-GTV0G9E2LCF	outlook.office36...	TC1	ConnectivityToO...	NetworkMonitori...	✗	9/11/2019 11:47:0...	51.63	0
WIN-8LGH868DJ...	outlook.office36...	TC1	ConnectivityToO...	NetworkMonitori...	✗	9/11/2019 11:47:0...	52.59	0
WIN-GTV0G9E2LCF	r1.res.office365.c...	TC1	TG1	TestCM	✗	9/11/2019 11:46:0...	138.65	0

To view the trends in RTT and the percentage of failed checks for a connection monitor:

1. Select the connection monitor that you want to investigate. By default, the monitoring data is organized by test group.

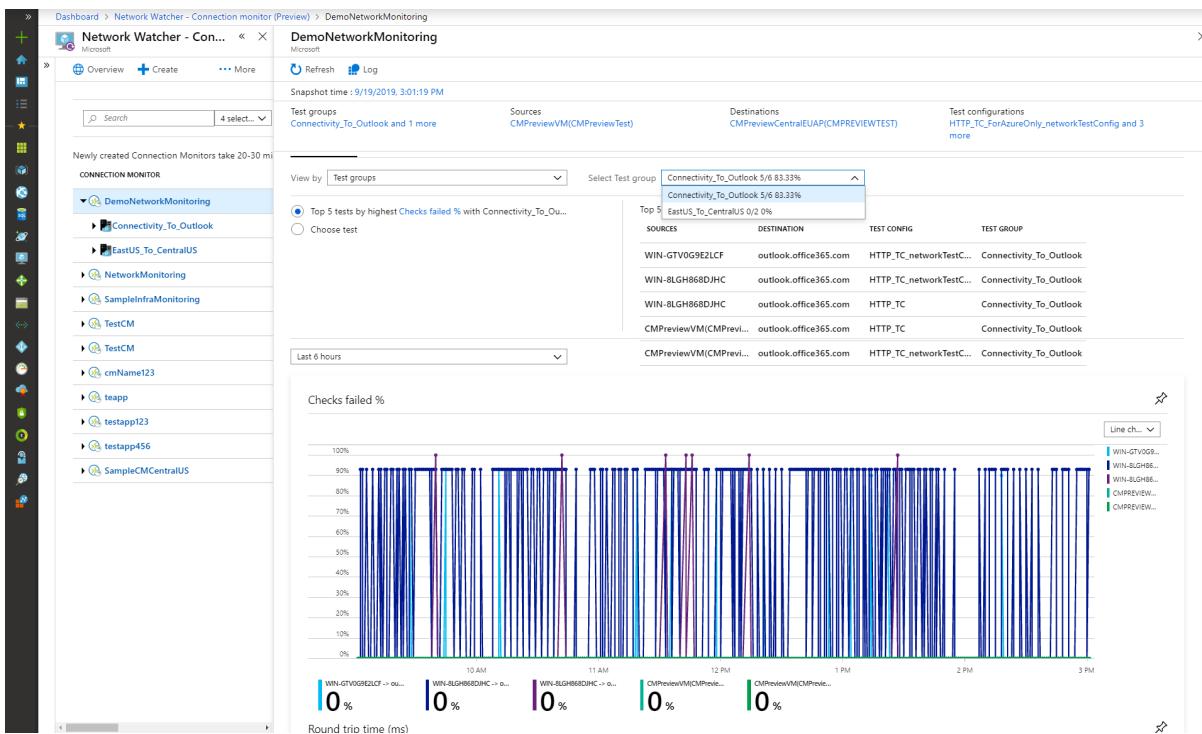
The screenshot shows the 'DemoNetworkMonitoring' connection monitor details. It includes sections for 'Sources' (CMPreviewVM(CMPREVIEWTEST)), 'Destinations' (CMPreviewCentralEUAP(CMPREVIEWTEST)), and 'Test configurations' (HTTP_TC_ForAzureOnly_networkTestConfig and 3 more).

The left sidebar shows a tree view of connection monitors, with 'DemoNetworkMonitoring' selected. The right side shows a 'View by' dropdown set to 'Test groups' and a 'Select Test group' dropdown set to 'Connectivity_To_Outlook 5/6 83.33%'. Below this is a table titled 'Top 5 tests' listing five entries:

SOURCES	DESTINATION	TEST CONFIG	TEST GROUP
WIN-GTV0G9E2LCF	outlook.office365.com	HTTP_TC_networkTestC...	Connectivity_To_Outlook
WIN-8LGH868DJHC	outlook.office365.com	HTTP_TC_networkTestC...	Connectivity_To_Outlook
WIN-8LGH868DJHC	outlook.office365.com	HTTP_TC	Connectivity_To_Outlook
CMPreviewVM(CMPREVI...	outlook.office365.com	HTTP_TC	Connectivity_To_Outlook
CMPreviewVM(CMPREVI...	outlook.office365.com	HTTP_TC_networkTestC...	Connectivity_To_Outlook

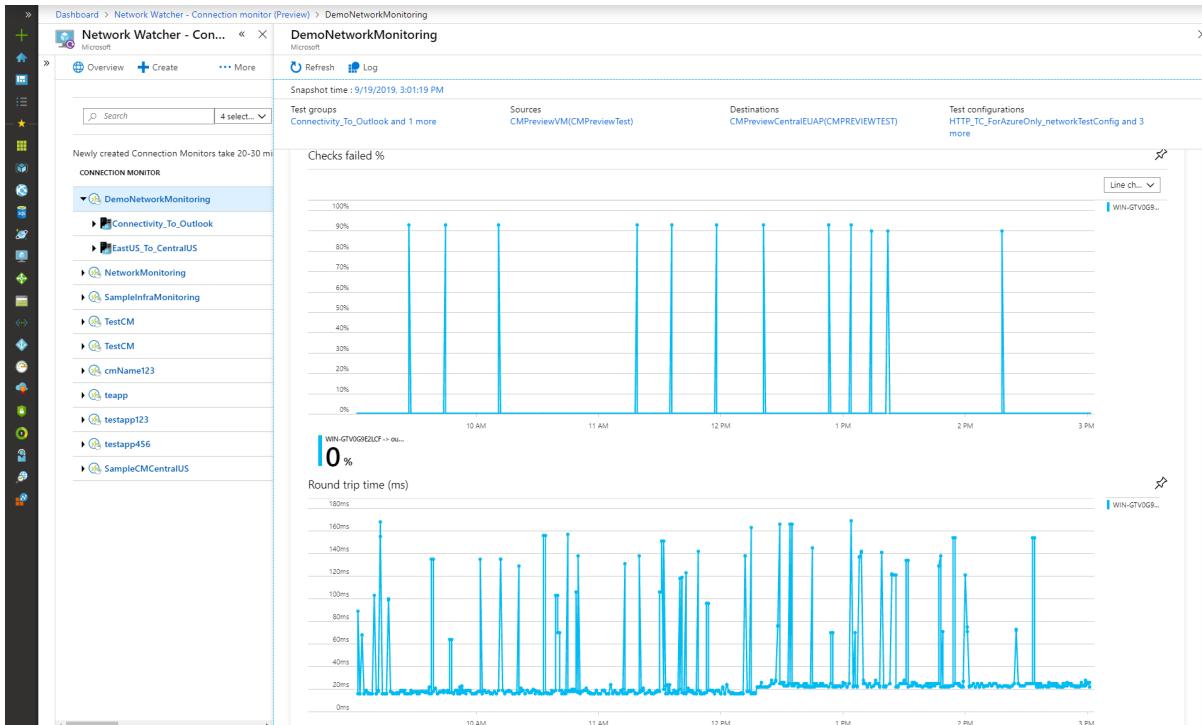
At the bottom is a chart titled 'Checks failed %' showing a histogram of failed checks over time. The x-axis represents time from 10 AM to 3 PM, and the y-axis represents the percentage of failed checks from 0% to 100%. The chart shows several vertical bars, each labeled with a source name and its corresponding failed percentage: WIN-GTV0G9E2LCF (0%), WIN-8LGH868DJHC (0%), WIN-8LGH868DJHC (0%), CMPreviewVM(CMPREVI... (0%), and CMPreviewVM(CMPREVI... (0%).

2. Choose the test group that you want to investigate.

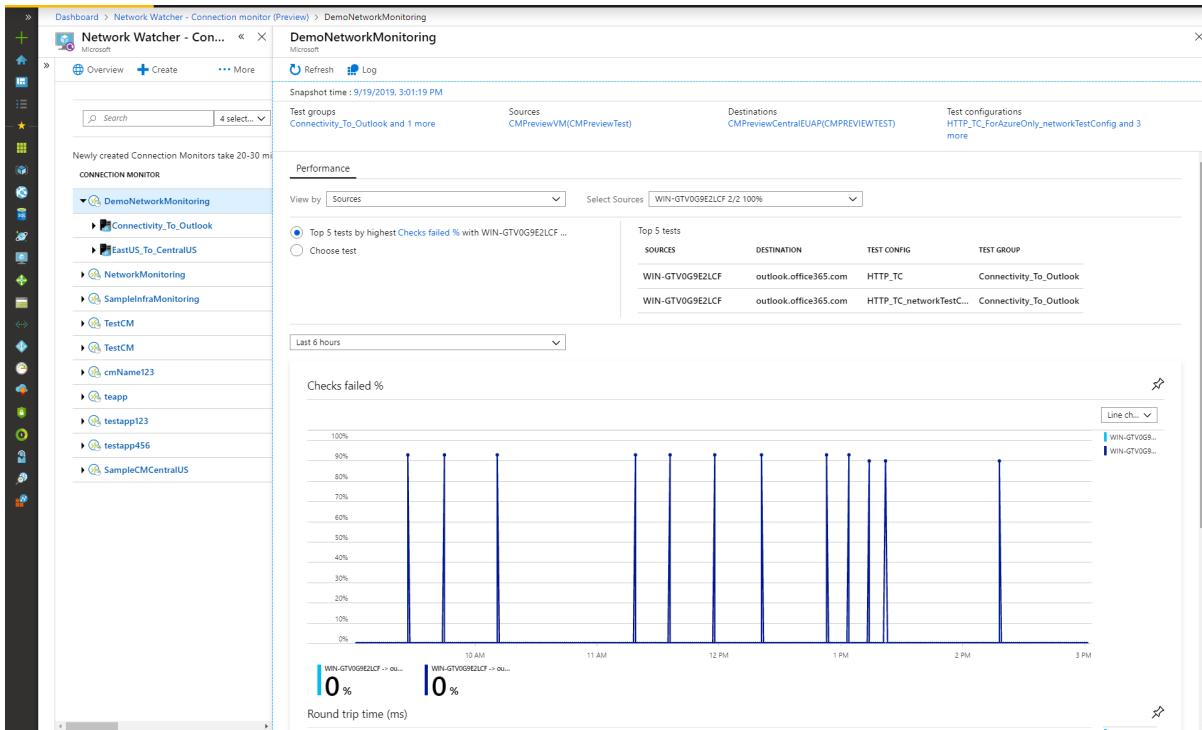


You see your test group's top five failed tests, based on the RTT or percentage of failed checks. For each test, you see the RTT and trend lines for the percentage of failed checks.

3. Select a test from the list, or choose another test to investigate. For your time interval and the percentage of failed checks, you see threshold and actual values. For RTT, you see the values for threshold, average, minimum, and maximum.



4. Change the time interval to view more data.
5. Change the view to see sources, destinations, or test configurations.
6. Choose a source based on failed tests, and investigate the top five failed tests. For example, choose **View by > Sources** and **View by > Destinations** to investigate the relevant tests in the connection monitor.

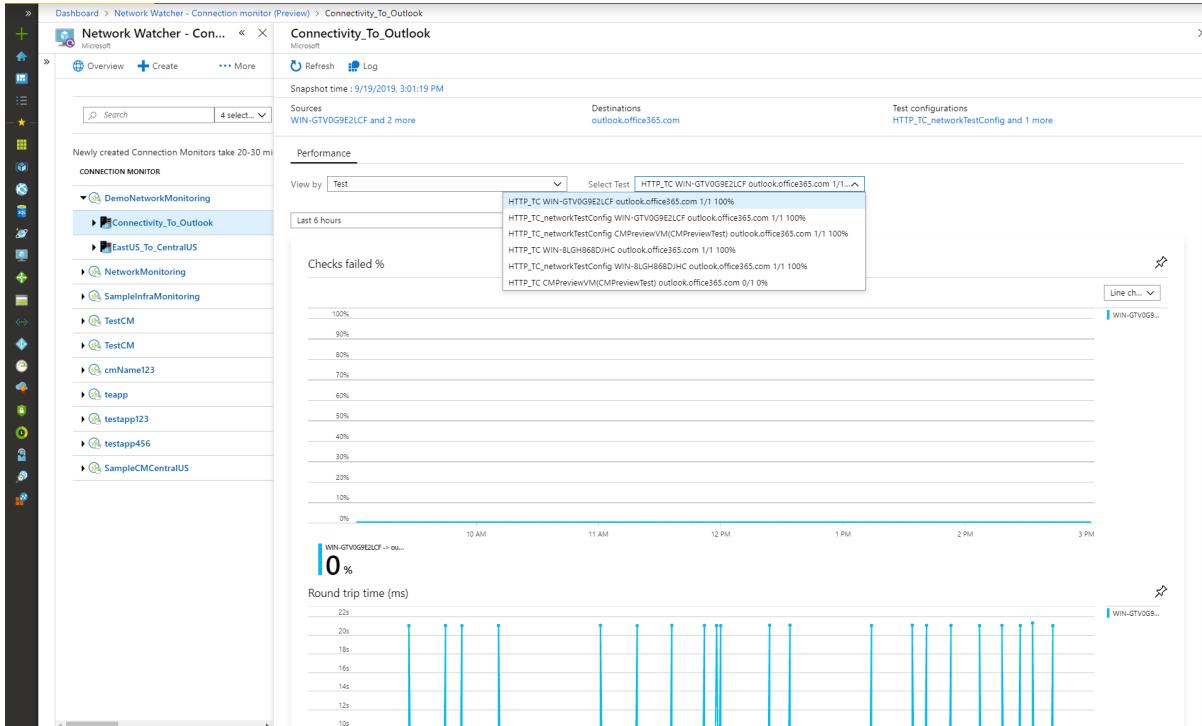


To view the trends in RTT and the percentage of failed checks for a test group:

1. Select the test group that you want to investigate.

By default, the monitoring data is arranged by sources, destinations, and test configurations (tests). Later, you can change the view from test groups to sources, destinations, or test configurations. Then choose an entity to investigate the top five failed tests. For example, change the view to sources and destinations to investigate the relevant tests in the selected connection monitor.

2. Choose the test that you want to investigate.



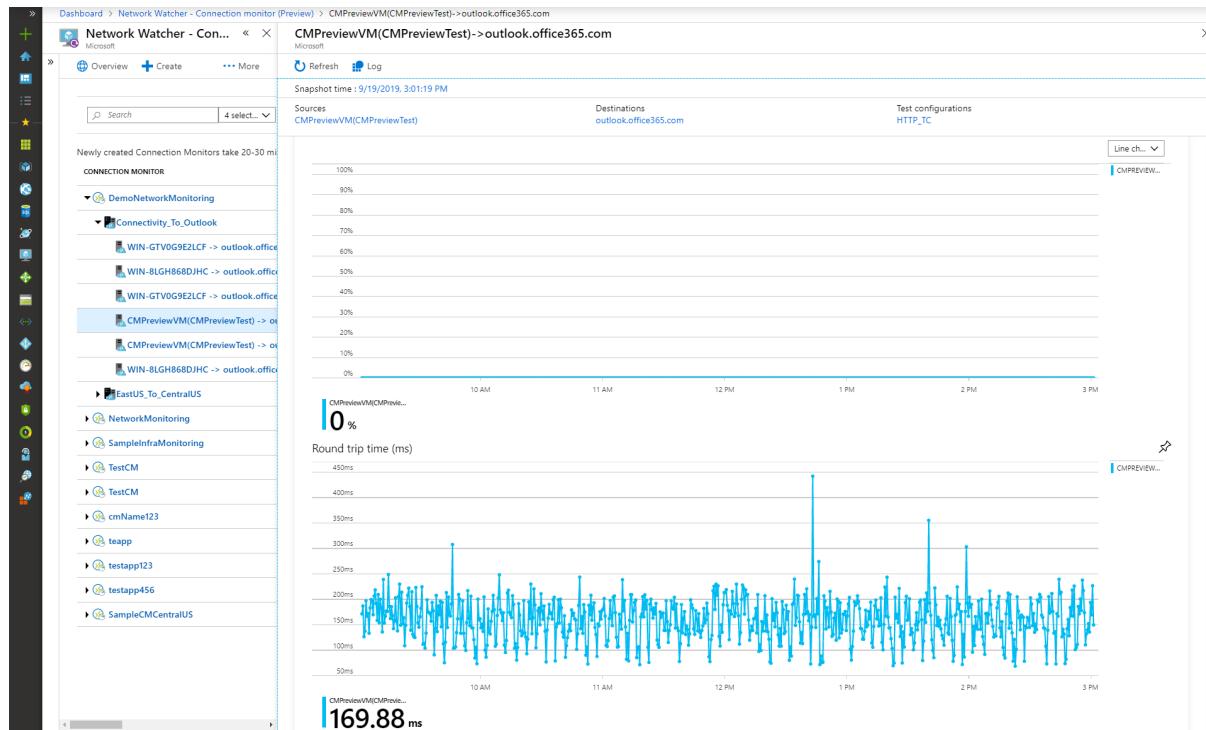
For your time interval and for your percentage of failed checks, you see threshold values and actual values. For RTT, you see values for threshold, average, minimum, and maximum. You also see fired alerts for the test that you selected.

3. Change the time interval to view more data.

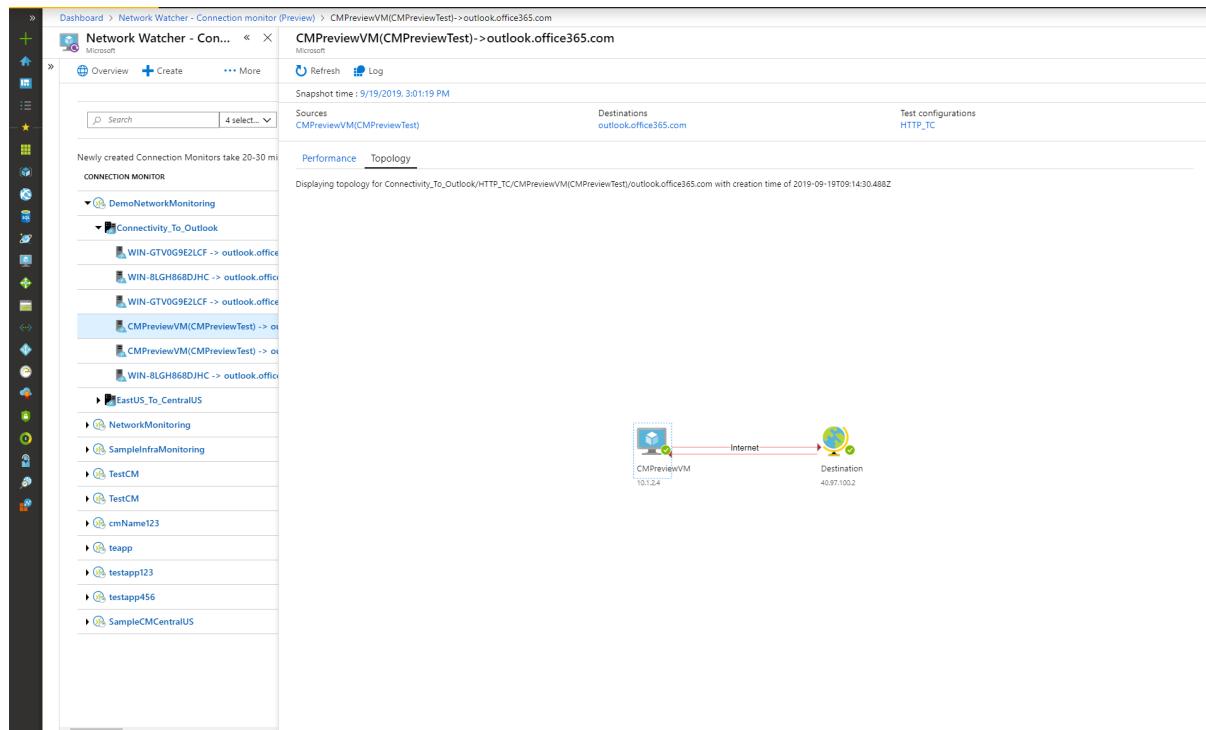
To view the trends in RTT and the percentage of failed checks for a test:

1. Select the source, destination, and test configuration that you want to investigate.

For your time interval and for the percentage of failed checks, you see threshold values and actual values. For RTT, you see values for threshold, average, minimum, and maximum. You also see fired alerts for the test that you selected.



2. To see the network topology, select **Topology**.



3. To see the identified issues, in the topology, select any hop in the path. (These hops are Azure resources.)
This functionality isn't currently available for on-premises networks.

The screenshot shows the Azure Network Watcher - Connection monitor (Preview) interface. On the left, there's a navigation pane with various monitoring options like Network Watcher, Network Monitoring, and Application Insights. The main area displays a tree view of connection monitors. A specific monitor, 'CMPreviewVM(CMPreviewTest) -> outlook.office365.com', is selected. Below it, a 'Performance' tab shows a topology diagram with nodes like 'CMPreviewVM(CMPreviewTest)', 'outlook.office365.com', and several intermediate hosts. A tooltip provides detailed information about one of the hops, including its IP address (10.1.2.4), next hop IP address (40.97.100.2), and issue status (No issues detected).

Log queries in Log Analytics

Use Log Analytics to create custom views of your monitoring data. All data that the UI displays is from Log Analytics. You can interactively analyze data in the repository. Correlate the data from Agent Health or other solutions that are based in Log Analytics. Export the data to Excel or Power BI, or create a shareable link.

Metrics in Azure Monitor

In connection monitors that were created before the Connection Monitor (Preview) experience, all four metrics are available: % Probes Failed, AverageRoundtripMs, ChecksFailedPercent (Preview), and RoundTripTimeMs (Preview). In connection monitors that were created in the Connection Monitor (Preview) experience, data is available only for the metrics that are tagged with (Preview).

The screenshot shows the Azure Monitor - Metrics interface. On the left, there's a sidebar with navigation links for Home, Monitor - Metrics, Activity log, Alerts, Metrics, Logs, Service Health, Workbooks (preview), Applications, Virtual Machines (preview), Storage Accounts (preview), Containers, Network, and More. The main area displays a line chart titled 'Avg Checks Failed Percent (Preview)' for the resource 'cmV2ira'. The chart shows the percentage of failed checks over time, with values ranging from 0% to 100%. A dropdown menu is open, showing properties like Destination address, Destination endpoint name, Destination port, Destination resource ID, Destination type, Protocol, Source address, and Source endpoint name, which can be used to filter the metric data.

When you use metrics, set the resource type as Microsoft.Network/networkWatchers/connectionMonitors

METRIC	DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION	DIMENSIONS
ProbesFailedPercent	% Probes Failed	Percentage	Average	Percentage of connectivity monitoring probes failed.	No dimensions

Metric	Display Name	Unit	Aggregation Type	Description	Dimensions
AverageRoundtripMs	Avg. Round-trip Time (ms)	Milliseconds	Average	Average network RTT for connectivity monitoring probes sent between source and destination.	No dimensions
ChecksFailedPercent (Preview)	% Checks Failed (Preview)	Percentage	Average	Percentage of failed checks for a test.	ConnectionMonitorResourceId SourceAddress SourceName SourceResourceId SourceType Protocol DestinationAddress DestinationName DestinationResourceId DestinationType DestinationPort TestGroupName TestConfigurationName Region
RoundTripTimeMs (Preview)	Round-trip Time (ms) (Preview)	Milliseconds	Average	RTT for checks sent between source and destination. This value isn't averaged.	ConnectionMonitorResourceId SourceAddress SourceName SourceResourceId SourceType Protocol DestinationAddress DestinationName DestinationResourceId DestinationType DestinationPort TestGroupName TestConfigurationName Region

Metric alerts in Azure Monitor

To create an alert in Azure Monitor:

1. Choose the connection monitor resource that you created in Connection Monitor (Preview).
2. Ensure that **Metric** shows up as signal type for the connection monitor.
3. In **Add Condition**, for the **Signal Name**, select **ChecksFailedPercent(Preview)** or **RoundTripTimeMs(Preview)**.
4. For **Signal Type**, choose **Metrics**. For example, select **ChecksFailedPercent(Preview)**.
5. All of the dimensions for the metric are listed. Choose the dimension name and dimension value. For

example, select **Source Address** and then enter the IP address of any source in your connection monitor.

6. In **Alert Logic**, fill in the following details:

- **Condition Type: Static.**
- **Condition and Threshold.**
- **Aggregation Granularity and Frequency of Evaluation:** Connection Monitor (Preview) updates data every minute.

7. In **Actions**, choose your action group.

8. Provide alert details.

9. Create the alert rule.

The screenshot shows the 'Create rule' interface in the Azure portal. On the left, there's a sidebar with 'Home > Monitor - Alerts > Create rule'. The main area has sections for 'RESOURCE' (NetworkWatcher_centraluseuap/ir1), 'HIERARCHY' (Network Watcher Slice Test), 'CONDITION' (No condition defined, click on 'Add condition' to select a signal and define its logic), 'ACTIONS' (No configured actions, Add), and 'ALERT DETAILS' (Alert rule name: Percentage CPU greater than 70%, Description: Specify alert description here..., Enable rule upon creation: Yes). On the right, the 'Configure signal logic' section is open, showing a timeline from 8 am to 1 pm with a single metric selected: 'Checks Failed Percent (Preview) (Avg) networkwatcher_centraluseuap/ir1'. Below the timeline, a note says: 'This metric supports dimensions. Selecting the dimension values will help you filter to the right time series. If you do not select any value for a dimension, that dimension will be ignored.' A table lists dimensions and their current state (0 selected) with 'SELECT' checkboxes. Dimensions include Source address, Source endpoint name, Source resource ID, Source type, Protocol, Destination address, Destination endpoint name, Destination resource ID, Destination type, and Destination port.

Diagnose issues in your network

Connection Monitor (Preview) helps you diagnose issues in your connection monitor and your network. Issues in your hybrid network are detected by the Log Analytics agents that you installed earlier. Issues in Azure are detected by the Network Watcher extension.

You can view issues in the Azure network in the network topology.

For networks whose sources are on-premises VMs, the following issues can be detected:

- Request timed out.
- Endpoint not resolved by DNS – temporary or persistent. URL invalid.
- No hosts found.
- Source unable to connect to destination. Target not reachable through ICMP.
- Certificate-related issues:
 - Client certificate required to authenticate agent.
 - Certificate relocation list isn't accessible.
 - Host name of the endpoint doesn't match the certificate's subject or subject alternate name.
 - Root certificate is missing in source's Local Computer Trusted Certification Authorities store.
 - SSL certificate is expired, invalid, revoked, or incompatible.

For networks whose sources are Azure VMs, the following issues can be detected:

- Agent issues:

- Agent stopped.
- Failed DNS resolution.
- No application or listener listening on the destination port.
- Socket could not be opened.
- VM state issues:
 - Starting
 - Stopping
 - Stopped
 - Deallocating
 - Deallocated
 - Rebooting
 - Not allocated
- ARP table entry is missing.
- Traffic was blocked because of local firewall issues or NSG rules.
- Virtual network gateway issues:
 - Missing routes.
 - The tunnel between two gateways is disconnected or missing.
 - The second gateway wasn't found by the tunnel.
 - No peering info was found.
- Route was missing in Microsoft Edge.
- Traffic stopped because of system routes or UDR.
- BGP isn't enabled on the gateway connection.
- The DIP probe is down at the load balancer.

Quickstart: Diagnose a virtual machine network traffic filter problem using the Azure portal

1/28/2020 • 6 minutes to read • [Edit Online](#)

In this quickstart, you deploy a virtual machine (VM), and then check communications to an IP address and URL and from an IP address. You determine the cause of a communication failure and how you can resolve it.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Log in to Azure

Log in to the Azure portal at <https://portal.azure.com>.

Create a VM

1. Select **+ Create a resource** found on the upper, left corner of the Azure portal.
2. Select **Compute**, and then select **Windows Server 2016 Datacenter** or a version of **Ubuntu Server**.
3. Enter, or select, the following information, accept the defaults for the remaining settings, and then select **OK**:

SETTING	VALUE
Name	myVm
User name	Enter a user name of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the defined complexity requirements .
Subscription	Select your subscription.
Resource group	Select Create new and enter myResourceGroup .
Location	Select East US

4. Select a size for the VM and then select **Select**.
5. Under **Settings**, accept all the defaults, and select **OK**.
6. Under **Create** of the **Summary**, select **Create** to start VM deployment. The VM takes a few minutes to deploy. Wait for the VM to finish deploying before continuing with the remaining steps.

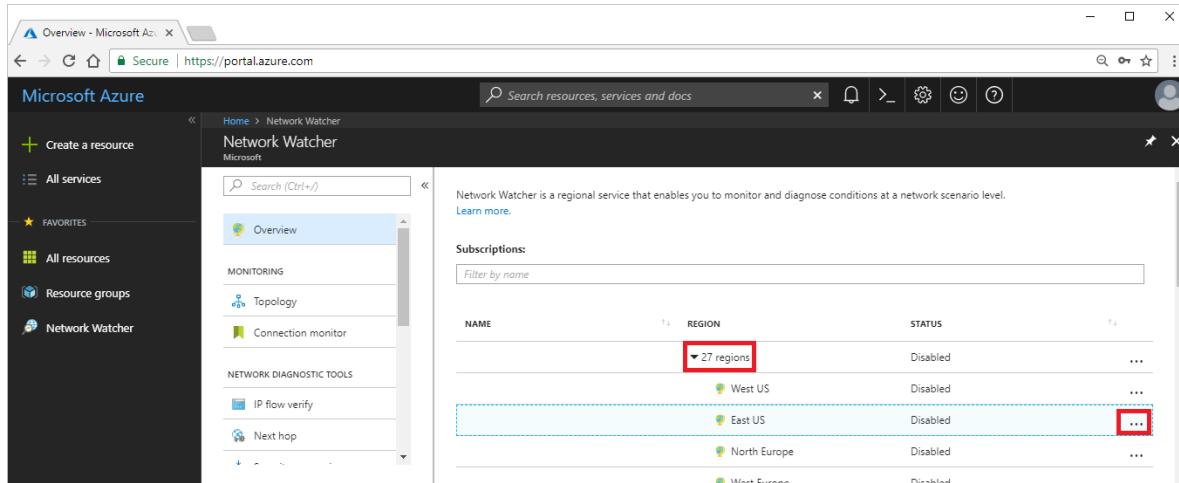
Test network communication

To test network communication with Network Watcher, first enable a network watcher in at least one Azure region, and then use Network Watcher's IP flow verify capability.

Enable network watcher

If you already have a network watcher enabled in at least one region, skip to [Use IP flow verify](#).

1. In the portal, select **All services**. In the **Filter box**, enter *Network Watcher*. When **Network Watcher** appears in the results, select it.
2. Enable a network watcher in the East US region, because that's the region the VM was deployed to in a previous step. Select **Regions**, to expand it, and then select ... to the right of **East US**, as shown in the following picture:



3. Select **Enable Network Watcher**.

Use IP flow verify

When you create a VM, Azure allows and denies network traffic to and from the VM, by default. You might later override Azure's defaults, allowing or denying additional types of traffic.

1. In the portal, select **All services**. In the **All services Filter box**, enter *Network Watcher*. When **Network Watcher** appears in the results, select it.
2. Select **IP flow verify**, under **NETWORK DIAGNOSTIC TOOLS**.
3. Select your subscription, enter or select the following values, and then select **Check**, as shown in the picture that follows:

SETTING	VALUE
Resource group	Select myResourceGroup
Virtual machine	Select myVm
Network interface	myvm - The name of the network interface the portal created when you created the VM is different.
Protocol	TCP
Direction	Outbound
Local IP address	10.0.0.4
Local port	60000
Remote IP address	13.107.21.200 - One of the addresses for <www.bing.com>.

SETTING	VALUE
Remote port	80

Network Watcher - IP flow verify

Subscription*: myResourceGroup

Virtual machine*: myVm

Network interface*: myvm

Protocol: TCP

Direction: Outbound

Local IP address*: 10.0.0.4

Local port*: 60000

Remote IP address*: 13.107.21.200

Remote port*: 80

Check

After a few seconds, the result returned informs you that access is allowed because of a security rule named **AllowInternetOutbound**. When you ran the check, Network Watcher automatically created a network watcher in the East US region, if you had an existing network watcher in a region other than the East US region before you ran the check.

- Complete step 3 again, but change the **Remote IP address** to **172.31.0.100**. The result returned informs you that access is denied because of a security rule named **DefaultOutboundDenyAll**.
- Complete step 3 again, but change the **Direction** to **Inbound**, the **Local port** to **80** and the **Remote port** to **60000**. The result returned informs you that access is denied because of a security rule named **DefaultInboundDenyAll**.

Now that you know which security rules are allowing or denying traffic to or from a VM, you can determine how to resolve the problems.

View details of a security rule

- To determine why the rules in steps 3-5 of [Use IP flow verify](#) allow or deny communication, review the effective security rules for the network interface in the VM. In the search box at the top of the portal, enter **myvm**. When the **myvm** (or whatever the name of your network interface is) network interface appears in the search results, select it.

2. Select **Effective security rules** under **SUPPORT + TROUBLESHOOTING**, as shown in the following picture:

NAME	PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
default-allow-rdp	1000	0.0.0.0/0	0-65535	0.0.0.0/0	3389-3389	TCP	Allow
AllowVnetInBound	65000	Virtual network (2 prefixes)	0-65535	Virtual network (2 prefixes)	0-65535	All	Allow
AllowAzureLoadBalancerInBound	65001	Azure load balancer (1 prefixes)	0-65535	0.0.0.0/0	0-65535	All	Allow
DenyAllInBound	65500	0.0.0.0/0	0-65535	0.0.0.0/0	0-65535	All	Deny

NAME	PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
AllowVnetOutBound	65000	Virtual network (2 prefixes)	0-65535	Virtual network (2 prefixes)	0-65535	All	Allow
AllowInternetOutBound	65001	0.0.0.0/0	0-65535	Internet (82 prefixes)	0-65535	All	Allow
DenyAllOutBound	65500	0.0.0.0/0	0-65535	0.0.0.0/0	0-65535	All	Deny

In step 3 of [Use IP flow verify](#), you learned that the reason the communication was allowed is because of the **AllowInternetOutbound** rule. You can see in the previous picture that the **DESTINATION** for the rule is **Internet**. It's not clear how 13.107.21.200, the address you tested in step 3 of [Use IP flow verify](#), relates to **Internet** though.

3. Select the **AllowInternetOutBound** rule, and then select **Destination**, as shown in the following picture:

Source	Destination
1.0.0.0/8	
2.0.0.0/7	
4.0.0.0/6	
8.0.0.0/7	
11.0.0.0/8	
12.0.0.0/6	
16.0.0.0/5	
24.0.0.0/8	
25.25.0.0/16	
25.26.0.0/15	
25.132.0.0/14	
25.136.0.0/14	

One of the prefixes in the list is **12.0.0.0/6**, which encompasses the 12.0.0.1-15.255.255.254 range of IP addresses. Since 13.107.21.200 is within that address range, the **AllowInternetOutBound** rule allows the outbound traffic. Additionally, there are no higher priority (lower number) rules shown in the picture in step 2 that override this rule. Close the **Address prefixes** box. To deny outbound communication to 13.107.21.200, you could add a security rule with a higher priority, that denies port 80 outbound to the IP address.

4. When you ran the outbound check to 172.131.0.100 in step 4 of [Use IP flow verify](#), you learned that the **DefaultOutboundDenyAll** rule denied communication. That rule equates to the **DenyAllOutBound**

rule shown in the picture in step 2 that specifies **0.0.0.0/0** as the **DESTINATION**. This rule denies the outbound communication to 172.131.0.100, because the address is not within the **DESTINATION** of any of the other **Outbound rules** shown in the picture. To allow the outbound communication, you could add a security rule with a higher priority, that allows outbound traffic to port 80 for the 172.131.0.100 address.

5. When you ran the inbound check from 172.131.0.100 in step 5 of [Use IP flow verify](#), you learned that the **DefaultInboundDenyAll** rule denied communication. That rule equates to the **DenyAllInBound** rule shown in the picture in step 2. The **DenyAllInBound** rule is enforced because no other higher priority rule exists that allows port 80 inbound to the VM from 172.31.0.100. To allow the inbound communication, you could add a security rule with a higher priority, that allows port 80 inbound from 172.31.0.100.

The checks in this quickstart tested Azure configuration. If the checks return expected results and you still have network problems, ensure that you don't have a firewall between your VM and the endpoint you're communicating with and that the operating system in your VM doesn't have a firewall that is allowing or denying communication.

Clean up resources

When no longer needed, delete the resource group and all of the resources it contains:

1. Enter *myResourceGroup* in the **Search** box at the top of the portal. When you see **myResourceGroup** in the search results, select it.
2. Select **Delete resource group**.
3. Enter *myResourceGroup* for **TYPE THE RESOURCE GROUP NAME:** and select **Delete**.

Next steps

In this quickstart, you created a VM and diagnosed inbound and outbound network traffic filters. You learned that network security group rules allow or deny traffic to and from a VM. Learn more about [security rules](#) and how to [create security rules](#).

Even with the proper network traffic filters in place, communication to a VM can still fail, due to routing configuration. To learn how to diagnose VM network routing problems, see [Diagnose VM routing problems](#) or, to diagnose outbound routing, latency, and traffic filtering problems, with one tool, see [Connection troubleshoot](#).

Quickstart: Diagnose a virtual machine network traffic filter problem - Azure PowerShell

1/28/2020 • 7 minutes to read • [Edit Online](#)

In this quickstart, you deploy a virtual machine (VM), and then check communications to an IP address and URL and from an IP address. You determine the cause of a communication failure and how you can resolve it.

If you don't have an Azure subscription, create a [free account](#) before you begin.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this quickstart requires the Azure PowerShell `Az` module. To find the installed version, run `Get-Module -ListAvailable Az`. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

Create a VM

Before you can create a VM, you must create a resource group to contain the VM. Create a resource group with [New-AzResourceGroup](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location.

```
New-AzResourceGroup -Name myResourceGroup -Location EastUS
```

Create the VM with [New-AzVm](#). When running this step, you are prompted for credentials. The values that you enter are configured as the user name and password for the VM.

```
$vM = New-AzVm `  
-ResourceGroupName "myResourceGroup" `  
-Name "myVm" `  
-Location "East US"
```

The VM takes a few minutes to create. Don't continue with remaining steps until the VM is created and PowerShell returns output.

Test network communication

To test network communication with Network Watcher, you must first enable a network watcher in the region the VM that you want to test is in, and then use Network Watcher's IP flow verify capability to test communication.

Enable network watcher

If you already have a network watcher enabled in the East US region, use [Get-AzNetworkWatcher](#) to retrieve the network watcher. The following example retrieves an existing network watcher named *NetworkWatcher_eastus* that is in the *NetworkWatcherRG* resource group:

```
$networkWatcher = Get-AzNetworkWatcher `  
-Name NetworkWatcher_eastus `  
-ResourceGroupName NetworkWatcherRG
```

If you don't already have a network watcher enabled in the East US region, use [New-AzNetworkWatcher](#) to create a network watcher in the East US region:

```
$networkWatcher = New-AzNetworkWatcher `  
-Name "NetworkWatcher_eastus" `  
-ResourceGroupName "NetworkWatcherRG" `  
-Location "East US"
```

Use IP flow verify

When you create a VM, Azure allows and denies network traffic to and from the VM, by default. You might later override Azure's defaults, allowing or denying additional types of traffic. To test whether traffic is allowed or denied to different destinations and from a source IP address, use the [Test-AzNetworkWatcherIPFlow](#) command.

Test outbound communication from the VM to one of the IP addresses for www.bing.com:

```
Test-AzNetworkWatcherIPFlow ` 
-NetworkWatcher $networkWatcher ` 
-TargetVirtualMachineId $vM.Id ` 
-Direction Outbound ` 
-Protocol TCP ` 
-LocalIPAddress 192.168.1.4 ` 
-LocalPort 60000 ` 
-RemoteIPAddress 13.107.21.200 ` 
-RemotePort 80
```

After several seconds, the result returned informs you that access is allowed by a security rule named **AllowInternetOutbound**.

Test outbound communication from the VM to 172.31.0.100:

```
Test-AzNetworkWatcherIPFlow ` 
-NetworkWatcher $networkWatcher ` 
-TargetVirtualMachineId $vM.Id ` 
-Direction Outbound ` 
-Protocol TCP ` 
-LocalIPAddress 192.168.1.4 ` 
-LocalPort 60000 ` 
-RemoteIPAddress 172.31.0.100 ` 
-RemotePort 80
```

The result returned informs you that access is denied by a security rule named **DefaultOutboundDenyAll**.

Test inbound communication to the VM from 172.31.0.100:

```
Test-AzNetworkWatcherIPFlow ` 
-NetworkWatcher $networkWatcher ` 
-TargetVirtualMachineId $vM.Id ` 
-Direction Inbound ` 
-Protocol TCP ` 
-LocalIPAddress 192.168.1.4 ` 
-LocalPort 80 ` 
-RemoteIPAddress 172.31.0.100 ` 
-RemotePort 60000
```

The result returned informs you that access is denied because of a security rule named **DefaultInboundDenyAll**.

Now that you know which security rules are allowing or denying traffic to or from a VM, you can determine how to resolve the problems.

View details of a security rule

To determine why the rules in [Test network communication](#) are allowing or preventing communication, review the effective security rules for the network interface with [Get-AzEffectiveNetworkSecurityGroup](#):

```
Get-AzEffectiveNetworkSecurityGroup ` 
-NetworkInterfaceName myVm ` 
-ResourceGroupName myResourceGroup
```

The returned output includes the following text for the **AllowInternetOutbound** rule that allowed outbound access to www.bing.com in [Use IP flow verify](#):

```
{
  "Name": "defaultSecurityRules/AllowInternetOutBound",
  "Protocol": "All",
  "SourcePortRange": [
    "0-65535"
  ],
  "DestinationPortRange": [
    "0-65535"
  ],
  "SourceAddressPrefix": [
    "0.0.0.0/0"
  ],
  "DestinationAddressPrefix": [
    "Internet"
  ],
  "ExpandedSourceAddressPrefix": [],
  "ExpandedDestinationAddressPrefix": [
    "1.0.0.0/8",
    "2.0.0.0/7",
    "4.0.0.0/6",
    "8.0.0.0/7",
    "11.0.0.0/8",
    "12.0.0.0/6",
    ...
  ],
  "Access": "Allow",
  "Priority": 65001,
  "Direction": "Outbound"
},
}
```

You can see in the output that **DestinationAddressPrefix** is **Internet**. It's not clear how 13.107.21.200, the address you tested in [Use IP flow verify](#), relates to **Internet** though. You see several address prefixes listed under **ExpandedDestinationAddressPrefix**. One of the prefixes in the list is **12.0.0.0/6**, which encompasses the 12.0.0.1-15.255.255.254 range of IP addresses. Since 13.107.21.200 is within that address range, the **AllowInternetOutBound** rule allows the outbound traffic. Additionally, there are no higher **priority** (lower number) rules listed in the output returned by `Get-AzEffectiveNetworkSecurityGroup`, that override this rule. To deny outbound communication to 13.107.21.200, you could add a security rule with a higher priority, that denies port 80 outbound to the IP address.

When you ran the `Test-AzNetworkWatcherIPFlow` command to test outbound communication to 172.131.0.100 in [Use IP flow verify](#), the output informed you that the **DefaultOutboundDenyAll** rule denied the communication. The **DefaultOutboundDenyAll** rule equates to the **DenyAllOutBound** rule listed in the following output from the `Get-AzEffectiveNetworkSecurityGroup` command:

```
{
  "Name": "defaultSecurityRules/DenyAllOutBound",
  "Protocol": "All",
  "SourcePortRange": [
    "0-65535"
  ],
  "DestinationPortRange": [
    "0-65535"
  ],
  "SourceAddressPrefix": [
    "0.0.0.0/0"
  ],
  "DestinationAddressPrefix": [
    "0.0.0.0/0"
  ],
  "ExpandedSourceAddressPrefix": [],
  "ExpandedDestinationAddressPrefix": [],
  "Access": "Deny",
  "Priority": 65500,
  "Direction": "Outbound"
}
```

The rule lists **0.0.0.0/0** as the **DestinationAddressPrefix**. The rule denies the outbound communication to 172.131.0.100, because the address is not within the **DestinationAddressPrefix** of any of the other outbound rules in the output from the `Get-AzEffectiveNetworkSecurityGroup` command. To allow the outbound communication, you could add a security rule with a higher priority, that allows outbound traffic to port 80 at 172.131.0.100.

When you ran the `Test-AzNetworkWatcherIPFlow` command to test inbound communication from 172.131.0.100 in [Use IP flow verify](#), the output informed you that the **DefaultInboundDenyAll** rule denied the communication. The **DefaultInboundDenyAll** rule equates to the **DenyAllInBound** rule listed in the following output from the `Get-AzEffectiveNetworkSecurityGroup` command:

```
{
  "Name": "defaultSecurityRules/DenyAllInBound",
  "Protocol": "All",
  "SourcePortRange": [
    "0-65535"
  ],
  "DestinationPortRange": [
    "0-65535"
  ],
  "SourceAddressPrefix": [
    "0.0.0.0/0"
  ],
  "DestinationAddressPrefix": [
    "0.0.0.0/0"
  ],
  "ExpandedSourceAddressPrefix": [],
  "ExpandedDestinationAddressPrefix": [],
  "Access": "Deny",
  "Priority": 65500,
  "Direction": "Inbound"
},
```

The **DenyAllInBound** rule is applied because, as shown in the output, no other higher priority rule exists in the output from the `Get-AzEffectiveNetworkSecurityGroup` command that allows port 80 inbound to the VM from 172.131.0.100. To allow the inbound communication, you could add a security rule with a higher priority that allows port 80 inbound from 172.131.0.100.

The checks in this quickstart tested Azure configuration. If the checks return expected results and you still have

network problems, ensure that you don't have a firewall between your VM and the endpoint you're communicating with and that the operating system in your VM doesn't have a firewall that is allowing or denying communication.

Clean up resources

When no longer needed, you can use [Remove-AzResourceGroup](#) to remove the resource group and all of the resources it contains:

```
Remove-AzResourceGroup -Name myResourceGroup -Force
```

Next steps

In this quickstart, you created a VM and diagnosed inbound and outbound network traffic filters. You learned that network security group rules allow or deny traffic to and from a VM. Learn more about [security rules](#) and how to [create security rules](#).

Even with the proper network traffic filters in place, communication to a VM can still fail, due to routing configuration. To learn how to diagnose VM network routing problems, see [Diagnose VM routing problems](#) or, to diagnose outbound routing, latency, and traffic filtering problems, with one tool, see [Connection troubleshoot](#).

Quickstart: Diagnose a virtual machine network traffic filter problem - Azure CLI

11/20/2019 • 7 minutes to read • [Edit Online](#)

In this quickstart you deploy a virtual machine (VM), and then check communications to an IP address and URL and from an IP address. You determine the cause of a communication failure and how you can resolve it.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this quickstart requires that you are running the Azure CLI version 2.0.28 or later. To find the installed version, run `az --version`. If you need to install or upgrade, see [Install the Azure CLI](#). After you verify the CLI version, run `az login` to create a connection with Azure. The CLI commands in this quickstart are formatted to run in a Bash shell.

Create a VM

Before you can create a VM, you must create a resource group to contain the VM. Create a resource group with `az group create`. The following example creates a resource group named `myResourceGroup` in the `eastus` location:

```
az group create --name myResourceGroup --location eastus
```

Create a VM with [az vm create](#). If SSH keys do not already exist in a default key location, the command creates them. To use a specific set of keys, use the `--ssh-key-value` option. The following example creates a VM named *myVm*:

```
az vm create \
--resource-group myResourceGroup \
--name myVm \
--image UbuntuLTS \
--generate-ssh-keys
```

The VM takes a few minutes to create. Don't continue with remaining steps until the VM is created and the CLI returns output.

Test network communication

To test network communication with Network Watcher, you must first enable a network watcher in the region the VM that you want to test is in, and then use Network Watcher's IP flow verify capability to test communication.

Enable network watcher

If you already have a network watcher enabled in the East US region, skip to [Use IP flow verify](#). Use the [az network watcher configure](#) command to create a network watcher in the EastUS region:

```
az network watcher configure \
--resource-group NetworkWatcherRG \
--locations eastus \
--enabled
```

Use IP flow verify

When you create a VM, Azure allows and denies network traffic to and from the VM, by default. You might later override Azure's defaults, allowing or denying additional types of traffic. To test whether traffic is allowed or denied to different destinations and from a source IP address, use the [az network watcher test-ip-flow](#) command.

Test outbound communication from the VM to one of the IP addresses for www.bing.com:

```
az network watcher test-ip-flow \
--direction outbound \
--local 10.0.0.4:60000 \
--protocol TCP \
--remote 13.107.21.200:80 \
--vm myVm \
--nic myVmVMNic \
--resource-group myResourceGroup \
--out table
```

After several seconds, the result returned informs you that access is allowed by a security rule named **AllowInternetOutbound**.

Test outbound communication from the VM to 172.31.0.100:

```
az network watcher test-ip-flow \
--direction outbound \
--local 10.0.0.4:60000 \
--protocol TCP \
--remote 172.31.0.100:80 \
--vm myVm \
--nic myVmVMNic \
--resource-group myResourceGroup \
--out table
```

The result returned informs you that access is denied by a security rule named **DefaultOutboundDenyAll**.

Test inbound communication to the VM from 172.31.0.100:

```
az network watcher test-ip-flow \
--direction inbound \
--local 10.0.0.4:80 \
--protocol TCP \
--remote 172.31.0.100:60000 \
--vm myVm \
--nic myVmVMNic \
--resource-group myResourceGroup \
--out table
```

The result returned informs you that access is denied because of a security rule named **DefaultInboundDenyAll**.

Now that you know which security rules are allowing or denying traffic to or from a VM, you can determine how to resolve the problems.

View details of a security rule

To determine why the rules in [Use IP flow verify](#) are allowing or preventing communication, review the effective security rules for the network interface with the [az network nic list-effective-nsg](#) command:

```
az network nic list-effective-nsg \
--resource-group myResourceGroup \
--name myVmVMNic
```

The returned output includes the following text for the **AllowInternetOutbound** rule that allowed outbound access to [www.bing.com](#) in a previous step under [Use IP flow verify](#):

```
{
  "access": "Allow",
  "additionalProperties": {},
  "destinationAddressPrefix": "Internet",
  "destinationAddressPrefixes": [
    "Internet"
  ],
  "destinationPortRange": "0-65535",
  "destinationPortRanges": [
    "0-65535"
  ],
  "direction": "Outbound",
  "expandedDestinationAddressPrefix": [
    "1.0.0.0/8",
    "2.0.0.0/7",
    "4.0.0.0/6",
    "8.0.0.0/7",
    "11.0.0.0/8",
    "12.0.0.0/6",
    ...
  ],
  "expandedSourceAddressPrefix": null,
  "name": "defaultSecurityRules/AllowInternetOutBound",
  "priority": 65001,
  "protocol": "All",
  "sourceAddressPrefix": "0.0.0.0/0",
  "sourceAddressPrefixes": [
    "0.0.0.0/0"
  ],
  "sourcePortRange": "0-65535",
  "sourcePortRanges": [
    "0-65535"
  ]
},
}
```

You can see in the previous output that **destinationAddressPrefix** is **Internet**. It's not clear how 13.107.21.200 relates to **Internet** though. You see several address prefixes listed under **expandedDestinationAddressPrefix**. One of the prefixes in the list is **12.0.0.0/6**, which encompasses the 12.0.0.1-15.255.255.254 range of IP addresses. Since 13.107.21.200 is within that address range, the **AllowInternetOutBound** rule allows the outbound traffic. Additionally, there are no higher priority (lower number) rules shown in the previous output that override this rule. To deny outbound communication to an IP address, you could add a security rule with a higher priority, that denies port 80 outbound to the IP address.

When you ran the `az network watcher test-ip-flow` command to test outbound communication to 172.131.0.100 in [Use IP flow verify](#), the output informed you that the **DefaultOutboundDenyAll** rule denied the communication. The **DefaultOutboundDenyAll** rule equates to the **DenyAllOutBound** rule listed in the following output from the `az network nic list-effective-nsg` command:

```
{  
  "access": "Deny",  
  "additionalProperties": {},  
  "destinationAddressPrefix": "0.0.0.0/0",  
  "destinationAddressPrefixes": [  
    "0.0.0.0/0"  
,  
    "destinationPortRange": "0-65535",  
    "destinationPortRanges": [  
      "0-65535"  
,  
      "direction": "Outbound",  
      "expandedDestinationAddressPrefix": null,  
      "expandedSourceAddressPrefix": null,  
      "name": "defaultSecurityRules/DenyAllOutBound",  
      "priority": 65500,  
      "protocol": "All",  
      "sourceAddressPrefix": "0.0.0.0/0",  
      "sourceAddressPrefixes": [  
        "0.0.0.0/0"  
,  
        "sourcePortRange": "0-65535",  
        "sourcePortRanges": [  
          "0-65535"  
        ]  
      ]  
    }  
}
```

The rule lists **0.0.0.0/0** as the **destinationAddressPrefix**. The rule denies the outbound communication to 172.131.0.100, because the address is not within the **destinationAddressPrefix** of any of the other outbound rules in the output from the `az network nic list-effective-nsg` command. To allow the outbound communication, you could add a security rule with a higher priority, that allows outbound traffic to port 80 at 172.131.0.100.

When you ran the `az network watcher test-ip-flow` command in [Use IP flow verify](#) to test inbound communication from 172.131.0.100, the output informed you that the **DefaultInboundDenyAll** rule denied the communication. The **DefaultInboundDenyAll** rule equates to the **DenyAllInBound** rule listed in the following output from the `az network nic list-effective-nsg` command:

```
{  
  "access": "Deny",  
  "additionalProperties": {},  
  "destinationAddressPrefix": "0.0.0.0/0",  
  "destinationAddressPrefixes": [  
    "0.0.0.0/0"  
,  
    "destinationPortRange": "0-65535",  
    "destinationPortRanges": [  
      "0-65535"  
,  
      "direction": "Inbound",  
      "expandedDestinationAddressPrefix": null,  
      "expandedSourceAddressPrefix": null,  
      "name": "defaultSecurityRules/DenyAllInBound",  
      "priority": 65500,  
      "protocol": "All",  
      "sourceAddressPrefix": "0.0.0.0/0",  
      "sourceAddressPrefixes": [  
        "0.0.0.0/0"  
,  
        "sourcePortRange": "0-65535",  
        "sourcePortRanges": [  
          "0-65535"  
        ]  
      ]  
    },  
  },  
},
```

The **DenyAllInBound** rule is applied because, as shown in the output, no other higher priority rule exists in the output from the `az network nic list-effective-nsg` command that allows port 80 inbound to the VM from 172.131.0.100. To allow the inbound communication, you could add a security rule with a higher priority that allows port 80 inbound from 172.131.0.100.

The checks in this quickstart tested Azure configuration. If the checks return expected results and you still have network problems, ensure that you don't have a firewall between your VM and the endpoint you're communicating with and that the operating system in your VM doesn't have a firewall that is allowing or denying communication.

Clean up resources

When no longer needed, you can use `az group delete` to remove the resource group and all of the resources it contains:

```
az group delete --name myResourceGroup --yes
```

Next steps

In this quickstart, you created a VM and diagnosed inbound and outbound network traffic filters. You learned that network security group rules allow or deny traffic to and from a VM. Learn more about [security rules](#) and how to [create security rules](#).

Even with the proper network traffic filters in place, communication to a VM can still fail, due to routing configuration. To learn how to diagnose VM network routing problems, see [Diagnose VM routing problems](#) or, to diagnose outbound routing, latency, and traffic filtering problems, with one tool, see [Connection troubleshoot](#).

Tutorial: Diagnose a virtual machine network routing problem using the Azure portal

1/28/2020 • 4 minutes to read • [Edit Online](#)

When you deploy a virtual machine (VM), Azure creates several default routes for it. You may create custom routes to override Azure's default routes. Sometimes, a custom route can result in a VM not being able to communicate with other resources. In this tutorial, you learn how to:

- Create a VM
- Test communication to a URL using the next hop capability of Network Watcher
- Test communication to an IP address
- Diagnose a routing problem, and learn how you can resolve it

If you prefer, you can diagnose a virtual machine network routing problem using the [Azure CLI](#) or [Azure PowerShell](#).

If you don't have an Azure subscription, create a [free account](#) before you begin.

Log in to Azure

Log in to the Azure portal at <https://portal.azure.com>.

Create a VM

1. Select **+ Create a resource** found on the upper, left corner of the Azure portal.
2. Select **Compute**, and then select **Windows Server 2016 Datacenter** or **Ubuntu Server 17.10 VM**.
3. Enter, or select, the following information, accept the defaults for the remaining settings, and then select **OK**:

SETTING	VALUE
Name	myVm
User name	Enter a user name of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the defined complexity requirements .
Subscription	Select your subscription.
Resource group	Select Create new and enter myResourceGroup .
Location	Select East US

4. Select a size for the VM and then select **Select**.
5. Under **Settings**, accept all the defaults, and select **OK**.

- Under **Create** of the **Summary**, select **Create** to start VM deployment. The VM takes a few minutes to deploy. Wait for the VM to finish deploying before continuing with the remaining steps.

Test network communication

To test network communication with Network Watcher, you must first enable a network watcher in at least one Azure region and then use Network Watcher's next hop capability to test communication.

Enable network watcher

If you already have a network watcher enabled in at least one region, skip to [Use next hop](#).

- In the portal, select **All services**. In the **Filter box**, enter *Network Watcher*. When **Network Watcher** appears in the results, select it.
- Select **Regions**, to expand it, and then select ... to the right of **East US**, as shown in the following picture:

NAME	REGION	STATUS	...
27 regions		Disabled	...
West US	West US	Disabled	...
East US	East US	Disabled	...
North Europe	North Europe	Disabled	...
West Europe	West Europe	Disabled	...

- Select **Enable Network Watcher**.

Use next hop

Azure automatically creates routes to default destinations. You may create custom routes that override the default routes. Sometimes, custom routes can cause communication to fail. Use the next hop capability of Network Watcher to determine which route Azure is using to route traffic.

- In the Azure portal, select **Next hop**, under **Network Watcher**.
- Select your subscription, enter or select the following values, and then select **Next hop**, as shown in the picture that follows:

SETTING	VALUE
Resource group	Select myResourceGroup
Virtual machine	Select myVm
Network interface	myvm - Your network interface name may be different.
Source IP address	10.0.0.4
Destination IP address	13.107.21.200 - One of the addresses for <www.bing.com>.

Specify a target virtual machine and destination IP address to view the next hop.

Subscription* ⓘ

Resource group* ⓘ

Virtual machine* ⓘ

Network interface* ⓘ

Source IP address* ⓘ

Destination IP address* ⓘ

Next hop

After a few seconds, the result informs you that the next hop type is **Internet**, and that the **Route table ID** is **System Route**. This result lets you know that there is a valid system route to the destination.

3. Change the **Destination IP address** to **172.31.0.100** and select **Next hop** again. The result returned informs you that **None** is the **Next hop type**, and that the **Route table ID** is also **System Route**. This result lets you know that, while there is a valid system route to the destination, there is no next hop to route the traffic to the destination.

View details of a route

1. To analyze routing further, review the effective routes for the network interface. In the search box at the top of the portal, enter **myvm** (or whatever the name was of the network interface you checked). When **myvm** appears in the search results, select it.
2. Select **Effective routes** under **SUPPORT + TROUBLESHOOTING**, as shown in the following picture:

SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE
Default	Active	10.0.0.0/24	Virtual network
Default	Active	0.0.0.0/0	Internet
Default	Active	10.0.0.0/8	None
Default	Active	100.64.0.0/10	None
Default	Active	172.16.0.0/12	None
Default	Active	192.168.0.0/16	None

When you ran the test using 13.107.21.200 in [Use next hop](#), the route with the address prefix 0.0.0.0/0 was used to route traffic to the address, since no other route includes the address. By default, all addresses not

specified within the address prefix of another route are routed to the internet.

When you ran the test using 172.31.0.100 however, the result informed you that there was no next hop type. As you can see in the previous picture, though there is a default route to the 172.16.0.0/12 prefix, which includes the 172.31.0.100 address, the **NEXT HOP TYPE** is **None**. Azure creates a default route to 172.16.0.0/12, but doesn't specify a next hop type until there is a reason to. If, for example, you added the 172.16.0.0/12 address range to the address space of the virtual network, Azure changes the **NEXT HOP TYPE** to **Virtual network** for the route. A check would then show **Virtual network** as the **NEXT HOP TYPE**.

Clean up resources

When no longer needed, delete the resource group and all of the resources it contains:

1. Enter *myResourceGroup* in the **Search** box at the top of the portal. When you see **myResourceGroup** in the search results, select it.
2. Select **Delete resource group**.
3. Enter *myResourceGroup* for **TYPE THE RESOURCE GROUP NAME:** and select **Delete**.

Next steps

In this tutorial, you created a VM and diagnosed network routing from the VM. You learned that Azure creates several default routes and tested routing to two different destinations. Learn more about [routing in Azure](#) and how to [create custom routes](#).

For outbound VM connections, you can also determine the latency, allowed and denied network traffic between the VM and an endpoint, and the route used to an endpoint, using Network Watcher's [connection troubleshoot](#) capability. Learn how you can monitor communication between a VM and an endpoint, such as an IP address or URL, over time using the Network Watcher connection monitor capability.

[Monitor a network connection](#)

Tutorial: Monitor network communication between two virtual machines using the Azure portal

1/28/2020 • 6 minutes to read • [Edit Online](#)

Successful communication between a virtual machine (VM) and an endpoint such as another VM, can be critical for your organization. Sometimes, configuration changes are introduced which can break communication. In this tutorial, you learn how to:

- Create two VMs
- Monitor communication between VMs with the connection monitor capability of Network Watcher
- Generate alerts on Connection Monitor metrics
- Diagnose a communication problem between two VMs, and learn how you can resolve it

If you don't have an Azure subscription, create a [free account](#) before you begin.

Sign in to Azure

Sign in to the [Azure portal](#).

Create VMs

Create two VMs.

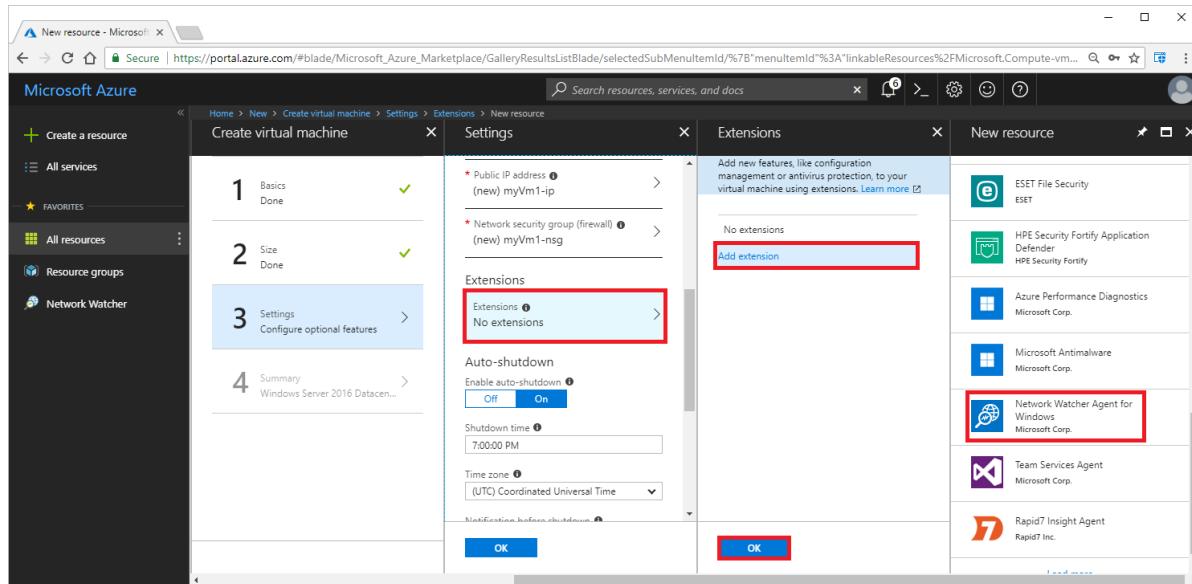
Create the first VM

1. Select **+ Create a resource** found on the upper, left corner of the Azure portal.
2. Select **Compute**, and then select an operating system. In this tutorial, **Windows Server 2016 Datacenter** is used.
3. Enter, or select, the following information, accept the defaults for the remaining settings, and then select **OK**:

SETTING	VALUE
Name	myVm1
User name	Enter a user name of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the defined complexity requirements .
Subscription	Select your subscription.
Resource group	Select Create new and enter myResourceGroup .
Location	Select East US

4. Select a size for the VM and then select **Select**.
5. Under **Settings**, select **Extensions**. Select **Add extension**, and select **Network Watcher Agent for**

Windows, as shown in the following picture:



6. Under **Network Watcher Agent for Windows**, select **Create**, under **Install extension** select **OK**, and then under **Extensions**, select **OK**.
7. Accept the defaults for the remaining **Settings** and select **OK**.
8. Under **Create** of the **Summary**, select **Create** to start VM deployment.

Create the second VM

Complete the steps in [Create the first VM](#) again, with the following changes:

STEP	SETTING	VALUE
1	Select a version of Ubuntu Server	
3	Name	myVm2
3	Authentication type	Paste your SSH public key or select Password , and enter a password.
3	Resource group	Select Use existing and select myResourceGroup .
6	Extensions	Network Watcher Agent for Linux

The VM takes a few minutes to deploy. Wait for the VM to finish deploying before continuing with the remaining steps.

Create a connection monitor

Create a connection monitor to monitor communication over TCP port 22 from *myVm1* to *myVm2*.

1. On the left side of the portal, select **All services**.
2. Start typing *network watcher* in the **Filter** box. When **Network Watcher** appears in the search results, select it.
3. Under **MONITORING**, select **Connection monitor**.
4. Select **+ Add**.

5. Enter or select the information for the connection you want to monitor, and then select **Add**. In the example shown in the following picture, the connection monitored is from the *myVm1* VM to the *myVm2* VM over port 22:

SETTING	VALUE
Name	myVm1-myVm2(22)
Source	
Virtual machine	myVm1
Destination	
Select a virtual machine	
Virtual machine	myVm2
Port	22

View a connection monitor

1. Complete steps 1-3 in [Create a connection monitor](#) to view connection monitoring. You see a list of existing connection monitors, as shown in the following picture:

Network Watcher - Connection monitor

Microsoft

Search (Ctrl+ /)

+ Add

Overview

MONITORING

Topology

Connection monitor

NAME

Subscription

Resource Group

Virtual Machine

Filter by name

myVm1

myVm2

myVm1-myVm2(22)

myResourceGroup

myVm1

myVm2

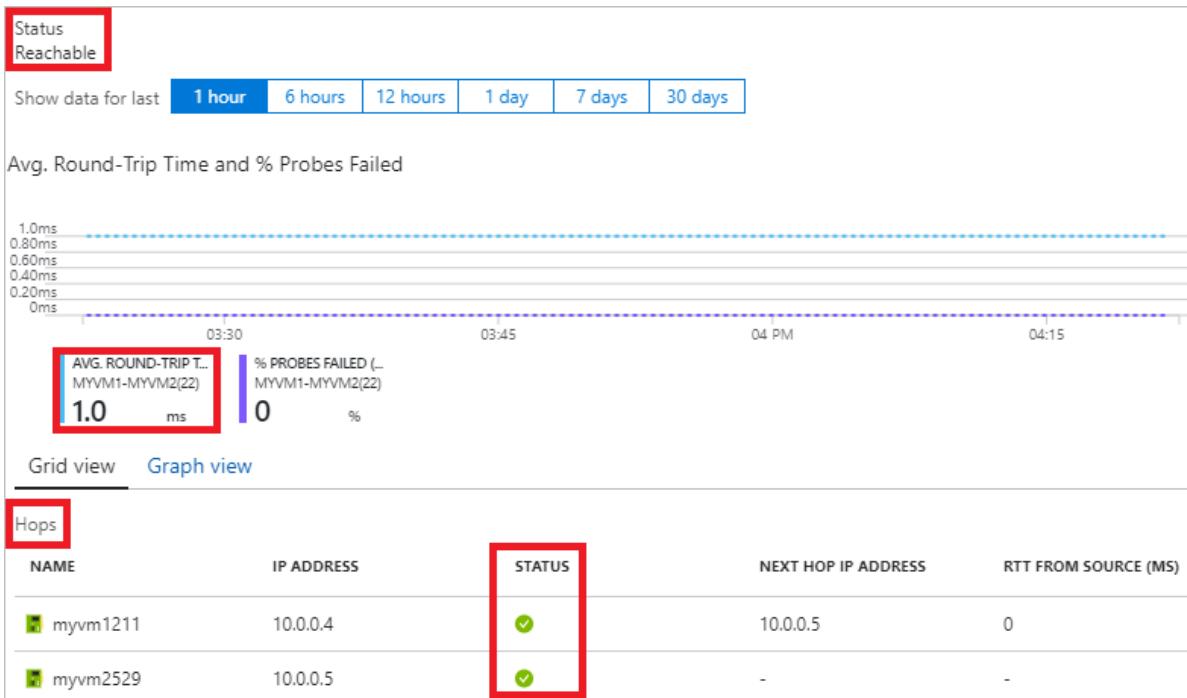
22

Running

60

INTERVAL (SECO...)

2. Select the monitor with the name **myVm1-myVm2(22)**, as shown in the previous picture, to see details for the monitor, as shown in the following picture:



Note the following information:

ITEM	VALUE	DETAILS
Status	Reachable	Lets you know whether the endpoint is reachable or not.
AVG. ROUND-TRIP	Lets you know the round-trip time to make the connection, in milliseconds. Connection monitor probes the connection every 60 seconds, so you can monitor latency over time.	
Hops	Connection monitor lets you know the hops between the two endpoints. In this example, the connection is between two VMs in the same virtual network, so there is only one hop, to the 10.0.0.5 IP address. If any existing system or custom routes, route traffic between the VMs through a VPN gateway, or network virtual appliance, for example, additional hops are listed.	

ITEM	VALUE	DETAILS
STATUS	The green check marks for each endpoint let you know that each endpoint is healthy.	

Generate alerts

Alerts are created by alert rules in Azure Monitor and can automatically run saved queries or custom log searches at regular intervals. A generated alert can automatically run one or more actions, such as to notify someone or start another process. When setting an alert rule, the resource that you target determines the list of available metrics that you can use to generate alerts.

1. In Azure portal, select the **Monitor** service, and then select **Alerts > New alert rule**.
2. Click **Select target**, and then select the resources that you want to target. Select the **Subscription**, and set **Resource type** to filter down to the Connection Monitor that you want to use.

3. Once you have selected a resource to target, select **Add criteria**. The Network Watcher has **metrics on which you can create alerts**. Set **Available signals** to the metrics ProbesFailedPercent and AverageRoundtripMs:

4. Fill out the alert details like alert rule name, description and severity. You can also add an action group to the alert to automate and customize the alert response.

View a problem

By default, Azure allows communication over all ports between VMs in the same virtual network. Over time, you, or someone in your organization, might override Azure's default rules, inadvertently causing a communication failure. Complete the following steps to create a communication problem and then view the connection monitor again:

1. In the search box at the top of the portal, enter *myResourceGroup*. When the **myResourceGroup** resource group appears in the search results, select it.
2. Select the **myVm2-nsg** network security group.
3. Select **Inbound security rules**, and then select **Add**, as shown in the following picture:

PRIORITY	NAME	PORT
1000	default-allow-ssh	22
65000	AllowVnetInBound	Any
65001	AllowAzureLoadBalancerInBound	Any
65500	DenyAllInBound	Any

4. The default rule that allows communication between all VMs in a virtual network is the rule named **AllowVnetInBound**. Create a rule with a higher priority (lower number) than the **AllowVnetInBound** rule that denies inbound communication over port 22. Select, or enter, the following information, accept the remaining defaults, and then select **Add**:

SETTING	VALUE
Destination port ranges	22
Action	Deny
Priority	100
Name	DenySshInbound

5. Since connection monitor probes at 60-second intervals, wait a few minutes and then on the left side of the portal, select **Network Watcher**, then **Connection monitor**, and then select the **myVm1-myVm2(22)** monitor again. The results are different now, as shown in the following picture:

NAME	IP ADDRESS	STATUS
myvm1211	10.0.0.4	✓
myvm2529	10.0.0.5	!

You can see that there's a red exclamation icon in the status column for the **myvm2529** network interface.

6. To learn why the status has changed, select 10.0.0.5, in the previous picture. Connection monitor informs you that the reason for the communication failure is: *Traffic blocked due to the following network security group rule: UserRule_DenySshInbound*.

If you didn't know that someone had implemented the security rule you created in step 4, you'd learn from connection monitor that the rule is causing the communication problem. You could then change, override, or remove the rule, to restore communication between the VMs.

Clean up resources

When no longer needed, delete the resource group and all of the resources it contains:

1. Enter *myResourceGroup* in the **Search** box at the top of the portal. When you see **myResourceGroup** in the search results, select it.
2. Select **Delete resource group**.
3. Enter *myResourceGroup* for **TYPE THE RESOURCE GROUP NAME:** and select **Delete**.

Next steps

In this tutorial, you learned how to monitor a connection between two VMs. You learned that a network security group rule prevented communication to a VM. To learn about all of the different responses connection monitor can return, see [response types](#). You can also monitor a connection between a VM, a fully qualified domain name, a uniform resource identifier, or an IP address.

At some point, you may find that resources in a virtual network are unable to communicate with resources in other networks connected by an Azure virtual network gateway. Advance to the next tutorial to learn how to diagnose a problem with a virtual network gateway.

[Diagnose communication problems between networks](#)

Tutorial: Diagnose a communication problem between networks using the Azure portal

1/28/2020 • 4 minutes to read • [Edit Online](#)

A virtual network gateway connects an Azure virtual network to an on-premises, or other virtual network. In this tutorial, you learn how to:

- Diagnose a problem with a virtual network gateway with Network Watcher's VPN diagnostics capability
- Diagnose a problem with a gateway connection
- Resolve a problem with a gateway

If you don't have an Azure subscription, create a [free account](#) before you begin.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Prerequisites

To use VPN diagnostics, you must have an existing, running VPN gateway. If you don't have an existing VPN gateway to diagnose, you can deploy one using a [PowerShell script](#). You can run the PowerShell script from:

- **A local PowerShell installation:** The script requires the Azure PowerShell `Az` module. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.
- **The Azure Cloud Shell:** The [Azure Cloud Shell](#) has the latest version of PowerShell installed and configured, and logs you into Azure.

The script takes approximately an hour to create a VPN gateway. The remaining steps assume that the gateway you're diagnosing is the one deployed by this script. If you diagnose your own existing gateway instead, your results will vary.

Sign in to Azure

Sign in to the [Azure portal](#).

Enable Network Watcher

If you already have a network watcher enabled in the East US region, skip to [Diagnose a gateway](#).

1. In the portal, select **All services**. In the **Filter box**, enter *Network Watcher*. When **Network Watcher** appears in the results, select it.
2. Select **Regions**, to expand it, and then select ... to the right of **East US**, as shown in the following picture:

NAME	REGION	STATUS
27 regions		Disabled
West US		Disabled
East US		Disabled
North Europe		Disabled
West Europe		Disabled

3. Select **Enable Network Watcher**.

Diagnose a gateway

1. On the left side of the portal, select **All services**.
2. Start typing *network watcher* in the **Filter** box. When **Network Watcher** appears in the search results, select it.
3. Under **NETWORK DIAGNOSTIC TOOLS**, select **VPN Diagnostics**.
4. Select **Storage account**, and then select the storage account you want to write diagnostic information to.
5. From the list of **Storage accounts**, select the storage account you want to use. If you don't have an existing storage account, select **+ Storage account**, enter, or select the required information, and then select **Create**, to create one. If you created a VPN gateway using the script in [prerequisites](#), you may want to create the storage account in the same resource group, *TestRG1*, as the gateway.
6. From the list of **Containers**, select the container you want to use, and then select **Select**. If you don't have any containers, select **+ Container**, enter a name for the container, then select **OK**.
7. Select a gateway, and then select **Start troubleshooting**. As shown in the following picture, the test is run against a gateway named **Vnet1GW**:

NAME	TROUBLESHOOTING STATUS	RESOURCE STATUS	RESOURCE GROUP	LOCATION
<input checked="" type="checkbox"/> Vnet1GW	Not started	Succeeded	TestRG1	East US
<input type="checkbox"/> VNet1toSite1	-	Succeeded	TestRG1	East US

8. While the test is running, **Running** appears in the **TROUBLESHOOTING STATUS** column where **Not**

started is shown, in the previous picture. The test may take several minutes to run.

- View the status of a completed test. The following picture shows the status results of a completed diagnostic test:

NAME	TROUBLESHOOTING STATUS	RESOURCE STATUS	RESOURCE GROUP	LOCATION
<input checked="" type="checkbox"/> VNet1GW	✖️ Unhealthy	Succeeded	TestRG1	East US
<input type="checkbox"/> VNet1toSite1	-	Succeeded	TestRG1	East US

Details

Status Action

Resource [VNet1GW](#)
Storage path <https://myvpndiagnostics.blob.core.windows.net/vpndiagnostics>

Summary
Your VPN connectivity is impacted because the S2S VPN tunnels are disconnected

Detail
The S2S VPN tunnels could not connect because of IKE or connectivity issues

You can see that the **TROUBLESHOOTING STATUS** is **Unhealthy**, as well as a **Summary** and **Detail** of the problem on the **Status** tab.

- When you select the **Action** tab, VPN diagnostics provides additional information. In the example, shown in the following picture, VPN diagnostics lets you know that you should check the health of each connection:

Status Action

Check health of each individual connection to get more details
contact support
If your VPN gateway isn't up and running by the expected resolution time, contact support
<http://azure.microsoft.com/support>

Diagnose a gateway connection

A gateway is connected to other networks via a gateway connection. Both the gateway and gateway connections must be healthy for successful communication between a virtual network and a connected network.

- Complete step 7 of [Diagnose a gateway](#) again, this time, selecting a connection. In the following example, a connection named **VNet1toSite1** is tested:

▶ Start troubleshooting

Network Watcher VPN Troubleshoot diagnoses the health of the virtual network gateway or connection. This request is a long running transaction, and the results are returned once the diagnosis is complete. You can select multiple gateways or connections to troubleshoot simultaneously.
[Learn more](#).

Choose a subscription i Resource group i Location i

* Storage account >
<https://myvpndiagnostics.blob.core...>

NAME	TROUBLESHOOTING STATUS	RESOURCE STATUS	RESOURCE GROUP	LOCATION
<input type="checkbox"/> VNet1GW	✖️ Unhealthy	Succeeded	TestRG1	East US
<input checked="" type="checkbox"/> VNet1toSite1	⌚ Running	Succeeded	TestRG1	East US

The test runs for several minutes.

- After the test of the connection is complete, you receive results similar to the results shown in the following pictures on the **Status** and **Action** tabs:

Status	Action
Resource VNet1GW Storage path https://myvpndiagnostics.blob.core.windows.net/vpndiagnostics Summary The connection cannot be established because the other VPN device is unreachable Detail If the on-premises VPN device is unreachable or not responding to the Azure VPN gateway IKE handshake, the VPN connection cannot establish	

Status	Action
Verify if the public IP address Verify if the public IP address of the corresponding Azure Local Network Gateway is configured correctly https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-modify-local-network-gateway/ Verify the configuration Verify the configuration of your on-premises VPN device so it allows your Azure VPN gateway to establish connections https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-about-vpn-devices/ Verify the configuration Verify if there is a network security group (NSG) applied to the GatewaySubnet https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-create-nsg-arm-pportal contact support If you are experiencing problems you believe are caused by Azure, contact support http://azure.microsoft.com/support	

VPN diagnostics informs you what is wrong on the **Status** tab, and gives you several suggestions for what may be causing the problem on the **Action** tab.

