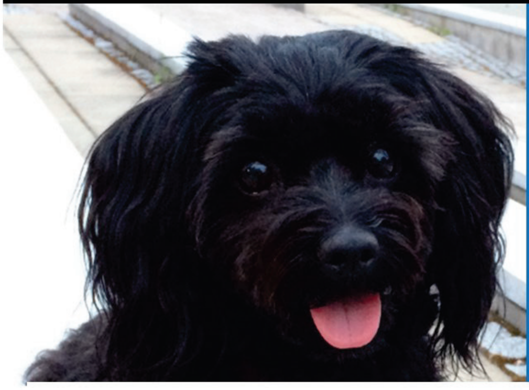
The Network Times
Handbook Series – Part VIII

Azure Networking Fundamentals

A Practical Guide to Understand How to Build a Virtual Datacenter into the Azure Cloud



Virtual Networks (VNet) are private routing instance for customers on **Azure Cloud**. On-premise sites can be connected to VNets by creating an **S2S VPN** tunnel to **VGW** or **vHub**, part of Azure's **vWAN** service. Customers may also choose the private connection to Azure resources using the Azure **ExpressRoute** service. vHub supports **Inter-VNe**t connection by default, while VGW supports **VNet-to-VNet VPN**. **VNet Peering** is the simplest Inter-Vnet solution. Among these services, this book explains various other network-related Azure Cloud services like **NSG** and **NAT gateway**.

Toni Pasanen, CCIE#28158

Azure Networking Fundamentals

A Practical Guide to Understand How to Build a Virtual Datacenter into the Azure Cloud

Toni Pasanen, CCIE 28158

Published – 6 August 2022 (The First Print)

About the Editor:

Toni Pasanen. CCIE No. 28158 (RS), Distinguished Engineer at Fujitsu Finland. Toni started his IT career in 1998 at Tieto, where he worked as a Service Desk Specialist moving via the LAN team to the Data Center team as a 3rd. Level Network Specialist. Toni joined Teleware (Cisco Learning partner) in 2004, where he spent two years teaching network technologies focusing on routing/switching and MPLS technologies. Toni joined Tieto again in 2006, where he spent the next six years as a Network Architect before joining Fujitsu. In his current role, Toni works closely with customers helping them in selecting the right network solutions not only from the technology perspective but also from the business perspective. He is also the author of books:

Virtual Extensible LAN – VXLAN:
A Practical Guide to VXLAN Solution –2019

LISP Control-Plane in Campus Fabric:
A Practical Guide to Understand the Operation of Campus Fabric– 2020

VXLAN Fabric with BGP EVPN Control-Plane:
Design Considerations – 2020

Object-Based Approach to Cisco ACI:
The Logic Behind the Application Centric Infrastructure - 2020.

Cisco SD-WAN:
A Practical Guide to Understand the Basics of Cisco Viptela Based SD-WAN Solution- 2021.

Network Virtualization – Free eBook:
LISP, OMP, and BGP EVPN Operation and Interaction
(August 2021)

AWS Networking Fundamentals:
A Practical Guide to Understand How to Build a Virtual Datacenter into the AWS Cloud-2021

About This Book

Chapter 1 - Azure Virtual Network (VNet):

This chapter starts by introducing Azure Geographies, Regions, and Availability Zones. Then it discusses Subscriptions and Resources Groups. After these basic Azure building block introductions, this chapter focuses on Virtual Networks (VNet). It shows how to create VNet using Azure Portal and Azure Resource Manager tool ARM. This chapter also explains how we can use tags to give informative descriptions to our resources.

Chapter 2 – Network Security Group (NSG):

This chapter introduces three NSG scenarios. The first section explains the *NSG to NIC association*. The second section describes the *NSG-Subnet association*. The last part of the chapter introduces an *Application Security Group (ASG)*, which we are using to form a logical VM group. Besides, this chapter explains how we create VM and allow an SSH connection from the external network.

Chapter 3 – Internet Access with VM-Specific Public IP:

We can assign a Public IP address, an *Instance Level Public IP (ILPIP)* for Azure resources using either ***the Static*** or ***Dynamic*** method. Each VMs has an implicit, default outbound access to the Internet because Azure assigns a default *Outbound Access IP address (OPIP)*. The OPIP address is not owned by a customer and it may change. Microsoft recommends disabling the default outbound access. The recommended method is using a NAT Gateway or Firewall.

Chapter 4 - Virtual Network NAT Service - NAT Gateway:

This chapter explains how VMs without ILPIP can use Azure *Virtual Network NAT service* provided by the Azure *NAT Gateway* for the Internet connection. NAT GW is a managed distributed service permitting an egress-only Internet connection from VMs using TCP or UDP transport layer protocols.

Chapter 5 – Hybrid Cloud – Site-to-Site VPN (S2S VPN):

A Hybrid Cloud is a model where we split application-specific workloads across the public and private clouds. This chapter introduces Azure's hybrid cloud solution using Site-to-Site (S2S) Active-Standby VPN connection between Azure and on-prem DC.

Chapter 6 – Hybrid Cloud – S2S VPN with BGP:

This chapter shows how to enable dynamic routing using BGP between on-prem DC and Azure VNet.

Chapter 7 – VNet-to-VNet VPN:

This chapter explains how we can use a VPN GW for an Inter-VNet connection.

Chapter 8 – VNet Peering:

This chapter introduces an Azure VNet Peering solution. VNet peering creates bidirectional IP connections between peered VNets. VNet peering links can be established within and across Azure regions and between VNets under the different Azure subscriptions or tenants. The unencrypted data path over peer links stays within Azure private infrastructure.

Chapter 9 – Transit VNet – Hub and Spoke:

VNet Peering alone is a non-transitive solution, which only allows IP connection between peered VNets. However, we can build a design where we configure one of the VNets as a Hub VNet, which has a VNet peering with Spoke VNets. Then, we set up a gateway VGW into the Hub VNet, which routes traffic between multiple Spoke VNets.

Chapter 10 – Hybrid Cloud – Routing Studies:

This chapter discusses route propagations between on-prem DC and Azure Virtual Network in a Hybrid Cloud solution.

Chapter 11 – Virtual WAN Part I – S2S VPN and VNet Connection:

This chapter introduces Azure *Virtual WAN (vWAN)* service. It offers a single deployment, management, and monitoring pane for connectivity services such as Inter-VNet, Site-to-Site VPN and Express Route. In this chapter we are focusing S2S VPN and VNet connections.

Chapter 12 - Virtual WAN Part II – VNet Segmentation:

VNets and VPN/ExpressRoute connections are associated with vHub's Default Route Table, which allows both VNet-to-VNet and VNet-to-Remote Site IP connectivity. This chapter explains how we can isolate vnet-swe3 from vnet-swe1 and vnet-swe2 using VNet-specific vHub Route Tables (RT), still allowing VNet-to-VPN Site connection.

Chapter 13 - Virtual WAN Part III – Global Transit Network:

This chapter introduces a global transit network architecture based on Azure vWAN.

Chapter 14 – Hybrid Cloud – Express Route:

ExpressRoute (ER) is an Azure service that offers a logical, private circuit between *Customer Edge (CE)* router(s) and the pair of *Microsoft Enterprise Edge (MSEE)* devices. MSEEs are in one of the Azure selected *Co-locations (Colo) - Carrier Neutral Facilities* having a connection straight to the Microsoft Backbone network and *ExpressRoute Provider (ERP)*.

Chapter 15 – Load Balancer & NAT - Ananta

This chapter introduces Azure's cloud scale load-balancing/NAT service Ananta. It consists of three building blocks. Ananta Manager (AM) is a highly available centralized Control Plane. Multiplexer (MUX) pool has a set of MUXs, which offer a Load Balancing service for inbound traffic. Ananta Host Agent (HA) is like a virtual switch within the Hypervisor's VMSwitch performing NAT functions.

Disclaimers

Table of Contents

Chapter 1: Azure Virtual Network (VNet)

Introduction

Geography, Region, and Availability Zone

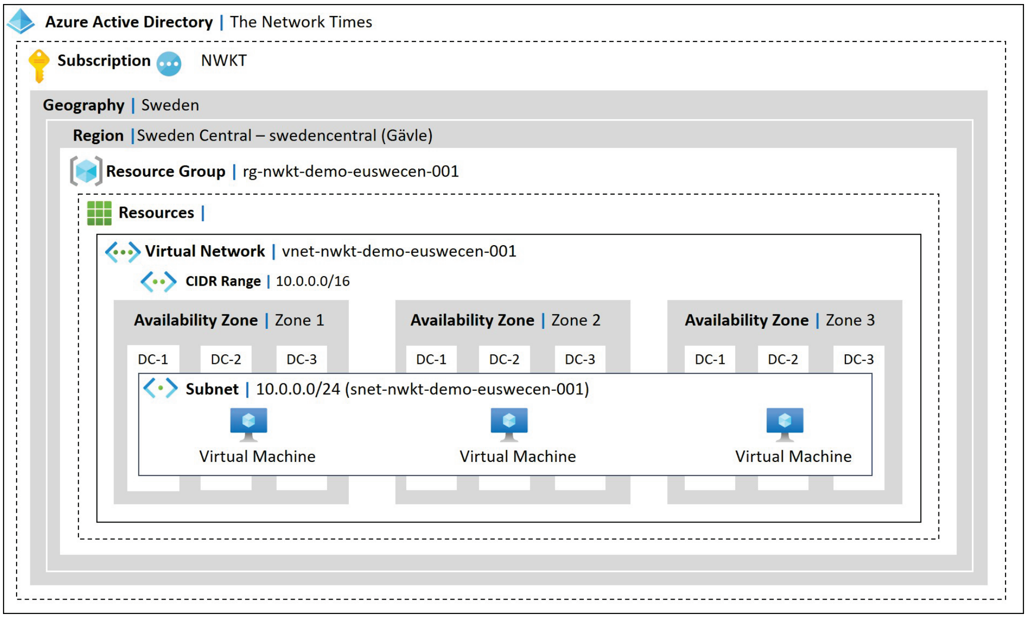
Geography, *Region*, and *Availability Zones* in figure 1-1 are related to Microsoft Datacenters and locations. Our example Geography Sweden comprises two Regions: Sweden Central and Sweden South. Sweden Central in Gävle is the primary Region. It hosts Microsoft Azure, Office 365, and Microsoft Dynamics 365, which are Microsoft's Software as a Service (SaaS) services. The second one, Sweden South (not shown in figure 1-1) in Staffanstorp, can only be used as a Disaster Recovery site for Sweden Central customers. Each Azure Region comprises three Availability Zones (AZ), which all have one or more DCs with dedicated DC infrastructure. AZs are connected with high-speed connections granting less than 2ms Round-Trip Time (RTT) between AZs. The Regional service is not affected if one of its three AZ is unreachable or down.

Resources and Resource Group

The *Virtual Network* (VNet), our virtual Datacenter, is a *Resource* under the *Networking Resource* category. Each VNet has an associated CIDR Range (Classless Inter-Domain Routing) from which we can allocate subnets. If we create a VNet-to-VNet connection, we need to use VNet-specific unique CIDR Ranges. VNets are Regional and cannot span between Regions. Subnets, in turn, can spread across regional Availability Zones. When creating a resource, we attach it to the *Resource Group*. We can group our application-specific resources under the same Resource Group and then tear down the whole application by deleting its logical Resource Group.

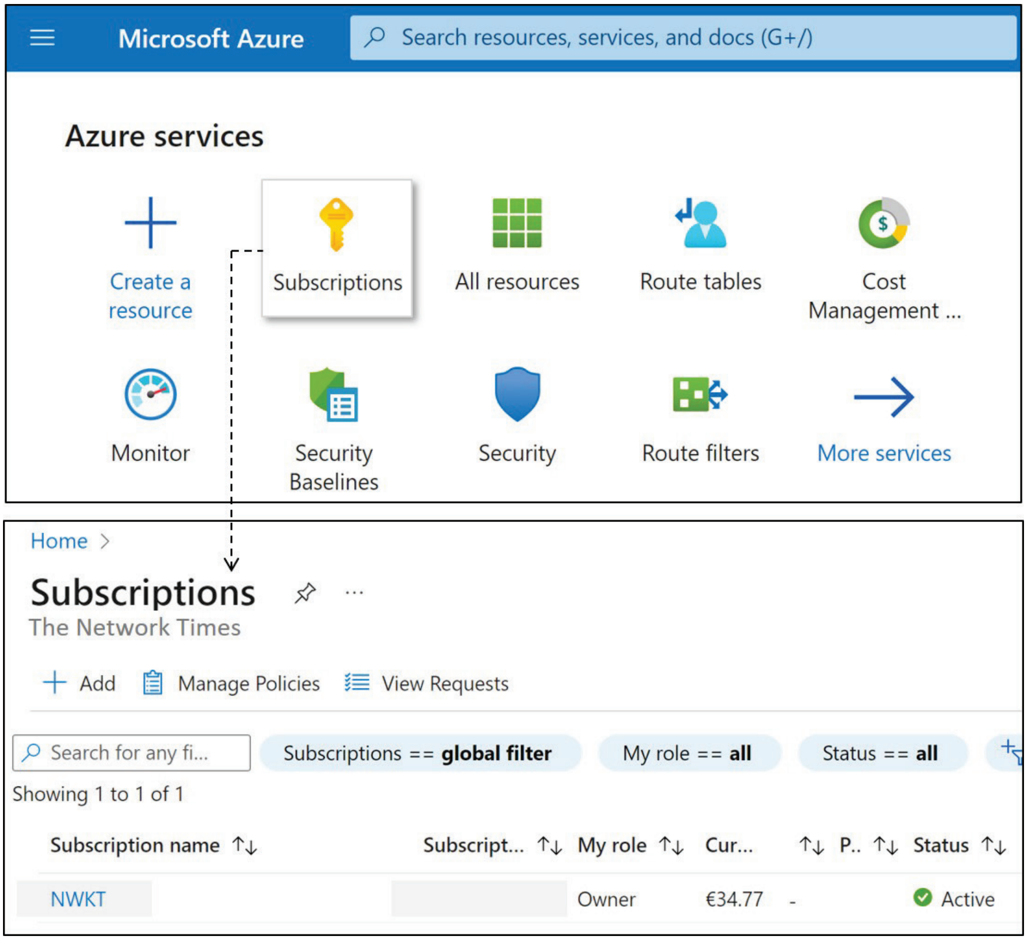
Azure Active Directory and Subscription

Microsoft Azure, Office 365, and Microsoft Dynamics 365 are Software as a Service (SaaS) products. *Azure Active Directory* (AAD) is a system where we define access policies to these services. Our example AAD is The Network Times, where the administrator has created a *Subscription* NWKT for me. Subscription is like my playground. As an Owner, I can manage Resource Groups, Resources, and services under my subscription. That said, when we create any resource, we associate it with a Resource group under our subscription.


Azure Active Directory | The Network Times
Subscription NWKT
Geography | Sweden
Region | Sweden Central – swedencentral (Gävle)
Resource Group | rg-nwkt-demo-euswecen-001
Resources |
Virtual Network | vnet-nwkt-demo-euswecen-001
CIDR Range | 10.0.0.0/16
Availability Zone | Zone 1
Availability Zone | Zone 2
Availability Zone | Zone 3
DC-1
DC-2
DC-3
DC-1
DC-2
DC-3
DC-1
DC-2
DC-3
Subnet | 10.0.0.0/24 (snet-nwkt-demo-euswecen-001)
Virtual Machine
Virtual Machine
Virtual Machine
Figure 1-1: *Virtual Network (VNet) Basic Building Blogs.*

Create Resource Group with Azure Portal

Navigate to Azure Portal (portal.azure.com). Log in with your user credentials. After successful login, your Azure home page opens. Select the Subscriptions icon. You can see the list of your Subscriptions in the *Subscription* view. In our example, we have only one Subscription NWKT. Next, click the NWKT Subscription hyperlink, and you'll be forwarded to the Subscription page.

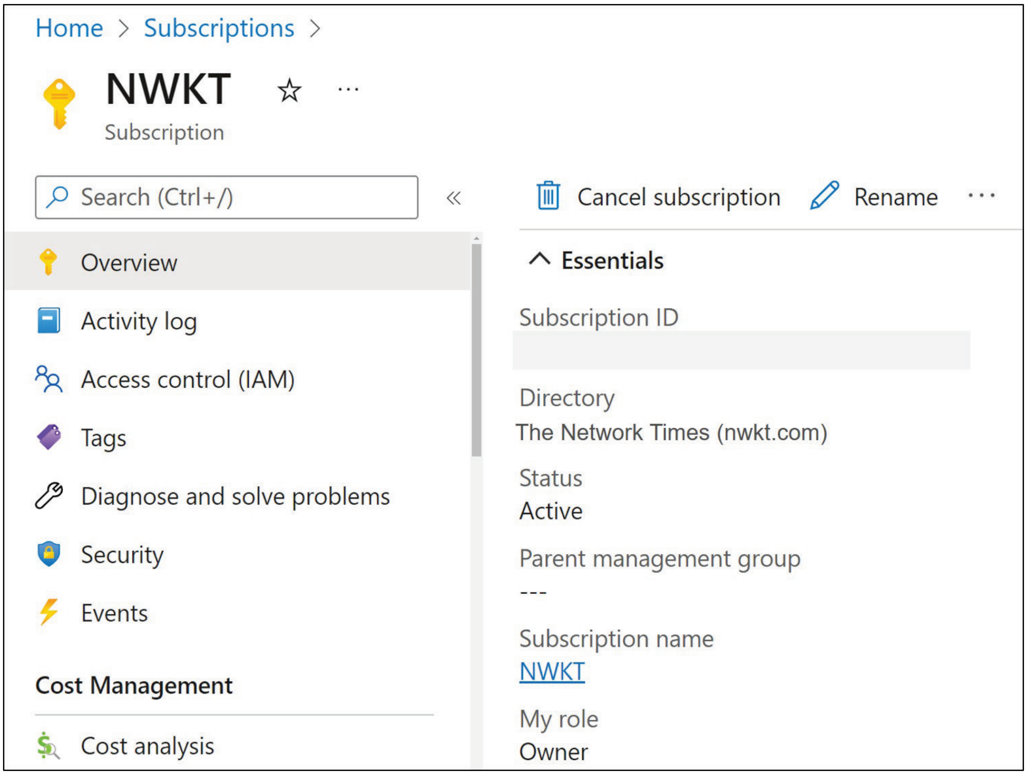


Showing 1 to 1 of 1

Subscription name	Subscript...	My role	Cur...	P..	Status
NWKT		Owner	€34.77	-	Active

Figure 1-2: *Create Resource Group – Open Your Subscription.*

The *Essentials* section under the *Overview* option shows that I am the Owner of the Subscription NWKT and that the Subscription state is Active.


Home > Subscriptions >

NWKT ☆ ···

Subscription

Search (Ctrl+/) «

Cancel subscription Rename ···

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Security
Events

Cost Management

Cost analysis

^ Essentials

Subscription ID

Directory
The Network Times (nwkt.com)

Status
Active

Parent management group

Subscription name
NWKT

My role
Owner

Figure 1-3: *Create Resource Group – Subscription Overview.*

Scroll down to the *Settings* section in the left pane and select the *Resource Group* option. Then click the *Create* button on the taskbar.

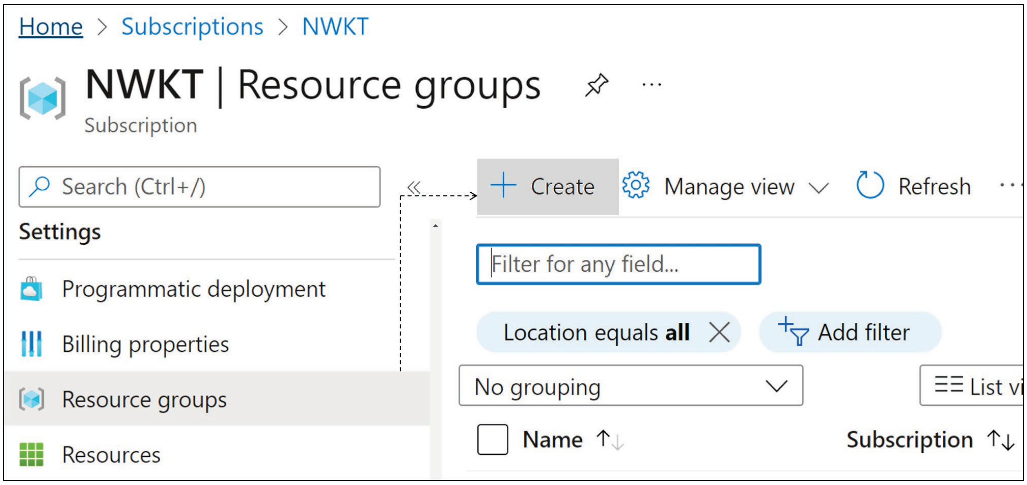
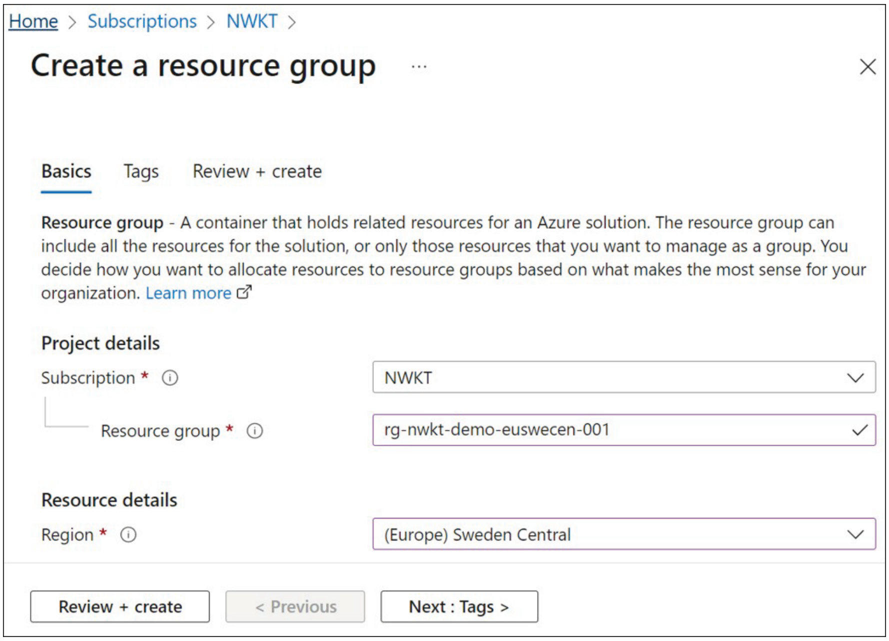


Figure 1-4: *Create Resource Group.*

Open the *Basics* tab in the *Create a resource group* window. First, Fill in the *Project details* information. Select your Subscription from the drop-down menu and give the name to Resource Group. I have used the naming convention recommended by Microsoft, which divides the name into five sections: Resource Type - Application – Environment - Region - Instance. I have also used recommended resource type abbreviation guidelines. I have named our example Resource Group as *rg-nwkt-demo-euswecen-001*. Next, select the Azure region where you want to launch the resource group. If you like to add tags to the resource group, click the *Next: Tags* button or select the *Tags* tab.