If the gateway you tested was the one deployed by the [script](#) in [Prerequisites](#), then the problem on the **Status** tab, and the first two items on the **Action** tab are exactly what the problem is. The script configures a placeholder IP address, 23.99.221.164, for the on-premises VPN gateway device.

To resolve the issue, you need to ensure that your on-premises VPN gateway is [configured properly](#), and change the IP address configured by the script for the local network gateway, to the actual public address of your on-premises VPN gateway.

Clean up resources

If you created a VPN gateway using the script in the [prerequisites](#) solely to complete this tutorial, and no longer need it, delete the resource group and all of the resources it contains:

- Enter *TestRG1* in the **Search** box at the top of the portal. When you see **TestRG1** in the search results, select it.
- Select **Delete resource group**.
- Enter *TestRG1* for **TYPE THE RESOURCE GROUP NAME:** and select **Delete**.

Next steps

In this tutorial, you learned how to diagnose a problem with a virtual network gateway. You may want to log network communication to and from a VM so that you can review the log for anomalies. To learn how, advance to the next tutorial.

[Log network traffic to and from a VM](#)

Tutorial: Log network traffic to and from a virtual machine using the Azure portal

1/30/2020 • 6 minutes to read • [Edit Online](#)

A network security group (NSG) enables you to filter inbound traffic to, and outbound traffic from, a virtual machine (VM). You can log network traffic that flows through an NSG with Network Watcher's NSG flow log capability. In this tutorial, you learn how to:

- Create a VM with a network security group
- Enable Network Watcher and register the Microsoft.Insights provider
- Enable a traffic flow log for an NSG, using Network Watcher's NSG flow log capability
- Download logged data
- View logged data

If you don't have an Azure subscription, create a [free account](#) before you begin.

Create a VM

1. Select + **Create a resource** found on the upper, left corner of the Azure portal.
2. Select **Compute**, and then select **Windows Server 2016 Datacenter** or a version of **Ubuntu Server**.
3. Enter, or select, the following information, accept the defaults for the remaining settings, and then select **OK**:

SETTING	VALUE
Name	myVm
User name	Enter a user name of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the defined complexity requirements .
Subscription	Select your subscription.
Resource group	Select Create new and enter myResourceGroup .
Location	Select East US

4. Select a size for the VM and then select **Select**.
5. Under **Settings**, accept all the defaults, and select **OK**.
6. Under **Create** of the **Summary**, select **Create** to start VM deployment. The VM takes a few minutes to deploy. Wait for the VM to finish deploying before continuing with the remaining steps.

The VM takes a few minutes to create. Don't continue with remaining steps until the VM has finished creating. While the portal creates the VM, it also creates a network security group with the name **myVm-nsg**, and associates it to the network interface for the VM.

Enable Network Watcher

If you already have a network watcher enabled in the East US region, skip to [Register Insights provider](#).

1. In the portal, select **All services**. In the **Filter box**, enter **Network Watcher**. When **Network Watcher** appears in the results, select it.
2. Select **Regions**, to expand it, and then select ... to the right of **East US**, as shown in the following picture:

NAME	REGION	STATUS	...
27 regions		Disabled	...
West US		Disabled	...
East US		Disabled	...
North Europe		Disabled	...
West Europe		Disabled	...

3. Select **Enable Network Watcher**.

Register Insights provider

NSG flow logging requires the **Microsoft.Insights** provider. To register the provider, complete the following steps:

1. In the top, left corner of portal, select **All services**. In the Filter box, type **Subscriptions**. When **Subscriptions** appear in the search results, select it.
2. From the list of subscriptions, select the subscription you want to enable the provider for.
3. Select **Resource providers**, under **SETTINGS**.
4. Confirm that the **STATUS** for the **microsoft.insights** provider is **Registered**, as shown in the picture that follows. If the status is **Unregistered**, then select **Register**, to the right of the provider.

PROVIDER	STATUS	Re-register	Unregister
Microsoft.Advisor	Registered	Re-register	Unregister
Microsoft.Automation	Registered	Re-register	Unregister
Microsoft.Batch	Unregistered	Register	
Microsoft.Cache	Registered	Re-register	Unregister
Microsoft.ClassicCompute	Registered	Re-register	Unregister
Microsoft.ClassicNetwork	Registered	Re-register	Unregister
Microsoft.ClassicStorage	Registered	Re-register	Unregister
Microsoft.ClassicInfrastructureMigrate	Registered	Re-register	Unregister
Microsoft.Compute	Registered	Re-register	Unregister
Microsoft.ContainerService	Registered	Re-register	Unregister
Microsoft.DevTestLab	Registered	Re-register	Unregister
microsoft.insights	Registered	Re-register	Unregister
Microsoft.KeyVault	Registered	Re-register	Unregister

Enable NSG flow log

1. NSG flow log data is written to an Azure Storage account. To create an Azure Storage account, select +

Create a resource at the top, left corner of the portal.

2. Select **Storage**, then select **Storage account - blob, file, table, queue**.
3. Enter, or select the following information, accept the remaining defaults, and then select **Create**.

SETTING	VALUE
Name	3-24 characters in length, can only contain lowercase letters and numbers, and must be unique across all Azure Storage accounts.
Location	Select East US
Resource group	Select Use existing , and then select myResourceGroup

The storage account may take around minute to create. Don't continue with remaining steps until the storage account is created. In all cases, the storage account must be in the same region as the NSG.

4. In the top, left corner of portal, select **All services**. In the **Filter** box, type *Network Watcher*. When **Network Watcher** appears in the search results, select it.
5. Under **LOGS**, select **NSG flow logs**, as shown in the following picture:

NAME	RESOURCE TYPE	RESOURCE GROUP	STATUS	LOCATION	STORAGE ACCOUNT	TRAFFIC ANALYTICS...	TRAFFIC ANALYTICS...
myVm-nsg	Network security ...	myResourceGroup	Disabled	East US	Disabled	-	-

6. From the list of NSGs, select the NSG named **myVm-nsg**.
7. Under **Flow logs settings**, select **On**.
8. Select the flow logging version. Version 2 contains flow-session statistics (Bytes and Packets)

Dashboard > Network Watcher - NSG flow logs > Flow logs settings

Flow logs settings

Save

Flow logs

Status

Flow Logs version ?

Version 1 logs ingress and egress IP traffic flows for both allowed and denied traffic. Version 2 provides additional throughput information (bytes and packets) per flow.
[Learn more.](#)

Storage account >
Configure

Retention (days) ?

Traffic Analytics

Traffic Analytics provides rich analytics and visualization derived from NSG flow logs and other Azure resources' data. Drill through geo-map, easily figure out traffic hotspots and get insights into optimization possibilities.

[Learn about all features](#)

i To use this feature, choose an OMS workspace. To minimize data egress costs, we recommend that you choose a workspace in the same region your flow logs storage account is located. Network Performance Monitor solution will be installed on the workspace. We also advise that you use the same workspace for all NSGs as much as possible. Additional meta-data is added to your flow logs data, to provide enhanced analytics.

Traffic Analytics status

[Microsoft privacy statement](#)
This privacy statement explains what personal data Microsoft collects from you, through our interactions with you and through our products, and how we use that data.

9. Select the storage account that you created in step 3.

NOTE

NSG Flow Logs do not work with storage accounts that have [hierarchical namespace](#) enabled.

10. In the top, left corner of portal, select **All services**. In the **Filter** box, type *Network Watcher*. When **Network Watcher** appears in the search results, select it.

11. Set **Retention (days)** to 5, and then select **Save**.

Download flow log

- From Network Watcher, in the portal, select **NSG flow logs** under **LOGS**.
- Select **You can download flow logs from configured storage accounts**, as shown in the following picture:

3. Select the storage account that you configured in step 2 of [Enable NSG flow log](#).
4. Under **Blob service**, select **Blobs**, and then select the **insights-logs-networksecuritygroupflowevent** container.
5. In the container, navigate the folder hierarchy until you get to a PT1H.json file, as shown in the picture that follows. Log files are written to a folder hierarchy that follows the following naming convention:
<https://<storageAccountName>.blob.core.windows.net/insights-logs-networksecuritygroupflowevent/resourceId=/SUBSCRIPTIONS/<subscriptionID>/RESOURCEGROUPS/<resourceGroupName>/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/<nsgName>/y=<year>/m=<month>/d=<day>/h=<hour>/m=00/macAddress=<macAddress>/PT1H.json>

6. Select ... to the right of the PT1H.json file and select **Download**.

View flow log

The following json is an example of what you'll see in the PT1H.json file for each flow that data is logged for:

Version 1 flow log event

```
{  
    "time": "2018-05-01T15:00:02.1713710Z",  
    "systemId": "<Id>",  
    "category": "NetworkSecurityGroupFlowEvent",  
    "resourceId":  
        "/SUBSCRIPTIONS/<Id>/RESOURCEGROUPS/MYRESOURCEGROUP/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/MYVM-  
        NSG",  
    "operationName": "NetworkSecurityGroupFlowEvents",  
    "properties": {  
        "Version": 1,  
        "flows": [  
            {  
                "rule": "UserRule_default-allow-rdp",  
                "flows": [  
                    {  
                        "mac": "000D3A170C69",  
                        "flowTuples": [  
                            "1525186745,192.168.1.4,10.0.0.4,55960,3389,T,I,A"  
                        ]  
                    }  
                ]  
            }  
        ]  
    }  
}
```

Version 2 flow log event

```
{
  "time": "2018-11-13T12:00:35.389926Z",
  "systemId": "a0fcfa5ce-022c-47b1-9735-89943b42f2fa",
  "category": "NetworkSecurityGroupFlowEvent",
  "resourceId": "/SUBSCRIPTIONS/00000000-0000-0000-0000-
000000000000/RESOURCEGROUPS/FABRIKAMRG/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/FABRIAKMVM1-NSG",
  "operationName": "NetworkSecurityGroupFlowEvents",
  "properties": {
    "Version": 2,
    "flows": [
      {
        "rule": "DefaultRule_DenyAllInBound",
        "flows": [
          {
            "mac": "000D3AF87856",
            "flowTuples": [
              "1542110402,94.102.49.190,10.5.16.4,28746,443,U,I,D,B,,,",
              "1542110424,176.119.4.10,10.5.16.4,56509,59336,T,I,D,B,,,",
              "1542110432,167.99.86.8,10.5.16.4,48495,8088,T,I,D,B,,,"
            ]
          }
        ]
      },
      {
        "rule": "DefaultRule_AllowInternetOutBound",
        "flows": [
          {
            "mac": "000D3AF87856",
            "flowTuples": [
              "1542110377,10.5.16.4,13.67.143.118,59831,443,T,O,A,B,,,",
              "1542110379,10.5.16.4,13.67.143.117,59932,443,T,O,A,E,1,66,1,66",
              "1542110379,10.5.16.4,13.67.143.115,44931,443,T,O,A,C,30,16978,24,14008",
              "1542110406,10.5.16.4,40.71.12.225,59929,443,T,O,A,E,15,8489,12,7054"
            ]
          }
        ]
      }
    ]
  }
}
```

The value for **mac** in the previous output is the MAC address of the network interface that was created when the VM was created. The comma-separated information for **flowTuples**, is as follows:

EXAMPLE DATA	WHAT DATA REPRESENTS	EXPLANATION
1542110377	Time stamp	The time stamp of when the flow occurred, in UNIX EPOCH format. In the previous example, the date converts to May 1, 2018 at 2:59:05 PM GMT.
10.0.0.4	Source IP address	The source IP address that the flow originated from. 10.0.0.4 is the private IP address of the VM you created in Create a VM .
13.67.143.118	Destination IP address	The destination IP address that the flow was destined to.
44931	Source port	The source port that the flow originated from.

EXAMPLE DATA	WHAT DATA REPRESENTS	EXPLANATION
443	Destination port	The destination port that the flow was destined to. Since the traffic was destined to port 443, the rule named UserRule_default-allow-rdp , in the log file processed the flow.
T	Protocol	Whether the protocol of the flow was TCP (T) or UDP (U).
O	Direction	Whether the traffic was inbound (I) or outbound (O).
A	Action	Whether the traffic was allowed (A) or denied (D).
C	Flow State Version 2 Only	Captures the state of the flow. Possible states are B : Begin, when a flow is created. Statistics aren't provided. C : Continuing for an ongoing flow. Statistics are provided at 5-minute intervals. E : End, when a flow is terminated. Statistics are provided.
30	Packets sent - Source to destination Version 2 Only	The total number of TCP or UDP packets sent from source to destination since last update.
16978	Bytes sent - Source to destination Version 2 Only	The total number of TCP or UDP packet bytes sent from source to destination since last update. Packet bytes include the packet header and payload.
24	Packets sent - Destination to source Version 2 Only	The total number of TCP or UDP packets sent from destination to source since last update.
14008	Bytes sent - Destination to source Version 2 Only	The total number of TCP and UDP packet bytes sent from destination to source since last update. Packet bytes include packet header and payload.

Next steps

In this tutorial, you learned how to enable NSG flow logging for an NSG. You also learned how to download and view data logged in a file. The raw data in the json file can be difficult to interpret. To visualize Flow Logs data, you can use [Azure Traffic Analytics](#), [Microsoft Power BI](#), and other tools. You can try alternate methods of enabling NSG Flow Logs like [PowerShell](#), [Azure CLI](#), [REST API](#) and [ARM templates](#).

Introduction to IP flow verify in Azure Network Watcher

1/28/2020 • 2 minutes to read • [Edit Online](#)

IP flow verify checks if a packet is allowed or denied to or from a virtual machine. The information consists of direction, protocol, local IP, remote IP, local port, and remote port. If the packet is denied by a security group, the name of the rule that denied the packet is returned. While any source or destination IP can be chosen, IP flow verify helps administrators quickly diagnose connectivity issues from or to the internet and from or to the on-premises environment.

IP flow verify looks at the rules for all Network Security Groups (NSGs) applied to the network interface, such as a subnet or virtual machine NIC. Traffic flow is then verified based on the configured settings to or from that network interface. IP flow verify is useful in confirming if a rule in a Network Security Group is blocking ingress or egress traffic to or from a virtual machine.

An instance of Network Watcher needs to be created in all regions that you plan to run IP flow verify. Network Watcher is a regional service and can only be ran against resources in the same region. The instance used does not affect the results of IP flow verify, as any route associated with the NIC or subnet is still be returned.

Network Watcher - IP flow verify

Microsoft

Search (Ctrl+)

Overview

MONITORING

Topology

NETWORK DIAGNOSTIC TOOLS

IP flow verify

Next hop

Security group view

Packet capture

METRICS

Network subscription limit

LOGS

NSG flow logs

Diagnostic logs

Specify a target virtual machine with associated network security groups, then run an inbound or outbound packet to see if access is allowed or denied.

Subscription* ⓘ

Microsoft Azure

Resource group* ⓘ

FabrikamRG

Virtual machine* ⓘ

fabrikmvm1

Network interface* ⓘ

fabrikmvm1161

Packet details

Protocol

TCP UDP

Direction

Inbound Outbound

Local IP address* ⓘ

10.1.0.4

Local port* ⓘ

443

Remote IP address* ⓘ

204.79.197.200

Remote port* ⓘ

443

Check

Result

Access allowed

Security rule

AllowInternetOutBound

Next steps

Visit the following article to learn if a packet is allowed or denied for a specific virtual machine through the portal.
[Check if traffic is allowed on a VM with IP Flow Verify using the portal](#)

Use next hop to diagnose virtual machine routing problems

1/28/2020 • 2 minutes to read • [Edit Online](#)

Traffic from a virtual machine (VM) is sent to a destination based on the effective routes associated with a network interface (NIC). Next hop gets the next hop type and IP address of a packet from a specific VM and NIC. Knowing the next hop helps you determine if traffic is being directed to the intended destination, or whether the traffic is being sent nowhere. An improper configuration of routes, where traffic is directed to an on-premises location, or a virtual appliance, can lead to connectivity issues. Next hop also returns the route table associated with the next hop. If the route is defined as a user-defined route, that route is returned. Otherwise, next hop returns **System Route**.

The screenshot shows the 'Network Watcher - Next hop' page in the Microsoft Azure portal. On the left, there's a navigation sidebar with sections like Overview, MONITORING (Topology), NETWORK DIAGNOSTIC TOOLS (IP flow verify, Next hop, Security group view, Packet capture), METRICS, and LOGS (NSG flow logs, Diagnostic logs). The 'Next hop' item under 'NETWORK DIAGNOSTIC TOOLS' is currently selected and highlighted in blue. The main content area has a dark background with white text. It starts with a search bar and a 'Specify a target virtual machine and destination IP address to view the next hop.' instruction. Below this are several dropdown and input fields: 'Subscription' set to 'Microsoft', 'Resource group' set to 'ContosoAppGateway', 'Virtual machine' set to 'webtestvm-backend-a76qj4rjyabaya-1-aehrwnnchvwx4', 'Network interface' set to 'webtestnic-backend-a76qj4rjyabaya-1-aehrwnnchvwx4', 'Source IP address' set to '10.0.0.4', and 'Destination IP address' which is empty. At the bottom is a large blue button labeled 'Next hop'.

The next hops that might be returned by the next hop capability are as follows:

- Internet
- VirtualAppliance
- VirtualNetworkGateway
- VirtualNetwork
- VirtualNetworkPeering
- VirtualNetworkServiceEndpoint
- MicrosoftEdge
- None

To learn more about each next hop type, see [Routing overview](#).

Next steps

To learn how to use next hop to diagnose VM network routing problems, see Diagnose VM network routing problems using the [Azure portal](#), [PowerShell](#), or the [Azure CLI](#).

Introduction to connection troubleshoot in Azure Network Watcher

1/28/2020 • 2 minutes to read • [Edit Online](#)

The connection troubleshoot feature of Network Watcher provides the capability to check a direct TCP connection from a virtual machine to a virtual machine (VM), fully qualified domain name (FQDN), URI, or IPv4 address. Network scenarios are complex, they are implemented using network security groups, firewalls, user-defined routes, and resources provided by Azure. Complex configurations make troubleshooting connectivity issues challenging. Network Watcher helps reduce the amount of time to find and detect connectivity issues. The results returned can provide insights into whether a connectivity issue is due to a platform or a user configuration issue. Connectivity can be checked with [PowerShell](#), [Azure CLI](#), and [REST API](#).

IMPORTANT

Connection troubleshoot requires that the VM you troubleshoot from has the [AzureNetworkWatcherExtension](#) VM extension installed. For installing the extension on a Windows VM visit [Azure Network Watcher Agent virtual machine extension for Windows](#) and for Linux VM visit [Azure Network Watcher Agent virtual machine extension for Linux](#). The extension is not required on the destination endpoint.

Response

The following table shows the properties returned when connection troubleshoot has finished running.

PROPERTY	DESCRIPTION
ConnectionStatus	The status of the connectivity check. Possible results are Reachable and Unreachable .
AvgLatencyInMs	Average latency during the connectivity check in milliseconds. (Only shown if check status is reachable)
MinLatencyInMs	Minimum latency during the connectivity check in milliseconds. (Only shown if check status is reachable)
MaxLatencyInMs	Maximum latency during the connectivity check in milliseconds. (Only shown if check status is reachable)
ProbesSent	Number of probes sent during the check. Max value is 100.
ProbesFailed	Number of probes that failed during the check. Max value is 100.
Hops	Hop by hop path from source to destination.
Hops[].Type	Type of resource. Possible values are Source , VirtualAppliance , VnetLocal , and Internet .
Hops[].Id	Unique identifier of the hop.

PROPERTY	DESCRIPTION
Hops[].Address	IP address of the hop.
Hops[].ResourceId	ResourceId of the hop if the hop is an Azure resource. If it is an internet resource, ResourceID is Internet .
Hops[].NextHopIds	The unique identifier of the next hop taken.
Hops[].Issues	A collection of issues that were encountered during the check at that hop. If there were no issues, the value is blank.
Hops[].Issues[].Origin	At the current hop, where issue occurred. Possible values are: Inbound - Issue is on the link from the previous hop to the current hop Outbound - Issue is on the link from the current hop to the next hop Local - Issue is on the current hop.
Hops[].Issues[].Severity	The severity of the issue detected. Possible values are Error and Warning .
Hops[].Issues[].Type	The type of issue found. Possible values are: CPU Memory GuestFirewall DnsResolution NetworkSecurityRule UserDefinedRoute
Hops[].Issues[].Context	Details regarding the issue found.
Hops[].Issues[].Context[].key	Key of the key value pair returned.
Hops[].Issues[].Context[].value	Value of the key value pair returned.

The following is an example of an issue found on a hop.

```
"Issues": [
  {
    "Origin": "Outbound",
    "Severity": "Error",
    "Type": "NetworkSecurityRule",
    "Context": [
      {
        "key": "RuleName",
        "value": "UserRule_Port80"
      }
    ]
  }
]
```

Fault types

Connection troubleshoot returns fault types about the connection. The following table provides a list of the current fault types returned.

TYPE	DESCRIPTION
CPU	High CPU utilization.
Memory	High Memory utilization.
GuestFirewall	Traffic is blocked due to a virtual machine firewall configuration.
DNSResolution	DNS resolution failed for the destination address.
NetworkSecurityRule	Traffic is blocked by an NSG Rule (Rule is returned)
UserDefinedRoute	Traffic is dropped due to a user defined or system route.

Next steps

Learn how to troubleshoot connections using the [Azure portal](#), [PowerShell](#), the [Azure CLI](#), or [REST API](#).

Introduction to resource troubleshooting in Azure Network Watcher

1/28/2020 • 6 minutes to read • [Edit Online](#)

Virtual Network Gateways provide connectivity between on-premises resources and other virtual networks within Azure. Monitoring gateways and their connections are critical to ensuring communication is not broken. Network Watcher provides the capability to troubleshoot gateways and connections. The capability can be called through the portal, PowerShell, Azure CLI, or REST API. When called, Network Watcher diagnoses the health of the gateway, or connection, and returns the appropriate results. The request is a long running transaction. The results are returned once the diagnosis is complete.

The screenshot shows the 'Network Watcher - VPN Diagnostics' interface. On the left, there's a navigation sidebar with links like Overview, Topology, IP flow verify, Next hop, Security group view, VPN Diagnostics (which is selected and highlighted in blue), and Packet capture. Below that are sections for Metrics (Network subscription limit) and Logs (NSG flow logs, Diagnostic logs). The main area is titled 'Start troubleshooting' and shows a table of results. The table has columns: NAME, TROUBLESHOOTING STATUS, RESOURCE STATUS, RESOURCE GROUP, and LOCATION. There are six entries listed:

NAME	TROUBLESHOOTING STATUS	RESOURCE STATUS	RESOURCE GROUP	LOCATION
DemoVnet1Gw	Not started	Succeeded	PortalTestRg	West Central US
Vnet1toVnet2Connection	Not started	Failed	PortalTestRg	West Central US
Vnet2toVnet1Connection	Not started	Succeeded	PortalTestRg	West Central US
DemoVnet2Gw	Not started	Succeeded	PortalTestRg	West Central US
Vnet1toVnet2Connection	Not started	Failed	PortalTestRg	West Central US
Vnet2toVnet1Connection	Not started	Succeeded	PortalTestRg	West Central US

Below the table, there's a 'Details' section with tabs for Status and Action. The Status tab shows the resource 'DemoVnet1Gw'.

Results

The preliminary results returned give an overall picture of the health of the resource. Deeper information can be provided for resources as shown in the following section:

The following list is the values returned with the troubleshoot API:

- **startTime** - This value is the time the troubleshoot API call started.
- **endTime** - This value is the time when the troubleshooting ended.
- **code** - This value is UnHealthy, if there is a single diagnosis failure.
- **results** - Results is a collection of results returned on the Connection or the virtual network gateway.
 - **id** - This value is the fault type.
 - **summary** - This value is a summary of the fault.
 - **detailed** - This value provides a detailed description of the fault.
 - **recommendedActions** - This property is a collection of recommended actions to take.
 - **actionText** - This value contains the text describing what action to take.
 - **actionUri** - This value provides the URI to documentation on how to act.
 - **actionUriText** - This value is a short description of the action text.

The following tables show the different fault types (id under results from the preceding list) that are available and

if the fault creates logs.

Gateway

FAULT TYPE	REASON	LOG
NoFault	When no error is detected	Yes
GatewayNotFound	Cannot find gateway or gateway is not provisioned	No
PlannedMaintenance	Gateway instance is under maintenance	No
UserDrivenUpdate	This fault occurs when a user update is in progress. The update could be a resize operation.	No
VipUnResponsive	This fault occurs when the primary instance of the gateway can't be reached due to a health probe failure.	No
PlatformInactive	There is an issue with the platform.	No
ServiceNotRunning	The underlying service is not running.	No
NoConnectionsFoundForGateway	No connections exist on the gateway. This fault is only a warning.	No
ConnectionsNotConnected	Connections are not connected. This fault is only a warning.	Yes
GatewayCPUUsageExceeded	The current gateway CPU usage is > 95%.	Yes

Connection

FAULT TYPE	REASON	LOG
NoFault	When no error is detected	Yes
GatewayNotFound	Cannot find gateway or gateway is not provisioned	No
PlannedMaintenance	Gateway instance is under maintenance	No
UserDrivenUpdate	This fault occurs when a user update is in progress. The update could be a resize operation.	No
VipUnResponsive	This fault occurs when the primary instance of the gateway can't be reached due to a health probe failure.	No
ConnectionEntityNotFound	Connection configuration is missing	No

FAULT TYPE	REASON	LOG
ConnectionIsMarkedDisconnected	The connection is marked "disconnected"	No
ConnectionNotConfiguredOnGateway	The underlying service does not have the connection configured.	Yes
ConnectionMarkedStandby	The underlying service is marked as standby.	Yes
Authentication	Preshared key mismatch	Yes
PeerReachability	The peer gateway is not reachable.	Yes
IkePolicyMismatch	The peer gateway has IKE policies that are not supported by Azure.	Yes
WfpParse Error	An error occurred parsing the WFP log.	Yes

Supported Gateway types

The following table lists which gateways and connections are supported with Network Watcher troubleshooting:

Gateway types	
VPN	Supported
ExpressRoute	Not Supported
VPN types	
Route Based	Supported
Policy Based	Not Supported
Connection types	
IPSec	Supported
VNet2Vnet	Supported
ExpressRoute	Not Supported
VPNClient	Not Supported

Log files

The resource troubleshooting log files are stored in a storage account after resource troubleshooting is finished.

The following image shows the example contents of a call that resulted in an error.

This PC > Downloads > GatewayTenantWorker_IN_1.zip		
	Name	Type
ss	ConnectionStats.txt	Text Document
ls	CPUStat.txt	Text Document
ts	IKEErrors.txt	Text Document
ts	Scrubbed-wfpdiag.txt	Text Document
ts	wfpdiag.txt.sum	SUM File
:	wfpdiag.xml	XML Document

NOTE

In some cases, only a subset of the logs files is written to storage.

For instructions on downloading files from azure storage accounts, refer to [Get started with Azure Blob storage using .NET](#). Another tool that can be used is Storage Explorer. More information about Storage Explorer can be found here at the following link: [Storage Explorer](#)

ConnectionStats.txt

The **ConnectionStats.txt** file contains overall stats of the Connection, including ingress and egress bytes, Connection status, and the time the Connection was established.

NOTE

If the call to the troubleshooting API returns healthy, the only thing returned in the zip file is a **ConnectionStats.txt** file.

The contents of this file are similar to the following example:

```
Connectivity State : Connected
Remote Tunnel Endpoint :
Ingress Bytes (since last connected) : 288 B
Egress Bytes (Since last connected) : 288 B
Connected Since : 2/1/2017 8:22:06 PM
```

CPUStats.txt

The **CPUStats.txt** file contains CPU usage and memory available at the time of testing. The contents of this file is similar to the following example:

```
Current CPU Usage : 0 % Current Memory Available : 641 MBs
```

IKEErrors.txt

The **IKEErrors.txt** file contains any IKE errors that were found during monitoring.

The following example shows the contents of an IKEErrors.txt file. Your errors may be different depending on the issue.

```
Error: Authentication failed. Check shared key. Check crypto. Check lifetimes.
      based on log : Peer failed with Windows error 13801(ERROR_IPSEC_IKE_AUTH_FAIL)
Error: On-prem device sent invalid payload.
      based on log : IkeFindPayloadInPacket failed with Windows error 13843(ERROR_IPSEC_IKE_INVALID_PAYLOAD)
```

Scrubbed-wfpdiag.txt

The **Scrubbed-wfpdiag.txt** log file contains the wfp log. This log contains logging of packet drop and IKE/AuthIP failures.

The following example shows the contents of the Scrubbed-wfpdiag.txt file. In this example, the shared key of a Connection was not correct as can be seen from the third line from the bottom. The following example is just a snippet of the entire log, as the log can be lengthy depending on the issue.

```
...
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36|Deleted ICookie from the high priority thread pool list
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36|IKE diagnostic event:
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36|Event Header:
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Timestamp: 1601-01-01T00:00:00.000Z
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Flags: 0x00000106
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Local address field set
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Remote address field set
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| IP version field set
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| IP version: IPv4
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| IP protocol: 0
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Local address: 13.78.238.92
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Remote address: 52.161.24.36
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Local Port: 0
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Remote Port: 0
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Application ID:
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| User SID: <invalid>
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36|Failure type: IKE/Authip Main Mode Failure
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36|Type specific info:
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Failure error code:0x000035e9
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| IKE authentication credentials are unacceptable
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36|
[0]0368.03A4::02/02/2017-17:36:01.496 [ikeext] 3038|52.161.24.36| Failure point: Remote
...
...
```

wfpdiag.txt.sum

The **wfpdiag.txt.sum** file is a log showing the buffers and events processed.

The following example is the contents of a wfpdiag.txt.sum file.

```

Files Processed:
C:\Resources\directory\924336c47dd045d5a246c349b8ae57f2.GatewayTenantWorker.DiagnosticsStorage\2017-02-02T17-
34-23\wfpdiag.etl
Total Buffers Processed 8
Total Events Processed 2169
Total Events Lost 0
Total Format Errors 0
Total Formats Unknown 486
Elapsed Time 330 sec
+-----+
|EventCount EventName EventType TMF
+-----+
| 36 ikeext ike_addr_utils_c844 a0c064ca-d954-350a-8b2f-1a7464eef8b6|
| 12 ikeext ike_addr_utils_c857 a0c064ca-d954-350a-8b2f-1a7464eef8b6|
| 96 ikeext ike_addr_utils_c832 a0c064ca-d954-350a-8b2f-1a7464eef8b6|
| 6 ikeext ike_bfe_callbacks_c133 1dc2d67f-8381-6303-e314-6c1452eeb529|
| 6 ikeext ike_bfe_callbacks_c61 1dc2d67f-8381-6303-e314-6c1452eeb529|
| 12 ikeext ike_sa_management_c5698 7857a320-42ee-6e90-d5d9-3f414e3ea2d3|
| 6 ikeext ike_sa_management_c8447 7857a320-42ee-6e90-d5d9-3f414e3ea2d3|
| 12 ikeext ike_sa_management_c494 7857a320-42ee-6e90-d5d9-3f414e3ea2d3|
| 12 ikeext ike_sa_management_c642 7857a320-42ee-6e90-d5d9-3f414e3ea2d3|
| 6 ikeext ike_sa_management_c3162 7857a320-42ee-6e90-d5d9-3f414e3ea2d3|
| 12 ikeext ike_sa_management_c3307 7857a320-42ee-6e90-d5d9-3f414e3ea2d3|

```

Next steps

To learn how to diagnose a problem with a gateway or gateway connection, see [Diagnose communication problems between networks](#).

Introduction to variable packet capture in Azure Network Watcher

1/28/2020 • 2 minutes to read • [Edit Online](#)

Network Watcher variable packet capture allows you to create packet capture sessions to track traffic to and from a virtual machine. Packet capture helps to diagnose network anomalies both reactively and proactively. Other uses include gathering network statistics, gaining information on network intrusions, to debug client-server communications and much more.

Packet capture is a virtual machine extension that is remotely started through Network Watcher. This capability eases the burden of running a packet capture manually on the desired virtual machine, which saves valuable time. Packet capture can be triggered through the portal, PowerShell, CLI, or REST API. One example of how packet capture can be triggered is with Virtual Machine alerts. Filters are provided for the capture session to ensure you capture traffic you want to monitor. Filters are based on 5-tuple (protocol, local IP address, remote IP address, local port, and remote port) information. The captured data is stored in the local disk or a storage blob. There is a limit of 10 packet capture sessions per region per subscription. This limit applies only to the sessions and does not apply to the saved packet capture files either locally on the VM or in a storage account.

IMPORTANT

Packet capture requires a virtual machine extension [AzureNetworkWatcherExtension](#). For installing the extension on a Windows VM visit [Azure Network Watcher Agent virtual machine extension for Windows](#) and for Linux VM visit [Azure Network Watcher Agent virtual machine extension for Linux](#).

To reduce the information you capture to only the information you want, the following options are available for a packet capture session:

Capture configuration

PROPERTY	DESCRIPTION
Maximum bytes per packet (bytes)	The number of bytes from each packet that are captured, all bytes are captured if left blank. The number of bytes from each packet that are captured, all bytes are captured if left blank. If you need only the IPv4 header – indicate 34 here
Maximum bytes per session (bytes)	Total number of bytes in that are captured, once the value is reached the session ends.
Time limit (seconds)	Sets a time constraint on the packet capture session. The default value is 18000 seconds or 5 hours.

Filtering (optional)

PROPERTY	DESCRIPTION
Protocol	The protocol to filter for the packet capture. The available values are TCP, UDP, and All.

PROPERTY	DESCRIPTION
Local IP address	This value filters the packet capture to packets where the local IP address matches this filter value.
Local port	This value filters the packet capture to packets where the local port matches this filter value.
Remote IP address	This value filters the packet capture to packets where the remote IP matches this filter value.
Remote port	This value filters the packet capture to packets where the remote port matches this filter value.

Next steps

Learn how you can manage packet captures through the portal by visiting [Manage packet capture in the Azure portal](#) or with PowerShell by visiting [Manage Packet Capture with PowerShell](#).

Learn how to create proactive packet captures based on virtual machine alerts by visiting [Create an alert triggered packet capture](#)

Introduction to flow logging for network security groups

2/27/2020 • 7 minutes to read • [Edit Online](#)

Network security group (NSG) flow logs are a feature of Network Watcher that allows you to view information about ingress and egress IP traffic through an NSG. Flow logs are written in JSON format, and show outbound and inbound flows on a per rule basis, the network interface (NIC) the flow applies to, 5-tuple information about the flow (Source/destination IP, source/destination port, and protocol), if the traffic was allowed or denied, and in Version 2, throughput information (Bytes and Packets).

The screenshot shows the 'Network Watcher - NSG flow logs' interface. On the left, there's a navigation sidebar with sections like Overview, MONITORING (Topology), NETWORK DIAGNOSTIC TOOLS (IP flow verify, Next hop, Security group view, Packet capture), METRICS (Network subscription limit), and LOGS (NSG flow logs, Diagnostic logs). The 'NSG flow logs' item is highlighted. The main area displays a table of NSGs:

NAME	RESOURCE TYPE	RESOURCE GROUP	STATUS	STORAGE ACCOUNT
webtestnsg-c3dxj32iloqq-	Network security group	ContosoAppGateway	Disabled	
webtestnsg-h7tfripb4hd-	Network security group	ContosoAppGateway	Enabled	webtestvhdc3dxj32iloqqo
fabrikmvm1-nsg	Network security group	FabrikamRG	Disabled	
fabrikamvm3-nsg	Network security group	FabrikamRG	Enabled	webtestvhdc3dxj32iloqqo
fabrikamvm4-nsg	Network security group	FabrikamRG	Disabled	
webtestnsg-r5wpjct4pltz-	Network security group	testresourcegroup	Disabled	
webtestnsg-xqpow6s7bp-	Network security group	testresourcegroup	Disabled	

While flow logs target NSGs, they are not displayed the same as the other logs. Flow logs are stored only within a storage account and follow the logging path shown in the following example:

```
https://{{storageAccountName}}.blob.core.windows.net/insights-logs-
networksecuritygroupflowevent/resourceId=/SUBSCRIPTIONS/{{subscriptionID}}/RESOURCEGROUPS/{{resourceGroupName}}/
PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/{{nsgName}}/y={{year}}/m={{month}}/d={{day}}/h=
{{hour}}/m=00/macAddress={{macAddress}}/PT1H.json
```

You can analyze flow logs and gain insights into your network traffic using [traffic analytics](#).

The same retention policies seen for other logs apply to flow logs. You can set log retention policy from 1 day to 365 days. If a retention policy is not set, the logs are maintained forever.

Log file

Flow logs include the following properties:

- **time** - Time when the event was logged
- **systemId** - Network Security Group resource Id.
- **category** - The category of the event. The category is always **NetworkSecurityGroupFlowEvent**
- **resourceid** - The resource Id of the NSG
- **operationName** - Always NetworkSecurityGroupFlowEvents

- **properties** - A collection of properties of the flow
 - **Version** - Version number of the Flow Log event schema
 - **flows** - A collection of flows. This property has multiple entries for different rules
 - **rule** - Rule for which the flows are listed
 - **flows** - a collection of flows
 - **mac** - The MAC address of the NIC for the VM where the flow was collected
 - **flowTuples** - A string that contains multiple properties for the flow tuple in comma-separated format
 - **Time Stamp** - This value is the time stamp of when the flow occurred in UNIX epoch format
 - **Source IP** - The source IP
 - **Destination IP** - The destination IP
 - **Source Port** - The source port
 - **Destination Port** - The destination Port
 - **Protocol** - The protocol of the flow. Valid values are **T** for TCP and **U** for UDP
 - **Traffic Flow** - The direction of the traffic flow. Valid values are **I** for inbound and **O** for outbound.
 - **Traffic Decision** - Whether traffic was allowed or denied. Valid values are **A** for allowed and **D** for denied.
 - **Flow State - Version 2 Only** - Captures the state of the flow. Possible states are **B**: Begin, when a flow is created. Statistics aren't provided. **C**: Continuing for an ongoing flow. Statistics are provided at 5-minute intervals. **E**: End, when a flow is terminated. Statistics are provided.
 - **Packets - Source to destination - Version 2 Only** The total number of TCP or UDP packets sent from source to destination since last update.
 - **Bytes sent - Source to destination - Version 2 Only** The total number of TCP or UDP packet bytes sent from source to destination since last update. Packet bytes include the packet header and payload.
 - **Packets - Destination to source - Version 2 Only** The total number of TCP or UDP packets sent from destination to source since last update.
 - **Bytes sent - Destination to source - Version 2 Only** The total number of TCP and UDP packet bytes sent from destination to source since last update. Packet bytes include packet header and payload.

NSG flow logs version 2

Version 2 of the logs introduces flow state. You can configure which version of flow logs you receive. To learn how to enable flow logs, see [Enabling NSG flow logging](#).

Flow state *B* is recorded when a flow is initiated. Flow state *C* and flow state *E* are states that mark the continuation of a flow and flow termination, respectively. Both *C* and *E* states contain traffic bandwidth information.

Example: Flow tuples from a TCP conversation between 185.170.185.105:35370 and 10.2.0.4:23:

```
"1493763938,185.170.185.105,10.2.0.4,35370,23,T,I,A,B,",
"1493695838,185.170.185.105,10.2.0.4,35370,23,T,I,A,C,1021,588096,8005,4610880",
"1493696138,185.170.185.105,10.2.0.4,35370,23,T,I,A,E,52,29952,47,27072"
```

For continuation *C* and end *E* flow states, byte and packet counts are aggregate counts from the time of the previous flow tuple record. Referencing the previous example conversation, the total number of packets

transferred is $1021+52+8005+47 = 9125$. The total number of bytes transferred is $588096+29952+4610880+27072 = 5256000$.

The text that follows is an example of a flow log. As you can see, there are multiple records that follow the property list described in the preceding section.

NSG flow logging considerations

Storage account considerations:

- Location: The storage account used must be in the same region as the NSG.
- Self-manage key rotation: If you change/rotate the access keys to your storage account, NSG Flow Logs will stop working. To fix this issue, you must disable and then re-enable NSG Flow Logs.

Enable NSG Flow Logging on all NSGs attached to a resource: Flow logging in Azure is configured on the NSG resource. A flow will only be associated to one NSG Rule. In scenarios where multiple NSGs are utilized, we recommend that NSG flow logging is enabled on all NSGs applied a resource's subnet or network interface to ensure that all traffic is recorded. For more information see [how traffic is evaluated](#) in Network Security Groups.

Flow Logging Costs: NSG flow logging is billed on the volume of logs produced. High traffic volume can result in large flow log volume and the associated costs. NSG Flow log pricing does not include the underlying costs of storage. Using the retention policy feature with NSG Flow Logging may result in a high volume of storage operations and the associated costs. If you do not require the retention policy feature, we recommend that you set this value to 0. For more information, see [Network Watcher Pricing](#) and [Azure Storage Pricing](#) for additional details.

Inbound flows logged from internet IPs to VMs without public IPs: VMs that don't have a public IP address assigned via a public IP address associated with the NIC as an instance-level public IP, or that are part of a basic load balancer back-end pool, use [default SNAT](#) and have an IP address assigned by Azure to facilitate outbound connectivity. As a result, you might see flow log entries for flows from internet IP addresses, if the flow is destined to a port in the range of ports assigned for SNAT. While Azure won't allow these flows to the VM, the attempt is logged and appears in Network Watcher's NSG flow log by design. We recommend that unwanted inbound internet traffic be explicitly blocked with NSG.

Incorrect byte and packet counts for Stateless flows: [Network Security Groups \(NSGs\)](#) are implemented as a [Stateful firewall](#). However many default/internal rules that control the flow of traffic are implemented in a stateless fashion. Due to platform limitations, the bytes and packets counts are not recorded for stateless flows (that is, traffic flows going through stateless rules), they are recorded only for stateful flows. Consequently the number of bytes and packets reported in NSG Flow Logs (and Traffic Analytics) could be different from actual flows. This limitation is scheduled to be fixed by June 2020.

Sample log records

The text that follows is an example of a flow log. As you can see, there are multiple records that follow the property list described in the preceding section.

NOTE

Values in the `*flowTuples` property are a comma-separated list.

Version 1 NSG flow log format sample

```
{  
    "records": [  
        {
```

```
    "time": "2017-02-16T22:00:32.895000Z",
    "systemId": "2c002c16-72f3-4dc5-b391-3444c3527434",
    "category": "NetworkSecurityGroupFlowEvent",
    "resourceId": "/SUBSCRIPTIONS/00000000-0000-0000-0000-
000000000000/RESOURCEGROUPS/FABRIKAMRG/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/FABRIAKMVM1-NSG",
    "operationName": "NetworkSecurityGroupFlowEvents",
    "properties": {
        "Version": 1,
        "flows": [
            {
                "rule": "DefaultRule_DenyAllInBound",
                "flows": [
                    {
                        "mac": "000D3AF8801A",
                        "flowTuples": [
                            "1487282421,42.119.146.95,10.1.0.4,51529,5358,T,I,D"
                        ]
                    }
                ]
            },
            {
                "rule": "UserRule_default-allow-rdp",
                "flows": [
                    {
                        "mac": "000D3AF8801A",
                        "flowTuples": [
                            "1487282370,163.28.66.17,10.1.0.4,61771,3389,T,I,A",
                            "1487282393,5.39.218.34,10.1.0.4,58596,3389,T,I,A",
                            "1487282393,91.224.160.154,10.1.0.4,61540,3389,T,I,A",
                            "1487282423,13.76.89.229,10.1.0.4,53163,3389,T,I,A"
                        ]
                    }
                ]
            }
        ]
    }
},
{
    "time": "2017-02-16T22:01:32.896000Z",
    "systemId": "2c002c16-72f3-4dc5-b391-3444c3527434",
    "category": "NetworkSecurityGroupFlowEvent",
    "resourceId": "/SUBSCRIPTIONS/00000000-0000-0000-0000-
000000000000/RESOURCEGROUPS/FABRIKAMRG/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/FABRIAKMVM1-NSG",
    "operationName": "NetworkSecurityGroupFlowEvents",
    "properties": {
        "Version": 1,
        "flows": [
            {
                "rule": "DefaultRule_DenyAllInBound",
                "flows": [
                    {
                        "mac": "000D3AF8801A",
                        "flowTuples": [
                            "1487282481,195.78.210.194,10.1.0.4,53,1732,U,I,D"
                        ]
                    }
                ]
            },
            {
                "rule": "UserRule_default-allow-rdp",
                "flows": [
                    {
                        "mac": "000D3AF8801A",
                        "flowTuples": [
                            "1487282435,61.129.251.68,10.1.0.4,57776,3389,T,I,A",
                            "1487282454,84.25.174.170,10.1.0.4,59085,3389,T,I,A",
                            "1487282477,77.68.9.50,10.1.0.4,65078,3389,T,I,A"
                        ]
                    }
                ]
            }
        ]
    }
},
```

```

        ]
    }
}
],
"records": [
{
    "time": "2017-02-16T22:00:32.895000Z",
    "systemId": "2c002c16-72f3-4dc5-b391-3444c3527434",
    "category": "NetworkSecurityGroupFlowEvent",
    "resourceId": "/SUBSCRIPTIONS/00000000-0000-0000-
000000000000/RESOURCEGROUPS/FABRIKAMRG/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/FABRIAKMVM1-NSG",
    "operationName": "NetworkSecurityGroupFlowEvents",
    "properties": {"Version":1,"flows": [{"rule": "DefaultRule_DenyAllInBound","flows": [{"mac": "000D3AF8801A","flowTuples": [{"1487282421,42.119.146.95,10.1.0.4,51529,5358,T,I,D"}]}], "rule": "UserRule_default-allow-rdp","flows": [{"mac": "000D3AF8801A","flowTuples": [{"1487282370,163.28.66.17,10.1.0.4,61771,3389,T,I,A"}, {"1487282393,5.39.218.34,10.1.0.4,58596,3389,T,I,A"}, {"1487282393,91.224.160.154,10.1.0.4,61540,3389,T,I,A"}, {"1487282423,13.76.89.229,10.1.0.4,53163,3389,T,I,A"}]}]}},
    },
    {
        "time": "2017-02-16T22:01:32.896000Z",
        "systemId": "2c002c16-72f3-4dc5-b391-3444c3527434",
        "category": "NetworkSecurityGroupFlowEvent",
        "resourceId": "/SUBSCRIPTIONS/00000000-0000-0000-
000000000000/RESOURCEGROUPS/FABRIKAMRG/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/FABRIAKMVM1-NSG",
        "operationName": "NetworkSecurityGroupFlowEvents",
        "properties": {"Version":1,"flows": [{"rule": "DefaultRule_DenyAllInBound","flows": [{"mac": "000D3AF8801A","flowTuples": [{"1487282481,195.78.210.194,10.1.0.4,53,1732,U,I,D"}]}], "rule": "UserRule_default-allow-rdp","flows": [{"mac": "000D3AF8801A","flowTuples": [{"1487282435,61.129.251.68,10.1.0.4,57776,3389,T,I,A"}, {"1487282454,84.25.174.170,10.1.0.4,59085,3389,T,I,A"}, {"1487282477,77.68.9.50,10.1.0.4,65078,3389,T,I,A"}]}]}},
    },
    {
        "time": "2017-02-16T22:02:32.904000Z",
        "systemId": "2c002c16-72f3-4dc5-b391-3444c3527434",
        "category": "NetworkSecurityGroupFlowEvent",
        "resourceId": "/SUBSCRIPTIONS/00000000-0000-0000-
000000000000/RESOURCEGROUPS/FABRIKAMRG/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/FABRIAKMVM1-NSG",
        "operationName": "NetworkSecurityGroupFlowEvents",
        "properties": {"Version":1,"flows": [{"rule": "DefaultRule_DenyAllInBound","flows": [{"mac": "000D3AF8801A","flowTuples": [{"1487282492,175.182.69.29,10.1.0.4,28918,5358,T,I,D"}, {"1487282505,71.6.216.55,10.1.0.4,8080,8080,T,I,D"}]}], "rule": "UserRule_default-allow-rdp","flows": [{"mac": "000D3AF8801A","flowTuples": [{"1487282512,91.224.160.154,10.1.0.4,59046,3389,T,I,A"}]}]}},
    },
    ...
}
]

```

Version 2 NSG flow log format sample

```

{
    "records": [
{
    "time": "2018-11-13T12:00:35.389926Z",
    "systemId": "a0fca5ce-022c-47b1-9735-89943b42f2fa",
    "category": "NetworkSecurityGroupFlowEvent",
    "resourceId": "/SUBSCRIPTIONS/00000000-0000-0000-
000000000000/RESOURCEGROUPS/FABRIKAMRG/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/FABRIAKMVM1-NSG",
    "operationName": "NetworkSecurityGroupFlowEvents",
    "properties": {
        "Version": 2,
        "flows": [
        ...
        ]
    }
}
]

```

```

        "rule": "DefaultRule_DenyAllInBound",
        "flows": [
            {
                "mac": "000D3AF87856",
                "flowTuples": [
                    "1542110402,94.102.49.190,10.5.16.4,28746,443,U,I,D,B,,,",
                    "1542110424,176.119.4.10,10.5.16.4,56509,59336,T,I,D,B,,,",
                    "1542110432,167.99.86.8,10.5.16.4,48495,8088,T,I,D,B,,,"
                ]
            }
        ],
    },
    {
        "rule": "DefaultRule_AllowInternetOutBound",
        "flows": [
            {
                "mac": "000D3AF87856",
                "flowTuples": [
                    "1542110377,10.5.16.4,13.67.143.118,59831,443,T,O,A,B,,,",
                    "1542110379,10.5.16.4,13.67.143.117,59932,443,T,O,A,E,1,66,1,66",
                    "1542110379,10.5.16.4,13.67.143.115,44931,443,T,O,A,C,30,16978,24,14008",
                    "1542110406,10.5.16.4,40.71.12.225,59929,443,T,O,A,E,15,8489,12,7054"
                ]
            }
        ],
    },
    {
        "time": "2018-11-13T12:01:35.3918317Z",
        "systemId": "a0fcfa5ce-022c-47b1-9735-89943b42f2fa",
        "category": "NetworkSecurityGroupFlowEvent",
        "resourceId": "/SUBSCRIPTIONS/00000000-0000-0000-0000-
000000000000/RESOURCEGROUPS/FABRIKAMRG/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/FABRIAKMVM1-NSG",
        "operationName": "NetworkSecurityGroupFlowEvents",
        "properties": {
            "Version": 2,
            "flows": [
                {
                    "rule": "DefaultRule_DenyAllInBound",
                    "flows": [
                        {
                            "mac": "000D3AF87856",
                            "flowTuples": [
                                "1542110437,125.64.94.197,10.5.16.4,59752,18264,T,I,D,B,,,",
                                "1542110475,80.211.72.221,10.5.16.4,37433,8088,T,I,D,B,,,",
                                "1542110487,46.101.199.124,10.5.16.4,60577,8088,T,I,D,B,,,",
                                "1542110490,176.119.4.30,10.5.16.4,57067,52801,T,I,D,B,,,"
                            ]
                        }
                    ]
                }
            ]
        }
    },
    ...

```

Next steps

- To learn how to enable flow logs, see [Enabling NSG flow logging](#).
- To learn how to read the flow logs, see [Read NSG flow logs](#).
- To learn more about NSG logging, see [Azure Monitor logs for network security groups \(NSGs\)](#).
- To determine whether traffic is allowed or denied to or from a VM, see [Diagnose a VM network traffic filter](#)

problem

Introduction to Effective security rules view in Azure Network Watcher

1/28/2020 • 2 minutes to read • [Edit Online](#)

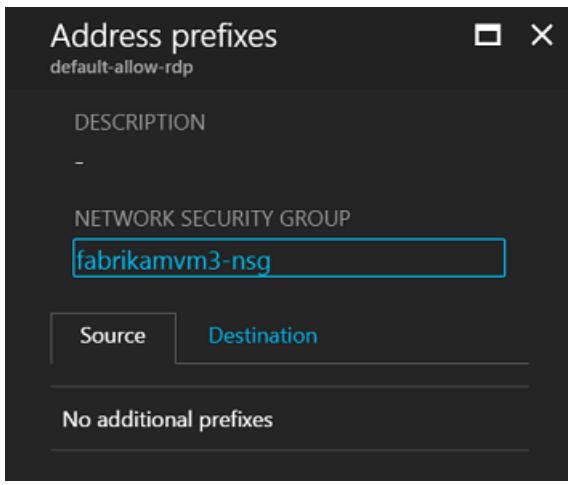
Network Security groups are associated at a subnet level or at a NIC level. When associated at a subnet level, it applies to all the VM instances in the subnet. Effective security rules view returns all the configured NSGs and rules that are associated at a NIC and subnet level for a virtual machine providing insight into the configuration. In addition, the effective security rules are returned for each of the NICs in a VM. Using Effective security rules view, you can assess a VM for network vulnerabilities such as open ports. You can also validate if your Network Security Group is working as expected based on a [comparison between the configured and the approved security rules](#).

A more extended use case is in security compliance and auditing. You can define a prescriptive set of security rules as a model for security governance in your organization. A periodic compliance audit can be implemented in a programmatic way by comparing the prescriptive rules with the effective rules for each of the VMs in your network.

In the portal rules are displayed for each Network Interface and grouped by inbound vs outbound. This provides a simple view into the rules applied to a virtual machine. A download button is provided to easily download all the security rules no matter the tab into a CSV file.

The screenshot shows the 'Network Watcher - Security group view' blade in the Azure portal. The left sidebar includes links for Overview, MONITORING, Topology, NETWORK DIAGNOSTIC TOOLS (IP flow verify, Next hop, Security group view, Packet capture), METRICS, LOGS (NSG flow logs, Diagnostic logs), and a search bar. The main area has a 'Download' button and filters for Subscription (Microsoft Azure), Resource group (FabrikamRG), and Virtual machine (fabriakvm1). It displays two tabs: 'Effective' and 'Subnet'. The 'Effective' tab is selected, showing 'Inbound rules' and 'Outbound rules' tables. The 'Inbound rules' table lists several rules, including 'DefaultInboundDenyAll' (Priority 65500, Deny) and 'UserRule_default-allow-rdp' (Priority 1000, Allow). The 'Outbound rules' table lists rules like 'DefaultOutboundDenyAll' (Priority 65500, Deny) and 'DefaultRule_AllowNetOut...' (Priority 65000, Allow).

Rules can be selected and a new blade opens up to show the Network Security Group and source and destination prefixes. From this blade you can navigate directly to the Network Security Group resource.



Next steps

Learn how to audit your Network Security Group settings by visiting [Audit Network Security Group settings with PowerShell](#)

Role-based access control permissions required to use Network Watcher capabilities

1/28/2020 • 2 minutes to read • [Edit Online](#)

Azure role-based access control (RBAC) enables you to assign only the specific actions to members of your organization that they require to complete their assigned responsibilities. To use Network Watcher capabilities, the account you log into Azure with, must be assigned to the [Owner](#), [Contributor](#), or [Network contributor](#) built-in roles, or assigned to a [custom role](#) that is assigned the actions listed for each Network Watcher capability in the sections that follow. To learn more about Network Watcher's capabilities, see [What is Network Watcher?](#).

Network Watcher

ACTION	DESCRIPTION
Microsoft.Network/networkWatchers/read	Get a network watcher
Microsoft.Network/networkWatchers/write	Create or update a network watcher
Microsoft.Network/networkWatchers/delete	Delete a network watcher

NSG flow logs

ACTION	DESCRIPTION
Microsoft.Network/networkWatchers/configureFlowLog/action	Configure a flow Log
Microsoft.Network/networkWatchers/queryFlowLogStatus/acti on	Query status for a flow log

Connection troubleshoot

ACTION	DESCRIPTION
Microsoft.Network/networkWatchers/connectivityCheck/actio n	Initiate a connection troubleshoot test
Microsoft.Network/networkWatchers/queryTroubleshootResul t/action	Query results of a connection troubleshoot test
Microsoft.Network/networkWatchers/troubleshoot/action	Run a connection troubleshoot test

Connection monitor

ACTION	DESCRIPTION
Microsoft.Network/networkWatchers/connectionMonitors/start/action	Start a connection monitor
Microsoft.Network/networkWatchers/connectionMonitors/stop/action	Stop a connection monitor
Microsoft.Network/networkWatchers/connectionMonitors/query/action	Query a connection monitor
Microsoft.Network/networkWatchers/connectionMonitors/read	Get a connection monitor
Microsoft.Network/networkWatchers/connectionMonitors/write	Create a connection monitor
Microsoft.Network/networkWatchers/connectionMonitors/delete	Delete a connection monitor

Packet capture

ACTION	DESCRIPTION
Microsoft.Network/networkWatchers/packetCaptures/queryStatus/action	Query the status of a packet capture
Microsoft.Network/networkWatchers/packetCaptures/stop/action	Stop a packet capture
Microsoft.Network/networkWatchers/packetCaptures/read	Get a packet capture
Microsoft.Network/networkWatchers/packetCaptures/write	Create a packet capture
Microsoft.Network/networkWatchers/packetCaptures/delete	Delete a packet capture

IP flow verify

ACTION	DESCRIPTION
Microsoft.Network/networkWatchers/ipFlowVerify/action	Verify an IP flow

Next hop

ACTION	DESCRIPTION
Microsoft.Network/networkWatchers/nextHop/action	Get the next hop from a VM

Network security group view

ACTION	DESCRIPTION
Microsoft.Network/networkWatchers/securityGroupView/action	View security groups

Topology

ACTION	DESCRIPTION
Microsoft.Network/networkWatchers/topology/action	Get topology

Reachability report

ACTION	DESCRIPTION
Microsoft.Network/networkWatchers/azureReachabilityReport /action	Get an Azure reachability report

Additional actions

Network Watcher capabilities also require the following actions:

ACTION(S)	DESCRIPTION
Microsoft.Authorization/*/Read	Used to fetch RBAC role assignments and policy definitions
Microsoft.Resources/subscriptions/resourceGroups/Read	Used to enumerate all the resource groups in a subscription
Microsoft.Storage/storageAccounts/Read	Used to get the properties for the specified storage account
Microsoft.Storage/storageAccounts/listServiceSas/Action, Microsoft.Storage/storageAccounts/listAccountSas/Action, Microsoft.Storage/storageAccounts/listKeys/Action	Used to fetch shared access signatures (SAS) enabling secure access to storage account and write to the storage account
Microsoft.Compute/virtualMachines/Read, Microsoft.Compute/virtualMachines/Write	Used to log in to the VM, do a packet capture and upload it to storage account
Microsoft.Compute/virtualMachines/extensions/Read Microsoft.Compute/virtualMachines/extensions/Write	Used to check if Network Watcher extension is present, and install if required
Microsoft.Compute/virtualMachineScaleSets/Read, Microsoft.Compute/virtualMachineScaleSets/Write	Used to access virtual machine scale sets, do packet captures and upload them to storage account
Microsoft.Compute/virtualMachineScaleSets/extensions/Read, Microsoft.Compute/virtualMachineScaleSets/extensions/Write	Used to check if Network Watcher extension is present, and install if required
Microsoft.Insights/alertRules/*	Used to set up metric alerts
Microsoft.Support/*	Used to create and update support tickets from Network Watcher

Create an Azure Network Watcher instance

2/12/2020 • 3 minutes to read • [Edit Online](#)

Network Watcher is a regional service that enables you to monitor and diagnose conditions at a network scenario level in, to, and from Azure. Scenario level monitoring enables you to diagnose problems at an end to end network level view. Network diagnostic and visualization tools available with Network Watcher help you understand, diagnose, and gain insights to your network in Azure. Network Watcher is enabled through the creation of a Network Watcher resource. This resource allows you to utilize Network Watcher capabilities.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Network Watcher is automatically enabled

When you create or update a virtual network in your subscription, Network Watcher will be enabled automatically in your Virtual Network's region. There is no impact to your resources or associated charge for automatically enabling Network Watcher.

Opt-out of Network Watcher automatic enablement

If you would like to opt out of Network Watcher automatic enablement, you can do so by running the following commands:

WARNING

Opting-out of Network Watcher automatic enablement is a permanent change. Once you opt-out you cannot opt-in without [contacting support](#)

```
Register-AzProviderFeature -FeatureName DisableNetworkWatcherAutocreation -ProviderNamespace Microsoft.Network  
Register-AzResourceProvider -ProviderNamespace Microsoft.Network
```

```
az feature register --name DisableNetworkWatcherAutocreation --namespace Microsoft.Network  
az provider register -n Microsoft.Network
```

Create a Network Watcher in the portal

Navigate to **All Services > Networking > Network Watcher**. You can select all the subscriptions you want to enable Network Watcher for. This action creates a Network Watcher in every region that is available.

The screenshot shows the Azure portal's Network Watcher Overview page. On the left, there's a sidebar with navigation links for Overview, MONITORING (Topology), NETWORK DIAGNOSTIC TOOLS (IP flow verify, Next hop, Security group view, Packet capture), METRICS, LOGS (NSG flow logs, Diagnostic logs), and a Microsoft Azure Internal Consumption section. The main area displays a table of subscriptions under the heading 'Subscriptions: 1 of 2 selected – Don't see a subscription? Switch directories'. The table has columns for NAME, REGION, and STATUS. It lists 'Microsoft Azure Internal Consumption' with regions 'West US', 'North Central US', and 'West Central US'. The 'West Central US' row is highlighted with a red box around the 'Enable network watcher' button. Below the table, a 'Details' panel shows the status as 'Disabled', region as 'North Central US', and subscription as 'Microsoft Azure Internal Consumption'. There are also 'Edit' and 'Delete' buttons at the bottom of the details panel.

When you enable Network Watcher using the portal, the name of the Network Watcher instance is automatically set to *NetworkWatcher_region_name* where *region_name* corresponds to the Azure region where the instance is enabled. For example, a Network Watcher enabled in the West Central US region is named *NetworkWatcher_westcentralus*.

The Network Watcher instance is automatically created in a resource group named *NetworkWatcherRG*. The resource group is created if it does not already exist.

If you wish to customize the name of a Network Watcher instance and the resource group it's placed into, you can use Powershell, the Azure CLI, the REST API, or ARMClient methods described in the sections that follow. In each option, the resource group must exist before you create a Network Watcher in it.

Create a Network Watcher with PowerShell

To create an instance of Network Watcher, run the following example:

```
New-AzNetworkWatcher -Name "NetworkWatcher_westcentralus" -ResourceGroupName "NetworkWatcherRG" -Location "West Central US"
```

Create a Network Watcher with the Azure CLI

To create an instance of Network Watcher, run the following example:

```
az network watcher configure --resource-group NetworkWatcherRG --locations westcentralus --enabled
```

Create a Network Watcher with the REST API

The ARMclient is used to call the REST API using PowerShell. The ARMClient is found on chocolatey at [ARMClient on Chocolatey](#)

Log in with ARMClient

```
armclient login
```

Create the network watcher

```
$subscriptionId = '<subscription id>'  
$networkWatcherName = '<name of network watcher>'  
$resourceGroupName = '<resource group name>'  
$apiversion = "2016-09-01"  
$requestBody = @"  
{  
    'location': 'West Central US'  
}  
"@  
  
armclient put  
"https://management.azure.com/subscriptions/${subscriptionId}/resourceGroups/${resourceGroupName}/providers/Microsoft.Network/networkWatchers/${networkWatcherName}?api-version=${api-version}" $requestBody
```

Delete a Network Watcher in the portal

Navigate to **All Services > Networking > Network Watcher**.

Select the overview tab, if you're not already there. Use the dropdown to select the subscription you want to disable network watcher in. Expand the list of regions for your chosen subscription by clicking on the arrow. For any given, use the 3 dots on the right to access the context menu. Click on "Disable network watcher" to start disabling. You will be asked to confirm this step. Click Yes to continue. On the portal, you will have to do this individually for every region in every subscription.

Delete a Network Watcher with PowerShell

To delete an instance of Network Watcher, run the following example:

```
New-AzResourceGroup -Name NetworkWatcherRG -Location westcentralus  
New-AzNetworkWatcher -Name NetworkWatcher_westcentralus -ResourceGroup NetworkWatcherRG -Location  
westcentralus  
Remove-AzNetworkWatcher -Name NetworkWatcher_westcentralus -ResourceGroup NetworkWatcherRG
```

Next steps

Now that you have an instance of Network Watcher, learn about the features available:

- [Topology](#)
- [Packet capture](#)
- [IP flow verify](#)
- [Next hop](#)
- [Security group view](#)
- [NSG flow logging](#)
- [Virtual Network Gateway troubleshooting](#)

2 minutes to read

2 minutes to read

Diagnose a virtual machine network routing problem - Azure PowerShell

1/28/2020 • 5 minutes to read • [Edit Online](#)

In this article, you deploy a virtual machine (VM), and then check communications to an IP address and URL. You determine the cause of a communication failure and how you can resolve it.

If you don't have an Azure subscription, create a [free account](#) before you begin.

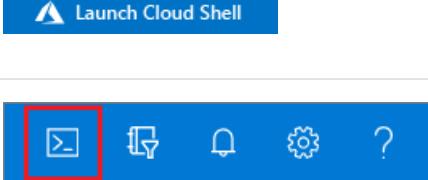
NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell `Az` module. To find the installed version, run `Get-Module -ListAvailable Az`. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

Create a VM

Before you can create a VM, you must create a resource group to contain the VM. Create a resource group with [New-AzResourceGroup](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location.

```
New-AzResourceGroup -Name myResourceGroup -Location EastUS
```

Create the VM with [New-AzVM](#). When running this step, you are prompted for credentials. The values that you enter are configured as the user name and password for the VM.

```
$vM = New-AzVm `  
-ResourceGroupName "myResourceGroup" `  
-Name "myVm" `  
-Location "East US"
```

The VM takes a few minutes to create. Don't continue with remaining steps until the VM is created and PowerShell returns output.

Test network communication

To test network communication with Network Watcher, you must first enable a network watcher in the region the VM that you want to test is in, and then use Network Watcher's next hop capability to test communication.

Enable network watcher

If you already have a network watcher enabled in the East US region, use [Get-AzNetworkWatcher](#) to retrieve the network watcher. The following example retrieves an existing network watcher named *NetworkWatcher_eastus* that is in the *NetworkWatcherRG* resource group:

```
$networkWatcher = Get-AzNetworkWatcher `  
-Name NetworkWatcher_eastus `  
-ResourceGroupName NetworkWatcherRG
```

If you don't already have a network watcher enabled in the East US region, use [New-AzNetworkWatcher](#) to create a network watcher in the East US region:

```
$networkWatcher = New-AzNetworkWatcher `  
-Name "NetworkWatcher_eastus" `  
-ResourceGroupName "NetworkWatcherRG" `  
-Location "East US"
```

Use next hop

Azure automatically creates routes to default destinations. You may create custom routes that override the default routes. Sometimes, custom routes can cause communication to fail. To test routing from a VM, use the [Get-AzNetworkWatcherNextHop](#) command to determine the next routing hop when traffic is destined for a specific address.

Test outbound communication from the VM to one of the IP addresses for www.bing.com:

```
Get-AzNetworkWatcherNextHop ` 
-NetworkWatcher $networkWatcher ` 
-TargetVirtualMachineId $VM.Id ` 
-SourceIPAddress 192.168.1.4 ` 
-DestinationIPAddress 13.107.21.200
```

After a few seconds, the output informs you that the **NextHopType** is **Internet**, and that the **RouteTableId** is **System Route**. This result lets you know that there is a valid route to the destination.

Test outbound communication from the VM to 172.31.0.100:

```
Get-AzNetworkWatcherNextHop ` 
-NetworkWatcher $networkWatcher ` 
-TargetVirtualMachineId $VM.Id ` 
-SourceIPAddress 192.168.1.4 ` 
-DestinationIPAddress 172.31.0.100
```

The output returned informs you that **None** is the **NextHopType**, and that the **RouteTableId** is also **System Route**. This result lets you know that, while there is a valid system route to the destination, there is no next hop to route the traffic to the destination.

View details of a route

To analyze routing further, review the effective routes for the network interface with the [Get-AzEffectiveRouteTable](#) command:

```
Get-AzEffectiveRouteTable ` 
-NetworkInterfaceName myVm ` 
-ResourceGroupName myResourceGroup | 
Format-table
```

Output that includes the following text is returned:

Name	State	Source	AddressPrefix	NextHopType	NextHopIpAddress
Active	Default	{192.168.0.0/16}		VnetLocal	{}
Active	Default	{0.0.0.0/0}		Internet	{}
Active	Default	{10.0.0.0/8}		None	{}
Active	Default	{100.64.0.0/10}		None	{}
Active	Default	{172.16.0.0/12}		None	{}

As you can see in the previous output, the route with the **AddressPrefix** of **0.0.0.0/0** routes all traffic not destined for addresses within other route's address prefixes with a next hop of **Internet**. As you can also see in the output, though there is a default route to the 172.16.0.0/12 prefix, which includes the 172.31.0.100 address, the **nextHopType** is **None**. Azure creates a default route to 172.16.0.0/12, but doesn't specify a next hop type until there is a reason to. If, for example, you added the 172.16.0.0/12 address range to the address space of the virtual network, Azure changes the **nextHopType** to **Virtual network** for the route. A check would then show **Virtual network** as the **nextHopType**.

Clean up resources

When no longer needed, you can use [Remove-AzResourceGroup](#) to remove the resource group and all of the resources it contains:

```
Remove-AzResourceGroup -Name myResourceGroup -Force
```

Next steps

In this article, you created a VM and diagnosed network routing from the VM. You learned that Azure creates several default routes and tested routing to two different destinations. Learn more about [routing in Azure](#) and how to [create custom routes](#).

For outbound VM connections, you can also determine the latency and allowed and denied network traffic between the VM and an endpoint using Network Watcher's [connection troubleshoot](#) capability. You can monitor communication between a VM and an endpoint, such as an IP address or URL, over time using the Network Watcher connection monitor capability. To learn how, see [Monitor a network connection](#).

Diagnose a virtual machine network routing problem - Azure CLI

1/28/2020 • 5 minutes to read • [Edit Online](#)

In this article, you deploy a virtual machine (VM), and then check communications to an IP address and URL. You determine the cause of a communication failure and how you can resolve it.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select Try It in the upper-right corner of a code block. Selecting Try It doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com , or select the Launch Cloud Shell button to open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu bar at the upper right in the Azure portal .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this article requires that you are running the Azure CLI version 2.0.28 or later. To find the installed version, run `az --version`. If you need to install or upgrade, see [Install the Azure CLI](#). After you verify the CLI version, run `az login` to create a connection with Azure. The CLI commands in this article are formatted to run in a Bash shell.

Create a VM

Before you can create a VM, you must create a resource group to contain the VM. Create a resource group with `az group create`. The following example creates a resource group named `myResourceGroup` in the `eastus` location:

```
az group create --name myResourceGroup --location eastus
```

Create a VM with [az vm create](#). If SSH keys do not already exist in a default key location, the command creates them. To use a specific set of keys, use the `--ssh-key-value` option. The following example creates a VM named *myVm*:

```
az vm create \  
  --resource-group myResourceGroup \  
  --name myVm \  
  --image UbuntuLTS \  
  --generate-ssh-keys
```

The VM takes a few minutes to create. Don't continue with remaining steps until the VM is created and the CLI returns output.

Test network communication

To test network communication with Network Watcher, you must first enable a network watcher in the region the VM that you want to test is in, and then use Network Watcher's next hop capability to test communication.

Enable network watcher

If you already have a network watcher enabled in the East US region, skip to [Use next hop](#). Use the [az network watcher configure](#) command to create a network watcher in the East US region:

```
az network watcher configure \  
  --resource-group NetworkWatcherRG \  
  --locations eastus \  
  --enabled
```

Use next hop

Azure automatically creates routes to default destinations. You may create custom routes that override the default routes. Sometimes, custom routes can cause communication to fail. To test routing from a VM, use [az network watcher show-next-hop](#) to determine the next routing hop when traffic is destined for a specific address.

Test outbound communication from the VM to one of the IP addresses for www.bing.com:

```
az network watcher show-next-hop \  
  --dest-ip 13.107.21.200 \  
  --resource-group myResourceGroup \  
  --source-ip 10.0.0.4 \  
  --vm myVm \  
  --nic myVmVMNic \  
  --out table
```

After a few seconds, the output informs you that the **nextHopType** is **Internet**, and that the **routeTableId** is **System Route**. This result lets you know that there is a valid route to the destination.

Test outbound communication from the VM to 172.31.0.100:

```
az network watcher show-next-hop \
--dest-ip 172.31.0.100 \
--resource-group myResourceGroup \
--source-ip 10.0.0.4 \
--vm myVm \
--nic myVmVMNic \
--out table
```

The output returned informs you that **None** is the **nextHopType**, and that the **routeTableId** is also **System Route**. This result lets you know that, while there is a valid system route to the destination, there is no next hop to route the traffic to the destination.

View details of a route

To analyze routing further, review the effective routes for the network interface with the [az network nic show-effective-route-table](#) command:

```
az network nic show-effective-route-table \
--resource-group myResourceGroup \
--name myVmVMNic
```

The following text is included in the returned output:

```
{
  "additionalProperties": {
    "disableBgpRoutePropagation": false
  },
  "addressPrefix": [
    "0.0.0.0/0"
  ],
  "name": null,
  "nextHopIpAddress": [],
  "nextHopType": "Internet",
  "source": "Default",
  "state": "Active"
},
```

When you used the `az network watcher show-next-hop` command to test outbound communication to 13.107.21.200 in [Use next hop](#), the route with the **addressPrefix** 0.0.0.0/0** was used to route traffic to the address, since no other route in the output includes the address. By default, all addresses not specified within the address prefix of another route are routed to the internet.

When you used the `az network watcher show-next-hop` command to test outbound communication to 172.31.0.100 however, the result informed you that there was no next hop type. In the returned output you also see the following text:

```
{  
  "additionalProperties": {  
    "disableBgpRoutePropagation": false  
  },  
  "addressPrefix": [  
    "172.16.0.0/12"  
  ],  
  "name": null,  
  "nextHopIpAddress": [],  
  "nextHopType": "None",  
  "source": "Default",  
  "state": "Active"  
},
```

As you can see in the output from the `az network watcher nic show-effective-route-table` command, though there is a default route to the 172.16.0.0/12 prefix, which includes the 172.31.0.100 address, the **nextHopType** is **None**. Azure creates a default route to 172.16.0.0/12, but doesn't specify a next hop type until there is a reason to. If, for example, you added the 172.16.0.0/12 address range to the address space of the virtual network, Azure changes the **nextHopType** to **Virtual network** for the route. A check would then show **Virtual network** as the **nextHopType**.

Clean up resources

When no longer needed, you can use `az group delete` to remove the resource group and all of the resources it contains:

```
az group delete --name myResourceGroup --yes
```

Next steps

In this article, you created a VM and diagnosed network routing from the VM. You learned that Azure creates several default routes and tested routing to two different destinations. Learn more about [routing in Azure](#) and how to [create custom routes](#).

For outbound VM connections, you can also determine the latency and allowed and denied network traffic between the VM and an endpoint using Network Watcher's [connection troubleshoot](#) capability. You can monitor communication between a VM and an endpoint, such as an IP address or URL, over time using the Network Watcher connection monitor capability. To learn how, see [Monitor a network connection](#).

Troubleshoot connections with Azure Network Watcher using the Azure portal

1/28/2020 • 2 minutes to read • [Edit Online](#)

Learn how to use connection troubleshoot to verify whether a direct TCP connection from a virtual machine to a given endpoint can be established.

Before you begin

This article assumes you have the following resources:

- An instance of Network Watcher in the region you want to troubleshoot a connection.
- Virtual machines to troubleshoot connections with.

IMPORTANT

Connection troubleshoot requires that the VM you troubleshoot from has the [AzureNetworkWatcherExtension](#) VM extension installed. For installing the extension on a Windows VM visit [Azure Network Watcher Agent virtual machine extension for Windows](#) and for Linux VM visit [Azure Network Watcher Agent virtual machine extension for Linux](#). The extension is not required on the destination endpoint.

Check connectivity to a virtual machine

This example checks connectivity to a destination virtual machine over port 80.

Navigate to your Network Watcher and click **Connection troubleshoot**. Select the virtual machine to check connectivity from. In the **Destination** section choose **Select a virtual machine** and choose the correct virtual machine and port to test.

Once you click **Check**, connectivity between the virtual machines on the port specified is checked. In the example, the destination VM is unreachable, a listing of hops are shown.

Status		
Unreachable		
Hops		
NAME	IP ADDRESS	STATUS
MyNic-1	10.0.1.4	✓
MyNic-Firewall	10.0.0.4	✓
MyNic-2	10.0.2.4	✓
Probes Sent		
100		
Probes Failed		
100		

Check remote endpoint connectivity

To check the connectivity and latency to a remote endpoint, choose the **Specify manually** radio button in the **Destination** section, input the url and the port and click **Check**. This is used for remote endpoints like websites and storage endpoints.

Status
Reachable

Hops

NAME	IP ADDRESS	STATUS
MyNic-Firewall	10.0.0.4	✓
Internet	104.44.10.90	✓
Internet	104.44.5.82	✓
Internet	104.44.10.53	✓
Internet	13.107.21.200	✓

Average Latency in milliseconds

1

Minimum Latency in milliseconds

1

Maximum Latency in milliseconds

8

Probes Sent

100

Probes Failed

0

Next steps

Learn how to automate packet captures with Virtual machine alerts by viewing [Create an alert triggered packet capture](#)

Find if certain traffic is allowed in or out of your VM by visiting [Check IP flow verify](#)

Troubleshoot connections with Azure Network Watcher using PowerShell

1/28/2020 • 4 minutes to read • [Edit Online](#)

Learn how to use connection troubleshoot to verify whether a direct TCP connection from a virtual machine to a given endpoint can be established.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Before you begin

- An instance of Network Watcher in the region you want to troubleshoot a connection.
- Virtual machines to troubleshoot connections with.

IMPORTANT

Connection troubleshoot requires that the VM you troubleshoot from has the [AzureNetworkWatcherExtension](#) VM extension installed. For installing the extension on a Windows VM visit [Azure Network Watcher Agent virtual machine extension for Windows](#) and for Linux VM visit [Azure Network Watcher Agent virtual machine extension for Linux](#). The extension is not required on the destination endpoint.

Check connectivity to a virtual machine

This example checks a connection to a destination virtual machine over port 80. This example requires that you have Network Watcher enabled in the region containing the source VM.

Example

```
$rgName = "ContosoRG"
$sourceVMName = "MultiTierApp0"
$destVMName = "Database0"

$RG = Get-AzResourceGroup -Name $rgName

$VM1 = Get-AzVM -ResourceGroupName $rgName | Where-Object -Property Name -EQ $sourceVMName
$VM2 = Get-AzVM -ResourceGroupName $rgName | Where-Object -Property Name -EQ $destVMName

$networkWatcher = Get-AzNetworkWatcher | Where-Object -Property Location -EQ -Value $VM1.Location

Test-AzNetworkWatcherConnectivity -NetworkWatcher $networkWatcher -SourceId $VM1.Id -DestinationId $VM2.Id -DestinationPort 80
```

Response

The following response is from the previous example. In this response, the [connectionStatus](#) is **Unreachable**. You can see that all the probes sent failed. The connectivity failed at the virtual appliance due to a user-configured

`NetworkSecurityRule` named **UserRule_Port80**, configured to block incoming traffic on port 80. This information can be used to research connection issues.

```
ConnectionStatus : Unreachable
AvgLatencyInMs :
MinLatencyInMs :
MaxLatencyInMs :
ProbesSent : 100
ProbesFailed : 100
Hops : [
    {
        "Type": "Source",
        "Id": "c5222ea0-3213-4f85-a642-cee63217c2f3",
        "Address": "10.1.1.4",
        "ResourceId": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/appNic0/ipConfigurations/ipconfig1",
        "NextHopIds": [
            "9283a9f0-cc5e-4239-8f5e-ae0f3c19fbba"
        ],
        "Issues": []
    },
    {
        "Type": "VirtualAppliance",
        "Id": "9283a9f0-cc5e-4239-8f5e-ae0f3c19fbba",
        "Address": "10.1.2.4",
        "ResourceId": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/fwNic/ipConfigurations/ipconfig1",
        "NextHopIds": [
            "0f1500cd-c512-4d43-b431-7267e4e67017"
        ],
        "Issues": []
    },
    {
        "Type": "VirtualAppliance",
        "Id": "0f1500cd-c512-4d43-b431-7267e4e67017",
        "Address": "10.1.3.4",
        "ResourceId": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/auNic/ipConfigurations/ipconfig1",
        "NextHopIds": [
            "a88940f8-5fbe-40da-8d99-1dee89240f64"
        ],
        "Issues": [
            {
                "Origin": "Outbound",
                "Severity": "Error",
                "Type": "NetworkSecurityRule",
                "Context": [
                    {
                        "key": "RuleName",
                        "value": "UserRule_Port80"
                    }
                ]
            }
        ]
    },
    {
        "Type": "VnetLocal",
        "Id": "a88940f8-5fbe-40da-8d99-1dee89240f64",
        "Address": "10.1.4.4",
        "ResourceId": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/dbNic0/ipConfigurations/ipconfig1",
        "NextHopIds": [],
        "Issues": []
    }
]
```

Validate routing issues

This example checks connectivity between a virtual machine and a remote endpoint. This example requires that you have Network Watcher enabled in the region containing the source VM.

Example

```
$rgName = "ContosoRG"
$sourceVMName = "MultiTierApp0"

$RG = Get-AzResourceGroup -Name $rgName
$VM1 = Get-AzVM -ResourceGroupName $rgName | Where-Object -Property Name -EQ $sourceVMName

$networkWatcher = Get-AzNetworkWatcher | Where-Object -Property Location -EQ -Value $VM1.Location

Test-AzNetworkWatcherConnectivity -NetworkWatcher $networkWatcher -SourceId $VM1.Id -DestinationAddress
13.107.21.200 -DestinationPort 80
```

Response

In the following example, the `ConnectionStatus` is shown as **Unreachable**. In the `Hops` details, you can see under `Issues` that the traffic was blocked due to a `UserDefinedRoute`.

```
ConnectionStatus : Unreachable
AvgLatencyInMs   :
MinLatencyInMs   :
MaxLatencyInMs   :
ProbesSent        : 100
ProbesFailed      : 100
Hops              : [
    {
        "Type": "Source",
        "Id": "b4f7bceb-07a3-44ca-8bae-adec6628225f",
        "Address": "10.1.1.4",
        "ResourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/appNic0/ipConfigurations/
ipconfig1",
        "NextHopIds": [
            "3fee8adf-692f-4523-b742-f6fdf6da6584"
        ],
        "Issues": [
            {
                "Origin": "Outbound",
                "Severity": "Error",
                "Type": "UserDefinedRoute",
                "Context": [
                    {
                        "key": "RouteType",
                        "value": "User"
                    }
                ]
            }
        ],
        {
            "Type": "Destination",
            "Id": "3fee8adf-692f-4523-b742-f6fdf6da6584",
            "Address": "13.107.21.200",
            "ResourceId": "Unknown",
            "NextHopIds": [],
            "Issues": []
        }
    ]
]
```

Check website latency

The following example checks connectivity to a website. This example requires that you have Network Watcher enabled in the region containing the source VM.

Example

```
$rgName = "ContosoRG"
$sourceVMName = "MultiTierApp0"

$RG = Get-AzResourceGroup -Name $rgName
$VM1 = Get-AzVM -ResourceGroupName $rgName | Where-Object -Property Name -EQ $sourceVMName

$networkWatcher = Get-AzNetworkWatcher | Where-Object -Property Location -EQ -Value $VM1.Location

Test-AzNetworkWatcherConnectivity -NetworkWatcher $networkWatcher -SourceId $VM1.Id -DestinationAddress
https://bing.com/
```

Response

In the following response, you can see the `ConnectionStatus` shows as **Reachable**. When a connection is successful, latency values are provided.

```
ConnectionStatus : Reachable
AvgLatencyInMs   : 1
MinLatencyInMs   : 0
MaxLatencyInMs   : 7
ProbesSent       : 100
ProbesFailed     : 0
Hops             : [
    {
        "Type": "Source",
        "Id": "1f0e3415-27b0-4bf7-a59d-3e19fb854e3e",
        "Address": "10.1.1.4",
        "ResourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/appNic0/ipConfigurations/
ipconfig1",
        "NextHopIds": [
            "f99f2bd1-42e8-4bbf-85b6-5d21d00c84e0"
        ],
        "Issues": []
    },
    {
        "Type": "Internet",
        "Id": "f99f2bd1-42e8-4bbf-85b6-5d21d00c84e0",
        "Address": "204.79.197.200",
        "ResourceId": "Internet",
        "NextHopIds": [],
        "Issues": []
    }
]
```

Check connectivity to a storage endpoint

The following example checks connectivity from a virtual machine to a blog storage account. This example requires that you have Network Watcher enabled in the region containing the source VM.

Example

```

$rgName = "ContosoRG"
$sourceVMName = "MultiTierApp0"

$RG = Get-AzResourceGroup -Name $rgName

$VM1 = Get-AzVM -ResourceGroupName $rgName | Where-Object -Property Name -EQ $sourceVMName

$networkWatcher = Get-AzNetworkWatcher | Where-Object -Property Location -EQ -Value $VM1.Location

Test-AzNetworkWatcherConnectivity -NetworkWatcher $networkWatcher -SourceId $VM1.Id -DestinationAddress
https://contosostorageexample.blob.core.windows.net/

```

Response

The following json is the example response from running the previous cmdlet. As the destination is reachable, the `ConnectionStatus` property shows as **Reachable**. You are provided the details regarding the number of hops required to reach the storage blob and latency.

```

ConnectionStatus : Reachable
AvgLatencyInMs   : 1
MinLatencyInMs   : 0
MaxLatencyInMs   : 8
ProbesSent        : 100
ProbesFailed      : 0
Hops              : [
    {
        "Type": "Source",
        "Id": "9e7f61d9-fb45-41db-83e2-c815a919b8ed",
        "Address": "10.1.1.4",
        "ResourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/appNic0/ipConfigurations/
ipconfig1",
        "NextHopIds": [
            "1e6d4b3c-7964-4afd-b959-aaa746ee0f15"
        ],
        "Issues": []
    },
    {
        "Type": "Internet",
        "Id": "1e6d4b3c-7964-4afd-b959-aaa746ee0f15",
        "Address": "13.71.200.248",
        "ResourceId": "Internet",
        "NextHopIds": [],
        "Issues": []
    }
]

```

Next steps

Determine whether certain traffic is allowed in or out of your VM by visiting [Check IP flow verify](#).

If traffic is being blocked and it should not be, see [Manage Network Security Groups](#) to track down the network security group and security rules that are defined.

Troubleshoot connections with Azure Network Watcher using the Azure CLI

1/28/2020 • 3 minutes to read • [Edit Online](#)

Learn how to use connection troubleshoot to verify whether a direct TCP connection from a virtual machine to a given endpoint can be established.

Before you begin

This article assumes you have the following resources:

- An instance of Network Watcher in the region you want to troubleshoot a connection.
- Virtual machines to troubleshoot connections with.

IMPORTANT

Connection troubleshoot requires that the VM you troubleshoot from has the [AzureNetworkWatcherExtension](#) VM extension installed. For installing the extension on a Windows VM visit [Azure Network Watcher Agent virtual machine extension for Windows](#) and for Linux VM visit [Azure Network Watcher Agent virtual machine extension for Linux](#). The extension is not required on the destination endpoint.

Check connectivity to a virtual machine

This example checks connectivity to a destination virtual machine over port 80.

Example

```
az network watcher test-connectivity --resource-group ContosoRG --source-resource MultiTierApp0 --dest-resource Database0 --dest-port 80
```

Response

The following response is from the previous example. In this response, the `ConnectionStatus` is **Unreachable**. You can see that all the probes sent failed. The connectivity failed at the virtual appliance due to a user-configured `NetworkSecurityRule` named **UserRule_Port80**, configured to block incoming traffic on port 80. This information can be used to research connection issues.

```
{
  "avgLatencyInMs": null,
  "connectionStatus": "Unreachable",
  "hops": [
    {
      "address": "10.1.1.4",
      "id": "bb01d336-d881-4808-9fbc-72f091974d68",
      "issues": [],
      "nextHopIds": [
        "f8b074e9-9980-496b-a35e-619f9bcbf648"
      ],
      "resourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/ap
pNic0/ipConfigurations/ipconfig1",
      "type": "Source"
    },
  ]}
```

```
{
    "address": "10.1.2.4",
    "id": "f8b074e9-9980-496b-a35e-619f9bcf648",
    "issues": [],
    "nextHopIds": [
        "8a5857f3-6ab8-4b11-b9bf-a046d66b8696"
    ],
    "resourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/fw
Nic/ipConfigurations/ipconfig1",
        "type": "VirtualAppliance"
    },
    {
        "address": "10.1.3.4",
        "id": "8a5857f3-6ab8-4b11-b9bf-a046d66b8696",
        "issues": [
            {
                "context": [
                    {
                        "key": "RuleName",
                        "value": "UserRule_Port80"
                    }
                ],
                "origin": "Outbound",
                "severity": "Error",
                "type": "NetworkSecurityRule"
            }
        ],
        "nextHopIds": [
            "6ce2f7a2-ceb4-4145-80e8-5d9f661655d6"
        ],
        "resourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/au
Nic/ipConfigurations/ipconfig1",
        "type": "VirtualAppliance"
    },
    {
        "address": "10.1.4.4",
        "id": "6ce2f7a2-ceb4-4145-80e8-5d9f661655d6",
        "issues": [],
        "nextHopIds": [],
        "resourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/db
Nic0/ipConfigurations/ipconfig1",
        "type": "VnetLocal"
    }
},
"maxLatencyInMs": null,
"minLatencyInMs": null,
"probesFailed": 100,
"probesSent": 100
}
```

Validate routing issues

This example checks connectivity between a virtual machine and a remote endpoint.

Example

```
az network watcher test-connectivity --resource-group ContosoRG --source-resource MultiTierApp0 --dest-
address 13.107.21.200 --dest-port 80
```

Response

In the following example, the `connectionStatus` is shown as **Unreachable**. In the `hops` details, you can see

under `issues` that the traffic was blocked due to a `UserDefinedRoute`.

```
{  
    "avgLatencyInMs": null,  
    "connectionStatus": "Unreachable",  
    "hops": [  
        {  
            "address": "10.1.1.4",  
            "id": "f2cb1868-2049-4839-b8ed-57a480d06f95",  
            "issues": [  
                {  
                    "context": [  
                        {  
                            "key": "RouteType",  
                            "value": "User"  
                        }  
                    ],  
                    "origin": "Outbound",  
                    "severity": "Error",  
                    "type": "UserDefinedRoute"  
                }  
            ],  
            "nextHopIds": [  
                "da4022db-0ab0-48c4-a507-dd4c03561ca5"  
            ],  
            "resourceId": "/subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/ap  
pNic0/ipConfigurations/ipconfig1",  
            "type": "Source"  
        },  
        {  
            "address": "13.107.21.200",  
            "id": "da4022db-0ab0-48c4-a507-dd4c03561ca5",  
            "issues": [],  
            "nextHopIds": [],  
            "resourceId": "Unknown",  
            "type": "Destination"  
        }  
    "maxLatencyInMs": null,  
    "minLatencyInMs": null,  
    "probesFailed": 100,  
    "probesSent": 100  
}
```

Check website latency

The following example checks the connectivity to a website.

Example

```
az network watcher test-connectivity --resource-group ContosoRG --source-resource MultiTierApp0 --dest-  
address https://bing.com --dest-port 80
```

Response

In the following response, you can see the `connectionStatus` shows as **Reachable**. When a connection is successful, latency values are provided.

```
{  
    "avgLatencyInMs": 2,  
    "connectionStatus": "Reachable",  
    "hops": [  
        {  
            "address": "10.1.1.4",  
            "id": "639c2d19-e163-4dfd-8737-5018dd1168ae",  
            "issues": [],  
            "nextHopIds": [  
                "fd43a6e7-c758-4f48-90aa-8db99105a4a3"  
            ],  
            "resourceId": "/subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/ap  
pNic0/ipConfigurations/ipconfig1",  
            "type": "Source"  
        },  
        {  
            "address": "204.79.197.200",  
            "id": "fd43a6e7-c758-4f48-90aa-8db99105a4a3",  
            "issues": [],  
            "nextHopIds": [],  
            "resourceId": "Internet",  
            "type": "Internet"  
        }  
    ],  
    "maxLatencyInMs": 7,  
    "minLatencyInMs": 0,  
    "probesFailed": 0,  
    "probesSent": 100  
}
```

Check connectivity to a storage endpoint

The following example checks the connectivity from a virtual machine to a blog storage account.

Example

```
az network watcher test-connectivity --resource-group ContosoRG --source-resource MultiTierApp0 --dest-  
address https://contosoexamplesa.blob.core.windows.net/
```

Response

The following json is the example response from running the previous cmdlet. As the check is successful, the `connectionStatus` property shows as **Reachable**. You are provided the details regarding the number of hops required to reach the storage blob and latency.

```
{  
    "avgLatencyInMs": 1,  
    "connectionStatus": "Reachable",  
    "hops": [  
        {  
            "address": "10.1.1.4",  
            "id": "5136acff-bf26-4c93-9966-4edb7dd40353",  
            "issues": [],  
            "nextHopIds": [  
                "f8d958b7-3636-4d63-9441-602c1eb2fd56"  
            ],  
            "resourceId": "/subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/appNic0/ipConfigurations/  
ipconfig1",  
            "type": "Source"  
        },  
        {  
            "address": "1.2.3.4",  
            "id": "f8d958b7-3636-4d63-9441-602c1eb2fd56",  
            "issues": [],  
            "nextHopIds": [],  
            "resourceId": "Internet",  
            "type": "Internet"  
        }  
    ],  
    "maxLatencyInMs": 7,  
    "minLatencyInMs": 0,  
    "probesFailed": 0,  
    "probesSent": 100  
}
```

Next steps

Learn how to automate packet captures with Virtual machine alerts by viewing [Create an alert triggered packet capture](#)

Find if certain traffic is allowed in or out of your VM by visiting [Check IP flow verify](#)

Troubleshoot connections with Azure Network Watcher using the Azure REST API

1/28/2020 • 5 minutes to read • [Edit Online](#)

Learn how to use connection troubleshoot to verify whether a direct TCP connection from a virtual machine to a given endpoint can be established.

Before you begin

This article assumes you have the following resources:

- An instance of Network Watcher in the region you want to troubleshoot a connection.
- Virtual machines to troubleshoot connections with.

IMPORTANT

Connection troubleshoot requires that the VM you troubleshoot from has the [AzureNetworkWatcherExtension](#) VM extension installed. For installing the extension on a Windows VM visit [Azure Network Watcher Agent virtual machine extension for Windows](#) and for Linux VM visit [Azure Network Watcher Agent virtual machine extension for Linux](#). The extension is not required on the destination endpoint.

Log in with ARMClient

Log in to armclient with your Azure credentials.

```
armclient login
```

Retrieve a virtual machine

Run the following script to return a virtual machine. This information is needed for running connectivity.

The following code needs values for the following variables:

- **subscriptionId** - The subscription ID to use.
- **resourceGroupName** - The name of a resource group that contains virtual machines.

```
$subscriptionId = '<subscription id>'  
$resourceGroupName = '<resource group name>'  
  
armclient get  
https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Compute/virtualMachines?api-version=2015-05-01-preview
```

From the following output, the ID of the virtual machine is used in the following example:

```

...
,
  "type": "Microsoft.Compute/virtualMachines",
  "location": "westcentralus",
  "id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoExampleRG/providers/Microsoft.Compute
/virtualMachines/ContosoVM",
  "name": "ContosoVM"
}
]
}

```

Check connectivity to a virtual machine

This example checks connectivity to a destination virtual machine over port 80.

Example

```

$subscriptionId = "00000000-0000-0000-0000-000000000000"
$resourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
$sourceResourceId = "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Compute/virtualMachines/MultiTierApp0"
$destinationResourceId = "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Compute/virtualMachines/Database0"
$destinationPort = "80"
$requestBody = @@
{
  'source': {
    'resourceId': '${sourceResourceId}',
    'port': 0
  },
  'destination': {
    'resourceId': '${destinationResourceId}',
    'port': ${destinationPort}
  }
}
"@

$response = armclient post
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Network/networkWatchers/${networkWatcherName}/connectivityCheck?api-version=2017-03-01" $requestBody

```

Since this operation is long running, the URI for the result is returned in the response header as shown in the following response:

Important Values

- **Location** - This property contains the URI where the results are when the operation is complete

```
HTTP/1.1 202 Accepted
Pragma: no-cache
Retry-After: 10
x-ms-request-id: f09b55fe-1d3a-4df7-817f-bceb8d2a94c8
Strict-Transport-Security: max-age=31536000; includeSubDomains
Cache-Control: no-cache
Location: https://management.azure.com/subscriptions/00000000-0000-0000-0000-
000000000000/providers/Microsoft.Network/locations/westcentralus/operationResults/f09b55fe-1d3a-4df7-817f-
bceb8d2a94c8?api-version=2017-03-01
Server: Microsoft-HTTPAPI/2.0; Microsoft-HTTPAPI/2.0
x-ms-ratelimit-remaining-subscription-writes: 1199
x-ms-correlation-request-id: 367a91aa-7142-436a-867d-d3a36f80bc54
x-ms-routing-request-id: WESTUS2:20170602T202117Z:367a91aa-7142-436a-867d-d3a36f80bc54
Date: Fri, 02 Jun 2017 20:21:16 GMT

null
```

Response

The following response is from the previous example. In this response, the `connectionStatus` is **Unreachable**. You can see that all the probes sent failed. The connectivity failed at the virtual appliance due to a user-configured `NetworkSecurityRule` named **UserRule_Port80**, configured to block incoming traffic on port 80. This information can be used to research connection issues.

```
{
  "hops": [
    {
      "type": "Source",
      "id": "0cb75c91-7ebf-4df8-8424-15594d6fb51c",
      "address": "10.1.1.4",
      "resourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/appNic0/ipConfigurations/
ipconfig1",
      "nextHopIds": [
        "06dee00a-9c4a-4fb1-b2ea-fa0a539ca684"
      ],
      "issues": []
    },
    {
      "type": "VirtualAppliance",
      "id": "06dee00a-9c4a-4fb1-b2ea-fa0a539ca684",
      "address": "10.1.2.4",
      "resourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/fwNic/ipConfigurations/ip
config1",
      "nextHopIds": [
        "75e0cfa5-f9d2-48d8-b705-2c7016f81570"
      ],
      "issues": []
    },
    {
      "type": "VirtualAppliance",
      "id": "75e0cfa5-f9d2-48d8-b705-2c7016f81570",
      "address": "10.1.3.4",
      "resourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/auNic/ipConfigurations/ip
config1",
      "nextHopIds": [
        "86caf6aa-33b0-48a1-b4da-f3c9ce785072"
      ],
      "issues": [
        {
          "origin": "Outbound",
          "severity": "Error",
          "type": "NetworkSecurityRule",
          "context": [
            {
              "key": "RuleName",
              "value": "UserRule_Port80"
            }
          ]
        }
      ]
    },
    {
      "type": "VnetLocal",
      "id": "86caf6aa-33b0-48a1-b4da-f3c9ce785072",
      "address": "10.1.4.4",
      "resourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/dbNic0/ipConfigurations/i
pconfig1",
      "nextHopIds": [],
      "issues": []
    }
  ],
  "connectionStatus": "Unreachable",
  "probesSent": 100,
  "probesFailed": 100
}
```

Validate routing issues

The example checks connectivity between a virtual machine and a remote endpoint.

Example

```
$subscriptionId = "00000000-0000-0000-0000-000000000000"
$resourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
$sourceResourceId = "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Compute/virtualMachines/MultiTierApp0"
$destinationResourceId = "13.107.21.200"
$destinationPort = "80"
$requestBody = @"
{
  'source': {
    'resourceId': '${sourceResourceId}',
    'port': 0
  },
  'destination': {
    'address': '${destinationResourceId}',
    'port': ${destinationPort}
  }
}
@"
$response = armclient post
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Network/networkWatchers/${networkWatcherName}/connectivityCheck?api-version=2017-03-01" $requestBody
```

Since this operation is long running, the URI for the result is returned in the response header as shown in the following response:

Important Values

- **Location** - This property contains the URI where the results are when the operation is complete

```
HTTP/1.1 202 Accepted
Pragma: no-cache
Retry-After: 10
x-ms-request-id: 15eeeb69-fcef-41db-bc4a-e2adcf2658e0
Strict-Transport-Security: max-age=31536000; includeSubDomains
Cache-Control: no-cache
Location: https://management.azure.com/subscriptions/00000000-0000-0000-0000-
000000000000/providers/Microsoft.Network/locations/westcentralus/operationResults/15eeeb69-fcef-41db-bc4a-
e2adcf2658e0?api-version=2017-03-01
Server: Microsoft-HTTPAPI/2.0; Microsoft-HTTPAPI/2.0
x-ms-ratelimit-remaining-subscription-writes: 1199
x-ms-correlation-request-id: 4370b798-cd8b-4d3e-ba28-22232bc81dc5
x-ms-routing-request-id: WESTUS:20170602T202606Z:4370b798-cd8b-4d3e-ba28-22232bc81dc5
Date: Fri, 02 Jun 2017 20:26:05 GMT

null
```

Response

In the following example, the `connectionStatus` is shown as **Unreachable**. In the `hops` details, you can see under `issues` that the traffic was blocked due to a `userDefinedRoute`.

```
{  
  "hops": [  
    {  
      "type": "Source",  
      "id": "5528055a-b393-4751-97bc-353d8c0aaeff",  
      "address": "10.1.1.4",  
      "resourceId": "/subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/appNic0/ipConfigurations/  
ipconfig1",  
      "nextHopIds": [  
        "66eefa79-5bfe-48b2-b6ca-eec8247457a3"  
      ],  
      "issues": [  
        {  
          "origin": "Outbound",  
          "severity": "Error",  
          "type": "UserDefinedRoute",  
          "context": [  
            {  
              "key": "RouteType",  
              "value": "User"  
            }  
          ]  
        }  
      ]  
    },  
    {  
      "type": "Destination",  
      "id": "66eefa79-5bfe-48b2-b6ca-eec8247457a3",  
      "address": "13.107.21.200",  
      "resourceId": "Unknown",  
      "nextHopIds": [],  
      "issues": []  
    }  
  ],  
  "connectionStatus": "Unreachable",  
  "probesSent": 100,  
  "probesFailed": 100  
}
```

Check website latency

The following example checks the connectivity to a website.

Example

```

$subscriptionId = "00000000-0000-0000-0000-000000000000"
$resourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
$sourceResourceId = "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/ContosoRG/providers/Microsoft.Compute/virtualMachines/MultiTierApp0"
$destinationResourceId = "https://bing.com"
$destinationPort = "80"
$requestBody = @"
{
  'source': {
    'resourceId': '${sourceResourceId}',
    'port': 0
  },
  'destination': {
    'address': '${destinationResourceId}',
    'port': ${destinationPort}
  }
}
"@

$response = armclient post
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Network/networkWatchers/${networkWatcherName}/connectivityCheck?api-version=2017-03-01" $requestBody

```

Since this operation is long running, the URI for the result is returned in the response header as shown in the following response:

Important Values

- **Location** - This property contains the URI where the results are when the operation is complete

```

HTTP/1.1 202 Accepted
Pragma: no-cache
Retry-After: 10
x-ms-request-id: e49b12c7-c232-472c-b6d2-6c257ce80fa5
Strict-Transport-Security: max-age=31536000; includeSubDomains
Cache-Control: no-cache
Location: https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/providers/Microsoft.Network/locations/westcentralus/operationResults/e49b12c7-c232-472c-b6d2-6c257ce80fa5?api-version=2017-03-01
Server: Microsoft-HTTPAPI/2.0; Microsoft-HTTPAPI/2.0
x-ms-ratelimit-remaining-subscription-writes: 1199
x-ms-correlation-request-id: c3d9744f-5683-427d-bdd1-636b68ab01b6
x-ms-routing-request-id: WESTUS:20170602T203101Z:c3d9744f-5683-427d-bdd1-636b68ab01b6
Date: Fri, 02 Jun 2017 20:31:00 GMT

null

```

Response

In the following response, you can see the `connectionStatus` shows as **Reachable**. When a connection is successful, latency values are provided.

```
{
  "hops": [
    {
      "type": "Source",
      "id": "6adc0fe1-e384-4220-b1b1-f0d181220072",
      "address": "10.1.1.4",
      "resourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/appNic0/ipConfigurations/
ipconfig1",
      "nextHopIds": [
        "b50b7076-9ff2-4782-b40e-0b89cf758f74"
      ],
      "issues": []
    },
    {
      "type": "Internet",
      "id": "b50b7076-9ff2-4782-b40e-0b89cf758f74",
      "address": "204.79.197.200",
      "resourceId": "Internet",
      "nextHopIds": [],
      "issues": []
    }
  ],
  "connectionStatus": "Reachable",
  "avgLatencyInMs": 1,
  "minLatencyInMs": 0,
  "maxLatencyInMs": 7,
  "probesSent": 100,
  "probesFailed": 0
}
```

Check connectivity to a storage endpoint

The following example checks the connectivity from a virtual machine to a blog storage account.

Example

```
$subscriptionId = "00000000-0000-0000-0000-000000000000"
$resourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
$sourceResourceId = "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Compute/virtualMachines/MultiTierApp0"
$destinationResourceId = "https://build2017nwdiag360.blob.core.windows.net/"
$destinationPort = "0"
$requestBody = @"
{
  'source': {
    'resourceId': '${sourceResourceId}',
    'port': 0
  },
  'destination': {
    'address': '${destinationResourceId}',
    'port': ${destinationPort}
  }
}
@

$response = armclient post
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Network/networkWatchers/${networkWatcherName}/connectivityCheck?api-version=2017-03-01" $requestBody
```

Since this operation is long running, the URI for the result is returned in the response header as shown in the following response:

Important Values

- **Location** - This property contains the URI where the results are when the operation is complete

```
HTTP/1.1 202 Accepted
Pragma: no-cache
Retry-After: 10
x-ms-request-id: c4ed3806-61ea-4a6b-abc1-9d6f2afc79c2
Strict-Transport-Security: max-age=31536000; includeSubDomains
Cache-Control: no-cache
Location: https://management.azure.com/subscriptions/00000000-0000-0000-0000-
000000000000/providers/Microsoft.Network/locations/westcentralus/operationResults/c4ed3806-61ea-4a6b-abc1-
9d6f2afc79c2?api-version=2017-03-01
Server: Microsoft-HTTPAPI/2.0; Microsoft-HTTPAPI/2.0
x-ms-ratelimit-remaining-subscription-writes: 1199
x-ms-correlation-request-id: 93bf5af0-fef5-4b7a-bb9e-9976ba5cdb95
x-ms-routing-request-id: WESTUS2:20170602T200504Z:93bf5af0-fef5-4b7a-bb9e-9976ba5cdb95
Date: Fri, 02 Jun 2017 20:05:03 GMT

null
```

Response

The following example is the response from running the previous API call. As the check is successful, the `connectionStatus` property shows as **Reachable**. You are provided the details regarding the number of hops required to reach the storage blob and latency.

```
{
  "hops": [
    {
      "type": "Source",
      "id": "6adc0fe1-e384-4220-b1b1-f0d181220072",
      "address": "10.1.1.4",
      "resourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/ContosoRG/providers/Microsoft.Network/networkInterfaces/appNic0/ipConfigurations/
ipconfig1",
      "nextHopIds": [
        "b50b7076-9ff2-4782-b40e-0b89cf758f74"
      ],
      "issues": []
    },
    {
      "type": "Internet",
      "id": "b50b7076-9ff2-4782-b40e-0b89cf758f74",
      "address": "13.71.200.248",
      "resourceId": "Internet",
      "nextHopIds": [],
      "issues": []
    }
  ],
  "connectionStatus": "Reachable",
  "avgLatencyInMs": 1,
  "minLatencyInMs": 0,
  "maxLatencyInMs": 7,
  "probesSent": 100,
  "probesFailed": 0
}
```

Next steps

Learn how to automate packet captures with Virtual machine alerts by viewing [Create an alert triggered packet capture](#).

Find if certain traffic is allowed in or out of your VM by visiting [Check IP flow verify](#).

Manage packet captures with Azure Network Watcher using the portal

1/28/2020 • 4 minutes to read • [Edit Online](#)

Network Watcher packet capture allows you to create capture sessions to track traffic to and from a virtual machine. Filters are provided for the capture session to ensure you capture only the traffic you want. Packet capture helps to diagnose network anomalies, both reactively, and proactively. Other uses include gathering network statistics, gaining information on network intrusions, to debug client-server communication, and much more. Being able to remotely trigger packet captures, eases the burden of running a packet capture manually on a desired virtual machine, which saves valuable time.

In this article, you learn to start, stop, download, and delete a packet capture.

Before you begin

Packet capture requires the following connectivity:

- Outbound connectivity to a storage account over port 443.
- Inbound and outbound connectivity to 169.254.169.254
- Inbound and outbound connectivity to 168.63.129.16

If a network security group is associated to the network interface, or subnet that the network interface is in, ensure that rules exist that allow the previous ports. Similarly, adding user-defined traffic routes to your network may prevent connectivity to the above mentioned IPs and ports. Please ensure they are reachable.

Start a packet capture

1. In your browser, navigate to the [Azure portal](#) and select **All services**, and then select **Network Watcher** in the **Networking section**.
2. Select **Packet capture** under **Network diagnostic tools**. Any existing packet captures are listed, regardless of their status.
3. Select **Add** to create a packet capture. You can select values for the following properties:
 - **Subscription**: The subscription that the virtual machine you want to create the packet capture for is in.
 - **Resource group**: The resource group of the virtual machine.
 - **Target virtual machine**: The virtual machine that you want to create the packet capture for.
 - **Packet capture name**: A name for the packet capture.
 - **Storage account or file**: Select **Storage account**, **File**, or both. If you select **File**, the capture is written to a path within the virtual machine.
 - **Local file path**: The local path on the virtual machine where the packet capture will be saved (valid only when *File* is selected). The path must be a valid path. If you are using a Linux virtual machine, the path must start with */var/captures*.
 - **Storage accounts**: Select an existing storage account, if you selected *Storage account*. This option is only available if you selected **Storage**.

NOTE

Premium storage accounts are currently not supported for storing packet captures.

- **Maximum bytes per packet:** The number of bytes from each packet that are captured. If left blank, all bytes are captured.
- **Maximum bytes per session:** The total number of bytes that are captured. Once the value is reached the packet capture stops.
- **Time limit (seconds):** The time limit before the packet capture is stopped. The default is 18,000 seconds.
- Filtering (Optional). Select + **Add filter**
 - **Protocol:** The protocol to filter for the packet capture. The available values are TCP, UDP, and Any.
 - **Local IP address:** Filters the packet capture for packets where the local IP address matches this value.
 - **Local port:** Filters the packet capture for packets where the local port matches this value.
 - **Remote IP address:** Filters the packet capture for packets where the remote IP address matches this value.
 - **Remote port:** Filters the packet capture for packets where the remote port matches this value.

NOTE

Port and IP address values can be a single value, range of values, or a range, such as 80-1024, for port. You can define as many filters as you need.

4. Select OK.

After the time limit set on the packet capture has expired, the packet capture is stopped, and can be reviewed. You can also manually stop a packet capture session.

NOTE

The portal automatically:

- Creates a network watcher in the same region as the region the virtual machine you selected exists in, if the region doesn't already have a network watcher.
- Adds the *AzureNetworkWatcherExtension* [Linux](#) or [Windows](#) virtual machine extension to the virtual machine, if it's not already installed.

Delete a packet capture

1. In the packet capture view, select ... on the right-side of the packet capture, or right-click an existing packet capture, and select **Delete**.
2. You are asked to confirm you want to delete the packet capture. Select **Yes**.

NOTE

Deleting a packet capture does not delete the capture file in the storage account or on the virtual machine.

Stop a packet capture

In the packet capture view, select ... on the right-side of the packet capture, or right-click an existing packet capture, and select **Stop**.

Download a packet capture

Once your packet capture session has completed, the capture file is uploaded to blob storage or to a local file on the virtual machine. The storage location of the packet capture is defined during creation of the packet capture. A convenient tool to access capture files saved to a storage account is Microsoft Azure Storage Explorer, which you can [download](#).

If a storage account is specified, packet capture files are saved to a storage account at the following location:

```
https://{{storageAccountName}}.blob.core.windows.net/network-watcher-logs/subscriptions/{{subscriptionId}}/resourcegroups/{{storageAccountResourceGroup}}/providers/microsoft.compute/virtualmachines/{{VMName}}/{{year}}/{{month}}/{{day}}/packetCapture_{creationTime}.cap
```

If you selected **File** when you created the capture, you can view or download the file from the path you configured on the virtual machine.

Next steps

- To learn how to automate packet captures with virtual machine alerts, see [Create an alert triggered packet capture](#).
- To determine whether specific traffic is allowed in or out of a virtual machine, see [Diagnose a virtual machine network traffic filter problem](#).

Manage packet captures with Azure Network Watcher using PowerShell

1/28/2020 • 5 minutes to read • [Edit Online](#)

Network Watcher packet capture allows you to create capture sessions to track traffic to and from a virtual machine. Filters are provided for the capture session to ensure you capture only the traffic you want. Packet capture helps to diagnose network anomalies both reactively and proactively. Other uses include gathering network statistics, gaining information on network intrusions, to debug client-server communications and much more. By being able to remotely trigger packet captures, this capability eases the burden of running a packet capture manually and on the desired machine, which saves valuable time.

This article takes you through the different management tasks that are currently available for packet capture.

- [Start a packet capture](#)
- [Stop a packet capture](#)
- [Delete a packet capture](#)
- [Download a packet capture](#)

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Before you begin

This article assumes you have the following resources:

- An instance of Network Watcher in the region you want to create a packet capture
- A virtual machine with the packet capture extension enabled.

IMPORTANT

Packet capture requires a virtual machine extension [AzureNetworkWatcherExtension](#). For installing the extension on a Windows VM visit [Azure Network Watcher Agent virtual machine extension for Windows](#) and for Linux VM visit [Azure Network Watcher Agent virtual machine extension for Linux](#).

Install VM extension

Step 1

```
$VM = Get-AzVM -ResourceGroupName testrg -Name VM1
```

Step 2

The following example retrieves the extension information needed to run the [Set-AzVMExtension](#) cmdlet. This cmdlet installs the packet capture agent on the guest virtual machine.

NOTE

The `Set-AzVMExtension` cmdlet may take several minutes to complete.

For Windows virtual machines:

```
$AzureNetworkWatcherExtension = Get-AzVMExtensionImage -Location WestCentralUS -PublisherName Microsoft.Azure.NetworkWatcher -Type NetworkWatcherAgentWindows -Version 1.4.585.2  
$ExtensionName = "AzureNetworkWatcherExtension"  
Set-AzVMExtension -ResourceGroupName $VM.ResourceGroupName -Location $VM.Location -VMName $VM.Name -Name $ExtensionName -Publisher $AzureNetworkWatcherExtension.PublisherName -ExtensionType $AzureNetworkWatcherExtension.Type -TypeHandlerVersion $AzureNetworkWatcherExtension.Version.Substring(0,3)
```

For Linux virtual machines:

```
$AzureNetworkWatcherExtension = Get-AzVMExtensionImage -Location WestCentralUS -PublisherName Microsoft.Azure.NetworkWatcher -Type NetworkWatcherAgentLinux -Version 1.4.13.0  
$ExtensionName = "AzureNetworkWatcherExtension"  
Set-AzVMExtension -ResourceGroupName $VM.ResourceGroupName -Location $VM.Location -VMName $VM.Name -Name $ExtensionName -Publisher $AzureNetworkWatcherExtension.PublisherName -ExtensionType $AzureNetworkWatcherExtension.Type -TypeHandlerVersion $AzureNetworkWatcherExtension.Version.Substring(0,3)
```

The following example is a successful response after running the `Set-AzVMExtension` cmdlet.

RequestId	IsSuccess	Status	Code	ReasonPhrase
	True	OK	OK	

Step 3

To ensure that the agent is installed, run the `Get-AzVMExtension` cmdlet and pass it the virtual machine name and the extension name.

```
Get-AzVMExtension -ResourceGroupName $VM.ResourceGroupName -VMName $VM.Name -Name $ExtensionName
```

The following sample is an example of the response from running `Get-AzVMExtension`

```
ResourceGroupName      : testrg  
VMName                : testvm1  
Name                  : AzureNetworkWatcherExtension  
Location              : westcentralus  
Etag                  : null  
Publisher             : Microsoft.Azure.NetworkWatcher  
ExtensionType         : NetworkWatcherAgentWindows  
TypeHandlerVersion    : 1.4  
Id                    : /subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/testrg/providers/Microsoft.Compute/virtualMachines/testvm1/  
extensions/AzureNetworkWatcherExtension  
PublicSettings         :  
ProtectedSettings      :  
ProvisioningState     : Succeeded  
Statuses              :  
SubStatuses            :  
AutoUpgradeMinorVersion : True  
ForceUpdateTag         :
```

Start a packet capture

Once the preceding steps are complete, the packet capture agent is installed on the virtual machine.

Step 1

The next step is to retrieve the Network Watcher instance. This variable is passed to the

```
New-AzNetworkWatcherPacketCapture cmdlet in step 4.
```

```
$networkWatcher = Get-AzResource | Where {$_.ResourceType -eq "Microsoft.Network/networkWatchers" -and  
$_.Location -eq "WestCentralUS" }
```

Step 2

Retrieve a storage account. This storage account is used to store the packet capture file.

```
$storageAccount = Get-AzStorageAccount -ResourceGroupName testrg -Name testrgsa123
```

Step 3

Filters can be used to limit the data that is stored by the packet capture. The following example sets up two filters. One filter collects outgoing TCP traffic only from local IP 10.0.0.3 to destination ports 20, 80 and 443. The second filter collects only UDP traffic.

```
$filter1 = New-AzPacketCaptureFilterConfig -Protocol TCP -RemoteIPAddress "1.1.1.1-255.255.255.255" -  
LocalIPAddress "10.0.0.3" -LocalPort "1-65535" -RemotePort "20;80;443"  
$filter2 = New-AzPacketCaptureFilterConfig -Protocol UDP
```

NOTE

Multiple filters can be defined for a packet capture.

Step 4

Run the `New-AzNetworkWatcherPacketCapture` cmdlet to start the packet capture process, passing the required values retrieved in the preceding steps.

```
New-AzNetworkWatcherPacketCapture -NetworkWatcher $networkWatcher -TargetVirtualMachineId $vm.Id -  
PacketCaptureName "PacketCaptureTest" -StorageAccountId $storageAccount.id -TimeLimitInSeconds 60 -Filter  
$filter1, $filter2
```

The following example is the expected output from running the `New-AzNetworkWatcherPacketCapture` cmdlet.

```

Name          : PacketCaptureTest
Id           : /subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatcher
s/NetworkWatcher_westcentralus/packetCaptures/PacketCaptureTest
Etag         : W/"3bf27278-8251-4651-9546-c7f369855e4e"
ProvisioningState : Succeeded
Target        : /subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/testrg/providers/Microsoft.Compute/virtualMachines/testvm1
BytesToCapturePerPacket : 0
TotalBytesPerSession : 1073741824
TimeLimitInSeconds : 60
StorageLocation   : {
    "StorageId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/testrg/providers/Microsoft.Storage/storageA
ccounts/examplestorage",
    "StoragePath": "https://examplestorage.blob.core.windows.net/network-watcher-
logs/subscriptions/00000000-0000-0000-0000-00000
000000/resourcegroups/testrg/providers/microsoft.compute/virtualmachines/testvm1/2017/02/01/packetcapture_22_
42_48_238.cap"
}
Filters       : [
    {
        "Protocol": "TCP",
        "RemoteIPAddress": "1.1.1.1-255.255.255",
        "LocalIPAddress": "10.0.0.3",
        "LocalPort": "1-65535",
        "RemotePort": "20;80;443"
    },
    {
        "Protocol": "UDP",
        "RemoteIPAddress": "",
        "LocalIPAddress": "",
        "LocalPort": "",
        "RemotePort": ""
    }
]

```

Get a packet capture

Running the `Get-AzNetworkWatcherPacketCapture` cmdlet, retrieves the status of a currently running, or completed packet capture.

```
Get-AzNetworkWatcherPacketCapture -NetworkWatcher $networkWatcher -PacketCaptureName "PacketCaptureTest"
```

The following example is the output from the `Get-AzNetworkWatcherPacketCapture` cmdlet. The following example is after the capture is complete. The `PacketCaptureStatus` value is `Stopped`, with a `StopReason` of `TimeExceeded`. This value shows that the packet capture was successful and ran its time.

```

Name          : PacketCaptureTest
Id           : /subscriptions/00000000-0000-0000-0000-
00000000/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatcher
s/NetworkWatcher_westcentralus/packetCaptures/PacketCaptureTest
Etag         : W/"4b9a81ed-dc63-472e-869e-96d7166ccb9b"
ProvisioningState : Succeeded
Target        : /subscriptions/00000000-0000-0000-0000-
00000000/resourceGroups/testrg/providers/Microsoft.Compute/virtualMachines/testvm1
BytesToCapturePerPacket : 0
TotalBytesPerSession : 1073741824
TimeLimitInSeconds : 60
StorageLocation   :
    "StorageId": "/subscriptions/00000000-0000-0000-0000-
00000000/resourceGroups/testrg/providers/Microsoft.Storage/storageA
ccounts/examplestorage",
    "StoragePath": "https://examplestorage.blob.core.windows.net/network-watcher-
logs/subscriptions/00000000-0000-0000-0000-0000
000000/resourcegroups/testrg/providers/microsoft.compute/virtualmachines/testvm1/2017/02/01/packetcapture_22_
42_48_238.cap"
}
Filters       : [
    {
        "Protocol": "TCP",
        "RemoteIPAddress": "1.1.1.1-255.255.255",
        "LocalIPAddress": "10.0.0.3",
        "LocalPort": "1-65535",
        "RemotePort": "20;80;443"
    },
    {
        "Protocol": "UDP",
        "RemoteIPAddress": "",
        "LocalIPAddress": "",
        "LocalPort": "",
        "RemotePort": ""
    }
]
CaptureStartTime : 2/1/2017 10:43:01 PM
PacketCaptureStatus : Stopped
StopReason      : TimeExceeded
PacketCaptureError : []

```

Stop a packet capture

By running the `Stop-AzNetworkWatcherPacketCapture` cmdlet, if a capture session is in progress it is stopped.

```
Stop-AzNetworkWatcherPacketCapture -NetworkWatcher $networkWatcher -PacketCaptureName "PacketCaptureTest"
```

NOTE

The cmdlet returns no response when ran on a currently running capture session or an existing session that has already stopped.

Delete a packet capture

```
Remove-AzNetworkWatcherPacketCapture -NetworkWatcher $networkWatcher -PacketCaptureName "PacketCaptureTest"
```

NOTE

Deleting a packet capture does not delete the file in the storage account.

Download a packet capture

Once your packet capture session has completed, the capture file can be uploaded to blob storage or to a local file on the VM. The storage location of the packet capture is defined at creation of the session. A convenient tool to access these capture files saved to a storage account is Microsoft Azure Storage Explorer, which can be downloaded here: <https://storageexplorer.com/>

If a storage account is specified, packet capture files are saved to a storage account at the following location:

```
https://{{storageAccountName}}.blob.core.windows.net/network-watcher-logs/subscriptions/{{subscriptionId}}/resourcegroups/{{storageAccountResourceGroup}}/providers/microsoft.compute/virtualmachines/{{VMName}}/{{year}}/{{month}}/{{day}}/packetCapture_{creationTime}.cap
```

Next steps

Learn how to automate packet captures with Virtual machine alerts by viewing [Create an alert triggered packet capture](#)

Find if certain traffic is allowed in or out of your VM by visiting [Check IP flow verify](#)

Manage packet captures with Azure Network Watcher using the Azure CLI

1/29/2020 • 4 minutes to read • [Edit Online](#)

Network Watcher packet capture allows you to create capture sessions to track traffic to and from a virtual machine. Filters are provided for the capture session to ensure you capture only the traffic you want. Packet capture helps to diagnose network anomalies both reactively and proactively. Other uses include gathering network statistics, gaining information on network intrusions, to debug client-server communications and much more. By being able to remotely trigger packet captures, this capability eases the burden of running a packet capture manually and on the desired machine, which saves valuable time.

To perform the steps in this article, you need to [install the Azure Command-Line Interface for Mac, Linux, and Windows \(Azure CLI\)](#).

This article takes you through the different management tasks that are currently available for packet capture.

- [Start a packet capture](#)
- [Stop a packet capture](#)
- [Delete a packet capture](#)
- [Download a packet capture](#)

Before you begin

This article assumes you have the following resources:

- An instance of Network Watcher in the region you want to create a packet capture
- A virtual machine with the packet capture extension enabled.

IMPORTANT

Packet capture requires an agent to be running on the virtual machine. The Agent is installed as an extension. For instructions on VM extensions, visit [Virtual Machine extensions and features](#).

Install VM extension

Step 1

Run the `az vm extension set` command to install the packet capture agent on the guest virtual machine.

For Windows virtual machines:

```
az vm extension set --resource-group resourceGroupName --vm-name virtualMachineName --publisher Microsoft.Azure.NetworkWatcher --name NetworkWatcherAgentWindows --version 1.4
```

For Linux virtual machines:

```
az vm extension set --resource-group resourceGroupName --vm-name virtualMachineName --publisher Microsoft.Azure.NetworkWatcher --name NetworkWatcherAgentLinux --version 1.4
```

Step 2

To ensure that the agent is installed, run the `vm extension show` command and pass it the resource group and virtual machine name. Check the resulting list to ensure the agent is installed.

For Windows virtual machines:

```
az vm extension show --resource-group resourceName --vm-name virtualMachineName --name NetworkWatcherAgentWindows
```

For Linux virtual machines:

```
az vm extension show --resource-group resourceName --vm-name virtualMachineName --name AzureNetworkWatcherExtension
```

The following sample is an example of the response from running `az vm extension show`

```
{
  "autoUpgradeMinorVersion": true,
  "forceUpdateTag": null,
  "id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachines/{vmName}/extensions/NetworkWatcherAgentWindows",
  "instanceView": null,
  "location": "westcentralus",
  "name": "NetworkWatcherAgentWindows",
  "protectedSettings": null,
  "provisioningState": "Succeeded",
  "publisher": "Microsoft.Azure.NetworkWatcher",
  "resourceGroup": "{resourceGroupName}",
  "settings": null,
  "tags": null,
  "type": "Microsoft.Compute/virtualMachines/extensions",
  "typeHandlerVersion": "1.4",
  "virtualMachineExtensionType": "NetworkWatcherAgentWindows"
}
```

Start a packet capture

Once the preceding steps are complete, the packet capture agent is installed on the virtual machine.

Step 1

Retrieve a storage account. This storage account is used to store the packet capture file.

```
az storage account list
```

Step 2

At this point, you are ready to create a packet capture. First, let's examine the parameters you may want to configure. Filters are one such parameter that can be used to limit the data that is stored by the packet capture. The following example sets up a packet capture with several filters. The first three filters collect outgoing TCP traffic only from local IP 10.0.0.3 to destination ports 20, 80 and 443. The last filter collects only UDP traffic.

```
az network watcher packet-capture create --resource-group {resourceGroupName} --vm {vmName} --name packetCaptureName --storage-account {storageAccountName} --filters "[{\\"protocol\\":\\"TCP\\", \\"remoteIPAddress\\":\\"1.1.1.1-255.255.255\\", \\"localIPAddress\\":\\"10.0.0.3\\", \\"remotePort\\":\\"20\\"}, {\\"protocol\\":\\"TCP\\", \\"remoteIPAddress\\":\\"1.1.1.1-255.255.255\\", \\"localIPAddress\\":\\"10.0.0.3\\", \\"remotePort\\":\\"80\\"}, {\\"protocol\\":\\"TCP\\", \\"remoteIPAddress\\":\\"1.1.1.1-255.255.255\\", \\"localIPAddress\\":\\"10.0.0.3\\", \\"remotePort\\":\\"443\\"}, {\\"protocol\\":\\"UDP\\"}]"
```

The following example is the expected output from running the `az network watcher packet-capture create` command.

```
{
  "bytesToCapturePerPacket": 0,
  "etag": "W/\"b8cf3528-2e14-45cb-a7f3-5712ffb687ac\"",
  "filters": [
    {
      "localIpAddress": "10.0.0.3",
      "localPort": "",
      "protocol": "TCP",
      "remoteIpAddress": "1.1.1.1-255.255.255",
      "remotePort": "20"
    },
    {
      "localIpAddress": "10.0.0.3",
      "localPort": "",
      "protocol": "TCP",
      "remoteIpAddress": "1.1.1.1-255.255.255",
      "remotePort": "80"
    },
    {
      "localIpAddress": "10.0.0.3",
      "localPort": "",
      "protocol": "TCP",
      "remoteIpAddress": "1.1.1.1-255.255.255",
      "remotePort": "443"
    },
    {
      "localIpAddress": "",
      "localPort": "",
      "protocol": "UDP",
      "remoteIpAddress": "",
      "remotePort": ""
    }
  ],
  "id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_westcentralus/packetCaptures/packetCaptureName",
  "name": "packetCaptureName",
  "provisioningState": "Succeeded",
  "resourceGroup": "NetworkWatcherRG",
  "storageLocation": {
    "filePath": null,
    "storageId": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/gwteststorage123abc",
    "storagePath": "https://gwteststorage123abc.blob.core.windows.net/network-watcher-logs/subscriptions/00000000-0000-0000-000000000000/resourcegroups/{resourceGroupName}/providers/microsoft.compute/virtualmachines/{vmName}/2017/05/25/packetcapture_16_22_34_630.cap"
  },
  "target": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachines/{vmName}",
  "timeLimitInSeconds": 18000,
  "totalBytesPerSession": 1073741824
}
```

Get a packet capture

Running the `az network watcher packet-capture show-status` command, retrieves the status of a currently running, or completed packet capture.

```
az network watcher packet-capture show-status --name packetCaptureName --location {networkWatcherLocation}
```

The following example is the output from the `az network watcher packet-capture show-status` command. The following example is when the capture is Stopped, with a StopReason of TimeExceeded.

```
{
  "additionalProperties": {
    "status": "Succeeded"
  },
  "captureStartTime": "2016-12-06T17:20:01.5671279Z",
  "id": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_westce
ntralus/pa
cketCaptures/packetCaptureName",
  "name": "packetCaptureName",
  "packetCaptureError": [],
  "packetCaptureStatus": "Stopped",
  "stopReason": "TimeExceeded"
}
```

Stop a packet capture

By running the `az network watcher packet-capture stop` command, if a capture session is in progress it is stopped.

```
az network watcher packet-capture stop --name packetCaptureName --location westcentralus
```

NOTE

The command returns no response when ran on a currently running capture session or an existing session that has already stopped.

Delete a packet capture

```
az network watcher packet-capture delete --name packetCaptureName --location westcentralus
```

NOTE

Deleting a packet capture does not delete the file in the storage account.

Download a packet capture

Once your packet capture session has completed, the capture file can be uploaded to blob storage or to a local file on the VM. The storage location of the packet capture is defined at creation of the session. A convenient tool to access these capture files saved to a storage account is Microsoft Azure Storage Explorer, which can be downloaded here: <https://storageexplorer.com/>

If a storage account is specified, packet capture files are saved to a storage account at the following location:

```
https://{{storageAccountName}}.blob.core.windows.net/network-watcher-logs/subscriptions/{{subscriptionId}}/resourcegroups/{{storageAccountResourceGroup}}/providers/microsoft.compute/virtualmachines/{{VMName}}/{{year}}/{{month}}/{{day}}/packetCapture_{creationTime}.cap
```

Next steps

Learn how to automate packet captures with Virtual machine alerts by viewing [Create an alert triggered packet capture](#)

Find if certain traffic is allowed in or out of your VM by visiting [Check IP flow verify](#)

Manage packet captures with Azure Network Watcher using Azure REST API

1/28/2020 • 4 minutes to read • [Edit Online](#)

Network Watcher packet capture allows you to create capture sessions to track traffic to and from a virtual machine. Filters are provided for the capture session to ensure you capture only the traffic you want. Packet capture helps to diagnose network anomalies both reactively and proactively. Other uses include gathering network statistics, gaining information on network intrusions, to debug client-server communications and much more. By being able to remotely trigger packet captures, this capability eases the burden of running a packet capture manually and on the desired machine, which saves valuable time.

This article takes you through the different management tasks that are currently available for packet capture.

- [Get a packet capture](#)
- [List all packet captures](#)
- [Query the status of a packet capture](#)
- [Start a packet capture](#)
- [Stop a packet capture](#)
- [Delete a packet capture](#)

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Before you begin

In this scenario, you call the Network Watcher Rest API to run IP Flow Verify. ARMclient is used to call the REST API using PowerShell. ARMClient is found on chocolatey at [ARMClient on Chocolatey](#)

This scenario assumes you have already followed the steps in [Create a Network Watcher](#) to create a Network Watcher.

Packet capture requires a virtual machine extension [AzureNetworkWatcherExtension](#). For installing the extension on a Windows VM visit [Azure Network Watcher Agent virtual machine extension for Windows](#) and for Linux VM visit [Azure Network Watcher Agent virtual machine extension for Linux](#).

Log in with ARMClient

```
armclient login
```

Retrieve a virtual machine

Run the following script to return a virtual machine. This information is needed for starting a packet capture.

The following code needs variables:

- **subscriptionId** - The subscription id can also be retrieved with the **Get-AzSubscription** cmdlet.
- **resourceGroupName** - The name of a resource group that contains virtual machines.

```
$subscriptionId = "<subscription id>"  
$resourceGroupName = "<resource group name>"  
  
armclient get  
https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Compute/virtualMachines?api-version=2015-05-01-preview
```

From the following output, the id of the virtual machine is used in the next example.

```
...  
,  
    "type": "Microsoft.Compute/virtualMachines",  
    "location": "westcentralus",  
    "id": "/subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/ContosoExampleRG/providers/Microsoft.Compute  
/virtualMachines/ContosoVM",  
    "name": "ContosoVM"  
}  
]  
}
```

Get a packet capture

The following example gets the status of a single packet capture

```
$subscriptionId = "<subscription id>"  
$resourceGroupName = "NetworkWatcherRG"  
$networkWatcherName = "NetworkWatcher_westcentralus"  
armclient post  
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Network/networkWatchers/${networkWatcherName}/packetCaptures/${packetCaptureName}/querystatus?api-  
version=2016-12-01"
```

The following responses are examples of a typical response returned when querying the status of a packet capture.

```
{  
    "name": "TestPacketCapture5",  
    "id": "/subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_westce  
ntralus/packetCaptures/TestPacketCapture6",  
    "captureStartTime": "2016-12-06T17:20:01.5671279Z",  
    "packetCaptureStatus": "Running",  
    "packetCaptureError": []  
}
```

```
{
  "name": "TestPacketCapture5",
  "id": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_westce
ntralus/packetCaptures/TestPacketCapture6",
  "captureStartTime": "2016-12-06T17:20:01.5671279Z",
  "packetCaptureStatus": "Stopped",
  "stopReason": "TimeExceeded",
  "packetCaptureError": []
}
```

List all packet captures

The following example gets all packet capture sessions in a region.

```
$subscriptionId = "<subscription id>"
$resourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
armclient get
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Mi
crosoft.Network/networkWatchers/${networkWatcherName}/packetCaptures?api-version=2016-12-01"
```

The following response is an example of a typical response returned when getting all packet captures

```
{
  "value": [
    {
      "name": "TestPacketCapture6",
      "id": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_westce
ntralus/packetCaptures/TestPacketCapture6",
      "etag": "W/\\"091762e1-c23f-448b-89d5-37cf56e4c045\\"",
      "properties": {
        "provisioningState": "Succeeded",
        "target": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/ContosoExampleRG/providers/Microsoft.Compute/virtualMachines/ContosoVM",
        "bytesToCapturePerPacket": 0,
        "totalBytesPerSession": 1073741824,
        "timeLimitInSeconds": 60,
        "storageLocation": {
          "storageId": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/ContosoExampleRG/providers/Microsoft.Storage/storageAccounts/contosoexamplergdiag3
74",
          "storagePath": "https://contosoexamplergdiag374.blob.core.windows.net/network-watcher-
logs/subscriptions/00000000-0000-0000-0000-
0000000000/resourcegroups/contosoexamplerg/providers/microsoft.compute/virtualmachines/contosovm/2016/12/06/
packetcap
ture_17_19_53_056.cap",
          "filePath": "c:\\temp\\packetcapture.cap"
        },
        "filters": [
          {
            "protocol": "Any",
            "localIPAddress": "",
            "localPort": "",
            "remoteIPAddress": "",
            "remotePort": ""
          }
        ]
      }
    },
    {
      "name": "TestPacketCapture7",
      "id": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_westce
ntralus/packetCaptures/TestPacketCapture7",
      "etag": "W/\\"091762e1-c23f-448b-89d5-37cf56e4c045\\"",
      "properties": {
        "provisioningState": "Succeeded",
        "target": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/ContosoExampleRG/providers/Microsoft.Compute/virtualMachines/ContosoVM",
        "bytesToCapturePerPacket": 0,
        "totalBytesPerSession": 1073741824,
        "timeLimitInSeconds": 60,
        "storageLocation": {
          "storageId": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/ContosoExampleRG/providers/Microsoft.Storage/storageAccounts/contosoexamplergdiag3
74",
          "storagePath": "https://contosoexamplergdiag374.blob.core.windows.net/network-watcher-
logs/subscriptions/00000000-0000-0000-0000-
0000000000/resourcegroups/contosoexamplerg/providers/microsoft.compute/virtualmachines/contosovm/2016/12/06/
packetcap
ture_17_19_53_056.cap",
          "filePath": "c:\\temp\\packetcapture.cap"
        },
        "filters": [
          {
            "protocol": "Any",
            "localIPAddress": "",
            "localPort": "",
            "remoteIPAddress": "",
            "remotePort": ""
          }
        ]
      }
    }
  ]
}
```

```

    "id": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_westce
ntralus/packetCaptures/TestPacketCapture7",
    "etag": "W/\"091762e1-c23f-448b-89d5-37cf56e4c045\"",
    "properties": {
        "provisioningState": "Failed",
        "target": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/ContosoExampleRG/providers/Microsoft.Compute/virtualMachines/ContosoVM",
        "bytesToCapturePerPacket": 0,
        "totalBytesPerSession": 1073741824,
        "timeLimitInSeconds": 60,
        "storageLocation": {
            "storageId": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/ContosoExmpleRG/providers/Microsoft.Storage/storageAccounts/contosoexamplergdiag3
74",
            "storagePath": "https://contosoexamplergdiag374.blob.core.windows.net/network-watcher-
logs/subscriptions/00000000-0000-0000-0000-
0000000000/resourcegroups/contosoexamplerg/providers/microsoft.compute/virtualmachines/contosovm/2016/12/06/
packetcap
ture_17_23_15_364.cap",
            "filePath": "c:\\temp\\packetcapture.cap"
        },
        "filters": [
            {
                "protocol": "Any",
                "localIPAddress": "",
                "localPort": "",
                "remoteIPAddress": "",
                "remotePort": ""
            }
        ]
    }
}

```

Query packet capture status

The following example gets all packet capture sessions in a region.

```

$subscriptionId = "<subscription id>"
$resourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
$packetCaptureName = "TestPacketCapture5"
armclient get
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Mi
crosoft.Network/networkWatchers/${networkWatcherName}/packetCaptures/${packetCaptureName}/querystatus?api-
version=2016-12-01"

```

The following response is an example of a typical response returned when querying the status of a packet capture.

```

{
    "name": "vm1PacketCapture",
    "id": "/subscriptions/{guid}/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkWatchers/{network
WatcherName}/packetCaptures/{packetCaptureName}",
    "captureStartTime" : "9/7/2016 12:35:24PM",
    "packetCaptureStatus" : "Stopped",
    "stopReason" : "TimeExceeded",
    "packetCaptureError" : [ ]
}

```

Start packet capture

The following example creates a packet capture on a virtual machine. The example is parameterized to allow for flexibility in creating an example.

```
$subscriptionId = '<subscription id>'  
$resourceGroupName = "NetworkWatcherRG"  
$networkWatcherName = "NetworkWatcher_westcentralus"  
$packetCaptureName = "TestPacketCapture5"  
$storageaccountname = "contosoexamplergdiag374"  
$vmName = "ContosoVM"  
$bytestoCaptureperPacket = "0"  
$bytesPerSession = "1073741824"  
$captureTimeInSeconds = "60"  
$localIP = ""  
$localPort = "" # Examples are: 80, or 80-120  
$remoteIP = ""  
$remotePort = "" # Examples are: 80, or 80-120  
$protocol = "" # Valid values are TCP, UDP and Any.  
$targetUri = "" # Example:  
/subscriptions/$subscriptionId/resourceGroups/$resourceGroupName/providers/Microsoft.compute/virtualMachine/$v  
mName  
$storageId = "" #Example "/subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/ContosoExampleRG/providers/Microsoft.Storage/storageAccounts/contosoexamplergdiag3  
74"  
$storagePath = "" # Example: "https://mytestaccountname.blob.core.windows.net/capture/vm1Capture.cap"  
$localFilePath = "c:\temp\packetcapture.cap" # Example: "d:\capture\vm1Capture.cap"  
  
$requestBody = @"  
{  
    'properties': {  
        'target': '/${targetUri}',  
        'bytesToCapturePerPacket': '${bytestoCaptureperPacket}',  
        'totalBytesPerSession': '${bytesPerSession}',  
        'timeLimitinSeconds': '${captureTimeInSeconds}',  
        'storageLocation': {  
            'storageId': '${storageId}',  
            'storagePath': '${storagePath}',  
            'filePath': '${localFilePath}'  
        },  
        'filters': [  
            {  
                'protocol': '${protocol}',  
                'localIPAddress': '${localIP}',  
                'localPort': '${localPort}',  
                'remoteIPAddress': '${remoteIP}',  
                'remotePort': '${remotePort}'  
            }  
        ]  
    }  
}  
"@  
  
armclient PUT  
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Mi  
crosoft.Network/networkWatchers/${networkWatcherName}/packetCaptures/${packetCaptureName}?api-version=2016-07-  
01" $requestbody
```

Stop packet capture

The following example stops a packet capture on a virtual machine. The example is parameterized to allow for flexibility in creating an example.

```
$subscriptionId = '<subscription id>'  
$resourceGroupName = "NetworkWatcherRG"  
$networkWatcherName = "NetworkWatcher_westcentralus"  
$packetCaptureName = "TestPacketCapture5"  
armclient post  
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Mi  
crosoft.Network/networkWatchers/${networkWatcherName}/packetCaptures/${packetCaptureName}/stop?api-  
version=2016-12-01"
```

Delete packet capture

The following example deletes a packet capture on a virtual machine. The example is parameterized to allow for flexibility in creating an example.

```
$subscriptionId = '<subscription id>'  
$resourceGroupName = "NetworkWatcherRG"  
$networkWatcherName = "NetworkWatcher_westcentralus"  
$packetCaptureName = "TestPacketCapture5"  
  
armclient delete  
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Mi  
crosoft.Network/networkWatchers/${networkWatcherName}/packetCaptures/${packetCaptureName}?api-version=2016-12-  
01"
```

NOTE

Deleting a packet capture does not delete the file in the storage account

Next steps

For instructions on downloading files from azure storage accounts, refer to [Get started with Azure Blob storage using .NET](#). Another tool that can be used is Storage Explorer. More information about Storage Explorer can be found here at the following link: [Storage Explorer](#)

Learn how to automate packet captures with Virtual machine alerts by viewing [Create an alert triggered packet capture](#)

Packet inspection with Azure Network Watcher

2/11/2020 • 5 minutes to read • [Edit Online](#)

Using the packet capture feature of Network Watcher, you can initiate and manage captures sessions on your Azure VMs from the portal, PowerShell, CLI, and programmatically through the SDK and REST API. Packet capture allows you to address scenarios that require packet level data by providing the information in a readily usable format. Leveraging freely available tools to inspect the data, you can examine communications sent to and from your VMs and gain insights into your network traffic. Some example uses of packet capture data include: investigating network or application issues, detecting network misuse and intrusion attempts, or maintaining regulatory compliance. In this article, we show how to open a packet capture file provided by Network Watcher using a popular open source tool. We will also provide examples showing how to calculate a connection latency, identify abnormal traffic, and examine networking statistics.

Before you begin

This article goes through some pre-configured scenarios on a packet capture that was run previously. These scenarios illustrate capabilities that can be accessed by reviewing a packet capture. This scenario uses [WireShark](#) to inspect the packet capture.

This scenario assumes you already ran a packet capture on a virtual machine. To learn how to create a packet capture visit [Manage packet captures with the portal](#) or with REST by visiting [Managing Packet Captures with REST API](#).

Scenario

In this scenario, you:

- Review a packet capture

Calculate network latency

In this scenario, we show how to view the initial Round Trip Time (RTT) of a Transmission Control Protocol (TCP) conversation occurring between two endpoints.

When a TCP connection is established, the first three packets sent in the connection follow a pattern commonly referred to as the three-way handshake. By examining the first two packets sent in this handshake, an initial request from the client and a response from the server, we can calculate the latency when this connection was established. This latency is referred to as the Round Trip Time (RTT). For more information on the TCP protocol and the three-way handshake refer to the following resource. <https://support.microsoft.com/en-us/help/172983/explanation-of-the-three-way-handshake-via-tcp-ip>

Step 1

Launch WireShark

Step 2

Load the **.cap** file from your packet capture. This file can be found in the blob it was saved in our locally on the virtual machine, depending on how you configured it.

Step 3

To view the initial Round Trip Time (RTT) in TCP conversations, we will only be looking at the first two packets involved in the TCP handshake. We will be using the first two packets in the three-way handshake, which are the

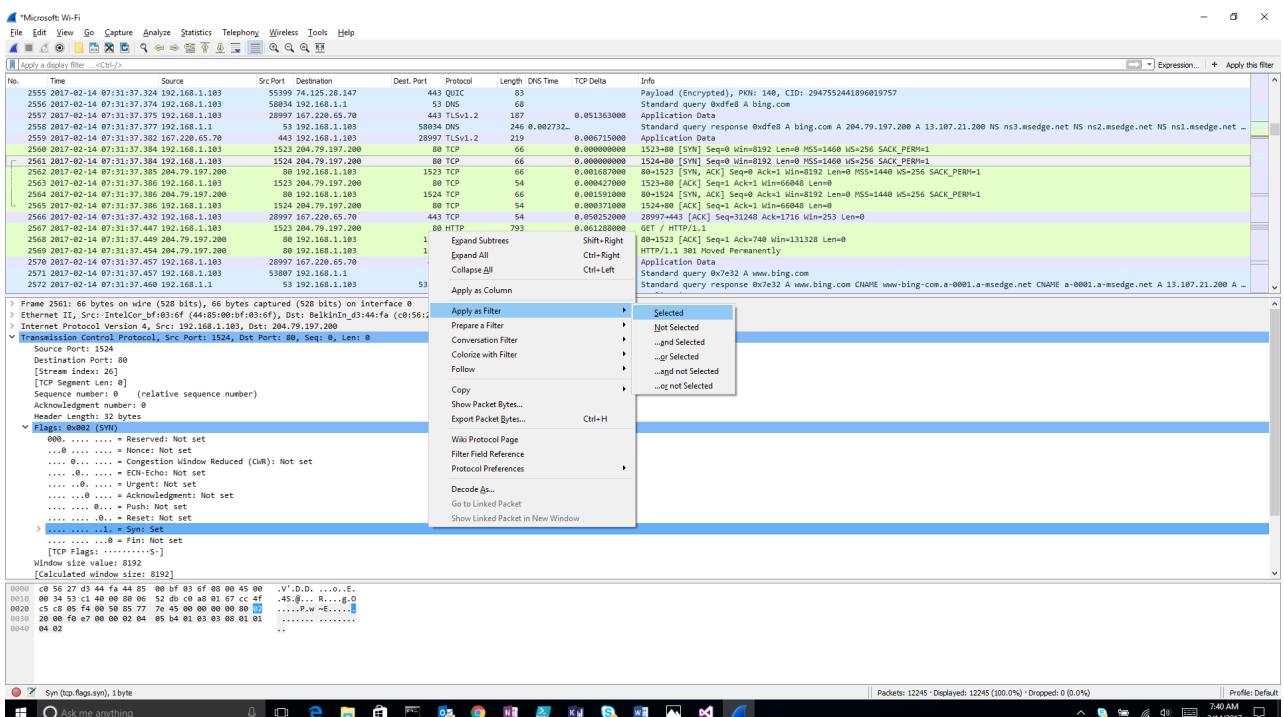
[SYN], [SYN, ACK] packets. They are named for flags set in the TCP header. The last packet in the handshake, the [ACK] packet, will not be used in this scenario. The [SYN] packet is sent by the client. Once it is received the server sends the [ACK] packet as an acknowledgment of receiving the SYN from the client. Leveraging the fact that the server's response requires very little overhead, we calculate the RTT by subtracting the time the [SYN, ACK] packet was received by the client by the time [SYN] packet was sent by the client.

Using Wireshark this value is calculated for us.

To more easily view the first two packets in the TCP three-way handshake, we will utilize the filtering capability provided by Wireshark.

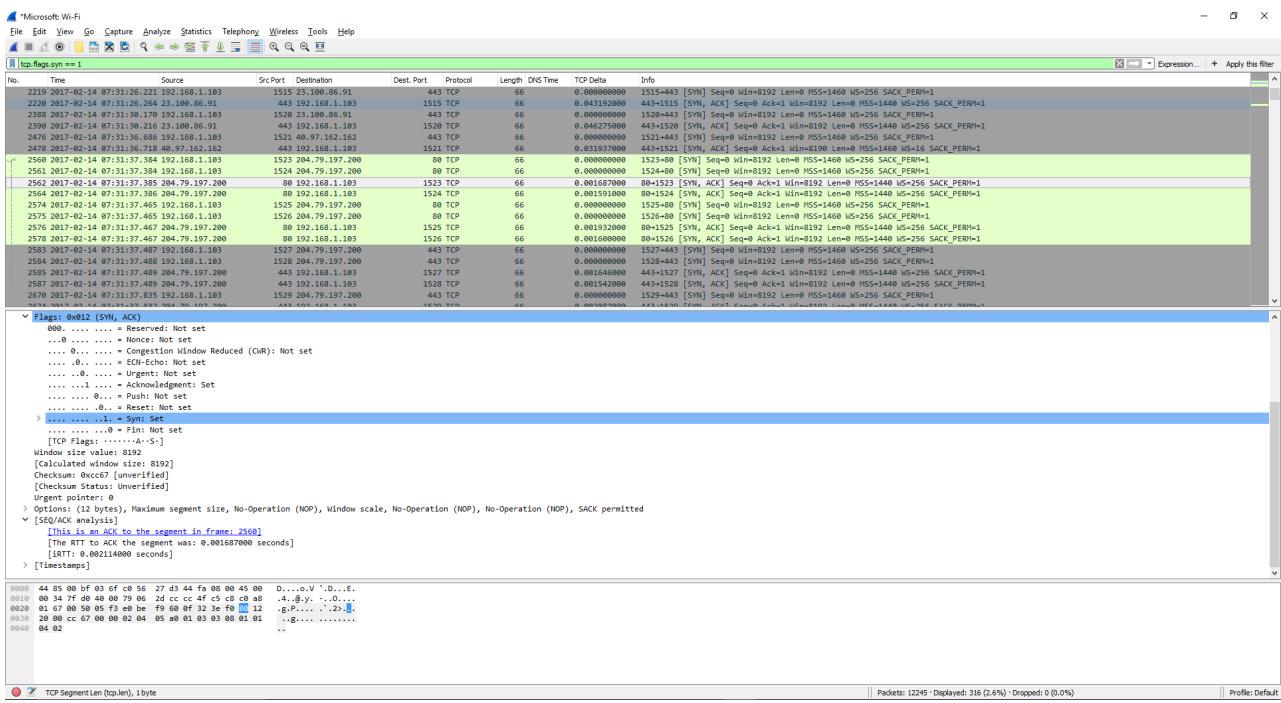
To apply the filter in Wireshark, expand the "Transmission Control Protocol" Segment of a [SYN] packet in your capture and examine the flags set in the TCP header.

Since we are looking to filter on all [SYN] and [SYN, ACK] packets, under flags confirm that the Syn bit is set to 1, then right click on the Syn bit -> Apply as Filter -> Selected.



Step 4

Now that you have filtered the window to only see packets with the [SYN] bit set, you can easily select conversations you are interested in to view the initial RTT. A simple way to view the RTT in Wireshark simply click the dropdown marked "SEQ/ACK" analysis. You will then see the RTT displayed. In this case, the RTT was 0.0022114 seconds, or 2.211 ms.



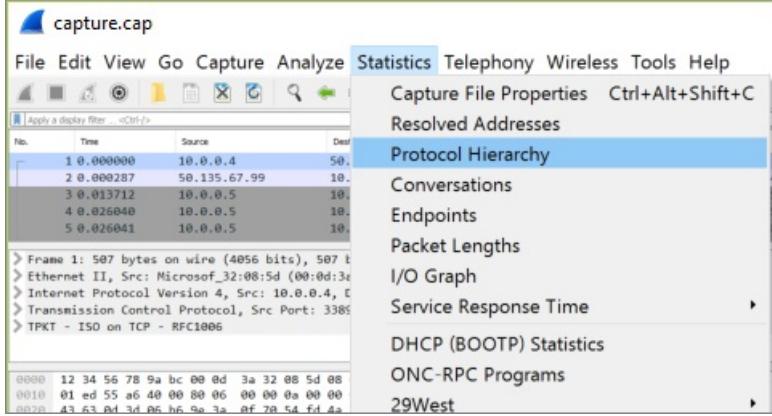
Unwanted protocols

You can have many applications running on a virtual machine instance you have deployed in Azure. Many of these applications communicate over the network, perhaps without your explicit permission. Using packet capture to store network communication, we can investigate how application are talking on the network and look for any issues.

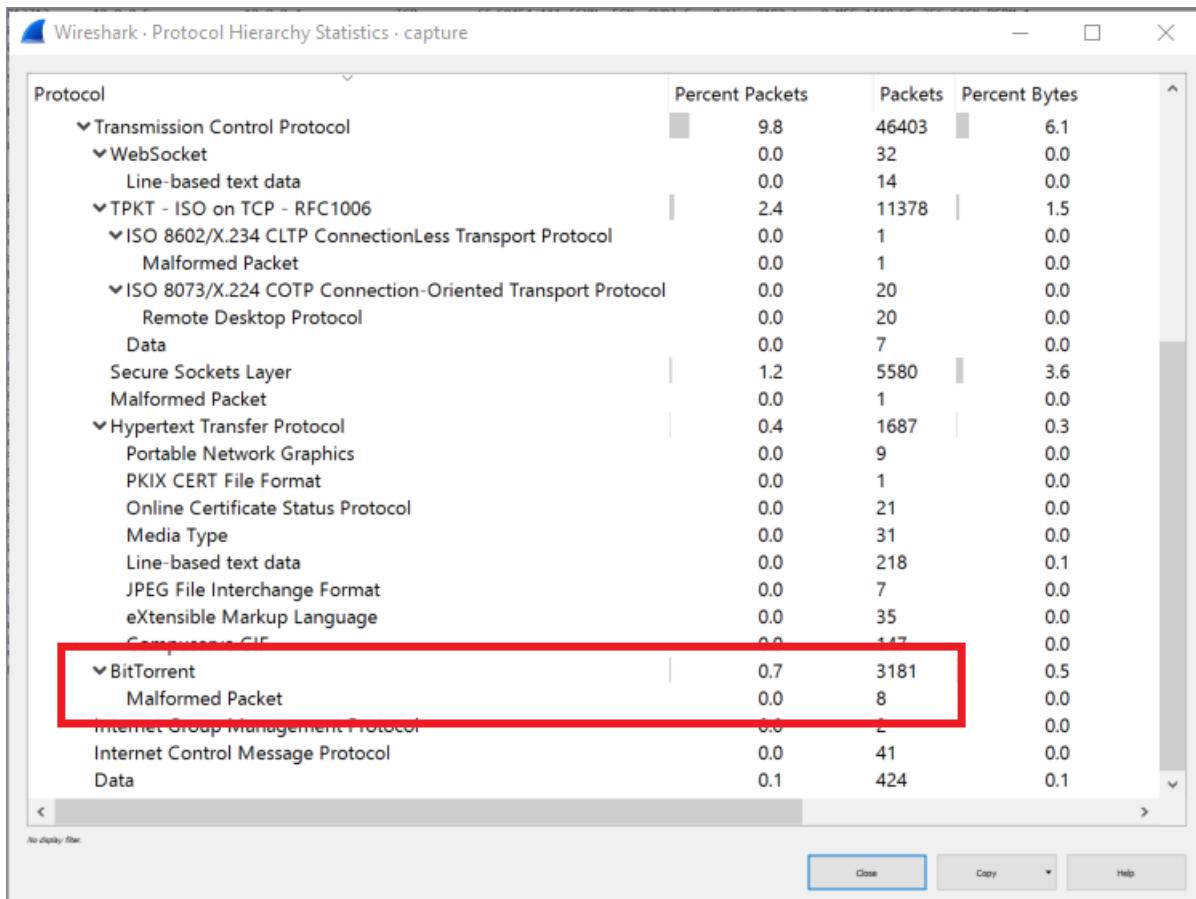
In this example, we review a previous ran packet capture for unwanted protocols that may indicate unauthorized communication from an application running on your machine.

Step 1

Using the same capture in the previous scenario click **Statistics > Protocol Hierarchy**



The protocol hierarchy window appears. This view provides a list of all the protocols that were in use during the capture session and the number of packets transmitted and received using the protocols. This view can be useful for finding unwanted network traffic on your virtual machines or network.



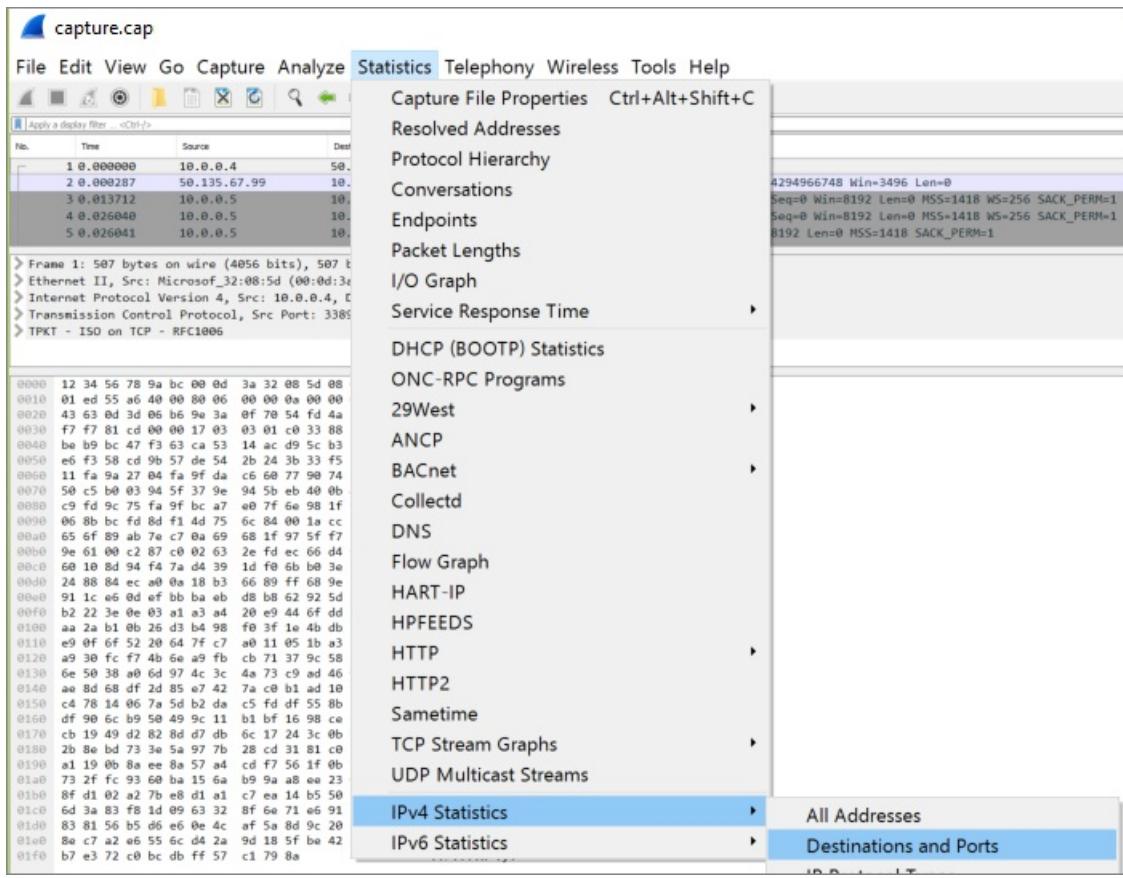
As you can see in the following screen capture, there was traffic using the BitTorrent protocol, which is used for peer to peer file sharing. As an administrator you do not expect to see BitTorrent traffic on this particular virtual machines. Now you aware of this traffic, you can remove the peer to peer software that installed on this virtual machine, or block the traffic using Network Security Groups or a Firewall. Additionally, you may elect to run packet captures on a schedule, so you can review the protocol use on your virtual machines regularly. For an example on how to automate network tasks in azure visit [Monitor network resources with azure automation](#)

Finding top destinations and ports

Understanding the types of traffic, the endpoints, and the ports communicated over is an important when monitoring or troubleshooting applications and resources on your network. Utilizing a packet capture file from above, we can quickly learn the top destinations our VM is communicating with and the ports being utilized.