Home > Subscriptions > NWKT >

Create a resource group

Basics | Tags | Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Learn more

Project details

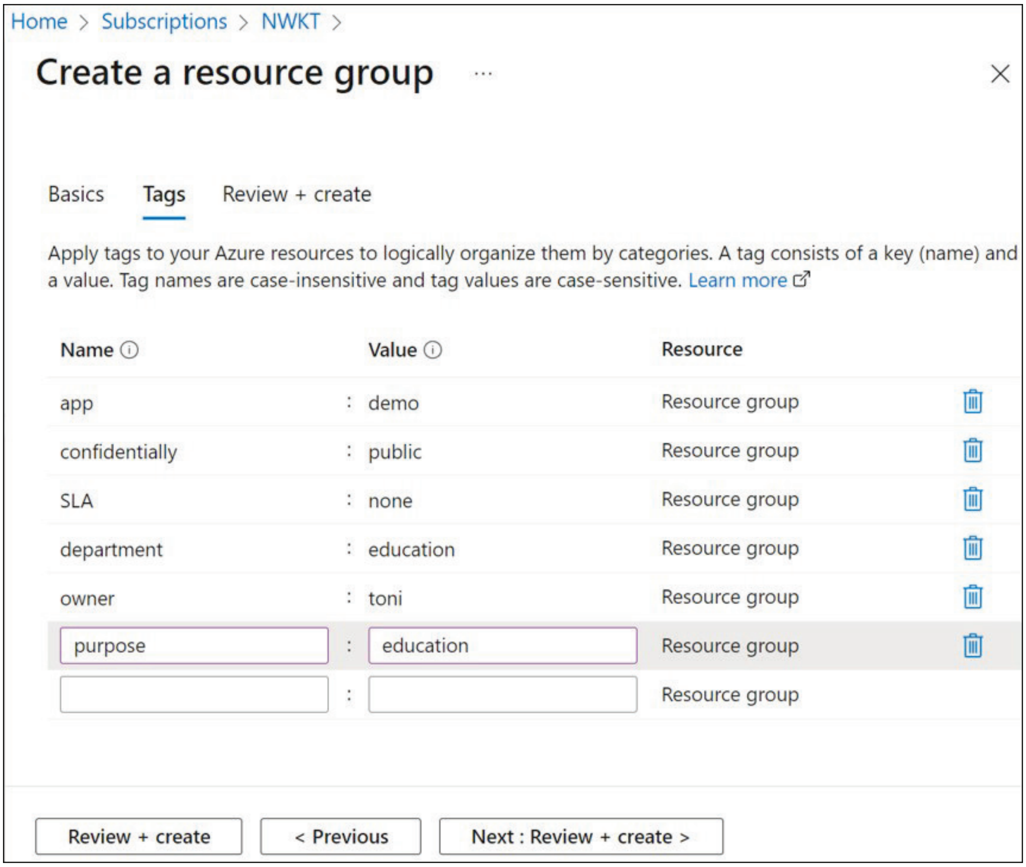
Subscription *	NWKT
Resource group *	rg-nwkt-demo-euswecen-001

Resource details

Region *	(Europe) Sweden Central

Figure 1-5: *Create Resource Group – Basic information.*

We can group application-specific services and resources under the same resource group. However, we might have a dedicated resource group for the development, testing, and production environment. Using *Tags*, you can give an informative description to resource groups or other Azure resources. You can also view resource-specific billing information based on tags. I have added six Tags to our example resource group: application name (=demo), confidentially (=public), SLA (=none), department (=education), the owner (Toni), and purpose (=education). Note that once you have created tags, you can use them with any of your Azure resources groups or resources. When you have created tags, click the *Next: Review + create* button or select the *Review + create* tab.

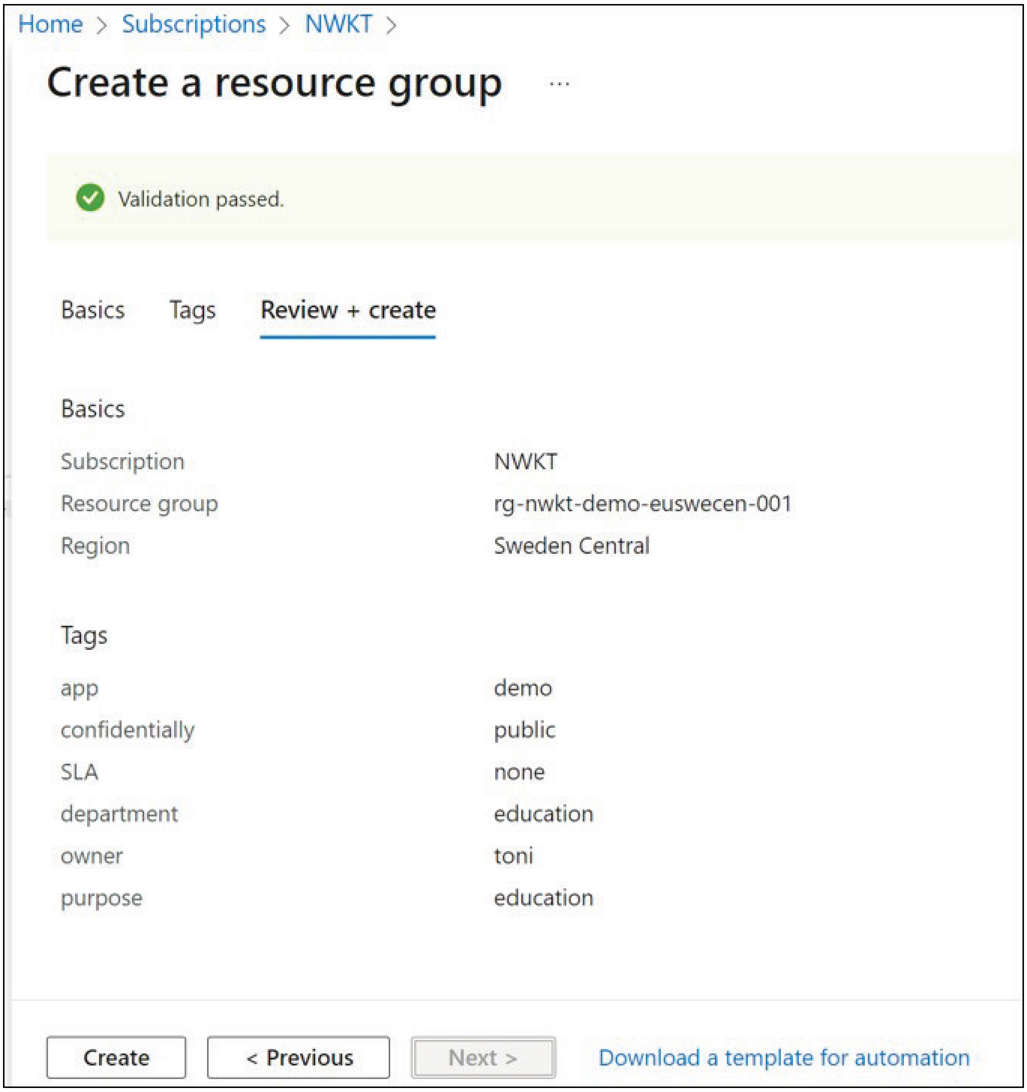


Apply tags to your Azure resources to logically organize them by categories. A tag consists of a key (name) and a value. Tag names are case-insensitive and tag values are case-sensitive. Learn more

Name		Value	Resource
app	:	demo	Resource group
confidentially	:	public	Resource group
SLA	:	none	Resource group
department	:	education	Resource group
owner	:	toni	Resource group
purpose	:	education	Resource group
	:		Resource group

Figure 1-6: *Create Resource Group – Tagging.*

Figure 1-7 shows the Review window. You can download template and parameters files by clicking the hyperlink *Download a template for automation*. I will go through the various automation options later. Click the *Create* button for deploying your resource group.


Home > Subscriptions > NWKT >
Create a resource group ...
Validation passed.
Basics
Tags
Review + create
Create
< Previous
Next >
Download a template for automation

Basics

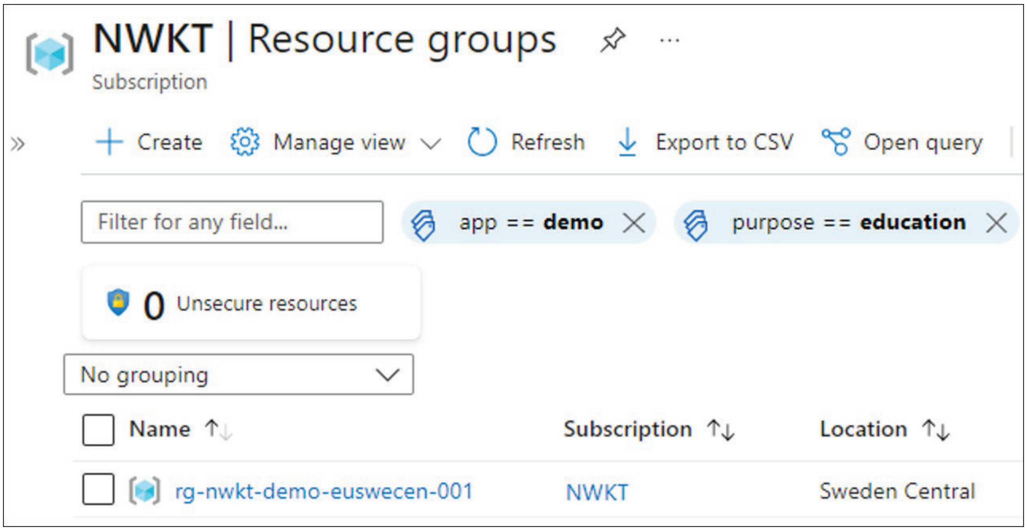
Subscription	NWKT
Resource group	rg-nwkt-demo-euswecen-001
Region	Sweden Central

Tags

app	demo
confidentially	public
SLA	none
department	education
owner	toni
purpose	education

Figure 1-7: *Create Resource Group – Overview.*

Figure 1-8 show our resource group under the NWKT subscription. You can filter out unwanted resource groups by using tags. In our example, I have built two filters, which show resource groups with tags: app==demo and purpose==education.



Name	Subscription	Location
rg-nwkt-demo-euswecen-001	NWKT	Sweden Central

Figure 1-8: *Create Resource Group – Tag Filtering.*

Create VNet with Azure Portal

After creating a resource group, navigate back to the subscription window and select the *Resources* option under the *Settings* section on the left pane. Then click the *Create* button. Click the *Networking* option from the *Categories* pane on the left pane. Then select Create under Virtual Network.

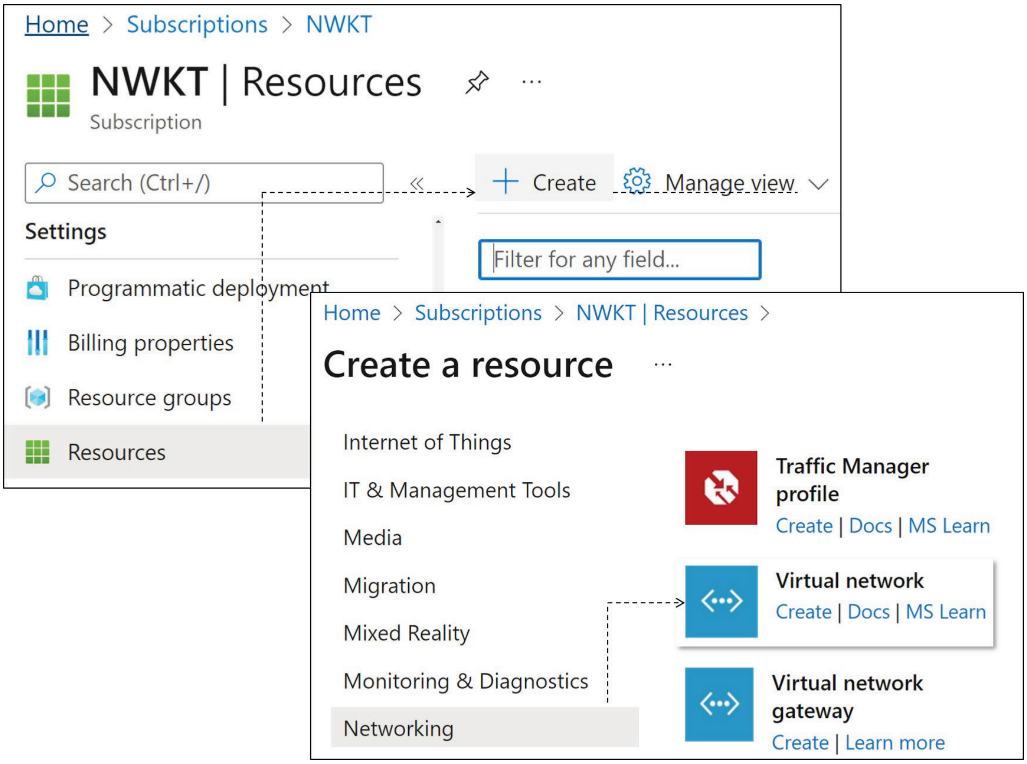
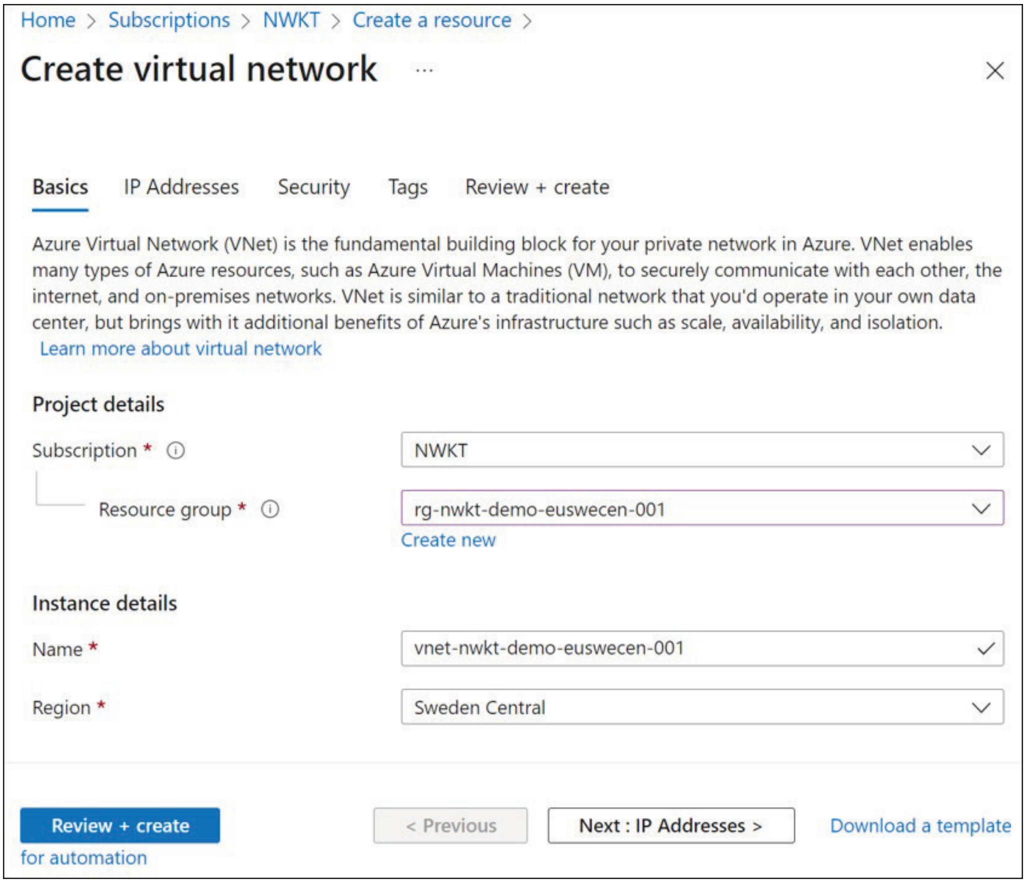


Figure 1-9: *Create Resource – VNet: Step-1.*

Open the *Basics* tab in the *Create virtual network* window. Fill in the Project details information. Select your Subscription from the drop-down menu and give the name to Resource Group. You can also create a new resource group in this window. Give the name to VNet and choose the region where you want to launch the resource group. I have used the same naming structure as what I used with the Resource Group (Resource Type - Application – Environment - Region - Instance). Next, click the *Next: IP Address* button or select it from the tab bar.


Home > Subscriptions > NWKT > Create a resource >

Create virtual network

Basics | IP Addresses | Security | Tags | Review + create

Azure Virtual Network (VNet) is the fundamental building block for your private network in Azure. VNet enables many types of Azure resources, such as Azure Virtual Machines (VM), to securely communicate with each other, the internet, and on-premises networks. VNet is similar to a traditional network that you'd operate in your own data center, but brings with it additional benefits of Azure's infrastructure such as scale, availability, and isolation.
Learn more about virtual network

Project details

Subscription *: NWKT

Resource group *: rg-nwkt-demo-euswecen-001

Create new

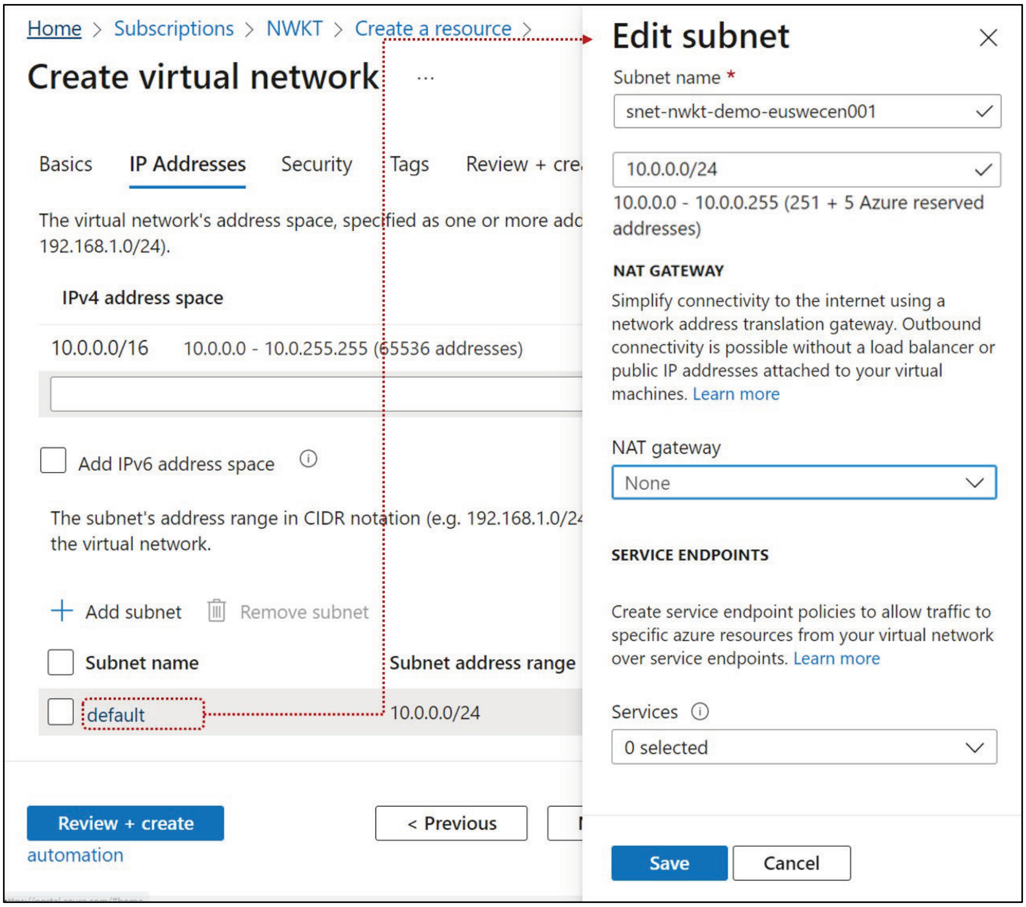
Instance details

Name *: vnet-nwkt-demo-euswecen-001

Region *: Sweden Central

Figure 1-10: *Create Resource – VNet: Basics Information.*

Azure proposes the CIDR 10.0.0.0/16 by default. If you have already assigned it to another VNet, Azure proposes some other CIDR. The default subnet is the first C-Class subnet with a /24 mask. You can edit the name and subnet by clicking the *default* hyperlink that opens the *Edit subnet* window. Once again, I have used the same naming as with Resource Group and VNet.



The virtual network's address space, specified as one or more add
192.168.1.0/24).

IPv4 address space	
10.0.0.0/16	10.0.0.0 - 10.0.255.255 (65536 addresses)

Add IPv6 address space

The subnet's address range in CIDR notation (e.g. 192.168.1.0/24
the virtual network.

Subnet name	Subnet address range
default	10.0.0.0/24

Edit subnet

Subnet name *
snet-nwkt-demo-euswecen001

10.0.0.0/24
10.0.0.0 - 10.0.0.255 (251 + 5 Azure reserved addresses)

NAT GATEWAY

Simplify connectivity to the internet using a network address translation gateway. Outbound connectivity is possible without a load balancer or public IP addresses attached to your virtual machines. Learn more

NAT gateway
None

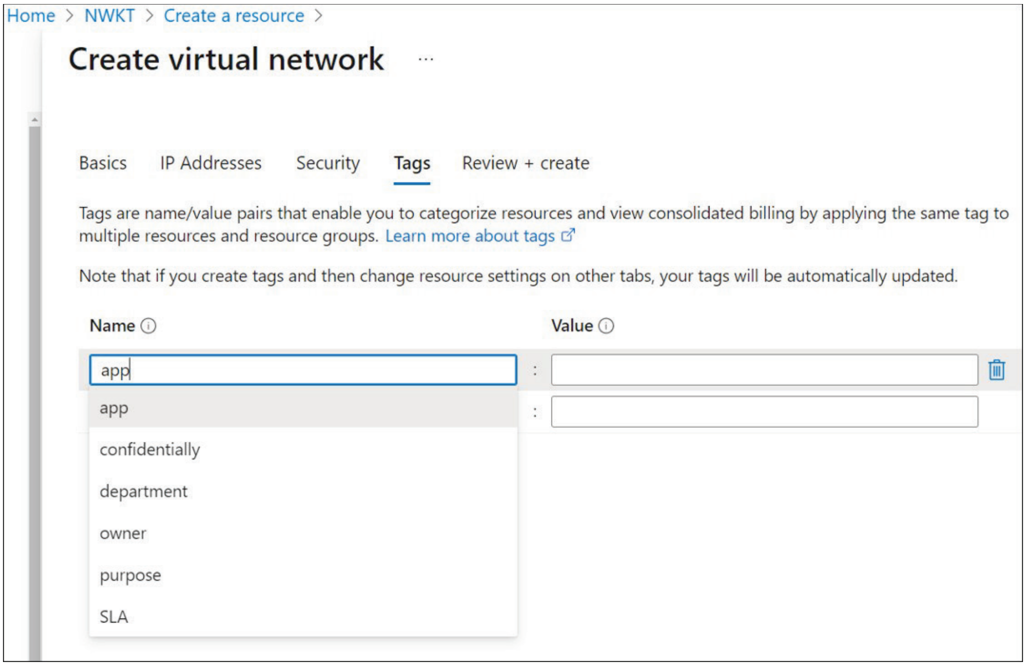
SERVICE ENDPOINTS

Create service endpoint policies to allow traffic to specific azure resources from your virtual network over service endpoints. Learn more

Services
0 selected

Figure 1-11: *Create Resource – VNet: CIDR & Subnet.*

We are not implementing VNet Security features in this example so continue to the *Tags* tab. We can reuse Tags that we created in the Resource Group example. The list of exciting Tags will open when you click the *Name* field. Select tags you want to use with this VNet. When you have selected the tag, click the *Value* field, and you'll see values associated with the tag. In this example, we use the same tags with VNet as with Resource Group. When you have added all necessary Tags, select the *Review + create* tab.