Step 1

Using the same capture in the previous scenario click **Statistics > IPv4 Statistics > Destinations and Ports**



Step 2

As we look through the results a line stands out, there were multiple connections on port 111. The most used port was 3389, which is remote desktop, and the remaining are RPC dynamic ports.

While this traffic may mean nothing, it is a port that was used for many connections and is unknown to the administrator.

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Destinations and Ports 439950								
50.135.67.99	2527				0.0134	0.57%	0.7200	183.711
TCP	2527				0.0134	100.00%	0.7200	183.711
1718	2527				0.0134	100.00%	0.7200	183.711
10.0.0.4	355347				1.8878	80.77%	10.1100	176.010
TCP	9364				0.0497	2.64%	2.0500	186.042
3389	3263				0.0173	34.85%	0.5900	183.751
111	558				0.0030	5.96%	0.0700	0.229
52137	880				0.0047	9.40%	0.7300	1.853
51555	24				0.0001	0.26%	0.0200	9.929
51553	27				0.0001	0.29%	0.0300	116.153
52336	3				0.0000	0.03%	0.0100	31.588
52334	5				0.0000	0.05%	0.0300	31.588
52335	5				0.0000	0.05%	0.0300	31.589
52339	4				0.0000	0.04%	0.0200	32.104
52350	6				0.0000	0.06%	0.0200	32.706
52352	6				0.0000	0.06%	0.0200	32.707
52353	5				0.0000	0.05%	0.0200	32.708
52343	5				0.0000	0.05%	0.0200	66.652
52340	3				0.0000	0.03%	0.0100	31.598
52342	3				0.0000	0.03%	0.0200	32.128
52244	10				0.0001	0.20%	0.0000	29.025

Step 3

Now that we have determined an out of place port we can filter our capture based on the port.

The filter in this scenario would be:

```
tcp.port == 111
```

We enter the filter text from above in the filter textbox and hit enter.

The screenshot shows the Wireshark interface with a capture file named "capture.cap". The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. A toolbar with various icons is at the top. The main window displays a list of network packets. A green bar at the top of the list indicates the filter: "tcp.port == 111". The columns in the packet list are No., Time, Source, Destination, Protocol, Length, and Info. The "Info" column shows detailed protocol analysis for each packet. Below the list, several status messages are listed, starting with "Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0". The last message in the list is highlighted in blue: "Transmission Control Protocol, Src Port: 60454, Dst Port: 111, Seq: 0, Len: 0". At the bottom of the window, there is a hex dump of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.013712	10.0.0.5	10.0.0.4	TCP	66	60454->111 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0
4	0.026040	10.0.0.5	10.0.0.4	TCP	66	60426->111 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0
5	0.026041	10.0.0.5	10.0.0.4	TCP	62	60371->111 [SYN] Seq=0 Win=8192 Len=0 MSS=141
6	0.026041	10.0.0.5	10.0.0.4	TCP	62	60372->111 [SYN] Seq=0 Win=8192 Len=0 MSS=141
9	0.125538	10.0.0.5	10.0.0.4	TCP	66	60455->111 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0
10	0.229182	10.0.0.5	10.0.0.4	TCP	66	60428->111 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0
11	0.229183	10.0.0.5	10.0.0.4	TCP	62	60373->111 [SYN] Seq=0 Win=8192 Len=0 MSS=141
12	0.229183	10.0.0.5	10.0.0.4	TCP	66	60427->111 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0
13	0.229183	10.0.0.5	10.0.0.4	TCP	62	60374->111 [SYN] Seq=0 Win=8192 Len=0 MSS=141
14	0.235814	10.0.0.5	10.0.0.4	TCP	66	60456->111 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0
15	0.322607	10.0.0.5	10.0.0.4	TCP	62	60375->111 [SYN] Seq=0 Win=8192 Len=0 MSS=141
16	0.322608	10.0.0.5	10.0.0.4	TCP	66	60429->111 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0
17	0.347255	10.0.0.5	10.0.0.4	TCP	66	60457->111 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0

Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: CiscoInc_e5:81:af (18:8b:9d:e5:81:af), Dst: Microsoft_32:08:5d (00:0d:3a:32:08:5d)
Internet Protocol Version 4, Src: 10.0.0.5, Dst: 10.0.0.4
Transmission Control Protocol, Src Port: 60454, Dst Port: 111, Seq: 0, Len: 0

0000	00 0d 3a 32 08 5d 18 8b 9d e5 81 af 08 00 45 02	..:2.]..E.
0010	00 34 79 05 40 00 80 06 6d b4 0a 00 00 05 0a 00	.4y.@... m.....
0020	00 04 ec 26 00 6f 50 71 34 65 00 00 00 00 80 c2	...&.oPq 4e.....
0030	20 00 c9 05 00 00 02 04 05 8a 01 03 03 08 01 01
0040	04 02	..

From the results, we can see all the traffic is coming from a local virtual machine on the same subnet. If we still don't understand why this traffic is occurring, we can further inspect the packets to determine why it is making these calls on port 111. With this information we can take the appropriate action.

Next steps

Learn about the other diagnostic features of Network Watcher by visiting [Azure network monitoring overview](#)

Use packet capture for proactive network monitoring with alerts and Azure Functions

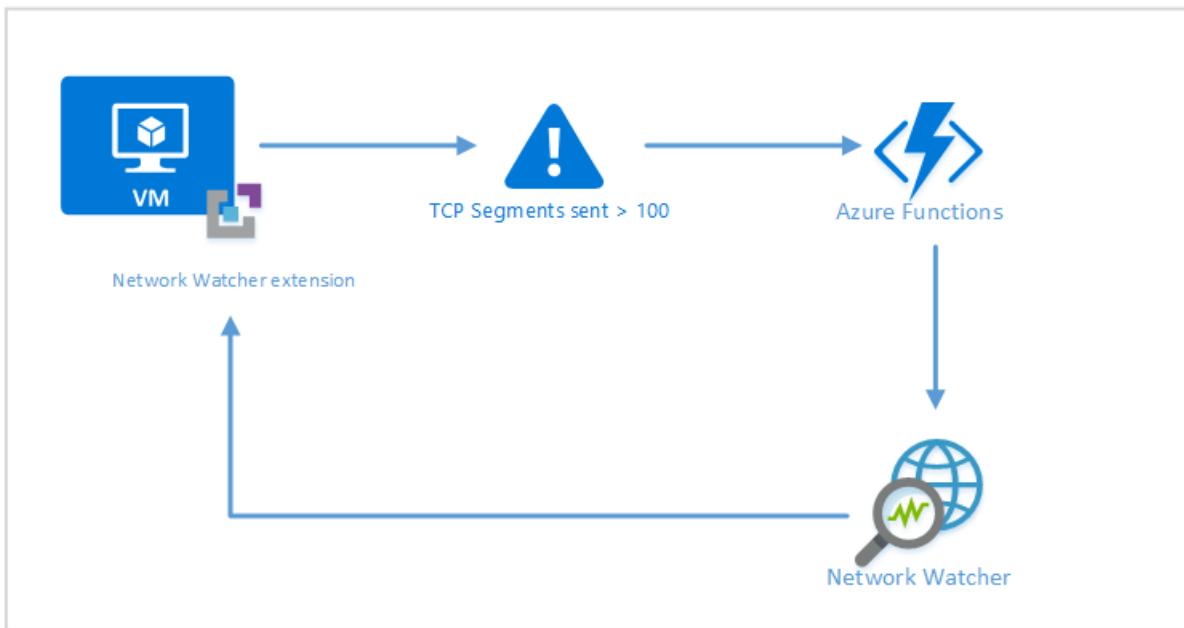
1/28/2020 • 10 minutes to read • [Edit Online](#)

Network Watcher packet capture creates capture sessions to track traffic in and out of virtual machines. The capture file can have a filter that is defined to track only the traffic that you want to monitor. This data is then stored in a storage blob or locally on the guest machine.

This capability can be started remotely from other automation scenarios such as Azure Functions. Packet capture gives you the capability to run proactive captures based on defined network anomalies. Other uses include gathering network statistics, getting information about network intrusions, debugging client-server communications, and more.

Resources that are deployed in Azure run 24/7. You and your staff cannot actively monitor the status of all resources 24/7. For example, what happens if an issue occurs at 2 AM?

By using Network Watcher, alerting, and functions from within the Azure ecosystem, you can proactively respond with the data and tools to solve problems in your network.



NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Prerequisites

- The latest version of [Azure PowerShell](#).
- An existing instance of Network Watcher. If you don't already have one, [create an instance of Network Watcher](#).
- An existing virtual machine in the same region as Network Watcher with the [Windows extension](#) or [Linux](#)

virtual machine extension.

Scenario

In this example, your VM is sending more TCP segments than usual, and you want to be alerted. TCP segments are used as an example here, but you can use any alert condition.

When you are alerted, you want to receive packet-level data to understand why communication has increased. Then you can take steps to return the virtual machine to regular communication.

This scenario assumes that you have an existing instance of Network Watcher and a resource group with a valid virtual machine.

The following list is an overview of the workflow that takes place:

1. An alert is triggered on your VM.
2. The alert calls your Azure function via a webhook.
3. Your Azure function processes the alert and starts a Network Watcher packet capture session.
4. The packet capture runs on the VM and collects traffic.
5. The packet capture file is uploaded to a storage account for review and diagnosis.

To automate this process, we create and connect an alert on our VM to trigger when the incident occurs. We also create a function to call into Network Watcher.

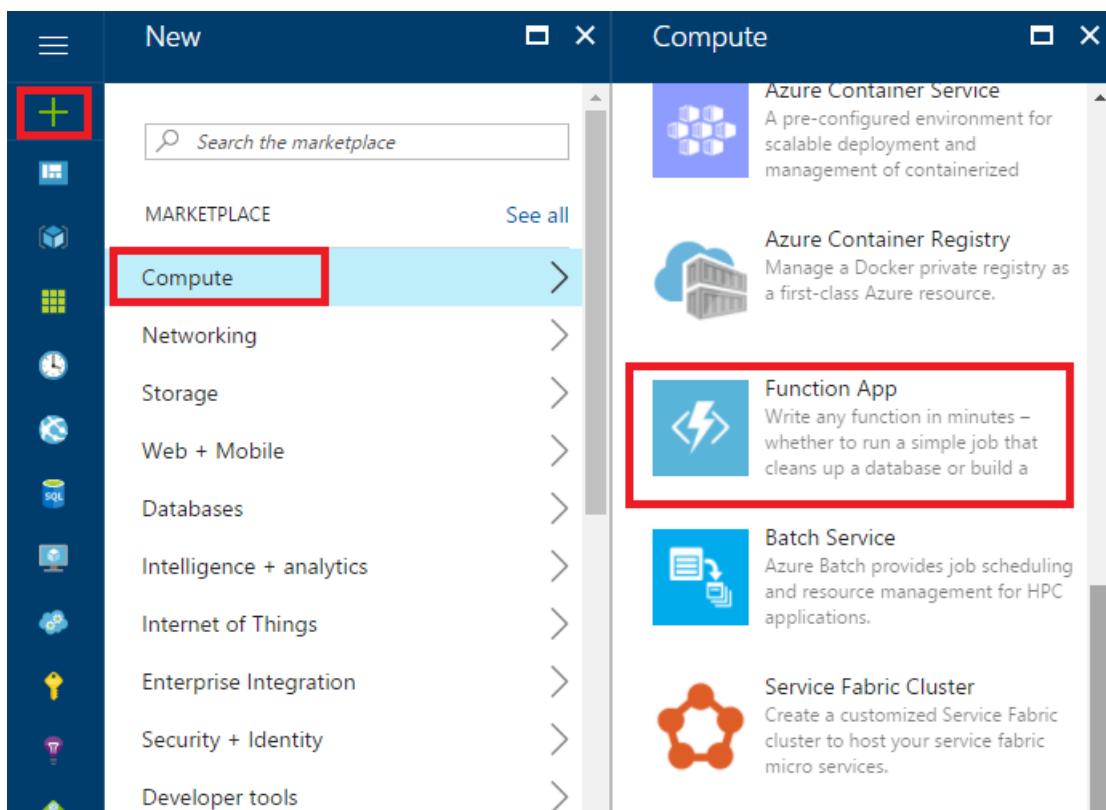
This scenario does the following:

- Creates an Azure function that starts a packet capture.
- Creates an alert rule on a virtual machine and configures the alert rule to call the Azure function.

Create an Azure function

The first step is to create an Azure function to process the alert and create a packet capture.

1. In the [Azure portal](#), select **Create a resource > Compute > Function App**.



2. On the **Function App** blade, enter the following values, and then select **OK** to create the app:

SETTING	VALUE	DETAILS
App name	PacketCaptureExample	The name of the function app.
Subscription	[Your subscription]The subscription for which to create the function app.	
Resource Group	PacketCaptureRG	The resource group to contain the function app.
Hosting Plan	Consumption Plan	The type of plan your function app uses. Options are Consumption or Azure App Service plan.
Location	Central US	The region in which to create the function app.
Storage Account	{autogenerated}	The storage account that Azure Functions needs for general-purpose storage.

3. On the **PacketCaptureExample Function Apps** blade, select **Functions > Custom function > +**.

4. Select **HttpTrigger-Powershell**, and then enter the remaining information. Finally, to create the function, select **Create**.

SETTING	VALUE	DETAILS
Scenario	Experimental	Type of scenario
Name your function	AlertPacketCapturePowerShell	Name of the function
Authorization level	Function	Authorization level for the function

contosofunctions
Function app

Search

Functions

+ New Function

Proxies (preview)

+ New proxy

Choose a template

Language: PowerShell Scenario: Experimental

HttpTrigger-Powershell (Preview) A PowerShell function that will be run whenever it receives an HTTP request	QueueTrigger-Powershell (Preview) A PowerShell function that will be run whenever a message is added to a specified Azure Queue Storage	TimerTrigger-Powershell (Preview) A PowerShell function that will be run on a specified schedule
---	--	---

This template is experimental and does not yet have full support. If you run into issues, please file a bug on our [GitHub repository](#).

Name your function

AlertPacketCapturePowerShell

HTTP trigger (req)

Authorization level ⓘ

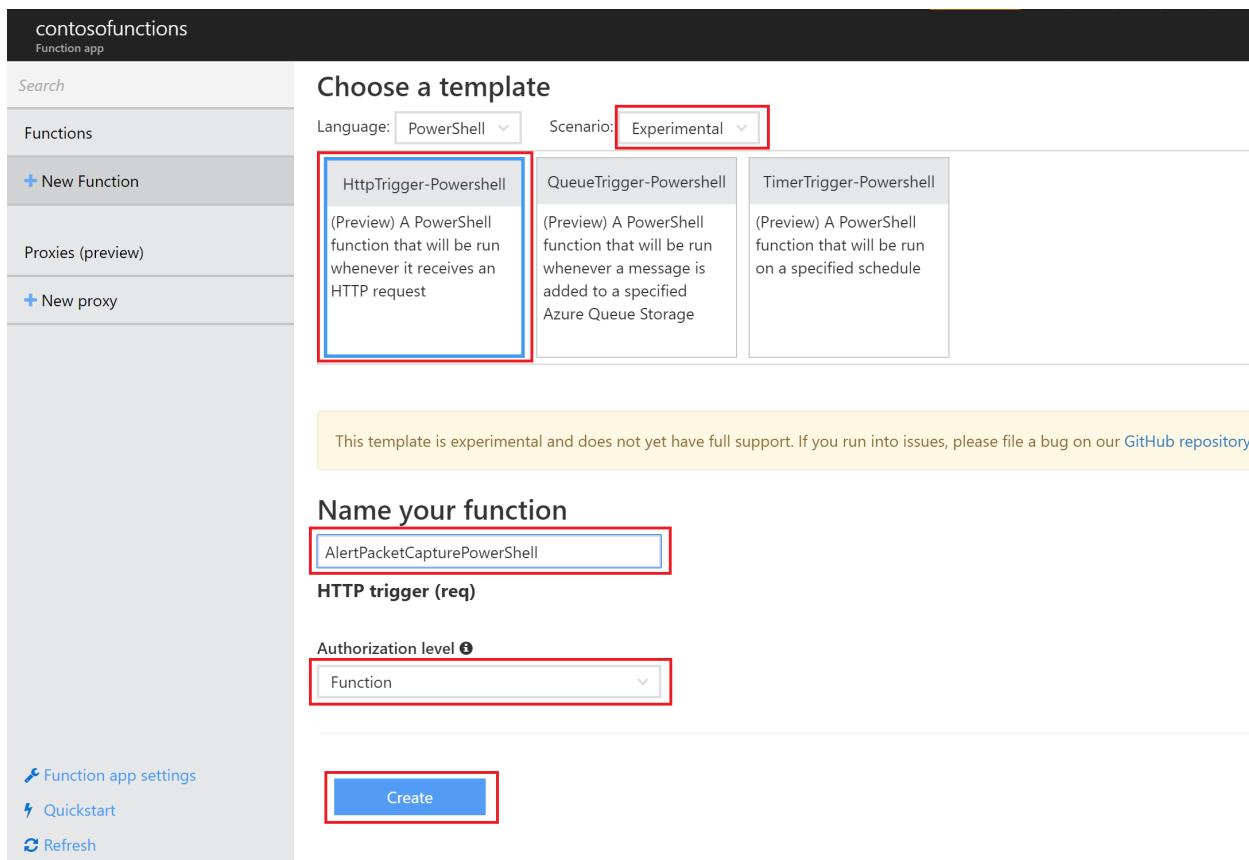
Function

Function app settings

Quickstart

Refresh

Create



NOTE

The PowerShell template is experimental and does not have full support.

Customizations are required for this example and are explained in the following steps.

Add modules

To use Network Watcher PowerShell cmdlets, upload the latest PowerShell module to the function app.

1. On your local machine with the latest Azure PowerShell modules installed, run the following PowerShell command:

```
(Get-Module Az.Network).Path
```

This example gives you the local path of your Azure PowerShell modules. These folders are used in a later step. The modules that are used in this scenario are:

- Az.Network
- Az.Accounts
- Az.Resources

File Explorer view of Azure Resource Manager PowerShell modules:

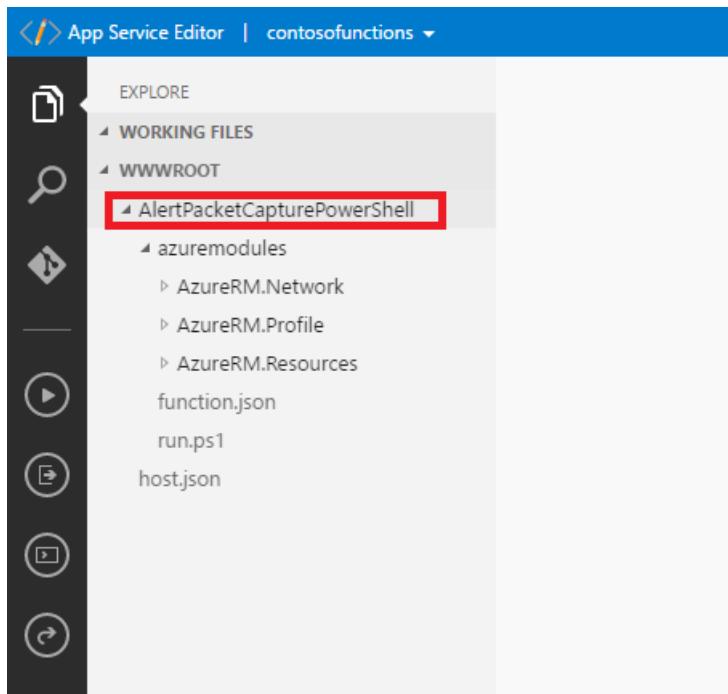
Name	Date modified	Type	Size
AzureRM.KeyVault	2/23/2017 9:05 AM	File folder	
AzureRM.LogicApp	2/23/2017 9:05 AM	File folder	
AzureRM.MachineLearning	2/23/2017 9:05 AM	File folder	
AzureRM.Media	2/23/2017 9:05 AM	File folder	
<input checked="" type="checkbox"/> AzureRM.Network	2/23/2017 9:05 AM	File folder	
AzureRM.NotificationHubs	2/23/2017 9:05 AM	File folder	
AzureRM.OperationalInsights	2/23/2017 9:05 AM	File folder	
AzureRM.PowerBIEmbedded	2/23/2017 9:05 AM	File folder	
<input checked="" type="checkbox"/> AzureRM.Profile	2/23/2017 9:05 AM	File folder	
AzureRM.RecoveryServices	2/23/2017 9:05 AM	File folder	
AzureRM.RecoveryServices.Backup	2/23/2017 9:05 AM	File folder	
AzureRM.RedisCache	2/23/2017 9:05 AM	File folder	
<input checked="" type="checkbox"/> AzureRM.Resources	2/23/2017 9:05 AM	File folder	
AzureRM.Scheduler	2/23/2017 9:05 AM	File folder	
AzureRM.ServerManagement	2/23/2017 9:05 AM	File folder	
AzureRM.ServiceBus	2/23/2017 9:05 AM	File folder	
AzureRM.SiteRecovery	2/23/2017 9:05 AM	File folder	
AzureRM.Sql	2/23/2017 9:05 AM	File folder	

2. Select **Function app settings** > **Go to App Service Editor**.

The screenshot shows the Azure Functions portal interface for a function named "contosofunctions". On the left, there's a sidebar with options like "Functions", "+ New Function", and "Proxies (preview)". At the bottom of the sidebar, the "Function app settings" link is highlighted with a red box. The main content area is divided into sections: "Develop", "Deploy", and "Manage". In the "Develop" section, the "App Service Editor" link is highlighted with a red box. Other links in this section include "Configure app settings" and "Open dev console". In the "Deploy" section, there are links for "Configure continuous integration" and "Go to Kudu". In the "Manage" section, there are links for "Go to App Service Settings", "Configure CORS", "Configure authentication", and "Configure API metadata".

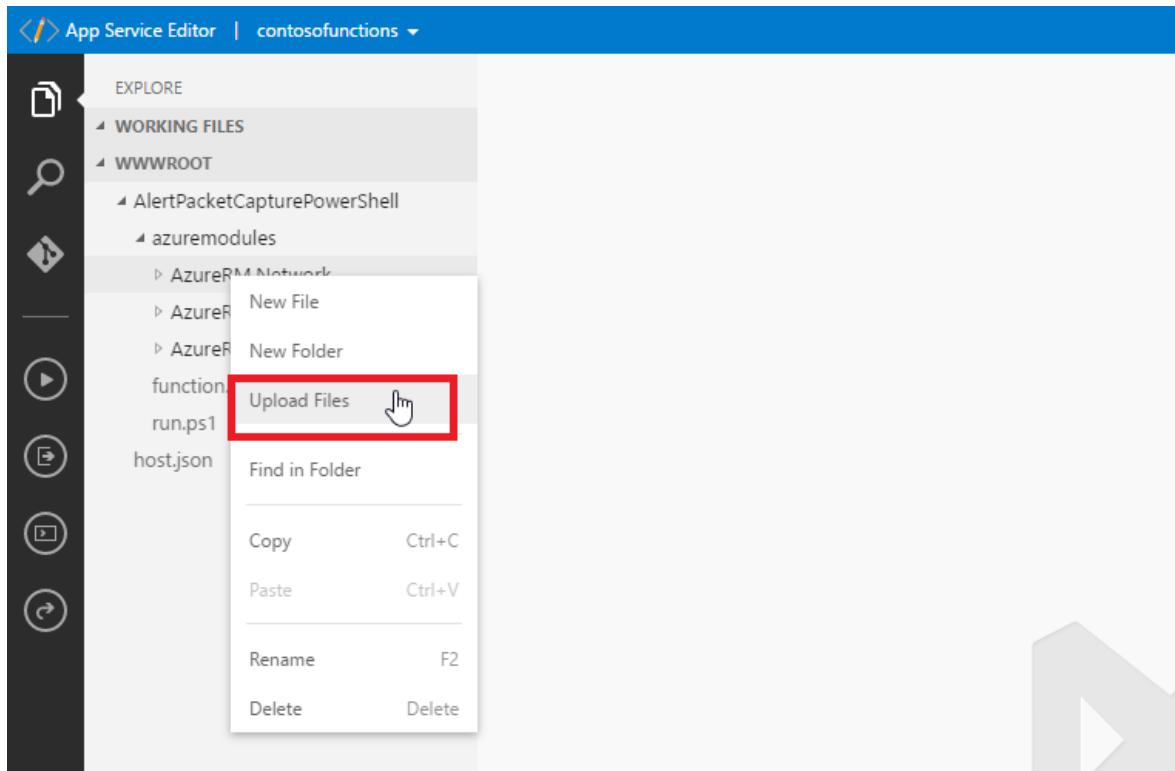
3. Right-click the **AlertPacketCapturePowershell** folder, and then create a folder called **azuremodules**.

4. Create a subfolder for each module that you need.



- Az.Network
- Az.Accounts
- Az.Resources

5. Right-click the **Az.Network** subfolder, and then select **Upload Files**.
6. Go to your Azure modules. In the local **Az.Network** folder, select all the files in the folder. Then select **OK**.
7. Repeat these steps for **Az.Accounts** and **Az.Resources**.



8. After you've finished, each folder should have the PowerShell module files from your local machine.

The screenshot shows the 'App Service Editor' interface with the title bar 'App Service Editor | contosofunctions'. On the left is a vertical toolbar with icons for file operations. The main area is titled 'EXPLORE' and shows the file structure of the 'contosofunctions' function app. The 'WWWROOT' folder is expanded, revealing several subfolders: 'AlertPacketCapturePowerShell', 'azuremodules', and 'AzureRM.Profile'. The 'AzureRM.Profile' folder is currently selected, indicated by a blue background. Within 'AzureRM.Profile', a list of files is displayed, including PowerShell script files like 'AzureRM.Profile.psd1' and 'AzureRmProfileStartup.ps1', and various DLL files from the Microsoft Azure command-line tools. A large, semi-transparent play button icon is overlaid on the right side of the editor window.

Authentication

To use the PowerShell cmdlets, you must authenticate. You configure authentication in the function app. To configure authentication, you must configure environment variables and upload an encrypted key file to the function app.

NOTE

This scenario provides just one example of how to implement authentication with Azure Functions. There are other ways to do this.

Encrypted credentials

The following PowerShell script creates a key file called **PassEncryptKey.key**. It also provides an encrypted version of the password that's supplied. This password is the same password that is defined for the Azure Active Directory application that's used for authentication.

```

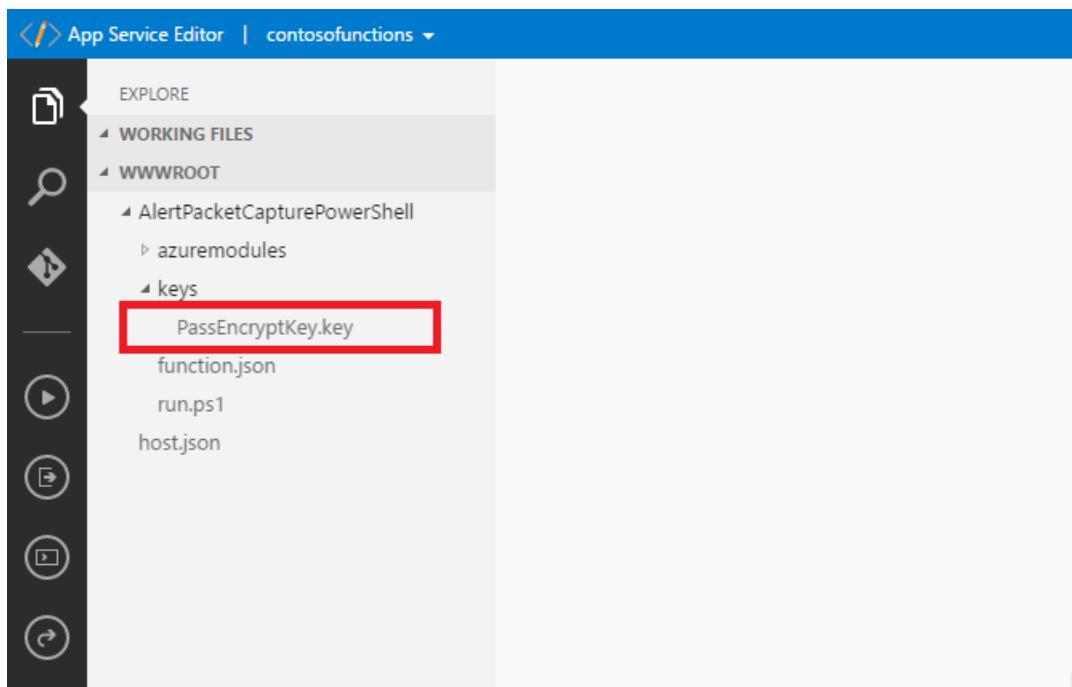
#Variables
$keypath = "C:\temp\PassEncryptKey.key"
$AESKey = New-Object Byte[] 32
$Password = "<insert a password here>

#Keys
[Security.Cryptography.RNGCryptoServiceProvider]::Create().GetBytes($AESKey)
Set-Content $keypath $AESKey

#Get encrypted password
$secPw = ConvertTo-SecureString -AsPlainText $Password -Force
$AESKey = Get-content $KeyPath
$Encryptedpassword = $secPw | ConvertFrom-SecureString -Key $AESKey
$Encryptedpassword

```

In the App Service Editor of the function app, create a folder called **keys** under **AlertPacketCapturePowerShell**. Then upload the **PassEncryptKey.key** file that you created in the previous PowerShell sample.



Retrieve values for environment variables

The final requirement is to set up the environment variables that are necessary to access the values for authentication. The following list shows the environment variables that are created:

- AzureClientId
- AzureTenant
- AzureCredPassword

AzureClientId

The client ID is the Application ID of an application in Azure Active Directory.

1. If you don't already have an application to use, run the following example to create an application.

```
$app = New-AzADApplication -DisplayName "ExampleAutomationAccount_MF" -HomePage
"https://exampleapp.com" -IdentifierUris "https://exampleapp1.com/ExampleFunctionsAccount" -Password "<
same password as defined earlier>"
New-AzADServicePrincipal -ApplicationId $app.ApplicationId
Start-Sleep 15
New-AzRoleAssignment -RoleDefinitionName Contributor -ServicePrincipalName $app.ApplicationId
```

NOTE

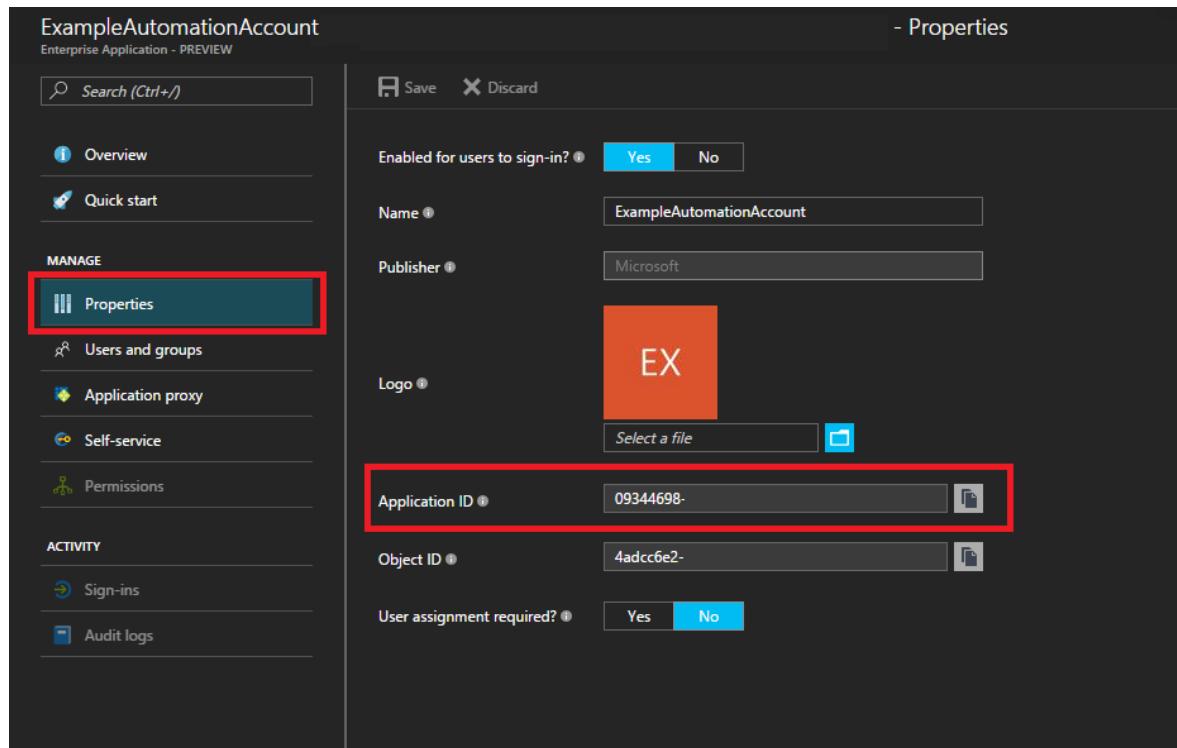
The password that you use when creating the application should be the same password that you created earlier when saving the key file.

2. In the Azure portal, select **Subscriptions**. Select the subscription to use, and then select **Access control (IAM)**.

The screenshot shows three windows from the Azure portal:

- Subscriptions**: Shows a list of subscriptions. The "Microsoft" subscription is selected and highlighted with a red box. It has dropdown filters for Role (All) and Status (All), and a blue "Apply" button.
- Microsoft Azure Subscription**: Shows the "Access control (IAM)" section highlighted with a red box. It includes a search bar, a "Overview" link, and a "Diagnose and solve problems" link.
- Access control (IAM)**: Shows a list of role assignments. It has filters for Name, Type (All), Role (2 selected), and Scope (All scopes). It lists three items: "ExampleAutomationAcco" (App, Contributor, Assigned), "gwazureauto_19vtCxvP" (App, Contributor, Assigned), and "Subscription admins" (Group, Owner, Inherited (Subscription)). The first item is also highlighted with a red box.

3. Choose the account to use, and then select **Properties**. Copy the Application ID.



AzureTenant

Obtain the tenant ID by running the following PowerShell sample:

```
(Get-AzSubscription -SubscriptionName "<subscriptionName>").TenantId
```

AzureCredPassword

The value of the AzureCredPassword environment variable is the value that you get from running the following PowerShell sample. This example is the same one that's shown in the preceding **Encrypted credentials** section. The value that's needed is the output of the `$Encryptedpassword` variable. This is the service principal password that you encrypted by using the PowerShell script.

```
#Variables
$keypath = "C:\temp\PassEncryptKey.key"
$AESKey = New-Object Byte[] 32
$Password = "<insert a password here>

#Keys
[Security.Cryptography.RNGCryptoServiceProvider]::Create().GetBytes($AESKey)
Set-Content $keypath $AESKey

#Get encrypted password
$secPw = ConvertTo-SecureString -AsPlainText $Password -Force
$AESKey = Get-content $KeyPath
$Encryptedpassword = $secPw | ConvertFrom-SecureString -Key $AESKey
$Encryptedpassword
```

Store the environment variables

1. Go to the function app. Then select **Function app settings > Configure app settings**.

contosofunctions

Function app

Search

Functions

+ New Function

⚡ AlertPacketCapturePowerShell

Proxies (preview)

+ New proxy

Function app settings >

Develop

App Service Editor
In portal editor with an integrated console and streaming logs

Go to App Service Editor

Configure app settings

Application settings
Manage environment variables and connection strings for your function app

Dev Console
In-portal console for accessing your function app's file system

Open dev console

Deploy

Continuous Integration
Deploy your function code from GitHub, Visual Studio Team Services, and more

Configure continuous integration

Kudu
Access advanced functionality of App Service like uploading zips, killing processes, and more

Go to Kudu

Manage

App Service Settings
Advanced Features. Access all the underlying features of Azure App Service

Go to App Service Settings

CORS
Allow your HTTP-triggered functions to be called from within a web browser

Configure CORS

Authentication/Authorization
For functions that use the HTTP trigger, you can require calls to be authenticated

Configure authentication

API definition
Allow clients to more easily consume your HTTP-triggered functions

Configure API metadata

Daily Usage Quota (GB-Sec)

Enter value in GB-sec

Set quota

The screenshot shows the Azure Functions portal interface. On the left, there's a sidebar with navigation links like 'Functions', '+ New Function', and 'Function app settings'. The main area is divided into sections: 'Develop', 'Deploy', 'Manage', and 'Daily Usage Quota (GB-Sec)'. In the 'Manage' section, there are four buttons: 'Go to App Service Settings', 'Configure CORS', 'Configure authentication', and 'Configure API metadata'. Below this is a 'Daily Usage Quota (GB-Sec)' section with an input field and a 'Set quota' button. Two specific buttons are highlighted with red boxes: 'Configure app settings' under the 'Manage' section and 'Function app settings' in the sidebar.

2. Add the environment variables and their values to the app settings, and then select **Save**.

The screenshot shows the 'Application settings' page for a function named 'contosofunctions'. At the top, there are 'Save' and 'Discard' buttons. Below that, under 'Debugging', there is a 'Remote debugging' switch set to 'Off' and a 'Remote Visual Studio version' dropdown set to '2012'. The main section is 'App settings', which lists several environment variables:

Setting	Value	Slot setting	...
AzureWebJobsDashboard	DefaultEndpointsProtocol=https	<input type="checkbox"/>	Slot setting
AzureWebJobsStorage	DefaultEndpointsProtocol=https	<input type="checkbox"/>	Slot setting
FUNCTIONS_EXTENSION_VERSION	v1	<input type="checkbox"/>	Slot setting
WEBSITE_CONTENTAZUREFILECONNECTIONSTRING	DefaultEndpointsProtocol=https	<input type="checkbox"/>	Slot setting
WEBSITE_CONTENTSHARE	contosofunctions	<input type="checkbox"/>	Slot setting
WEBSITE_NODE_DEFAULT_VERSION	6.5.0	<input type="checkbox"/>	Slot setting
AzureTenant	72f988bf-86f1	<input type="checkbox"/>	Slot setting
AzureClientID	8b29f95f-d110	<input type="checkbox"/>	Slot setting
AzureCredPassword	76492d111	<input type="checkbox"/>	Slot setting

Below the app settings is a 'Connection strings' section with a 'No results' message and a search bar.

Add PowerShell to the function

It's now time to make calls into Network Watcher from within the Azure function. Depending on the requirements, the implementation of this function can vary. However, the general flow of the code is as follows:

1. Process input parameters.
2. Query existing packet captures to verify limits and resolve name conflicts.
3. Create a packet capture with appropriate parameters.
4. Poll packet capture periodically until it's complete.
5. Notify the user that the packet capture session is complete.

The following example is PowerShell code that can be used in the function. There are values that need to be replaced for **subscriptionId**, **resourceGroupName**, and **storageAccountName**.

```
#Import Azure PowerShell modules required to make calls to Network Watcher
Import-Module
"D:\home\site\wwwroot\AlertPacketCapturePowerShell\azurerm\Az.Accounts.ps1" -Global
Import-Module
"D:\home\site\wwwroot\AlertPacketCapturePowerShell\azurerm\Az.Network.ps1" -Global
Import-Module
"D:\home\site\wwwroot\AlertPacketCapturePowerShell\azurerm\Az.Resources.ps1" -Global

#Process alert request body
```

```

$ requestBody = Get-Content $req -Raw | ConvertFrom-Json

#Storage account ID to save captures in
$storageaccountid =
"/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/{storageAccountName}"

#Packet capture vars
$packetcapturename = "PSAzureFunction"
$packetCaptureLimit = 10
$packetCaptureDuration = 10

#Credentials
$tenant = $env:AzureTenant
$pw = $env:AzureCredPassword
$clientid = $env:AzureClientId
$keypath = "D:\home\site\wwwroot\AlertPacketCapturePowerShell\keys\PassEncryptKey.key"

#Authentication
$secpassword = $pw | ConvertTo-SecureString -Key (Get-Content $keypath)
$credential = New-Object System.Management.Automation.PSCredential ($clientid, $secpassword)
Connect-AzAccount -ServicePrincipal -Tenant $tenant -Credential $credential #-WarningAction
SilentlyContinue | out-null

#Get the VM that fired the alert
if($requestBody.context.resourceType -eq "Microsoft.Compute/virtualMachines")
{
    Write-Output ("Subscription ID: {0}" -f $requestBody.context.subscriptionId)
    Write-Output ("Resource Group: {0}" -f $requestBody.context.resourceGroupName)
    Write-Output ("Resource Name: {0}" -f $requestBody.context.resourceName)
    Write-Output ("Resource Type: {0}" -f $requestBody.context.resourceType)

    #Get the Network Watcher in the VM's region
    $networkWatcher = Get-AzResource | Where {$_.ResourceType -eq
    "Microsoft.Network/networkWatchers" -and $_.Location -eq $requestBody.context.resourceRegion}

    #Get existing packetCaptures
    $packetCaptures = Get-AzNetworkWatcherPacketCapture -NetworkWatcher $networkWatcher

    #Remove existing packet capture created by the function (if it exists)
    $packetCaptures | %{$if($_.Name -eq $packetCaptureName)
    {
        Remove-AzNetworkWatcherPacketCapture -NetworkWatcher $networkWatcher -PacketCaptureName
    $packetCaptureName
    }}

    #Initiate packet capture on the VM that fired the alert
    if ((Get-AzNetworkWatcherPacketCapture -NetworkWatcher $networkWatcher).Count -lt
    $packetCaptureLimit){
        echo "Initiating Packet Capture"
        New-AzNetworkWatcherPacketCapture -NetworkWatcher $networkWatcher -
        TargetVirtualMachineId $requestBody.context.resourceId -PacketCaptureName $packetCaptureName -
        StorageAccountId $storageaccountid -TimeLimitInSeconds $packetCaptureDuration
        Out-File -Encoding Ascii -FilePath $res -inputObject "Packet Capture created on
    ${requestBody.context.resourceID}"
    }
}

```

Retrieve the function URL

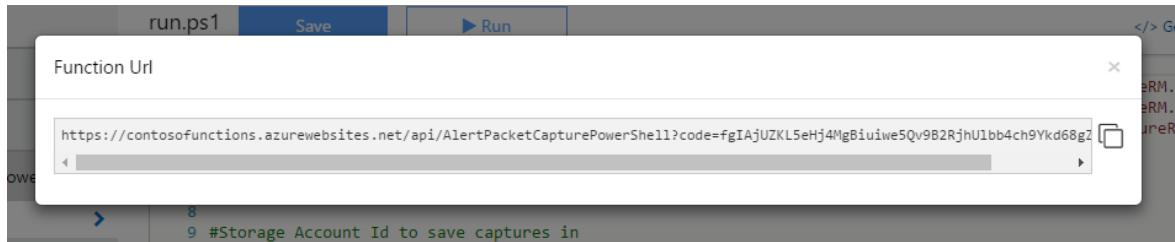
- After you've created your function, configure your alert to call the URL that's associated with the function. To get this value, copy the function URL from your function app.

```

1 #Import Azure PowerShell modules required to make calls to Network Watcher
2 Import-Module "D:\home\site\wwwroot\AlertPacketCapturePowerShell\azuremodules\AzureRM.Profile\AzureRM.Profile.psd1"
3 Import-Module "D:\home\site\wwwroot\AlertPacketCapturePowerShell\azuremodules\AzureRM.Network\AzureRM.Network.psd1"
4 Import-Module "D:\home\site\wwwroot\AlertPacketCapturePowerShell\azuremodules\AzureRM.Resources\AzureRM.Resources.psd1"
5
6 #Process Alert Request Body
7 $requestBody = Get-Content $req -Raw | ConvertFrom-Json
8
9 #Storage Account Id to save captures in
10 $storageaccountid = "/subscriptions/{subscriptionID}/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/{storageAccountName}"
11
12 #Packet Capture Vars
13 $packetcapturename = "PSAzureFunction"
14 $packetcapturesize = 10

```

2. Copy the function URL for your function app.



If you require custom properties in the payload of the webhook POST request, refer to [Configure a webhook on an Azure metric alert](#).

Configure an alert on a VM

Alerts can be configured to notify individuals when a specific metric crosses a threshold that's assigned to it. In this example, the alert is on the TCP segments that are sent, but the alert can be triggered for many other metrics. In this example, an alert is configured to call a webhook to call the function.

Create the alert rule

Go to an existing virtual machine, and then add an alert rule. More detailed documentation about configuring alerts can be found at [Create alerts in Azure Monitor for Azure services - Azure portal](#). Enter the following values in the **Alert rule** blade, and then select **OK**.

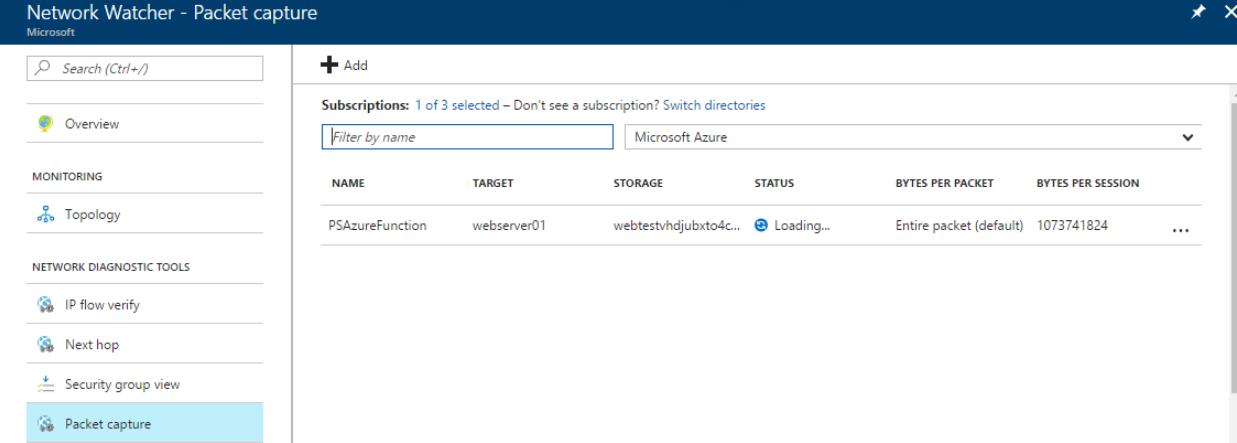
SETTING	VALUE	DETAILS
Name	TCP_Segments_Sent_Exceeded	Name of the alert rule.
Description	TCP segments sent exceeded threshold	The description for the alert rule.
Metric	TCP segments sent	The metric to use to trigger the alert.
Condition	Greater than	The condition to use when evaluating the metric.
Threshold	100	The value of the metric that triggers the alert. This value should be set to a valid value for your environment.
Period	Over the last five minutes	Determines the period in which to look for the threshold on the metric.
Webhook	[webhook URL from function app]	The webhook URL from the function app that was created in the previous steps.

NOTE

The TCP segments metric is not enabled by default. Learn more about how to enable additional metrics by visiting [Enable monitoring and diagnostics](#).

Review the results

After the criteria for the alert triggers, a packet capture is created. Go to Network Watcher, and then select **Packet capture**. On this page, you can select the packet capture file link to download the packet capture.



The screenshot shows the Microsoft Azure portal interface for Network Watcher. The left sidebar has a 'Search (Ctrl+ /)' bar and links for Overview, MONITORING (Topology), NETWORK DIAGNOSTIC TOOLS (IP flow verify, Next hop, Security group view), and Packet capture (which is highlighted with a blue background). The main content area is titled 'Subscriptions: 1 of 3 selected – Don't see a subscription? Switch directories' and shows a table with one row:

NAME	TARGET	STORAGE	STATUS	BYTES PER PACKET	BYTES PER SESSION
PSAzureFunction	webserver01	webtestvhdjubxto4c...	>Loading...	Entire packet (default)	1073741824

If the capture file is stored locally, you can retrieve it by signing in to the virtual machine.

For instructions about downloading files from Azure storage accounts, see [Get started with Azure Blob storage using .NET](#). Another tool you can use is [Storage Explorer](#).

After your capture has been downloaded, you can view it by using any tool that can read a **.cap** file. Following are links to two of these tools:

- [Microsoft Message Analyzer](#)
- [WireShark](#)

Next steps

Learn how to view your packet captures by visiting [Packet capture analysis with Wireshark](#).

Perform network intrusion detection with Network Watcher and open source tools

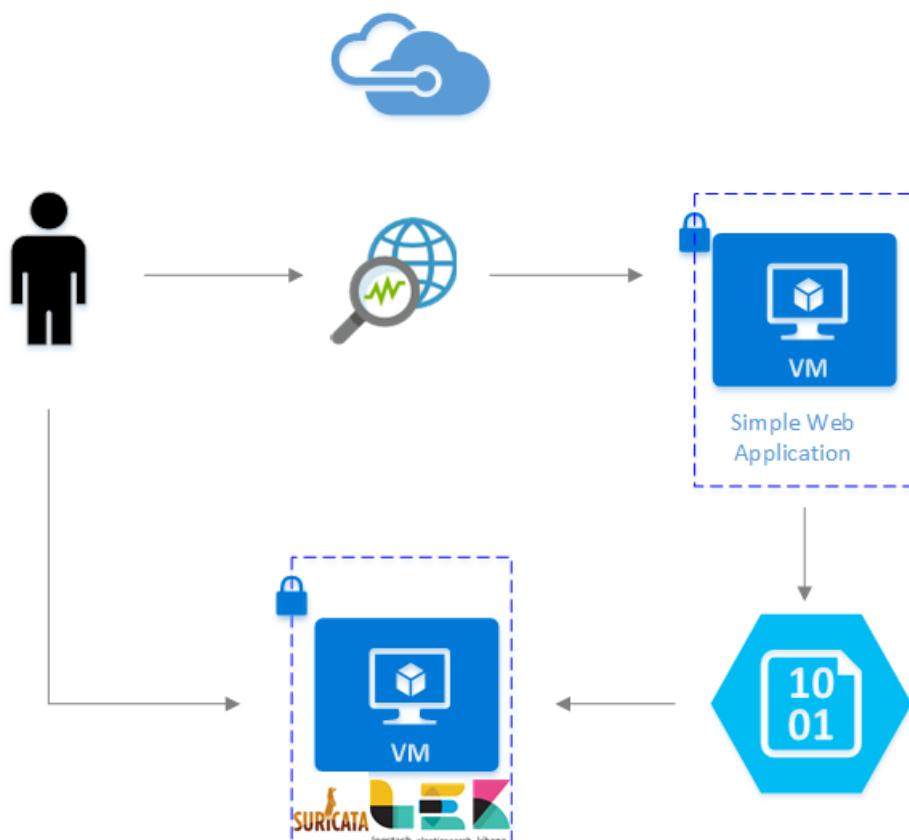
1/28/2020 • 6 minutes to read • [Edit Online](#)

Packet captures are a key component for implementing network intrusion detection systems (IDS) and performing Network Security Monitoring (NSM). There are several open source IDS tools that process packet captures and look for signatures of possible network intrusions and malicious activity. Using the packet captures provided by Network Watcher, you can analyze your network for any harmful intrusions or vulnerabilities.

One such open source tool is Suricata, an IDS engine that uses rulesets to monitor network traffic and triggers alerts whenever suspicious events occur. Suricata offers a multi-threaded engine, meaning it can perform network traffic analysis with increased speed and efficiency. For more details about Suricata and its capabilities, visit their website at <https://suricata-ids.org/>.

Scenario

This article explains how to set up your environment to perform network intrusion detection using Network Watcher, Suricata, and the Elastic Stack. Network Watcher provides you with the packet captures used to perform network intrusion detection. Suricata processes the packet captures and trigger alerts based on packets that match its given ruleset of threats. These alerts are stored in a log file on your local machine. Using the Elastic Stack, the logs generated by Suricata can be indexed and used to create a Kibana dashboard, providing you with a visual representation of the logs and a means to quickly gain insights to potential network vulnerabilities.



Both open source tools can be set up on an Azure VM, allowing you to perform this analysis within your own

Azure network environment.

Steps

Install Suricata

For all other methods of installation, visit <https://suricata.readthedocs.io/en/latest/install.html>

1. In the command-line terminal of your VM run the following commands:

```
sudo add-apt-repository ppa:oisf/suricata-stable  
sudo apt-get update  
sudo apt-get install suricata
```

2. To verify your installation, run the command `suricata -h` to see the full list of commands.

Download the Emerging Threats ruleset

At this stage, we do not have any rules for Suricata to run. You can create your own rules if there are specific threats to your network you would like to detect, or you can also use developed rule sets from a number of providers, such as Emerging Threats, or VRT rules from Snort. We use the freely accessible Emerging Threats ruleset here:

Download the rule set and copy them into the directory:

```
wget https://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz  
tar zxf emerging.rules.tar.gz  
sudo cp -r rules /etc/suricata/
```

Process packet captures with Suricata

To process packet captures using Suricata, run the following command:

```
sudo suricata -c /etc/suricata/suricata.yaml -r <location_of_pcapyfile>
```

To check the resulting alerts, read the fast.log file:

```
tail -f /var/log/suricata/fast.log
```

Set up the Elastic Stack

While the logs that Suricata produces contain valuable information about what's happening on our network, these log files aren't the easiest to read and understand. By connecting Suricata with the Elastic Stack, we can create a Kibana dashboard what allows us to search, graph, analyze, and derive insights from our logs.

Install Elasticsearch

1. The Elastic Stack from version 5.0 and above requires Java 8. Run the command `java -version` to check your version. If you do not have java installed, refer to documentation on the [Azure-supported JDKs](#).
2. Download the correct binary package for your system:

```
curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.2.0.deb  
sudo dpkg -i elasticsearch-5.2.0.deb  
sudo /etc/init.d/elasticsearch start
```

Other installation methods can be found at [Elasticsearch Installation](#)

- Verify that Elasticsearch is running with the command:

```
curl http://127.0.0.1:9200
```

You should see a response similar to this:

```
{
  "name" : "Angela Del Toro",
  "cluster_name" : "elasticsearch",
  "version" : {
    "number" : "5.2.0",
    "build_hash" : "8ff36d139e16f8720f2947ef62c8167a888992fe",
    "build_timestamp" : "2016-01-27T13:32:39Z",
    "build_snapshot" : false,
    "lucene_version" : "6.1.0"
  },
  "tagline" : "You Know, for Search"
}
```

For further instructions on installing Elastic search, refer to the page [Installation](#)

Install Logstash

- To install Logstash run the following commands:

```
curl -L -O https://artifacts.elastic.co/downloads/logstash/logstash-5.2.0.deb
sudo dpkg -i logstash-5.2.0.deb
```

- Next we need to configure Logstash to read from the output of eve.json file. Create a logstash.conf file using:

```
sudo touch /etc/logstash/conf.d/logstash.conf
```

- Add the following content to the file (make sure that the path to the eve.json file is correct):

```
input {
  file {
    path => ["/var/log/suricata/eve.json"]
    codec => "json"
    type => "SuricataIDPS"
  }
}

filter {
  if [type] == "SuricataIDPS" {
    date {
      match => [ "timestamp", "ISO8601" ]
    }
    ruby {
      code => "
        if event.get('[event_type]') == 'fileinfo'
          event.set('[fileinfo][type]', event.get('[fileinfo][magic]').to_s.split(',')[0])
        end
      "
    }
    ruby{
      code => "
        if event.get('[event_type]') == 'alert'
          sp = event.get('[alert][signature]').to_s.split(' group ')
          if (sp.length == 2) and /\A\d+\z/.match(sp[1])
            event.set('[alert][signature]', sp[1])
          end
        end
      "
    }
  }
}

output {
  file {
    path => "/var/log/suricata/logstash.log"
    type => "log"
  }
}
```

```

        event.set('[alert][signature]', sp[0])
    end
end
"
}
}

if [src_ip] {
    geoip {
        source => "src_ip"
        target => "geoip"
        #database => "/opt/logstash/vendor/geoip/GeoLiteCity.dat"
        add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}" ]
        add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}" ]
    }
    mutate {
        convert => [ "[geoip][coordinates]", "float" ]
    }
    if ![geoip.ip] {
        if [dest_ip] {
            geoip {
                source => "dest_ip"
                target => "geoip"
                #database => "/opt/logstash/vendor/geoip/GeoLiteCity.dat"
                add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}" ]
                add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}" ]
            }
            mutate {
                convert => [ "[geoip][coordinates]", "float" ]
            }
        }
    }
}

output {
    elasticsearch {
        hosts => "localhost"
    }
}

```

4. Make sure to give the correct permissions to the eve.json file so that Logstash can ingest the file.

```
sudo chmod 775 /var/log/suricata/eve.json
```

5. To start Logstash run the command:

```
sudo /etc/init.d/logstash start
```

For further instructions on installing Logstash, refer to the [official documentation](#)

Install Kibana

1. Run the following commands to install Kibana:

```
curl -L -O https://artifacts.elastic.co/downloads/kibana/kibana-5.2.0-linux-x86_64.tar.gz
tar xzvf kibana-5.2.0-linux-x86_64.tar.gz
```

2. To run Kibana use the commands:

```
cd kibana-5.2.0-linux-x86_64/
./bin/kibana
```

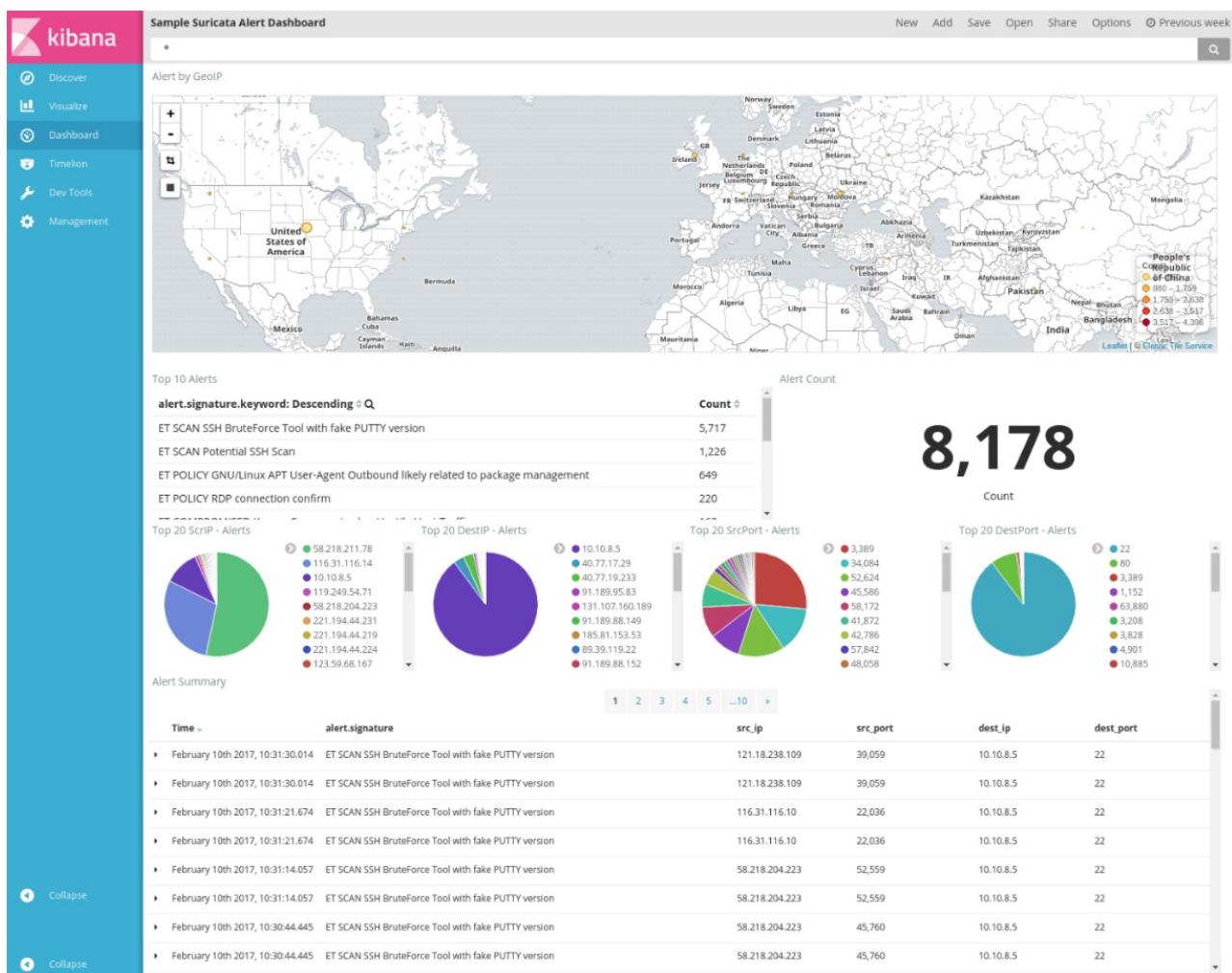
3. To view your Kibana web interface, navigate to <http://localhost:5601>
4. For this scenario, the index pattern used for the Suricata logs is "logstash-*"
5. If you want to view the Kibana dashboard remotely, create an inbound NSG rule allowing access to **port 5601**.

Create a Kibana dashboard

For this article, we have provided a sample dashboard for you to view trends and details in your alerts.

1. Download the dashboard file [here](#), the visualization file [here](#), and the saved search file [here](#).
2. Under the **Management** tab of Kibana, navigate to **Saved Objects** and import all three files. Then from the **Dashboard** tab you can open and load the sample dashboard.

You can also create your own visualizations and dashboards tailored towards metrics of your own interest. Read more about creating Kibana visualizations from Kibana's [official documentation](#).



Visualize IDS alert logs

The sample dashboard provides several visualizations of the Suricata alert logs:

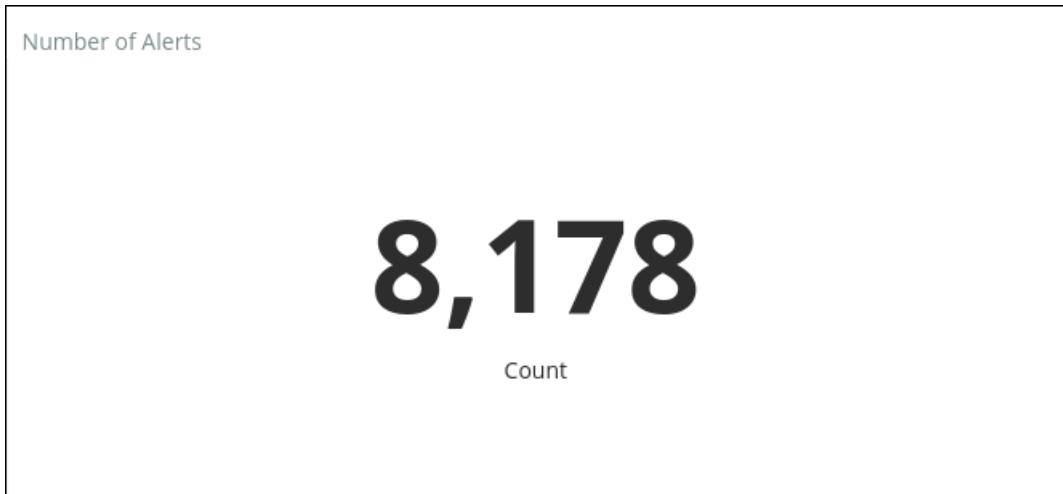
1. Alerts by GeolP – a map showing the distribution of alerts by their country/region of origin based on geographic location (determined by IP)



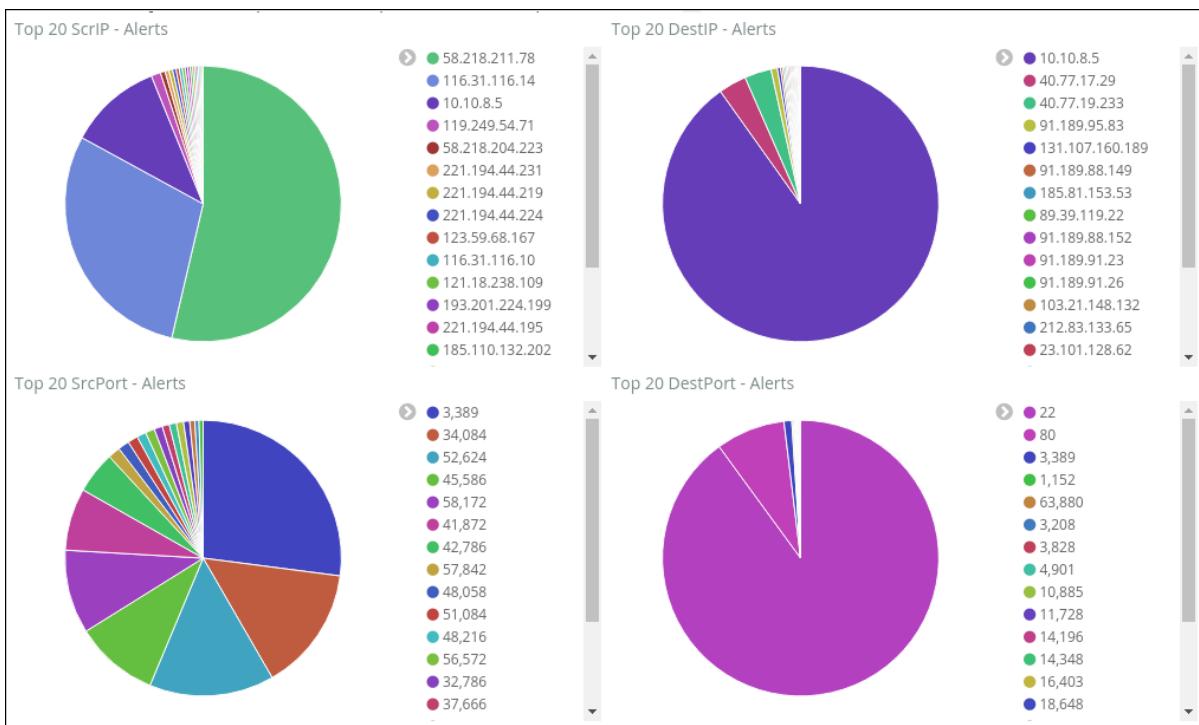
2. Top 10 Alerts – a summary of the 10 most frequent triggered alerts and their description. Clicking an individual alert filters down the dashboard to the information pertaining to that specific alert.

Top 10 Alerts	
	Count
ET SCAN SSH BruteForce Tool with fake PUTTY version	5,717
ET SCAN Potential SSH Scan	1,226
ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management	649
ET POLICY RDP connection confirm	220
ET COMPROMISED Known Compromised or Hostile Host Traffic	163
ET DROP Dshield Block Listed Source	122
ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DoS Attempt	43
FT CINS Active Threat Intelligence Poor Reputation IP	15

3. Number of Alerts – the total count of alerts triggered by the ruleset



4. Top 20 Source/Destination IPs/Ports - pie charts showing the top 20 IPs and ports that alerts were triggered on. You can filter down on specific IPs/ports to see how many and what kind of alerts are being triggered.



5. Alert Summary – a table summarizing specific details of each individual alert. You can customize this table to show other parameters of interest for each alert.

Alert Summary						
Time	alert.signature	src_ip	src_port	dest_ip	dest_port	
▶ February 10th 2017, 10:30:44.445	ET SCAN SSH BruteForce Tool with fake PUTTY version	58.218.204.223	45,760	10.10.8.5	22	
▶ February 10th 2017, 10:30:27.241	ET COMPROMISED Known Compromised or Hostile Host Traffic	193.201.224.199	12,091	10.10.8.5	22	
▶ February 10th 2017, 10:30:27.241	ET COMPROMISED Known Compromised or Hostile Host Traffic	193.201.224.199	12,091	10.10.8.5	22	
▶ February 10th 2017, 10:30:19.883	ET SCAN SSH BruteForce Tool with fake PUTTY version	116.31.116.10	56,602	10.10.8.5	22	
▶ February 10th 2017, 10:30:19.883	ET SCAN SSH BruteForce Tool with fake PUTTY version	116.31.116.10	56,602	10.10.8.5	22	
▶ February 10th 2017, 10:30:10.601	ET SCAN SSH BruteForce Tool with fake PUTTY version	58.218.204.223	23,775	10.10.8.5	22	
▶ February 10th 2017, 10:30:10.601	ET SCAN SSH BruteForce Tool with fake PUTTY version	58.218.204.223	23,775	10.10.8.5	22	
▶ February 10th 2017, 10:30:09.888	ET SCAN Potential SSH Scan	58.218.204.223	23,775	10.10.8.5	22	

For more documentation on creating custom visualizations and dashboards, see [Kibana's official documentation](#).

Conclusion

By combining packet captures provided by Network Watcher and open source IDS tools such as Suricata, you can perform network intrusion detection for a wide range of threats. These dashboards allow you to quickly spot trends and anomalies within your network, as well dig into the data to discover root causes of alerts such as malicious user agents or vulnerable ports. With this extracted data, you can make informed decisions on how to react to and protect your network from any harmful intrusion attempts, and create rules to prevent future intrusions to your network.

Next steps

Learn how to trigger packet captures based on alerts by visiting [Use packet capture to do proactive network monitoring with Azure Functions](#)

Learn how to visualize your NSG flow logs with Power BI by visiting [Visualize NSG flows logs with Power BI](#)

Visualize network traffic patterns to and from your VMs using open-source tools

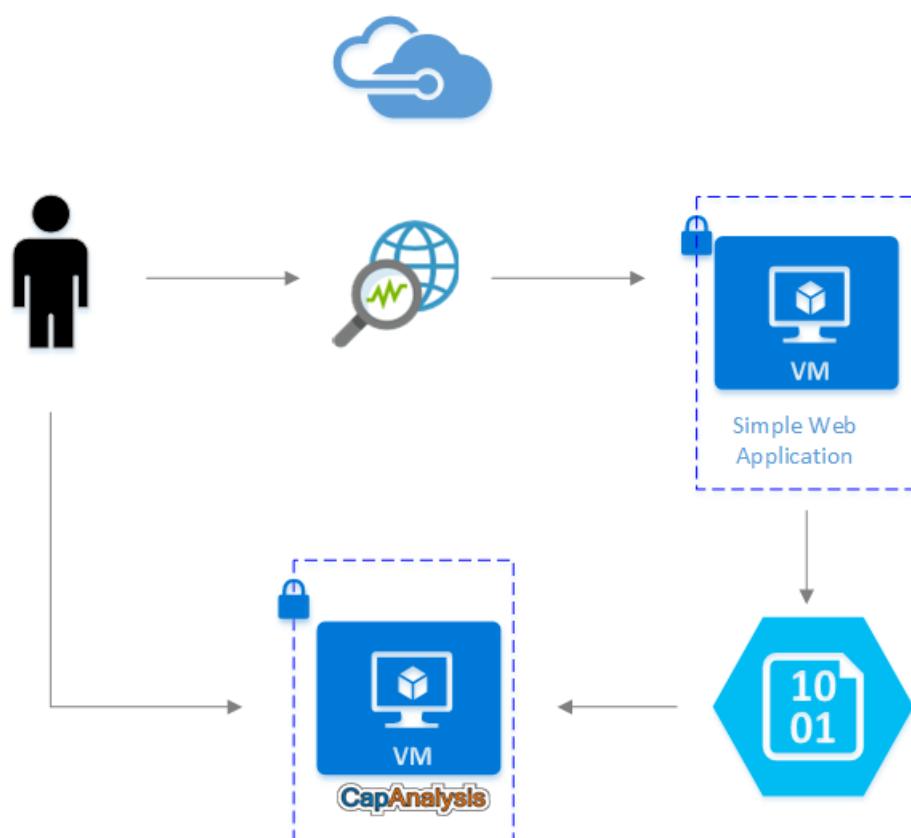
1/28/2020 • 3 minutes to read • [Edit Online](#)

Packet captures contain network data that allow you to perform network forensics and deep packet inspection. There are many open-source tools you can use to analyze packet captures to gain insights about your network. One such tool is CapAnalysis, an open-source packet capture visualization tool. Visualizing packet capture data is a valuable way to quickly derive insights on patterns and anomalies within your network. Visualizations also provide a means of sharing such insights in an easily consumable manner.

Azure's Network Watcher provides you the ability to capture data by allowing you to perform packet captures on your network. This article provides a walk-through of how to visualize and gain insights from packet captures using CapAnalysis with Network Watcher.

Scenario

You have a simple web application deployed on a VM in Azure and want to use open-source tools to visualize its network traffic to quickly identify flow patterns and any possible anomalies. With Network Watcher, you can obtain a packet capture of your network environment and directly store it on your storage account. CapAnalysis can then ingest the packet capture directly from the storage blob and visualize its contents.



Steps

Install CapAnalysis

To install CapAnalysis on a virtual machine, you can refer to the official instructions here <https://www.capanalysis.net/ca/how-to-install-capanalysis>. In order access CapAnalysis remotely, you need to open port 9877 on your VM by adding a new inbound security rule. For more about creating rules in Network Security Groups, refer to [Create rules in an existing NSG](#). Once the rule has been successfully added, you should be able to access CapAnalysis from `http://<PublicIP>:9877`

Use Azure Network Watcher to start a packet capture session

Network Watcher allows you to capture packets to track traffic in and out of a virtual machine. You can refer to the instructions at [Manage packet captures with Network Watcher](#) to start a packet capture session. A packet capture can be stored in a storage blob to be accessed by CapAnalysis.

Upload a packet capture to CapAnalysis

You can directly upload a packet capture taken by network watcher using the "Import from URL" tab and providing a link to the storage blob where the packet capture is stored.

When providing a link to CapAnalysis, make sure to append a SAS token to the storage blob URL. To do this, navigate to Shared access signature from the storage account, designate the allowed permissions, and press the Generate SAS button to create a token. You can then append the SAS token to the packet capture storage blob URL.

The resulting URL will look something like the following URL:

<http://storageaccount.blob.core.windows.net/container/location?addSASkeyhere>

Analyzing packet captures

CapAnalysis offers various options to visualize your packet capture, each providing analysis from a different perspective. With these visual summaries, you can understand your network traffic trends and quickly spot any unusual activity. A few of these features are shown in the following list:

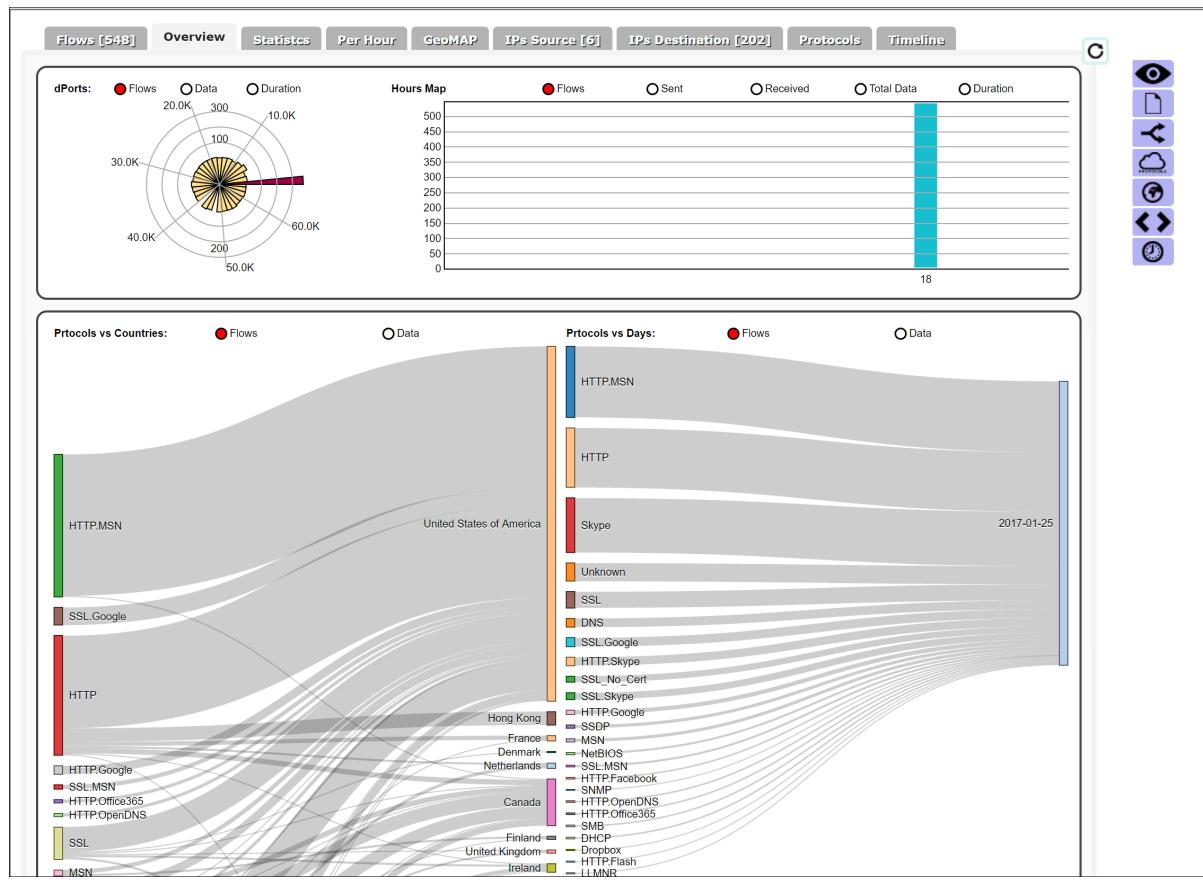
1. Flow Tables

This table gives you the list of flows in the packet data, the time stamp associated with the flows and the various protocols associated with the flow, as well as source and destination IP.

Flows [548]										Overview	Statistics	Per Hour	GeoMAP	IPs Source [6]	IPs Destination [202]	Protocols	Timeline
Date ↑	Time	Source IP	Destination IP	Destination Name		Source Port	Destination Port	L4	Prot								
1	2017-01-25	18:57:18	192.168.3.131	204.14.234.101		57231	8443	TCP	Unkn								
1	2017-01-25	18:57:18	192.168.3.131	192.168.3.1		60629	53	ESP	DNS								
1	2017-01-25	18:57:18	192.168.3.131	66.235.136.89	omtr2.partners.salesforce.com	57250	443	TCP	SSL								
1	2017-01-25	18:57:18	192.168.3.131	66.235.136.89	omtr2.partners.salesforce.com	57250	8443	TCP	SSL								
1	2017-01-25	18:57:18	192.168.3.131	192.168.3.1		58673	53	ESP	DNS								
1	2017-01-25	18:57:17	192.168.3.131	204.14.234.101		57231	443	TCP	SSL								
1	2017-01-25	18:57:15	192.168.3.131	204.14.234.85		57249	443	TCP	SSL								
1	2017-01-25	18:57:15	192.168.3.131	204.14.234.85		57249	8443	TCP	SSL								
1	2017-01-25	18:57:14	172.16.255.1	172.16.0.1		63805	5351	UDP	Unkn								
1	2017-01-25	18:57:14	172.16.255.1	239.255.255.250		63802	1900	UDP	SSDP								
1	2017-01-25	18:57:08	172.16.255.1	204.9.163.158		10714	80	TCP	HTTP.S								
1	2017-01-25	18:57:01	10.0.2.15	70.37.129.34	Img4.catalog.video.msn.com	2554	5480	TCP	HTTP.N								
1	2017-01-25	18:57:00	10.0.2.15	70.37.129.34	Img4.catalog.video.msn.com	2553	5480	TCP	HTTP.N								

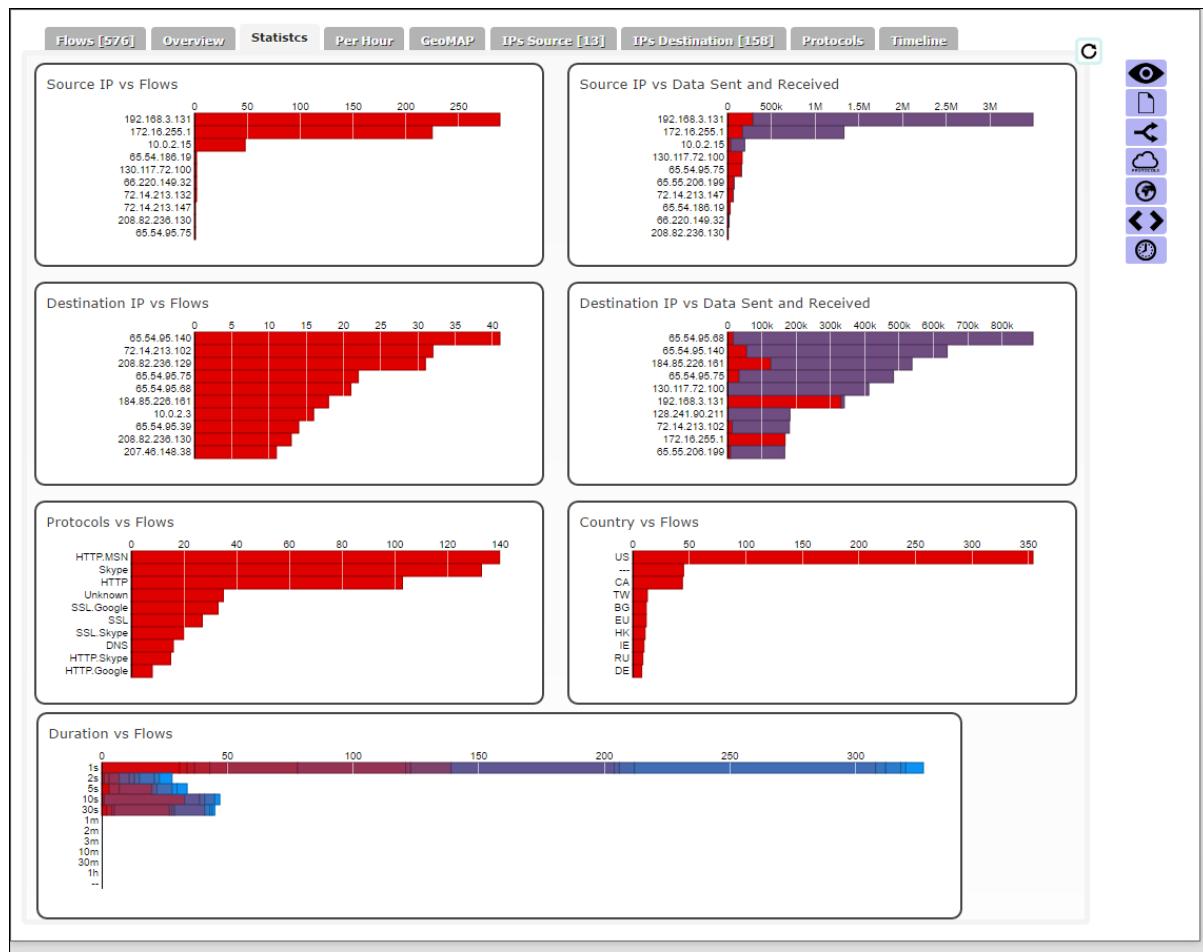
2. Protocol Overview

This pane allows you to quickly see the distribution of network traffic over the various protocols and geographies.



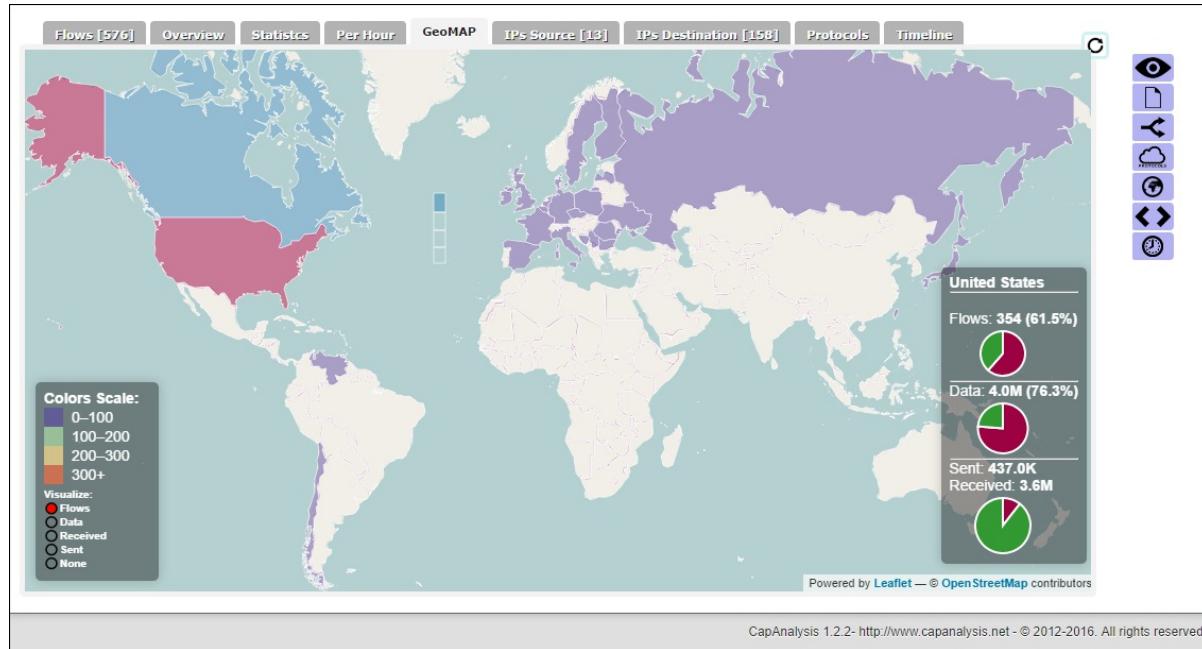
3. Statistics

This pane allows you to view network traffic statistics – bytes sent and received from source and destination IPs, flows for each of the source and destination IPs, protocol used for various flows, and the duration of flows.



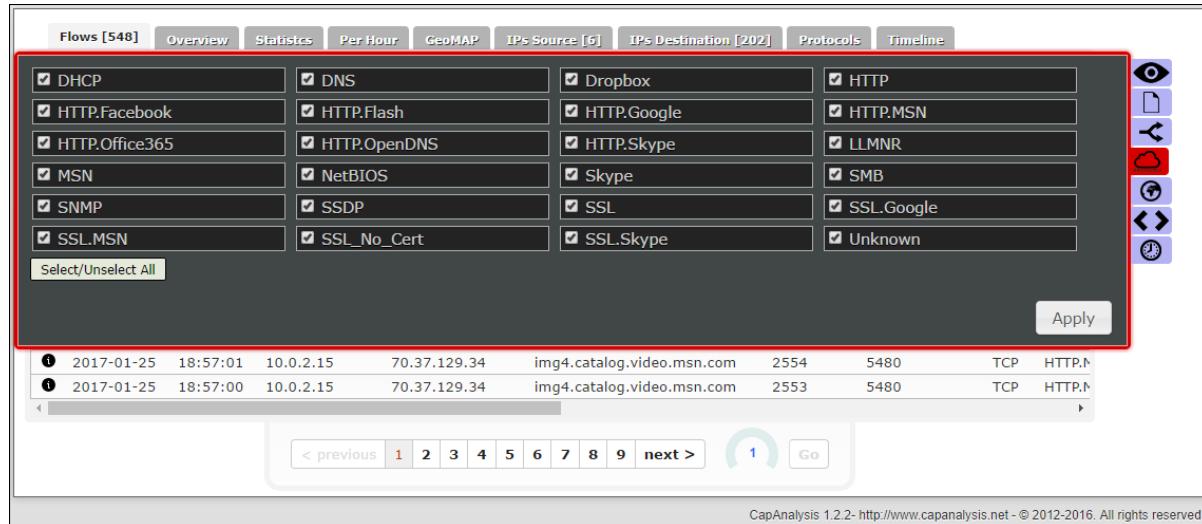
4. Geomap

This pane provides you with a map view of your network traffic, with colors scaling to the volume of traffic from each country/region. You can select highlighted countries/regions to view additional flow statistics such as the proportion of data sent and received from IPs in that country/region.



5. Filters

CapAnalysis provides a set of filters for quick analysis of specific packets. For example, you can choose to filter the data by protocol to gain specific insights on that subset of traffic.



Visit <https://www.capanalysis.net/ca/#about> to learn more about all CapAnalysis' capabilities.

Conclusion

Network Watcher's packet capture feature allows you to capture the data necessary to perform network forensics and better understand your network traffic. In this scenario, we showed how packet captures from Network Watcher can easily be integrated with open-source visualization tools. By using open-source tools such as CapAnalysis to visualize packets captures, you can perform deep packet inspection and quickly identify trends within your network traffic.

Next steps

To learn more about NSG flow logs, visit [NSG Flow logs](#)

Learn how to visualize your NSG flow logs with Power BI by visiting [Visualize NSG flows logs with Power BI](#)

Configuring Network Security Group Flow logs with PowerShell

1/28/2020 • 2 minutes to read • [Edit Online](#)

Network Security Group flow logs are a feature of Network Watcher that allows you to view information about ingress and egress IP traffic through a Network Security Group. These flow logs are written in json format and show outbound and inbound flows on a per rule basis, the NIC the flow applies to, 5-tuple information about the flow (Source/Destination IP, Source/Destination Port, Protocol), and if the traffic was allowed or denied.

Register Insights provider

In order for flow logging to work successfully, the **Microsoft.Insights** provider must be registered. If you are not sure if the **Microsoft.Insights** provider is registered, run the following script.

```
Register-AzResourceProvider -ProviderNamespace Microsoft.Insights
```

Enable Network Security Group Flow logs and Traffic Analytics

The command to enable flow logs is shown in the following example:

```
$NW = Get-AzNetworkWatcher -ResourceGroupName NetworkWatcherRg -Name NetworkWatcher_westcentralus
$nsg = Get-AzNetworkSecurityGroup -ResourceGroupName nsgRG -Name nsgName
$storageAccount = Get-AzStorageAccount -ResourceGroupName StorageRG -Name contosostorage123
Get-AzNetworkWatcherFlowLogStatus -NetworkWatcher $NW -TargetResourceId $nsg.Id

#Traffic Analytics Parameters
$workspaceResourceId = "/subscriptions/bbbbbbbb-bbbb-bbbb-bbbb-
bbbbbbbbbb/resourcegroups/trafficanalyticsrg/providers/microsoft.operationalinsights/workspaces/taworkspace"
$workspaceGUID = "cccccccc-cccc-cccc-cccc-cccccccccc"
$workspaceLocation = "westeurope"

#Configure Version 1 Flow Logs
Set-AzNetworkWatcherConfigFlowLog -NetworkWatcher $NW -TargetResourceId $nsg.Id -StorageAccountId
$storageAccount.Id -EnableFlowLog $true -FormatType Json -FormatVersion 1

#Configure Version 2 Flow Logs, and configure Traffic Analytics
Set-AzNetworkWatcherConfigFlowLog -NetworkWatcher $NW -TargetResourceId $nsg.Id -StorageAccountId
$storageAccount.Id -EnableFlowLog $true -FormatType Json -FormatVersion 2

#Configure Version 2 Flow Logs with Traffic Analytics Configured
Set-AzNetworkWatcherConfigFlowLog -NetworkWatcher $NW -TargetResourceId $nsg.Id -StorageAccountId
$storageAccount.Id -EnableFlowLog $true -FormatType Json -FormatVersion 2 -EnableTrafficAnalytics -
WorkspaceResourceId $workspaceResourceId -WorkspaceGUID $workspaceGUID -WorkspaceLocation $workspaceLocation

#Query Flow Log Status
Get-AzNetworkWatcherFlowLogStatus -NetworkWatcher $NW -TargetResourceId $nsg.Id
```

The storage account you specify cannot have network rules configured for it that restrict network access to only Microsoft services or specific virtual networks. The storage account can be in the same, or a different Azure subscription, than the NSG that you enable the flow log for. If you use different subscriptions, they must both be associated to the same Azure Active Directory tenant. The account you use for each subscription must have the [necessary permissions](#).

Disable Traffic Analytics and Network Security Group Flow logs

Use the following example to disable traffic analytics and flow logs:

```
#Disable Traffic Analytics by removing -EnableTrafficAnalytics property  
Set-AzNetworkWatcherConfigFlowLog -NetworkWatcher $NW -TargetResourceId $nsg.Id -StorageAccountId  
$storageAccount.Id -EnableFlowLog $true -FormatType Json -FormatVersion 2 -WorkspaceResourceId  
$workspaceResourceId -WorkspaceGUID $workspaceGUID -WorkspaceLocation $workspaceLocation  
  
#Disable Flow Logging  
Set-AzNetworkWatcherConfigFlowLog -NetworkWatcher $NW -TargetResourceId $nsg.Id -StorageAccountId  
$storageAccount.Id -EnableFlowLog $false
```

Download a Flow log

The storage location of a flow log is defined at creation. A convenient tool to access these flow logs saved to a storage account is Microsoft Azure Storage Explorer, which can be downloaded here: <https://storageexplorer.com/>

If a storage account is specified, flow log files are saved to a storage account at the following location:

```
https://{{storageAccountName}}.blob.core.windows.net/insights-logs-  
networksecuritygroupflowevent/resourceId=/SUBSCRIPTIONS/{{subscriptionID}}/RESOURCEGROUPS/{{resourceGroupName}}/PR  
OVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/{{nsgName}}/y={{year}}/m={{month}}/d={{day}}/h={{hour}}/m=00/macAddress=  
{macAddress}/PT1H.json
```

For information about the structure of the log visit [Network Security Group Flow log Overview](#)

Next Steps

Learn how to [Visualize your NSG flow logs with PowerBI](#)

Learn how to [Visualize your NSG flow logs with open source tools](#)

Configuring Network Security Group Flow logs with Azure CLI

1/28/2020 • 2 minutes to read • [Edit Online](#)

Network Security Group flow logs are a feature of Network Watcher that allows you to view information about ingress and egress IP traffic through a Network Security Group. These flow logs are written in json format and show outbound and inbound flows on a per rule basis, the NIC the flow applies to, 5-tuple information about the flow (Source/Destination IP, Source/Destination Port, Protocol), and if the traffic was allowed or denied.

To perform the steps in this article, you need to [install the Azure command-line interface for Mac, Linux, and Windows \(CLI\)](#).

Register Insights provider

In order for flow logging to work successfully, the **Microsoft.Insights** provider must be registered. If you are not sure if the **Microsoft.Insights** provider is registered, run the following script.

```
az provider register --namespace Microsoft.Insights
```

Enable Network Security Group Flow logs

The command to enable flow logs is shown in the following example:

```
az network watcher flow-log configure --resource-group resourceGroupName --enabled true --nsg nsgName --storage-account storageAccountName
# Configure
az network watcher flow-log configure --resource-group resourceGroupName --enabled true --nsg nsgName --storage-account storageAccountName --format JSON --log-version 2
```

The storage account that you specify cannot have network rules configured for it that restrict network access to only Microsoft services or specific virtual networks. The storage account can be in the same, or a different Azure subscription, than the NSG that you enable the flow log for. If you use different subscriptions, they must both be associated to the same Azure Active Directory tenant. The account you use for each subscription must have the [necessary permissions](#).

If the storage account is in a different resource group, or subscription, than the network security group, specify the full ID of the storage account, rather than its name. For example, if the storage account is in a resource group named *RG-Storage*, rather than specifying *storageAccountName* in the previous command, you'd specify */subscriptions/{SubscriptionID}/resourceGroups/RG-Storage/providers/Microsoft.Storage/storageAccounts/storageAccountName*.

Disable Network Security Group Flow logs

Use the following example to disable flow logs:

```
az network watcher flow-log configure --resource-group resourceGroupName --enabled false --nsg nsgName
```

Download a Flow log

The storage location of a flow log is defined at creation. A convenient tool to access these flow logs saved to a storage account is Microsoft Azure Storage Explorer, which can be downloaded here: <https://storageexplorer.com/>

If a storage account is specified, flow log files are saved to a storage account at the following location:

```
https://{{storageAccountName}}.blob.core.windows.net/insights-logs-
networksecuritygroupflowevent/resourceId={{subscriptionID}}/RESOURCEGROUPS/{{resourceGroupName}}/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/{{nsgName}}/y={{year}}/m={{month}}/d={{day}}/h={{hour}}/m=00/macAddress=
{{macAddress}}/PT1H.json
```

Next Steps

Learn how to [Visualize your NSG flow logs with PowerBI](#)

Learn how to [Visualize your NSG flow logs with open source tools](#)

Configuring Network Security Group flow logs using REST API

1/28/2020 • 3 minutes to read • [Edit Online](#)

Network Security Group flow logs are a feature of Network Watcher that allows you to view information about ingress and egress IP traffic through a Network Security Group. These flow logs are written in json format and show outbound and inbound flows on a per rule basis, the NIC the flow applies to, 5-tuple information about the flow (Source/Destination IP, Source/Destination Port, Protocol), and if the traffic was allowed or denied.

Before you begin

ARMclient is used to call the REST API using PowerShell. ARMClient is found on chocolatey at [ARMClient on Chocolatey](#)

This scenario assumes you have already followed the steps in [Create a Network Watcher](#) to create a Network Watcher.

IMPORTANT

For Network Watcher REST API calls the resource group name in the request URI is the resource group that contains the Network Watcher, not the resources you are performing the diagnostic actions on.

Scenario

The scenario covered in this article shows you how to enable, disable, and query flow logs using the REST API. To learn more about Network Security Group flow loggings, visit [Network Security Group flow logging - Overview](#).

In this scenario, you will:

- Enable flow logs (Version 2)
- Disable flow logs
- Query flow logs status

Log in with ARMClient

Log in to armclient with your Azure credentials.

```
armclient login
```

Register Insights provider

In order for flow logging to work successfully, the **Microsoft.Insights** provider must be registered. If you are not sure if the **Microsoft.Insights** provider is registered, run the following script.

```
$subscriptionId = "00000000-0000-0000-0000-000000000000"  
armclient post  
"https://management.azure.com//subscriptions/${subscriptionId}/providers/Microsoft.Insights/register?api-version=2016-09-01"
```

Enable Network Security Group flow logs

The command to enable flow logs version 2 is shown in the following example. For version 1 replace the 'version' field with '1':

```
$subscriptionId = "00000000-0000-0000-0000-000000000000"
$targetUri = "" # example /subscriptions/00000000-0000-0000-0000-
00000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkSecurityGroups/{nsgName}"
$storageId = "/subscriptions/00000000-0000-0000-0000-
00000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/{saName}"
$resourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
$requestBody = @@
{
    'targetResourceId': '${targetUri}',
    'properties': {
        'storageId': '${storageId}',
        'enabled': 'true',
        'retentionPolicy' : {
            days: 5,
            enabled: true
        },
        'format': {
            'type': 'JSON',
            'version': 2
        }
    }
}
"@

armclient post
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Network/networkWatchers/${networkWatcherName}/configureFlowLog?api-version=2016-12-01" $requestBody
```

The response returned from the preceding example is as follows:

```
{
    "targetResourceId": "/subscriptions/00000000-0000-0000-0000-
00000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkSecurityGroups/{nsgName}",
    "properties": {
        "storageId": "/subscriptions/00000000-0000-0000-0000-
00000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/{saName}",
        "enabled": true,
        "retentionPolicy": {
            "days": 5,
            "enabled": true
        },
        "format": {
            "type": "JSON",
            "version": 2
        }
    }
}
```

Disable Network Security Group flow logs

Use the following example to disable flow logs. The call is the same as enabling flow logs, except **false** is set for the **enabled** property.

```

$subscriptionId = "00000000-0000-0000-0000-000000000000"
$targetUri = "" # example /subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkSecurityGroups/{nsgName}"
$storageId = "/subscriptions/00000000-0000-0000-0000-
000000000000/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/{saName}"
$resourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
$requestBody = @"
{
    'targetResourceId': '${targetUri}',
    'properties': {
        'storageId': '${storageId}',
        'enabled': 'false',
        'retentionPolicy' : {
            days: 5,
            enabled: true
        },
        'format': {
            'type': 'JSON',
            'version': 2
        }
    }
}
"@

armclient post
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Network/networkWatchers/${networkWatcherName}/configureFlowLog?api-version=2016-12-01" $requestBody

```

The response returned from the preceding example is as follows:

```
{
    "targetResourceId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkSecurityGroups/{nsgName}",
    "properties": {
        "storageId": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/{saName}",
        "enabled": false,
        "retentionPolicy": {
            "days": 5,
            "enabled": true
        },
        "format": {
            "type": "JSON",
            "version": 2
        }
    }
}
```

Query flow logs

The following REST call queries the status of flow logs on a Network Security Group.

```

$subscriptionId = "00000000-0000-0000-0000-000000000000"
$targetUri = "" # example /subscriptions/00000000-0000-0000-0000-
00000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkSecurityGroups/{nsgName}"
$resourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
$requestBody = @@
{
    'targetResourceId': '${targetUri}',
}
"@

armclient post
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Network/networkWatchers/${networkWatcherName}/queryFlowLogStatus?api-version=2016-12-01"
$requestBody

```

The following is an example of the response returned:

```
{
    "targetResourceId": "/subscriptions/00000000-0000-0000-0000-
00000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkSecurityGroups/{nsgName}",
    "properties": {
        "storageId": "/subscriptions/00000000-0000-0000-0000-
00000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/{saName}",
        "enabled": true,
        "retentionPolicy": {
            "days": 5,
            "enabled": true
        },
        "format": {
            "type": "JSON",
            "version": 2
        }
    }
}
```

Download a flow log

The storage location of a flow log is defined at creation. A convenient tool to access these flow logs saved to a storage account is Microsoft Azure Storage Explorer, which can be downloaded here: <https://storageexplorer.com/>

If a storage account is specified, packet capture files are saved to a storage account at the following location:

```

https://'{storageAccountName}.blob.core.windows.net/insights-logs-
networksecuritygroupflowevent/resourceId=/SUBSCRIPTIONS/{subscriptionID}/RESOURCEGROUPS/{resourceGroupName}/P
ROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/{nsgName}/y={year}/m={month}/d={day}/h=
{hour}/m=00/macAddress={macAddress}/PT1H.json

```

Next steps

Learn how to [Visualize your NSG flow logs with PowerBI](#)

Learn how to [Visualize your NSG flow logs with open source tools](#)

Configure NSG Flow Logs from an Azure Resource Manager template

2/21/2020 • 2 minutes to read • [Edit Online](#)

Azure Resource Manager is Azure's native and powerful way to manage your [infrastructure as code](#).

This article shows how you to enable [NSG Flow Logs](#) programmatically using an Azure Resource Manager template and Azure PowerShell. We start by providing an overview of the properties of the NSG Flow Log object, followed by a few sample templates. Then we the deploy template using a local PowerShell instance.

NSG Flow Logs object

The NSG Flow Logs object with all parameters is shown below. For a complete overview of the properties, you may read the [NSG Flow Logs template reference](#).

```
{  
  "name": "string",  
  "type": "Microsoft.Network/networkWatchers/flowLogs",  
  "location": "string",  
  "apiVersion": "2019-09-01",  
  "properties": {  
    "targetResourceId": "string",  
    "storageId": "string",  
    "enabled": "boolean",  
    "flowAnalyticsConfiguration": {  
      "networkWatcherFlowAnalyticsConfiguration": {  
        "enabled": "boolean",  
        "workspaceResourceId": "string",  
        "trafficAnalyticsInterval": "integer"  
      },  
      "retentionPolicy": {  
        "days": "integer",  
        "enabled": "boolean"  
      },  
      "format": {  
        "type": "string",  
        "version": "integer"  
      }  
    }  
  }  
}
```

To create a Microsoft.Network/networkWatchers/flowLogs resource, add the above JSON to the resources section of your template.

Creating your template

If you are using Azure Resource Manager templates for the first time, you can learn more about them using the links below.

- [Deploy resources with Resource Manager templates and Azure PowerShell](#)
- [Tutorial: Create and deploy your first Azure Resource Manager template](#)

Below are two examples of complete templates to set up NSG Flow Logs.

Example 1: The simplest version of the above with minimum parameters passed. The below template enables NSG Flow Logs on a target NSG and stores them in a given storage account.

```
{  
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",  
  "contentVersion": "1.0.0.0",  
  "apiProfile": "2019-09-01",  
  "resources": [  
    {  
      "name": "NetworkWatcher_centraluseuap/Microsoft.Network/DalanDemoPerimeterNSG",  
      "type": "Microsoft.Network/networkWatchers/FlowLogs/",  
      "location": "centraluseuap",  
      "apiVersion": "2019-09-01",  
      "properties": {  
        "targetResourceId": "/subscriptions/56abfb6-ec72-4ce9-831f-  
bc2b6f2c5505/resourceGroups/DalanDemo/providers/Microsoft.Network/networkSecurityGroups/PerimeterNSG",  
        "storageId": "/subscriptions/56abfb6-ec72-4ce9-831f-  
bc2b6f2c5505/resourceGroups/MyCanaryFlowLog/providers/Microsoft.Storage/storageAccounts/storagev2ira",  
        "enabled": true,  
        "flowAnalyticsConfiguration": {},  
        "retentionPolicy": {},  
        "format": {}  
      }  
    }  
  ]  
}
```

NOTE

- The name of resource has the format "Parent Resource>/Child resource". Here, the parent resource is the regional Network Watcher instance (Format: NetworkWatcher_. Example: NetworkWatcher_centraluseuap)
- targetResourceId is the resource ID of the target NSG
- storageId is the resource ID of the destination storage account

Example 2: The following templates enabling NSG Flow Logs (version 2) with a retention for 5 days. Enabling Traffic Analytics with a processing interval of 10 minutes.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "apiProfile": "2019-09-01",
  "resources": [
    {
      "name": "NetworkWatcher_centraluseuap/Microsoft.Network/DalanDemoPerimeterNSG",
      "type": "Microsoft.Network/networkWatchers/FlowLogs/",
      "location": "centraluseuap",
      "apiVersion": "2019-09-01",
      "properties": {
        "targetResourceId": "/subscriptions/56abfb6-ec72-4ce9-831f-
bc2b6f2c5505/resourceGroups/DalanDemo/providers/Microsoft.Network/networkSecurityGroups/PerimeterNSG",
        "storageId": "/subscriptions/56abfb6-ec72-4ce9-831f-
bc2b6f2c5505/resourceGroups/MyCanaryFlowLog/providers/Microsoft.Storage/storageAccounts/storagev2ira",
        "enabled": true,
        "flowAnalyticsConfiguration": {
          "networkWatcherFlowAnalyticsConfiguration": {
            "enabled": true,
            "workspaceResourceId": "/subscriptions/56abfb6-ec72-4ce9-831f-
bc2b6f2c5505/resourceGroups/defaultresourcegroup-
wcus/providers/Microsoft.OperationalInsights/workspaces/1c4f42e5-3a02-4146-ac9b-3051d8501db0",
            "trafficAnalyticsInterval": 10
          }
        },
        "retentionPolicy": {
          "days": 5,
          "enabled": true
        },
        "format": {
          "type": "JSON",
          "version": 2
        }
      }
    }
  ]
}
```

Deploying your Azure Resource Manager template

This tutorial assumes you have an existing Resource group and an NSG you can enable Flow logging on. You can save any of the above example templates locally as `azuredeploy.json`. Update the property values so that they point to valid resources in your subscription.

To deploy the template, run the following command in PowerShell.

```
New-AzResourceGroupDeployment -Name EnableFlowLog -ResourceGroupName NetworkWatcherRG ` 
-TemplateFile "C:\MyTemplates\azuredeploy.json"
```

Verifying your deployment

There are a couple of ways to check if your deployment has Succeeded. Your PowerShell console should show "ProvisioningState" as "Succeeded". Additionally, you can visit the [NSG Flow Logs portal page](#) to confirm your changes. If there were issues with the deployment, take a look at [Troubleshoot common Azure deployment errors with Azure Resource Manager](#).

Next steps

Learn how to visualize your NSG Flow data using:

- Microsoft Power BI
- Open source tools
- Azure Traffic Analytics

Delete network security group flow log storage blobs in Network Watcher

2/24/2020 • 2 minutes to read • [Edit Online](#)

Currently, there's an issue where [network security group \(NSG\) flow logs](#) for Network Watcher are not automatically deleted from Blob storage based on retention policy settings. You must now run a PowerShell script to manually delete the flow logs from your storage account as described in this article.

Run PowerShell script to delete NSG flow logs

Copy and save the following script to a location such as your current working directory.

```
# This powershell script deletes all NSG flow log blobs that should not be retained anymore as per configured
# retention policy.
# While configuring NSG flow logs on Azure portal, the user configures the retention period of NSG flow log
# blobs in
# their storage account (in days).
# This script reads all blobs and deletes blobs that are not to be retained (outside retention window)
# if the retention days are zero; all blobs are retained forever and hence no blobs are deleted.
#
#
param (
    [string] [Parameter(Mandatory=$true)] $SubscriptionId,
    [string] [Parameter(Mandatory=$true)] $Location,
    [switch] [Parameter(Mandatory=$false)] $Confirm
)

Login-AzAccount

$SubId = Get-AzSubscription | Where-Object {$_.Id.contains($SubscriptionId.ToLower())}

if ($SubId.Count -eq 0)
{
    Write-Error 'The SubscriptionId does not exist' -ErrorAction Stop
}

Set-AzContext -SubscriptionId $SubscriptionId

$NsgList = Get-AzNetworkSecurityGroup | Where-Object {$_ .Location -eq $Location}
$NW = Get-AzNetworkWatcher | Where-Object {$_ .Location -eq $Location}

$FlowLogsList = @()
foreach ($Nsg in $NsgList)
{
    # Query Flow Log Status which are enabled
    $NsgFlowLog = Get-AzNetworkWatcherFlowLogStatus -NetworkWatcher $NW -TargetResourceId $Nsg.Id | Where-
Object {$_ .Enabled -eq "True"}
    if ($NsgFlowLog.Count -gt 0)
    {
        $FlowLogsList += $NsgFlowLog
        Write-Output ('Enabled NSG found: ' + $NsgFlowLog.TargetResourceId)
    }
}

foreach ($Psflowlog in $FlowLogsList)
{
    $RetentionDays = $Psflowlog.RetentionPolicy.Days
    if ($RetentionDays -le 0)
```

```

{
    continue
}

$Strings = $Psflowlog.StorageId -split '/'
$RGName = $Strings[4]
$StorageAccountName = $Strings[-1]

$key = (Get-AzStorageAccountKey -ResourceGroupName $RGName -Name $StorageAccountName).Value[1]
$StorageAccount = New-AzStorageContext -StorageAccountName $StorageAccountName -StorageAccountKey $key

$ContainerName = 'insights-logs-networksecuritygroupflowevent'
$BLobsList = Get-AzStorageBlob -Container $ContainerName -Context $StorageAccount.Context

$TargetBLobsList = $BLobsList | Where-Object {$_ .Name.contains($Psflowlog.TargetResourceId.ToUpper())}

$RetentionDate = Get-Date
$RetentionDate = $RetentionDate.AddDays(-1*$RetentionDays)
$RetentionDateInUTC = $RetentionDate.ToUniversalTime()

foreach ($blob in $TargetBLobsList)
{
    $blobLastModifietedDTinUTC = [datetime]$blob.LastModified.UtcDateTime

    if ($blobLastModifietedDTinUTC -ge $RetentionDateInUTC)
    {
        Write-Output ($blob.Name + ' ==> ' + $blobLastModifietedDTinUTC + ' ==> RETAINED')
        continue
    }

    if ($Confirm)
    {
        Write-Output (blob to be deleted: $blob.Name)
        $confirmation = Read-Host "Are you sure you want to remove this blob (Y/N)?"
    }

    if ((-not $Confirm) -or ($confirmation -eq 'Y'))
    {
        Write-Output ($blob.Name + ' ==> ' + $blobLastModifietedDTinUTC + ' ==> DELETED')
        Remove-AzStorageBlob -Container $ContainerName -Context $StorageAccount.Context -Blob $blob.Name
    }
    else
    {
        Write-Output ($blob.Name + ' ==> ' + $blobLastModifietedDTinUTC + ' ==> RETAINED')
    }
}
}

Write-Output ('Retention policy for all NSGs evaluated and completed successfully')

```

1. Enter the following parameters in the script as needed:

- **SubscriptionId** [Mandatory]: The subscription ID from where you would like to delete NSG Flow Log blobs.
- **Location** [Mandatory]: The *location string* of the region of the NSGs for which you would like to delete NSG Flow Log blobs. You can view this information on the Azure portal or on [GitHub](#).
- **Confirm** [Optional]: Pass the confirm flag if you want to manually confirm the deletion of each storage blob.

2. Run the saved script as shown in the following example, where the script file was saved as **Delete-NsgFlowLogsBlobs.ps1**:

```
. \Delete-NsgFlowLogsBlobs.ps1 -SubscriptionId <subscriptionId> -Location <location> -Confirm
```

Next steps

- Customers can automate running the script by using [Azure Logic Apps](#) or [Azure Automation](#)
- To learn more about NSG logging, see [Azure Monitor logs for network security groups \(NSGs\)](#).

Read NSG flow logs

2/26/2020 • 5 minutes to read • [Edit Online](#)

Learn how to read NSG flow logs entries with PowerShell.

NSG flow logs are stored in a storage account in [block blobs](#). Block blobs are made up of smaller blocks. Each log is a separate block blob that is generated every hour. New logs are generated every hour, the logs are updated with new entries every few minutes with the latest data. In this article you learn how to read portions of the flow logs.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Scenario

In the following scenario, you have an example flow log that is stored in a storage account. You learn how to selectively read the latest events in NSG flow logs. In this article you use PowerShell, however, the concepts discussed in the article are not limited to the programming language, and are applicable to all languages supported by the Azure Storage APIs.

Setup

Before you begin, you must have Network Security Group Flow Logging enabled on one or many Network Security Groups in your account. For instructions on enabling Network Security flow logs, refer to the following article: [Introduction to flow logging for Network Security Groups](#).

Retrieve the block list

The following PowerShell sets up the variables needed to query the NSG flow log blob and list the blocks within the [CloudBlockBlob](#) block blob. Update the script to contain valid values for your environment.

```

function Get-NSGFlowLogCloudBlockBlob {
    [CmdletBinding()]
    param (
        [string] [Parameter(Mandatory=$true)] $subscriptionId,
        [string] [Parameter(Mandatory=$true)] $NSGResourceGroupName,
        [string] [Parameter(Mandatory=$true)] $NSGName,
        [string] [Parameter(Mandatory=$true)] $storageAccountName,
        [string] [Parameter(Mandatory=$true)] $storageAccountResourceGroup,
        [string] [Parameter(Mandatory=$true)] $macAddress,
        [datetime] [Parameter(Mandatory=$true)] $logTime
    )
}

process {
    # Retrieve the primary storage account key to access the NSG logs
    $StorageAccountKey = (Get-AzStorageAccountKey -ResourceGroupName $storageAccountResourceGroup -Name
    $storageAccountName).Value[0]

    # Setup a new storage context to be used to query the logs
    $ctx = New-AzStorageContext -StorageAccountName $StorageAccountName -StorageAccountKey
    $StorageAccountKey

    # Container name used by NSG flow logs
    $ContainerName = "insights-logs-networksecuritygroupflowevent"

    # Name of the blob that contains the NSG flow log
    $BlobName =
    "resourceId=/SUBSCRIPTIONS/${subscriptionId}/RESOURCEGROUPS/${NSGResourceGroupName}/PROVIDERS/MICROSOFT.NETWOR
    K/NETWORKSECURITYGROUPS/${NSGName}/y=$($logTime.Year)/m=$(($logTime).ToString("MM"))/d=$(($logTime).ToString("
    dd"))/h=$(($logTime).ToString("HH"))/m=00/macAddress=$($macAddress)/PT1H.json"

    # Gets the storage blob
    $Blob = Get-AzStorageBlob -Context $ctx -Container $ContainerName -Blob $BlobName

    # Gets the block blob of type 'Microsoft.Azure.Storage.Blob.CloudBlockBlob' from the storage blob
    $CloudBlockBlob = [Microsoft.Azure.Storage.Blob.CloudBlockBlob] $Blob.ICloudBlob

    #Return the Cloud Block Blob
    $CloudBlockBlob
}
}

function Get-NSGFlowLogBlockList {
    [CmdletBinding()]
    param (
        [Microsoft.Azure.Storage.Blob.CloudBlockBlob] [Parameter(Mandatory=$true)] $CloudBlockBlob
    )
}

process {
    # Stores the block list in a variable from the block blob.
    $blockList = $CloudBlockBlob.DownloadBlockListAsync()

    # Return the Block List
    $blockList
}
}

$CloudBlockBlob = Get-NSGFlowLogCloudBlockBlob -subscriptionId "yourSubscriptionId" -NSGResourceGroupName
"FLOWLOGSVALIDATIONWESTCENTRALUS" -NSGName "V2VALIDATIONVM-NSG" -storageAccountName "yourStorageAccountName" -
storageAccountResourceGroup "ml-rg" -macAddress "000D3AF87856" -logTime "11/11/2018 03:00"

$blockList = Get-NSGFlowLogBlockList -CloudBlockBlob $CloudBlockBlob

```

The `$blockList` variable returns a list of the blocks in the blob. Each block blob contains at least two blocks. The first block has a length of `12` bytes, this block contains the opening brackets of the json log. The other block is the closing brackets and has a length of `2` bytes. As you can see the following example log has seven entries in it,

each being an individual entry. All new entries in the log are added to the end right before the final block.

Name	Length	Committed
ZDk5MTk5N2FkNGE0MmY5MTk5ZWViYjA0YmZhODRhYzY=	12	True
NzQxNDA5MTRhNDUzMGI2M2Y1MDMyOWZ1N2QwNDZiYzQ=	2685	True
ODdjM2UyMWY3NzFhZTU3MmV1MmU5MDN10WEwNWE3YWY=	2586	True
ZDU2MjA3OGQ2ZDU3MjczMWQ4MTRmYWhhYjAzOGJkMTg=	2688	True
ZmM3ZWJjMGQ0ZDA1ODJ10WMyODh10WE3MDI1MGJhMTc=	2775	True
ZGVkYTc4MzQzNjEyMzlmZWE5MmRiNjc10WE50Tc0OTQ=	2676	True
ZmY2MjUzYTIwYWIyOGU1OTA2ZDY10WYzNmY2NmU4ZTY=	2777	True
Mzk1YzQwM2U0ZwY1ZDRh0WF1MTNhYjQ3OGVhYmUzNjk=	2675	True
ZjAyZTliYWE3OTI1YWZmYjFmMWI0MjJhNzMxZTI4MDM=	2	True

Read the block blob

Next you need to read the `$blocklist` variable to retrieve the data. In this example we iterate through the blocklist, read the bytes from each block and store them in an array. Use the [DownloadRangeToByteArray](#) method to retrieve the data.

```
function Get-NSGFlowLogReadBlock {
    [CmdletBinding()]
    param (
        [System.Array] [Parameter(Mandatory=$true)] $blockList,
        [Microsoft.Azure.Storage.Blob.CloudBlockBlob] [Parameter(Mandatory=$true)] $CloudBlockBlob
    )
    # Set the size of the byte array to the largest block
    $maxvalue = ($blocklist | measure Length -Maximum).Maximum

    # Create an array to store values in
    $valuearray = @()

    # Define the starting index to track the current block being read
    $index = 0

    # Loop through each block in the block list
    for($i=0; $i -lt $blocklist.count; $i++)
    {
        # Create a byte array object to store the bytes from the block
        $downloadArray = New-Object -TypeName byte[] -ArgumentList $maxvalue

        # Download the data into the ByteArray, starting with the current index, for the number of bytes in
        # the current block. Index is increased by 3 when reading to remove preceding comma.
        $CloudBlockBlob.DownloadRangeToByteArray($downloadArray,0,$index, $($blockList[$i].Length)) | Out-Null

        # Increment the index by adding the current block length to the previous index
        $index = $index + $blockList[$i].Length

        # Retrieve the string from the byte array

        $value = [System.Text.Encoding]::ASCII.GetString($downloadArray)

        # Add the log entry to the value array
        $valuearray += $value
    }
    #Return the Array
    $valuearray
}
```

`$valuearray = Get-NSGFlowLogReadBlock -blockList $blockList -CloudBlockBlob $CloudBlockBlob`

Now the `$valuearray` array contains the string value of each block. To verify the entry, get the second to the last

value from the array by running `$valuearray[$valuearray.Length-2]`. You do not want the last value, because it is the closing bracket.

The results of this value are shown in the following example:

```
{  
    "time": "2017-06-16T20:59:43.734000Z",  
    "systemId": "5f4d02d3-a7d0-4ed4-9ce8-c0ae9377951c",  
    "category": "NetworkSecurityGroupFlowEvent",  
    "resourceId": "/SUBSCRIPTIONS/00000000-0000-0000-0000-  
000000000000/RESOURCEGROUPS/CONTOSORG/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/CONTOSONSG",  
    "operationName": "NetworkSecurityGroupFlowEvents",  
    "properties": {"Version":1,"flows": [{"rule":"DefaultRule_AllowInternetOutBound","flows":  
[{"mac":"000D3A18077E","flowTuples":  
["1497646722,10.0.0.4,168.62.32.14,44904,443,T,O,A","1497646722,10.0.0.4,52.240.48.24,45218,443,T,O,A","149764  
6725,10.  
0.0.4,168.62.32.14,44910,443,T,O,A","1497646725,10.0.0.4,52.240.48.24,45224,443,T,O,A","1497646728,10.0.0.4,16  
8.62.32.14,44916,443,T,O,A","1497646728,10.0.0.4,52.240.48.24,45230,443,T,O,A","1497646732,10.0.0.4,168.62.32.  
14,44922,443,T,O,A","1497646732,10.0.0.4,52.240.48.24,45236,443,T,O,A","1497646735,10.0.0.4,168.62.32.14,44928,443,T,O,A","1497646735,10  
.0.0.4,52.240.48.24,45242,443,T,O,A","1497646738,10.0.0.4,168.62.32.14,44934,443,T,O,A","1497646738,10.0.0.4,5  
2.240.48.24,45248,443,T,O,A  
,"1497646742,10.0.0.4,168.62.32.14,44942,443,T,O,A","1497646742,10.0.0.4,52.240.48.24,45256,443,T,O,A","1497  
646745,10.0.0.4,168.62.32.14,44948,443,T,O,A","1497646745,10.0.0.4,52.240.48.24,45262,443,T,O,A","1497646749,1  
0.0.0.4,168.62.32.14,44954  
,"1497646749,10.0.0.4,52.240.48.24,45268,443,T,O,A","1497646753,10.0.0.4,168.62.32.14,44960,443,T,O,A  
,"1497646753,10.0.0.4,52.240.48.24,45274,443,T,O,A","1497646756,10.0.0.4,168.62.32.14,44966,443,T,O,A  
,"1497646756,10.0.0.4,52.240.48  
.24,45280,443,T,O,A","1497646759,10.0.0.4,168.62.32.14,44972,443,T,O,A","1497646759,10.0.0.4,52.240.48.24,4528  
6,443,T,O,A","1497646763,10.0.0.4,168.62.32.14,44978,443,T,O,A","1497646763,10.0.0.4,52.240.48.24,45292,443,T,  
O,A","1497646766,10.0.0.4  
,"1497646773,10.0.0.4,52.240.48.24,45310,443,T,O,A","1497646776,10.0.0.4,52.240.48.24,45298,443,T,O,A  
,"1497646769,10.0.0.4,168.62.32.14,45002,443,T,O,A","1497646776,10.0.0.4,52.240.48.24,45316,443,T,O,A  
,"1497646779,10.0.0.4,168.62.32.14,45008,443,T,O,A","1497646773,10.0.0.4,168.62.32.14,44990,443,T,O,A  
,"1497646773,10.0.0.4,52.240.48.24,45322,443,T,O,A"}]}  
,"rule":"DefaultRule_DenyAllInBound","flows":[]}, {"rule":"UserRule_ssh-rule","flows":[]},  
 {"rule":"UserRule_web-rule","flows": [{"mac":"000D3A18077E","flowTuples":  
["1497646738,13.82.225.93,10.0.0.4,1180,80,T,I,A","1497646750,13.82.225.93,10.0.0.4,  
1184,80,T,I,A","1497646768,13.82.225.93,10.0.0.4,1181,80,T,I,A","1497646780,13.82.225.93,10.0.0.4,1336,80,T,I,  
A"]}]}}}  
}
```

This scenario is an example of how to read entries in NSG flow logs without having to parse the entire log. You can read new entries in the log as they are written by using the block ID or by tracking the length of blocks stored in the block blob. This allows you to read only the new entries.

Next steps

Visit [Use Elastic Stack](#), [Use Grafana](#), and [Use Graylog](#) to learn more about ways to view NSG flow logs. An Open Source Azure Function approach to consuming the blobs directly and emitting to various log analytics consumers may be found here: [Azure Network Watcher NSG Flow Logs Connector](#).

You can use [Azure Traffic Analytics](#) to get insights on your traffic flows. Traffic Analytics uses [Log Analytics](#) to make your traffic flow queryable.

To learn more about storage blobs visit: [Azure Functions Blob storage bindings](#)

Traffic Analytics

2/26/2020 • 15 minutes to read • [Edit Online](#)

Traffic Analytics is a cloud-based solution that provides visibility into user and application activity in cloud networks. Traffic analytics analyzes Network Watcher network security group (NSG) flow logs to provide insights into traffic flow in your Azure cloud. With traffic analytics, you can:

- Visualize network activity across your Azure subscriptions and identify hot spots.
- Identify security threats to, and secure your network, with information such as open-ports, applications attempting internet access, and virtual machines (VM) connecting to rogue networks.
- Understand traffic flow patterns across Azure regions and the internet to optimize your network deployment for performance and capacity.
- Pinpoint network misconfigurations leading to failed connections in your network.

NOTE

Traffic Analytics now supports collecting NSG Flow Logs data at a higher frequency of 10 mins

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Why traffic analytics?

It is vital to monitor, manage, and know your own network for uncompromised security, compliance, and performance. Knowing your own environment is of paramount importance to protect and optimize it. You often need to know the current state of the network, who is connecting, where they're connecting from, which ports are open to the internet, expected network behavior, irregular network behavior, and sudden rises in traffic.

Cloud networks are different than on-premises enterprise networks, where you have netflow or equivalent protocol capable routers and switches, which provide the capability to collect IP network traffic as it enters or exits a network interface. By analyzing traffic flow data, you can build an analysis of network traffic flow and volume.

Azure virtual networks have NSG flow logs, which provide you information about ingress and egress IP traffic through a Network Security Group associated to individual network interfaces, VMs, or subnets. By analyzing raw NSG flow logs, and inserting intelligence of security, topology, and geography, traffic analytics can provide you with insights into traffic flow in your environment. Traffic Analytics provides information such as most communicating hosts, most communicating application protocols, most conversing host pairs, allowed/blocked traffic, inbound/outbound traffic, open internet ports, most blocking rules, traffic distribution per Azure datacenter, virtual network, subnets, or, rogue networks.

Key components

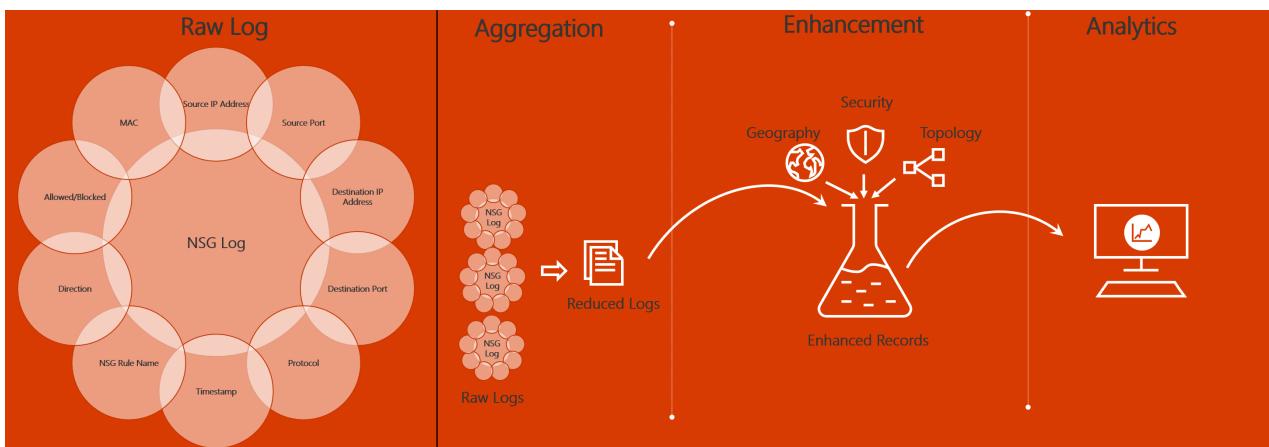
- **Network security group (NSG):** Contains a list of security rules that allow or deny network traffic to resources connected to an Azure Virtual Network. NSGs can be associated to subnets, individual VMs (classic), or individual network interfaces (NIC) attached to VMs (Resource Manager). For more information, see

Network security group overview.

- **Network security group (NSG) flow logs:** Allow you to view information about ingress and egress IP traffic through a network security group. NSG flow logs are written in json format and show outbound and inbound flows on a per rule basis, the NIC the flow applies to, five-tuple information about the flow (source/destination IP address, source/destination port, and protocol), and if the traffic was allowed or denied. For more information about NSG flow logs, see [NSG flow logs](#).
- **Log Analytics:** An Azure service that collects monitoring data and stores the data in a central repository. This data can include events, performance data, or custom data provided through the Azure API. Once collected, the data is available for alerting, analysis, and export. Monitoring applications such as network performance monitor and traffic analytics are built using Azure Monitor logs as a foundation. For more information, see [Azure Monitor logs](#).
- **Log Analytics workspace:** An instance of Azure Monitor logs, where the data pertaining to an Azure account, is stored. For more information about Log Analytics workspaces, see [Create a Log Analytics workspace](#).
- **Network Watcher:** A regional service that enables you to monitor and diagnose conditions at a network scenario level in Azure. You can turn NSG flow logs on and off with Network Watcher. For more information, see [Network Watcher](#).

How traffic analytics works

Traffic analytics examines the raw NSG flow logs and captures reduced logs by aggregating common flows among the same source IP address, destination IP address, destination port, and protocol. For example, Host 1 (IP address: 10.10.10.10) communicating to Host 2 (IP address: 10.10.20.10), 100 times over a period of 1 hour using port (for example, 80) and protocol (for example, http). The reduced log has one entry, that Host 1 & Host 2 communicated 100 times over a period of 1 hour using port 80 and protocol HTTP, instead of having 100 entries. Reduced logs are enhanced with geography, security, and topology information, and then stored in a Log Analytics workspace. The following picture shows the data flow:



Supported regions: NSG

You can use traffic analytics for NSGs in any of the following supported regions:

- Canada Central
- West Central US
- East US
- East US 2
- North Central US
- South Central US
- Central US
- West US
- West US 2

- France Central
- West Europe
- North Europe
- Brazil South
- UK West
- UK South
- Australia East
- Australia Southeast
- East Asia
- Southeast Asia
- Korea Central
- Central India
- South India
- Japan East
- Japan West
- US Gov Virginia
- China East 2

Supported regions: Log Analytics Workspaces

The Log Analytics workspace must exist in the following regions:

- Canada Central
- West Central US
- East US
- East US 2
- North Central US
- South Central US
- Central US
- West US
- West US 2
- Central US
- France Central
- West Europe
- North Europe
- Brazil South
- UK West
- UK South
- Australia East
- Australia Southeast
- East Asia
- Southeast Asia
- Korea Central
- Central India
- Japan East
- US Gov Virginia
- China East 2

Prerequisites

User access requirements

Your account must be a member of one of the following Azure [built-in roles](#):

DEPLOYMENT MODEL	ROLE
Resource Manager	Owner
	Contributor
	Reader
	Network Contributor

If your account is not assigned to one of the built-in roles, it must be assigned to a [custom role](#) that is assigned the following actions, at the subscription level:

- "Microsoft.Network/applicationGateways/read"
- "Microsoft.Network/connections/read"
- "Microsoft.Network/loadBalancers/read"
- "Microsoft.Network/localNetworkGateways/read"
- "Microsoft.Network/networkInterfaces/read"
- "Microsoft.Network/networkSecurityGroups/read"
- "Microsoft.Network/publicIPAddresses/read"
- "Microsoft.Network/routeTables/read"
- "Microsoft.Network/virtualNetworkGateways/read"
- "Microsoft.Network/virtualNetworks/read"

For information on how to check user access permissions, see [Traffic analytics FAQ](#).

Enable Network Watcher

To analyze traffic, you need to have an existing network watcher, or [enable a network watcher](#) in each region that you have NSGs that you want to analyze traffic for. Traffic analytics can be enabled for NSGs hosted in any of the [supported regions](#).

Select a network security group

Before enabling NSG flow logging, you must have a network security group to log flows for. If you don't have a network security group, see [Create a network security group](#) to create one.

In Azure portal, go to **Network watcher**, and then select **NSG flow logs**. Select the network security group that you want to enable an NSG flow log for, as shown in the following picture:

The screenshot shows the Network Watcher - NSG flow logs interface. On the left, there's a sidebar with various monitoring tools like Overview, Topology, Connection monitor (Preview), IP flow verify, Next hop, Security group view, VPN Diagnostics, Packet capture, and Connection troubleshoot. Under Metrics, it shows Network subscription limit. The LOGS section has a button for NSG flow logs, which is highlighted with a red box. The main pane displays a table of NSGs:

NAME	RESOURCE TYPE	RESOURCE GROUP	STATUS	LOCATION
myNsg1	Network security group	myResourceGroup	...	West Central US
myNsg2	Network security group	myResourceGroup	...	West Central US
myNsg3	Network security group	myResourceGroup	...	West Central US

If you try to enable traffic analytics for an NSG that is hosted in any region other than the [supported regions](#), you receive a "Not found" error.

Enable flow log settings

Before enabling flow log settings, you must complete the following tasks:

Register the Azure Insights provider, if it's not already registered for your subscription:

```
Register-AzResourceProvider -ProviderNamespace Microsoft.Insights
```

If you don't already have an Azure Storage account to store NSG flow logs in, you must create a storage account. You can create a storage account with the command that follows. Before running the command, replace <replace-with-your-unique-storage-account-name> with a name that is unique across all Azure locations, between 3-24 characters in length, using only numbers and lower-case letters. You can also change the resource group name, if necessary.