Home > NWKT > Create a resource >

Create virtual network ...

Basics | IP Addresses | Security | Tags | Review + create

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. Learn more about tags

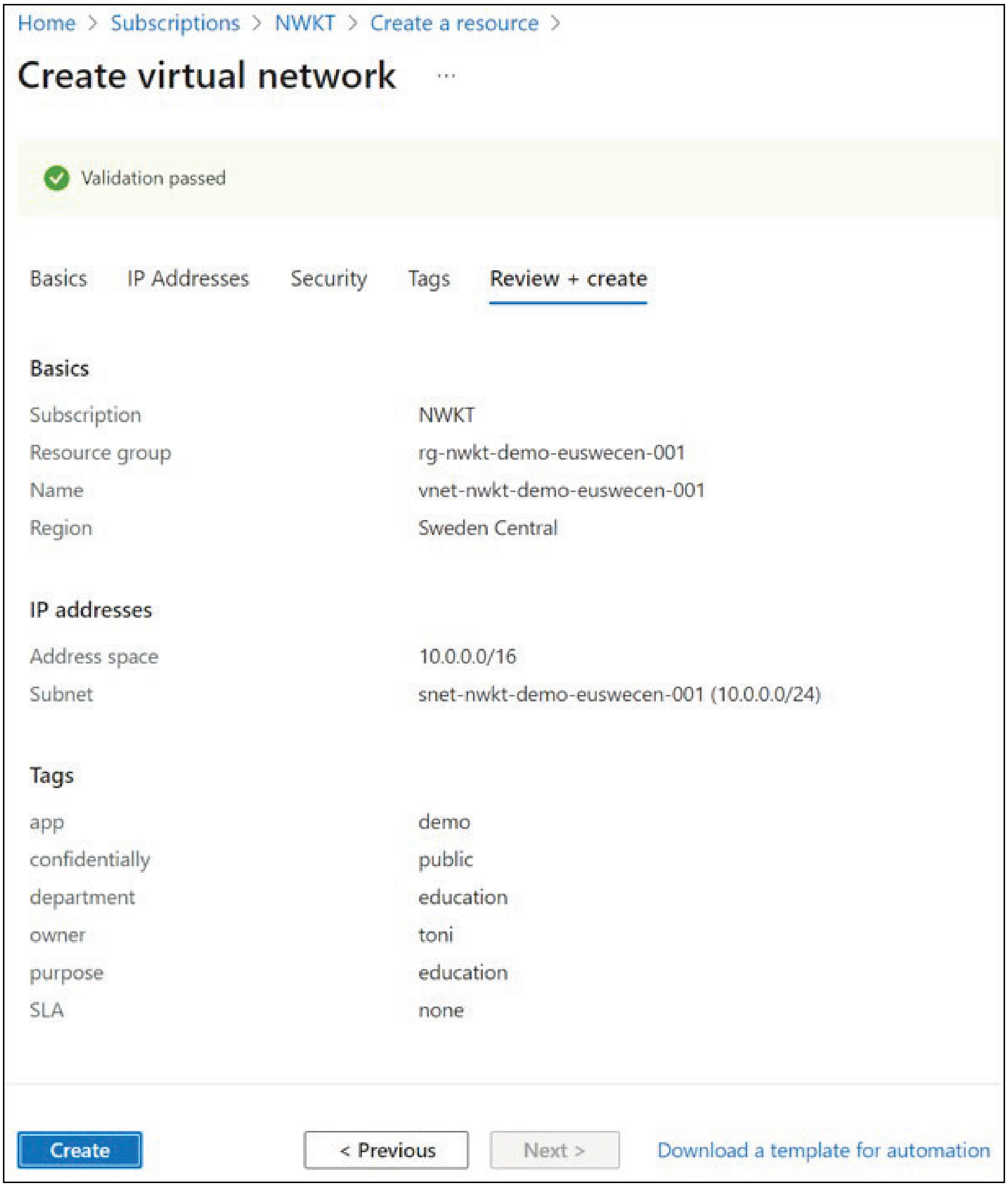
Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

Name		Value
app	:	
	:	

app
confidentially
department
owner
purpose
SLA

Figure 1-12: *Create Resource – VNet: Tags.*

Figure 1-13 shows the Review window. You can download the template and parameters files by clicking the hyperlink *Download a template for automation* as we did in the resource Group example. Click the Create button for deploying net VNet.


Home > Subscriptions > NWKT > Create a resource >
Create virtual network ...
Validation passed
Basics
IP Addresses
Security
Tags
Review + create
Create
< Previous
Next >
Download a template for automation

Basics

Subscription	NWKT
Resource group	rg-nwkt-demo-euswecen-001
Name	vnet-nwkt-demo-euswecen-001
Region	Sweden Central

IP addresses

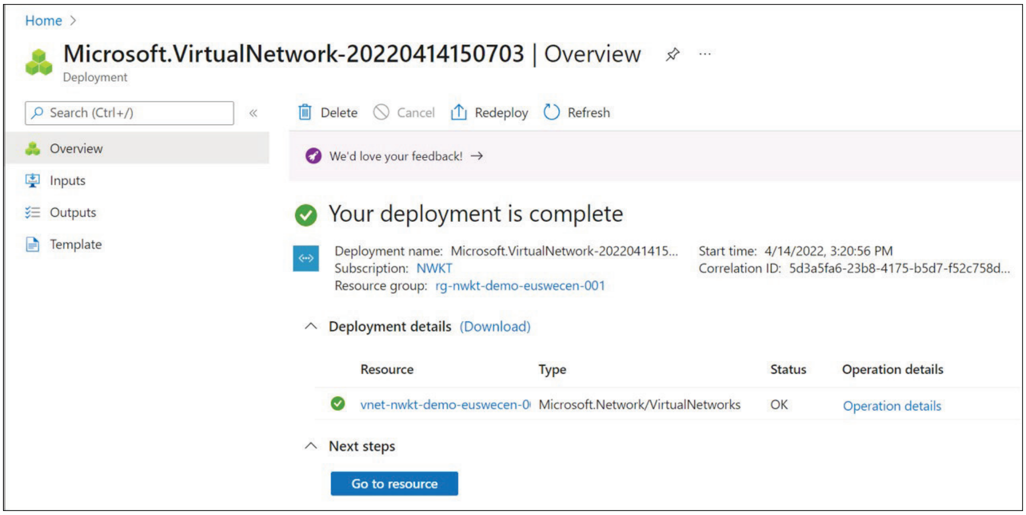
Address space	10.0.0.0/16
Subnet	snet-nwkt-demo-euswecen-001 (10.0.0.0/24)

Tags

app	demo
confidentially	public
department	education
owner	toni
purpose	education
SLA	none

Figure 1-13: *Create Resource – VNet: Review and Create.*

Figure 1-14 shows the status of the deployment (complete). You can click the Go to resource button to proceed to the vnet-nwkt-euswecen-001 page shown in figure 1-15.

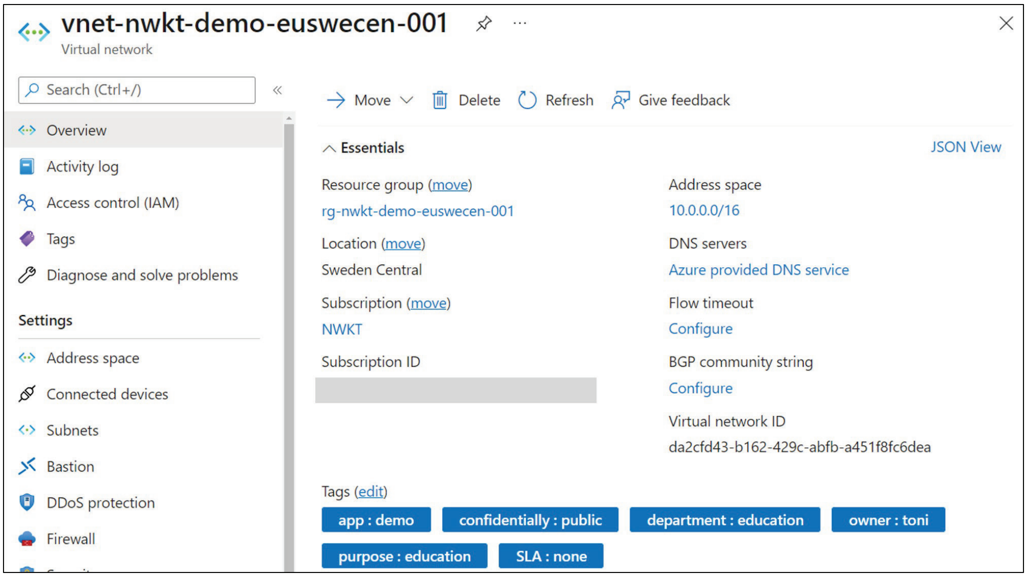


Your deployment is complete

Deployment name: Microsoft.VirtualNetwork-2022041415...
Subscription: NWKT
Resource group: rg-nwkt-demo-euswecen-001
Start time: 4/14/2022, 3:20:56 PM
Correlation ID: 5d3a5fa6-23b8-4175-b5d7-f52c758d...

	Resource	Type	Status	Operation details
✓	vnet-nwkt-demo-euswecen-0	Microsoft.Network/VirtualNetworks	OK	Operation details

Figure 1-14: *Create Resource – VNet: Verification.*



Resource group (move)
rg-nwkt-demo-euswecen-001

Location (move)
Sweden Central

Subscription (move)
NWKT

Subscription ID

Address space
10.0.0.0/16

DNS servers
Azure provided DNS service

Flow timeout
Configure

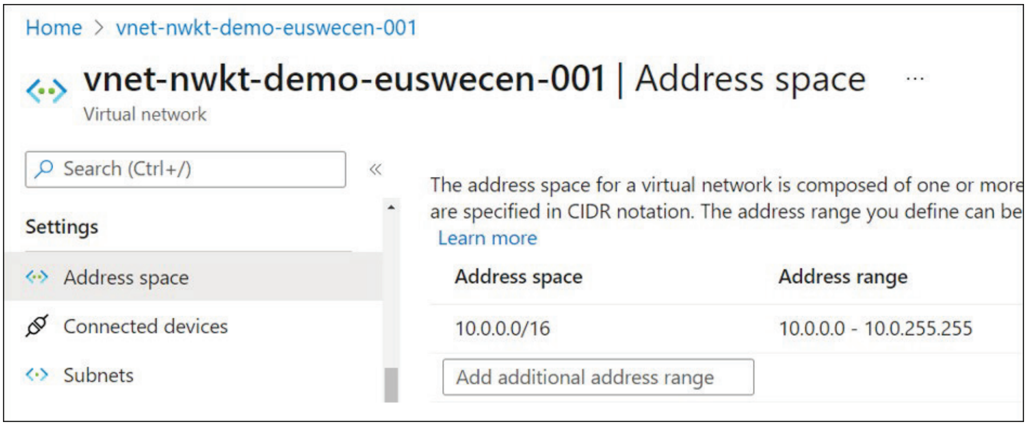
BGP community string
Configure

Virtual network ID
da2cfd43-b162-429c-abfb-a451f8fc6dea

Tags (edit)
app : demo
confidentially : public
department : education
owner : toni
purpose : education
SLA : none

Figure 1-15: *Create Resource – VNet: Overview.*

The *Address space* option under the *Settings* section shows the CIDR we assigned to VNet.



Home > vnet-nwkt-demo-euswecen-001

vnet-nwkt-demo-euswecen-001 | Address space ...

Virtual network

Search (Ctrl+/)

Settings

Address space

Connected devices

Subnets

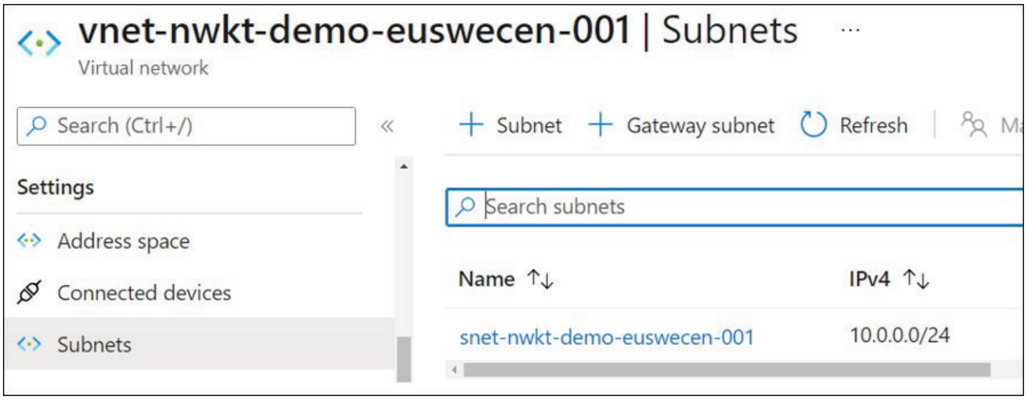
The address space for a virtual network is composed of one or more
are specified in CIDR notation. The address range you define can be
Learn more

Address space	Address range
10.0.0.0/16	10.0.0.0 - 10.0.255.255

Add additional address range

Figure 1-16: *Create Resource – VNet: Address Space.*

The *Subnet* option under the *Settings* section shows the subnet 10.0.0.0/24, which we create during the VNet implementation process.



vnet-nwkt-demo-euswecen-001 | Subnets ...

Virtual network

Search (Ctrl+/)

Settings

Address space

Connected devices

Subnets

+ Subnet + Gateway subnet Refresh | M

Search subnets

Name ↑↓	IPv4 ↑↓
snet-nwkt-demo-euswecen-001	10.0.0.0/24

Figure 1-17: *Create Resource – VNet: Subnet.*

Deploy VNet Azure Resource Manager Templates

Azure Resource Manager (ARM) is a management service platform that enables us to manage Azure resources using Azure Portal, Azure CLI, Azure PowerShell, and SDK. This section focuses on the VNets deployment process using *Visual Studio Code* and its *ARM Templates* extension. ARM Templates describes our resources in JSON format. Figure 1-18 illustrates the workflows and overall building blocks and processes. We intend to create two VNets, one for the production workloads and the other one for development workloads. First, we create an ARM *Deployment Template* (1), which will be a base template for our development and production VNets. In this template, we define the naming policy for VNet and its subnet. We also define allowed tags associated with VNets. However, we don't describe the actual names of these objects. Next, we create a dedicated ARM *Deployment Parameters Template* for production (2a) and development (2b) VNets. These templates define values for objects vnetName, tagName, and snetName. At this phase, we have three ARM templates, a common *Deployment Template* for VNets, and two VNet-specific *Deployment Parameters Templates*. When we have done templates, we create VNet-specific Azure PowerShell scripts (3a-b). These scripts create JSON files, where deployed resources are taken from the VNet base template. The Deployment Template and its values of its, in turn, are taken from the VNet-specific Deployment Parameters template. The script itself defines a resource group and its location. The script deploys these JSON files to ARM. Before converting the deployment file to a REST API call, ARM verifies our user credentials (authentication) and checks if we have rights to create VNets (authorization). If our deployment passes the security checks, ARM makes a REST API call to Resource Provider (Microsoft. Network) and its specific Service (Virtual Network) to finish the deployment (4a-b). The Deployment Template is also validated before implementation.

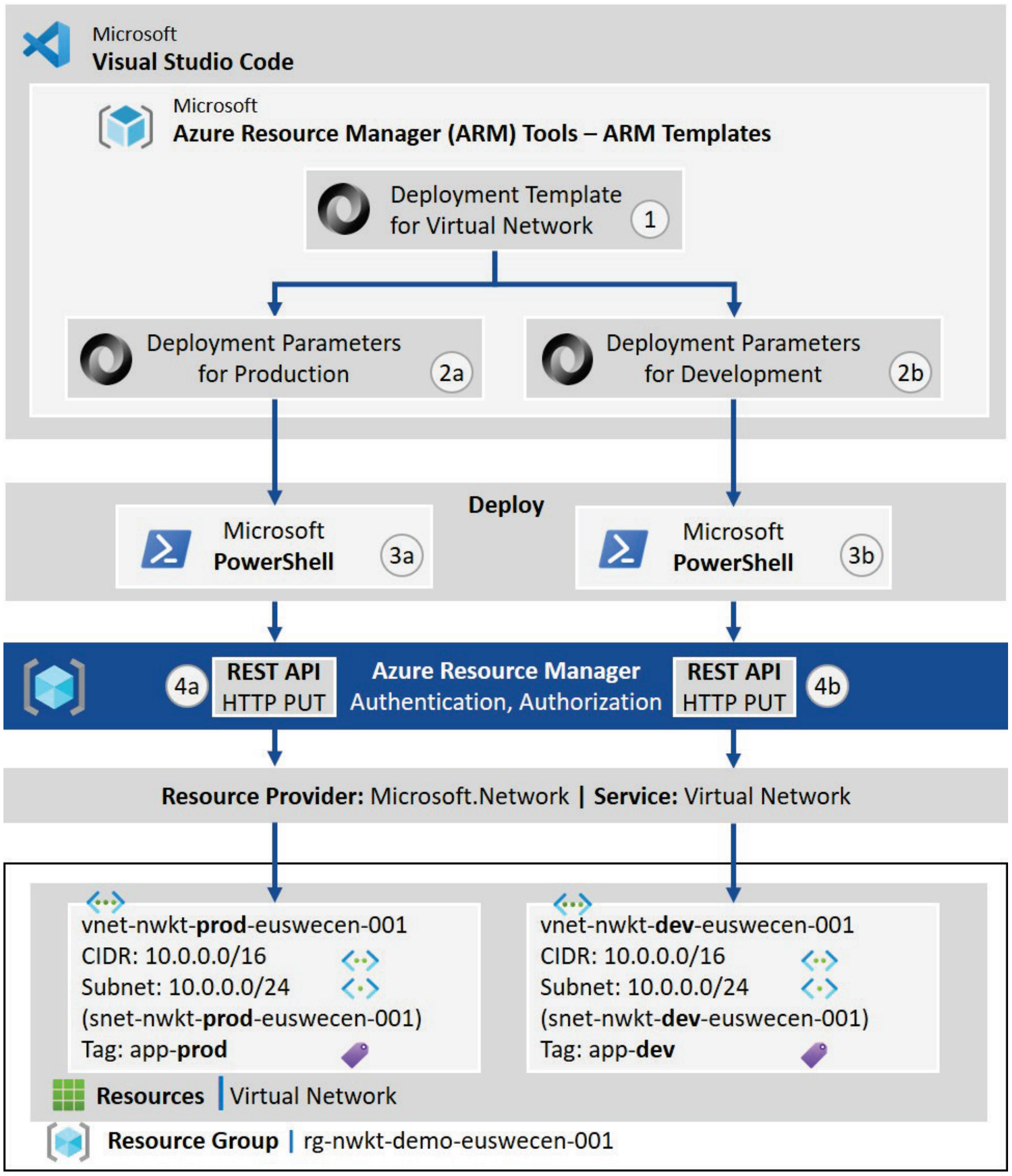
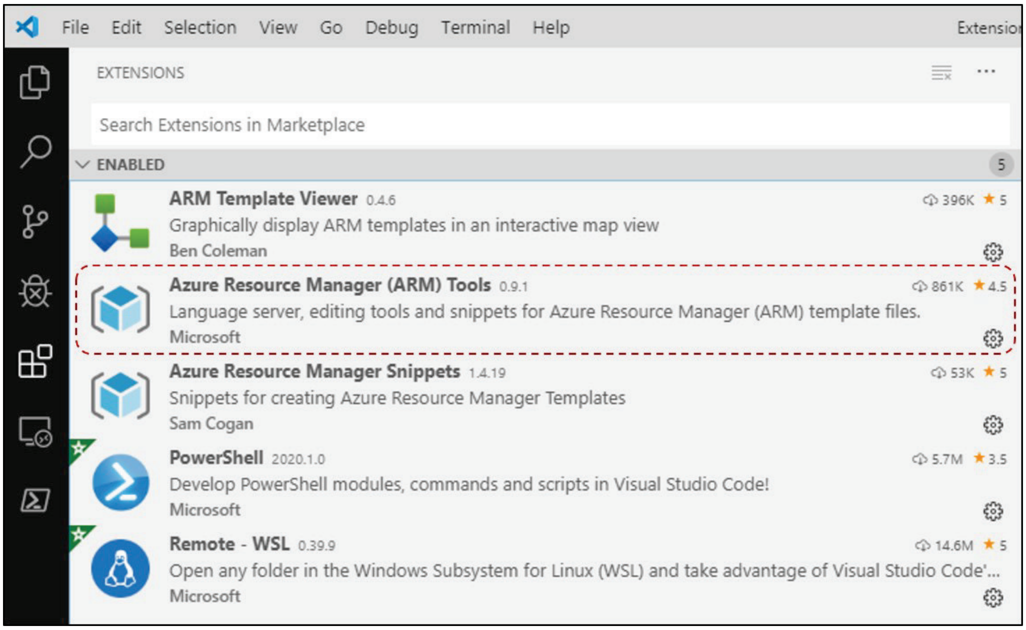

Microsoft
Visual Studio Code
Microsoft
Azure Resource Manager (ARM) Tools – ARM Templates
Deployment Template for Virtual Network 1
Deployment Parameters for Production 2a
Deployment Parameters for Development 2b
Deploy
Microsoft PowerShell 3a
Microsoft PowerShell 3b
4a REST API HTTP PUT
Azure Resource Manager Authentication, Authorization
REST API HTTP PUT 4b
Resource Provider: Microsoft.Network | Service: Virtual Network
vnet-nwkt-prod-euswecen-001
CIDR: 10.0.0.0/16
Subnet: 10.0.0.0/24
(snet-nwkt-prod-euswecen-001)
Tag: app-prod
vnet-nwkt-dev-euswecen-001
CIDR: 10.0.0.0/16
Subnet: 10.0.0.0/24
(snet-nwkt-dev-euswecen-001)
Tag: app-dev
Resources | Virtual Network
Resource Group | rg-nwkt-demo-euswecen-001

Figure 1-18: *ARM Template Deployment Building Blocks and Work Flows.*

Pre-Tasks

The first thing to do is install the *Microsoft Visual Studio Code*. You can download the latest version from code.visualstudio.com. You also need to install the Azure *Resource Manager (ARM) Tools* extension. We are using these tools for creating ARM templates.