```
New-AzStorageAccount ` 
-Location westcentralus ` 
-Name <replace-with-your-unique-storage-account-name> ` 
-ResourceGroupName myResourceGroup ` 
-SkuName Standard_LRS ` 
-Kind StorageV2
```

Select the following options, as shown in the picture:

1. Select **On** for **Status**
2. Select **Version 2** for **Flow Logs version**. Version 2 contains flow-session statistics (Bytes and Packets)
3. Select an existing storage account to store the flow logs in. If you want to store the data forever, set the value to 0. You incur Azure Storage fees for the storage account. Ensure that your storage does not have "Data Lake Storage Gen2 Hierarchical Namespace Enabled" set to true.
4. Set **Retention** to the number of days you want to store data for.
5. Select **On** for **Traffic Analytics Status**.

- Select processing interval. Based on your choice, flow logs will be collected from storage account and processed by Traffic Analytics. You can choose processing interval of every 1 hour or every 10 mins.
- Select an existing Log Analytics (OMS) Workspace, or select **Create New Workspace** to create a new one. A Log Analytics workspace is used by Traffic Analytics to store the aggregated and indexed data that is then used to generate the analytics. If you select an existing workspace, it must exist in one of the [supported regions](#) and have been upgraded to the new query language. If you do not wish to upgrade an existing workspace, or do not have a workspace in a supported region, create a new one. For more information about query languages, see [Azure Log Analytics upgrade to new log search](#).

NOTE

The log analytics workspace hosting the traffic analytics solution and the NSGs do not have to be in the same region. For example, you may have traffic analytics in a workspace in the West Europe region, while you may have NSGs in East US and West US. Multiple NSGs can be configured in the same workspace.

8. Select **Save**.

Dashboard > AA-contoso-01-nsa - NSG flow logs > Flow logs settings

Flow logs settings

Save **Discard**

Flow logs

Status **On**

Flow Logs version **Version 2**

Version 1 logs ingress and egress IP traffic flows for both allowed and denied traffic. Version 2 provides additional throughput information (bytes and packets) per flow.
[Learn more](#)

Storage account contosobackupacc

Retention (days) 0

Traffic Analytics

Traffic Analytics provides rich analytics and visualization derived from NSG flow logs and other Azure resources' data. Drill through geo-map, easily figure out traffic hotspots and get insights into optimization possibilities.
[Learn about all features](#)

To use this feature, choose an Log Analytics workspace. To minimize data egress costs, we recommend that you choose a workspace in the same region your flow logs storage account is located. Network Performance Monitor solution will be installed on the workspace. We also advise that you use the same workspace for all NSGs as much as possible. Additional meta-data is added to your flow logs data, to provide enhanced analytics.

Traffic Analytics status **On**

Traffic Analytics processing interval **Every 1 hour**

Every 1 hour
Every 10 mins

Microsoft privacy statement
This privacy statement explains what personal data Microsoft collects from you, through our interactions with you and through our products, and how we use that data.

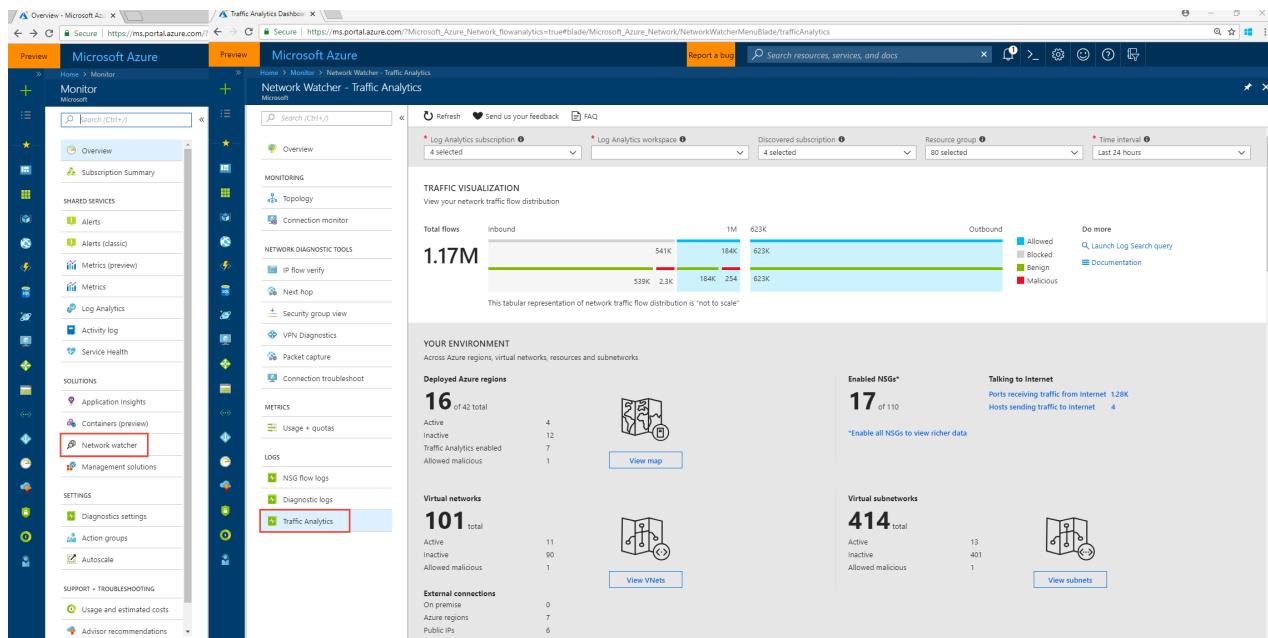
Repeat the previous steps for any other NSGs for which you wish to enable traffic analytics for. Data from flow logs is sent to the workspace, so ensure that the local laws and regulations in your country permit data storage in

the region where the workspace exists. If you have set different processing intervals for different NSGs, data will be collected at different intervals. For example: You can choose to enable processing interval of 10 mins for critical VNETs and 1 hour for noncritical VNETs.

You can also configure traffic analytics using the [Set-AzNetworkWatcherConfigFlowLog](#) PowerShell cmdlet in Azure PowerShell. Run `Get-Module -ListAvailable Az` to find your installed version. If you need to upgrade, see [Install Azure PowerShell module](#).

View traffic analytics

To view Traffic Analytics, search for **Network Watcher** in the portal search bar. Once inside Network Watcher, to explore traffic analytics and its capabilities, select **Traffic Analytics** from the left menu.



The dashboard may take up to 30 minutes to appear the first time because Traffic Analytics must first aggregate enough data for it to derive meaningful insights, before it can generate any reports.

Usage scenarios

Some of the insights you might want to gain after Traffic Analytics is fully configured, are as follows:

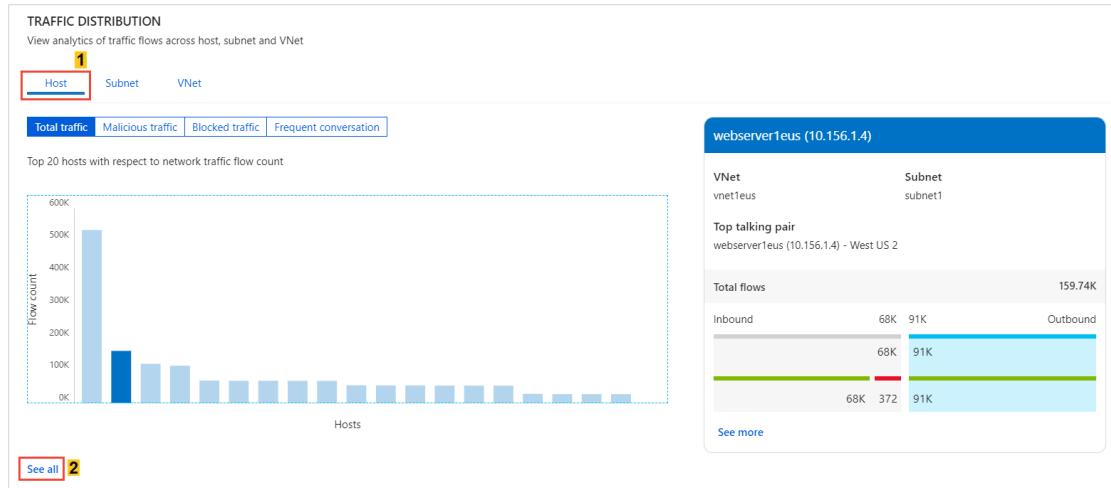
Find traffic hotspots

Look for

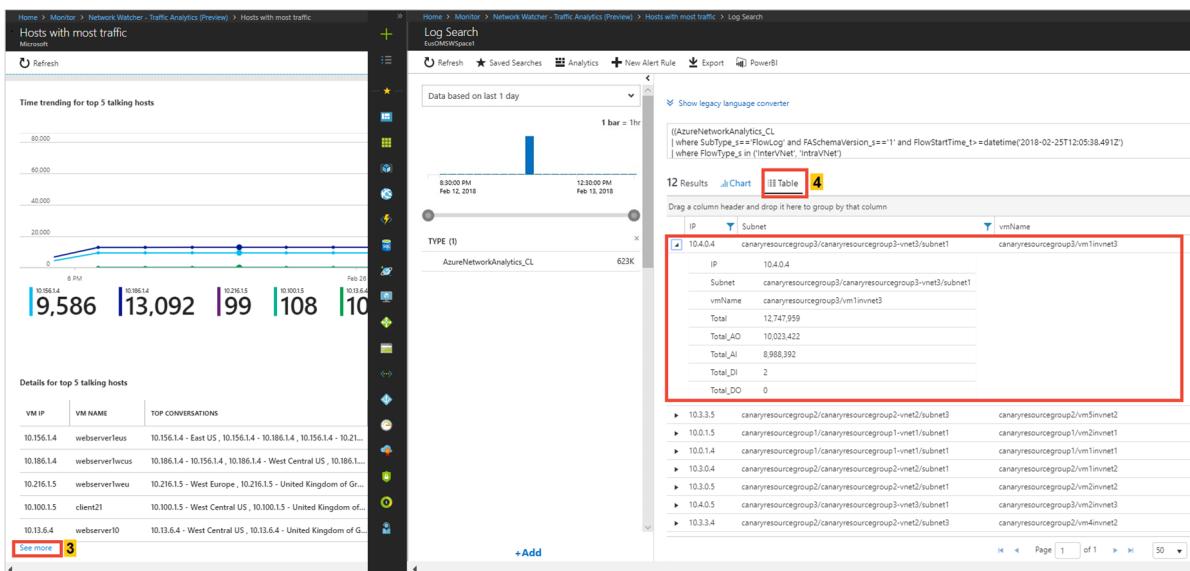
- Which hosts, subnets, and virtual networks are sending or receiving the most traffic, traversing maximum malicious traffic and blocking significant flows?
 - Check comparative chart for host, subnet, and virtual network. Understanding which hosts, subnets, and virtual networks are sending or receiving the most traffic can help you identify the hosts that are processing the most traffic, and whether the traffic distribution is done properly.
 - You can evaluate if the volume of traffic is appropriate for a host. Is the volume of traffic normal behavior, or does it merit further investigation?
- How much inbound/outbound traffic is there?
 - Is the host expected to receive more inbound traffic than outbound, or vice-versa?
- Statistics of blocked traffic.
 - Why is a host blocking a significant volume of benign traffic? This behavior requires further investigation and probably optimization of configuration

- Statistics of malicious allowed/blocked traffic
 - Why is a host receiving malicious traffic and why flows from malicious source is allowed? This behavior requires further investigation and probably optimization of configuration.

Select **See all**, under **Host**, as shown in the following picture:

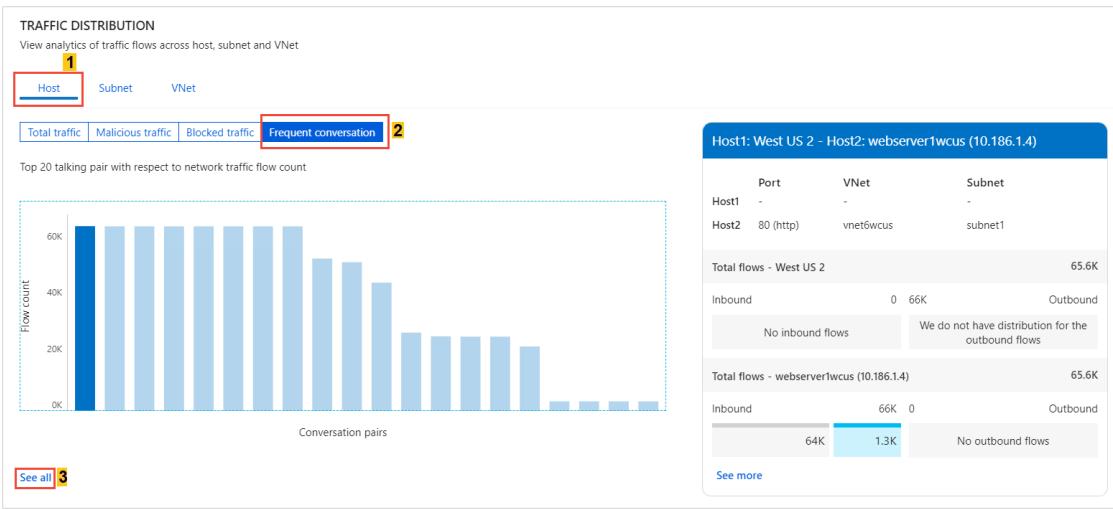


- The following picture shows time trending for the top five talking hosts and the flow-related details (allowed – inbound/outbound and denied - inbound/outbound flows) for a host:

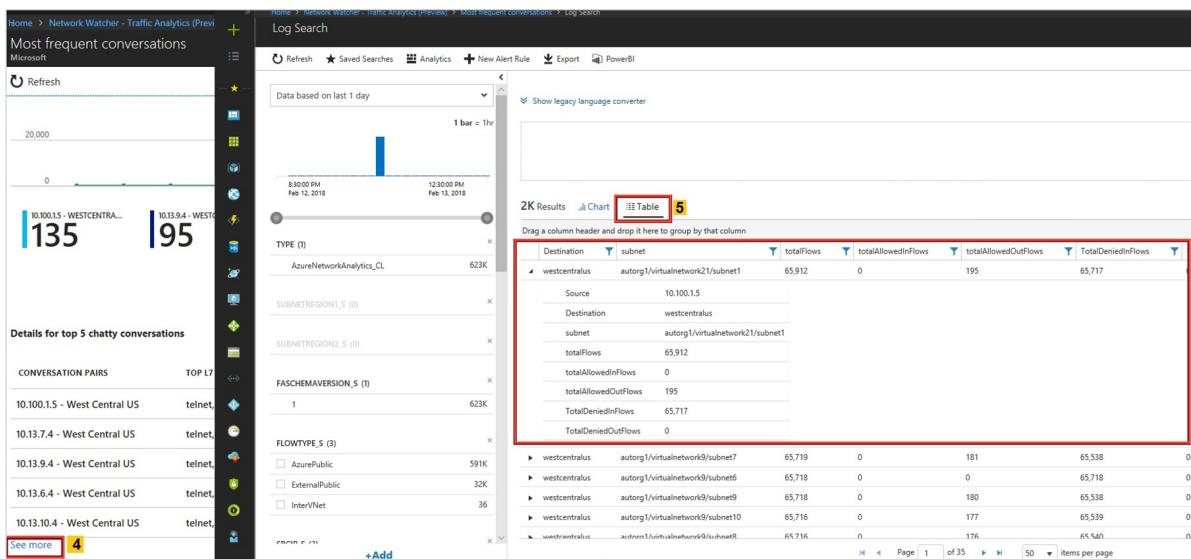


Look for

- Which are the most conversing host pairs?
 - Expected behavior like front-end or back-end communication or irregular behavior, like back-end internet traffic.
 - Statistics of allowed/blocked traffic
 - Why a host is allowing or blocking significant traffic volume
 - Most frequently used application protocol among most conversing host pairs:
 - Are these applications allowed on this network?
 - Are the applications configured properly? Are they using the appropriate protocol for communication? Select **See all** under **Frequent conversation**, as show in the following picture:

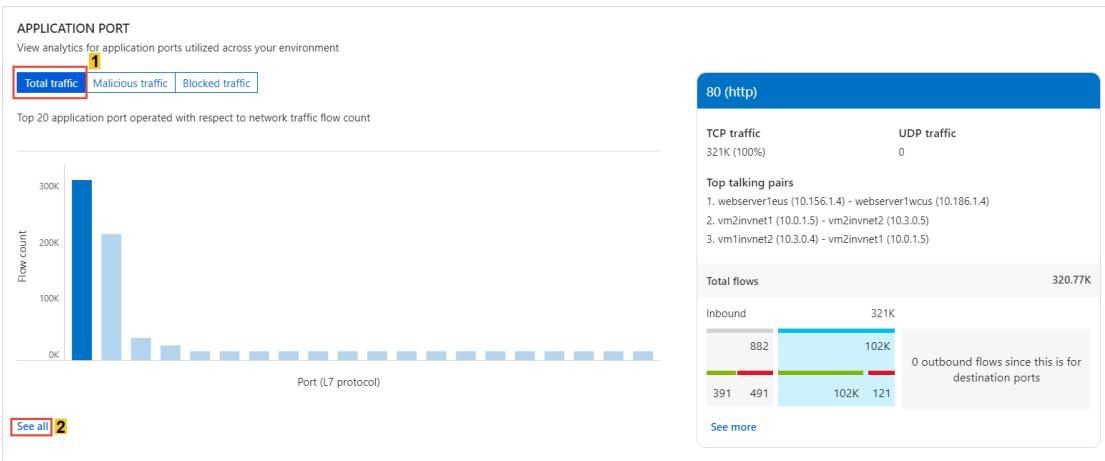


- The following picture shows time trending for the top five conversations and the flow-related details such as allowed and denied inbound and outbound flows for a conversation pair:



Look for

- Which application protocol is most used in your environment, and which conversing host pairs are using the application protocol the most?
 - Are these applications allowed on this network?
 - Are the applications configured properly? Are they using the appropriate protocol for communication? Expected behavior is common ports such as 80 and 443. For standard communication, if any unusual ports are displayed, they might require a configuration change. Select **See all** under **Application port**, in the following picture:



- The following pictures show time trending for the top five L7 protocols and the flow-related details (for example, allowed and denied flows) for an L7 protocol:

Home > Network Watcher - Traffic Analytics (Preview) > Most frequent L7 protocol

Most frequent L7 protocols Microsoft

Refresh

10,000
5,000
0

Feb 13
3389 443 60700 80 123 152

Details for top 5 L7 protocols

L7 PROTOCOL	L7 PROTOCOLS/DST PORT	TOP CONVERSATIONS
http	80	5 10.186.1.4 - 10.156.1.4
https	443	10.14.1.4 - West Central
ntp	123	10.4.1.5 - West Central
ms-wbt-server	3389	10.156.1.4 - East US , 10.156.1.4 - 10.156.1.4
Unknown	60700	10.186.1.4 - West Central

See more 3

Log Search

Data based on last 1 day

1 bar = 1hr

TYPE (1)

AzureNetworkAnalytics_CL 628K

SUBNETREGION1_S (0)

SUBNETREGION2_S (0)

FASCHEMAVERSION_S (1) 628K

+Add

66K Results Table 4

Drag a column header and drop it here to group by that column

L7Protocol_s	DestPort_d	TotalAllowedFlows	TotalDeniedFlows
http	80	103,540	624
		TotalAllowedFlows	103,540
		TotalDeniedFlows	624
		L7Protocol_s	http
		DestPort_d	80
https	443	6,683	419
ntp	123	1,059	239
ms-wbt-server	3389	0	0

Page 1 of 100

Log Search

Data based on last 1 day

1 bar = 1hr

AzureNetworkAnalytics_CL
| where SubType_s == "FlowLog" and FASchemaVersion_s == "1" and FlowStartTime_t >= datetime("2018-02-19T11:42:10.936Z")
| where FlowType_s != "MaliciousFlow"
| where FlowStartPort_d == 80 and FlowEndPort_d == 1024

973 Results Table 6

Drag a column header and drop it here to group by that column

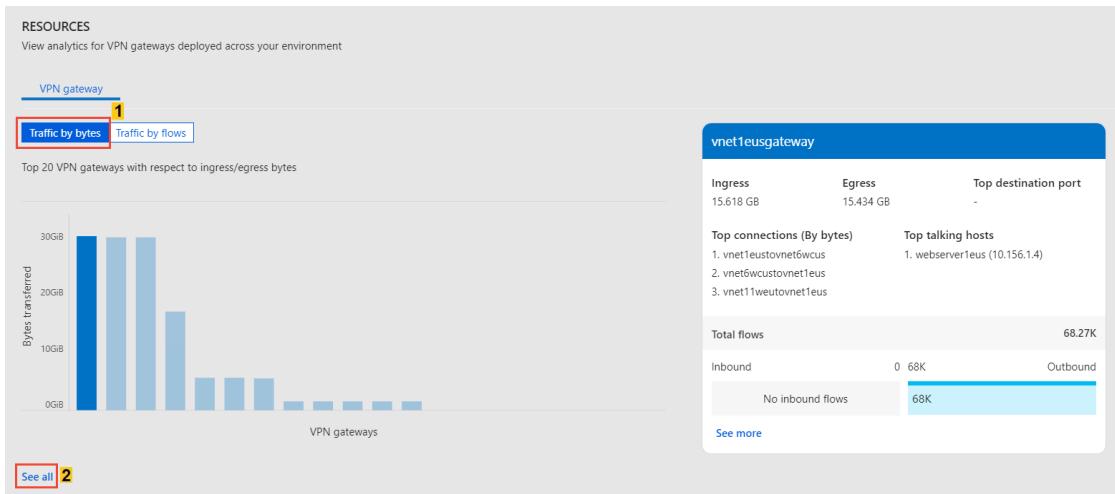
TimeGenerated	FASchemaVersion_s	FlowType_s	SrcIP_s	DestIP_s	VMIP_s	L4Protocol_s	L7Protocol_s	FlowDirection_s	NSGList_s
2018-02-20T08:43:56.169Z	1	ExternalPublic	10.156.1.4	T	http	I			a38f78b2-fb

Page 1 of 3 50 items per page 1 - 50 of 150 items

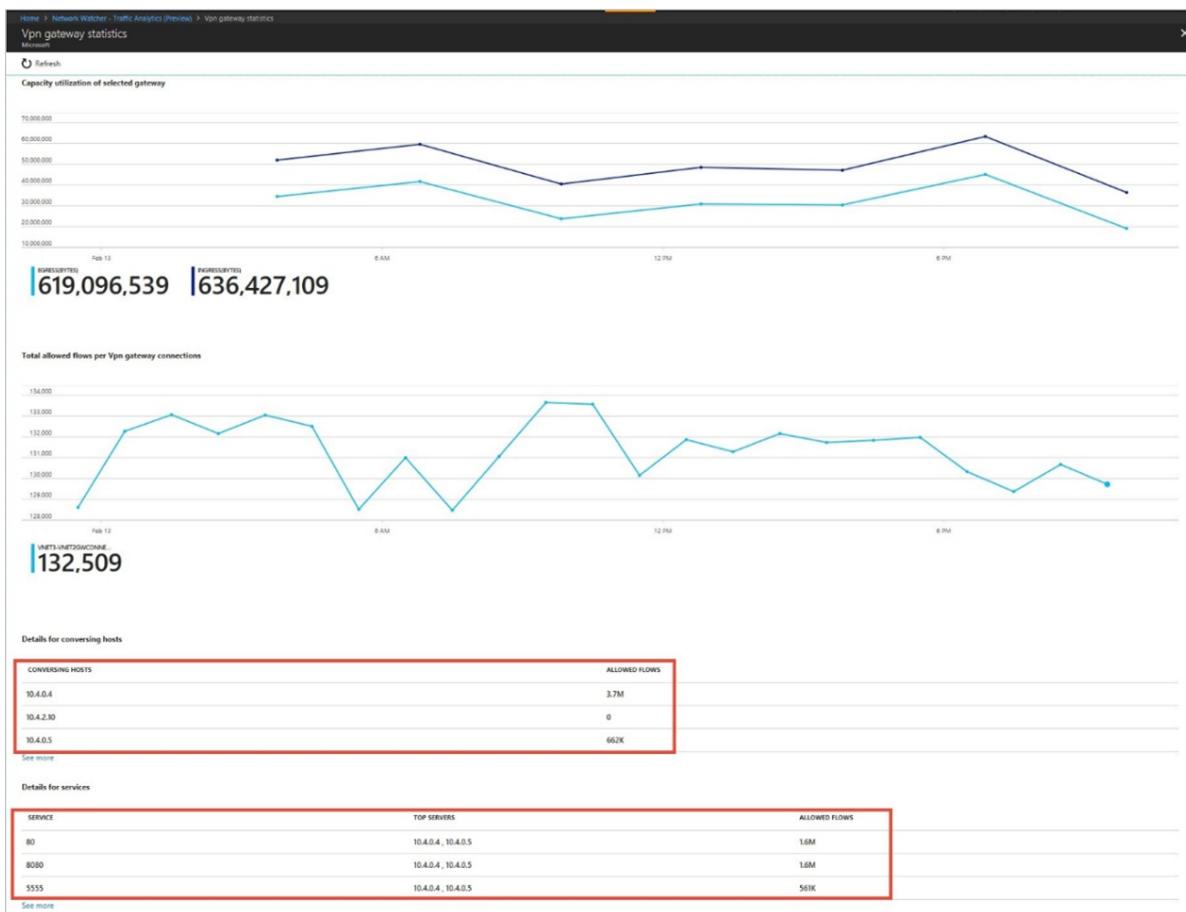
Look for

- Capacity utilization trends of a VPN gateway in your environment.
 - Each VPN SKU allows a certain amount of bandwidth. Are the VPN gateways underutilized?
 - Are your gateways reaching capacity? Should you upgrade to the next higher SKU?
- Which are the most conversing hosts, via which VPN gateway, over which port?

- Is this pattern normal? Select **See all** under **VPN gateway**, as shown in the following picture:



- The following picture shows time trending for capacity utilization of an Azure VPN Gateway and the flow-related details (such as allowed flows and ports):



Visualize traffic distribution by geography

Look for

- Traffic distribution per data center such as top sources of traffic to a datacenter, top rogue networks conversing with the data center, and top conversing application protocols.
 - If you observe more load on a data center, you can plan for efficient traffic distribution.
 - If rogue networks are conversing in the data center, then correct NSG rules to block them.

Select **View map** under **Your environment**, as shown in the following picture:

Home > Monitor > Network Watcher - Traffic Analytics

Network Watcher - Traffic Analytics

Missions

Search (Ctrl+ /)

Overview

MONITORING

- Topology
- Connection monitor

NETWORK DIAGNOSTIC TOOLS

- IP flow verify
- Next hop
- Security group view
- VPN Diagnostics
- Packet capture
- Connection troubleshoot

METRICS

- Usage + quotas

LOGS

- NSG flow logs
- Diagnostic logs
- Traffic Analytics 1

TRAFFIC VISUALIZATION

View your network traffic flow distribution

Total flows

Inbound

Outbound

1.17M

541K 184K 623K
539K 2.3K 184K 254 623K

This tabular representation of network traffic flow distribution is "not to scale".

Do more

- Launch Log Search query
- Documentation

YOUR ENVIRONMENT

Across Azure regions, virtual networks, resources and subnetworks

Deployed Azure regions

16 of 42 total

Active	4
Inactive	12
Traffic Analytics enabled	7
Allowed malicious	1

View map 2

Virtual networks

101 total

Active	11
Inactive	90
Allowed malicious	1

View VNets

External connections

On premise	0
Azure regions	7
Public IP	6

Enabled NSGs*

17 of 110

Talking to Internet

Ports receiving traffic from Internet 1.28K
Hosts sending traffic to Internet 4

*Enable all NSGs to view richer data

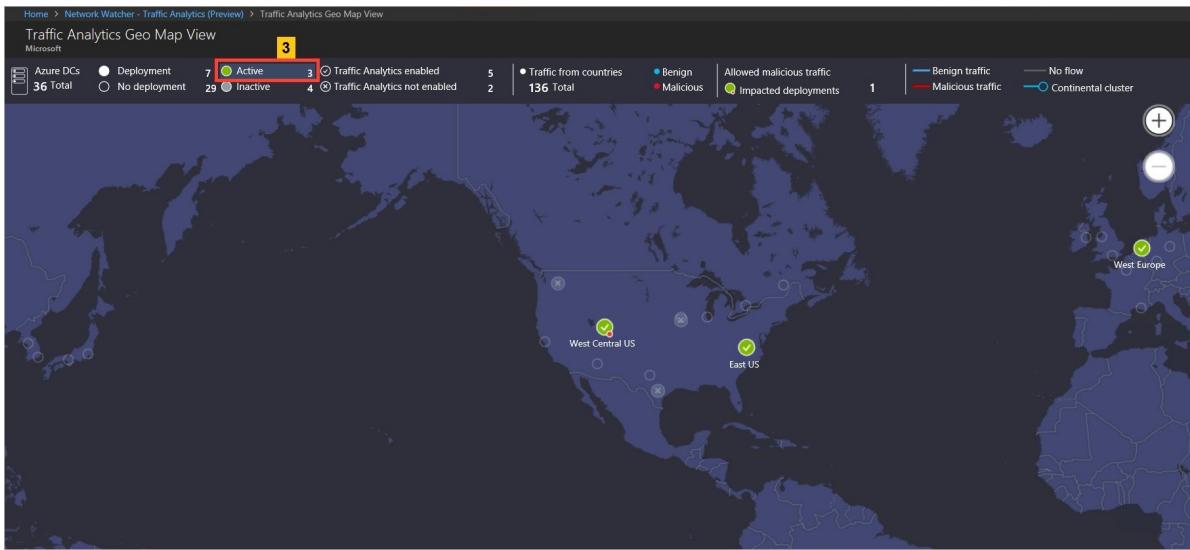
Virtual subnetworks

414 total

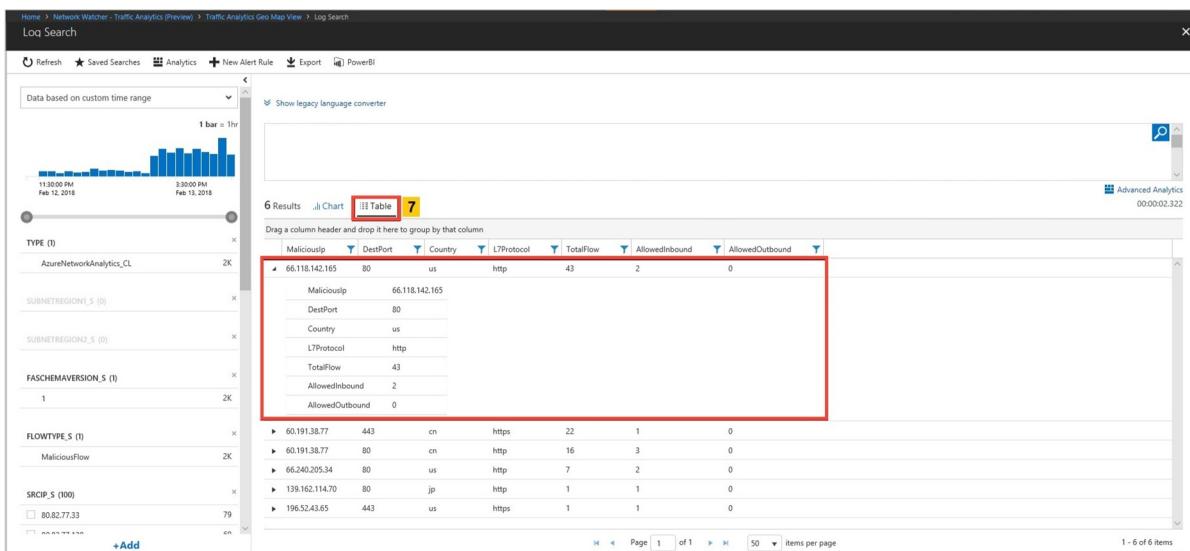
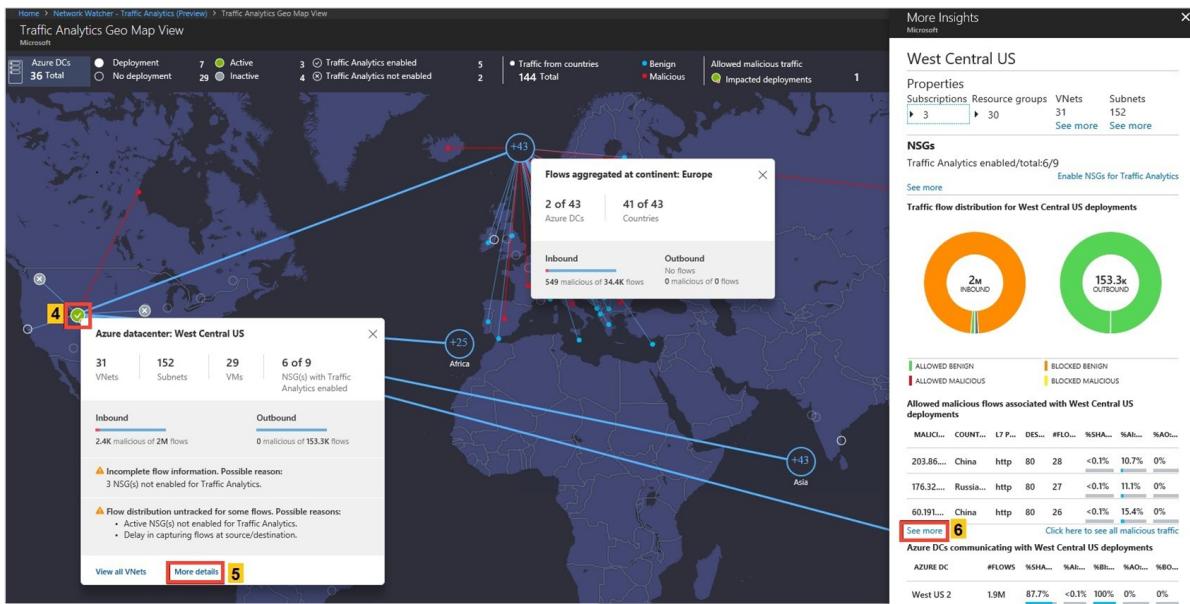
Active	13
Inactive	401
Allowed malicious	1

View subnets

- The geo-map shows the top ribbon for selection of parameters such as data centers (Deployed/No-deployment/Active/Inactive/Traffic Analytics Enabled/Traffic Analytics Not Enabled) and countries/regions contributing Benign/Malicious traffic to the active deployment:



- The geo-map shows the traffic distribution to a data center from countries/regions and continents communicating to it in blue (Benign traffic) and red (malicious traffic) colored lines:

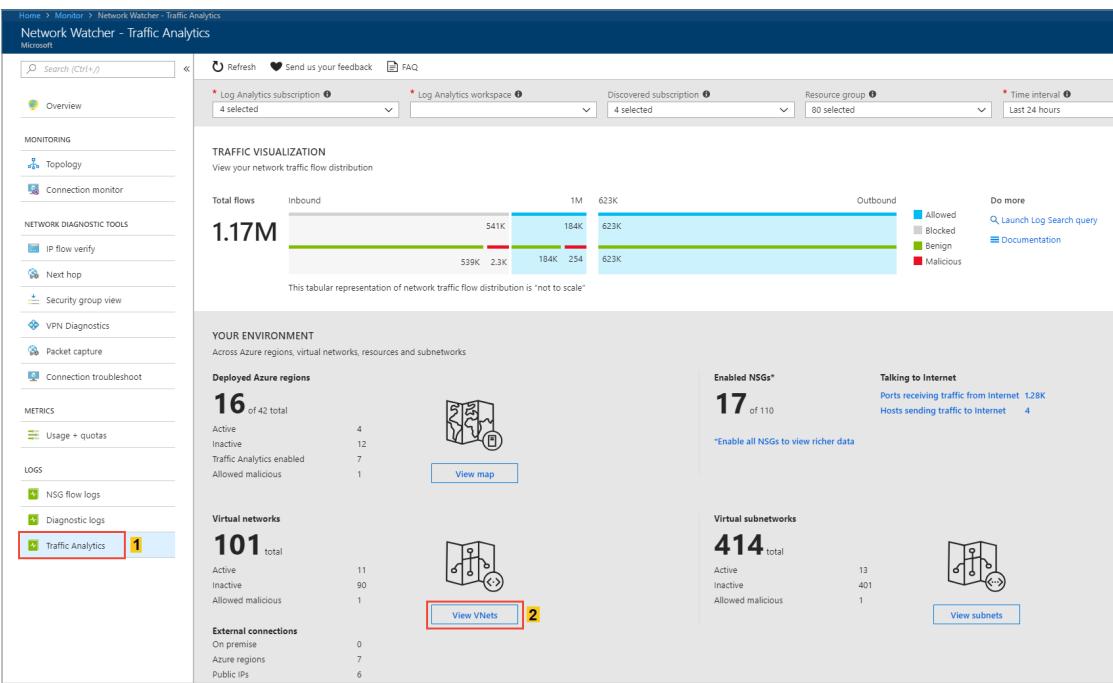


Visualize traffic distribution by virtual networks

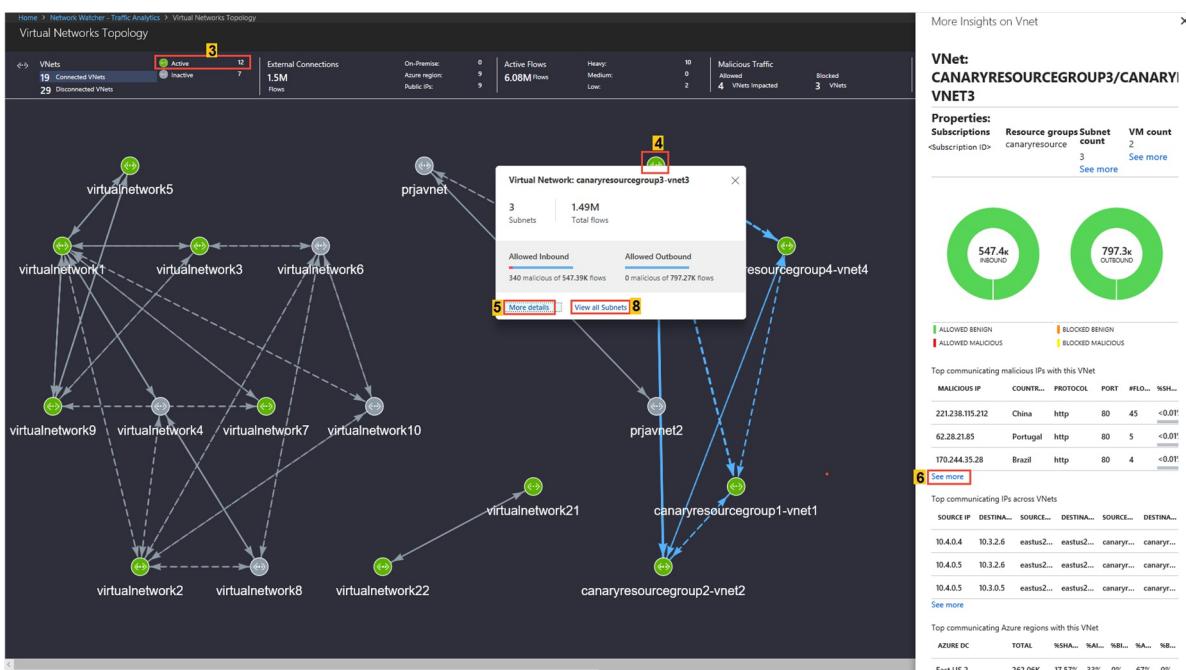
Look for

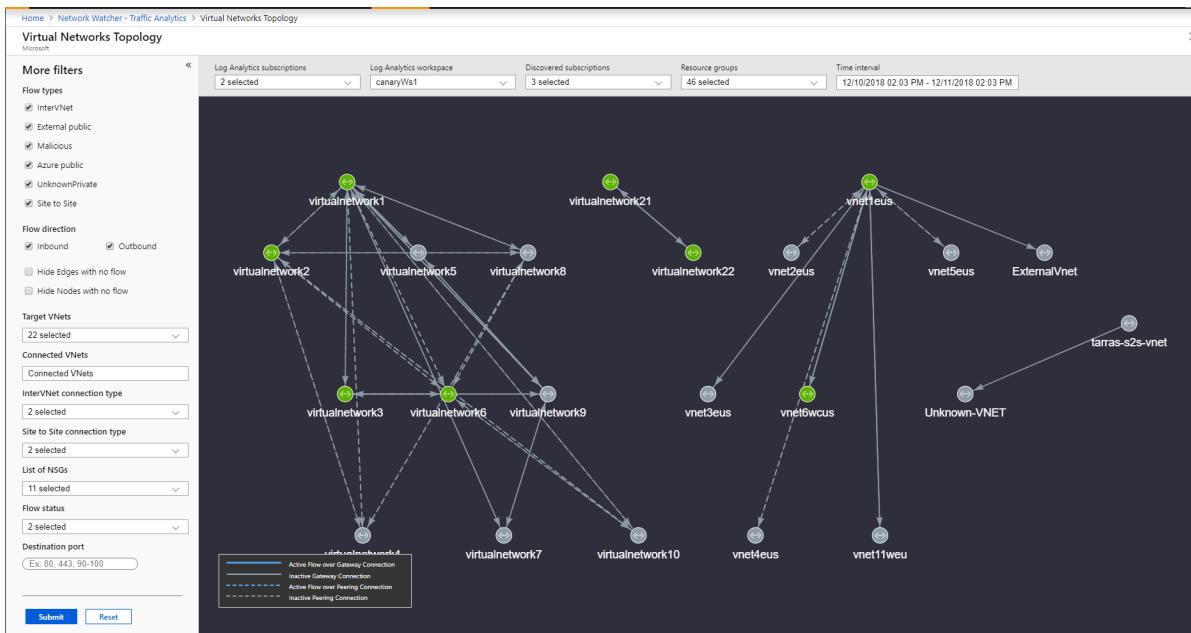
- Traffic distribution per virtual network, topology, top sources of traffic to the virtual network, top rogue networks conversing to the virtual network, and top conversing application protocols.
 - Knowing which virtual network is conversing to which virtual network. If the conversation is not expected, it can be corrected.
 - If rogue networks are conversing with a virtual network, you can correct NSG rules to block the rogue networks.

Select **View VNets** under **Your environment**, as shown in the following picture:



- The Virtual Network Topology shows the top ribbon for selection of parameters like a virtual network's (Inter virtual network Connections/Active/Inactive), External Connections, Active Flows, and Malicious flows of the virtual network.
- You can filter the Virtual Network Topology based on subscriptions, workspaces, resource groups and time interval. Additional filters that help you understand the flow are: Flow Type (InterVNet, IntraVNET, and so on), Flow Direction (Inbound, Outbound), Flow Status (Allowed, Blocked), VNets (Targeted and Connected), Connection Type (Peering or Gateway - P2S and S2S), and NSG. Use these filters to focus on VNets that you want to examine in detail.
- The Virtual Network Topology shows the traffic distribution to a virtual network with regards to flows (Allowed/Blocked/Inbound/Outbound/Benign/Malicious), application protocol, and network security groups, for example:





Home > Network Watcher - Traffic Analytics (Preview) > Log Search

Log Search

Refresh ★ Saved Searches Analytics + New Alert Rule Export PowerBI

Data based on last 1 day

Show legacy language converter

500 Results 7

Drag a column header and drop it here to group by that column

Maliciousip	GrandTotal	Country_s	DestPort_d	L7Protocol_s
2.224.61.44	21	it	8,080	http-alt
Maliciousip	2.224.61.44			
Country_s	it			
DestPort_d	8,080			
L7Protocol_s	http-alt			
GrandTotal	21			
118.166.242.122	21	tw	8,080	http-alt
217.96.11.7	21	pl	8,080	http-alt
65.118.142.165	14	us	80	http
2.224.61.44	12	it	80	http
118.166.242.122	12	tw	80	http
217.96.11.7	12	pl	80	http
107.170.237.16	6	us	118	sqlserv
A0.101.38.77	4	vn	4,432	https

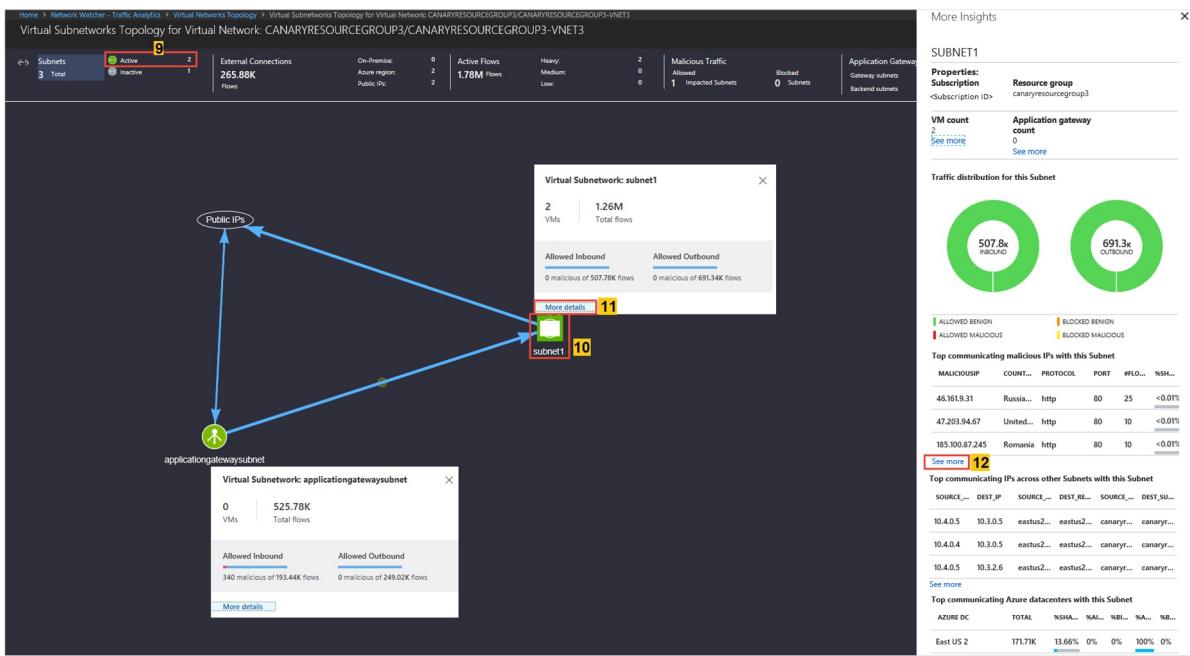
Advanced Analytics 00:00:04.92

+Add

Page 1 of 13 50 items per page 1 - 50 of 650 items

Look for

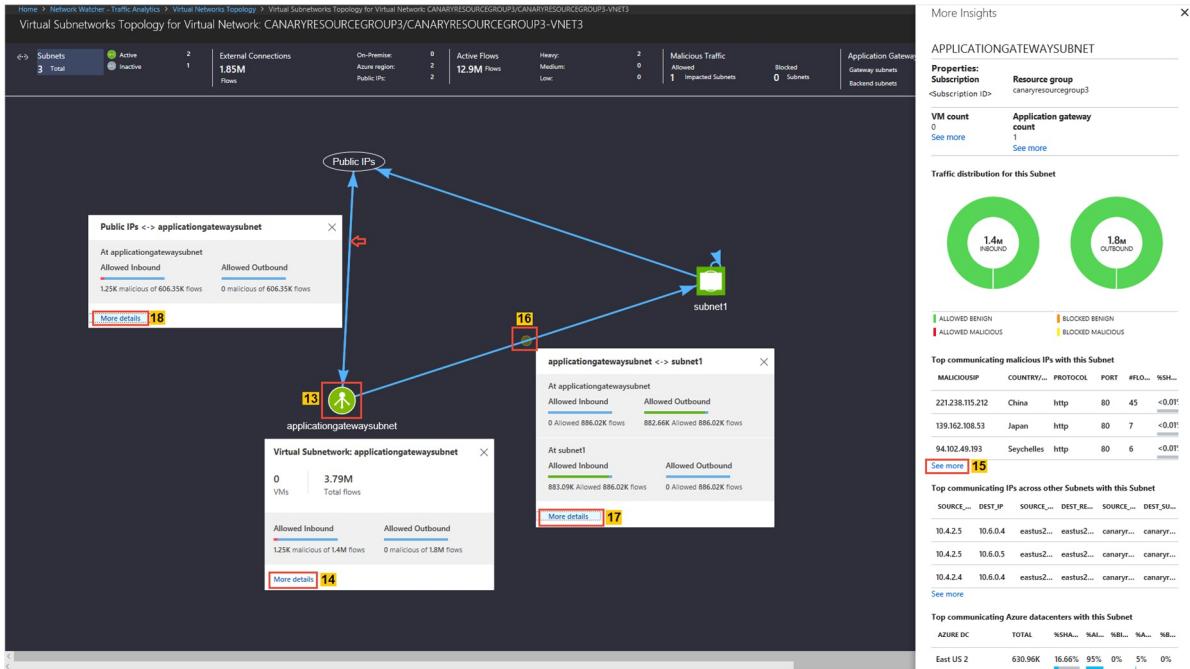
- Traffic distribution per subnet, topology, top sources of traffic to the subnet, top rogue networks conversing to the subnet, and top conversing application protocols.
 - Knowing which subnet is conversing to which subnet. If you see unexpected conversations, you can correct your configuration.
 - If rogue networks are conversing with a subnet, you are able to correct it by configuring NSG rules to block the rogue networks.
- The Subnets Topology shows the top ribbon for selection of parameters such as Active/Inactive subnet, External Connections, Active Flows, and Malicious flows of the subnet.
- The Subnet Topology shows the traffic distribution to a virtual network with regards to flows (Allowed/Blocked/Inbound/Outbound/Benign/Malicious), application protocol, and NSGs, for example:



Look for

Traffic distribution per Application gateway & Load Balancer, topology, top sources of traffic, top rogue networks conversing to the Application gateway & Load Balancer, and top conversing application protocols.

- Knowing which subnet is conversing to which Application gateway or Load Balancer. If you observe unexpected conversations, you can correct your configuration.
- If rogue networks are conversing with an Application gateway or Load Balancer, you are able to correct it by configuring NSG rules to block the rogue networks.



View ports and virtual machines receiving traffic from the internet

Look for

- Which open ports are conversing over the internet?
 - If unexpected ports are found open, you can correct your configuration:

Home > Monitor > Network Watcher - Traffic Analytics

Network Watcher - Traffic Analytics

Missions

Search (Ctrl + /)

Overview

MONITORING

- Topology
- Connection monitor

NETWORK DIAGNOSTIC TOOLS

- IP flow verify
- Next hop
- Security group view
- VPN Diagnostics
- Packet capture
- Connection troubleshoot

METRICS

- Usage + quotas

LOGS

- NSG flow logs
- Diagnostic logs
- Traffic Analytics 1

TRAFFIC VISUALIZATION

View your network traffic flow distribution

Total flows

Inbound

Outbound

1.17M

541K 2.3K 184K 623K 623K 254K 5

This tabular representation of network traffic flow distribution is "not to scale"

YOUR ENVIRONMENT

Across Azure regions, virtual networks, resources and subnetworks

Deployed Azure regions

16 of 42 total

Active: 4
Inactive: 12
Traffic Analytics enabled: 7
Allowed malicious: 1

View map

Virtual networks

101 total

Active: 11
Inactive: 90
Allowed malicious: 1

View VNets

External connections

On premise: 0
Azure regions: 7
Public IP: 6

Enabled NSGs*

17 of 110

Talking to Internet

Ports receiving traffic from internet: 128K 2
Hosts sending traffic to Internet: 4

*Enable all NSGs to view richer data

Virtual subnetworks

414 total

Active: 13
Inactive: 401
Allowed malicious: 1

View subnets

Home > Network Watcher - Traffic Analytics (Preview) > Ports receiving traffic from internet

Ports receiving traffic from internet

Refresh

Log Analytics workspace: Discovered subscription: Resource group: 11 selected

Details for Azure destination ports

L4 PROTOCOLS	PORTS	TOP L7 PRO
TCP	80, 8080, 5555	http, https
UDP	-	-

See more

Details for Azure destination hosts opened for TCP

AZURE DESTINATION HOST IP	AZURE DESTINATION HOST VM	SUBNET	VNET	PORT
10.3.0.4	vml1vnnet2	subnet1	canaryresourcegroup2-vnet2	80
10.3.0.4	vml1vnnet2	subnet1	canaryresourcegroup2-vnet2	8080
10.3.3.4	vm4vnnet2	subnet3	canaryresourcegroup2-vnet2	5555

See more 3

Details for Azure destination hosts opened for UDP

See more

Log Search

Data based on last 1 day

1 bar = 1hr

TYPE (1)

AzureNetworkAnalytics_CL 290

SUBNETREGION1_S (0)

SUBNETREGION2_S (0)

NEXTHOP_IP_V5 (0)

NEXTHOP_TYPE_V5 (0)

ALLOWFORWARDEDTRAFFIC_B (0)

ALLOWGATEWAYTRANSIT_B (0)

ALLOWVIRTUALNETWORKACCESS_B (0)

8 Results 4

Drag a column header and drop it here to group by that column

sum_FlowCount_d	sum_AllowedOutflows_d	sum_AllowedInflows_d
290	0	691.126
VMP_3	10.3.3.4	canaryresourcegroup2-vml1vnnet2
VMProtocol_3	personal-agent	
DestPort_d	5.555	
sum_FlowCount_d	691.126	
sum_AllowedOutflows_d	0	
sum_AllowedInflows_d	691.126	
sum_DeniedFlows_d	0	
sum_DeniedOutflows_d	0	
691.121	0	691.121
683.103	0	683.103
683.075	0	683.075

Look for

Do you have malicious traffic in your environment? Where is it originating from? Where is it destined to?

Home > Monitor > Network Watcher - Traffic Analytics (Preview) > Log Search

Log Search

Refresh 5

Saved Searches 6

Analytics 7

New Alert Rule 8

Export 9

PowerBI 10

Data based on last 1 day

1 bar = 1hr

TimeGenerated FlowDirection_s FlowType_s FlowCount_d AllowedInFlows_d AllowedOutFlows_d SrcIP_s DestIP_s L4Protocol_s

32 Results 6

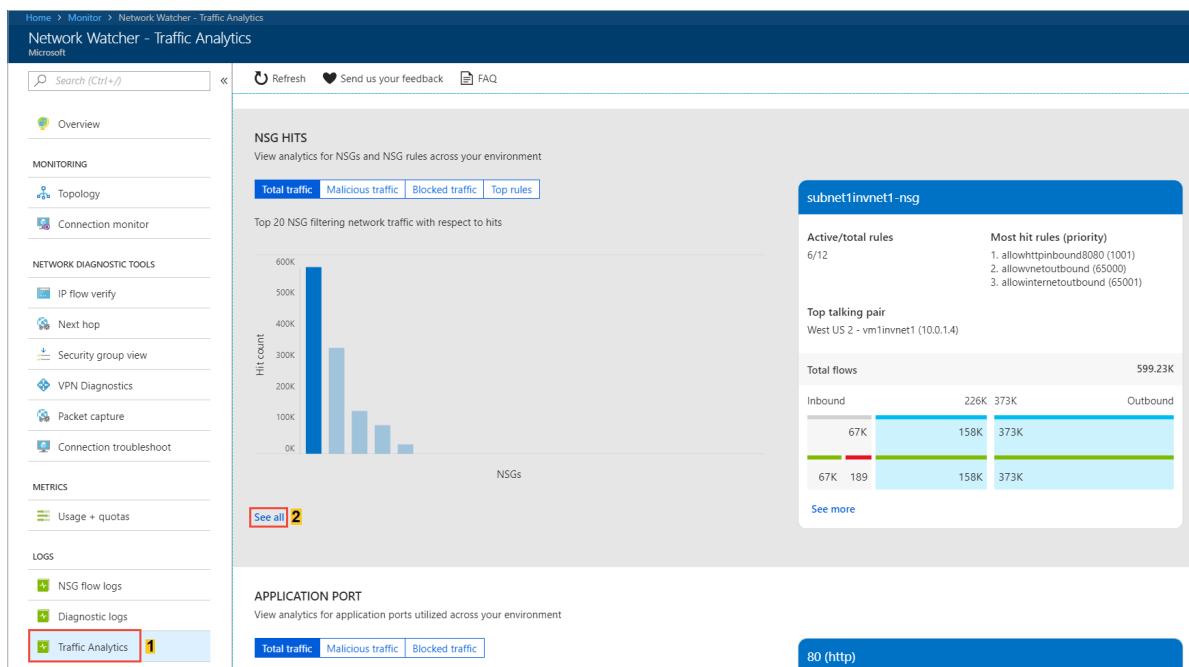
Drag a column header and drop it here to group by that column

TimeGenerated	FlowDirection_s	FlowType_s	FlowCount_d	AllowedInFlows_d	AllowedOutFlows_d	SrcIP_s	DestIP_s	L4Protocol_s
2/14/2018 3:40:14.114 PM	I	MaliciousFlow	1	1	0	196.52.43.57	10.4.1.4	T
2/14/2018 4:40:15.739 PM	I	MaliciousFlow	1	1	0	60.191.38.77	10.7.1.4	T
2/14/2018 1:40:26.930 PM	I	MaliciousFlow	2	2	0	60.191.38.77	10.14.1.5	T
2/14/2018 1:40:14.189 PM	I	MaliciousFlow	1	1	0	66.118.142.165	10.7.1.4	T
2/14/2018 2:40:14.189 PM	I	MaliciousFlow	3	3	0	188.152.239.48	10.14.1.4	T
2/14/2018 11:40:07.363 AM	I	MaliciousFlow	1	1	0	137.226.113.10	10.14.1.5	T
2/14/2018 11:40:07.363 AM	I	MaliciousFlow	1	1	0	176.32.33.214	10.14.1.5	T
2/14/2018 10:40:03.918 AM	I	MaliciousFlow	1	1	0	176.32.33.214	10.4.1.4	T
2/14/2018 9:40:12.897 AM	I	MaliciousFlow	2	2	0	54.162.46.58	10.4.1.4	T
2/14/2018 9:40:12.897 AM	I	MaliciousFlow	1	1	0	196.52.43.55	10.7.1.4	T
66.118.142.165			7					

Visualize the trends in NSG/NSG rules hits

Look for

- Which NSG/NSG rules have the most hits in comparative chart with flows distribution?
- What are the top source and destination conversation pairs per NSG/NSG rules?



- The following pictures show time trending for hits of NSG rules and source-destination flow details for a network security group:
 - Quickly detect which NSGs and NSG rules are traversing malicious flows and which are the top malicious IP addresses accessing your cloud environment
 - Identify which NSG/NSG rules are allowing/blocking significant network traffic
 - Select top filters for granular inspection of an NSG or NSG rules

NSG rule hits

Refresh

Log Analytics workspace: Discovered subscription: Resource group: NSG: NSG name/Rule name: Time interval: Last 7 days

Time trending chart for hits of NSGs

600,000
400,000
200,000
0

Jun 7 Jun 8 Jun 9 Jun 10 Jun 11 Jun 12 Jun 13

66204AB-8NC6-FD2D-AB... 464
AFT5E2TS-PB8-49AC-BC... 28,935
A3F78B2-FAAD-462B-90F... 92
A3F78B2-FAAD-462B-90F... 116
AFT5E2TS-PB8-49AC-BC... 87

Details of top 5 NSGs

NSG NAME	TOP 3 RULES (PRIORITY)	TOP TALKING PAIR	TOP 3 MALICIOUS IP	TOTAL HITS	TOTAL INBOUND (ALLOWED/BLOCKED)	INBOUND ALLOWED (BENIGN/MALICIOUS)	INBOUND BLOCKED (BENIGN/MALICIOUS)	TOTAL OUTBOUND (ALLOWED/BLOCKED)	OUTBOUND ALLOWED (BENIGN/MALICIOUS)	OUTBOUND BLOCKED (BENIGN/MALICIOUS)
subnetInVnet1-nsg	defaultrule, allowmetabound(55000) defaultrule, allowhttpinbound(55001)	10.0.1.5 - eastus2eup	10.0.1.4	5.9M	2.3M (1.7M/518.5K)	1.7M (1.7M/106)	538.5K (537.6K/90)	5.3M (5.3M/0)	5.3M (5.3M/0)	0 (0/0)
autonsg3	defaultrule, denyallinbound(65500)	10.100.1.5 - westcentralus	10.9.1.5 10.9.1.4	2.4M	2.4M (0/2.4M)	0 (0/0)	2.4M (2.4M/7.3K)	2.1K (2.1K/0)	2.1K (2.1K/0)	0 (0/0)
webserverInVwos-nsg	defaultrule, denyallinbound(65500) userrule, port-23-1000(100)	10.186.1.4 - westcentralus	-	611.5K	609.8K (20.1K/589.6K)	20.1K (18.4K/1.8K)	589.6K (589.1K/519)	1.7K (1.7K/0)	1.7K (1.7K/0)	0 (0/0)
webserverInVwo-nsg	defaultrule, denyallinbound(65500)	10.216.1.5 - westeurope	10.216.1.5	543.2K	542.2K (0/542.2K)	0 (0/0)	542.2K (540.4K/1.7K)	1.1K (1.1K/0)	1.1K (1.1K/0)	0 (0/0)
azmonet4951	defaultrule, denyallinbound(65500) defaultrule, allowinternetcntrbound(55001)	10.0.3.4 - eastus2eup	10.0.3.4	402K	402K (0/402K)	0 (0/0)	402K (400.9K/0)	32 (32/0)	32 (32/0)	0 (0/0)

[See more](#)

Time trending chart for hits of NSG rules

600,000
400,000
200,000
0

Jun 7 Jun 8 Jun 9 Jun 10 Jun 11 Jun 12 Jun 13

DEFALUTABLE DENY... 464
DEFALUTABLE ALLOW... 12,146
DEFALUTABLE ALLOW... 8,315
USERRULE ALLOWIT... 4,200
USERRULE ALLOWIT... 4,200

Details of top 5 NSG rules

RULE NAME	ACCESS/PRIORITY	RULE TYPE	NSG	TOP 3 SOURCE-DESTINATION	TOP 3 ASSOCIATED SUBNETS	TOP 3 ASSOCIATED NETWORK INTERFACES	HITS	TOP 3 MALICIOUS IP	MALICIOUS (ALLOWED/DENIED)
defaultrule_denyinbound	(Deny/65500)	↓ Inbound	autonsg3	West Central US-10.100.1.5 West Central US-10.101.4 West Central US-10.102.5	virtualnetwork2/vnet1 virtualnetwork2/vnet1 virtualnetwork2/vnet1	client2252 webserver2259 vm2innvme429	2.4M	10.101.1.4	7.3K (0 / 7.3K)
defaultrule_allowmetabound	(Allow/65500)	↑ Outbound	subnetInVnet1-nsg	10.0.1.5-10.3.2.6 10.0.1.5-10.3.0.4 10.0.1.5-10.3.0.5	canaryresourcegroup1/vnet1/subnet1 canaryresourcegroup1/vnet2/subnet1 canaryresourcegroup1/vnet2/loadbalancerubert	Unknown-NIC Unknown-NIC vm2innvme4102	2.2M	-	0 (0 / 0)
defaultrule_allowinternetcntrbound	(Allow/65500)	↑ Outbound	subnetInVnet1-nsg	10.0.1.5-40.80.71.185 10.0.1.4-40.80.71.185	canaryresourcegroup1/vnet1/subnet1	vm2innvme4102 vm2innvme4143	1.5M	-	0 (0 / 0)
userrule_allowhttpinbound80	(Allows/1000)	↓ Inbound	subnetInVnet1-nsg	10.0.1.5-10.0.1.5 10.0.1.5-10.0.1.5 10.0.1.4-10.0.1.5	canaryresourcegroup4/vnet4/subnet1 canaryresourcegroup4/vnet4/subnet1 canaryresourcegroup4/vnet4/subnet1	vm2innvme4156 vm2innvme4142 vm2innvme4548	847.1K	10.0.1.4	868.6 (0 / 0)
azmonet4951	allow/1000	↓ Inbound	autonsg3	10.0.4.0-10.0.1.5 10.0.4.0-10.0.1.5 10.0.4.0-10.0.1.5	canaryresourcegroup4/vnet4/subnet1 canaryresourcegroup4/vnet4/subnet1 canaryresourcegroup4/vnet4/subnet1	vm2innvme4156 vm2innvme4142 vm2innvme4548	1.5M	-	0 (0 / 0)

Frequently asked questions

To get answers to frequently asked questions, see [Traffic analytics FAQ](#).

Next steps

- To learn how to enable flow logs, see [Enabling NSG flow logging](#).

- To understand the schema and processing details of Traffic Analytics, see [Traffic analytics schema](#).

Traffic Analytics frequently asked questions

1/28/2020 • 15 minutes to read • [Edit Online](#)

This article collects in one place many of the most frequently asked questions about traffic analytics in Azure Network Watcher.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

What are the prerequisites to use traffic analytics?

Traffic Analytics requires the following prerequisites:

- A Network Watcher enabled subscription.
- Network Security Group (NSG) flow logs enabled for the NSGs you want to monitor.
- An Azure Storage account, to store raw flow logs.
- An Azure Log Analytics workspace, with read and write access.

Your account must meet one of the following to enable traffic analytics:

- Your account must have any one of the following role-based access control (RBAC) roles at the subscription scope: owner, contributor, reader, or network contributor.
- If your account is not assigned to one of the previously listed roles, it must be assigned to a custom role that is assigned the following actions, at the subscription level.
 - Microsoft.Network/applicationGateways/read
 - Microsoft.Network/connections/read
 - Microsoft.Network/loadBalancers/read
 - Microsoft.Network/localNetworkGateways/read
 - Microsoft.Network/networkInterfaces/read
 - Microsoft.Network/networkSecurityGroups/read
 - Microsoft.Network/publicIPAddresses/read
 - Microsoft.Network/routeTables/read
 - Microsoft.Network/virtualNetworkGateways/read
 - Microsoft.Network/virtualNetworks/read

To check roles assigned to a user for a subscription:

1. Sign in to Azure by using **Login-AzAccount**.
2. Select the required subscription by using **Select-AzSubscription**.
3. To list all the roles that are assigned to a specified user, use **Get-AzRoleAssignment -SignInName [user email] -IncludeClassicAdministrators**.

If you are not seeing any output, contact the respective subscription admin to get access to run the commands. For

more details, see [Manage role-based access control with Azure PowerShell](#).

In which Azure regions is Traffic Analytics available?

You can use traffic analytics for NSGs in any of the following supported regions:

- Canada Central
- West Central US
- East US
- East US 2
- North Central US
- South Central US
- Central US
- West US
- West US 2
- France Central
- West Europe
- North Europe
- Brazil South
- UK West
- UK South
- Australia East
- Australia Southeast
- East Asia
- Southeast Asia
- Korea Central
- Central India
- South India
- Japan East
- Japan West
- US Gov Virginia
- China East 2

The Log Analytics workspace must exist in the following regions:

- Canada Central
- West Central US
- East US
- East US 2
- North Central US
- South Central US
- Central US
- West US
- West US 2
- France Central
- West Europe
- North Europe
- UK West
- UK South

- Australia East
- Australia Southeast
- East Asia
- Southeast Asia
- Korea Central
- Central India
- Japan East
- US Gov Virginia
- China East 2

Can the NSGs I enable flow logs for be in different regions than my workspace?

Yes, these NSGs can be in different regions than your Log Analytics workspace.

Can multiple NSGs be configured within a single workspace?

Yes.

Can I use an existing workspace?

Yes. If you select an existing workspace, make sure that it has been migrated to the new query language. If you do not want to upgrade the workspace, you need to create a new one. For more information about the new query language, see [Azure Monitor logs upgrade to new log search](#).

Can my Azure Storage Account be in one subscription and my Log Analytics workspace be in a different subscription?

Yes, your Azure Storage account can be in one subscription, and your Log Analytics workspace can be in a different subscription.

Can I store raw logs in a different subscription?

No. You can store raw logs in any storage account where an NSG is enabled for flow logs. However, both the storage account and the raw logs must be in the same subscription and region.

What if I can't configure an NSG for traffic analytics due to a "Not found" error?

Select a supported region. If you select a non-supported region, you receive a "Not found" error. The supported regions are listed earlier in this article.

What if I am getting the status, "Failed to load," under the NSG flow logs page?

The Microsoft.Insights provider must be registered for flow logging to work properly. If you are not sure whether the Microsoft.Insights provider is registered for your subscription, replace `xxxxx-xxxxx-xxxxx-xxxx` in the following command, and run the following commands from PowerShell:

```
**Select-AzSubscription** -SubscriptionId xxxxx-xxxxx-xxxxxx-xxxx  
**Register-AzResourceProvider** -ProviderNamespace Microsoft.Insights
```

I have configured the solution. Why am I not seeing anything on the dashboard?

The dashboard might take up to 30 minutes to appear the first time. The solution must first aggregate enough data for it to derive meaningful insights. Then it generates reports.

What if I get this message: "We could not find any data in this workspace for selected time interval. Try changing the time interval or select a different workspace."?

Try the following options:

- Change the time interval in the upper bar.
- Select a different Log Analytics workspace in the upper bar.
- Try accessing traffic analytics after 30 minutes, if it was recently enabled.

If problems persist, raise concerns in the [User voice forum](#).

What if I get this message: "Analyzing your NSG flow logs for the first time. This process may take 20-30 minutes to complete. Check back after some time. 2) If the above step doesn't work and your workspace is under the free SKU, then check your workspace usage here to validate over quota, else refer to FAQs for further information."?

You might see this message because:

- Traffic Analytics was recently enabled, and might not yet have aggregated enough data for it to derive meaningful insights.
- You are using the free version of the Log Analytics workspace, and it exceeded the quota limits. You might need to use a workspace with a larger capacity.

If problems persist, raise concerns in the [User voice forum](#).

What if I get this message: "Looks like we have resources data (Topology) and no flows information. Meanwhile, click here to see resources data and refer to FAQs for further information."?

You are seeing the resources information on the dashboard; however, no flow-related statistics are present. Data might not be present because of no communication flows between the resources. Wait for 60 minutes, and recheck status. If the problem persists, and you're sure that communication flows among resources exist, raise concerns in the [User voice forum](#).

Can I configure traffic analytics using PowerShell or an Azure Resource Manager template or client?

You can configure traffic analytics by using Windows PowerShell from version 6.2.1 onwards. To configure flow logging and traffic analytics for a specific NSG by using the Set cmdlet, see [Set-](#)

[AzNetworkWatcherConfigFlowLog](#). To get the flow logging and traffic analytics status for a specific NSG, see [Get-AzNetworkWatcherFlowLogStatus](#).

Currently, you can't use an Azure Resource Manager template to configure traffic analytics.

To configure traffic analytics by using an Azure Resource Manager client, see the following examples.

Set cmdlet example:

```
#Requestbody parameters
$TAtargetUri ="/subscriptions/<NSG subscription id>/resourceGroups/<NSG resource group name>/providers/Microsoft.Network/networkSecurityGroups/<name of NSG>"
$TAstorageId = "/subscriptions/<storage subscription id>/resourcegroups/<storage resource group name>/providers/microsoft.storage/storageaccounts/<storage account name>"
$networkWatcherResourceGroupName = "<network watcher resource group name>"
$networkWatcherName = "<network watcher name>

$requestBody =
@"
{
    'targetResourceId': '${TAtargetUri}',
    'properties':
    {
        'storageId': '${TAstorageId}',
        'enabled': '<true to enable flow log or false to disable flow log>',
        'retentionPolicy' :
        {
            'days: <enter number of days like to retain flow logs in storage account>,
            'enabled: <true to enable retention or false to disable retention>
        }
    },
    'flowAnalyticsConfiguration':
    {
        'networkWatcherFlowAnalyticsConfiguration':
        {
            'enabled':<true to enable traffic analytics or false to disable traffic analytics>
            'workspaceId':'bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbb',
            'workspaceRegion':'<workspace region>',
            'workspaceResourceId':"/subscriptions/<workspace subscription id>/resourcegroups/<workspace resource group name>/providers/microsoft.operationalinsights/workspaces/<workspace name>'"
        }
    }
}
"@
$apiversion = "2016-09-01"

armclient login
armclient post "https://management.azure.com/subscriptions/<NSG subscription id>/resourceGroups/<network watcher resource group name>/providers/Microsoft.Network/networkWatchers/<network watcher name>/configureFlowlog?api-version=${apiversion}" $requestBody
```

Get cmdlet example:

```

#Requestbody parameters
$TAtargetUri ="/subscriptions/<NSG subscription id>/resourceGroups/<NSG resource group
name>/providers/Microsoft.Network/networkSecurityGroups/<NSG name>"


$requestBody =
@"
{
  'targetResourceId': '${TAtargetUri}'
}
“@

armclient login
armclient post "https://management.azure.com/subscriptions/<NSG subscription id>/resourceGroups/<network
watcher resource group name>/providers/Microsoft.Network/networkWatchers/<network watcher
name>/queryFlowLogStatus?api-version=${apiversion}" $requestBody

```

How is Traffic Analytics priced?

Traffic Analytics is metered. The metering is based on processing of flow log data by the service, and storing the resulting enhanced logs in a Log Analytics workspace.

For example, as per the [pricing plan](#), considering West Central US region, if flow logs data stored in a storage account processed by Traffic Analytics is 10 GB and enhanced logs ingested in Log Analytics workspace is 1 GB then the applicable charges are: $10 \times 2.3\$ + 1 \times 2.76\$ = 25.76\$$

How frequently does Traffic Analytics process data?

Refer to the [data aggregation section](#) in Traffic Analytics Schema and Data Aggregation Document

How does Traffic Analytics decide that an IP is malicious?

Traffic Analytics relies on Microsoft internal threat intelligence systems to deem an IP as malicious. These systems leverage diverse telemetry sources like Microsoft products and services, the Microsoft Digital Crimes Unit (DCU), the Microsoft Security Response Center (MSRC), and external feeds and build a lot of intelligence on top of it. Some of this data is Microsoft Internal. If a known IP is getting flagged as malicious, please raise a support ticket to know the details.

How can I set alerts on Traffic Analytics data?

Traffic Analytics does not have inbuilt support for alerts. However, since Traffic Analytics data is stored in Log Analytics you can write custom queries and set alerts on them. Steps :

- You can use the shortlink for Log Analytics in Traffic Analytics.
- Use the [schema documented here](#) to write your queries
- Click "New alert rule" to create the alert
- Refer to [log alerts documentation](#) to create the alert

How do I check which VMs are receiving most on-premise traffic

```

AzureNetworkAnalytics_CL
| where SubType_s == "FlowLog" and FlowType_s == "S2S"
| where <Scoping condition>
| mvexpand vm = pack_array(VM1_s, VM2_s) to typeof(string)
| where isnotempty(vm)
| extend traffic = AllowedInFlows_d + DeniedInFlows_d + AllowedOutFlows_d + DeniedOutFlows_d // For
bytes use: | extend traffic = InboundBytes_d + OutboundBytes_d
| make-series TotalTraffic = sum(traffic) default = 0 on FlowStartTime_t from datetime(<time>) to
datetime(<time>) step 1m by vm
| render timechart

```

For IPs:

```

AzureNetworkAnalytics_CL
| where SubType_s == "FlowLog" and FlowType_s == "S2S"
||| where <Scoping condition>
| mvexpand IP = pack_array(SrcIP_s, DestIP_s) to typeof(string)
| where isnotempty(IP)
| extend traffic = AllowedInFlows_d + DeniedInFlows_d + AllowedOutFlows_d + DeniedOutFlows_d // For
bytes use: | extend traffic = InboundBytes_d + OutboundBytes_d
| make-series TotalTraffic = sum(traffic) default = 0 on FlowStartTime_t from datetime(<time>) to
datetime(<time>) step 1m by IP
| render timechart

```

For time, use format : yyyy-mm-dd 00:00:00

How do I check standard deviation in traffic received by my VMs from on-premise machines

```

AzureNetworkAnalytics_CL
| where SubType_s == "FlowLog" and FlowType_s == "S2S"
||| where <Scoping condition>
| mvexpand vm = pack_array(VM1_s, VM2_s) to typeof(string)
| where isnotempty(vm)
| extend traffic = AllowedInFlows_d + DeniedInFlows_d + AllowedOutFlows_d + DeniedOutFlows_d // For
bytes use: | extend traffic = InboundBytes_d + OutboundBytes_d
| summarize deviation = stdev(traffic) by vm

```

For IPs:

```

AzureNetworkAnalytics_CL
| where SubType_s == "FlowLog" and FlowType_s == "S2S"
||| where <Scoping condition>
| mvexpand IP = pack_array(SrcIP_s, DestIP_s) to typeof(string)
| where isnotempty(IP)
| extend traffic = AllowedInFlows_d + DeniedInFlows_d + AllowedOutFlows_d + DeniedOutFlows_d // For
bytes use: | extend traffic = InboundBytes_d + OutboundBytes_d
| summarize deviation = stdev(traffic) by IP

```

How do I check which ports are reachable (or blocked) between IP pairs with NSG rules

```

AzureNetworkAnalytics_CL
| where SubType_s == "FlowLog" and TimeGenerated between (startTime .. endTime)
| extend sourceIPs = iif(isempty(SrcIP_s), split(SrcPublicIPs_s, " ") , pack_array(SrcIP_s)),
destIPs = iif(isempty(DestIP_s), split(DestPublicIPs_s, " ") , pack_array(DestIP_s))
| mvexpand SourceIp = sourceIPs to typeof(string)
| mvexpand DestIp = destIPs to typeof(string)
| project SourceIp = tostring(split(SourceIp, "|")[0]), DestIp = tostring(split(DestIp, "|")[0]),
NSGList_s, NSGRule_s, DestPort_d, L4Protocol_s, FlowStatus_s
| summarize DestPorts= makeset(DestPort_d) by SourceIp, DestIp, NSGList_s, NSGRule_s, L4Protocol_s,
FlowStatus_s

```

How can I navigate by using the keyboard in the geo map view?

The geo map page contains two main sections:

- **Banner:** The banner at the top of the geo map provides buttons to select traffic distribution filters (for example, Deployment, Traffic from countries/regions, and Malicious). When you select a button, the respective filter is applied on the map. For example, if you select the Active button, the map highlights the active datacenters in your deployment.
- **Map:** Below the banner, the map section shows traffic distribution among Azure datacenters and countries/regions.

Keyboard navigation on the banner

- By default, the selection on the geo map page for the banner is the "Azure DCs" filter.
- To move to another filter, use either the **Tab** or the **Right arrow** key. To move backward, use either the **Shift+Tab** or the **Left arrow** key. Forward navigation is left to right, followed by top to bottom.
- Press **Enter** or the **Down** arrow key to apply the selected filter. Based on filter selection and deployment, one or multiple nodes under the map section are highlighted.
- To switch between banner and map, press **ctrl+F6**.

Keyboard navigation on the map

- After you have selected any filter on the banner and pressed **ctrl+F6**, focus moves to one of the highlighted nodes (**Azure datacenter** or **Country/Region**) in the map view.
- To move to other highlighted nodes in the map, use either **Tab** or the **Right arrow** key for forward movement. Use **Shift+Tab** or the **Left arrow** key for backward movement.
- To select any highlighted node in the map, use the **Enter** or **Down arrow** key.
- On selection of any such nodes, focus moves to the **Information Tool Box** for the node. By default, focus moves to the closed button on the **Information Tool Box**. To further move inside the **Box** view, use **Right arrow** and **Left arrow** keys to move forward and backward, respectively. Pressing **Enter** has same effect as selecting the focused button in the **Information Tool Box**.
- When you press **Tab** while the focus is on the **Information Tool Box**, the focus moves to the end points in the same continent as the selected node. Use the **Right arrow** and **Left arrow** keys to move through these endpoints.
- To move to other flow endpoints or continent clusters, use **Tab** for forward movement and **Shift+Tab** for backward movement.
- When the focus is on **Continent clusters**, use the **Enter** or **Down** arrow keys to highlight the endpoints inside the continent cluster. To move through endpoints and the close button on the information box of the continent cluster, use either the **Right arrow** or **Left arrow** key for forward and backward movement, respectively. On any endpoint, you can use **Shift+L** to switch to the connection line from the selected node to the endpoint. You can press **Shift+L** again to move to the selected endpoint.

Keyboard navigation at any stage

- `Esc` collapses the expanded selection.
- The `Up arrow` key performs the same action as `Esc`. The `Down arrow` key performs the same action as `Enter`.
- Use `Shift+Plus` to zoom in, and `Shift-Minus` to zoom out.

How can I navigate by using the keyboard in the virtual network topology view?

The virtual networks topology page contains two main sections:

- **Banner:** The banner at the top of the virtual networks topology provides buttons to select traffic distribution filters (for example, Connected virtual networks, Disconnected virtual networks, and Public IPs). When you select a button, the respective filter is applied on the topology. For example, if you select the Active button, the topology highlights the active virtual networks in your deployment.
- **Topology:** Below the banner, the topology section shows traffic distribution among virtual networks.

Keyboard navigation on the banner

- By default, the selection on the virtual networks topology page for the banner is the "Connected VNets" filter.
- To move to another filter, use the `Tab` key to move forward. To move backward, use the `Shift+Tab` key. Forward navigation is left to right, followed by top to bottom.
- Press `Enter` to apply the selected filter. Based on the filter selection and deployment, one or multiple nodes (virtual network) under the topology section are highlighted.
- To switch between the banner and the topology, press `Ctrl+F6`.

Keyboard navigation on the topology

- After you have selected any filter on the banner and pressed `Ctrl+F6`, focus moves to one of the highlighted nodes (**VNet**) in the topology view.
- To move to other highlighted nodes in the topology view, use the `Shift+Right arrow` key for forward movement.
- On highlighted nodes, focus moves to the **Information Tool Box** for the node. By default, focus moves to the **More details** button on the **Information Tool Box**. To further move inside the **Box** view, use the `Right arrow` and `Left arrow` keys to move forward and backward, respectively. Pressing `Enter` has same effect as selecting the focused button in the **Information Tool Box**.
- On selection of any such nodes, you can visit all its connections, one by one, by pressing the `Shift+Left arrow` key. Focus moves to the **Information Tool Box** of that connection. At any point, the focus can be shifted back to the node by pressing `Shift+Right arrow` again.

How can I navigate by using the keyboard in the subnet topology view?

The virtual subnetworks topology page contains two main sections:

- **Banner:** The banner at the top of the virtual subnetworks topology provides buttons to select traffic distribution filters (for example, Active, Medium, and Gateway subnets). When you select a button, the respective filter is applied on the topology. For example, if you select the Active button, the topology highlights the active virtual subnetwork in your deployment.
- **Topology:** Below the banner, the topology section shows traffic distribution among virtual subnetworks.

Keyboard navigation on the banner

- By default, the selection on the virtual subnetworks topology page for the banner is the "Subnets" filter.
- To move to another filter, use the `Tab` key to move forward. To move backward, use the `Shift+Tab` key. Forward navigation is left to right, followed by top to bottom.

- Press `Enter` to apply the selected filter. Based on filter selection and deployment, one or multiple nodes (Subnet) under the topology section are highlighted.
- To switch between the banner and the topology, press `Ctrl+F6`.

Keyboard navigation on the topology

- After you have selected any filter on the banner and pressed `Ctrl+F6`, focus moves to one of the highlighted nodes (**Subnet**) in the topology view.
- To move to other highlighted nodes in the topology view, use the `Shift+Right arrow` key for forward movement.
- On highlighted nodes, focus moves to the **Information Tool Box** for the node. By default, focus moves to the **More details** button on the **Information Tool Box**. To further move inside the **Box** view, use `Right arrow` and `Left arrow` keys to move forward and backward, respectively. Pressing `Enter` has same effect as selecting the focused button in the **Information Tool Box**.
- On selection of any such nodes, you can visit all its connections, one by one, by pressing `Shift+Left arrow` key. Focus moves to the **Information Tool Box** of that connection. At any point, the focus can be shifted back to the node by pressing `Shift+Right arrow` again.

Schema and data aggregation in Traffic Analytics

12/1/2019 • 11 minutes to read • [Edit Online](#)

Traffic Analytics is a cloud-based solution that provides visibility into user and application activity in cloud networks. Traffic Analytics analyzes Network Watcher network security group (NSG) flow logs to provide insights into traffic flow in your Azure cloud. With traffic analytics, you can:

- Visualize network activity across your Azure subscriptions and identify hot spots.
- Identify security threats to, and secure your network, with information such as open-ports, applications attempting internet access, and virtual machines (VM) connecting to rogue networks.
- Understand traffic flow patterns across Azure regions and the internet to optimize your network deployment for performance and capacity.
- Pinpoint network misconfigurations leading to failed connections in your network.
- Know network usage in bytes, packets, or flows.

Data aggregation

1. All flow logs at an NSG between "FlowIntervalStartTime_t" and "FlowIntervalEndTime_t" are captured at one-minute intervals in the storage account as blobs before being processed by Traffic Analytics.
2. Default processing interval of Traffic Analytics is 60 minutes. This means that every 60 mins Traffic Analytics picks blobs from storage for aggregation. If processing interval chosen is 10 mins, Traffic Analytics will pick blobs from storage account after every 10 mins.
3. Flows that have the same Source IP, Destination IP, Destination port, NSG name, NSG rule, Flow Direction, and Transport layer protocol (TCP or UDP) (Note: Source port is excluded for aggregation) are clubbed into a single flow by Traffic Analytics
4. This single record is decorated (Details in the section below) and ingested in Log Analytics by Traffic Analytics. This process can take upto 1 hour max.
5. FlowStartTime_t field indicates the first occurrence of such an aggregated flow (same four-tuple) in the flow log processing interval between "FlowIntervalStartTime_t" and "FlowIntervalEndTime_t".
6. For any resource in TA, the flows indicated in the UI are total flows seen by the NSG, but in Log Analytics user will see only the single, reduced record. To see all the flows, use the blob_id field, which can be referenced from Storage. The total flow count for that record will match the individual flows seen in the blob.

The below query helps you looks at all flow logs from on-premises in the last 30 days.

```
AzureNetworkAnalytics_CL  
| where SubType_s == "FlowLog" and FlowStartTime_t >= ago(30d) and FlowType_s == "ExternalPublic"  
| project Subnet_s
```

To view the blob path for the flows in the above mentioned query, use the query below:

```

let TableWithBlobId =
(AzureNetworkAnalytics_CL
| where SubType_s == "Topology" and ResourceType == "NetworkSecurityGroup" and DiscoveryRegion_s ==
Region_s and IsFlowEnabled_b
| extend binTime = bin(TimeProcessed_t, 6h),
    nsgId = strcat(Subscription_g, "/", Name_s),
    saNameSplit = split(FlowLogStorageAccount_s, "/")
| extend saName = iif(arraylength(saNameSplit) == 3, saNameSplit[2], '')
| distinct nsgId, saName, binTime)
| join kind = rightouter (
    AzureNetworkAnalytics_CL
    | where SubType_s == "FlowLog"
    | extend binTime = bin(FlowEndTime_t, 6h)
) on binTime, $left.nsgId == $right.NSGList_s
| extend blobTime = format_datetime(todatetime(FlowIntervalStartTime_t), "yyyy MM dd hh")
| extend nsgComponents = split(toupper(NSGList_s), "/"), dateTimeComponents = split(blobTime, " ")
| extend BlobPath = strcat("https://", saName,
    "@insights-logs-networksecuritygroupflowevent/resoureId=/SUBSCRIPTIONS/",
    nsgComponents[0],
    "/RESOURCEGROUPS/", nsgComponents[1],
    "/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/", nsgComponents[2],
    "/y=", dateTimeComponents[0], "/m=", dateTimeComponents[1], "/d=",
    dateTimeComponents[2], "/h=", dateTimeComponents[3],
    "/m=00/macAddress=", replace(@"-", "", MACAddress_s),
    "/PT1H.json")
| project-away nsgId, saName, binTime, blobTime, nsgComponents, dateTimeComponents;

TableWithBlobId
| where SubType_s == "FlowLog" and FlowStartTime_t >= ago(30d) and FlowType_s == "ExternalPublic"
| project Subnet_s , BlobPath

```

The above query constructs a URL to access the blob directly. The URL with place-holders is below:

```

https://{saName}@insights-logs-
networksecuritygroupflowevent/resoureId=/SUBSCRIPTIONS/{subscriptionId}/RESOURCEGROUPS/{resourceGroup}/PROVIDE
RS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/{nsgName}/y={year}/m={month}/d={day}/h={hour}/m=00/macAddress=
{macAddress}/PT1H.json

```

Fields used in Traffic Analytics schema

IMPORTANT

The Traffic Analytics Schema has been updated on 22nd August, 2019. The new schema provides source and destination IPs separately removing need to parse FlowDirection field making queries simpler.

FASchemaVersion_s updated from 1 to 2.

Deprecated fields: VMIP_s, Subscription_s, Region_s, NSGRules_s, Subnet_s, VM_s, NIC_s, PublicIPs_s, FlowCount_d

New fields: SrcPublicIPs_s, DestPublicIPs_s, NSGRule_s

Deprecated fields will be available until 22nd November, 2019.

Traffic Analytics is built on top of Log Analytics, so you can run custom queries on data decorated by Traffic Analytics and set alerts on the same.

Listed below are the fields in the schema and what they signify

FIELD	FORMAT	COMMENTS
TableName	AzureNetworkAnalytics_CL	Table for Traffic Analytics data

FIELD	FORMAT	COMMENTS
SubType_s	FlowLog	Subtype for the flow logs. Use only "FlowLog", other values of SubType_s are for internal workings of the product
FASchemaVersion_s	2	Schema version. Does not reflect NSG Flow Log version
TimeProcessed_t	Date and Time in UTC	Time at which the Traffic Analytics processed the raw flow logs from the storage account
FlowIntervalStartTime_t	Date and Time in UTC	Starting time of the flow log processing interval. This is time from which flow interval is measured
FlowIntervalEndTime_t	Date and Time in UTC	Ending time of the flow log processing interval
FlowStartTime_t	Date and Time in UTC	First occurrence of the flow (which will get aggregated) in the flow log processing interval between "FlowIntervalStartTime_t" and "FlowIntervalEndTime_t". This flow gets aggregated based on aggregation logic
FlowEndTime_t	Date and Time in UTC	Last occurrence of the flow (which will get aggregated) in the flow log processing interval between "FlowIntervalStartTime_t" and "FlowIntervalEndTime_t". In terms of flow log v2, this field contains the time when the last flow with the same four-tuple started (marked as "B" in the raw flow record)
FlowType_s	<ul style="list-style-type: none"> * IntraVNet * InterVNet * S2S * P2S * AzurePublic * ExternalPublic * MaliciousFlow * Unknown Private * Unknown 	Definition in notes below the table
SrcIP_s	Source IP address	Will be blank in case of AzurePublic and ExternalPublic flows
DestIP_s	Destination IP address	Will be blank in case of AzurePublic and ExternalPublic flows
VMIP_s	IP of the VM	Used for AzurePublic and ExternalPublic flows
PublicIP_s	Public IP addresses	Used for AzurePublic and ExternalPublic flows

FIELD	FORMAT	COMMENTS
DestPort_d	Destination Port	Port at which traffic is incoming
L4Protocol_s	* T * U	Transport Protocol. T = TCP U = UDP
L7Protocol_s	Protocol Name	Derived from destination port
FlowDirection_s	* I = Inbound * O = Outbound	Direction of the flow in/out of NSG as per flow log
FlowStatus_s	* A = Allowed by NSG Rule * D = Denied by NSG Rule	Status of flow allowed/nblocked by NSG as per flow log
NSGList_s	<SUBSCRIPTIONID>/<RESOURCEGROUP_NAME>/<NSG_NAME>	Network Security Group (NSG) associated with the flow
NSGRules_s	<Index value 0> <NSG_RULENAME> <Flow Direction> <Flow Status> <FlowCount ProcessedByRule>	NSG rule that allowed or denied this flow
NSGRule_s	NSG_RULENAME	NSG rule that allowed or denied this flow
NSGRuleType_s	* User Defined * Default	The type of NSG Rule used by the flow
MACAddress_s	MAC Address	MAC address of the NIC at which the flow was captured
Subscription_s	Subscription of the Azure virtual network/ network interface/ virtual machine is populated in this field	Applicable only for FlowType = S2S, P2S, AzurePublic, ExternalPublic, MaliciousFlow, and UnknownPrivate flow types (flow types where only one side is azure)
Subscription1_s	Subscription ID	Subscription ID of virtual network/ network interface/ virtual machine to which the source IP in the flow belongs to
Subscription2_s	Subscription ID	Subscription ID of virtual network/ network interface/ virtual machine to which the destination IP in the flow belongs to
Region_s	Azure region of virtual network/ network interface/ virtual machine to which the IP in the flow belongs to	Applicable only for FlowType = S2S, P2S, AzurePublic, ExternalPublic, MaliciousFlow, and UnknownPrivate flow types (flow types where only one side is azure)
Region1_s	Azure Region	Azure region of virtual network/ network interface/ virtual machine to which the source IP in the flow belongs to

FIELD	FORMAT	COMMENTS
Region2_s	Azure Region	Azure region of virtual network to which the destination IP in the flow belongs to
NIC_s	<resourcegroup_Name>/<NetworkInterfaceName>	NIC associated with the VM sending or receiving the traffic
NIC1_s	<resourcegroup_Name>/<NetworkInterfaceName>	NIC associated with the source IP in the flow
NIC2_s	<resourcegroup_Name>/<NetworkInterfaceName>	NIC associated with the destination IP in the flow
VM_s	<resourcegroup_Name>/<NetworkInterfaceName>	Virtual Machine associated with the Network interface NIC_s
VM1_s	<resourcegroup_Name>/<VirtualMachineName>	Virtual Machine associated with the source IP in the flow
VM2_s	<resourcegroup_Name>/<VirtualMachineName>	Virtual Machine associated with the destination IP in the flow
Subnet_s	<ResourceGroup_Name>/<VNET_Name>/<SubnetName>	Subnet associated with the NIC_s
Subnet1_s	<ResourceGroup_Name>/<VNET_Name>/<SubnetName>	Subnet associated with the Source IP in the flow
Subnet2_s	<ResourceGroup_Name>/<VNET_Name>/<SubnetName>	Subnet associated with the Destination IP in the flow
ApplicationGateway1_s	<SubscriptionID>/<ResourceGroupName>/<ApplicationGatewayName>	Application gateway associated with the Source IP in the flow
ApplicationGateway2_s	<SubscriptionID>/<ResourceGroupName>/<ApplicationGatewayName>	Application gateway associated with the Destination IP in the flow
LoadBalancer1_s	<SubscriptionID>/<ResourceGroupName>/<LoadBalancerName>	Load balancer associated with the Source IP in the flow
LoadBalancer2_s	<SubscriptionID>/<ResourceGroupName>/<LoadBalancerName>	Load balancer associated with the Destination IP in the flow
LocalNetworkGateway1_s	<SubscriptionID>/<ResourceGroupName>/<LocalNetworkGatewayName>	Local network gateway associated with the Source IP in the flow
LocalNetworkGateway2_s	<SubscriptionID>/<ResourceGroupName>/<LocalNetworkGatewayName>	Local network gateway associated with the Destination IP in the flow
ConnectionType_s	Possible values are VNetPeering, VpnGateway, and ExpressRoute	Connection Type
ConnectionName_s	<SubscriptionID>/<ResourceGroupName>/<ConnectionName>	Connection Name. For flowtype P2S, this will be formatted as _

FIELD	FORMAT	COMMENTS
ConnectingVNets_s	Space separated list of virtual network names	In case of hub and spoke topology, hub virtual networks will be populated here
Country_s	Two letter country code (ISO 3166-1 alpha-2)	Populated for flow type ExternalPublic. All IP addresses in PublicIPs_s field will share the same country code
AzureRegion_s	Azure region locations	Populated for flow type AzurePublic. All IP addresses in PublicIPs_s field will share the Azure region
AllowedInFlows_d		Count of inbound flows that were allowed. This represents the number of flows that shared the same four-tuple inbound to the network interface at which the flow was captured
DeniedInFlows_d		Count of inbound flows that were denied. (Inbound to the network interface at which the flow was captured)
AllowedOutFlows_d		Count of outbound flows that were allowed (Outbound to the network interface at which the flow was captured)
DeniedOutFlows_d		Count of outbound flows that were denied (Outbound to the network interface at which the flow was captured)
FlowCount_d	Deprecated. Total flows that matched the same four-tuple. In case of flow types ExternalPublic and AzurePublic, count will include the flows from various PublicIP addresses as well.	
InboundPackets_d	Packets received as captured at the network interface where NSG rule was applied	This is populated only for the Version 2 of NSG flow log schema
OutboundPackets_d	Packets sent as captured at the network interface where NSG rule was applied	This is populated only for the Version 2 of NSG flow log schema
InboundBytes_d	Bytes received as captured at the network interface where NSG rule was applied	This is populated only for the Version 2 of NSG flow log schema
OutboundBytes_d	Bytes sent as captured at the network interface where NSG rule was applied	This is populated only for the Version 2 of NSG flow log schema
CompletedFlows_d		This is populated with non-zero value only for the Version 2 of NSG flow log schema

FIELD	FORMAT	COMMENTS
PublicIPs_s	<PUBLIC_IP> <FLOW_STARTED_COUNT> <FLOW_ENDED_COUNT> <OUTBOUND_PACKETS> <INBOUND_PACKETS> <OUTBOUND_BYTES> <INBOUND_BYTES>	Entries separated by bars
SrcPublicIPs_s	<SOURCE_PUBLIC_IP> <FLOW_STARTED_COUNT> <FLOW_ENDED_COUNT> <OUTBOUND_PACKETS> <INBOUND_PACKETS> <OUTBOUND_BYTES> <INBOUND_BYTES>	Entries separated by bars
DestPublicIPs_s	<DESTINATION_PUBLIC_IP> <FLOW_STARTED_COUNT> <FLOW_ENDED_COUNT> <OUTBOUND_PACKETS> <INBOUND_PACKETS> <OUTBOUND_BYTES> <INBOUND_BYTES>	Entries separated by bars

Notes

1. In case of AzurePublic and ExternalPublic flows, the customer owned Azure VM IP is populated in VMIP_s field, while the Public IP addresses are being populated in the PublicIPs_s field. For these two flow types, we should use VMIP_s and PublicIPs_s instead of SrcIP_s and DestIP_s fields. For AzurePublic and ExternalPublicIP addresses, we aggregate further, so that the number of records ingested to customer log analytics workspace is minimal.(This field will be deprecated soon and we should be using SrcIP_ and DestIP_s depending on whether azure VM was the source or the destination in the flow)
2. Details for flow types: Based on the IP addresses involved in the flow, we categorize the flows in to the following flow types:
 3. IntraVNet – Both the IP addresses in the flow reside in the same Azure Virtual Network.
 4. InterVNet - IP addresses in the flow reside in the two different Azure Virtual Networks.
 5. S2S – (Site To Site) One of the IP addresses belongs to Azure Virtual Network while the other IP address belongs to customer network (Site) connected to the Azure Virtual Network through VPN gateway or Express Route.
 6. P2S - (Point To Site) One of the IP addresses belongs to Azure Virtual Network while the other IP address belongs to customer network (Site) connected to the Azure Virtual Network through VPN gateway.
 7. AzurePublic - One of the IP addresses belongs to Azure Virtual Network while the other IP address belongs to Azure Internal Public IP addresses owned by Microsoft. Customer owned Public IP addresses won't be part of this flow type. For instance, any customer owned VM sending traffic to an Azure Service (Storage endpoint) would be categorized under this flow type.
 8. ExternalPublic - One of the IP addresses belongs to Azure Virtual Network while the other IP address is a public IP that is not in Azure, is not reported as malicious in the ASC feeds that Traffic Analytics consumes for the processing interval between "FlowIntervalStartTime_t" and "FlowIntervalEndTime_t".
 9. MaliciousFlow - One of the IP addresses belong to azure virtual network while the other IP address is a public IP that is not in Azure and is reported as malicious in the ASC feeds that Traffic Analytics consumes for the processing interval between "FlowIntervalStartTime_t" and "FlowIntervalEndTime_t".
 10. UnknownPrivate - One of the IP addresses belong to Azure Virtual Network while the other IP address belongs to private IP range as defined in RFC 1918 and could not be mapped by Traffic Analytics to a customer

owned site or Azure Virtual Network.

11. Unknown – Unable to map the either of the IP addresses in the flows with the customer topology in Azure as well as on-premises (site).
12. Some field names are appended with _s or _d. These do NOT signify source and destination but indicate the data types string and decimal respectively.

Next Steps

To get answers to frequently asked questions, see [Traffic analytics FAQ](#) To see details about functionality, see [Traffic analytics documentation](#)

Visualizing Network Security Group flow logs with Power BI

1/28/2020 • 5 minutes to read • [Edit Online](#)

Network Security Group flow logs allow you to view information about ingress and egress IP traffic on Network Security Groups. These flow logs show outbound and inbound flows on a per rule basis, the NIC the flow applies to, 5-tuple information about the flow (Source/Destination IP, Source/Destination Port, Protocol), and if the traffic was allowed or denied.

It can be difficult to gain insights into flow logging data by manually searching the log files. In this article, we provide a solution to visualize your most recent flow logs and learn about traffic on your network.

WARNING

The following steps work with flow logs version 1. For details, see [Introduction to flow logging for network security groups](#).
The following instructions will not work with version 2 of the log files, without modification.

Scenario

In the following scenario, we connect Power BI desktop to the storage account we have configured as the sink for our NSG Flow Logging data. After we connect to our storage account, Power BI downloads and parses the logs to provide a visual representation of the traffic that is logged by Network Security groups.

Using the visuals supplied in the template you can examine:

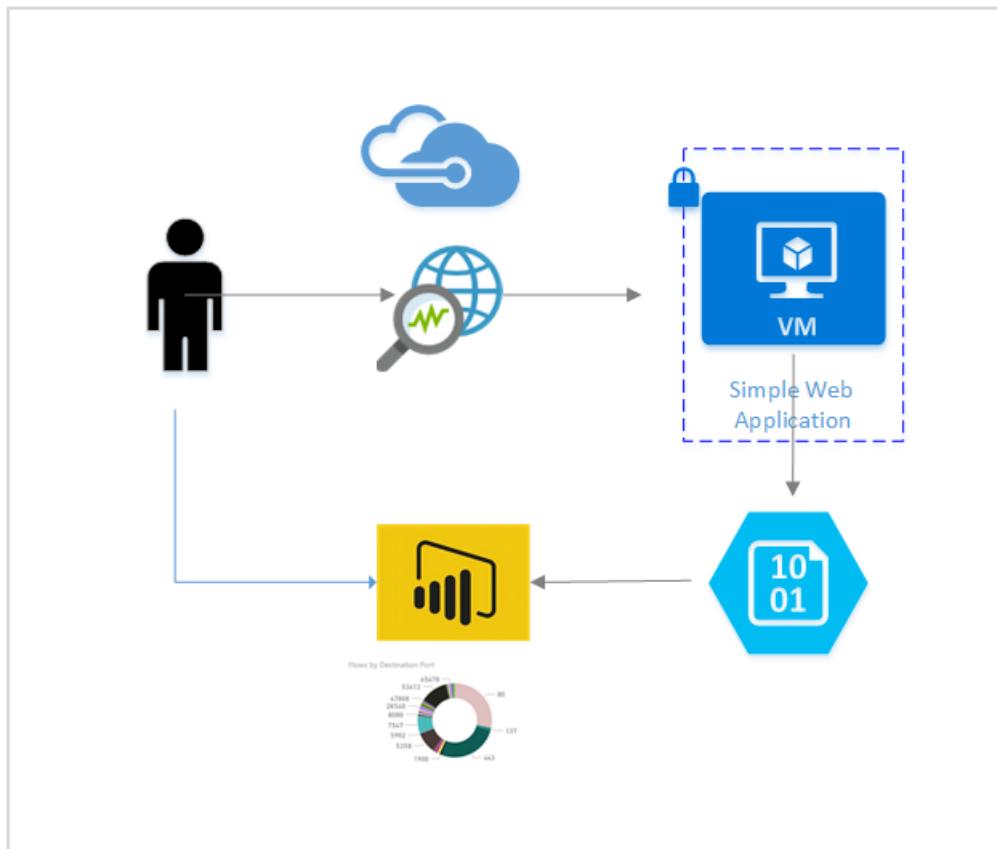
- Top Talkers
- Time Series Flow Data by direction and rule decision
- Flows by Network Interface MAC address
- Flows by NSG and Rule
- Flows by Destination Port

The template provided is editable so you can modify it to add new data, visuals, or edit queries to suit your needs.

Setup

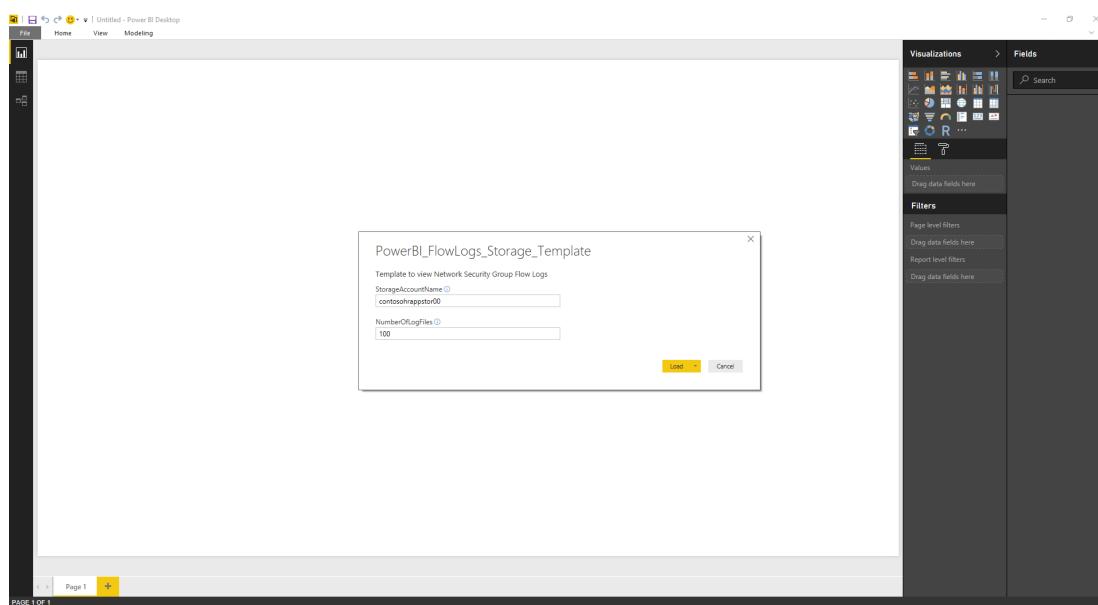
Before you begin, you must have Network Security Group Flow Logging enabled on one or many Network Security Groups in your account. For instructions on enabling Network Security flow logs, refer to the following article: [Introduction to flow logging for Network Security Groups](#).

You must also have the Power BI Desktop client installed on your machine, and enough free space on your machine to download and load the log data that exists in your storage account.



Steps

1. Download and open the following Power BI template in the Power BI Desktop Application [Network Watcher PowerBI flow logs template](#)
2. Enter the required Query parameters
 - a. **StorageAccountName** – Specifies to the name of the storage account containing the NSG flow logs that you would like to load and visualize.
 - b. **NumberOfLogFile**s – Specifies the number of log files that you would like to download and visualize in Power BI. For example, if 50 is specified, the 50 latest log files. If we have 2 NSGs enabled and configured to send NSG flow logs to this account, then the past 25 hours of logs can be viewed.



3. Enter the Access Key for your storage account. You can find valid access keys by navigating to your storage account in the Azure portal and selecting **Access Keys** from the Settings menu. Click **Connect**

then apply changes.

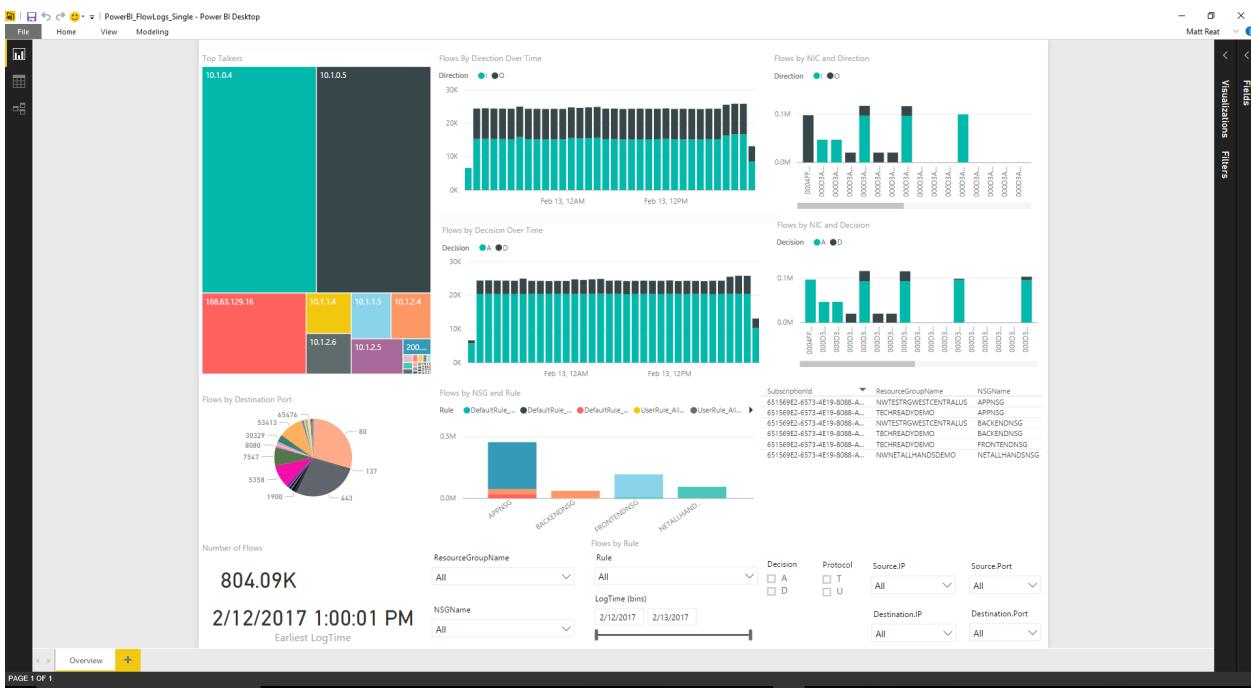
The screenshot shows the 'Access keys' section of the Azure Storage account settings. On the left, a sidebar lists navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and a Settings group containing Access keys (selected), Configuration, Shared access signature, Properties, Locks, and Automation script. The main content area displays the storage account name 'contosohappstor00'. A table lists two access keys: 'key1' with the value 'fTDR0BsFnd4nAk+u+rWbhPfmCyr+EsfrTHC...' and 'key2' with the value '3rlk8gYGGU3ZQ/0MI6GGkRibFQ3EKTW0rW...'. A note at the top states: 'Use access keys to authenticate your applications when making requests to this Azure Storage account. You are provided two access keys so that you can maintain connections using different keys.' Below this, it says: 'When you regenerate your access keys, you must update any Azure resources and applications that use them.'

The screenshot shows Power BI Desktop with a connection dialog open. The dialog is titled 'Microsoft Azure Blob Storage' and shows the URL 'https://contosohappstor00.blob.core.windows.net/'. It has fields for 'Account key' (with a redacted value) and 'NSGName'. The 'Connected' button is highlighted in yellow. The background shows several Power BI visualizations, including line charts for 'Flows by direction over time' and 'Flows by NIC and decision', and a table for 'Flows by rule'. The Power BI ribbon is visible at the top, and the Fields pane is open on the right.

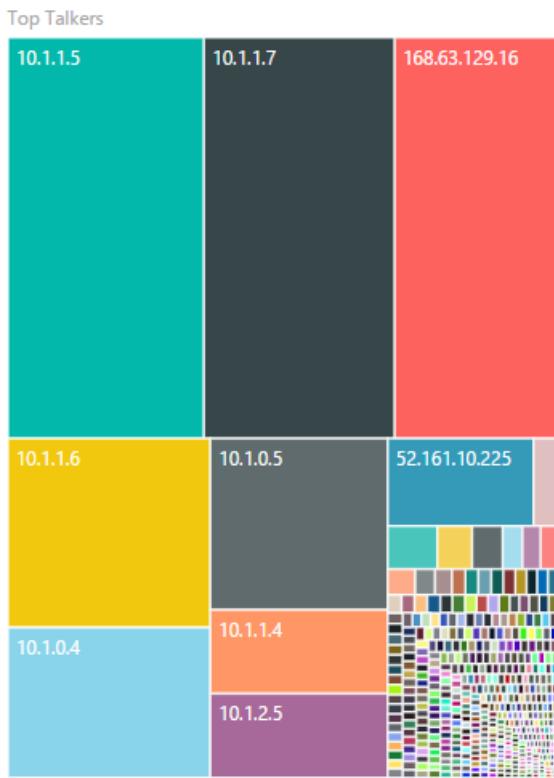
4. Your logs are download and parsed and you can now utilize the pre-created visuals.

Understanding the visuals

Provided in the template are a set of visuals that help make sense of the NSG Flow Log data. The following images show a sample of what the dashboard looks like when populated with data. Below we examine each visual in greater detail



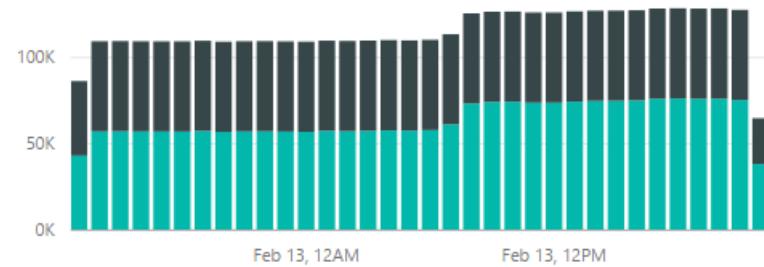
The Top Talkers visual shows the IPs that have initiated the most connections over the period specified. The size of the boxes corresponds to the relative number of connections.



The following time series graphs show the number of flows over the period. The upper graph is segmented by the flow direction, and the lower is segmented by the decision made (allow or deny). With this visual, you can examine your traffic trends over time, and spot any abnormal spikes or decline in traffic or traffic segmentation.

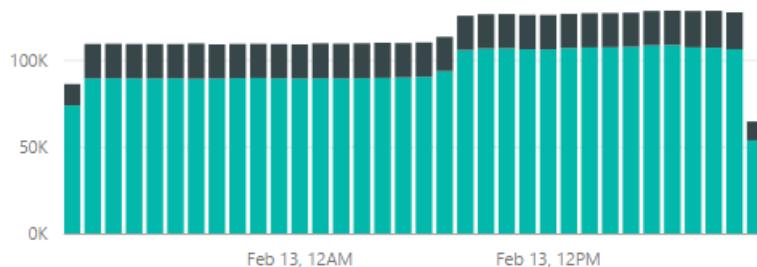
Flows By Direction Over Time

Direction ● I ● O



Flows by Decision Over Time

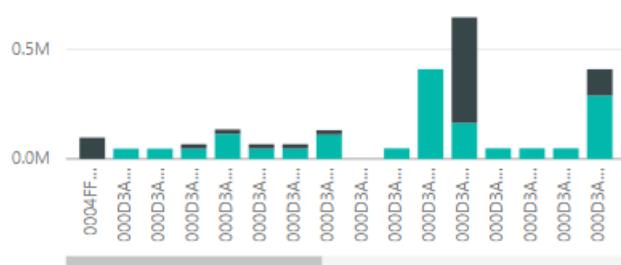
Decision ● A ● D



The following graphs show the flows per Network interface, with the upper segmented by flow direction and the lower segmented by decision made. With this information, you can gain insights into which of your VMs communicated the most relative to others, and if traffic to a specific VM is being allowed or denied.

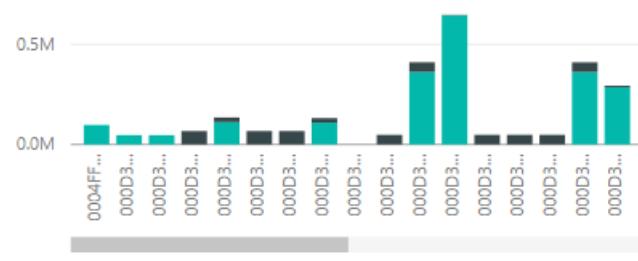
Flows by NIC and Direction

Direction ● I ● O



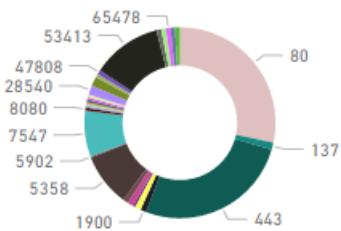
Flows by NIC and Decision

Decision ● A ● D



The following donut wheel chart shows a breakdown of Flows by Destination Port. With this information, you can view the most commonly used destination ports used within the specified period.

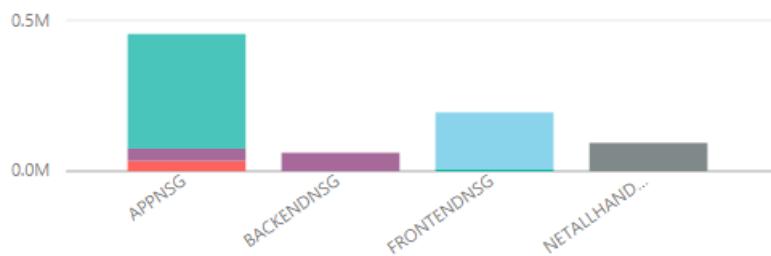
Flows by Destination Port



The following bar chart shows the Flow by NSG and Rule. With this information, you can see the NSGs responsible for the most traffic, and the breakdown of traffic on an NSG by rule.

Flows by NSG and Rule

Rule ▲ UserRule_All... ● UserRule_All... ○ UserRule_A... ■ UserRule_D... ▼ UserRule_D...



The following informational charts display information about the NSGs present in the logs, the number of Flows captured over the period, and the date of the earliest log captured. This information gives you an idea of what NSGs are being logged and the date range of flows.

SubscriptionId	ResourceGroupName	NSGName
651569E2-6573-4E19-8088-A...	NWTESTRGWESTCENTRALUS	APPNSG
651569E2-6573-4E19-8088-A...	TECHREADYDEMO	APPNSG
651569E2-6573-4E19-8088-A...	NWTESTRGWESTCENTRALUS	BACKENDNSG
651569E2-6573-4E19-8088-A...	TECHREADYDEMO	BACKENDNSG
651569E2-6573-4E19-8088-A...	TECHREADYDEMO	FRONTENDNSG
651569E2-6573-4E19-8088-A...	NWNETALLHANDSDEMO	NETALLHANDNSNG

Number of Flows

804.09K

2/12/2017 1:00:01 PM

Earliest LogTime

This template includes the following slicers to allow you to view only the data you are most interested in. You can filter on your resource groups, NSGs, and rules. You can also filter on 5-tuple information, decision, and the time the log was written.

Flows by Rule

ResourceGroupName	Rule	Decision	Protocol	Source.IP	Source.Port
All	All	A D	T U	All	All
NSGName	LogTime (bins)	Destination.IP	Destination.Port		
All	2/12/2017 2/13/2017	All	All		

Conclusion

We showed in this scenario that by using Network Security Group Flow logs provided by Network Watcher and Power BI, we are able to visualize and understand the traffic. Using the provided template, Power BI downloads

the logs directly from storage and processes them locally. Time taken to load the template varies depending on the number of files requested and total size of files downloaded.

Feel free to customize this template for your needs. There are many numerous ways that you can use Power BI with Network Security Group Flow Logs.

Notes

- Logs by default are stored in
`https://{{storageAccountName}}.blob.core.windows.net/insights-logs-networksecuritygroupflowevent/`
 - If other data exists in another directory they the queries to pull and process the data must be modified.
- The provided template is not recommended for use with more than 1 GB of logs.
- If you have a large amount of logs, we recommend that you investigate a solution using another data store like Data Lake or SQL server.

Next Steps

Learn how to visualize your NSG flow logs with the Elastic Stack by visiting [Visualize Azure Network Watcher NSG flow logs using open source tools](#)

Visualize Azure Network Watcher NSG flow logs using open source tools

1/28/2020 • 6 minutes to read • [Edit Online](#)

Network Security Group flow logs provide information that can be used understand ingress and egress IP traffic on Network Security Groups. These flow logs show outbound and inbound flows on a per rule basis, the NIC the flow applies to, 5 tuple information about the flow (Source/Destination IP, Source/Destination Port, Protocol), and if the traffic was allowed or denied.

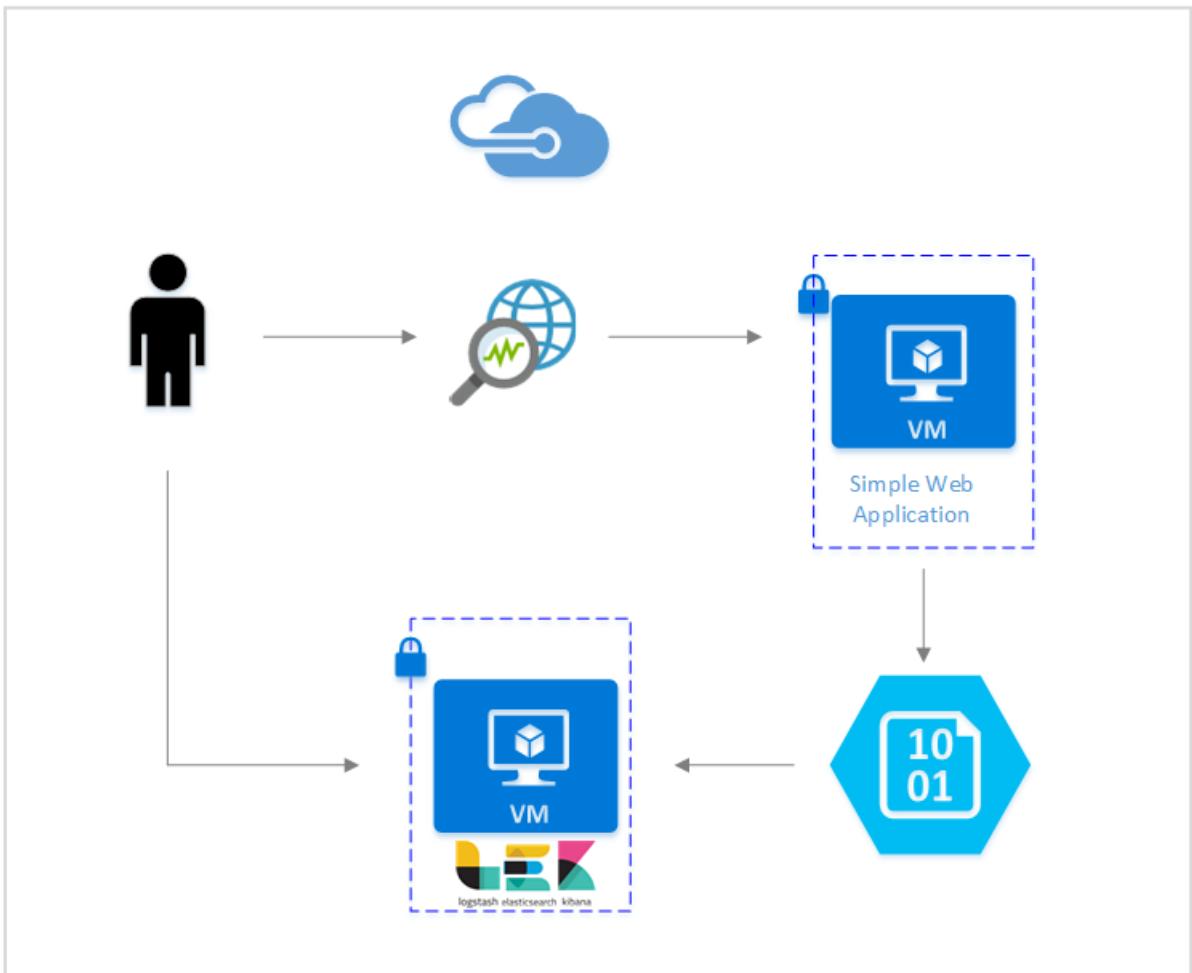
These flow logs can be difficult to manually parse and gain insights from. However, there are several open source tools that can help visualize this data. This article will provide a solution to visualize these logs using the Elastic Stack, which will allow you to quickly index and visualize your flow logs on a Kibana dashboard.

WARNING

The following steps work with flow logs version 1. For details, see [Introduction to flow logging for network security groups](#). The following instructions will not work with version 2 of the log files, without modification.

Scenario

In this article, we will set up a solution that will allow you to visualize Network Security Group flow logs using the Elastic Stack. A Logstash input plugin will obtain the flow logs directly from the storage blob configured for containing the flow logs. Then, using the Elastic Stack, the flow logs will be indexed and used to create a Kibana dashboard to visualize the information.



Steps

Enable Network Security Group flow logging

For this scenario, you must have Network Security Group Flow Logging enabled on at least one Network Security Group in your account. For instructions on enabling Network Security Flow Logs, refer to the following article [Introduction to flow logging for Network Security Groups](#).

Set up the Elastic Stack

By connecting NSG flow logs with the Elastic Stack, we can create a Kibana dashboard what allows us to search, graph, analyze, and derive insights from our logs.

Install Elasticsearch

1. The Elastic Stack from version 5.0 and above requires Java 8. Run the command `java -version` to check your version. If you do not have java installed, refer to documentation on the [Azure-supported JDKs](#).
2. Download the correct binary package for your system:

```
curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.2.0.deb
sudo dpkg -i elasticsearch-5.2.0.deb
sudo /etc/init.d/elasticsearch start
```

Other installation methods can be found at [Elasticsearch Installation](#)

3. Verify that Elasticsearch is running with the command:

```
curl http://127.0.0.1:9200
```

You should see a response similar to this:

```
{  
  "name" : "Angela Del Toro",  
  "cluster_name" : "elasticsearch",  
  "version" : {  
    "number" : "5.2.0",  
    "build_hash" : "8ff36d139e16f8720f2947ef62c8167a888992fe",  
    "build_timestamp" : "2016-01-27T13:32:39Z",  
    "build_snapshot" : false,  
    "lucene_version" : "6.1.0"  
  },  
  "tagline" : "You Know, for Search"  
}
```

For further instructions on installing Elastic search, refer to [Installation instructions](#).

Install Logstash

1. To install Logstash run the following commands:

```
curl -L -O https://artifacts.elastic.co/downloads/logstash/logstash-5.2.0.deb  
sudo dpkg -i logstash-5.2.0.deb
```

2. Next we need to configure Logstash to access and parse the flow logs. Create a logstash.conf file using:

```
sudo touch /etc/logstash/conf.d/logstash.conf
```

3. Add the following content to the file:

```

input {
  azureblob
  {
    storage_account_name => "mystorageaccount"
    storage_access_key => "VGhpcyBpcyBhIGZha2Uga2V5Lg=="
    container => "insights-logs-networksecuritygroupflowevent"
    codec => "json"
    # Refer https://docs.microsoft.com/azure/network-watcher/network-watcher-read-nsg-flow-logs
    # Typical numbers could be 21/9 or 12/2 depends on the nsg log file types
    file_head_bytes => 12
    file_tail_bytes => 2
    # Enable / tweak these settings when event is too big for codec to handle.
    # break_json_down_policy => "with_head_tail"
    # break_json_batch_count => 2
  }
}

filter {
  split { field => "[records]" }
  split { field => "[records][properties][flows]"}
  split { field => "[records][properties][flows][flows]"}
  split { field => "[records][properties][flows][flows][flowTuples]"}

  mutate{
    split => { "[records][resourceId]" => "/"}
    add_field => {"Subscription" => "%{[records][resourceId][2]}"
                  "ResourceGroup" => "%{[records][resourceId][4]}"
                  "NetworkSecurityGroup" => "%{[records][resourceId][8]}"}
    convert => {"Subscription" => "string"}
    convert => {"ResourceGroup" => "string"}
    convert => {"NetworkSecurityGroup" => "string"}
    split => { "[records][properties][flows][flows][flowTuples]" => ","}
    add_field => {
      "unixtimestamp" => "%{[records][properties][flows][flows][flowTuples][0]}"
      "srcIp" => "%{[records][properties][flows][flows][flowTuples][1]}"
      "destIp" => "%{[records][properties][flows][flows][flowTuples][2]}"
      "srcPort" => "%{[records][properties][flows][flows][flowTuples][3]}"
      "destPort" => "%{[records][properties][flows][flows][flowTuples][4]}"
      "protocol" => "%{[records][properties][flows][flows][flowTuples][5]}"
      "trafficflow" => "%{[records][properties][flows][flows][flowTuples][6]}"
      "traffic" => "%{[records][properties][flows][flows][flowTuples][7]}"
    }
    convert => {"unixtimestamp" => "integer"}
    convert => {"srcPort" => "integer"}
    convert => {"destPort" => "integer"}
  }
}

date{
  match => ["unixtimestamp" , "UNIX"]
}

output {
  stdout { codec => rubydebug }
  elasticsearch {
    hosts => "localhost"
    index => "nsg-flow-logs"
  }
}

```

For further instructions on installing Logstash, refer to the [official documentation](#).

Install the Logstash input plugin for Azure blob storage

This Logstash plugin will allow you to directly access the flow logs from their designated storage account. To install this plugin, from the default Logstash installation directory (in this case /usr/share/logstash/bin) run the command:

```
logstash-plugin install logstash-input-azureblob
```

To start Logstash run the command:

```
sudo /etc/init.d/logstash start
```

For more information about this plugin, refer to the [documentation](#).

Install Kibana

1. Run the following commands to install Kibana:

```
curl -L -O https://artifacts.elastic.co/downloads/kibana/kibana-5.2.0-linux-x86_64.tar.gz  
tar xzvf kibana-5.2.0-linux-x86_64.tar.gz
```

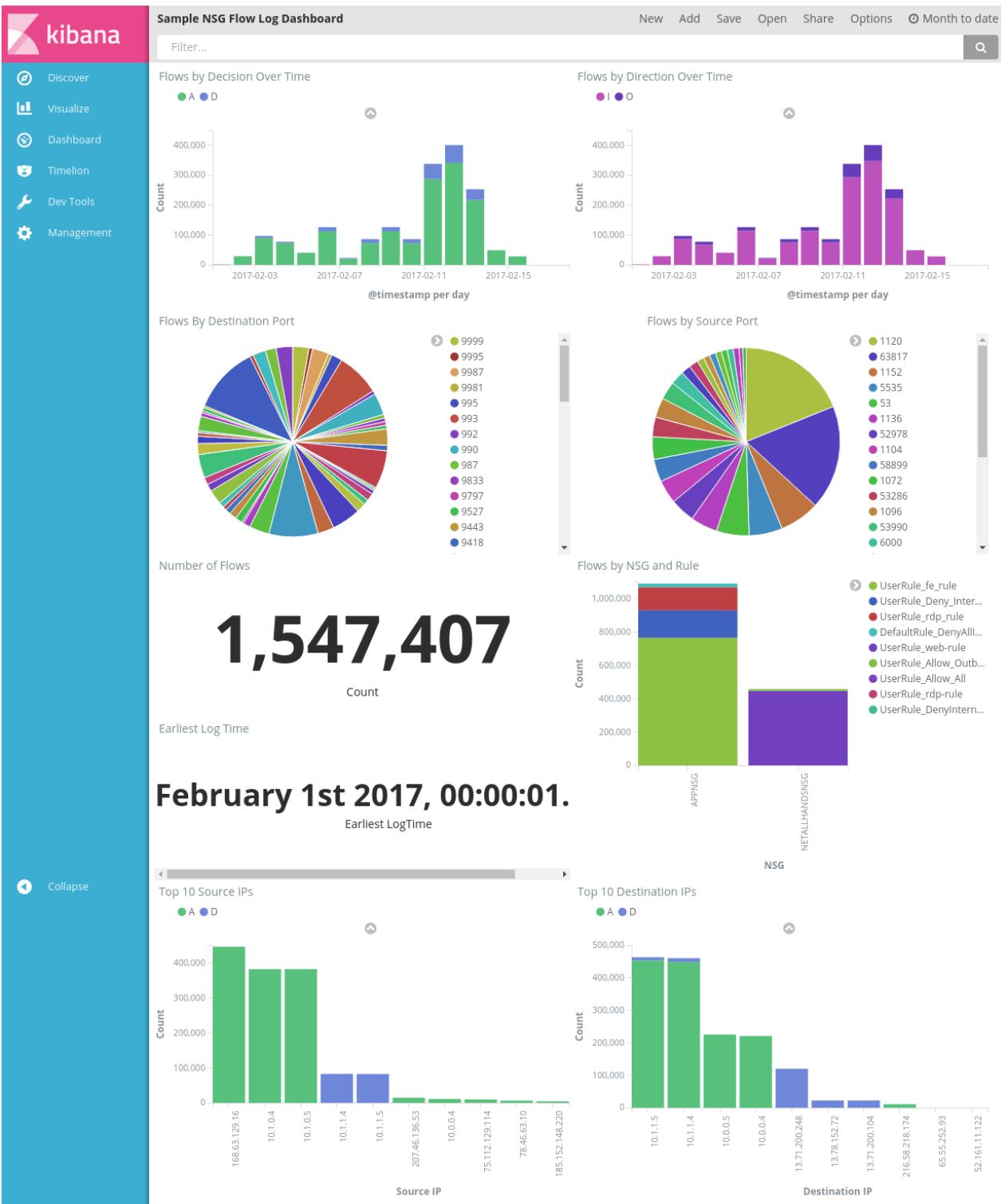
2. To run Kibana use the commands:

```
cd kibana-5.2.0-linux-x86_64/  
.bin/kibana
```

3. To view your Kibana web interface, navigate to <http://localhost:5601>
4. For this scenario, the index pattern used for the flow logs is "nsg-flow-logs". You may change the index pattern in the "output" section of your logstash.conf file.
5. If you want to view the Kibana dashboard remotely, create an inbound NSG rule allowing access to **port 5601**.

Create a Kibana dashboard

A sample dashboard to view trends and details in your alerts is shown in the following picture:



Download the [dashboard file](#), the [visualization file](#), and the [saved search file](#).

Under the **Management** tab of Kibana, navigate to **Saved Objects** and import all three files. Then from the **Dashboard** tab you can open and load the sample dashboard.

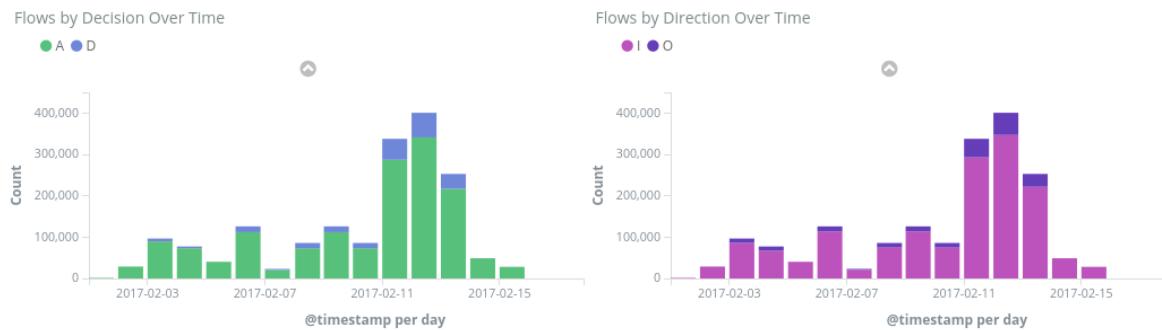
You can also create your own visualizations and dashboards tailored towards metrics of your own interest. Read more about creating Kibana visualizations from Kibana's [official documentation](#).

Visualize NSG flow logs

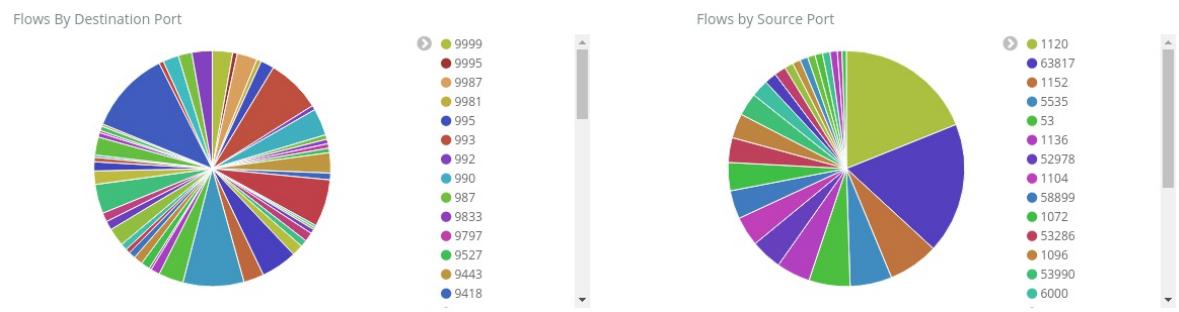
The sample dashboard provides several visualizations of the flow logs:

1. Flows by Decision/Direction Over Time - time series graphs showing the number of flows over the time period. You can edit the unit of time and span of both these visualizations. Flows by Decision shows the proportion of allow or deny decisions made, while Flows by Direction shows the proportion of inbound and outbound traffic. With these visuals you can examine traffic trends over time and look for any spikes

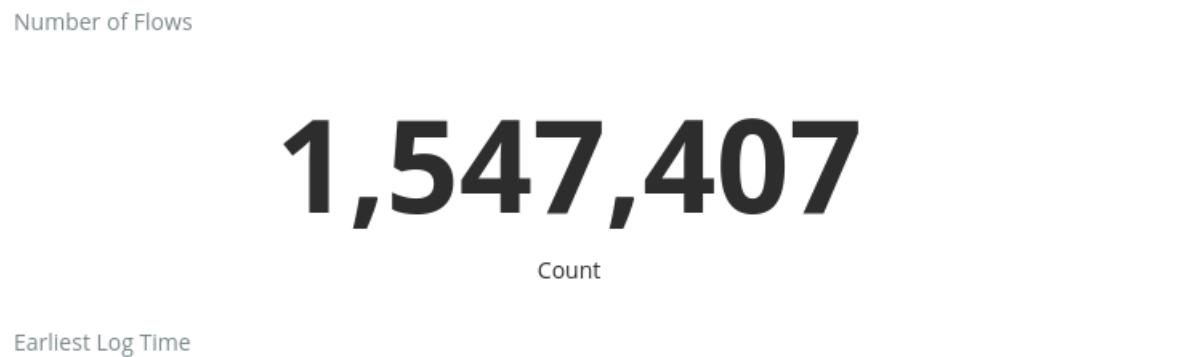
or unusual patterns.



2. Flows by Destination/Source Port – pie charts showing the breakdown of flows to their respective ports. With this view you can see your most commonly used ports. If you click on a specific port within the pie chart, the rest of the dashboard will filter down to flows of that port.



3. Number of Flows and Earliest Log Time – metrics showing you the number of flows recorded and the date of the earliest log captured.

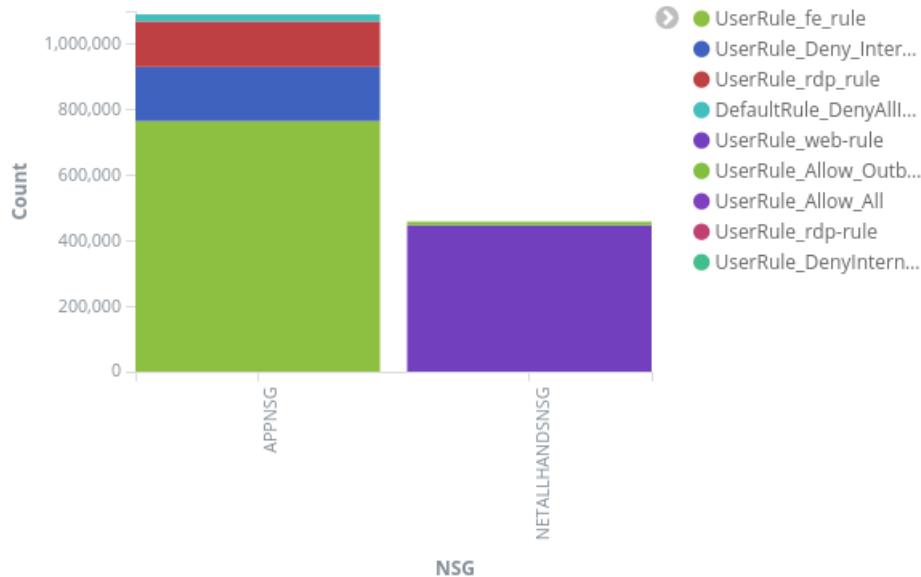


February 1st 2017, 00:00:01.000

Earliest LogTime

4. Flows by NSG and Rule – a bar graph showing you the distribution of flows within each NSG, as well as the distribution of rules within each NSG. From here you can see which NSG and rules generated the most traffic.

Flows by NSG and Rule



5. Top 10 Source/Destination IPs – bar charts showing the top 10 source and destination IPs. You can adjust these charts to show more or less top IPs. From here you can see the most commonly occurring IPs as well as the traffic decision (allow or deny) being made towards each IP.



6. Flow Tuples – this table shows you the information contained within each flow tuple, as well as its corresponding NGS and rule.

Flow Tuples									
Time	NetworkSecurityGroup	records.properties.flows.rule	srcip	destip	srcPort	destPort	protocol	trafficflow	traffic
			1	2	3	4	5	...10	>
▶ February 15th 2017, 09:22:35.000	NETALLHANDSNSG	UserRule_DenyInternetInbound	71.6.158.166	52.161.15.40	18639	1200	T	I	D
▶ February 15th 2017, 07:53:10.000	NETALLHANDSNSG	UserRule_DenyInternetInbound	71.6.146.130	52.161.15.40	13944	1177	T	I	D
▶ February 15th 2017, 06:44:35.000	NETALLHANDSNSG	UserRule_Allow_Outbound_All	10.0.0.5	65.55.252.190	55788	443	T	O	A
▶ February 15th 2017, 06:37:35.000	NETALLHANDSNSG	UserRule_Allow_Outbound_All	10.0.0.4	65.55.252.190	50021	443	T	O	A
▶ February 15th 2017, 06:26:28.000	NETALLHANDSNSG	UserRule_Allow_Outbound_All	10.0.0.5	66.119.144.158	55781	443	T	O	A
▶ February 15th 2017, 06:26:24.000	NETALLHANDSNSG	UserRule_Allow_Outbound_All	10.0.0.5	157.55.240.89	55779	443	T	O	A
▶ February 15th 2017, 04:46:41.000	NETALLHANDSNSG	UserRule_DenyInternetInbound	120.132.3.187	52.161.15.40	63410	1099	T	I	D
▶ February 15th 2017, 00:00:48.000	NETALLHANDSNSG	UserRule_DenyInternetInbound	89.248.167.13	52.161.15.40	58022	1311	T	I	D

Using the query bar at the top of the dashboard, you can filter down the dashboard based on any parameter of the flows, such as subscription ID, resource groups, rule, or any other variable of interest. For more about Kibana's queries and filters, refer to the [official documentation](#)

Conclusion

By combining the Network Security Group flow logs with the Elastic Stack, we have come up with powerful and

customizable way to visualize our network traffic. These dashboards allow you to quickly gain and share insights about your network traffic, as well as filter down and investigate on any potential anomalies. Using Kibana, you can tailor these dashboards and create specific visualizations to meet any security, audit, and compliance needs.

Next steps

Learn how to visualize your NSG flow logs with Power BI by visiting [Visualize NSG flows logs with Power BI](#)

Manage and analyze Network Security Group flow logs using Network Watcher and Grafana

1/28/2020 • 5 minutes to read • [Edit Online](#)

Network Security Group (NSG) flow logs provide information that can be used to understand ingress and egress IP traffic on network interfaces. These flow logs show outbound and inbound flows on a per NSG rule basis, the NIC the flow applies to, 5-tuple information about the flow (Source/Destination IP, Source/Destination Port, Protocol), and if the traffic was allowed or denied.

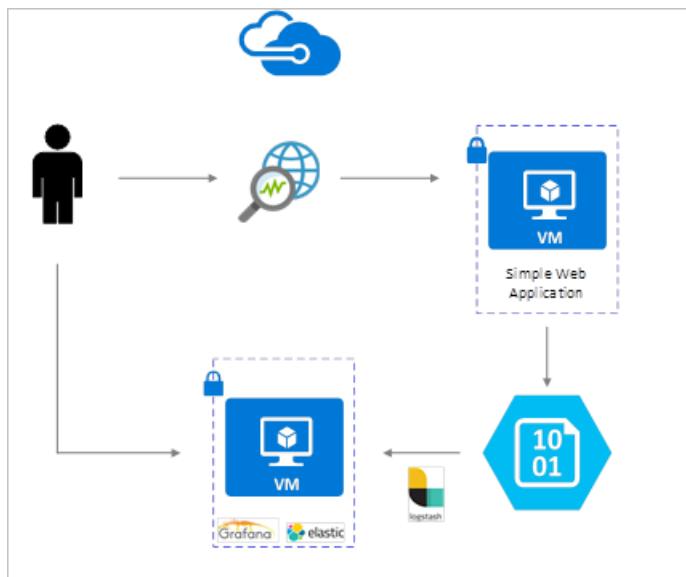
WARNING

The following steps work with flow logs version 1. For details, see [Introduction to flow logging for network security groups](#). The following instructions will not work with version 2 of the log files, without modification.

You can have many NSGs in your network with flow logging enabled. This amount of logging data makes it cumbersome to parse and gain insights from your logs. This article provides a solution to centrally manage these NSG flow logs using Grafana, an open source graphing tool, ElasticSearch, a distributed search and analytics engine, and Logstash, which is an open source server-side data processing pipeline.

Scenario

NSG flow logs are enabled using Network Watcher and are stored in Azure blob storage. A Logstash plugin is used to connect and process flow logs from blob storage and send them to ElasticSearch. Once the flow logs are stored in ElasticSearch, they can be analyzed and visualized into customized dashboards in Grafana.



Installation steps

Enable Network Security Group flow logging

For this scenario, you must have Network Security Group Flow Logging enabled on at least one Network Security Group in your account. For instructions on enabling Network Security Flow Logs, refer to the following article [Introduction to flow logging for Network Security Groups](#).

Setup considerations

In this example Grafana, ElasticSearch, and Logstash are configured on an Ubuntu 16.04 LTS Server deployed in Azure. This minimal setup is used for running all three components – they are all running on the same VM. This setup should only be used for testing and non-critical workloads. Logstash, Elasticsearch, and Grafana can all be architected to scale independently across many instances. For more information, see the documentation for each of these components.

Install Logstash

You use Logstash to flatten the JSON formatted flow logs to a flow tuple level.

1. To install Logstash, run the following commands:

```
curl -L -O https://artifacts.elastic.co/downloads/logstash/logstash-5.2.0.deb  
sudo dpkg -i logstash-5.2.0.deb
```

2. Configure Logstash to parse the flow logs and send them to ElasticSearch. Create a Logstash.conf file using:

```
sudo touch /etc/logstash/conf.d/logstash.conf
```

3. Add the following content to the file. Change the storage account name and access key to reflect your storage account details:

```
input {  
    azureblob  
    {  
        storage_account_name => "mystorageaccount"  
        storage_access_key => "VGhpcyBpcyBhIGZha2Uga2V5Lg=="  
        container => "insights-logs-networksecuritygroupflowevent"  
        codec => "json"  
        # Refer https://docs.microsoft.com/azure/network-watcher/network-watcher-read-nsg-flow-logs  
        # Typical numbers could be 21/9 or 12/2 depends on the nsg log file types  
        file_head_bytes => 12  
        file_tail_bytes => 2  
        # Enable / tweak these settings when event is too big for codec to handle.  
        # break_json_down_policy => "with_head_tail"  
        # break_json_batch_count => 2  
    }  
}  
filter {  
    split { field => "[records]" }  
    split { field => "[records][properties][flows]" }  
    split { field => "[records][properties][flows][flows]" }  
    split { field => "[records][properties][flows][flows][flowTuples]" }  
  
    mutate {  
        split => { "[records][resourceId]" => "/" }  
        add_field => { "Subscription" => "%{[records][resourceId][2]}"  
                      "ResourceGroup" => "%{[records][resourceId][4]}"  
                      "NetworkSecurityGroup" => "%{[records][resourceId][8]}"  
        }  
        convert => {"Subscription" => "string"}  
        convert => {"ResourceGroup" => "string"}  
        convert => {"NetworkSecurityGroup" => "string"}  
        split => { "[records][properties][flows][flows][flowTuples]" => "," }  
        add_field => {  
            "unixtimestamp" => "%{[records][properties][flows][flows][flowTuples][0]}"  
            "srcIp" => "%{[records][properties][flows][flows][flowTuples][1]}"  
            "destIp" => "%{[records][properties][flows][flows][flowTuples][2]}"  
            "srcPort" => "%{[records][properties][flows][flows][flowTuples][3]}"  
            "destPort" => "%{[records][properties][flows][flows][flowTuples][4]}"  
            "protocol" => "%{[records][properties][flows][flows][flowTuples][5]}"  
            "trafficflow" => "%{[records][properties][flows][flows][flowTuples][6]}"  
            "traffic" => "%{[records][properties][flows][flows][flowTuples][7]}"  
        }  
    }  
}
```

```

    }
    add_field => {
      "time" => "%{[records][time]}"
      "systemId" => "%{[records][systemId]}"
      "category" => "%{[records][category]}"
      "resourceId" => "%{[records][resourceId]}"
      "operationName" => "%{[records][operationName]}"
      "Version" => "%{[records][properties][Version]}"
      "rule" => "%{[records][properties][flows][rule]}"
      "mac" => "%{[records][properties][flows][flows][mac]}"
    }
    convert => {"unixtimestamp" => "integer"}
    convert => {"srcPort" => "integer"}
    convert => {"destPort" => "integer"}
    add_field => { "message" => "%{Message}" }
  }

  date {
    match => ["unixtimestamp" , "UNIX"]
  }
}

output {
  stdout { codec => rubydebug }
  elasticsearch {
    hosts => "localhost"
    index => "nsg-flow-logs"
  }
}

```

The Logstash config file provided is composed of three parts: the input, filter, and output. The input section designates the input source of the logs that Logstash will process – in this case we are going to use an “azureblob” input plugin (installed in the next steps) that will allow us to access the NSG flow log JSON files stored in blob storage.

The filter section then flattens each flow log file so that each individual flow tuple and its associated properties becomes a separate Logstash event.

Finally, the output section forwards each Logstash event to the ElasticSearch server. Feel free to modify the Logstash config file to suit your specific needs.

Install the Logstash input plugin for Azure Blob storage

This Logstash plugin enables you to directly access the flow logs from their designated blob storage account. To install this plug in, from the default Logstash installation directory (in this case /usr/share/logstash/bin) run the command:

```

cd /usr/share/logstash/bin
sudo ./logstash-plugin install logstash-input-azureblob

```

For more information about this plug in, see [Logstash input plugin for Azure Storage Blobs](#).

Install ElasticSearch

You can use the following script to install ElasticSearch. For information about installing ElasticSearch, see [Elastic Stack](#).

```
apt-get install apt-transport-https openjdk-8-jre-headless uuid-runtime pwgen -y
wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | apt-key add -
echo "deb https://packages.elastic.co/elasticsearch/5.x/debian stable main" | tee -a
/etc/apt/sources.list.d/elasticsearch-5.x.list
apt-get update && apt-get install elasticsearch
sed -i s/#cluster.name:.*$/cluster.name:\ grafana/ /etc/elasticsearch/elasticsearch.yml
systemctl daemon-reload
systemctl enable elasticsearch.service
systemctl start elasticsearch.service
```

Install Grafana

To install and run Grafana, run the following commands:

```
wget https://s3-us-west-2.amazonaws.com/grafana-releases/release/grafana_4.5.1_amd64.deb
sudo apt-get install -y adduser libfontconfig
sudo dpkg -i grafana_4.5.1_amd64.deb
sudo service grafana-server start
```

For additional installation information, see [Installing on Debian / Ubuntu](#).

Add the ElasticSearch server as a data source

Next, you need to add the ElasticSearch index containing flow logs as a data source. You can add a data source by selecting **Add data source** and completing the form with the relevant information. A sample of this configuration can be found in the following screenshot:

The screenshot shows the 'Add data source' configuration dialog in Grafana. The 'Type' is set to 'Elasticsearch'. The 'Name' is 'nsg-flow-logs', and the 'Default' checkbox is checked. Under 'Http settings', the 'Url' is 'http://localhost:9200' and 'Access' is 'proxy'. In the 'Http Auth' section, 'Basic Auth' is selected. Under 'Elasticsearch details', the 'Index name' is 'nsg-flow-logs', 'Time field name' is '@timestamp', 'Version' is '5.x', and 'Min interval' is '10s'. At the bottom, there are 'Add' and 'Cancel' buttons.

Name	nsg-flow-logs	Default
Type	Elasticsearch	

Http settings	
Url	http://localhost:9200
Access	proxy

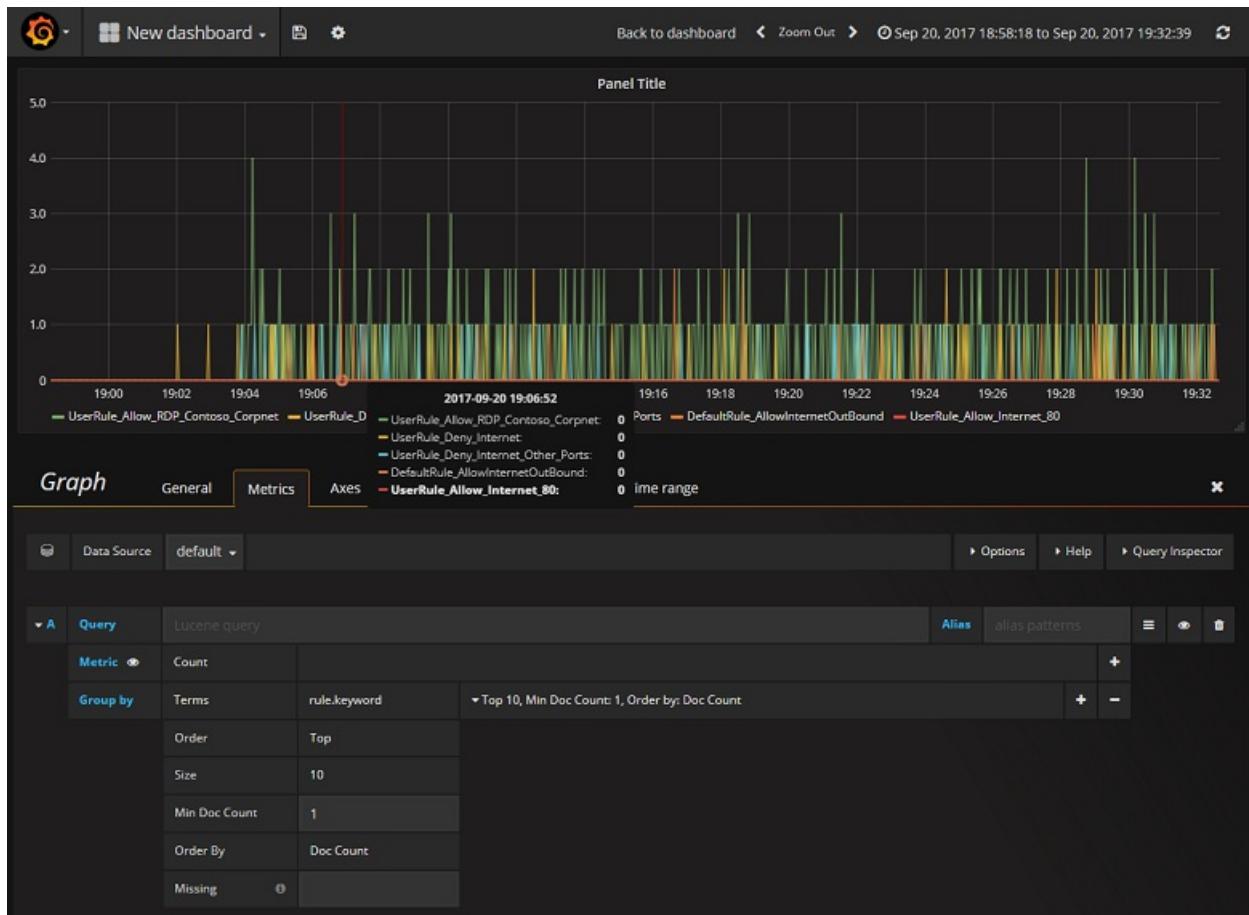
Http Auth	
Basic Auth	<input type="checkbox"/>
TLS Client Auth	<input type="checkbox"/>

Elasticsearch details	
Index name	nsg-flow-logs
Time field name	@timestamp
Version	5.x
Min interval	10s

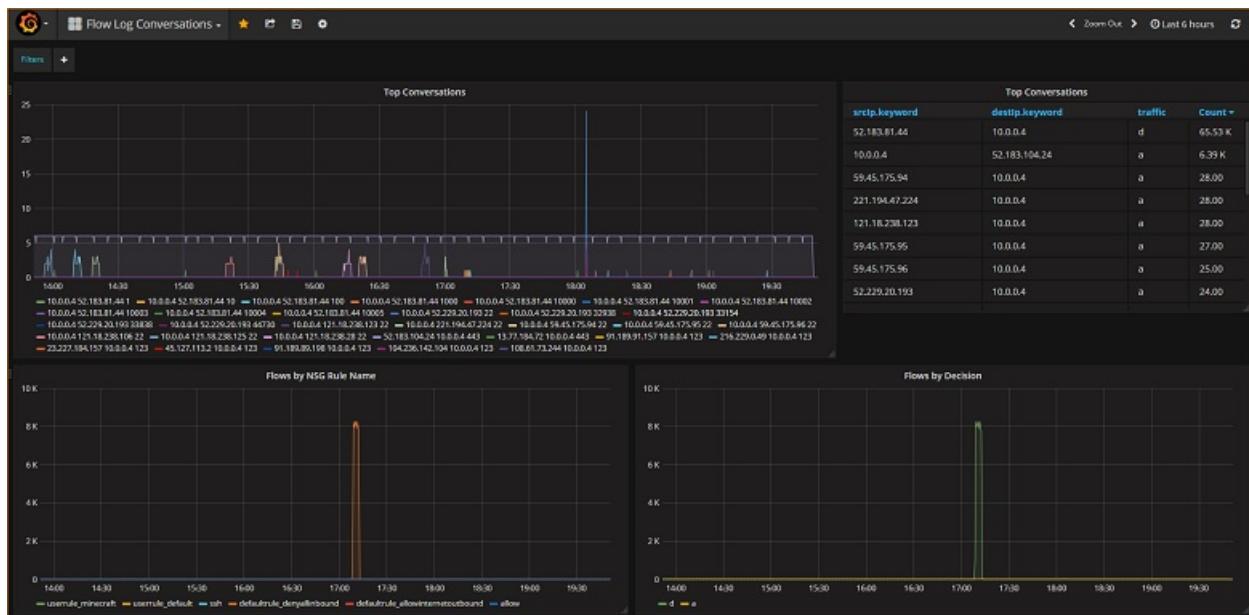
Add Cancel

[Create a dashboard](#)

Now that you have successfully configured Grafana to read from the ElasticSearch index containing NSG flow logs, you can create and personalize dashboards. To create a new dashboard, select **Create your first dashboard**. The following sample graph configuration shows flows segmented by NSG rule:



The following screenshot depicts a graph and chart showing the top flows and their frequency. Flows are also shown by NSG rule and flows by decision. Grafana is highly customizable so it's advisable that you create dashboards to suit your specific monitoring needs. The following example shows a typical dashboard:



Conclusion

By integrating Network Watcher with ElasticSearch and Grafana, you now have a convenient and centralized way to manage and visualize NSG flow logs as well as other data. Grafana has a number of other powerful graphing features that can also be used to further manage flow logs and better understand your network traffic. Now that

you have a Grafana instance set up and connected to Azure, feel free to continue to explore the other functionality that it offers.

Next steps

- Learn more about using [Network Watcher](#).

Manage and analyze network security group flow logs in Azure using Network Watcher and Graylog

1/28/2020 • 8 minutes to read • [Edit Online](#)

Network security group flow logs provide information that you can use to understand ingress and egress IP traffic for Azure network interfaces. Flow logs show outbound and inbound flows on a per network security group rule basis, the network interface the flow applies to, 5-tuple information (Source/Destination IP, Source/Destination Port, Protocol) about the flow, and if the traffic was allowed or denied.

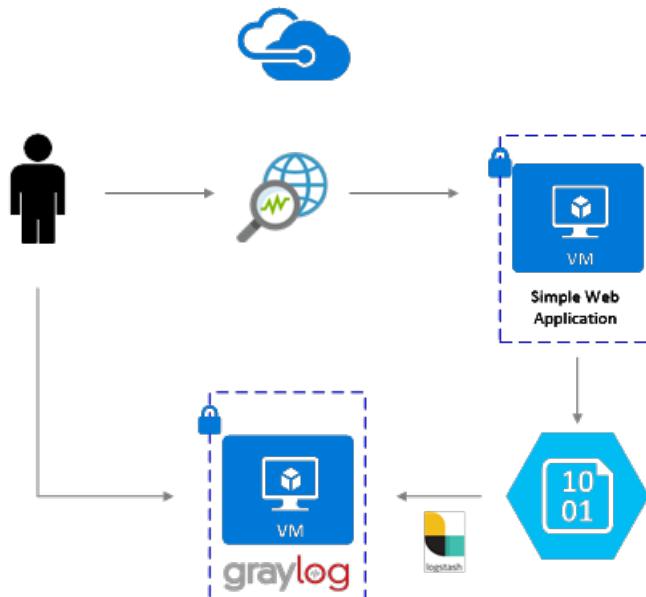
You can have many network security groups in your network with flow logging enabled. Several network security groups with flow logging enabled can make it cumbersome to parse and gain insights from your logs. This article provides a solution to centrally manage these network security group flow logs using Graylog, an open source log management and analysis tool, and Logstash, an open source server-side data processing pipeline.

WARNING

The following steps work with flow logs version 1. For details, see [Introduction to flow logging for network security groups](#). The following instructions will not work with version 2 of the log files, without modification.

Scenario

Network security group flow logs are enabled using Network Watcher. Flow logs flow in to Azure blob storage. A Logstash plugin is used to connect and process flow logs from blob storage and send them to Graylog. Once the flow logs are stored in Graylog, they can be analyzed and visualized into customized dashboards.



Installation Steps

Enable network security group flow logging

For this scenario, you must have network security group flow logging enabled on at least one network security group in your account. For instructions on enabling network security group flow logs, refer to the following article [Introduction to flow logging for network security groups](#).

Setting up Graylog

In this example, both Graylog and Logstash are configured on an Ubuntu 14.04 Server, deployed in Azure.

- Refer to the [documentation](#) from Graylog, for step by step instructions on how install onto Ubuntu.
- Make sure to also configure the Graylog web interface by following the [documentation](#).

This example uses the minimum Graylog setup (i.e a single instance of a Graylog), but Graylog can be architected to scale across resources depending on your system and production needs. For more information on architectural considerations or a deep architectural guide, see Graylog's [documentation](#) and [architectural guide](#).

Graylog can be installed in many ways, depending on your platform and preferences. For a full list of possible installation methods, refer to Graylog's official [documentation](#). The Graylog server application runs on Linux distributions and has the following prerequisites:

- Java SE 8 or later – [Azul Azure JDK documentation](#)
- Elastic Search 2.x (2.1.0 or later) - [Elasticsearch installation documentation](#)
- MongoDB 2.4 or later – [MongoDB installation documentation](#)

Install Logstash

Logstash is used to flatten the JSON formatted flow logs to a flow tuple level. Flattening the flow logs makes the logs easier to organize and search in Graylog.

1. To install Logstash, run the following commands:

```
curl -L -O https://artifacts.elastic.co/downloads/logstash/logstash-5.2.0.deb  
sudo dpkg -i logstash-5.2.0.deb
```

2. Configure Logstash to parse the flow logs and send them to Graylog. Create a Logstash.conf file:

```
sudo touch /etc/logstash/conf.d/logstash.conf
```

3. Add the following content to the file. Change the highlighted values to reflect your storage account details:

```
input {  
    azureblob  
    {  
        storage_account_name => "mystorageaccount"  
        storage_access_key =>  
"NrUZmx7pJSKaRJzvQbeiZWi5nBRWOTr7Wwr9DrvK7YtDBrADYxT1y0oEExtSlkDnGRt7qcRiZZEBCCyRYND8SxSt"  
        container => "insights-logs-networksecuritygroupflowevent"  
        registry_create_policy => "start_over"  
        codec => "json"  
        file_head_bytes => 21  
        file_tail_bytes => 9  
        # Possible options: `do_not_break`, `with_head_tail`, `without_head_tail`  
        break_json_down_policy => 'with_head_tail'  
        break_json_batch_count => 2  
        interval => 5  
    }  
}  
  
filter {  
    split { field => "[records]" }  
    split { field => "[records][properties][flows]" }  
    split { field => "[records][properties][flows][flows]" }  
    split { field => "[records][properties][flows][flows][flowTuples]" }  
}  
  
mutate {  
    split => { "[records][resourceId]" => "/" }
```

```

    add_field =>{
        "Subscription" => "%{[records][resourceId][2]}"
        "ResourceGroup" => "%{[records][resourceId][4]}"
        "NetworkSecurityGroup" => "%{[records][resourceId][8]}"
    }
    convert => {"Subscription" => "string"}
    convert => {"ResourceGroup" => "string"}
    convert => {"NetworkSecurityGroup" => "string"}
    split => { "[records][properties][flows][flows][flowTuples]" => ","}
    add_field => {
        "unixtimestamp" => "%{[records][properties][flows][flows][flowTuples][0]}"
        "srcIp" => "%{[records][properties][flows][flows][flowTuples][1]}"
        "destIp" => "%{[records][properties][flows][flows][flowTuples][2]}"
        "srcPort" => "%{[records][properties][flows][flows][flowTuples][3]}"
        "destPort" => "%{[records][properties][flows][flows][flowTuples][4]}"
        "protocol" => "%{[records][properties][flows][flows][flowTuples][5]}"
        "trafficflow" => "%{[records][properties][flows][flows][flowTuples][6]}"
        "traffic" => "%{[records][properties][flows][flows][flowTuples][7]}"
    }
    add_field => {
        "time" => "%{[records][time]}"
        "systemId" => "%{[records][systemId]}"
        "category" => "%{[records][category]}"
        "resourceId" => "%{[records][resourceId]}"
        "operationName" => "%{[records][operationName]}"
        "Version" => "%{[records][properties][Version]}"
        "rule" => "%{[records][properties][flows][rule]}"
        "mac" => "%{[records][properties][flows][mac]}"
    }
    convert => {"unixtimestamp" => "integer"}
    convert => {"srcPort" => "integer"}
    convert => {"destPort" => "integer"}
    add_field => { "message" => "%{Message}" }
}
date {
    match => [ "unixtimestamp" , "UNIX"]
}
}
output {
    stdout { codec => rubydebug }
    udp {
        host => "127.0.0.1"
        port => 12201
    }
}

```

The Logstash config file provided is composed of three parts: the input, filter, and output. The input section designates the input source of the logs that Logstash will process – in this case, you are going to use an Azure blob input plugin (installed in the next steps) that allows us to access the network security group flow log JSON files stored in blob storage.

The filter section then flattens each flow log file so that each individual flow tuple and its associated properties becomes a separate Logstash event.

Finally, the output section forwards each Logstash event to the Graylog server. To suit your specific needs, modify the Logstash config file, as required.

NOTE

The previous config file assumes that the Graylog server has been configured on the local host loopback IP address 127.0.0.1. If not, be sure to change the host parameter in the output section to the correct IP address.

For further instructions on installing Logstash, see the Logstash [documentation](#).

Install the Logstash input plug-in for Azure blob storage

The Logstash plugin allows you to directly access the flow logs from their designated blob storage account. To install the plug-in, from the default Logstash installation directory (in this case /usr/share/logstash/bin), run the following command:

```
cd /usr/share/logstash/bin  
sudo ./logstash-plugin install logstash-input-azureblob
```

For more information about this plug in, see the [documentation](#).

Set up connection from Logstash to Graylog

Now that you have established a connection to the flow logs using Logstash and set up the Graylog server, you need to configure Graylog to accept the incoming log files.

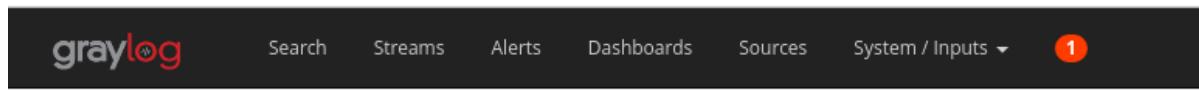
1. Navigate to your Graylog Server web interface using the URL you configured for it. You can access the interface by directing your browser to <http://<graylog-server-ip>:9000>
2. To navigate to the configuration page, select the **System** drop-down menu in the top navigation bar to the right, and then click **Inputs**. Alternatively, navigate to <http://<graylog-server-ip>:9000/system/inputs>

The screenshot shows the Graylog web interface. At the top, there's a navigation bar with links for Search, Streams, Alerts, Dashboards, Sources, and System. The System dropdown is open, showing options like Overview, Configurations, Nodes, Inputs, Outputs, Indices, Logging, Authentication, Content Packs, Grok Patterns, Collectors, Enterprise, and Pipelines. A red circle with the number '1' is visible on the System button. Below the navigation, the main content area displays the 'Getting Started - Graylog v2.2.2+69' page. It includes a welcome message and three numbered steps: 1. Send in first log messages, 2. Do something with your data, and 3. Create a dashboard.

3. To launch the new input, select **GELF UDP** in the **Select input** drop-down, and then fill out the form. GELF stands for Graylog Extended Log Format. The GELF format is developed by Graylog. To learn more about its advantages, see the Graylog [documentation](#).

Make sure to bind the input to the IP you configured your Graylog server on. The IP address should match the **host** field of the UDP output of the Logstash configuration file. The default port should be **12201**.

Ensure the port matches the **port** field in the UDP output designated in the Logstash config file.



Inputs

Graylog nodes accept data via inputs. Launch or terminate as many inputs as you want here.

GELF UDP x ▾ Launch new input Find more inputs

Global inputs 0 configured

There are no global inputs.

Once you launch the input, you should see it appear under the **Local inputs** section, as shown in the following picture:

Local inputs 1 configured

Logstash UDP GELF UDP RUNNING
On node ★d16ca94b / 10.26.0.6

bind_address: 127.0.0.1
decompress_size_limit: 8388608
override_source: <empty>
port: 12201
recv_buffer_size: 262144

Show received messages Manage extractors Stop input More actions ▾

Throughput / Metrics
1 minute average rate: 0 msg/s
Network IO: 0B 0B (total: 0B ▲0B)
Empty messages discarded: 0

To learn more about Graylog message inputs, refer to the [documentation](#).

4. Once these configurations have been made, you can start Logstash to begin reading in flow logs with the following command: `sudo systemctl start logstash.service`.

Search through Graylog messages

After allowing some time for your Graylog server to collect messages, you are able to search through the messages. To check the messages being sent to your Graylog server, from the **Inputs** configuration page click the **"Show received messages"** button of the GELF UDP input you created. You are directed to a screen that looks similar to the following picture:

The screenshot shows the Graylog search interface. At the top, there's a navigation bar with links for Search, Streams, Alerts, Dashboards, Sources, System, and Help. The status bar indicates "In 0 / Out 0 msg/s". On the left, a sidebar titled "Search result" shows a count of 12,529 messages found in 1 index, retrieved at 2017-04-11 00:00:31. It includes buttons for "Add count to dashboard", "Save search criteria", and "More actions". Below this is a list of fields: Default, All, None, Filter fields. A detailed list of fields is shown on the right, including @timestamp, @version, category, destip, destPort, mac, message (which is checked), NetworkSecurityGroup, operationName, and nrerror. A link "List fields of current page or all fields." is also present. The main area features a "Histogram" chart showing message counts by minute from 23:55 to 23:59, with a peak around 23:58. Below the histogram is a table titled "Messages" with columns for Timestamp and source. The table lists four rows of data, each containing the timestamp 2017-04-11 00:00:29.027 and the source "unknown". Each row also contains a blue link "%{Message}".

Clicking on the blue "%{Message}" link expands each message to show the parameters of each flow tuple, as shown in the following picture:

The screenshot shows the expanded view of a single message. The title is "Messages" with a page number 1. The message details are as follows:

Timestamp	source
2017-04-11 00:02:10.145	unknown
%{Message}	
✉ 1c1438f4-1e4a-11e7-9f4e-000d3a30cdb7	Permalink Copy ID Show surrounding messages Test against stream

Below the message details, there are sections for "Received by", "Stored in index", "Routed into streams", and various log parameters:

- Received by**: Logstash UDP on `P d16ca94b / 10.26.0.6`
- Stored in index**: graylog_0
- Routed into streams**: All messages
- @timestamp**: 2017-02-16T12:47:44.000Z
- @version**: 1
- NetworkSecurityGroup**: NETANALYTICSVM-NSG
- ResourceGroup**: NETANALYTICS
- Subscription**: 6926FC75-CE7D-4C9E-A87F-C4E38C594EB5
- Version**: 1}
- category**: NetworkSecurityGroupFlowEvent
- destip**: 10.2.0.4
- destPort**: 3389

By default, all message fields are included in the search if you don't select a specific message field to search for. If you want to search for specific messages (i.e – flow tuples from a specific source IP) you can use the Graylog search query language as [documented](#)

Analyze network security group flow logs using Graylog

Now that Graylog is set up running, you can use some of its functionality to better understand your flow log data. One such way is by using dashboards to create specific views of your data.

Create a dashboard

1. In the top navigation bar, select **Dashboards** or navigate to <http://<graylog-server-ip>:9000/dashboards/>
2. From there, click the green **Create dashboard** button and fill out the short form with the title and description of your dashboard. Hit the **Save** button to create the new dashboard. You see a dashboard similar to the following picture:

The screenshot shows the Graylog web interface. At the top, the navigation bar includes 'Search', 'Streams', 'Alerts', 'Dashboards' (which is the active tab), 'Sources', 'System' (with a red notification badge '1'), 'In 0 / Out 0 msg/s', 'Help', and 'Administrator'. Below the navigation is a section titled 'Dashboards' with a sub-section titled 'Sample NSG Flow Log Dashboard'. This dashboard has a title 'test nsg flow log dashboard' and two buttons: 'Edit dashboard' and 'More actions'. A note says 'Take a look at the [dashboard tutorial](#) for lots of other useful tips.'

Add widgets

You can click the title of the dashboard to see it, but right now it's empty, since we haven't added any widgets. An easy and useful type widget to add to the dashboard are **Quick Values** charts, which display a list of values of the selected field, and their distribution.

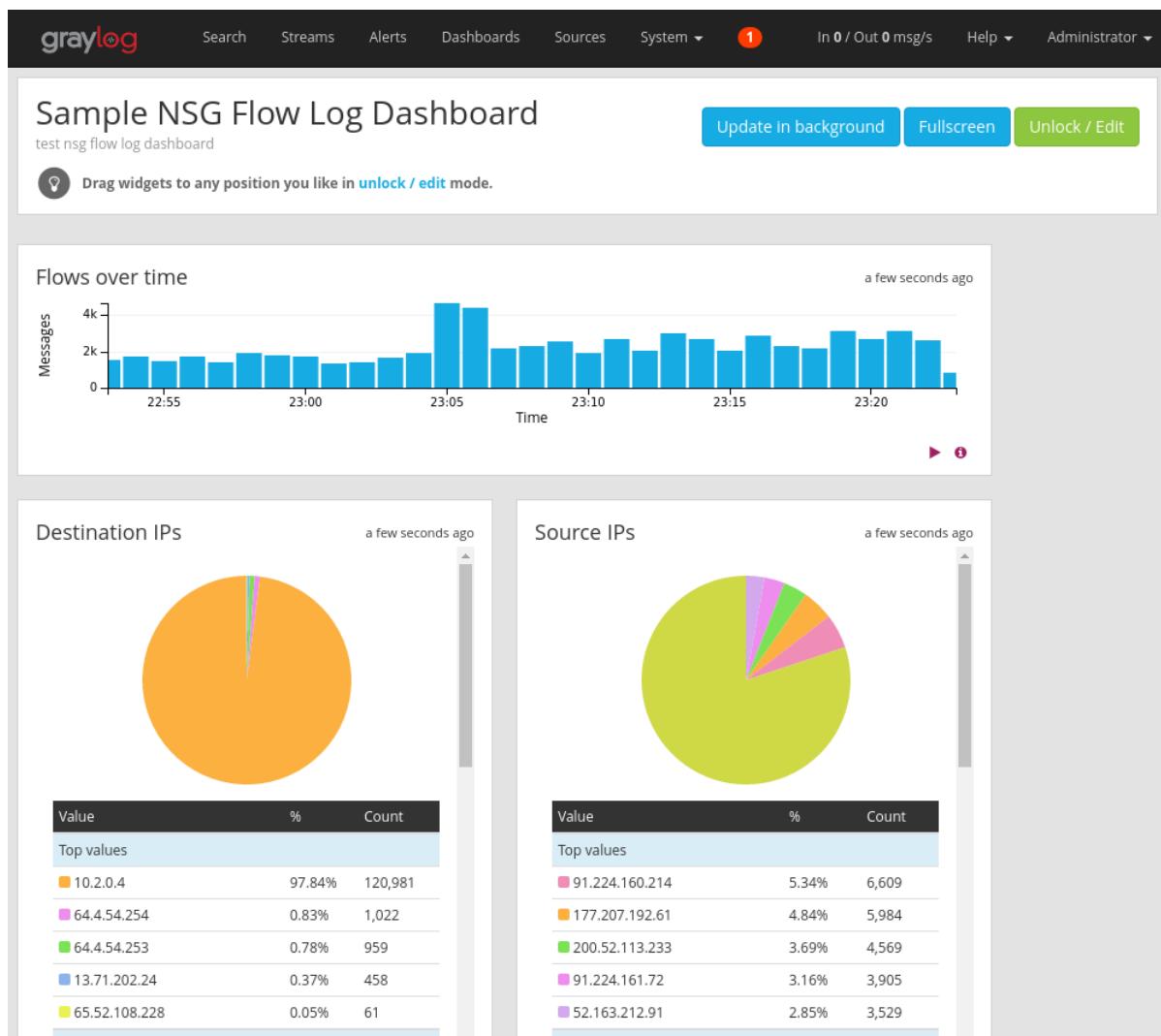
1. Navigate back to the search results of the UDP input that's receiving flow logs by selecting **Search** from the top navigation bar.
2. Under the **Search result** panel to the left side of the screen, find the **Fields** tab, which lists the various fields of each incoming flow tuple message.
3. Select any desired parameter in which to visualize (in this example, the IP source is selected). To show the list of possible widgets, click the blue drop-down arrow to the left of the field, then select **Quick values** to generate the widget. You should see something similar to the following picture:

The screenshot shows the Graylog interface with a search result for 'gl2_source_input:58be0be8f0e1fc0e94f80450'. On the left, the 'Search result' panel shows '8,512 messages' found in 18 ms, searched in 1 index, and retrieved at 2017-04-12 22:41:20. It includes buttons for 'Add count to dashboard', 'Save search criteria', and 'More actions'. Below this are tabs for 'Fields' (selected) and 'Decorators', and a dropdown for 'srclp' with options: 'Generate chart', 'Quick values' (which is selected and highlighted in blue), 'Statistics', and 'World Map'. On the right, a 'Quick Values for srclp' chart is displayed, showing a pie chart of the distribution of source IPs and a table of the top values. The table data is as follows:

Value	%	Count
Top values		
173.50.68.20	21.11%	2,429
107.183.247.111	10.43%	1,200
91.224.161.72	5.13%	590
176.120.46.132	3.79%	436
144.76.16.176	3.70%	426
Others		
52.163.212.91	2.65%	305
40.121.152.218	2.13%	245
103.25.200.147	1.90%	219
10.2.0.4	1.73%	199

4. From there, you can select the **Add to dashboard** button at the top right corner of the widget and select the corresponding dashboard to add.
5. Navigate back to the dashboard to see the widget you just added.

You can add a variety of other widgets such as histograms and counts to your dashboard to keep track of important metrics, such as the sample dashboard shown in the following picture:



For further explanation on dashboards and the other types of widgets, refer to Graylog's [documentation](#).

By integrating Network Watcher with Graylog, you now have a convenient and centralized way to manage and visualize network security group flow logs. Graylog has a number of other powerful features such as streams and alerts that can also be used to further manage flow logs and better understand your network traffic. Now that you have Graylog set up and connected to Azure, feel free to continue to explore the other functionality that it offers.

Next steps

Learn how to visualize your network security group flow logs with Power BI by visiting [Visualize network security group flows logs with Power BI](#).

Analyze your Virtual Machine security with Security Group View using PowerShell

1/28/2020 • 2 minutes to read • [Edit Online](#)

Security group view returns configured and effective network security rules that are applied to a virtual machine. This capability is useful to audit and diagnose Network Security Groups and rules that are configured on a VM to ensure traffic is being correctly allowed or denied. In this article, we show you how to retrieve the configured and effective security rules to a virtual machine using PowerShell

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Before you begin

In this scenario, you run the `Get-AzNetworkWatcherSecurityGroupView` cmdlet to retrieve the security rule information.

This scenario assumes you have already followed the steps in [Create a Network Watcher](#) to create a Network Watcher.

Scenario

The scenario covered in this article retrieves the configured and effective security rules for a given virtual machine.

Retrieve Network Watcher

The first step is to retrieve the Network Watcher instance. This variable is passed to the `Get-AzNetworkWatcherSecurityGroupView` cmdlet.

```
$networkWatcher = Get-AzResource | Where {$_.ResourceType -eq "Microsoft.Network/networkWatchers" -and  
$_.Location -eq "WestCentralUS" }
```

Get a VM

A virtual machine is required to run the `Get-AzNetworkWatcherSecurityGroupView` cmdlet against. The following example gets a VM object.

```
$VM = Get-AzVM -ResourceGroupName testrg -Name testvm1
```

Retrieve security group view

The next step is to retrieve the security group view result.

```
$secgroup = Get-AzNetworkWatcherSecurityGroupView -NetworkWatcher $networkWatcher -TargetVirtualMachineId  
$VM.Id
```

Viewing the results

The following example is a shortened response of the results returned. The results show all the effective and applied security rules on the virtual machine broken down in groups of **NetworkInterfaceSecurityRules**, **DefaultSecurityRules**, and **EffectiveSecurityRules**.

```

NetworkInterfaces : [
    {
        "NetworkInterfaceSecurityRules": [
            {
                "Name": "default-allow-rdp",
                "Etag": "W/\"d4c411d4-0d62-49dc-8092-3d4b57825740\"",
                "Id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/testrg2/providers/Microsoft.Network/networkSecurityGroups/testvm2-
nsg/securityRules/default-allow-rdp",
                "Protocol": "TCP",
                "SourcePortRange": "*",
                "DestinationPortRange": "3389",
                "SourceAddressPrefix": "*",
                "DestinationAddressPrefix": "*",
                "Access": "Allow",
                "Priority": 1000,
                "Direction": "Inbound",
                "ProvisioningState": "Succeeded"
            }
            ...
        ],
        "DefaultSecurityRules": [
            {
                "Name": "AllowVnetInBound",
                "Id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/testrg2/providers/Microsoft.Network/networkSecurityGroups/testvm2-
nsg/defaultSecurityRules/",
                "Description": "Allow inbound traffic from all VMs in VNET",
                "Protocol": "*",
                "SourcePortRange": "*",
                "DestinationPortRange": "*",
                "SourceAddressPrefix": "VirtualNetwork",
                "DestinationAddressPrefix": "VirtualNetwork",
                "Access": "Allow",
                "Priority": 65000,
                "Direction": "Inbound",
                "ProvisioningState": "Succeeded"
            }
            ...
        ],
        "EffectiveSecurityRules": [
            {
                "Name": "DefaultOutboundDenyAll",
                "Protocol": "All",
                "SourcePortRange": "0-65535",
                "DestinationPortRange": "0-65535",
                "SourceAddressPrefix": "*",
                "DestinationAddressPrefix": "*",
                "ExpandedSourceAddressPrefix": [],
                "ExpandedDestinationAddressPrefix": [],
                "Access": "Deny",
                "Priority": 65500,
                "Direction": "Outbound"
            },
            ...
        ]
    }
]

```

Next steps

Visit [Auditing Network Security Groups \(NSG\) with Network Watcher](#) to learn how to automate validation of Network Security Groups.

Analyze your Virtual Machine security with Security Group View using Azure CLI

1/28/2020 • 2 minutes to read • [Edit Online](#)

Security group view returns configured and effective network security rules that are applied to a virtual machine. This capability is useful to audit and diagnose Network Security Groups and rules that are configured on a VM to ensure traffic is being correctly allowed or denied. In this article, we show you how to retrieve the configured and effective security rules to a virtual machine using Azure CLI

To perform the steps in this article, you need to [install the Azure command-line interface for Mac, Linux, and Windows \(CLI\)](#).

Before you begin

This scenario assumes you have already followed the steps in [Create a Network Watcher](#) to create a Network Watcher.

Scenario

The scenario covered in this article retrieves the configured and effective security rules for a given virtual machine.

Get a VM

A virtual machine is required to run the `vm list` cmdlet. The following command lists the virtual machines in a resource group:

```
az vm list -resource-group resourceGroupName
```

Once you know the virtual machine, you can use the `vm show` cmdlet to get its resource Id:

```
az vm show -resource-group resourceGroupName -name virtualMachineName
```

Retrieve security group view

The next step is to retrieve the security group view result.