ARM Template Viewer 0.4.6 — 396K ★ 5
Graphically display ARM templates in an interactive map view
Ben Coleman

Azure Resource Manager (ARM) Tools 0.9.1 — 861K ★ 4.5
Language server, editing tools and snippets for Azure Resource Manager (ARM) template files.
Microsoft

Azure Resource Manager Snippets 1.4.19 — 53K ★ 5
Snippets for creating Azure Resource Manager Templates
Sam Cogan

PowerShell 2020.1.0 — 5.7M ★ 3.5
Develop PowerShell modules, commands and scripts in Visual Studio Code!
Microsoft

Remote - WSL 0.39.9 — 14.6M ★ 5
Open any folder in the Windows Subsystem for Linux (WSL) and take advantage of Visual Studio Code'...
Microsoft

Figure 1-19: *Microsoft Visual Code Studio and the ARM Tools Extension.*

Besides, you need to install the Azure RM PowerShell or the newer Azure AZ PowerShell module for PowerShell. Examples 1-1 and 1-2 show how I first log in to my Azure account with PowerShell and then verify that the AzureRM module is installed.

```
PS C:\Users\fitonpasa> Login-AzureRmAccount

Account                 SubscriptionName    TenantId      Environment
-------                 ----------------    --------      -----------
toni.pasanen@***.com    NWKT                d944c18e-...  AzureCloud
```

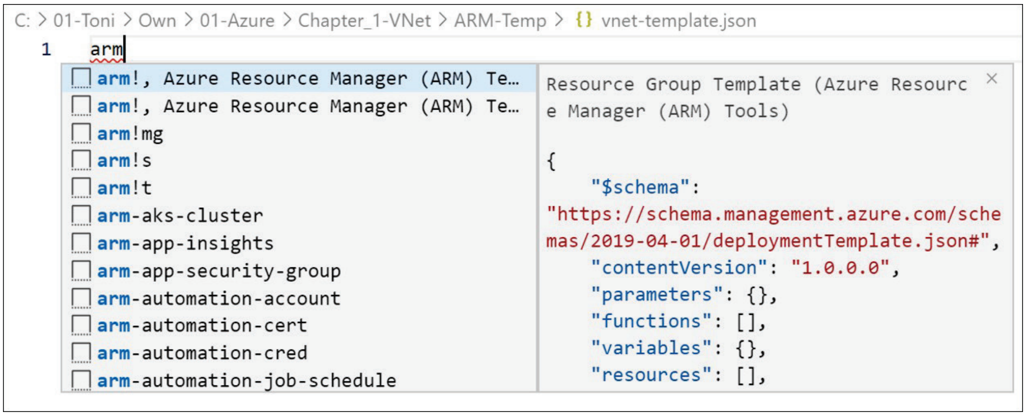
Example 1-1: *Login to Azure Account.*

```
PS C:\Users\fitonpasa> Get-InstalledModule -Name AzureRM

Version     Name         Repository       Description
-------     ----         ----------       -----------
6.13.2      AzureRM      PSGallery        Azure Resource Manager Module
```

Example 1-2: *Verifying Installed Modules.*

Deployment Template for VNet

In Visual Studio Code (VSC), click the *File* tab from the menu bar and select *New File*. The default file type is text (.txt). To use the ARM tools extension, you have to save the file in JSON format. I have named the example file vnet-template.json. When you type the *arm* into the first row, you'll get the list of arm-starting options. Select the *arm!,* Azure Resource Manager (ARM) template from the list.

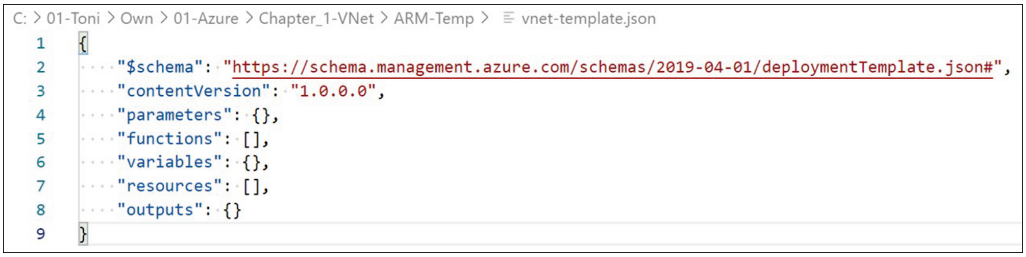

C: > 01-Toni > Own > 01-Azure > Chapter_1-VNet > ARM-Temp > {} vnet-template.json

```
arm
arm!, Azure Resource Manager (ARM) Te…
arm!, Azure Resource Manager (ARM) Te…
arm!mg
arm!s
arm!t
arm-aks-cluster
arm-app-insights
arm-app-security-group
arm-automation-account
arm-automation-cert
arm-automation-cred
arm-automation-job-schedule
```

```
Resource Group Template (Azure Resource Manager (ARM) Tools)

{
    "$schema":
"https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {},
    "functions": [],
    "variables": {},
    "resources": [],
```

Figure 1-20: *Create the VNet Deployment Template – Step#1.*

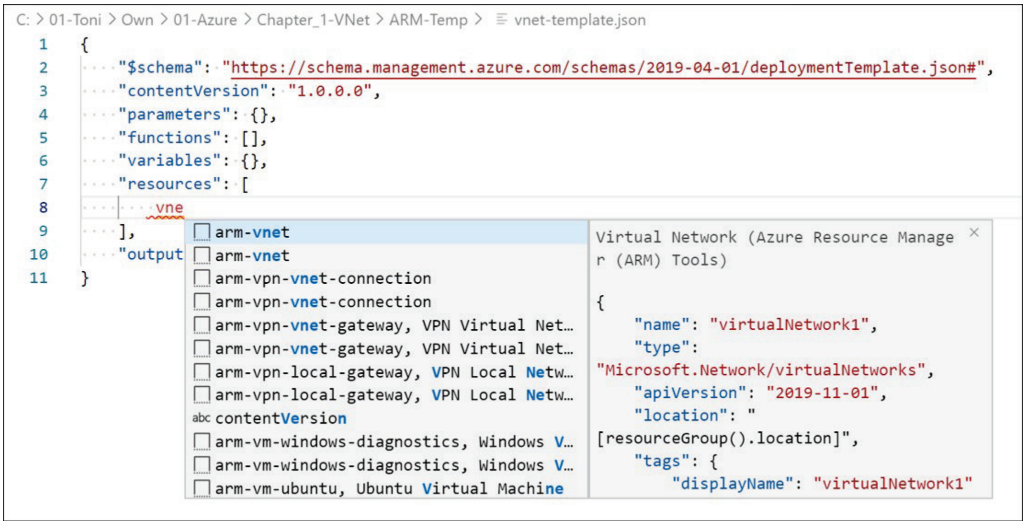
The *arm!,* option opens the deployment template that lists basic elements of the JSON schema file. There are three required elements for every JSON file: *$schema, contentVersion,* and *resources*. The *$schema* element describes the location and version of the top-level schema file (schmema.management.azure.com, version: 2019-04-01). The *contentVersion* element tells the version. Though it is a mandatory element, we don't have to use it for versioning. However, it is good practice to update the version numbering when modifying an existing file. The *Resources* array is used for creating resource objects and their attributes. Our focus is on Resource arrays and Parameter objects (not mandatory).



```
C: > 01-Toni > Own > 01-Azure > Chapter_1-VNet > ARM-Temp > vnet-template.json
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {},
    "functions": [],
    "variables": {},
    "resources": [],
    "outputs": {}
}
```

Figure 1-21: *Create the VNet Deployment Template – Step#1 Continues.*

Next, we modify the Resources array. Square brackets [ ] after the Resource states that the following JSON structure is an array. An array contains zero or more JSON values (object, array, number, string, true or false, or null). We are going to create an object for VNet. Point the cursor between square brackets and then press Enter. Type the vne, and you'll get the list of all options which name includes vne. Select the arm-vnet. It is a template for virtual networks.



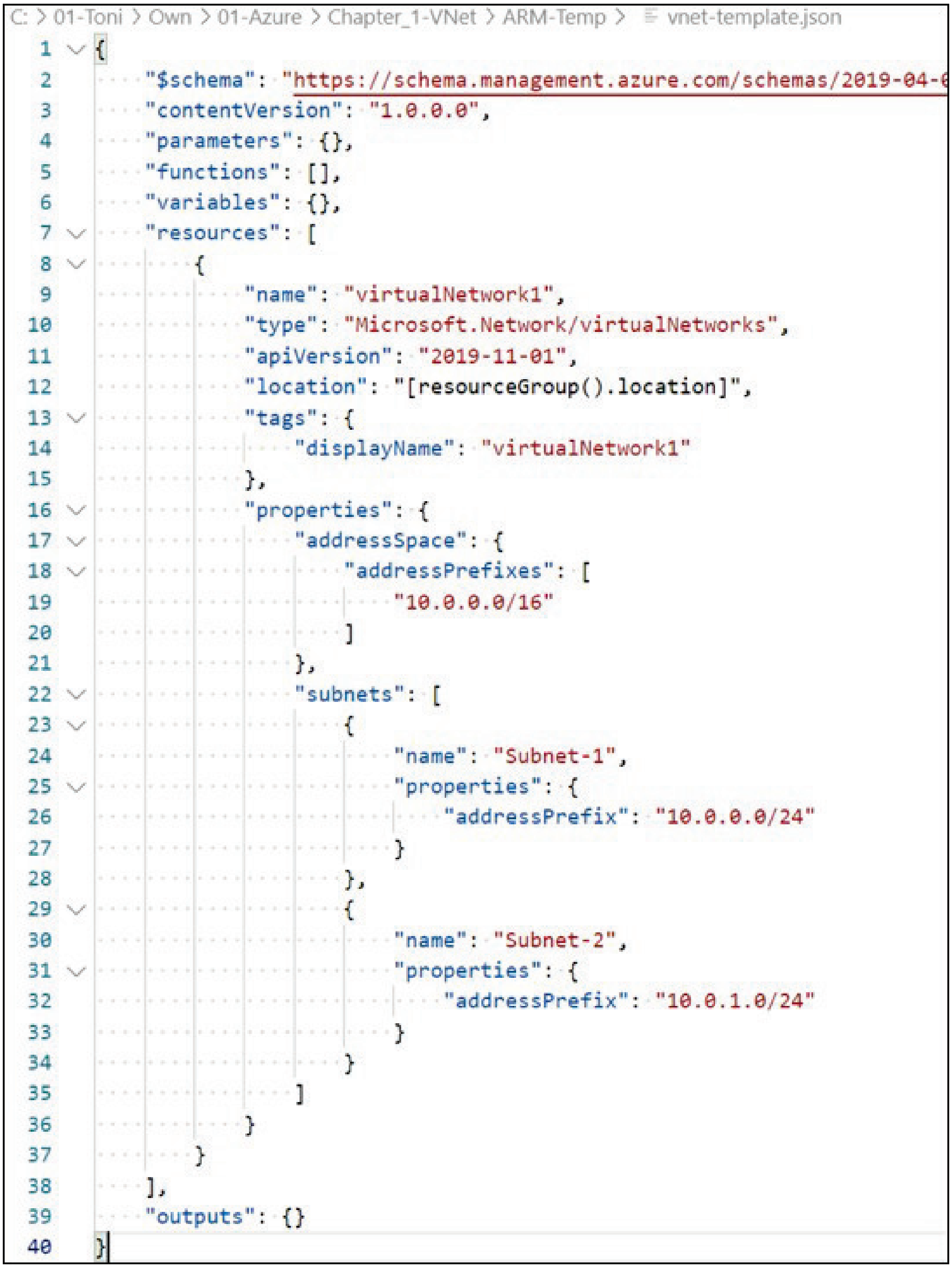
```
C: > 01-Toni > Own > 01-Azure > Chapter_1-VNet > ARM-Temp > vnet-template.json
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {},
    "functions": [],
    "variables": {},
    "resources": [
        vne
    ],
    "output
}
```

```
arm-vnet
arm-vnet
arm-vpn-vnet-connection
arm-vpn-vnet-connection
arm-vpn-vnet-gateway, VPN Virtual Net…
arm-vpn-vnet-gateway, VPN Virtual Net…
arm-vpn-local-gateway, VPN Local Netw…
arm-vpn-local-gateway, VPN Local Netw…
contentVersion
arm-vm-windows-diagnostics, Windows V…
arm-vm-windows-diagnostics, Windows V…
arm-vm-ubuntu, Ubuntu Virtual Machine
```

```
Virtual Network (Azure Resource Manage
r (ARM) Tools)

{
    "name": "virtualNetwork1",
    "type":
"Microsoft.Network/virtualNetworks",
    "apiVersion": "2019-11-01",
    "location": "
[resourceGroup().location]",
    "tags": {
        "displayName": "virtualNetwork1"
```

Figure 1-22: *Create the VNet Deployment Template – Step#2.*

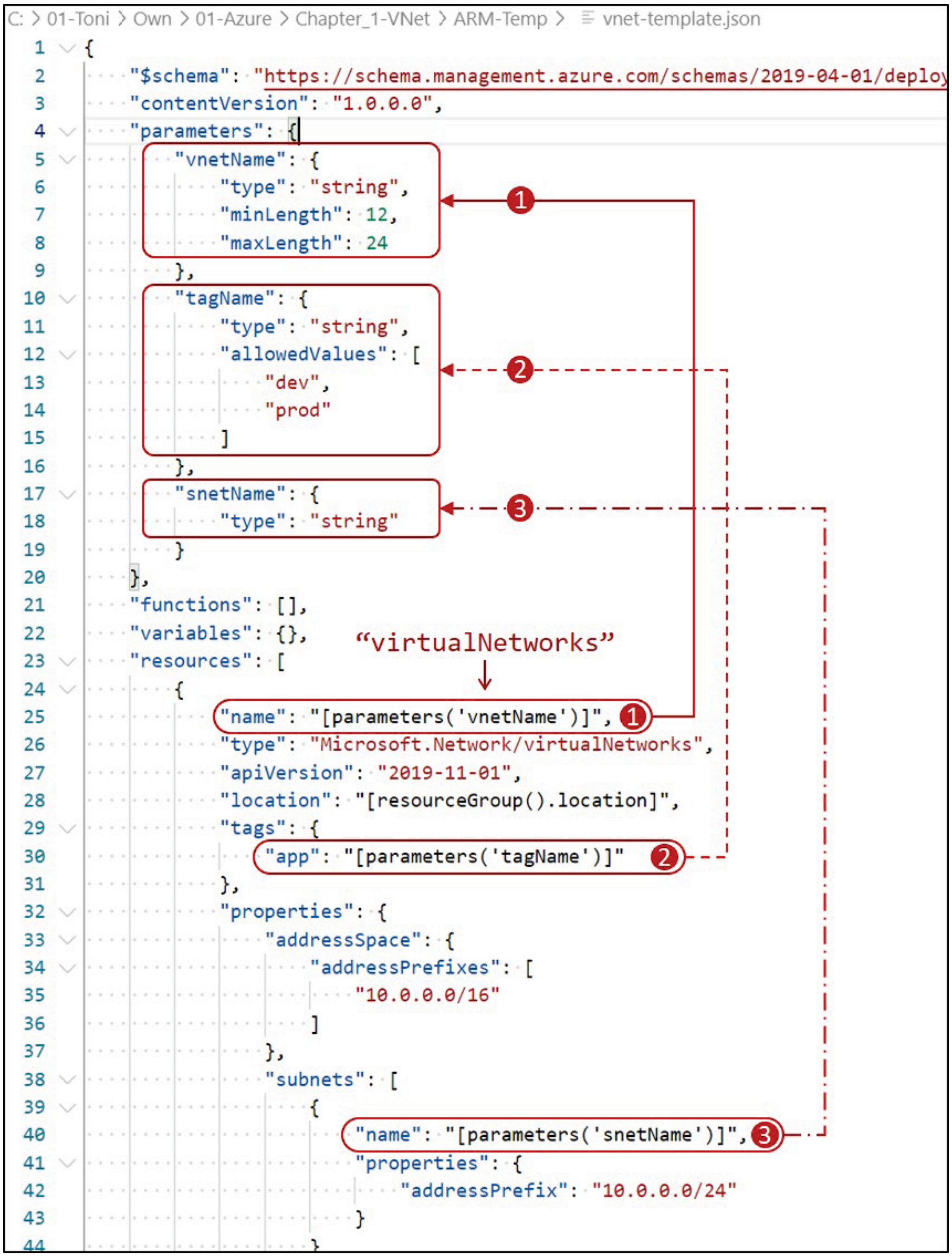
JSON object is represented within curly brackets { }. The object includes zero or more name/value pairs. Figure 1-23 shows the *vnet-template.json* template file where all arrays (values) and objects (name/value) are with their defaults. The name/value pair in row 10 describes the Service Provider and the Service. The apiVersion in this template is 2019-11-01. You can find the template format of each API version for Virtual Network resource by navigating to https://docs.microsoft.com/en-us/azure/templates. Then type Virtual Network in the Filter by title field. Select the Virtual Network > Reference > Network > (API versions) option from the list. The name location is the same as the location of the Resource Group where we deploy our VNet. The Properties object defines CIDR Range (as an array) and subnets (as an object).


C: > 01-Toni > Own > 01-Azure > Chapter_1-VNet > ARM-Temp > vnet-template.json

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-0
    "contentVersion": "1.0.0.0",
    "parameters": {},
    "functions": [],
    "variables": {},
    "resources": [
        {
            "name": "virtualNetwork1",
            "type": "Microsoft.Network/virtualNetworks",
            "apiVersion": "2019-11-01",
            "location": "[resourceGroup().location]",
            "tags": {
                "displayName": "virtualNetwork1"
            },
            "properties": {
                "addressSpace": {
                    "addressPrefixes": [
                        "10.0.0.0/16"
                    ]
                },
                "subnets": [
                    {
                        "name": "Subnet-1",
                        "properties": {
                            "addressPrefix": "10.0.0.0/24"
                        }
                    },
                    {
                        "name": "Subnet-2",
                        "properties": {
                            "addressPrefix": "10.0.1.0/24"
                        }
                    }
                ]
            }
        }
    ],
    "outputs": {}
}
```

Figure 1-23: *The Default ARM template for Virtual Networks.*

We intend to use *vnet-template* for deploying both Development and Production VNets. We use the same CIDR range 10.0.0.0/16 and subnet 10.0.0.0/24 with both VNets. However, we use the naming structure, which describes the resource *type - application – environment - region – instance*. We are also using the tag naming model, which describes the environment. Now, instead of using default values, we are using *functions*. First, replace the *"virtualNetworks"* value with square brackets between the quotation marks *"[ ]"*. Click between square brackets and type *para.* Select the *parameters* function from the list. By doing this, line 25 is changed to ***"name": "[parameters()]",*** **and** add the *'vnetName'* within brackets. At this phase, line 25 is **"name": "[parameters('vnetName')]"**. This function calls the vnetName object (line 5) under the parameters object (line 4). The vnetName object under the parameters element defines the naming policy, not the name itself. The type of the name is a string, the min length is 12 characters, and the max length is 24 characters. If we violate this policy, the validation process fails. The function *parameters* under object *tags,* call *a parameters* element on line 4 and its object *tagName* on line 10. The *allowedValues* array defines values (dev or prod) that we can use. Just for recap, allowedValues is an array that includes strings without names. If we try to use any other tags than what we have defined in allowedValues, the validation process notices that, and the deployment fails.

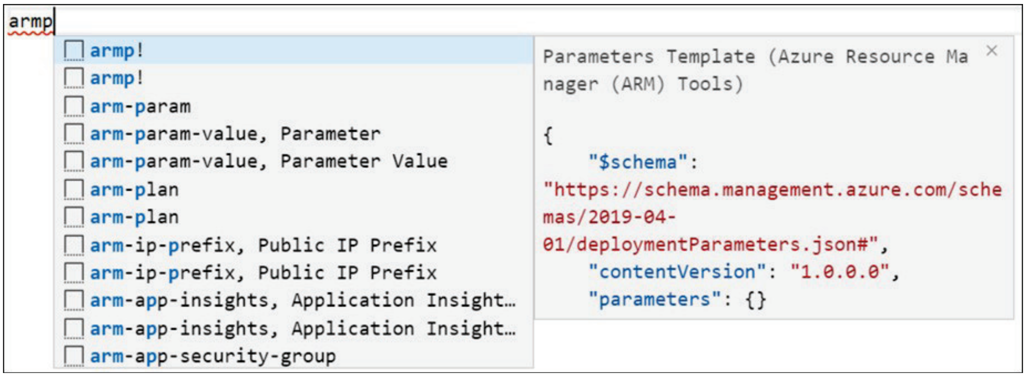


```
C: > 01-Toni > Own > 01-Azure > Chapter_1-VNet > ARM-Temp > vnet-template.json
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploy
    "contentVersion": "1.0.0.0",
    "parameters": {
        "vnetName": {
            "type": "string",
            "minLength": 12,
            "maxLength": 24
        },
        "tagName": {
            "type": "string",
            "allowedValues": [
                "dev",
                "prod"
            ]
        },
        "snetName": {
            "type": "string"
        }
    },
    "functions": [],
    "variables": {},
    "resources": [
        {
            "name": "[parameters('vnetName')]",
            "type": "Microsoft.Network/virtualNetworks",
            "apiVersion": "2019-11-01",
            "location": "[resourceGroup().location]",
            "tags": {
                "app": "[parameters('tagName')]"
            },
            "properties": {
                "addressSpace": {
                    "addressPrefixes": [
                        "10.0.0.0/16"
                    ]
                },
                "subnets": [
                    {
                        "name": "[parameters('snetName')]",
                        "properties": {
                            "addressPrefix": "10.0.0.0/24"
                        }
                    }
```

Figure 1-24: *The Modified ARM template for Virtual Networks*

Deployment Parameters for VNet

After building a deployment template JSON file (vnet-template.json), we create separate deployment parameters JSON files for the development and production VNets. Open a new JSON file in Visual Studio Code. Type *armp* on the first line and select *armp!* From the list.

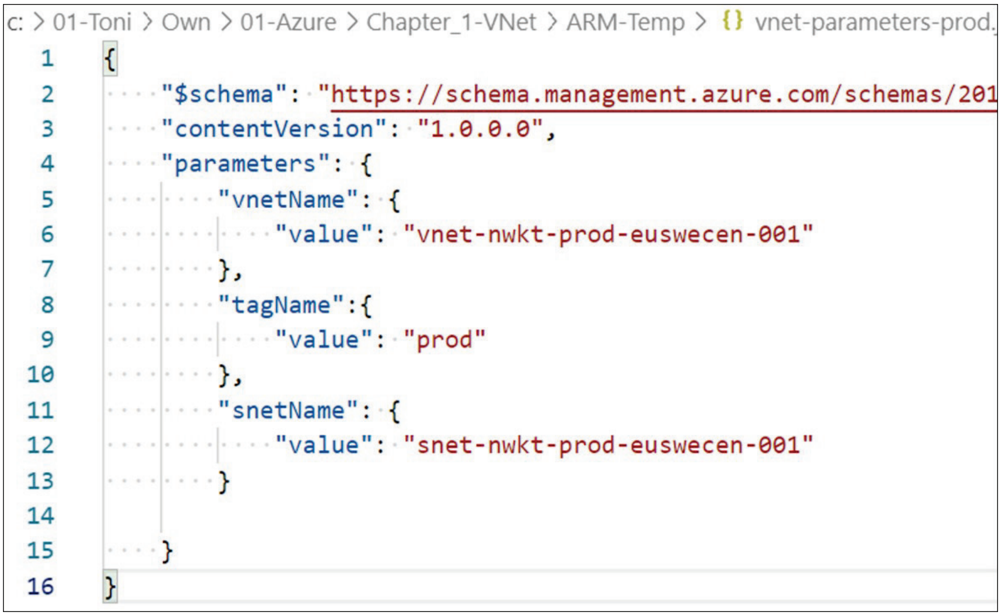


```
armp
armp!
armp!
arm-param
arm-param-value, Parameter
arm-param-value, Parameter Value
arm-plan
arm-plan
arm-ip-prefix, Public IP Prefix
arm-ip-prefix, Public IP Prefix
arm-app-insights, Application Insight…
arm-app-insights, Application Insight…
arm-app-security-group
```

```
Parameters Template (Azure Resource Ma
nager (ARM) Tools)

{
    "$schema":
"https://schema.management.azure.com/sche
mas/2019-04-
01/deploymentParameters.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {}
```

Figure 1-25: *Create DeploymentParameters JSON File.*

Figure 1-26 illustrates the JSON file vnet-parameters-prod.json. It includes three mandatory elements *$schema*, *contentVersion*, and *parameters*. The value of the *vnetName* object describes the vnet name *vnet-nwkt-prod-euswecen-001*. The value of the *tagName* object attaches the tag *prod* to VNet. The value of the *snetName* describes the subnet name *snet-nwkt-prod-euswecen-001*.

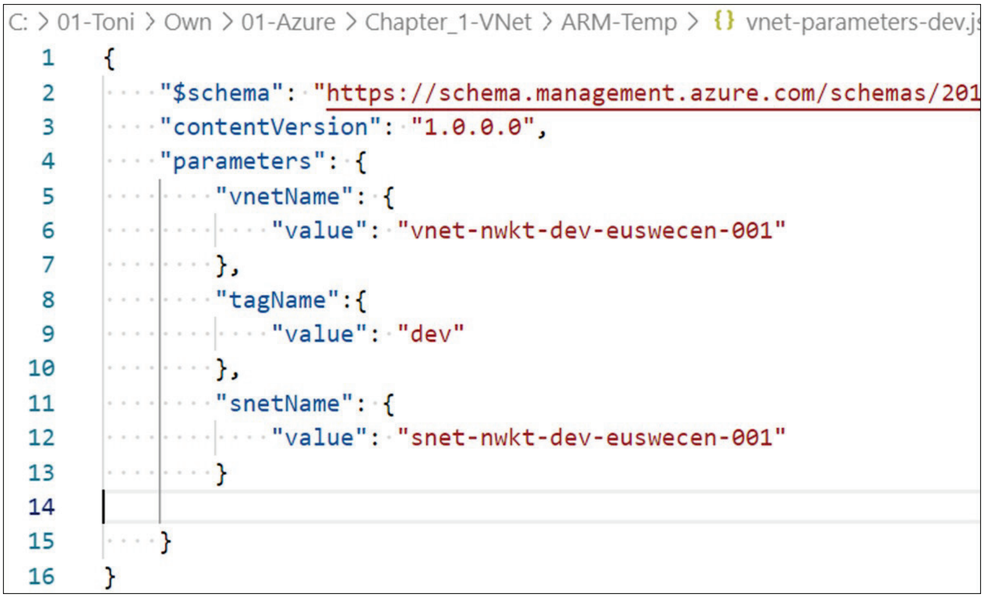


c: > 01-Toni > Own > 01-Azure > Chapter_1-VNet > ARM-Temp > {} vnet-parameters-prod.

```
{
    "$schema": "https://schema.management.azure.com/schemas/201
    "contentVersion": "1.0.0.0",
    "parameters": {
        "vnetName": {
            "value": "vnet-nwkt-prod-euswecen-001"
        },
        "tagName":{
            "value": "prod"
        },
        "snetName": {
            "value": "snet-nwkt-prod-euswecen-001"
        }

    }
}
```

Figure 1-26: *Create DeploymentParameters JSON File for Prod.*

Figure 1-27 shows the JSON file vnet-parameters-dev.json.

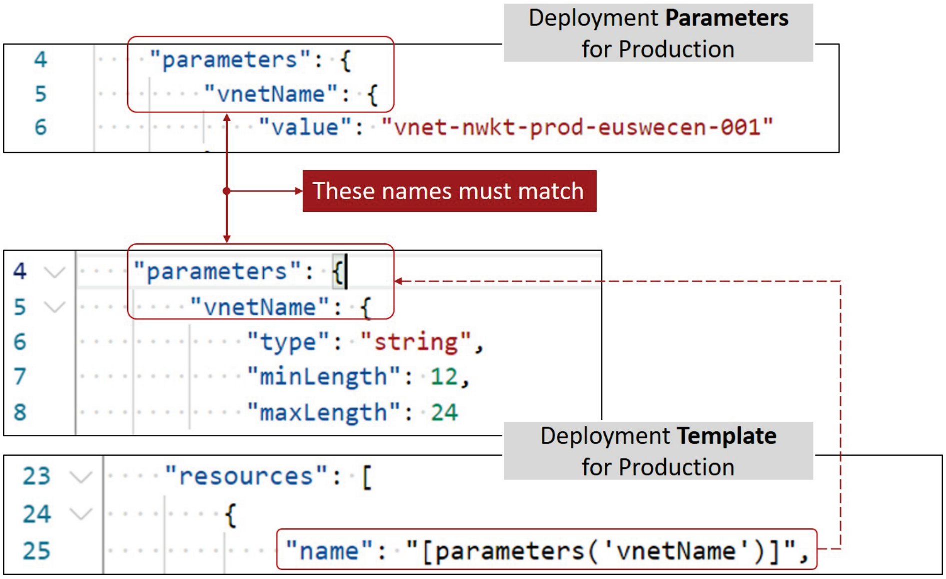


C: > 01-Toni > Own > 01-Azure > Chapter_1-VNet > ARM-Temp > {} vnet-parameters-dev.js

```
{
    "$schema": "https://schema.management.azure.com/schemas/201
    "contentVersion": "1.0.0.0",
    "parameters": {
        "vnetName": {
            "value": "vnet-nwkt-dev-euswecen-001"
        },
        "tagName":{
            "value": "dev"
        },
        "snetName": {
            "value": "snet-nwkt-dev-euswecen-001"
        }

    }
}
```

Figure 1-27: *Create DeploymentParameters JSON File for Dev.*

When using functions for naming, we have to use equal names for called objects in both Parameter and Deployment JSON files. Figure 1-28 shows that the name value calls the *parameters* schema and its *vnetName* object in Deployment Template. These objects have to use the same name.



Deployment **Parameters** for Production
These names must match
Deployment **Template** for Production

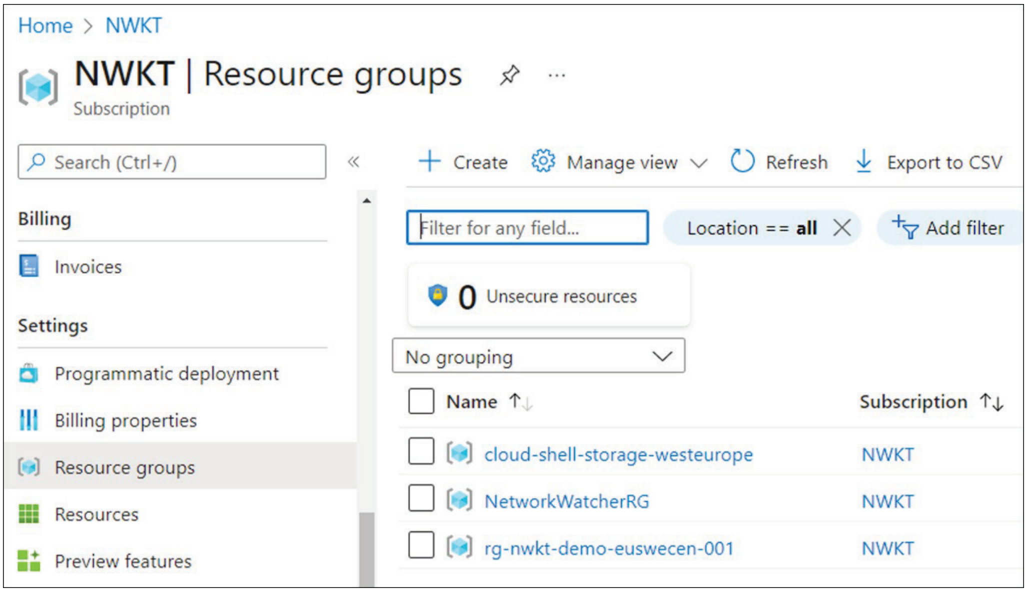
```
    "parameters": {
        "vnetName": {
            "value": "vnet-nwkt-prod-euswecen-001"
```

```
    "parameters": {
        "vnetName": {
            "type": "string",
            "minLength": 12,
            "maxLength": 24
```

```
    "resources": [
        {
            "name": "[parameters('vnetName')]",
```

Figure 1-28: *Create DeploymentParameters JSON File for Prod.*

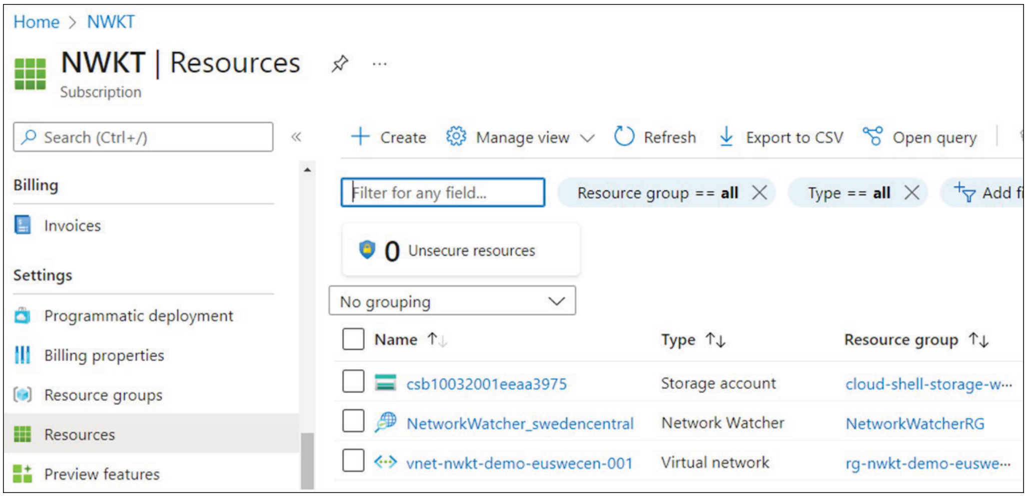
Figure 1-29 lists the Resource Groups we have before we implement new virtual networks.



Name	Subscription
cloud-shell-storage-westeurope	NWKT
NetworkWatcherRG	NWKT
rg-nwkt-demo-euswecen-001	NWKT

Figure 1-29: *Existing Resource Groups.*

Figure 1-30 lists the Resource Groups we have before implementing new virtual networks.



Name	Type	Resource group
csb10032001eeaa3975	Storage account	cloud-shell-storage-w...
NetworkWatcher_swedencentral	Network Watcher	NetworkWatcherRG
vnet-nwkt-demo-euswecen-001	Virtual network	rg-nwkt-demo-euswe...

Figure 1-30: *Existing VNets.*

Deploying Process

Example 1-3 shows the PowerShell script for creating the VNet *vnet-nwkt-dev-euswecen-001*. The Azure RM PowerShell command *New-AzureRmResource Group* deploys the Resource Group on Azure Region swedencentral using the name defined on the first row. The *Force* command at the end of the line executes the command without asking for user confirmation. The second command, *New-AzureRmResourceGroupDeployment,* deploys a resource described in the file *vnet-template.json* using parameters from the file *vnet-parameters-dev.json.* The new resource will be launched in RG rg-arm-demo-euswecen-011. Note that the -Name rg-arm-demo-dev-001 is used as a Deployment name.

```
$rg = 'rg-arm-demo-euswecen-011'
New-AzureRmResourceGroup -Name $rg -Location swedencentral -Force
New-AzureRmResourceGroupDeployment `
    -Name 'rg-arm-demo-dev-011' `
    -ResourceGroupName $rg `
    -TemplateFile 'C:\01-Toni\...\ARM-Temp\vnet-template.json' `
    -TemplateParameterFile 'C:\01-Toni...\ARM-Temp\vnet-parameters-dev.json'
```

Example 1-3: *PowerShell Script for VNet Development Deployment (vnet-deploy-dev.ps1).*

Example 1-4 shows the PowerShell script that creates *vnet-nwkt-prod-euswecen-001* into the same region as *vnet-nwkt-dev-euswecen-001*.

```
$rg = 'rg-arm-demo-euswecen-011'
New-AzureRmResourceGroup -Name $rg -Location swedencentral -Force
New-AzureRmResourceGroupDeployment `
    -Name 'rg-arm-demo-prod-011' `
    -ResourceGroupName $rg `
    -TemplateFile 'C:\01-Toni\...\ARM-Temp\vnet-template.json' `
    -TemplateParameterFile 'C:\01-Toni...\ARM-Temp\vnet-parameters-prod.json'
```

Example 1-4: *PowerShell Script for VNet Production Deployment (vnet-deploy-prod.ps1).*

Example 1-5 illustrates the deployment process. First, we run the *vnet-deploy-dev.ps1* command. Note that you have to use the .\ prefix if the executed PowerShell file is in the same directory where you run the command. After a short delay (time depends on the deployment size), Azure gives feedback. The first section shows that we have successfully provisioned a new Resource Group to the swedencentral region. The second section verifies that we have successfully deployed a new VNet. It also describes the parameters associated with resources.

```
PS C:\01-Toni\Own\01-Azure\Chapter_1-VNet\ARM-Temp> .\vnet-deploy-dev.ps1

ResourceGroupName : rg-arm-demo-euswecen-011
Location          : swedencentral
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/***/rg-arm-demo-euswecen-011

ResourceGroupName       : rg-arm-demo-euswecen-011
OnErrorDeployment       :
DeploymentName          : rg-arm-demo-011
CorrelationId           : 40731f62-6d40-45e0-822f-5f4c7f9587aa
ProvisioningState       : Succeeded
Timestamp               : 3.5.2022 8.36.51
Mode                    : Incremental
TemplateLink            :
TemplateLinkString      :
DeploymentDebugLogLevel :
Parameters              : {[vnetName,
Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.DeploymentVariable],
[tagName,
Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.DeploymentVariable],
[snetName,
Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.DeploymentVariable]}
ParametersString        :
                          Name             Type             Value
                          ===============  ===============  ==========
                          vnetName         String           vnet-nwkt-dev-euswecen-001
                          tagName          String           dev
                          snetName         String           snet-nwkt-dev-euswecen-001

Outputs                 : {}
OutputsString           :
```

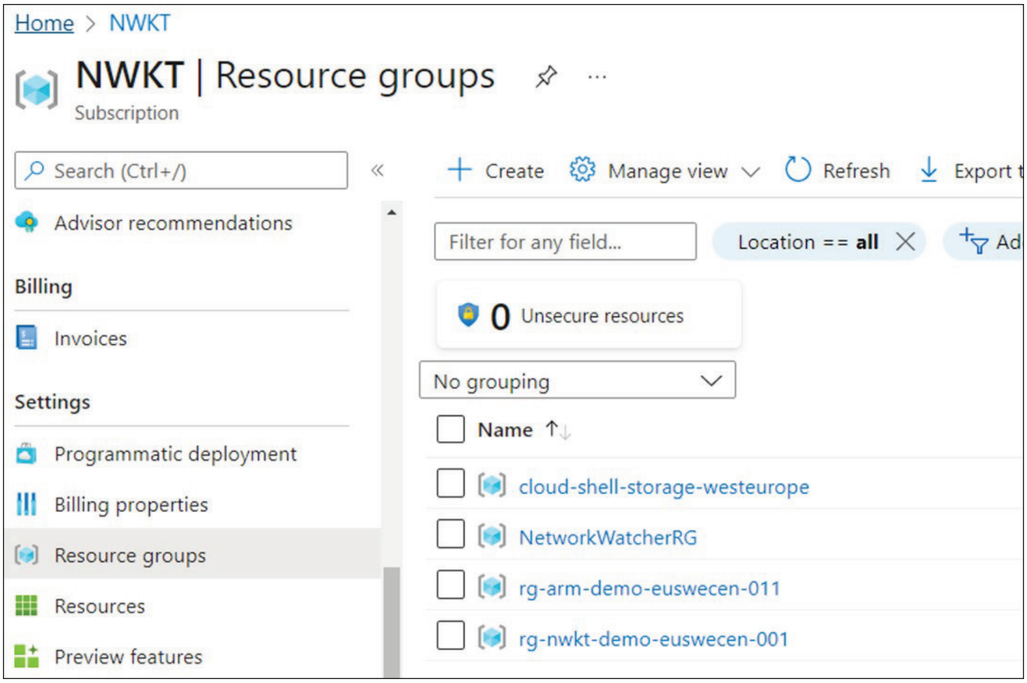
Example 1-5: *Running PowerShell "script vnet-deploy-dev.ps1".*

Example 1-6 shows the deployment state and result when running the PowerShell script vnet-deploy-**prod**.ps1 script.

```
PS C:\01-Toni\Own\01-Azure\Chapter_1-VNet\ARM-Temp> .\vnet-deploy-prod.ps1

ResourceGroupName : rg-arm-demo-euswecen-011
Location          : swedencentral
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/***/resourceGroups/rg-arm-demo-euswecen-011


ResourceGroupName       : rg-arm-demo-euswecen-011
OnErrorDeployment       :
DeploymentName          : rg-arm-demo-011
CorrelationId           : e70bab8b-1a7e-4704-b5f0-c76a7929e646
ProvisioningState       : Succeeded
Timestamp               : 3.5.2022 8.38.34
Mode                    : Incremental
TemplateLink            :
TemplateLinkString      :
DeploymentDebugLogLevel :
Parameters              : {[vnetName,
Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.DeploymentVariable],
[tagName, Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.Dep
                          loymentVariable], [snetName,
Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.DeploymentVariable]}
ParametersString        :
                          Name             Type             Value
                          ===============  ==============   ==========
                          vnetName         String           vnet-nwkt-prod-euswecen-001
                          tagName          String           prod
                          snetName         String           snet-nwkt-prod-euswecen-001

Outputs                 : {}
OutputsString           :
```

Example 1-6: *Running PowerShell "script vnet-deploy-prod.ps1".*

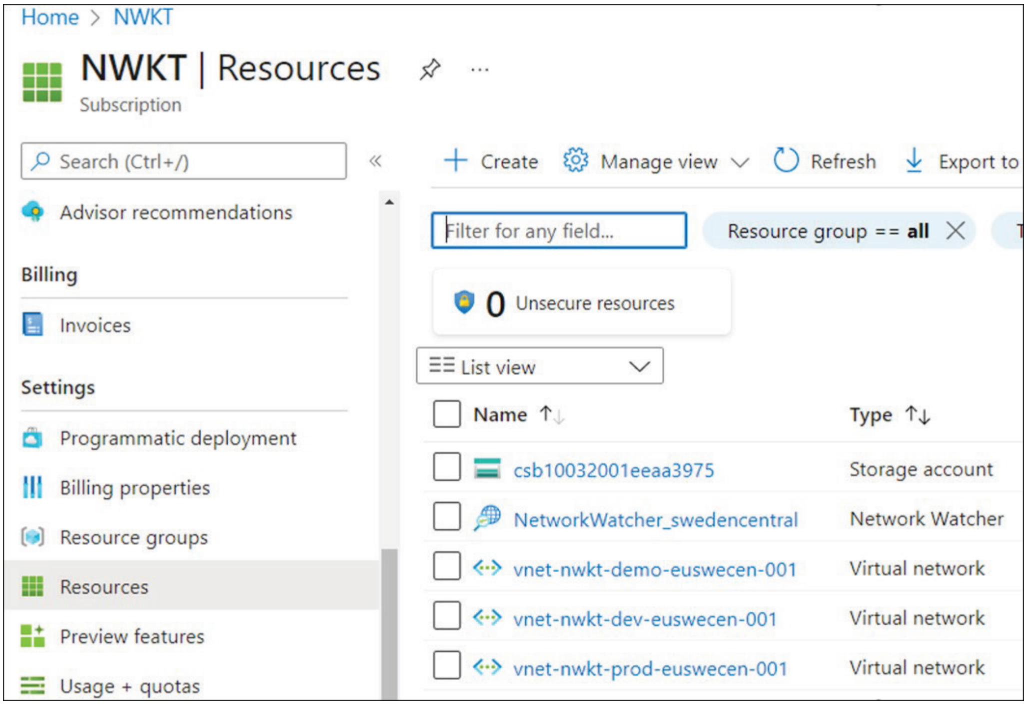
We can also verify from the Azure portal that we now have a new Recourse Group *rg-arm-demo-euswecen-011* (figure 1-31), new VNets *vnet-nwkt-prod-euswecen-001,* and *vnet-nwkt-dev-euswecen-001* (figure 1-32) with their associated subnets (figures 1-33, and 1-34).


Home > NWKT

NWKT | Resource groups
Subscription

Name
cloud-shell-storage-westeurope
NetworkWatcherRG
rg-arm-demo-euswecen-011
rg-nwkt-demo-euswecen-001

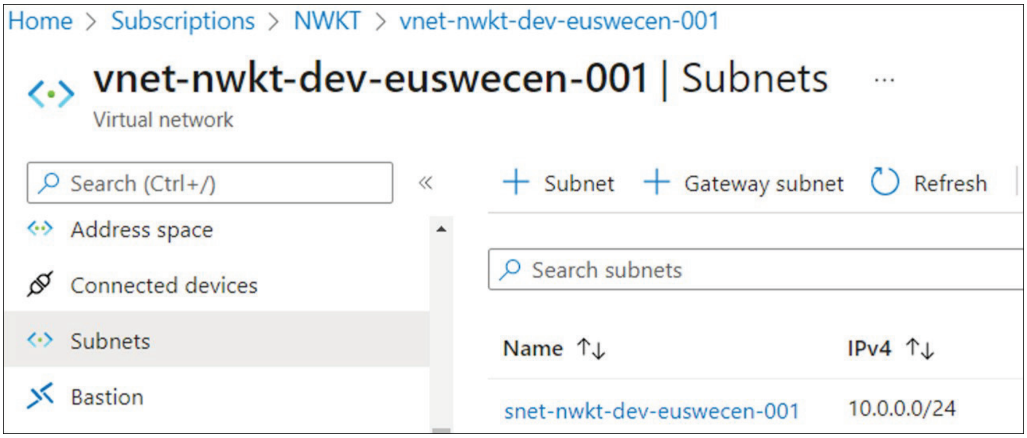
Figure 1-31: *New Resource Group.*


Home > NWKT

NWKT | Resources
Subscription

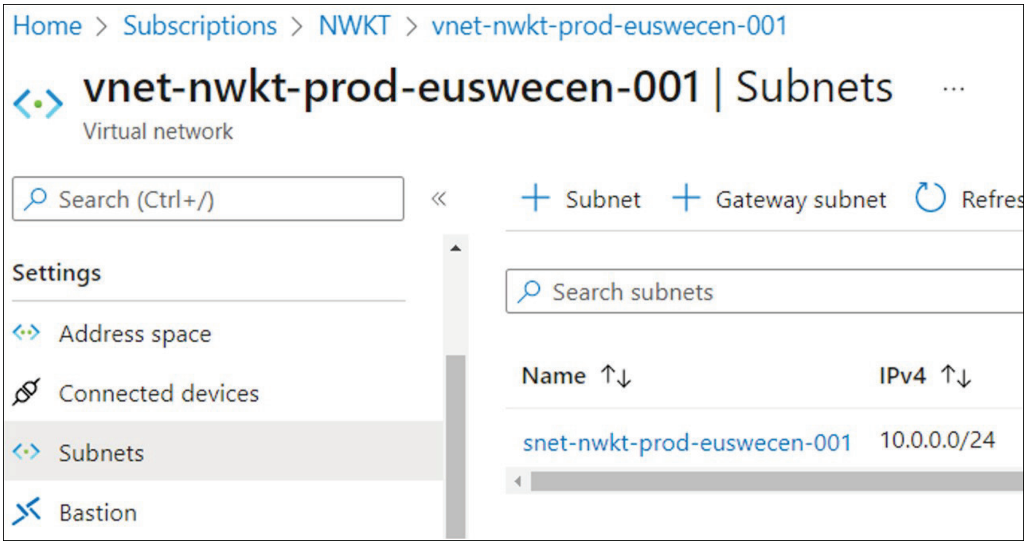
Name	Type
csb10032001eeaa3975	Storage account
NetworkWatcher_swedencentral	Network Watcher
vnet-nwkt-demo-euswecen-001	Virtual network
vnet-nwkt-dev-euswecen-001	Virtual network
vnet-nwkt-prod-euswecen-001	Virtual network

Figure 1-32: *New Virtual Network Resources.*



Name ↑↓	IPv4 ↑↓
snet-nwkt-dev-euswecen-001	10.0.0.0/24

Figure 1-33: *New Subnet 10.0.0.0/24 for VNet vnet-nwkt-dev-euswecen-001.*



Name ↑↓	IPv4 ↑↓
snet-nwkt-prod-euswecen-001	10.0.0.0/24

Figure 1-34: *New Subnet 10.0.0.0/24 for VNet vnet-nwkt-prod-euswecen-001.*

Examples 1-7 and 1-8 show how we can get information about Virtual Networks using Azure AZ PowerShell.