```
az network watcher show-security-group-view --resource-group resourceGroupName --vm vmName
```

Viewing the results

The following example is a shortened response of the results returned. The results show all the effective and applied security rules on the virtual machine broken down in groups of **NetworkInterfaceSecurityRules**, **DefaultSecurityRules**, and **EffectiveSecurityRules**.

```
{
  "networkInterfaces": [
    {
      "id": "/subscriptions/00000000-0000-0000-0000-
```

```
000000000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkInterfaces/{nicName}",
  "resourceGroup": "{resourceGroupName}",
  "securityRuleAssociations": {
    "defaultSecurityRules": [
      {
        "access": "Allow",
        "description": "Allow inbound traffic from all VMs in VNET",
        "destinationAddressPrefix": "VirtualNetwork",
        "destinationPortRange": "*",
        "direction": "Inbound",
        "etag": null,
        "id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups//providers/Microsoft.Network/networkSecurityGroups/{nsgName}/defaultSecurityRules
/AllowVnetInBound",
          "name": "AllowVnetInBound",
          "priority": 65000,
          "protocol": "*",
          "provisioningState": "Succeeded",
          "resourceGroup": "",
          "sourceAddressPrefix": "VirtualNetwork",
          "sourcePortRange": "*"
        }...
      ],
      "effectiveSecurityRules": [
        {
          "access": "Deny",
          "destinationAddressPrefix": "*",
          "destinationPortRange": "0-65535",
          "direction": "Outbound",
          "expandedDestinationAddressPrefix": null,
          "expandedSourceAddressPrefix": null,
          "name": "DefaultOutboundDenyAll",
          "priority": 65500,
          "protocol": "All",
          "sourceAddressPrefix": "*",
          "sourcePortRange": "0-65535"
        },
        {
          "access": "Allow",
          "destinationAddressPrefix": "VirtualNetwork",
          "destinationPortRange": "0-65535",
          "direction": "Outbound",
          "expandedDestinationAddressPrefix": [
            "10.1.0.0/24",
            "168.63.129.16/32"
          ],
          "expandedSourceAddressPrefix": [
            "10.1.0.0/24",
            "168.63.129.16/32"
          ],
          "name": "DefaultRule_AllowVnetOutBound",
          "priority": 65000,
          "protocol": "All",
          "sourceAddressPrefix": "VirtualNetwork",
          "sourcePortRange": "0-65535"
        }...
      ],
      "networkInterfaceAssociation": {
        "id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkInterfaces/{nicName}",
        "resourceGroup": "{resourceGroupName}",
        "securityRules": [
          {
            "access": "Allow",
            "description": null,
            "destinationAddressPrefix": "*",
            "destinationPortRange": "3389",
            "direction": "Inbound",
            "etag": null,
            "id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups//providers/Microsoft.Network/networkSecurityGroups/{nsgName}/securityRules
/AllowRDPInBound"
          }
        ]
      }
    ]
  }
}
```

```
etag : "W/\"ETU000001-2054-4/5d-d020-udc78a0cc//\"",
"id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkSecurityGroups/{nsgName}/securityRules/default-allow-rdp",
"name": "default-allow-rdp",
"priority": 1000,
"protocol": "TCP",
"provisioningState": "Succeeded",
"resourceGroup": "{resourceGroupName}",
"sourceAddressPrefix": "*",
"sourcePortRange": "*"
}
]
},
"subnetAssociation": null
}
}
]
```

Next steps

Visit [Auditing Network Security Groups \(NSG\) with Network Watcher](#) to learn how to automate validation of Network Security Groups.

Learn more about the security rules that are applied to your network resources by visiting [Security group view overview](#)

Analyze your Virtual Machine security with Security Group View using REST API

1/28/2020 • 2 minutes to read • [Edit Online](#)

Security group view returns configured and effective network security rules that are applied to a virtual machine. This capability is useful to audit and diagnose Network Security Groups and rules that are configured on a VM to ensure traffic is being correctly allowed or denied. In this article, we show you how to retrieve the effective and applied security rules to a virtual machine using REST API

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Before you begin

In this scenario, you call the Network Watcher Rest API to get the security group view for a virtual machine. ARMclient is used to call the REST API using PowerShell. ARMClient is found on chocolatey at [ARMClient on Chocolatey](#)

This scenario assumes you have already followed the steps in [Create a Network Watcher](#) to create a Network Watcher. The scenario also assumes that a Resource Group with a valid virtual machine exists to be used.

Scenario

The scenario covered in this article retrieves the effective and applied security rules for a given virtual machine.

Log in with ARMClient

```
armclient login
```

Retrieve a virtual machine

Run the following script to return a virtual machineThe following code needs variables:

- **subscriptionId** - The subscription id can also be retrieved with the **Get-AzSubscription** cmdlet.
- **resourceGroupName** - The name of a resource group that contains virtual machines.

```
$subscriptionId = '<subscription id>'  
$resourceGroupName = '<resource group name>'  
  
armclient get  
https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Microsoft.Compute/virtualMachines?api-version=2015-05-01-preview
```

The information that is needed is the **id** under the type `Microsoft.Compute/virtualMachines` in response, as seen in

the following example:

```
...,
  "networkProfile": {
    "networkInterfaces": [
      {
        "id": "/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkInterfaces/{nicName}"
      }
    ],
    "provisioningState": "Succeeded"
  },
  "resources": [
    {
      "id": "/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachines/{vmName}/extensions/CustomScriptExtension"
    }
  ],
  "type": "Microsoft.Compute/virtualMachines",
  "location": "westcentralus",
  "id": "/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachines/{vmName}",
  "name": "{vmName}"
}
]
```

Get security group view for virtual machine

The following example requests the security group view of a targeted virtual machine. The results from this example can be used to compare to the rules and security defined by the origination to look for configuration drift.

```
$subscriptionId = "<subscription id>"
$resourceGroupName = "<resource group name>"
$networkWatcherName = "<network watcher name>"
$targetUri = "<uri of target resource>" # Example:
/subscriptions/$subscriptionId/resourceGroups/$resourceGroupName/providers/Microsoft.compute/virtualMachine/$v
mName

$requestBody = @"
{
  'targetResourceId': '${targetUri}'
}

@"
armclient post
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${resourceGroupName}/providers/Mi
crosoft.Network/networkWatchers/${networkWatcherName}/securityGroupView?api-version=2016-12-01" $requestBody -
verbose
```

View the response

The following sample is the response returned from the preceding command. The results show all the effective and applied security rules on the virtual machine broken down in groups of **NetworkInterfaceSecurityRules**, **DefaultSecurityRules**, and **EffectiveSecurityRules**.

```
{
  "networkInterfaces": [
    {

```

```
"securityRuleAssociations": {
    "networkInterfaceAssociation": {
        "id": "/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkInterfaces/{nicName}",
        "securityRules": [
            {
                "name": "default-allow-rdp",
                "id": "/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkSecurityGroups/{nsgName}/securityRules/default-allow-rdp",
                "etag": "W/\\"d4c411d4-0d62-49dc-8092-3d4b57825740\\\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "protocol": "TCP",
                    "sourcePortRange": "*",
                    "destinationPortRange": "3389",
                    "sourceAddressPrefix": "*",
                    "destinationAddressPrefix": "*",
                    "access": "Allow",
                    "priority": 1000,
                    "direction": "Inbound"
                }
            }
        ]
    },
    "defaultSecurityRules": [
        {
            "name": "AllowVnetInBound",
            "id": "/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkSecurityGroups/{nsgName}/defaultSecurityRules/",
            "properties": {
                "provisioningState": "Succeeded",
                "description": "Allow inbound traffic from all VMs in VNET",
                "protocol": "*",
                "sourcePortRange": "*",
                "destinationPortRange": "*",
                "sourceAddressPrefix": "VirtualNetwork",
                "destinationAddressPrefix": "VirtualNetwork",
                "access": "Allow",
                "priority": 65000,
                "direction": "Inbound"
            }
        },
        ...
    ],
    "effectiveSecurityRules": [
        {
            "name": "DefaultOutboundDenyAll",
            "protocol": "All",
            "sourcePortRange": "0-65535",
            "destinationPortRange": "0-65535",
            "sourceAddressPrefix": "*",
            "destinationAddressPrefix": "*",
            "access": "Deny",
            "priority": 65500,
            "direction": "Outbound"
        },
        ...
    ]
}
```

Next steps

Visit [Auditing Network Security Groups \(NSG\) with Network Watcher](#) to learn how to automate validation of Network Security Groups.

Automate NSG auditing with Azure Network Watcher Security group view

1/28/2020 • 3 minutes to read • [Edit Online](#)

Customers are often faced with the challenge of verifying the security posture of their infrastructure. This challenge is no different for their VMs in Azure. It is important to have a similar security profile based on the Network Security Group (NSG) rules applied. Using the Security Group View, you can now get the list of rules applied to a VM within an NSG. You can define a golden NSG security profile and initiate Security Group View on a weekly cadence and compare the output to the golden profile and create a report. This way you can identify with ease all the VMs that do not conform to the prescribed security profile.

If you are unfamiliar with Network Security Groups, see [Network Security Overview](#).

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Before you begin

In this scenario, you compare a known good baseline to the security group view results returned for a virtual machine.

This scenario assumes you have already followed the steps in [Create a Network Watcher](#) to create a Network Watcher. The scenario also assumes that a Resource Group with a valid virtual machine exists to be used.

Scenario

The scenario covered in this article gets the security group view for a virtual machine.

In this scenario, you will:

- Retrieve a known good rule set
- Retrieve a virtual machine with Rest API
- Get security group view for virtual machine
- Evaluate Response

Retrieve rule set

The first step in this example is to work with an existing baseline. The following example is some json extracted from an existing Network Security Group using the `Get-AzNetworkSecurityGroup` cmdlet that is used as the baseline for this example.

```
[  
 {  
   "Description": null,  
   "Protocol": "TCP",  
   "SourcePortRange": "*",  
   "DestinationPortRange": "3389",
```

```

        "SourceAddressPrefix": "*",
        "DestinationAddressPrefix": "*",
        "Access": "Allow",
        "Priority": 1000,
        "Direction": "Inbound",
        "ProvisioningState": "Succeeded",
        "Name": "default-allow-rdp",
        "Etag": "W/\"d8859256-1c4c-4b93-ba7d-73d9bf67c4f1\",
        "Id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/testrg/providers/Microsoft.Network/networkSecurityGroups/testvm1-
nsg/securityRules/default-allow-rdp"
    },
    {
        "Description": null,
        "Protocol": "*",
        "SourcePortRange": "*",
        "DestinationPortRange": "111",
        "SourceAddressPrefix": "*",
        "DestinationAddressPrefix": "*",
        "Access": "Allow",
        "Priority": 1010,
        "Direction": "Inbound",
        "ProvisioningState": "Succeeded",
        "Name": "MyRuleDoNotDelete",
        "Etag": "W/\"d8859256-1c4c-4b93-ba7d-73d9bf67c4f1\",
        "Id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/testrg/providers/Microsoft.Network/networkSecurityGroups/testvm1-
nsg/securityRules/MyRuleDoNotDelete"
    },
    {
        "Description": null,
        "Protocol": "*",
        "SourcePortRange": "*",
        "DestinationPortRange": "112",
        "SourceAddressPrefix": "*",
        "DestinationAddressPrefix": "*",
        "Access": "Allow",
        "Priority": 1020,
        "Direction": "Inbound",
        "ProvisioningState": "Succeeded",
        "Name": "My2ndRuleDoNotDelete",
        "Etag": "W/\"d8859256-1c4c-4b93-ba7d-73d9bf67c4f1\",
        "Id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/testrg/providers/Microsoft.Network/networkSecurityGroups/testvm1-
nsg/securityRules/My2ndRuleDoNotDelete"
    },
    {
        "Description": null,
        "Protocol": "TCP",
        "SourcePortRange": "*",
        "DestinationPortRange": "5672",
        "SourceAddressPrefix": "*",
        "DestinationAddressPrefix": "*",
        "Access": "Deny",
        "Priority": 1030,
        "Direction": "Inbound",
        "ProvisioningState": "Succeeded",
        "Name": "ThisRuleNeedsToStay",
        "Etag": "W/\"d8859256-1c4c-4b93-ba7d-73d9bf67c4f1\",
        "Id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/testrg/providers/Microsoft.Network/networkSecurityGroups/testvm1-
nsg/securityRules/ThisRuleNeedsToStay"
    }
]

```

Convert rule set to PowerShell objects

In this step, we are reading a json file that was created earlier with the rules that are expected to be on the Network Security Group for this example.

```
$nsgbaserules = Get-Content -Path C:\temp\testvm1-nsg.json | ConvertFrom-Json
```

Retrieve Network Watcher

The next step is to retrieve the Network Watcher instance. The `$networkWatcher` variable is passed to the `AzNetworkWatcherSecurityGroupView` cmdlet.

```
$networkWatcher = Get-AzResource | Where {$_.ResourceType -eq "Microsoft.Network/networkWatchers" -and  
$_.Location -eq "WestCentralUS" }
```

Get a VM

A virtual machine is required to run the `Get-AzNetworkWatcherSecurityGroupView` cmdlet against. The following example gets a VM object.

```
$VM = Get-AzVM -ResourceGroupName "testrg" -Name "testvm1"
```

Retrieve security group view

The next step is to retrieve the security group view result. This result is compared to the "baseline" json that was shown earlier.

```
$secgroup = Get-AzNetworkWatcherSecurityGroupView -NetworkWatcher $networkWatcher -TargetVirtualMachineId  
$VM.Id
```

Analyzing the results

The response is grouped by Network interfaces. The different types of rules returned are effective and default security rules. The result is further broken down by how it is applied, either on a subnet or a virtual NIC.

The following PowerShell script compares the results of the Security Group View to an existing output of an NSG. The following example is a simple example of how the results can be compared with `Compare-Object` cmdlet.

```
Compare-Object -ReferenceObject $nsgbaserules `  
-DifferenceObject $secgroup.NetworkInterfaces[0].NetworkInterfaceSecurityRules `  
-Property  
Name,Description,Protocol,SourcePortRange,DestinationPortRange,SourceAddressPrefix,DestinationAddressPrefix,Access,Priority,Direction
```

The following example is the result. You can see two of the rules that were in the first rule set were not present in the comparison.

```
Name          : My2ndRuleDoNotDelete
Description   :
Protocol     : *
SourcePortRange : *
DestinationPortRange : 112
SourceAddressPrefix : *
DestinationAddressPrefix : *
Access        : Allow
Priority      : 1020
Direction     : Inbound
SideIndicator : <=

Name          : ThisRuleNeedsToStay
Description   :
Protocol     : TCP
SourcePortRange : *
DestinationPortRange : 5672
SourceAddressPrefix : *
DestinationAddressPrefix : *
Access        : Deny
Priority      : 1030
Direction     : Inbound
SideIndicator : <=
```

Next steps

If settings have been changed, see [Manage Network Security Groups](#) to track down the network security group and security rules that are in question.

Troubleshoot Virtual Network Gateway and Connections using Azure Network Watcher PowerShell

1/28/2020 • 2 minutes to read • [Edit Online](#)

Network Watcher provides many capabilities as it relates to understanding your network resources in Azure. One of these capabilities is resource troubleshooting. Resource troubleshooting can be called through the portal, PowerShell, CLI, or REST API. When called, Network Watcher inspects the health of a Virtual Network Gateway or a Connection and returns its findings.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Before you begin

This scenario assumes you have already followed the steps in [Create a Network Watcher](#) to create a Network Watcher.

For a list of supported gateway types visit, [Supported Gateway types](#).

Overview

Resource troubleshooting provides the ability troubleshoot issues that arise with Virtual Network Gateways and Connections. When a request is made to resource troubleshooting, logs are being queried and inspected. When inspection is complete, the results are returned. Resource troubleshooting requests are long running requests, which could take multiple minutes to return a result. The logs from troubleshooting are stored in a container on a storage account that is specified.

Retrieve Network Watcher

The first step is to retrieve the Network Watcher instance. The `$networkWatcher` variable is passed to the `Start-AzNetworkWatcherResourceTroubleshooting` cmdlet in step 4.

```
$networkWatcher = Get-AzResource | Where {$_.ResourceType -eq "Microsoft.Network/networkWatchers" -and  
$_.Location -eq "WestCentralUS" }
```

Retrieve a Virtual Network Gateway Connection

In this example, resource troubleshooting is being ran on a Connection. You can also pass it a Virtual Network Gateway.

```
$connection = Get-AzVirtualNetworkGatewayConnection -Name "2to3" -ResourceGroupName "testrg"
```

Create a storage account

Resource troubleshooting returns data about the health of the resource, it also saves logs to a storage account to be reviewed. In this step, we create a storage account, if an existing storage account exists you can use it.

```
$sa = New-AzStorageAccount -Name "contosoexamplesa" -SKU "Standard_LRS" -ResourceGroupName "testrg" -Location "WestCentralUS"  
Set-AzCurrentStorageAccount -ResourceGroupName $sa.ResourceGroupName -Name $sa.StorageAccountName  
$sc = New-AzStorageContainer -Name logs
```

Run Network Watcher resource troubleshooting

You troubleshoot resources with the `Start-AzNetworkWatcherResourceTroubleshooting` cmdlet. We pass the cmdlet the Network Watcher object, the Id of the Connection or Virtual Network Gateway, the storage account id, and the path to store the results.

NOTE

The `Start-AzNetworkWatcherResourceTroubleshooting` cmdlet is long running and may take a few minutes to complete.

```
Start-AzNetworkWatcherResourceTroubleshooting -NetworkWatcher $networkWatcher -TargetResourceId $connection.Id  
-StorageId $sa.Id -StoragePath "$($sa.PrimaryEndpoints.Blob)$($sc.name)"
```

Once you run the cmdlet, Network Watcher reviews the resource to verify the health. It returns the results to the shell and stores logs of the results in the storage account specified.

Understanding the results

The action text provides general guidance on how to resolve the issue. If an action can be taken for the issue, a link is provided with additional guidance. In the case where there is no additional guidance, the response provides the url to open a support case. For more information about the properties of the response and what is included, visit [Network Watcher Troubleshoot overview](#)

For instructions on downloading files from azure storage accounts, refer to [Get started with Azure Blob storage using .NET](#). Another tool that can be used is Storage Explorer. More information about Storage Explorer can be found here at the following link: [Storage Explorer](#)

Next steps

If settings have been changed that stop VPN connectivity, see [Manage Network Security Groups](#) to track down the network security group and security rules that may be in question.

Troubleshoot Virtual Network Gateway and Connections using Azure Network Watcher Azure CLI

1/28/2020 • 2 minutes to read • [Edit Online](#)

Network Watcher provides many capabilities as it relates to understanding your network resources in Azure. One of these capabilities is resource troubleshooting. Resource troubleshooting can be called through the portal, PowerShell, CLI, or REST API. When called, Network Watcher inspects the health of a Virtual Network Gateway or a Connection and returns its findings.

To perform the steps in this article, you need to [install the Azure command-line interface for Mac, Linux, and Windows \(CLI\)](#).

Before you begin

This scenario assumes you have already followed the steps in [Create a Network Watcher](#) to create a Network Watcher.

For a list of supported gateway types visit, [Supported Gateway types](#).

Overview

Resource troubleshooting provides the ability troubleshoot issues that arise with Virtual Network Gateways and Connections. When a request is made to resource troubleshooting, logs are being queried and inspected. When inspection is complete, the results are returned. Resource troubleshooting requests are long running requests, which could take multiple minutes to return a result. The logs from troubleshooting are stored in a container on a storage account that is specified.

Retrieve a Virtual Network Gateway Connection

In this example, resource troubleshooting is being ran on a Connection. You can also pass it a Virtual Network Gateway. The following cmdlet lists the vpn-connections in a resource group.

```
az network vpn-connection list --resource-group resourceGroupName
```

Once you have the name of the connection, you can run this command to get its resource Id:

```
az network vpn-connection show --resource-group resourceGroupName --ids vpnConnectionIds
```

Create a storage account

Resource troubleshooting returns data about the health of the resource, it also saves logs to a storage account to be reviewed. In this step, we create a storage account, if an existing storage account exists you can use it.

1. Create the storage account

```
az storage account create --name storageAccountName --location westcentralus --resource-group resourceGroupName --sku Standard_LRS
```

2. Get the storage account keys

```
az storage account keys list --resource-group resourcegroupName --account-name storageAccountName
```

3. Create the container

```
az storage container create --account-name storageAccountName --account-key {storageAccountKey} --name logs
```

Run Network Watcher resource troubleshooting

You troubleshoot resources with the `az network watcher troubleshooting` cmdlet. We pass the cmdlet the resource group, the name of the Network Watcher, the Id of the connection, the Id of the storage account, and the path to the blob to store the troubleshoot results in.

```
az network watcher troubleshooting start --resource-group resourceGroupName --resource resourceName --resource-type {vnetGateway/vpnConnection} --storage-account storageAccountName --storage-path https://{storageAccountName}.blob.core.windows.net/{containerName}
```

Once you run the cmdlet, Network Watcher reviews the resource to verify the health. It returns the results to the shell and stores logs of the results in the storage account specified.

Understanding the results

The action text provides general guidance on how to resolve the issue. If an action can be taken for the issue, a link is provided with additional guidance. In the case where there is no additional guidance, the response provides the url to open a support case. For more information about the properties of the response and what is included, visit [Network Watcher Troubleshoot overview](#)

For instructions on downloading files from azure storage accounts, refer to [Get started with Azure Blob storage using .NET](#). Another tool that can be used is Storage Explorer. More information about Storage Explorer can be found here at the following link: [Storage Explorer](#)

Next steps

If settings have been changed that stop VPN connectivity, see [Manage Network Security Groups](#) to track down the network security group and security rules that may be in question.

Troubleshoot Virtual Network gateway and Connections using Azure Network Watcher

1/28/2020 • 6 minutes to read • [Edit Online](#)

Network Watcher provides many capabilities as it relates to understanding your network resources in Azure. One of these capabilities is resource troubleshooting. Resource troubleshooting can be called through the portal, PowerShell, CLI, or REST API. When called, Network Watcher inspects the health of a Virtual Network Gateway or a Connection and returns its findings.

This article takes you through the different management tasks that are currently available for resource troubleshooting.

- [Troubleshoot a Virtual Network gateway](#)
- [Troubleshoot a Connection](#)

Before you begin

ARMclient is used to call the REST API using PowerShell. ARMClient is found on chocolatey at [ARMClient on Chocolatey](#)

This scenario assumes you have already followed the steps in [Create a Network Watcher](#) to create a Network Watcher.

For a list of supported gateway types visit, [Supported Gateway types](#).

Overview

Network Watcher troubleshooting provides the ability troubleshoot issues that arise with Virtual Network gateways and Connections. When a request is made to the resource troubleshooting, logs are querying and inspected. When inspection is complete, the results are returned. The troubleshoot API requests are long running requests, which could take multiple minutes to return a result. Logs are stored in a container on a storage account.

Log in with ARMClient

```
armclient login
```

Troubleshoot a Virtual Network gateway

POST the troubleshoot request

The following example queries the status of a Virtual Network gateway.

```

$subscriptionId = "00000000-0000-0000-0000-000000000000"
$resourceGroupName = "ContosoRG"
$NWresourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
$vnetGatewayName = "ContosoVNETGateway"
$storageAccountName = "contososa"
$containerName = "gwlogs"
$requestBody = @"
{
  'TargetResourceId':
    '/subscriptions/${subscriptionId}/resourceGroups/${resourceGroupName}/providers/Microsoft.Network/virtualNetwo
    rkGateways/${vnetGatewayName}',
  'Properties': {
    'StorageId':
      '/subscriptions/${subscriptionId}/resourceGroups/${resourceGroupName}/providers/Microsoft.Storage/storageAccou
      nts/${storageAccountName}',
    'StoragePath': 'https://${storageAccountName}.blob.core.windows.net/${containerName}'
  }
}
"
@"

armclient post
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${NWresourceGroupName}/providers/
Microsoft.Network/networkWatchers/${networkWatcherName}/troubleshoot?api-version=2016-03-30" $requestBody -
verbose

```

Since this operation is long running, the URI for querying the operation and the URI for the result is returned in the response header as shown in the following response:

Important Values

- Azure-AsyncOperation** - This property contains the URI to query the Async troubleshoot operation
- Location** - This property contains the URI where the results are when the operation is complete

```

HTTP/1.1 202 Accepted
Pragma: no-cache
Retry-After: 10
x-ms-request-id: 8a1167b7-6768-4ac1-85dc-703c9c9b9247
Azure-AsyncOperation: https://management.azure.com/subscriptions/00000000-0000-0000-0000-
000000000000/providers/Microsoft.Network/locations/westcentralus/operations/8a1167b7-6768-4ac1-85dc-
703c9c9b9247?api-version=2016-03-30
Strict-Transport-Security: max-age=31536000; includeSubDomains
Cache-Control: no-cache
Location: https://management.azure.com/subscriptions/00000000-0000-0000-0000-
000000000000/providers/Microsoft.Network/locations/westcentralus/operationResults/8a1167b7-6768-4ac1-85dc-
703c9c9b9247?api-version=2016-03-30
Server: Microsoft-HTTPAPI/2.0; Microsoft-HTTPAPI/2.0
x-ms-ratelimit-remaining-subscription-writes: 1199
x-ms-correlation-request-id: 4364d88a-bd08-422c-a716-dbb0cdc99f7b
x-ms-routing-request-id: NORTHCENTRALUS:20170112T183202Z:4364d88a-bd08-422c-a716-dbb0cdc99f7b
Date: Thu, 12 Jan 2017 18:32:01 GMT

null

```

Query the async operation for completion

Use the operations URI to query for the progress of the operation as seen in the following example:

```
armclient get "https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/providers/Microsoft.Network/locations/westcentralus/operations/8a1167b7-6768-4ac1-85dc-703c9c9b9247?api-version=2016-03-30" -verbose
```

While the operation is in progress, the response shows **InProgress** as seen in the following example:

```
{  
    "status": "InProgress"  
}
```

When the operation is complete the status changes to **Succeeded**.

```
{  
    "status": "Succeeded"  
}
```

Retrieve the results

Once the status returned is **Succeeded**, call a GET Method on the operationResult URI to retrieve the results.

```
armclient get "https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/providers/Microsoft.Network/locations/westcentralus/operationResults/8a1167b7-6768-4ac1-85dc-703c9c9b9247?api-version=2016-03-30" -verbose
```

The following responses are examples of a typical degraded response returned when querying the results of troubleshooting a gateway. See [Understanding the results](#) to get clarification on what the properties in the response mean.

```
{
  "startTime": "2017-01-12T10:31:41.562646-08:00",
  "endTime": "2017-01-12T18:31:48.677Z",
  "code": "Degraded",
  "results": [
    {
      "id": "PlatformInActive",
      "summary": "We are sorry, your VPN gateway is in standby mode",
      "detail": "During this time the gateway will not initiate or accept VPN connections with on premises VPN devices or other Azure VPN Gateways. This is a transient state while the Azure platform is being updated.",
      "recommendedActions": [
        {
          "actionText": "If the condition persists, please try resetting your Azure VPN gateway",
          "actionUri": "https://azure.microsoft.com/documentation/articles/vpn-gateway-resetgw-classic/",
          "actionUriText": "resetting the VPN Gateway"
        },
        {
          "actionText": "If your VPN gateway isn't up and running by the expected resolution time, contact support",
          "actionUri": "https://azure.microsoft.com/support",
          "actionUriText": "contact support"
        }
      ]
    },
    {
      "id": "NoFault",
      "summary": "This VPN gateway is running normally",
      "detail": "There aren't any known Azure platform problems affecting this VPN Connection",
      "recommendedActions": [
        {
          "actionText": "If you are still experiencing problems with the VPN gateway, please try resetting the VPN gateway.",
          "actionUri": "https://azure.microsoft.com/documentation/articles/vpn-gateway-resetgw-classic/",
          "actionUriText": "resetting VPN gateway"
        },
        {
          "actionText": "If you are experiencing problems you believe are caused by Azure, contact support",
          "actionUri": "https://azure.microsoft.com/support",
          "actionUriText": "contact support"
        }
      ]
    }
  ]
}
```

Troubleshoot Connections

The following example queries the status of a Connection.

```

$subscriptionId = "00000000-0000-0000-0000-000000000000"
$resourceGroupName = "ContosoRG"
$NWresourceGroupName = "NetworkWatcherRG"
$networkWatcherName = "NetworkWatcher_westcentralus"
$connectionName = "VNET2toVNET1Connection"
$storageAccountName = "contososa"
$containerName = "gwlogs"
$requestBody = @{
    "TargetResourceId": "/subscriptions/${subscriptionId}/resourceGroups/${resourceGroupName}/providers/Microsoft.Network/connections/${connectionName}",
    "Properties": {
        "StorageId": "/subscriptions/${subscriptionId}/resourceGroups/${resourceGroupName}/providers/Microsoft.Storage/storageAccounts/${storageAccountName}",
        "StoragePath": "https://${storageAccountName}.blob.core.windows.net/${containerName}"
    }
}

}
armclient post
"https://management.azure.com/subscriptions/${subscriptionId}/ResourceGroups/${NWresourceGroupName}/providers/Microsoft.Network/networkWatchers/${networkWatcherName}/troubleshoot?api-version=2016-03-30 $requestBody"

```

NOTE

The troubleshoot operation cannot be run in parallel on a Connection and its corresponding gateways. The operation must complete prior to running it on the previous resource.

Since this is a long running transaction, in the response header, the URI for querying the operation and the URI for the result is returned as shown in the following response:

Important Values

- **Azure-AsyncOperation** - This property contains the URI to query the Async troubleshoot operation
- **Location** - This property contains the URI where the results are when the operation is complete

```

HTTP/1.1 202 Accepted
Pragma: no-cache
Retry-After: 10
x-ms-request-id: 8a1167b7-6768-4ac1-85dc-703c9c9b9247
Azure-AsyncOperation: https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/providers/Microsoft.Network/locations/westcentralus/operations/8a1167b7-6768-4ac1-85dc-703c9c9b9247?api-version=2016-03-30
Strict-Transport-Security: max-age=31536000; includeSubDomains
Cache-Control: no-cache
Location: https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/providers/Microsoft.Network/locations/westcentralus/operationResults/8a1167b7-6768-4ac1-85dc-703c9c9b9247?api-version=2016-03-30
Server: Microsoft-HTTPAPI/2.0; Microsoft-HTTPAPI/2.0
x-ms-ratelimit-remaining-subscription-writes: 1199
x-ms-correlation-request-id: 4364d88a-bd08-422c-a716-dbb0cdc99f7b
x-ms-routing-request-id: NORTHCENTRALUS:20170112T183202Z:4364d88a-bd08-422c-a716-dbb0cdc99f7b
Date: Thu, 12 Jan 2017 18:32:01 GMT

null

```

Query the async operation for completion

Use the operations URI to query for the progress of the operation as seen in the following example:

```
armclient get "https://management.azure.com/subscriptions/00000000-0000-0000-0000-  
000000000000/providers/Microsoft.Network/locations/westcentralus/operations/843b1c31-4717-4fdd-b7a6-  
4c786ca9c501?api-version=2016-03-30"
```

While the operation is in progress, the response shows **InProgress** as seen in the following example:

```
{  
    "status": "InProgress"  
}
```

When the operation is complete, the status changes to **Succeeded**.

```
{  
    "status": "Succeeded"  
}
```

The following responses are examples of a typical response returned when querying the results of troubleshooting a Connection.

Retrieve the results

Once the status returned is **Succeeded**, call a GET Method on the operationResult URI to retrieve the results.

```
armclient get "https://management.azure.com/subscriptions/00000000-0000-0000-0000-  
000000000000/providers/Microsoft.Network/locations/westcentralus/operationResults/843b1c31-4717-4fdd-b7a6-  
4c786ca9c501?api-version=2016-03-30"
```

The following responses are examples of a typical response returned when querying the results of troubleshooting a Connection.

```
{
  "startTime": "2017-01-12T14:09:19.1215346-08:00",
  "endTime": "2017-01-12T22:09:23.747Z",
  "code": "UnHealthy",
  "results": [
    {
      "id": "PlatformInActive",
      "summary": "We are sorry, your VPN gateway is in standby mode",
      "detail": "During this time the gateway will not initiate or accept VPN connections with on premises VPN devices or other Azure VPN Gateways. This is a transient state while the Azure platform is being updated.",
      "recommendedActions": [
        {
          "actionText": "If the condition persists, please try resetting your Azure VPN gateway",
          "actionUri": "https://azure.microsoft.com/documentation/articles/vpn-gateway-resetgw-classic/",
          "actionUriText": "resetting the VPN gateway"
        },
        {
          "actionText": "If your VPN Connection isn't up and running by the expected resolution time, contact support",
          "actionUri": "https://azure.microsoft.com/support",
          "actionUriText": "contact support"
        }
      ]
    },
    {
      "id": "NoFault",
      "summary": "This VPN Connection is running normally",
      "detail": "There aren't any known Azure platform problems affecting this VPN Connection",
      "recommendedActions": [
        {
          "actionText": "If you are still experiencing problems with the VPN gateway, please try resetting the VPN gateway.",
          "actionUri": "https://azure.microsoft.com/documentation/articles/vpn-gateway-resetgw-classic/",
          "actionUriText": "resetting VPN gateway"
        },
        {
          "actionText": "If you are experiencing problems you believe are caused by Azure, contact support",
          "actionUri": "https://azure.microsoft.com/support",
          "actionUriText": "contact support"
        }
      ]
    }
  ]
}
```

Understanding the results

The action text provides general guidance on how to resolve the issue. If an action can be taken for the issue, a link is provided with additional guidance. In the case where there is no additional guidance, the response provides the url to open a support case. For more information about the properties of the response and what is included, visit [Network Watcher Troubleshoot overview](#)

For instructions on downloading files from azure storage accounts, refer to [Get started with Azure Blob storage using .NET](#). Another tool that can be used is Storage Explorer. More information about Storage Explorer can be found here at the following link: [Storage Explorer](#)

Next steps

If settings have been changed that stop VPN connectivity, see [Manage Network Security Groups](#) to track down the network security group and security rules that may be in question.

Monitor VPN gateways with Network Watcher troubleshooting

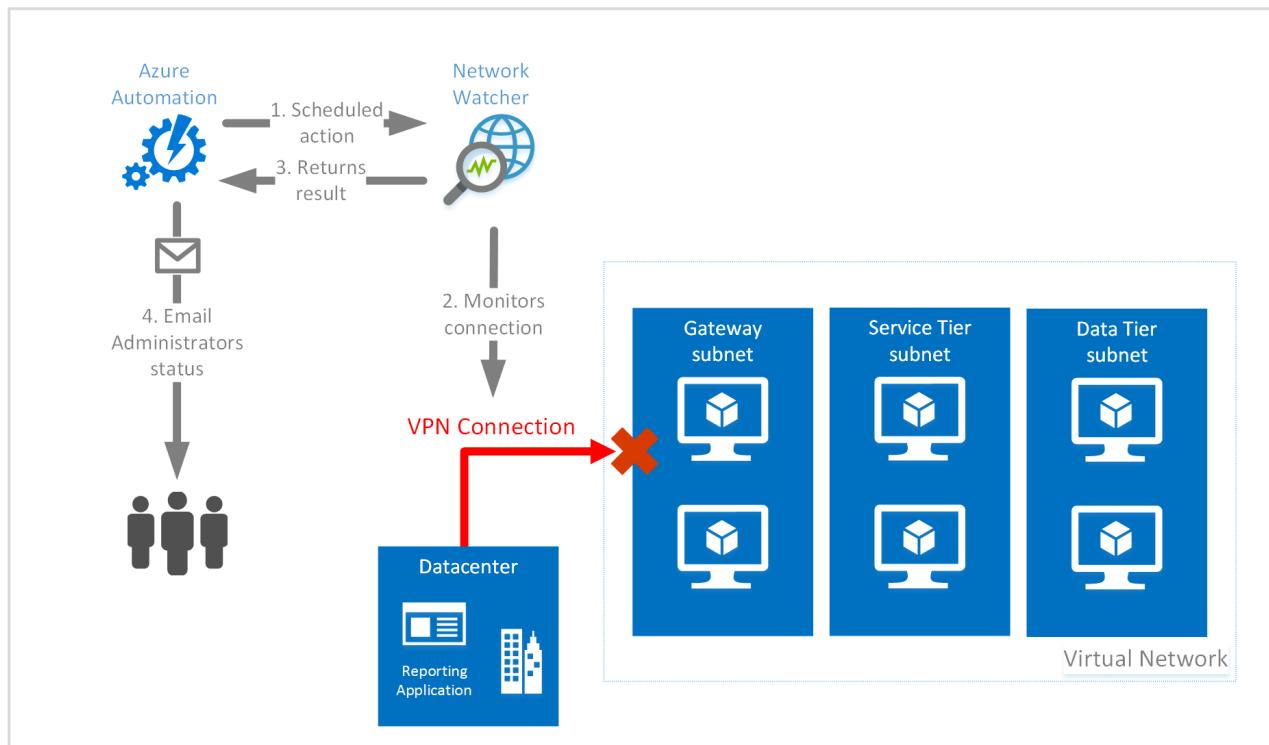
1/28/2020 • 4 minutes to read • [Edit Online](#)

Gaining deep insights on your network performance is critical to provide reliable services to customers. It is therefore critical to detect network outage conditions quickly and take corrective action to mitigate the outage condition. Azure Automation enables you to implement and run a task in a programmatic fashion through runbooks. Using Azure Automation creates a perfect recipe for performing continuous and proactive network monitoring and alerting.

Scenario

The scenario in the following image is a multi-tiered application, with on premises connectivity established using a VPN Gateway and tunnel. Ensuring the VPN Gateway is up and running is critical to the applications performance.

A runbook is created with a script to check for connection status of the VPN tunnel, using the Resource Troubleshooting API to check for connection tunnel status. If the status is not healthy, an email trigger is sent to administrators.



This scenario will:

- Create a runbook calling the `Start-AzureRmNetworkWatcherResourceTroubleshooting` cmdlet to troubleshoot connection status
- Link a schedule to the runbook

Before you begin

Before you start this scenario, you must have the following pre-requisites:

- An Azure automation account in Azure. Ensure that the automation account has the latest modules and also has

the AzureRM.Network module. The AzureRM.Network module is available in the module gallery if you need to add it to your automation account.

- You must have a set of credentials configure in Azure Automation. Learn more at [Azure Automation security](#)
- A valid SMTP server (Office 365, your on-premises email or another) and credentials defined in Azure Automation
- A configured Virtual Network Gateway in Azure.
- An existing storage account with an existing container to store the logs in.

NOTE

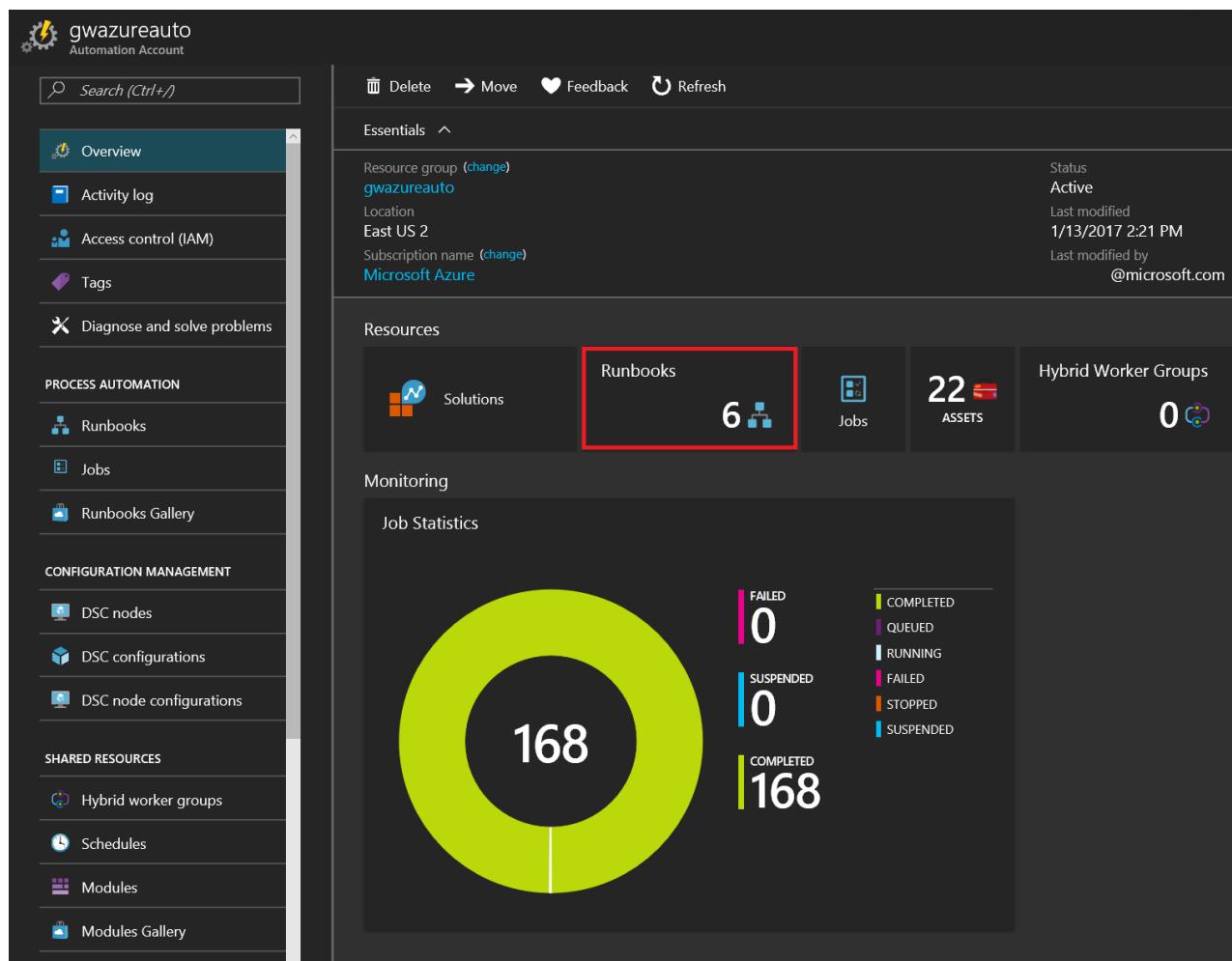
The infrastructure depicted in the preceding image is for illustration purposes and are not created with the steps contained in this article.

Create the runbook

The first step to configuring the example is to create the runbook. This example uses a run-as account. To learn about run-as accounts, visit [Authenticate Runbooks with Azure Run As account](#)

Step 1

Navigate to Azure Automation in the [Azure portal](#) and click **Runbooks**



The screenshot shows the Azure Automation Overview page for the 'gwazureauto' Automation Account. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Process Automation (Runbooks, Jobs, Runbooks Gallery), Configuration Management (DSC nodes, DSC configurations, DSC node configurations), Shared Resources (Hybrid worker groups, Schedules, Modules, Modules Gallery), and Solutions. The main content area displays the 'Essentials' summary, which includes the Resource group (gwazureauto), Location (East US 2), Subscription name (Microsoft Azure), Status (Active), Last modified (1/13/2017 2:21 PM), and Last modified by (@microsoft.com). Below this is the 'Resources' section, which shows 6 Runbooks (highlighted with a red box), 22 Assets, and 0 Hybrid Worker Groups. The 'Monitoring' section features a large circular chart showing Job Statistics: 168 Completed, 0 Failed, 0 Suspended, and 0 Queued. A legend indicates the colors for each status: Completed (light green), Queued (purple), Running (light blue), Failed (pink), Stopped (orange), and Suspended (cyan).

Step 2

Click **Add a runbook** to start the creation process of the runbook.

The screenshot shows the 'Runbooks' list page. At the top, there are three buttons: '+ Add a runbook' (highlighted with a red box), 'Browse gallery', and 'Refresh'. Below the buttons is a search bar with the placeholder 'Search runbooks...'. The main area displays a table with four columns: 'NAME', 'AUTHORING STATUS', 'LAST MODIFIED', and 'TAGS'. The table contains five rows of runbook data:

NAME	AUTHORING STATUS	LAST MODIFIED	TAGS
AzureAutomationTutorial	✓ Published	8/30/2016 1:01 PM	
AzureAutomationTutorialScript	✓ Published	8/30/2016 1:01 PM	
AzureClassicAutomationTutorial	✓ Published	8/30/2016 1:01 PM	
AzureClassicAutomationTutorial...	✓ Published	8/30/2016 1:01 PM	

Step 3

Under **Quick Create**, click **Create a new runbook** to create the runbook.

The screenshot shows the 'Runbooks' list page with an open 'Add Runbook' modal. The modal has two main sections: 'Quick Create' (highlighted with a red box) and 'Import' (with a sub-option 'Import an existing runbook'). The 'Quick Create' section contains the text 'Create a new runbook'.

Step 4

In this step, we give the runbook a name, in the example it is called **Get-VPNGatewayStatus**. It is important to give the runbook a descriptive name, and recommended giving it a name that follows standard PowerShell naming standards. The runbook type for this example is **PowerShell**, the other options are Graphical, PowerShell workflow, and Graphical PowerShell workflow.

The screenshot shows the 'Runbook' creation dialog. It has three main fields:

- * Name: The input field contains 'Get-VPNGatewayStatus1' with a green checkmark indicating it is valid.
- * Runbook type: A dropdown menu is open, showing 'PowerShell' as the selected option.
- Description: An empty text area for entering a description.

Step 5

In this step the runbook is created, the following code example provides all the code needed for the example. The

items in the code that contain <value> need to be replaced with the values from your subscription.

Use the following code as click **Save**

```
# Set these variables to the proper values for your environment
$o365AutomationCredential = "<Office 365 account>"
$fromEmail = "<from email address>"
$toEmail = "<to email address>"
$smtpServer = "<smtp.office365.com>"
$smtpPort = 587
$runAsConnectionName = "<AzureRunAsConnection>"
$subscriptionId = "<subscription id>"
$region = "<Azure region>"
$vpnConnectionName = "<vpn connection name>"
$vpnConnectionResourceGroup = "<resource group name>"
$storageAccountName = "<storage account name>"
$storageAccountResourceGroup = "<resource group name>"
$storageAccountContainer = "<container name>

# Get credentials for Office 365 account
$cred = Get-AutomationPSCredential -Name $o365AutomationCredential

# Get the connection "AzureRunAsConnection "
$servicePrincipalConnection=Get-AutomationConnection -Name $runAsConnectionName

Logging in to Azure...
Connect-AzureRmAccount `

    -ServicePrincipal `

        -TenantId $servicePrincipalConnection.TenantId `

        -ApplicationId $servicePrincipalConnection.ApplicationId `

        -CertificateThumbprint $servicePrincipalConnection.CertificateThumbprint

"Setting context to a specific subscription"
Set-AzureRmContext -SubscriptionId $subscriptionId

$nw = Get-AzurermResource | Where {$_.ResourceType -eq "Microsoft.Network/networkWatchers" -and $_.Location -eq $region }
$networkWatcher = Get-AzureRmNetworkWatcher -Name $nw.Name -ResourceGroupName $nw.ResourceGroupName
$connection = Get-AzureRmVirtualNetworkGatewayConnection -Name $vpnConnectionName -ResourceGroupName $vpnConnectionResourceGroup
$sa = Get-AzureRmStorageAccount -Name $storageAccountName -ResourceGroupName $storageAccountResourceGroup
$storagePath = "$($sa.PrimaryEndpoints.Blob)$($storageAccountContainer)"
$result = Start-AzureRmNetworkWatcherResourceTroubleshooting -NetworkWatcher $networkWatcher -TargetResourceId $connection.Id -StorageId $sa.Id -StoragePath $storagePath

if($result.code -ne "Healthy")
{
    $body = "Connection for $($connection.name) is: $($result.code) `n $($result.results[0].summary) `n View the logs at $($storagePath) to learn more."
    Write-Output $body
    $subject = "$($connection.name) Status"
    Send-MailMessage `

        -To $toEmail `

        -Subject $subject `

        -Body $body `

        -UseSsl `

        -Port $smtpPort `

        -SmtpServer $smtpServer `

        -From $fromEmail `

        -BodyAsHtml `

        -Credential $cred
}
else
{
    Write-Output ("Connection Status is: $($result.code)")
}
```

Step 6

Once the runbook is saved, a schedule must be linked to it to automate the start of the runbook. To start the process, click **Schedule**.

► Start </> View Edit **Schedule** Webhook Delete Export Refresh

Essentials ^

Resource group
[gwazureauto](#)

Account
[gwazureauto](#)

Location
East US 2

Subscription name
[Microsoft Azure](#)

Status
In edit

Runbook type
PowerShell Runbook

Last modified
9/8/2016 10:52 AM

Last modified by
@microsoft.com

Details

	Schedules	Webhooks
Jobs	1	0

Link a schedule to the runbook

A new schedule must be created. Click **Link a schedule to your runbook**.

Schedule Runbook X

Get-VPNGatewayStatus

Schedule

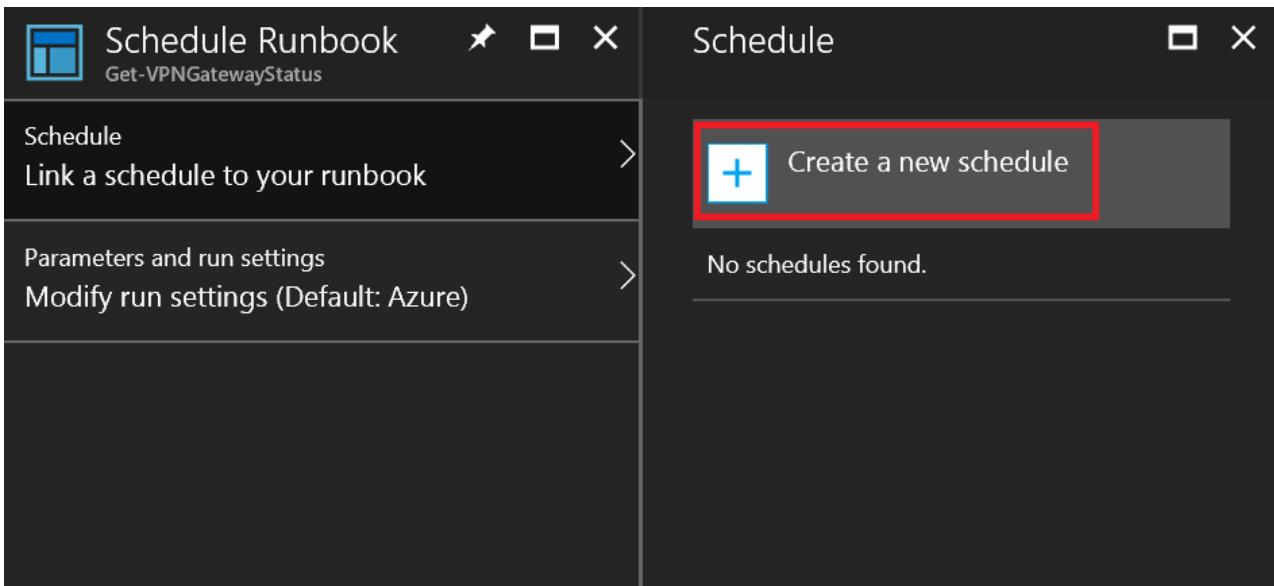
Link a schedule to your runbook

Parameters and run settings

Modify run settings (Default: Azure)

Step 1

On the **Schedule** blade, click **Create a new schedule**



Step 2

On the **New Schedule** blade fill out the schedule information. The values that can be set are in the following list:

- **Name** - The friendly name of the schedule.
- **Description** - A description of the schedule.
- **Starts** - This value is a combination of date, time, and time zone that make up the time the schedule triggers.
- **Recurrence** - This value determines the schedule's repetition. Valid values are **Once** or **Recurring**.
- **Recur every** - The recurrence interval of the schedule in hours, days, weeks, or months.
- **Set Expiration** - The value determines if the schedule should expire or not. Can be set to **Yes** or **No**. A valid date and time are to be provided if yes is chosen.

NOTE

If you need to have a runbook run more often than every hour, multiple schedules must be created at different intervals (that is, 15, 30, 45 minutes after the hour)

 New Schedule ✖️ □ ×

* Name
 ✓

Description

* Starts (i)
2017-02-11 [] 6:36:44 PM

▼

Recurrence
Once Recurring

* Recur every
 Hour ▼

Set expiration
Yes No

Expires
Never

Step 3

Click Save to save the schedule to the runbook.

 Schedule Runbook ✖️ □ ×

Get-VPNGatewayStatus

Schedule >
Every Hour1

Parameters and run settings >
Modify run settings (Default: Azure)

Next steps

Now that you have an understanding on how to integrate Network Watcher troubleshooting with Azure Automation, learn how to trigger packet captures on VM alerts by visiting [Create an alert triggered packet capture](#)

with Azure Network Watcher.

Diagnose on-premises connectivity via VPN gateways

1/28/2020 • 5 minutes to read • [Edit Online](#)

Azure VPN Gateway enables you to create hybrid solution that address the need for a secure connection between your on-premises network and your Azure virtual network. As your requirements are unique, so is the choice of on-premises VPN device. Azure currently supports [several VPN devices](#) that are constantly validated in partnership with the device vendors. Review the device-specific configuration settings before configuring your on-premises VPN device. Similarly, Azure VPN Gateway is configured with a set of [supported IPsec parameters](#) that are used for establishing connections. Currently there is no way for you to specify or select a specific combination of IPsec parameters from the Azure VPN Gateway. For establishing a successful connection between on-premises and Azure, the on-premises VPN device settings must be in accordance with the IPsec parameters prescribed by Azure VPN Gateway. If the settings are incorrect, there is a loss of connectivity and until now troubleshooting these issues was not trivial and usually took hours to identify and fix the issue.

With the Azure Network Watcher troubleshoot feature, you are able to diagnose any issues with your Gateway and Connections and within minutes have enough information to make an informed decision to rectify the issue.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Scenario

You want to configure a site-to-site connection between Azure and on-premises using FortiGate as the on-premises VPN Gateway. To achieve this scenario, you would require the following setup:

1. Virtual Network Gateway - The VPN Gateway on Azure
2. Local Network Gateway - The [on-premises \(FortiGate\) VPN Gateway](#) representation in Azure cloud
3. Site-to-site connection (route based) - [Connection between the VPN Gateway and the on-premises router](#)
4. [Configuring FortiGate](#)

Detailed step by step guidance for configuring a Site-to-Site configuration can be found by visiting: [Create a VNet with a Site-to-Site connection using the Azure portal](#).

One of the critical configuration steps is configuring the IPsec communication parameters, any misconfiguration leads to loss of connectivity between the on-premises network and Azure. Currently Azure VPN Gateways are configured to support the following IPsec parameters for Phase 1. As you can see in the table below, the encryption algorithms supported by Azure VPN Gateway are AES256, AES128, and 3DES.

IKE phase 1 setup

PROPERTY	POLICYBASED	ROUTEBASED AND STANDARD OR HIGH-PERFORMANCE VPN GATEWAY
IKE Version	IKEv1	IKEv2
Diffie-Hellman Group	Group 2 (1024 bit)	Group 2 (1024 bit)

PROPERTY	POLICYBASED	ROUTEBASED AND STANDARD OR HIGH-PERFORMANCE VPN GATEWAY
Authentication Method	Pre-Shared Key	Pre-Shared Key
Encryption Algorithms	AES256 AES128 3DES	AES256 3DES
Hashing Algorithm	SHA1(SHA128)	SHA1(SHA128), SHA2(SHA256)
Phase 1 Security Association (SA) Lifetime (Time)	28,800 seconds	28,800 seconds

As a user, you would be required to configure your FortiGate, a sample configuration can be found on [GitHub](#). Unknowingly you configured your FortiGate to use SHA-512 as the hashing algorithm. As this algorithm is not a supported algorithm for policy-based connections, your VPN connection does work.

These issues are hard to troubleshoot and root causes are often non-intuitive. In this case, you can open a support ticket to get help on resolving the issue. But with Azure Network Watcher troubleshoot API, you can identify these issues on your own.

Troubleshooting using Azure Network Watcher

To diagnose your connection, connect to Azure PowerShell and initiate the

`Start-AzNetworkWatcherResourceTroubleshooting` cmdlet. You can find the details on using this cmdlet at [Troubleshoot Virtual Network Gateway and connections - PowerShell](#). This cmdlet may take up to few minutes to complete.

Once the cmdlet completes, you can navigate to the storage location specified in the cmdlet to get detailed information on about the issue and logs. Azure Network Watcher creates a zip folder that contains the following log files:

- 📄 ConnectionStats.txt
- 📄 CPUStat.txt
- 📄 IKEErrors.txt
- 📄 Scrubbed-wfpdiag.txt
- 📝 wfpdiag.txt.sum
- 📄 wfpdiag.xml

Open the file called IKEErrors.txt and it displays the following error, indicating an issue with on-premises IKE setting misconfiguration.

```
Error: On-premises device rejected Quick Mode settings. Check values.  
based on log : Peer sent NO_PROPOSAL_CHOSEN notify
```

You can get detailed information from the Scrubbed-wfpdiag.txt about the error, as in this case it mentions that there was `ERROR_IPSEC_IKE_POLICY_MATCH` that lead to connection not working properly.

Another common misconfiguration is the specifying incorrect shared keys. If in the preceding example you had specified different shared keys, the IKEErrors.txt shows the following error:

```
Error: Authentication failed. Check shared key .
```

Azure Network Watcher troubleshoot feature enables you to diagnose and troubleshoot your VPN Gateway and Connection with the ease of a simple PowerShell cmdlet. Currently we support diagnosing the following conditions and are working towards adding more condition.

Gateway

FAULT TYPE	REASON	LOG
NoFault	When no error is detected.	Yes
GatewayNotFound	Cannot find Gateway or Gateway is not provisioned.	No
PlannedMaintenance	Gateway instance is under maintenance.	No
UserDrivenUpdate	When a user update is in progress. This could be a resize operation.	No
VipUnResponsive	Cannot reach the primary instance of the Gateway. This happens when the health probe fails.	No
PlatformInactive	There is an issue with the platform.	No
ServiceNotRunning	The underlying service is not running.	No
NoConnectionsFoundForGateway	No Connections exists on the gateway. This is only a warning.	No
ConnectionsNotConnected	None of the Connections are connected. This is only a warning.	Yes
GatewayCPUUsageExceeded	The current Gateway usage CPU usage is > 95%.	Yes

Connection

FAULT TYPE	REASON	LOG
NoFault	When no error is detected.	Yes
GatewayNotFound	Cannot find Gateway or Gateway is not provisioned.	No
PlannedMaintenance	Gateway instance is under maintenance.	No
UserDrivenUpdate	When a user update is in progress. This could be a resize operation.	No
VipUnResponsive	Cannot reach the primary instance of the Gateway. It happens when the health probe fails.	No
ConnectionEntityNotFound	Connection configuration is missing.	No
ConnectionIsMarkedDisconnected	The Connection is marked "disconnected."	No

FAULT TYPE	REASON	LOG
ConnectionNotConfiguredOnGateway	The underlying service does not have the Connection configured.	Yes
ConnectionMarkedStandby	The underlying service is marked as standby.	Yes
Authentication	Preshared Key mismatch.	Yes
PeerReachability	The peer gateway is not reachable.	Yes
IkePolicyMismatch	The peer gateway has IKE policies that are not supported by Azure.	Yes
WfpParse Error	An error occurred parsing the WFP log.	Yes

Next steps

Learn to check VPN Gateway connectivity with PowerShell and Azure Automation by visiting [Monitor VPN gateways with Azure Network Watcher troubleshooting](#)

View the topology of an Azure virtual network

1/28/2020 • 7 minutes to read • [Edit Online](#)

In this article, you learn how to view resources in a Microsoft Azure virtual network, and the relationships between the resources. For example, a virtual network contains subnets. Subnets contain resources, such as Azure Virtual Machines (VM). VMs have one or more network interfaces. Each subnet can have a network security group and a route table associated to it. The topology capability of Azure Network Watcher enables you to view all of the resources in a virtual network, the resources associated to resources in a virtual network, and the relationships between the resources.

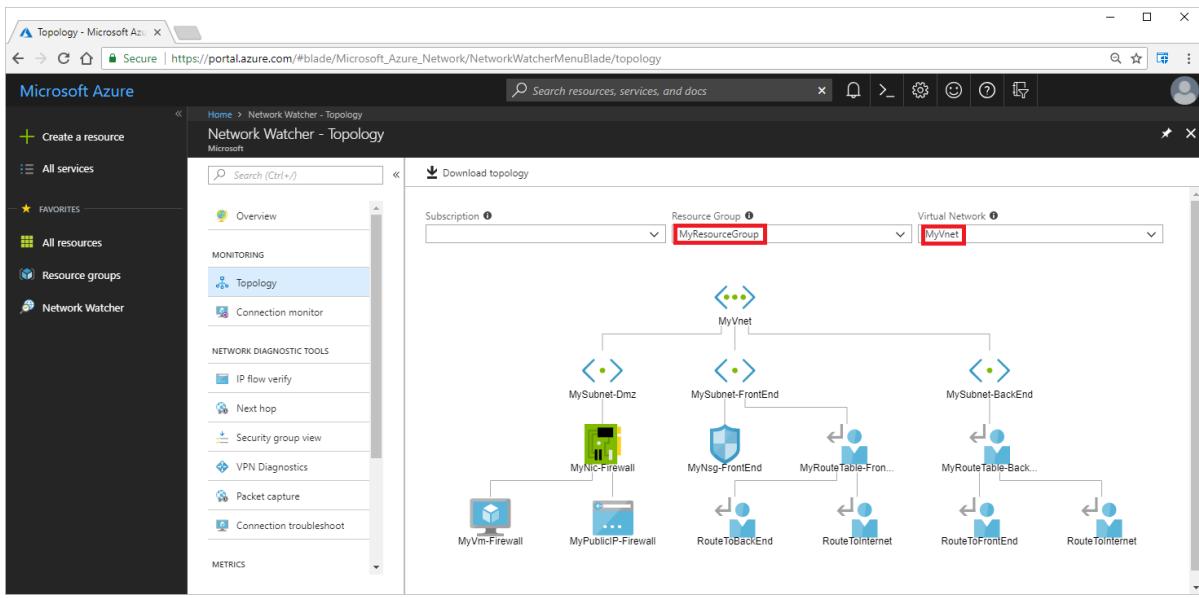
You can use the [Azure portal](#), the [Azure CLI](#), or [PowerShell](#) to view a topology.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

View topology - Azure portal

1. Log into the [Azure portal](#) with an account that has the necessary [permissions](#).
2. On the top, left corner of the portal, select **All services**.
3. In the **All services** filter box, enter *Network Watcher*. When **Network Watcher** appears in the results, select it.
4. Select **Topology**. Generating a topology requires a network watcher in the same region that the virtual network that you want to generate the topology for exists in. If you don't have a network watcher enabled in the region that the virtual network you want to generate a topology for is in, network watchers are automatically created for you in all regions. The network watchers are created in a resource group named **NetworkWatcherRG**.
5. Select a subscription, the resource group of a virtual network you want to view the topology for, and then select the virtual network. In the following picture, a topology is shown for a virtual network named *MyVnet*, in the resource group named *MyResourceGroup*:



As you can see in the previous picture, the virtual network contains three subnets. One subnet has a VM deployed in it. The VM has one network interface attached to it and a public IP address associated to it. The other two subnets have a route table associated to them. Each route table contains two routes. One subnet has a network security group associated to it. Topology information is only shown for resources that are:

- Within the same resource group and region as the *myVnet* virtual network. For example, a network security group that exists in a resource group other than *MyResourceGroup*, isn't shown, even if the network security group is associated to a subnet in the *MyVnet* virtual network.
- Within, or associated to resources within, the *myVnet* virtual network. For example, a network security group that isn't associated to a subnet or network interface in the *myVnet* virtual network isn't shown, even if the network security group is in the *MyResourceGroup* resource group.

The topology shown in the picture is for the virtual network created after deploying the **Route traffic through a network virtual appliance script sample**, which you can deploy using the [Azure CLI](#), or [PowerShell](#).

6. Select **Download topology** to download the image as an editable file, in svg format.

The resources shown in the diagram are a subset of the networking components in the virtual network. For example, while a network security group is shown, the security rules within it are not shown in the diagram. Though not differentiated in the diagram, the lines represent one of two relationships: *Containment* or *associated*. To see the full list of resources in the virtual network, and the type of relationship between the resources, generate the topology with [PowerShell](#) or the [Azure CLI](#).

View topology - Azure CLI

You can run the commands in the steps that follow:

- In the Azure Cloud Shell, by selecting **Try It** at the top right of any command. The Azure Cloud Shell is a free interactive shell that has common Azure tools preinstalled and configured to use with your account.
- By running the CLI from your computer. If you run the CLI from your computer, steps in this article require the Azure CLI version 2.0.31 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install the Azure CLI](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

The account that you use must have the necessary [permissions](#).

1. If you already have a network watcher in the same region as the virtual network that you want to create a topology for, skip to step 3. Create a resource group to contain a network watcher with `az group create`. The following example creates the resource group in the *eastus* region:

```
az group create --name NetworkWatcherRG --location eastus
```

2. Create a network watcher with [az network watcher configure](#). The following example creates a network watcher in the *eastus* region:

```
az network watcher configure \
--resource-group NetworkWatcherRG \
--location eastus \
--enabled true
```

3. View the topology with [az network watcher show-topology](#). The following example views the topology for a resource group named *MyResourceGroup*:

```
az network watcher show-topology --resource-group MyResourceGroup
```

Topology information is only returned for resources that are within the same resource group as the *MyResourceGroup* resource group and the same region as the network watcher. For example, a network security group that exists in a resource group other than *MyResourceGroup*, isn't shown, even if the network security group is associated to a subnet in the *MyVnet* virtual network.

Learn more about the relationships and [properties](#) in the returned output. If you don't have an existing virtual network to view a topology for, you can create one using the [Route traffic through a network virtual appliance](#) script sample. To view a diagram of the topology and download it in an editable file, use the [portal](#).

View topology - PowerShell

You can run the commands in the steps that follow:

- In the Azure Cloud Shell, by selecting **Try It** at the top right of any command. The Azure Cloud Shell is a free interactive shell that has common Azure tools preinstalled and configured to use with your account.
- By running PowerShell from your computer. If you run PowerShell from your computer, this article requires the Azure PowerShell `Az` module. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

The account that you use must have the necessary [permissions](#).

1. If you already have a network watcher in the same region as the virtual network that you want to create a topology for, skip to step 3. Create a resource group to contain a network watcher with [New-AzResourceGroup](#). The following example creates the resource group in the *eastus* region:

```
New-AzResourceGroup -Name NetworkWatcherRG -Location EastUS
```

2. Create a network watcher with [New-AzNetworkWatcher](#). The following example creates a network watcher in the *eastus* region:

```
New-AzNetworkWatcher ` 
-Name NetworkWatcher_eastus ` 
-ResourceGroupName NetworkWatcherRG
```

3. Retrieve a Network Watcher instance with [Get-AzNetworkWatcher](#). The following example retrieves a

network watcher in the East US region:

```
$nw = Get-AzResource `| Where {$_.ResourceType -eq "Microsoft.Network/networkWatchers" -and $_.Location -eq "EastUS" }$networkWatcher = Get-AzNetworkWatcher ` -Name $nw.Name ` -ResourceGroupName $nw.ResourceGroupName
```

4. Retrieve a topology with [Get-AzNetworkWatcherTopology](#). The following example retrieves a topology for a virtual network in the resource group named *MyResourceGroup*:

```
Get-AzNetworkWatcherTopology ` -NetworkWatcher $networkWatcher ` -TargetResourceGroupName MyResourceGroup
```

Topology information is only returned for resources that are within the same resource group as the *MyResourceGroup* resource group and the same region as the network watcher. For example, a network security group that exists in a resource group other than *MyResourceGroup*, isn't shown, even if the network security group is associated to a subnet in the *MyVnet* virtual network.

Learn more about the relationships and [properties](#) in the returned output. If you don't have an existing virtual network to view a topology for, you can create one using the [Route traffic through a network virtual appliance](#) script sample. To view a diagram of the topology and download it in an editable file, use the [portal](#).

Relationships

All resources returned in a topology have one of the following types of relationship to another resource:

RELATIONSHIP TYPE	EXAMPLE
Containment	A virtual network contains a subnet. A subnet contains a network interface.
Associated	A network interface is associated with a VM. A public IP address is associated to a network interface.

Properties

All resources returned in a topology have the following properties:

- **Name:** The name of the resource
- **Id:** The URI of the resource.
- **Location:** The Azure region that the resource exists in.
- **Associations:** A list of associations to the referenced object. Each association has the following properties:
 - **AssociationType:** References the relationship between the child object and the parent. Valid values are *Contains* or *Associated*.
 - **Name:** The name of the referenced resource.
 - **ResourceId:** - The URI of the resource referenced in the association.

Next steps

- Learn how to [diagnose a network traffic filter problem to or from a VM](#) using Network Watcher's IP flow verify capability

- Learn how to [diagnose a network traffic routing problem from a VM](#) using Network Watcher's next hop capability

View relative latency to Azure regions from specific locations

1/28/2020 • 5 minutes to read • [Edit Online](#)

WARNING

This feature is currently in preview and still being tested for stability.

In this tutorial, learn how to use the Azure [Network Watcher](#) service to help you decide what Azure region to deploy your application or service in, based on your user demographic. Additionally, you can use it to help evaluate service providers' connections to Azure.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Create a network watcher

If you already have a network watcher in at least one Azure [region](#), you can skip the tasks in this section. Create a resource group for the network watcher. In this example, the resource group is created in the East US region, but you can create the resource group in any Azure region.

```
New-AzResourceGroup -Name NetworkWatcherRG -Location eastus
```

Create a network watcher. You must have a network watcher created in at least one Azure region. In this example, a network watcher is created in the East US Azure region.

```
New-AzNetworkWatcher -Name NetworkWatcher_eastus -ResourceGroupName NetworkWatcherRG -Location eastus
```

Compare relative network latencies to a single Azure region from a specific location

Evaluate service providers, or troubleshoot a user reporting an issue such as "the site was slow," from a specific location to the azure region where a service is deployed. For example, the following command returns the average relative Internet service provider latencies between the state of Washington in the United States and the West US 2 Azure region between December 13-15, 2017:

```
Get-AzNetworkWatcherReachabilityReport ` 
-NetworkWatcherName NetworkWatcher_eastus ` 
-ResourceGroupName NetworkWatcherRG ` 
-Location "West US 2" ` 
-Country "United States" ` 
-State "washington" ` 
-StartTime "2017-12-13" ` 
-EndTime "2017-12-15"
```

NOTE

The region you specify in the previous command doesn't need to be the same as the region you specified when you retrieved the network watcher. The previous command simply requires that you specify an existing network watcher. The network watcher can be in any region. If you specify values for `-Country` and `-State`, they must be valid. The values are case-sensitive. Data is available for a limited number of countries/regions, states, and cities. Run the commands in [View available countries/regions, states, cities, and providers](#) to view a list of available countries/regions, cities, and states to use with the previous command.

WARNING

You must specify a date within the past 30 days for `-StartTime` and `-EndTime`. Specifying a prior date will result in no data being returned.

The output from the previous command follows:

```

AggregationLevel : State
ProviderLocation : {
    "Country": "United States",
    "State": "washington"
}
ReachabilityReport : [
    {
        "Provider": "Qwest Communications Company, LLC - ASN 209",
        "AzureLocation": "West US 2",
        "Latencies": [
            {
                "TimeStamp": "2017-12-14T00:00:00Z",
                "Score": 92
            },
            {
                "TimeStamp": "2017-12-13T00:00:00Z",
                "Score": 92
            }
        ]
    },
    {
        "Provider": "Comcast Cable Communications, LLC - ASN 7922",
        "AzureLocation": "West US 2",
        "Latencies": [
            {
                "TimeStamp": "2017-12-14T00:00:00Z",
                "Score": 96
            },
            {
                "TimeStamp": "2017-12-13T00:00:00Z",
                "Score": 96
            }
        ]
    }
]

```

In the returned output, the value for **Score** is the relative latency across regions and providers. A score of 1 is the worst (highest) latency, whereas 100 is the lowest latency. The relative latencies are averaged for the day. In the previous example, while it's clear that the latencies were the same both days and that there is a small difference between the latency of the two providers, it's also clear that the latencies for both providers are low on the 1-100 scale. While this is expected, since the state of Washington in the United States is physically close to the West US 2 Azure region, sometimes results aren't as expected. The larger the date range you specify, the more you can average latency over time.

Compare relative network latencies across Azure regions from a specific location

If, instead of specifying the relative latencies between a specific location and a specific Azure region using `-Location`, you wanted to determine the relative latencies to all Azure regions from a specific physical location, you can do that too. For example, the following command helps you evaluate what azure region to deploy a service in if your primary users are Comcast users located in Washington state:

```

Get-AzNetworkWatcherReachabilityReport ` 
    -NetworkWatcherName NetworkWatcher_eastus ` 
    -ResourceGroupName NetworkWatcherRG ` 
    -Provider "Comcast Cable Communications, LLC - ASN 7922" ` 
    -Country "United States" ` 
    -State "washington" ` 
    -StartTime "2017-12-13" ` 
    -EndTime "2017-12-15"

```

NOTE

Unlike when you specify a single location, if you don't specify a location, or specify multiple locations, such as "West US2", "West US", you must specify an Internet service provider when running the command.

View available countries/regions, states, cities, and providers

Data is available for specific Internet service providers, countries/regions, states, and cities. To view a list of all available Internet service providers, countries/regions, states, and cities, that you can view data for, enter the following command:

```
Get-AzNetworkWatcherReachabilityProvidersList -NetworkWatcherName NetworkWatcher_eastus -ResourceGroupName NetworkWatcherRG
```

Data is only available for the countries/regions, states, and cities returned by the previous command. The previous command requires you to specify an existing network watcher. The example specified the *NetworkWatcher_eastus* network watcher in a resource group named *NetworkWatcherRG*, but you can specify any existing network watcher. If you don't have an existing network watcher, create one by completing the tasks in [Create a network watcher](#).

After running the previous command, you can filter the output returned by specifying valid values for **Country**, **State**, and **City**, if desired. For example, to view the list of Internet service providers available in Seattle, Washington, in the United States, enter the following command:

```
Get-AzNetworkWatcherReachabilityProvidersList ` 
    -NetworkWatcherName NetworkWatcher_eastus ` 
    -ResourceGroupName NetworkWatcherRG ` 
    -City Seattle ` 
    -Country "United States" ` 
    -State washington
```

WARNING

The value specified for **Country** must be upper and lowercase. The values specified for **State** and **City** must be lowercase. The values must be listed in the output returned after running the command with no values for **Country**, **State**, and **City**. If you specify the incorrect case, or specify a value for **Country**, **State**, or **City** that is not in the output returned after running the command with no values for these properties, the returned output is empty.

2 minutes to read

Frequently asked questions (FAQ) about Azure Network Watcher

2/19/2020 • 4 minutes to read • [Edit Online](#)

The [Azure Network Watcher](#) service provides a suite of tools to monitor, diagnose, view metrics, and enable or disable logs for resources in an Azure virtual network. This article answers common questions about the service.

General

What is Network Watcher?

Network Watcher is designed to monitor and repair the network health of IaaS (Infrastructure-as-a-Service) components, which includes Virtual Machines, Virtual Networks, Application Gateways, Load balancers, and other resources in an Azure virtual network. It is not a solution for monitoring PaaS (Platform-as-a-Service) infrastructure or getting web/mobile analytics.

What tools does Network Watcher provide?

Network Watcher provides three major sets of capabilities

- Monitoring
 - [Topology view](#) shows you the resources in your virtual network and the relationships between them.
 - [Connection Monitor](#) allows you to monitor connectivity and latency between a VM and another network resource.
 - [Network performance monitor](#) allows you to monitor connectivity and latencies across hybrid network architectures, Expressroute circuits, and service/application endpoints.
- Diagnostics
 - [IP Flow Verify](#) allows you to detect traffic filtering issues at a VM level.
 - [Next Hop](#) helps you verify traffic routes and detect routing issues.
 - [Connection Troubleshoot](#) enables a one-time connectivity and latency check between a VM and another network resource.
 - [Packet Capture](#) enables you to capture all traffic on a VM in your virtual network.
 - [VPN Troubleshoot](#) runs multiple diagnostics checks on your VPN gateways and connections to help debug issues.
- Logging
 - [NSG Flow Logs](#) allows you to log all traffic in your [Network Security Groups \(NSGs\)](#)
 - [Traffic Analytics](#) processes your NSG Flow Log data enabling you to visualize, query, analyze, and understand your network traffic.

For more detailed information, see the [Network Watcher overview page](#).

How does Network Watcher pricing work?

Visit the [Pricing page](#) for Network Watcher components and their pricing.

Which regions is Network Watcher supported/available in?

You can view the latest regional availability on the [Azure Service availability page](#)

Which permissions are needed to use Network Watcher?

See the list of [RBAC permissions required to use Network Watcher](#). For deploying resources, you need contributor permissions to the NetworkWatcherRG (see below).

How do I enable Network Watcher?

The Network Watcher service is [enabled automatically](#) for every subscription.

What is the Network Watcher deployment model?

The Network Watcher parent resource is deployed with a unique instance in every region. Naming format: NetworkWatcher_RegionName. Example: NetworkWatcher_centralus is the Network Watcher resource for the "Central US" region.

What is the NetworkWatcherRG?

Network Watcher resources are located in the hidden **NetworkWatcherRG** resource group which is created automatically. For example, the NSG Flow Logs resource is a child resource of Network Watcher and is enabled in the NetworkWatcherRG.

Why do I need to install the Network Watcher extension?

The Network Watcher extension is required for any feature that needs to generate or intercept traffic from a VM.

Which features require the Network Watcher extension?

The Packet Capture, Connection Troubleshoot and Connection Monitor features need the Network Watcher extension to be present.

What are resource limits on Network Watcher?

See the [Service limits](#) page for all limits.

Why is only one instance of Network Watcher allowed per region?

Network Watcher just needs to be enabled once for a subscription for its features to work, this is a not a service limit.

How can I manage the Network Watcher Resource?

The Network Watcher resource represents the backend service for Network Watcher and is fully managed by Azure. Customers do no need to manage it. Operations like move are not supported on the resource. However, [the resource can be deleted](#).

NSG Flow Logs

What does NSG Flow Logs do?

Azure network resources can be combined and managed through [Network Security Groups \(NSGs\)](#). NSG Flow Logs enable you to log 5-tuple flow information about all traffic through your NSGs. The raw flow logs are written to an Azure Storage account from where they can be further processed, analyzed, queried, or exported as needed.

How do I use NSG Flow Logs with a Storage account behind a firewall?

To use a Storage account behind a firewall, you have to provide an exception for Trusted Microsoft Services to access your storage account:

- Navigate to the storage account by typing the storage account's name in the global search on the portal or from the [Storage Accounts page](#)
- Under the **SETTINGS** section, select **Firewalls and virtual networks**
- In "Allow access from", select **Selected networks**. Then under **Exceptions**, tick the box next to "**Allow trusted Microsoft services to access this storage account**"
- If it is already selected, no change is needed.
- Locate your target NSG on the [NSG Flow Logs overview page](#) and enable NSG Flow Logs with the above storage account selected.

You can check the storage logs after a few minutes, you should see an updated TimeStamp or a new JSON file created.

How do I use NSG Flow Logs with a Storage account behind a Service Endpoint?

NSG Flow Logs are compatible with Service Endpoints without requiring any extra configuration. Please see the [tutorial on enabling Service Endpoints](#) in your virtual network.

What is the difference between flow logs versions 1 & 2?

Flow Logs version 2 introduces the concept of *Flow State* & stores information about bytes and packets transmitted. [Read more.](#)

Next Steps

- Head over to our [documentation overview page](#) for some tutorials to get you started with Network Watcher.