```
PS /home/toni_pasanen> Get-AzVirtualNetwork -Name vnet-nwkt-prod-euswecen-001

Name                   : vnet-nwkt-prod-euswecen-001
ResourceGroupName      : rg-arm-demo-euswecen-011
Location               : swedencentral
Id                     : /subscriptions/***/resourceGroups/rg-arm-demo-euswecen-
011/providers/Microsoft.Network/virtualNetworks/vnet-nwkt-prod-euswecen-001
Etag                   : W/"***"
ResourceGuid           : ***
ProvisioningState      : Succeeded
Tags                   :
                         Name  Value
                         ====  =====
                         app   prod

AddressSpace           : {
                           "AddressPrefixes": [
                             "10.0.0.0/16"
                           ]
                         }
DhcpOptions            : null
FlowTimeoutInMinutes   : null
Subnets                : [
                           {
                             "Delegations": [],
                             "Name": "snet-nwkt-prod-euswecen-001",
                             "Etag": "W/\"***\"",
                             "Id": "/subscriptions/***/resourceGroups/rg-arm-demo-
euswecen-011/providers/Microsoft.Network/virtualNetworks/vnet-nwkt-prod-euswec
                         en-001/subnets/snet-nwkt-prod-euswecen-001",
                             "AddressPrefix": [
                               "10.0.0.0/24"
                             ],
                             "IpConfigurations": [],
                             "ServiceAssociationLinks": [],
                             "ResourceNavigationLinks": [],
                             "ServiceEndpoints": [],
                             "ServiceEndpointPolicies": [],
                             "PrivateEndpoints": [],
                             "ProvisioningState": "Succeeded",
                             "PrivateEndpointNetworkPolicies": "Enabled",
                             "PrivateLinkServiceNetworkPolicies": "Enabled",
                             "IpAllocations": []
                           }
                         ]
VirtualNetworkPeerings : []
EnableDdosProtection   : false
DdosProtectionPlan     : null
ExtendedLocation       : null
```

Example 1-7: *Verification Using Azure AZ PowerShell-1.*

```
PS /home/toni_pasanen> Get-AzVirtualNetwork -Name vnet-nwkt-dev-euswecen-001

Name                   : vnet-nwkt-dev-euswecen-001
ResourceGroupName      : rg-arm-demo-euswecen-011
Location               : swedencentral
Id                     : /subscriptions/***/resourceGroups/rg-arm-demo-euswecen-
011/providers/Microsoft.Network/virtualNetworks/vnet-nwkt-dev-euswecen-001
Etag                   : W/"***"
ResourceGuid           : ***
ProvisioningState      : Succeeded
Tags                   :
                         Name  Value
                         ====  =====
                         app   dev

AddressSpace           : {
                           "AddressPrefixes": [
                             "10.0.0.0/16"
                           ]
                         }
DhcpOptions            : null
FlowTimeoutInMinutes   : null
Subnets                : [
                           {
                             "Delegations": [],
                             "Name": "snet-nwkt-dev-euswecen-001",
                             "Etag": "W/\"***\"",
                             "Id": "/subscriptions/***/resourceGroups/rg-arm-demo-
euswecen-011/providers/Microsoft.Network/virtualNetworks/vnet-nwkt-dev-euswece
                         n-001/subnets/snet-nwkt-dev-euswecen-001",
                             "AddressPrefix": [
                               "10.0.0.0/24"
                             ],
                             "IpConfigurations": [],
                             "ServiceAssociationLinks": [],
                             "ResourceNavigationLinks": [],
                             "ServiceEndpoints": [],
                             "ServiceEndpointPolicies": [],
                             "PrivateEndpoints": [],
                             "ProvisioningState": "Succeeded",
                             "PrivateEndpointNetworkPolicies": "Enabled",
                             "PrivateLinkServiceNetworkPolicies": "Enabled",
                             "IpAllocations": []
                           }
                         ]
VirtualNetworkPeerings : []
EnableDdosProtection   : false
DdosProtectionPlan     : null
ExtendedLocation       : null
```

Example 1-8: *Verification Using Azure AZ PowerShell.*

References

[1] Define your naming convention:
https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming

[2] Recommended abbreviations for Azure resource types:
https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-abbreviations

[3] Resource naming and tagging decision guide
https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/decision-guides/resource-tagging/?toc=/azure/azure-resource-manager/management/toc.json

[4] Define your tagging strategy
https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-tagging

[5] Develop your naming and tagging strategy for Azure resources
https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/naming-and-tagging

[6] Introducing JSON
https://www.json.org/json-en.html

[7] The JavaScript Object Notation (JSON) Data Interchange Format
T. Bray Dec 2017
https://www.rfc-editor.org/rfc/rfc8259.txt

[8] ECMA-404: The JSON data interchange syntax. 2nd edition Dec 2017
https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

[9] Azure PowerShell Documentation
https://docs.microsoft.com/en-us/powershell/azure/?view=azps-7.5.0

[10] Understand the structure and syntax of ARM templates, 03/18/2022
https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/syntax

[11] Create Resource Manager parameter file
https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/parameter-files

[12] Data residency in Azure
https://azure.microsoft.com/en-us/global-infrastructure/data-residency/#overview

[13] Regions and availability zones
https://docs.microsoft.com/en-us/azure/availability-zones/az-overview

Chapter 2: Network Security Groups (NSG)

Introduction

The Network Security Group (NSG) is a stateful packet filter with zero or more Inbound/Outbound security rules. Each rule includes IP level source and destination (Any, IP address, Subnet, or Application Security Group), Pre-defined Services like HTTP and HTTPS or custom services with the destination protocol (Any, TCP, UDP, and ICMP), and port ranges. Policy rules within NSG are processed from the lowest to highest rule-based priority value. Each NSG also has default Inbound/Outbound security rules allowing traffic a) between subnets within VNet, b) traffic from Load Balancer, and c) traffic to the Internet. All other Inbound/Outbound traffic is denied by default. NSGs are processed after the Network Address Translation (NAT). It means that the destination IP address/subnet on the inbound security rule has to be VM's private IP address, even when we have attached a public IP address to VM. Each subscription may have up to 5000 NSGs, and one NSG can have 1000 security rules.

This chapter introduces three NSG scenarios. The first example explains the NSG-NIC association. In this section, we create a VM that acts as a Bastion host*). Instead of using the Azure Bastion service, we deploy a custom-made vm-Bastion to snet-dmz and allow SSH connection from the external network. The second example describes the NSG-Subnet association. In this section, we launch vm-Front-1 in the front-end subnet. Then we deploy an NSG that allows SSH connection from the Bastion host IP address. The last part of the chapter introduces an Application Security Group (ASG), which we are using to form a logical VM group. We can then use the ASG as a destination in the security rule in NSG. There are two ASGs in figure 2-1. We can create a logical group of VMs by associating them with the same Application Security Group (ASG). The ASG can then be used as a source or destination in NSG security rules. In our example, we have two ASGs, asg-Back (associated with VMs 10.0.2.4-6) and asg-Back#2 (associated with VMs 10.0.2.7-9). The first ASG (asg-Back) is used as a destination in the security rule on the NSG nsg-Back that allows ICMP from VM vm-Front-1. The second ASG (asg-Back#2) is used as a destination in the security rule on the same NSG nsg-Back that allows ICMP from VM vm-Bastion. Examples 1-7 and 1-8 show how we can get information about Virtual Networks using Azure AZ PowerShell.

*) Azure Bastion is a managed service for allowing SSH and RDP connections to VMs without a public IP address. Azure Bastion has a fixed price per hour and outbound data traffic-based charge.

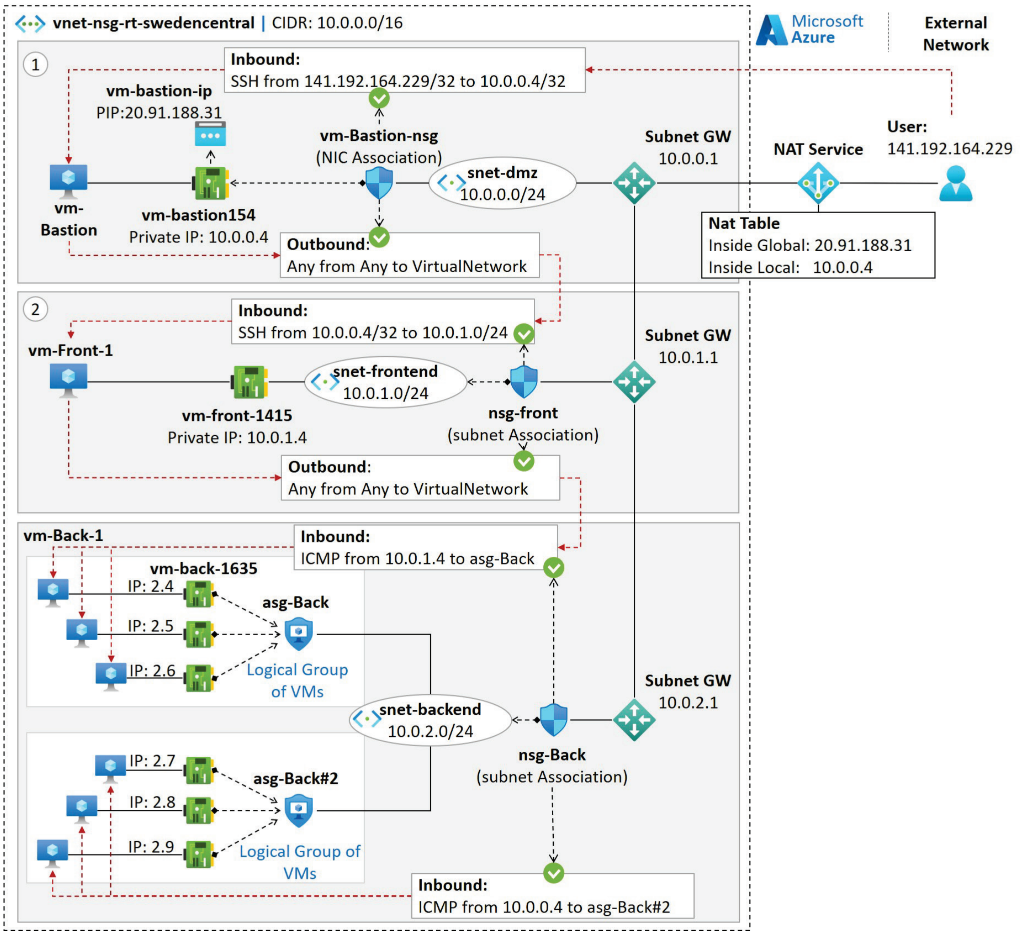


Figure 2-1: *Network Security Group (NSG) – Example Scenarios.*

VM to NSG Association

Step-1: Deploy VM vm-Bastion

I have deployed subnets snet-dmz, snet-frontend, and snet-backend into VNet vnet-nsg-rt-swedencentral. The VNet deploying processes are described in the first chapter.

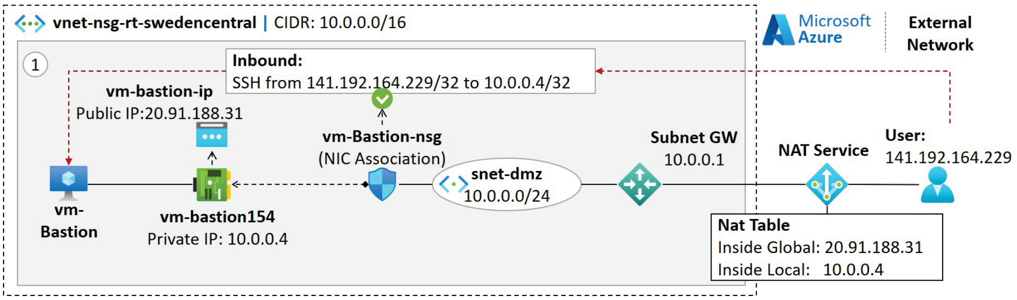
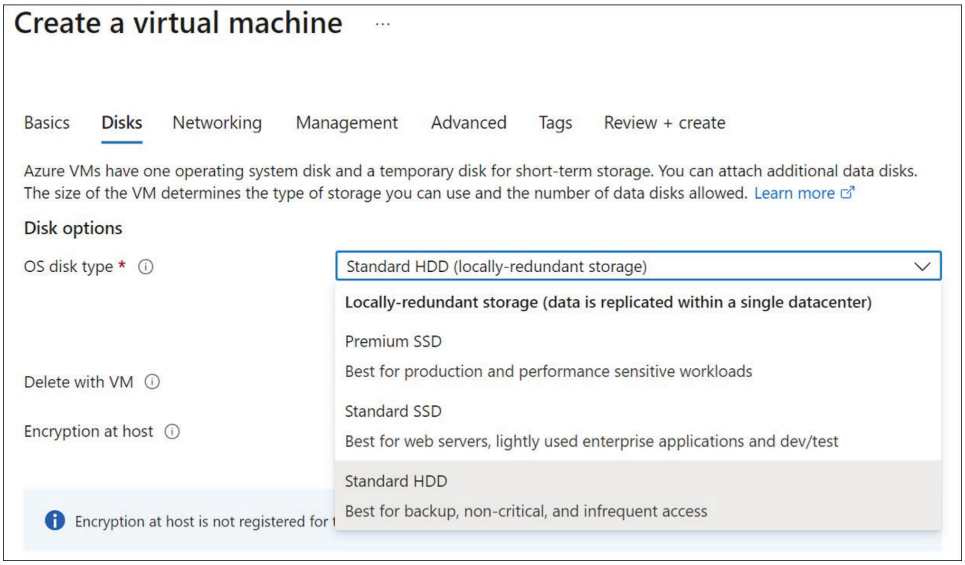


Figure 2-2: *Example VNets: snet-dmz, snet-frontend, and snet-backend.*

We start by creating a Virtual Machine vm-Bastion. Navigate to Azure Portal Home/Subscription/<subscription>, and select *Resources* under the *Settings* section in the left pane. By clicking the +*Create* button, you will be forwarded to *Create a resource* page. Select *Create* under *Virtual Machine*. I am using VM size *Standard_B1ls* (1 CPU, 0.5 GB memory, $3.94/month), and that option is available only with *Standard HDD* disk. For this reason, start by changing the default disk type *Standard SDD* to *Standard HDD*. I am using *Standard_B1ls* because it is the cheapest option.



Azure VMs have one operating system disk and a temporary disk for short-term storage. You can attach additional data disks. The size of the VM determines the type of storage you can use and the number of data disks allowed. Learn more

Disk options

OS disk type * : Standard HDD (locally-redundant storage)

Locally-redundant storage (data is replicated within a single datacenter)

Premium SSD
Best for production and performance sensitive workloads

Standard SSD
Best for web servers, lightly used enterprise applications and dev/test

Standard HDD
Best for backup, non-critical, and infrequent access

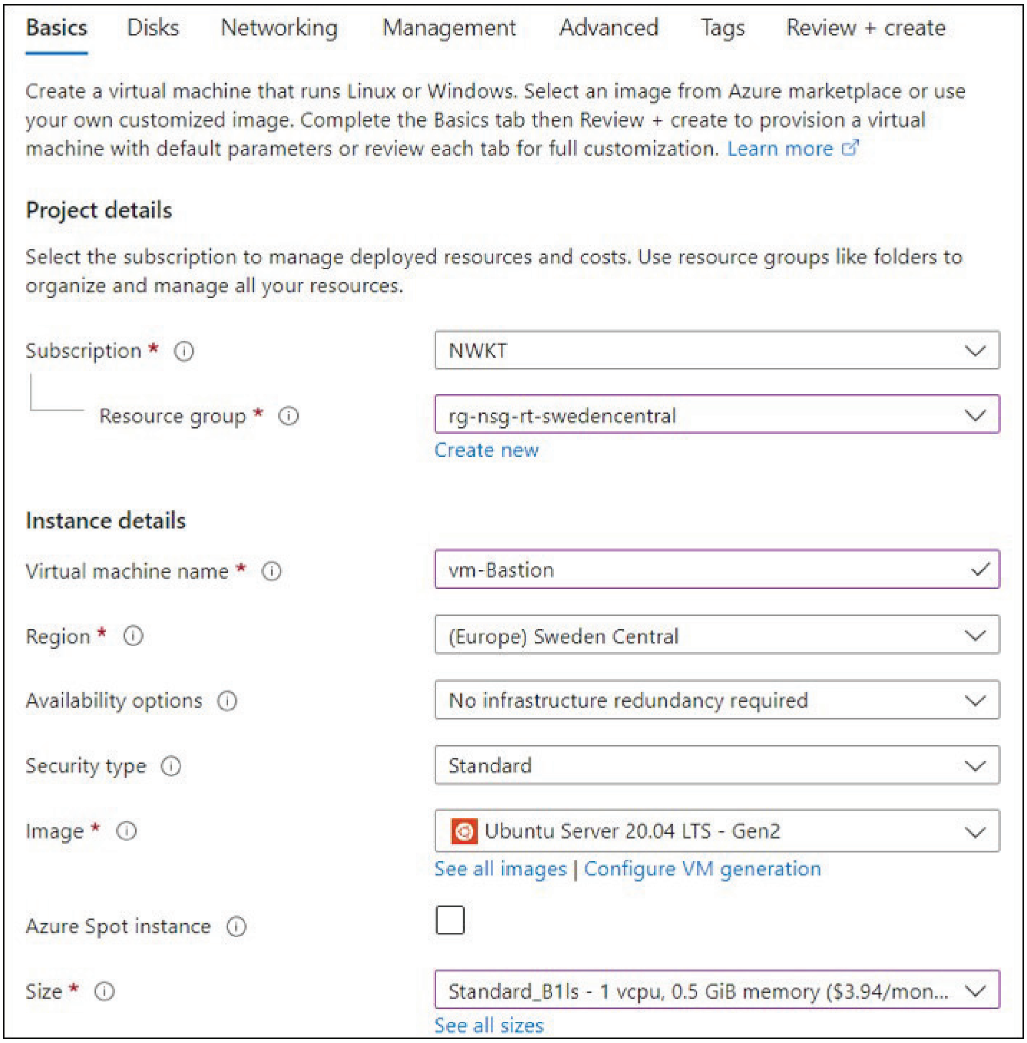
Delete with VM

Encryption at host

Encryption at host is not registered for t

Figure 2-3: *Create Virtual Machine: Select Disk.*

After the disk selection, go to the *Basics* tab. Then select the Subscription and Resource Group. Then name the VM and choose the Region. Leave the Availability and the Security type to their defaults. Select your preferred Image and the size of the VM (figure 2-5).



Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

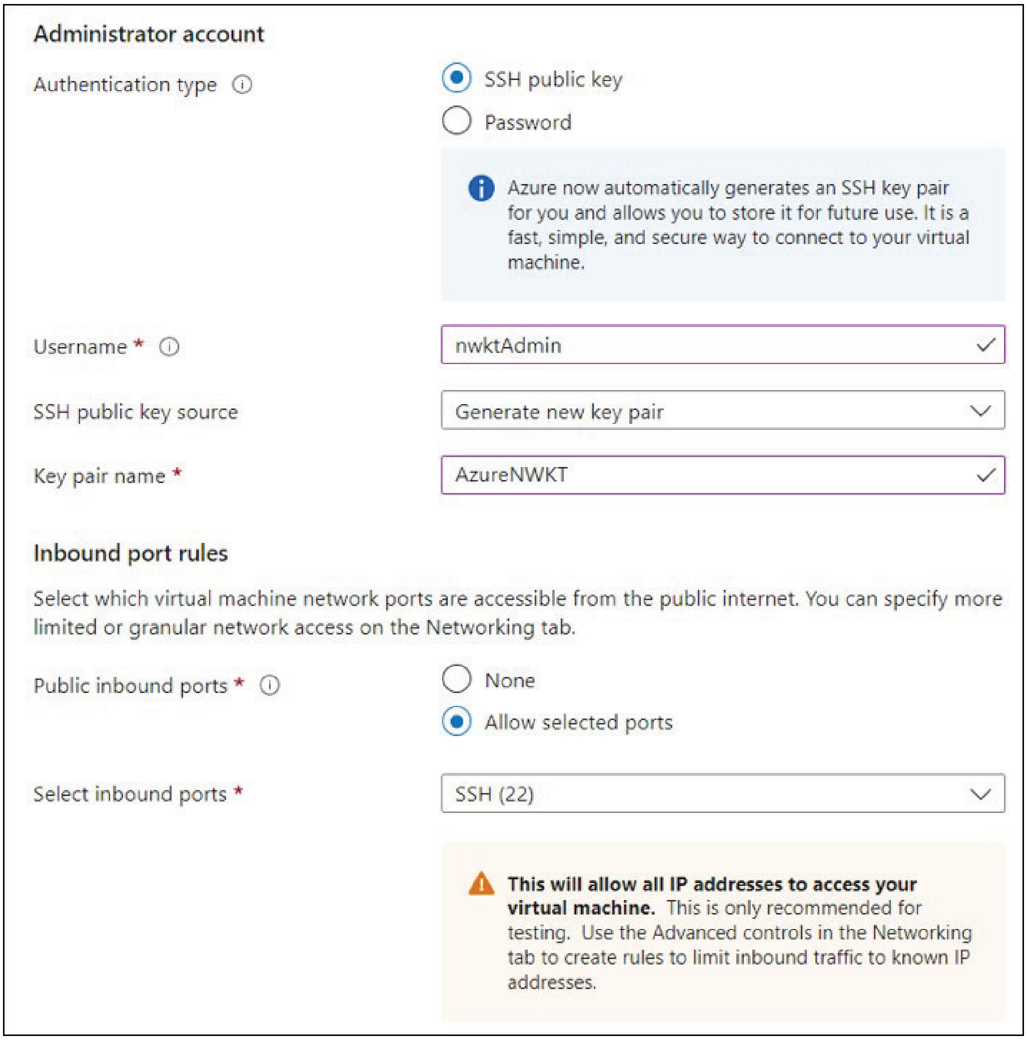
Subscription *	NWKT
Resource group *	rg-nsg-rt-swedencentral
	Create new

Instance details

Virtual machine name *	vm-Bastion
Region *	(Europe) Sweden Central
Availability options	No infrastructure redundancy required
Security type	Standard
Image *	Ubuntu Server 20.04 LTS - Gen2
	See all images | Configure VM generation
Azure Spot instance	☐
Size *	Standard_B1ls - 1 vcpu, 0.5 GiB memory ($3.94/mon...
	See all sizes

Figure 2-4: *Create Virtual Machine: Basics – Project Details and Instance Details.*

Scroll down to the *Administrator account* section and select the *SSH public key* as the Authentication type. Give the username and generate a new SSH public key. Next, select the Allow selected ports radio button and choose SSH (22) as an inbound port. This assigns a dynamic public IP address (vm-Bastion-ip) to VM and creates an NSG vm-Bastion-nsg that allows SSH connection from any source to VM (to its' private IP address).



Administrator account

Authentication type: [✓ SSH public key] Password

Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.

Username *: nwktAdmin

SSH public key source: Generate new key pair

Key pair name *: AzureNWKT

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

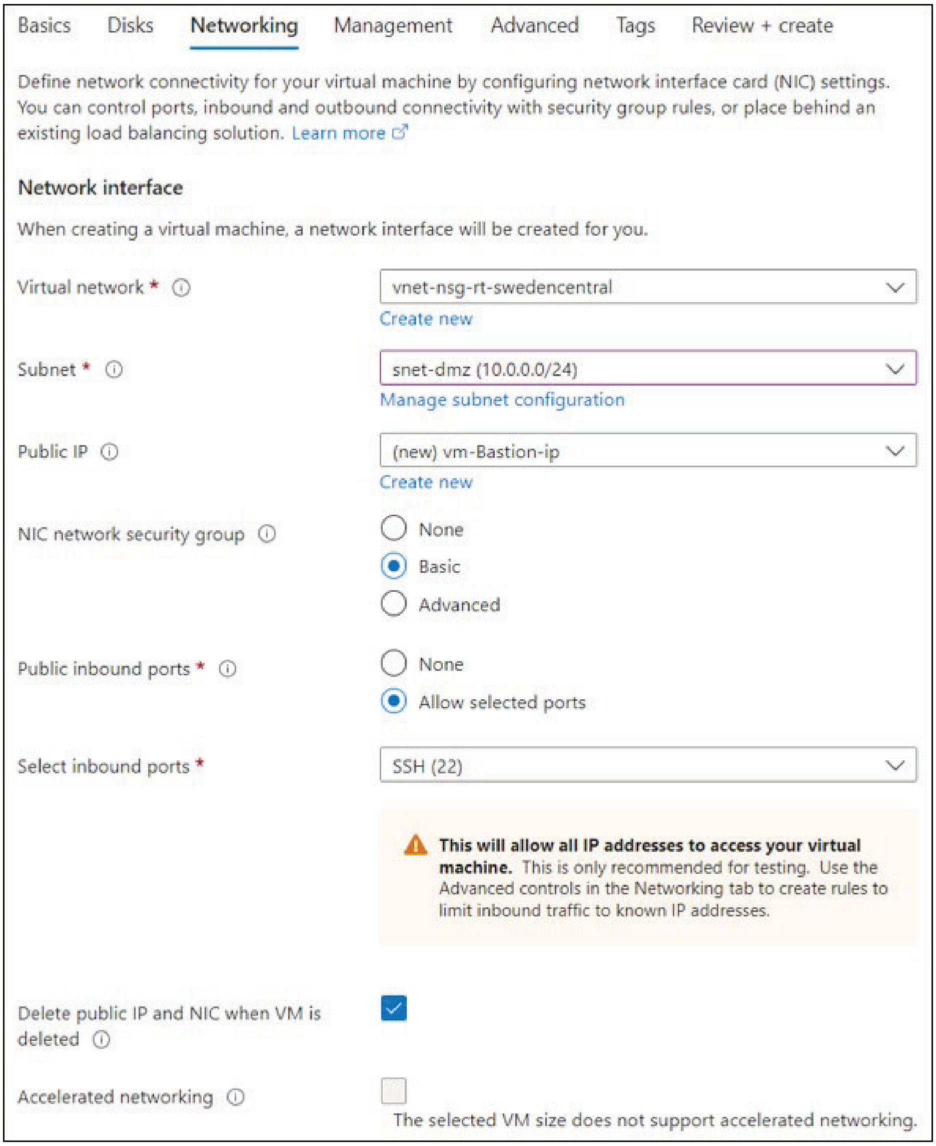
Public inbound ports *: None [✓ Allow selected ports]

Select inbound ports *: SSH (22)

This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

Figure 2-5: *Create Virtual Machine: Basics – Admin Account and Inbound port rules.*

Go to the *Networking* tab. Select Virtual Network and the subnet. Select the Public IP vm-Basic-ip. Then choose the *Basic* option for the NIC network security group. You also need to specify public inbound ports. I have selected the *Delete public IP and NIC when VM is deleted* option. You may want to leave this option unselected if this is your first public IP address because it is free of charge. If you select this option, you can disassociate the public IP from the VM later.



Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution. Learn more

Network interface

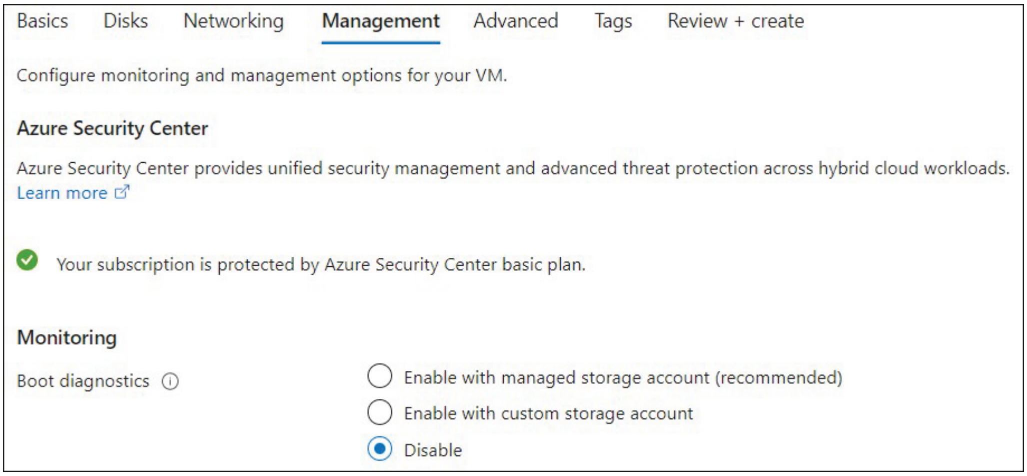
When creating a virtual machine, a network interface will be created for you.

Virtual network *	vnet-nsg-rt-swedencentral (Create new)
Subnet *	snet-dmz (10.0.0.0/24) (Manage subnet configuration)
Public IP	(new) vm-Bastion-ip (Create new)
NIC network security group	None [✓ Basic] Advanced
Public inbound ports *	None [✓ Allow selected ports]
Select inbound ports *	SSH (22)
Delete public IP and NIC when VM is deleted	✓
Accelerated networking	☐ The selected VM size does not support accelerated networking.

This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

Figure 2-6: *Create Virtual Machine: Networking – Network Interface.*

I want to minimize all costs so I disable boot diagnostic.



Configure monitoring and management options for your VM.

Azure Security Center

Azure Security Center provides unified security management and advanced threat protection across hybrid cloud workloads.
Learn more

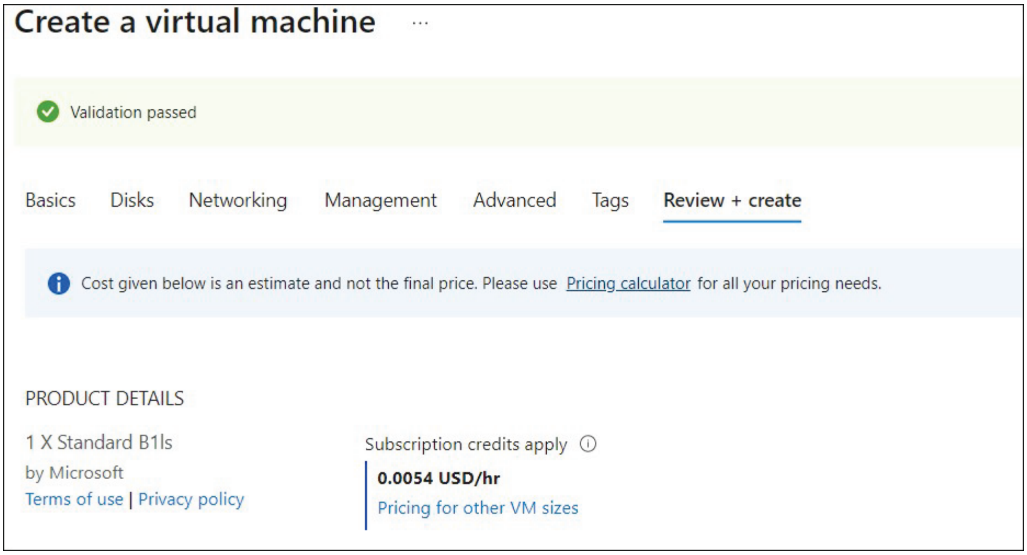
✓ Your subscription is protected by Azure Security Center basic plan.

Monitoring

Boot diagnostics: Enable with managed storage account (recommended) / Enable with custom storage account / [✓ Disable]

Figure 2-7: *Create Virtual Machine: Management.*

Now we are ready to create a VM. If everything looks ok, click the *Create* button (not shown in figure 2-8).



Create a virtual machine

✓ Validation passed

Cost given below is an estimate and not the final price. Please use Pricing calculator for all your pricing needs.

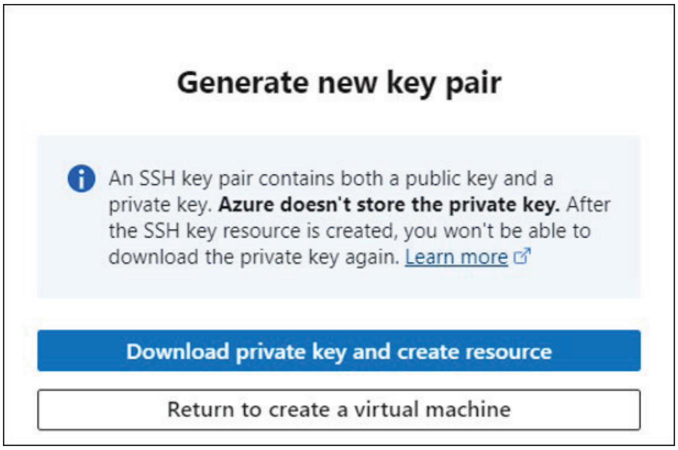
PRODUCT DETAILS

1 X Standard B1ls
by Microsoft
Terms of use | Privacy policy

Subscription credits apply
0.0054 USD/hr
Pricing for other VM sizes

Figure 2-8: *Create Virtual Machine: Review and Create.*

Before the deploying process starts, you have to download the private key.



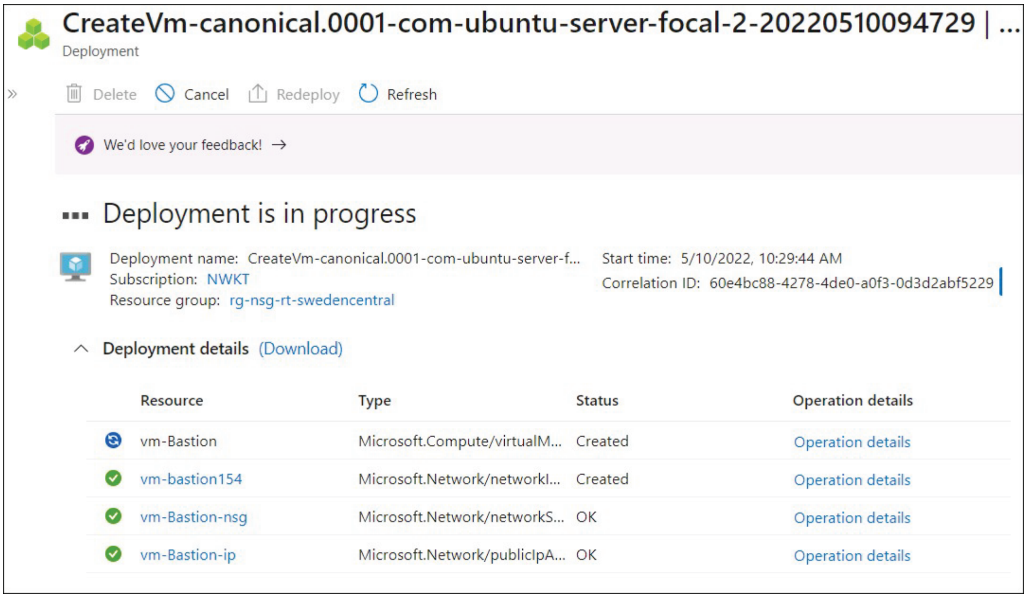
Generate new key pair

An SSH key pair contains both a public key and a private key. **Azure doesn't store the private key.** After the SSH key resource is created, you won't be able to download the private key again. Learn more

Download private key and create resource
Return to create a virtual machine

Figure 2-9: *Create Virtual Machine: Download Private Key and Create.*

Figure 2-10 shows that our deployment includes four resources. From top to bottom, we have Virtual Machine vm-Bastion, Network Interface vm-bastion154, Network Security Group vm-Bastion-nsg, and the public IP address vm-Bastion-ip.



CreateVm-canonical.0001-com-ubuntu-server-focal-2-20220510094729 | ...
Deployment

Deployment is in progress

Deployment name: CreateVm-canonical.0001-com-ubuntu-server-f...
Subscription: NWKT
Resource group: rg-nsg-rt-swedencentral
Start time: 5/10/2022, 10:29:44 AM
Correlation ID: 60e4bc88-4278-4de0-a0f3-0d3d2abf5229

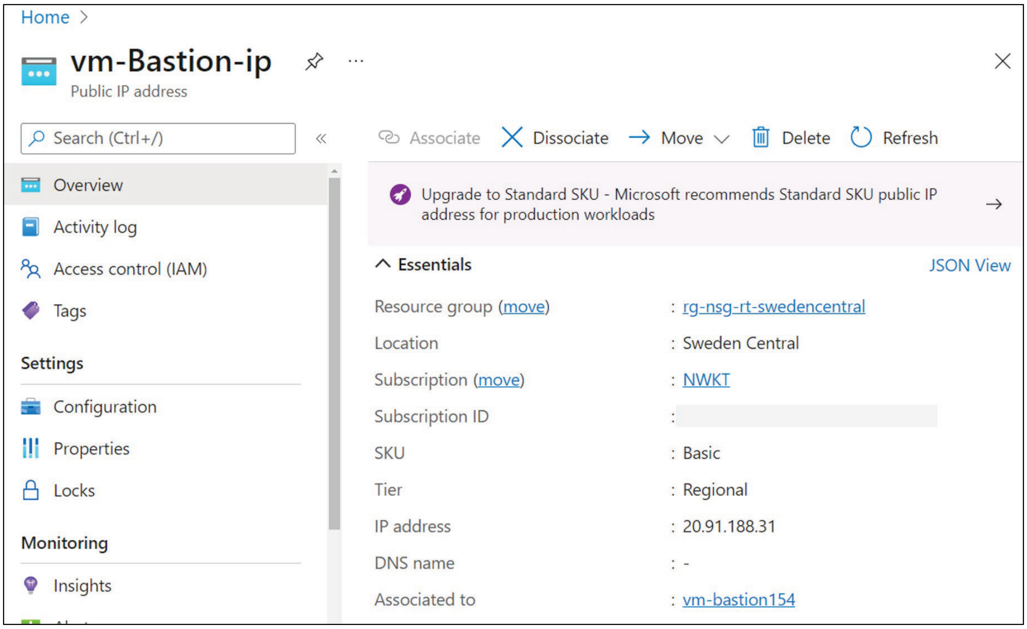
Deployment details (Download)

Resource	Type	Status	Operation details
vm-Bastion	Microsoft.Compute/virtualM...	Created	Operation details
vm-bastion154	Microsoft.Network/networkI...	Created	Operation details
vm-Bastion-nsg	Microsoft.Network/networkS...	OK	Operation details
vm-Bastion-ip	Microsoft.Network/publicIpA...	OK	Operation details

Figure 2-10: *Create Virtual Machine: Deploying Process.*

Public IP Verification

When the deployment process is ready, navigate to *Resources* under the *Settings* section in the left pane on your subscription page and select *vm-Bastion-ip*. The *Overview* option shows the basic information like IP address and its association.


Home >

vm-Bastion-ip

Public IP address

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Settings

Configuration

Properties

Locks

Monitoring

Insights

Associate | Dissociate | Move | Delete | Refresh

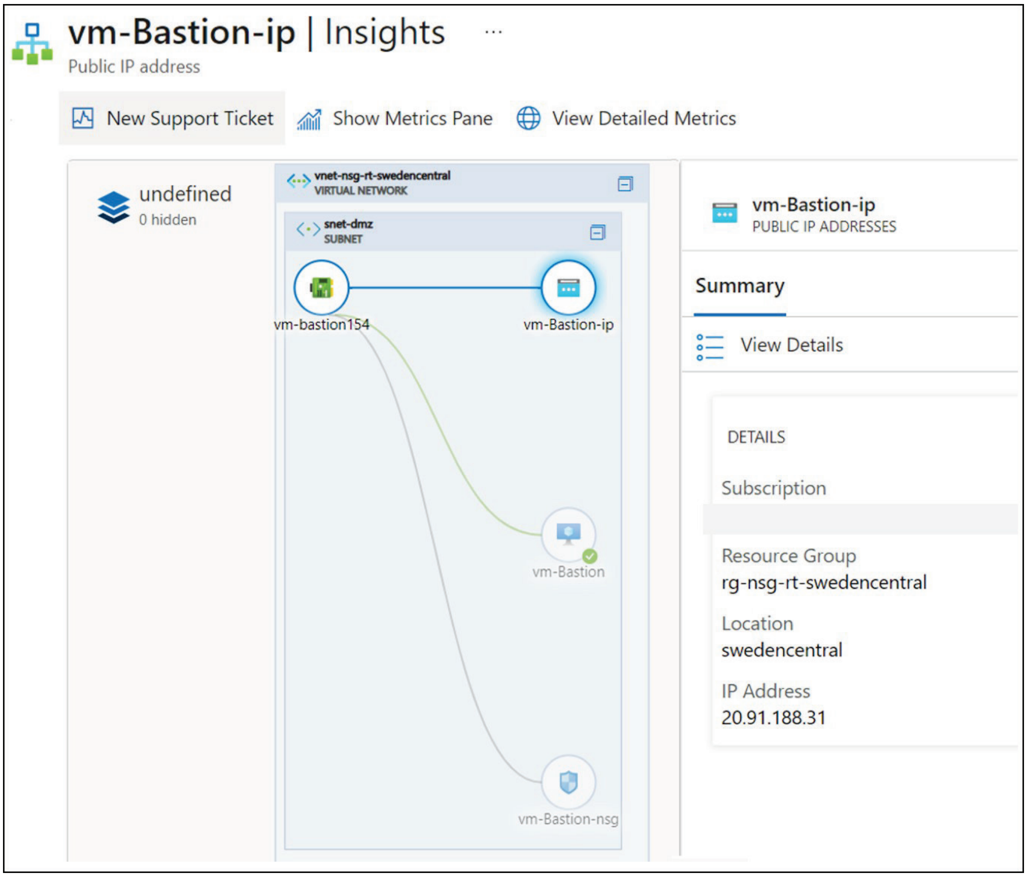
Upgrade to Standard SKU - Microsoft recommends Standard SKU public IP address for production workloads

Essentials — JSON View

Resource group (move)	: rg-nsg-rt-swedencentral
Location	: Sweden Central
Subscription (move)	: NWKT
Subscription ID	: [redacted]
SKU	: Basic
Tier	: Regional
IP address	: 20.91.188.31
DNS name	: -
Associated to	: vm-bastion154

Figure 2-11: *Create Virtual Machine: Public IP Overview.*

You can get a visual overview and summary information by selecting an *Insight* option under the *Monitoring* section. Figure 2-12 shows that the public IP address vm-Bastion-ip is (20.91.188.31) associated with the network interface vm-bastion154. The Insight view also illustrates the Virtual Network and Subnet where we have attached these resources.



Subscription

Resource Group
rg-nsg-rt-swedencentral

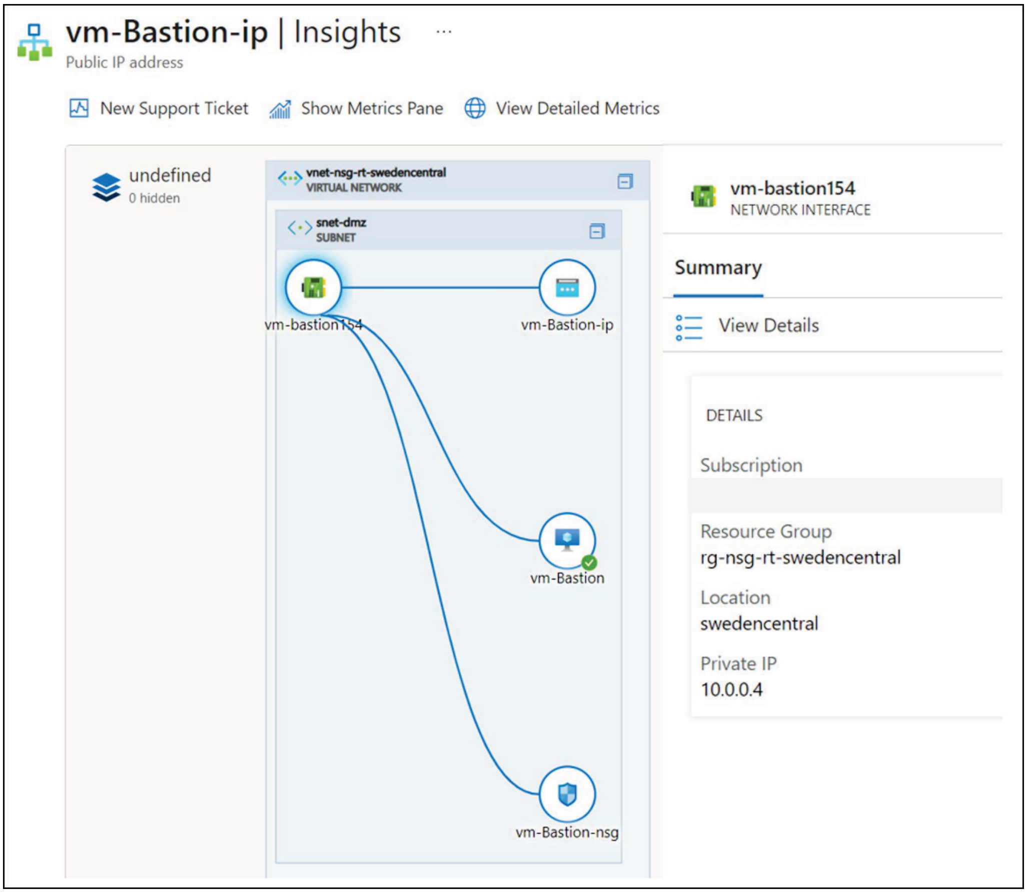
Location
swedencentral

IP Address
20.91.188.31

Figure 2-12: *Create Virtual Machine: Public IP Summary.*

Network Interface Verification

You can see the summary information of the network interface vm-bastion-154 by clicking its icon. We can see that the dynamically assigned private IP address is 10.0.0.4. Click the *View Details* to navigate the network interface vm-bastion154 *Overview* window (figure 2-14).



DETAILS

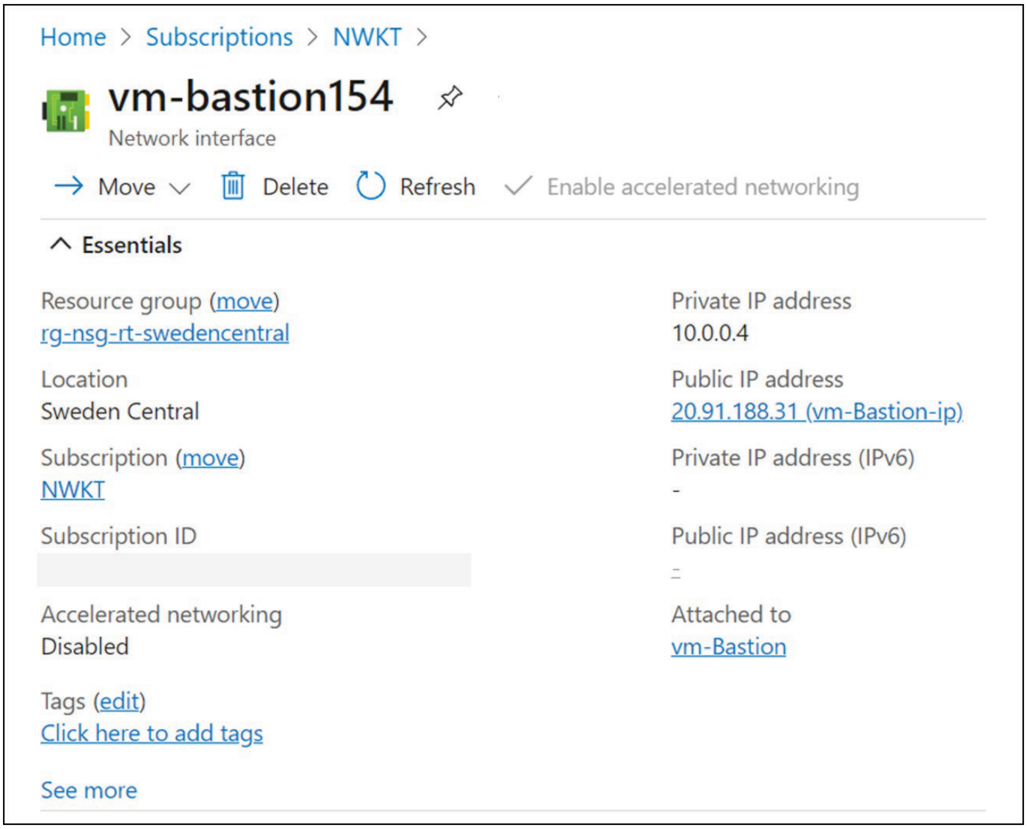
Subscription

Resource Group
rg-nsg-rt-swedencentral

Location
swedencentral

Private IP
10.0.0.4

Figure 2-13: *Create Virtual Machine: Network Interface Summary.*


Home > Subscriptions > NWKT >

vm-bastion154

Network interface

Resource group (move)
rg-nsg-rt-swedencentral

Location
Sweden Central

Subscription (move)
NWKT

Subscription ID

Accelerated networking
Disabled

Tags (edit)
Click here to add tags

Private IP address
10.0.0.4

Public IP address
20.91.188.31 (vm-Bastion-ip)

Private IP address (IPv6)
-

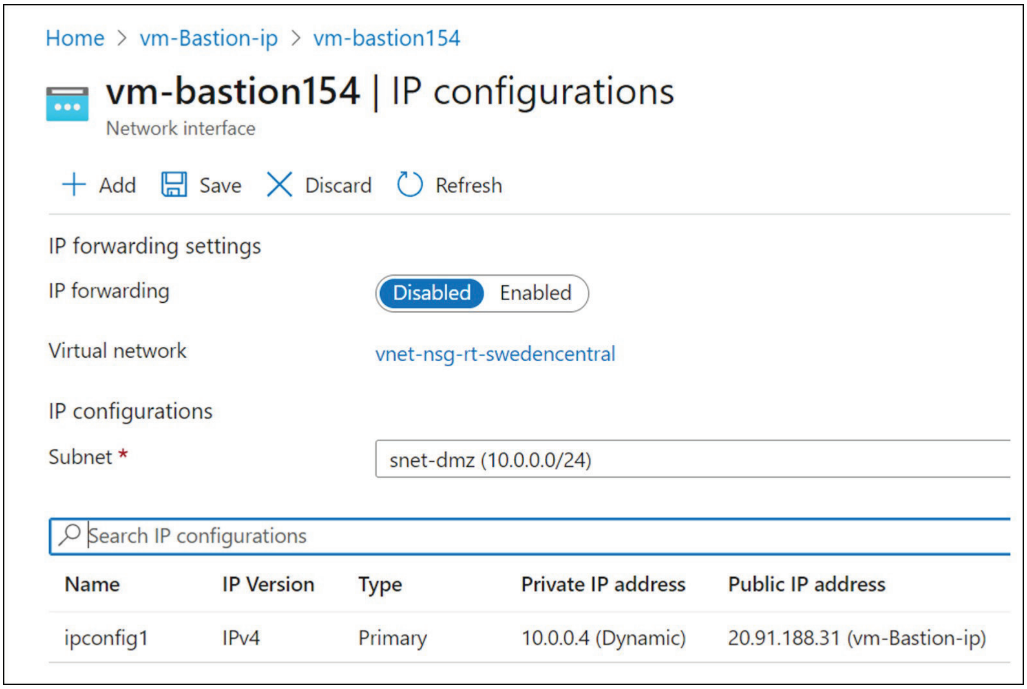
Public IP address (IPv6)
-

Attached to
vm-Bastion

See more

Figure 2-14: *Create Virtual Machine: Network Interface Overview.*

The IP configuration option on the left pane shows the associated subnet and IP addresses. We can see that the private IP is assigned dynamically and that the resource name of the public IP address is vm-bastion-ip. Besides, figure 2-15 shows that the VM associated with the network interface does not forward IP traffic. IP forwarding has to be enabled, for example, if the VM works as a FireWall or Load Balancer. Note, I will exclude the navigation bar from the screen captures from now on. There are two reasons for that. First, this way information window gets more room. Second, the Azure portal displays the option name and its icon. In the figure below, we see that the option *IP configuration* is shown after vm-Base154.



Home > vm-Bastion-ip > vm-bastion154

vm-bastion154 | IP configurations

Network interface

IP forwarding settings

IP forwarding: [✓ Disabled] Enabled

Virtual network: vnet-nsg-rt-swedencentral

IP configurations

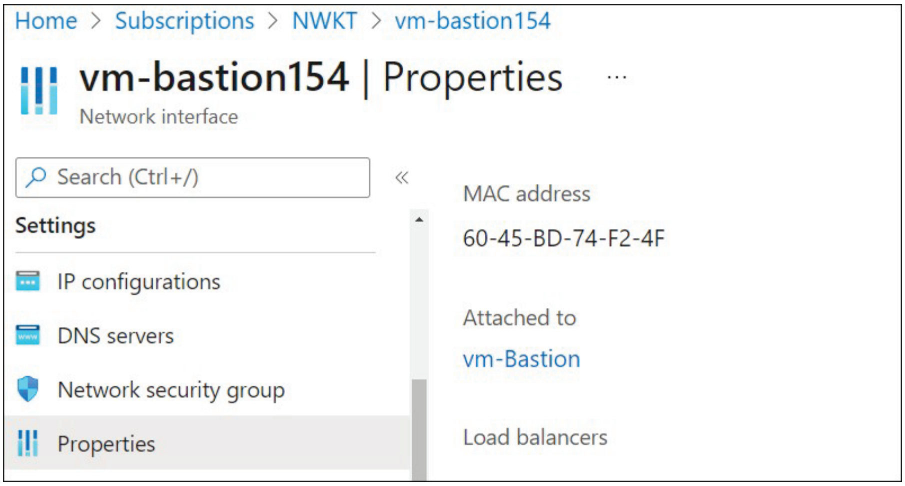
Subnet *: snet-dmz (10.0.0.0/24)

Search IP configurations

Name	IP Version	Type	Private IP address	Public IP address
ipconfig1	IPv4	Primary	10.0.0.4 (Dynamic)	20.91.188.31 (vm-Bastion-ip)

Figure 2-15: *Create Virtual Machine: Network Interface IP Configuration.*

If you want to check the MAC address of the network interface, select the *Properties* option under the Settings section (figure 2-16).



Home > Subscriptions > NWKT > vm-bastion154

vm-bastion154 | Properties ...

Network interface

MAC address

60-45-BD-74-F2-4F

Attached to

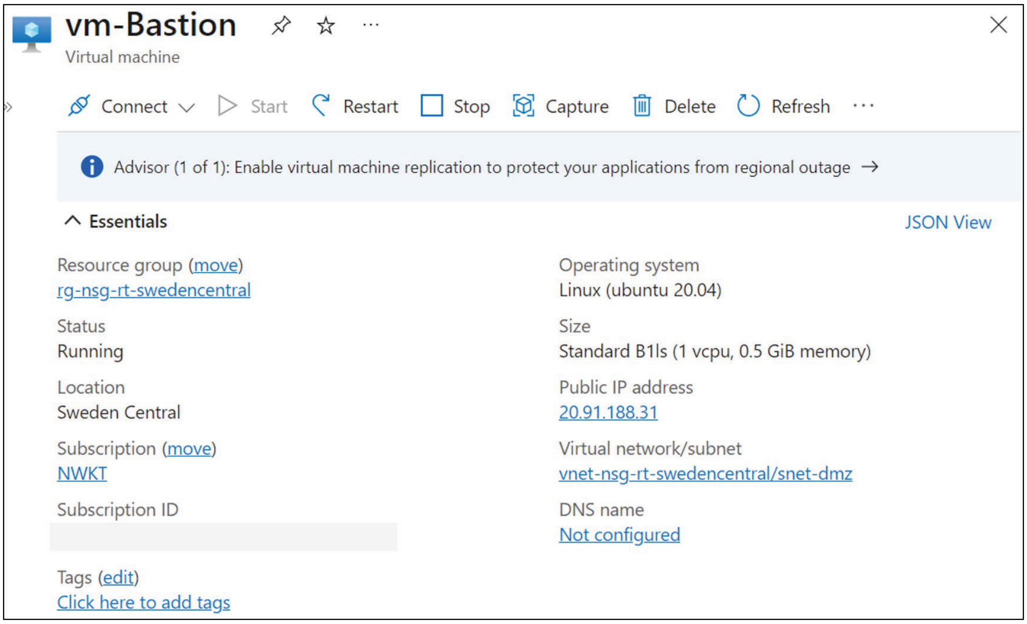
vm-Bastion

Load balancers

Figure 2-16: *Create Virtual Machine: Public IP Overview.*

VM Verification

The *Connect* drop-down menu in the horizontal bar on the Overview page gives you instructions on connection settings to VM using RDP, SSH, or Bastion. You can also manage your VM (stop, start, restart, or delete) in this view. The Essentials section describes the Subscription, Resource Group, and location. It also shows the public IP address and which VNet/Subnet VM is attached.



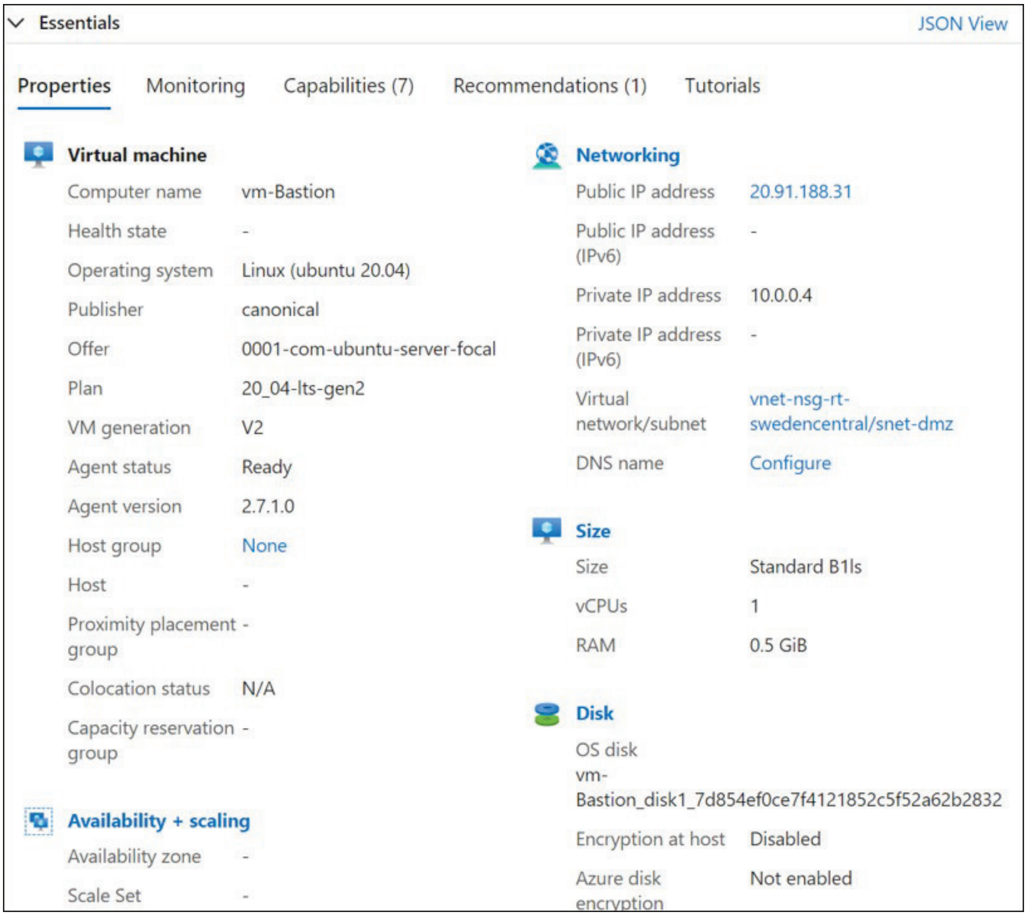
Advisor (1 of 1): Enable virtual machine replication to protect your applications from regional outage →

Essentials

Resource group (move) rg-nsg-rt-swedencentral	Operating system Linux (ubuntu 20.04)
Status Running	Size Standard B1ls (1 vcpu, 0.5 GiB memory)
Location Sweden Central	Public IP address 20.91.188.31
Subscription (move) NWKT	Virtual network/subnet vnet-nsg-rt-swedencentral/snet-dmz
Subscription ID	DNS name Not configured
Tags (edit) Click here to add tags	

Figure 2-17: *Create Virtual Machine: Virtual Machine Overview - Essentials.*

The Properties tab after the Essentials section shows more detailed information about VM (figure 2-18). The monitoring tab, in turn, shows the statistics about CPU, disk, Memory, and Network usage.


Essentials — JSON View

Properties | Monitoring | Capabilities (7) | Recommendations (1) | Tutorials

Virtual machine

Computer name	vm-Bastion
Health state	-
Operating system	Linux (ubuntu 20.04)
Publisher	canonical
Offer	0001-com-ubuntu-server-focal
Plan	20_04-lts-gen2
VM generation	V2
Agent status	Ready
Agent version	2.7.1.0
Host group	None
Host	-
Proximity placement group	-
Colocation status	N/A
Capacity reservation group	-

Availability + scaling

Availability zone	-
Scale Set	-

Networking

Public IP address	20.91.188.31
Public IP address (IPv6)	-
Private IP address	10.0.0.4
Private IP address (IPv6)	-
Virtual network/subnet	vnet-nsg-rt-swedencentral/snet-dmz
DNS name	Configure

Size

Size	Standard B1ls
vCPUs	1
RAM	0.5 GiB

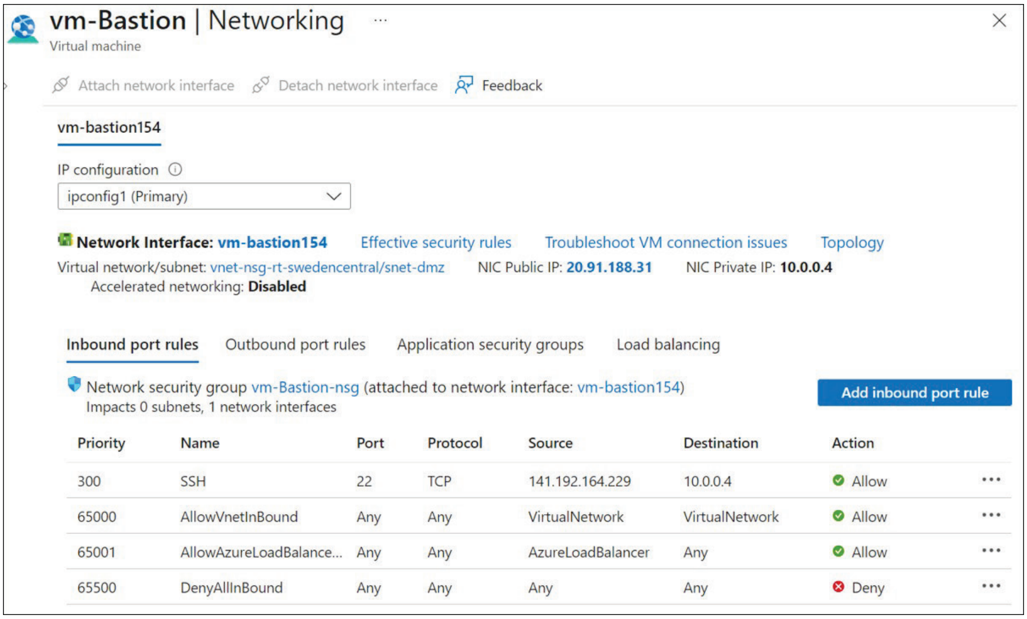
Disk

OS disk
vm-Bastion_disk1_7d854ef0ce7f4121852c5f52a62b2832

Encryption at host	Disabled
Azure disk encryption	Not enabled

Figure 2-18: *Create Virtual Machine: Virtual Machine Overview - Properties.*

The *Networking* option (figure 2-19) shows private and public IP addresses and VNet/Subnet where VM is launched. Each VM has a default NSG with a default policy rule. We added a security rule that allows SSH connection from my remote PC to vm-Bastion. The Inbound and Outbound port rules describe the filters we have configured in the associated NSG. Note that the destination IP address is vm-Bastion's private IP 10.0.0.4. If you allow SSH from any source, you will see an orange warning triangle with an exclamation mark. The Network interface and Network security group names are hyperlinks that you can use to navigate to the resource-specific page.



Network Interface: vm-bastion154
Virtual network/subnet: vnet-nsg-rt-swedencentral/snet-dmz NIC Public IP: 20.91.188.31 NIC Private IP: 10.0.0.4
Accelerated networking: Disabled

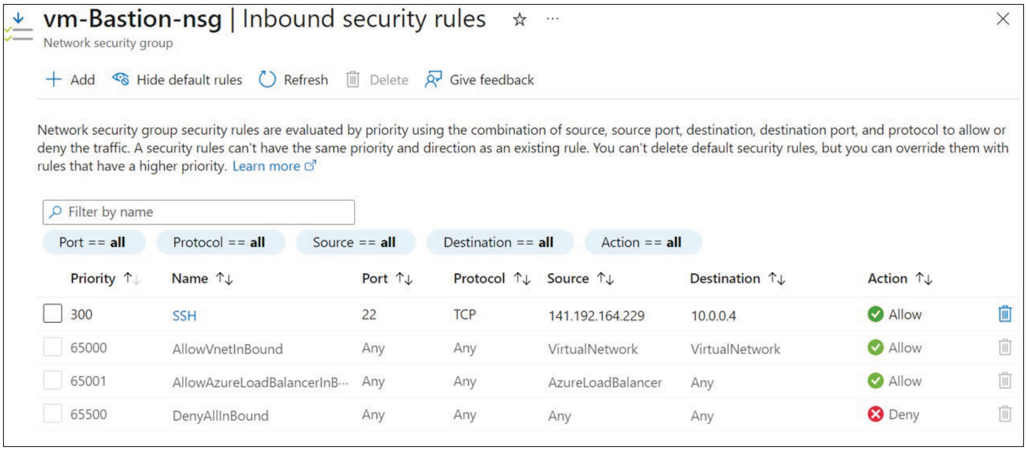
Network security group vm-Bastion-nsg (attached to network interface: vm-bastion154)
Impacts 0 subnets, 1 network interfaces

Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	141.192.164.229	10.0.0.4	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalance...	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Figure 2-19: *Create Virtual Machine: Networking.*

NSG Verification

The Network security group page has its own tabs for Inbound and Outbound security rules. If you want to add a new rule, click the +*Add* button. You can edit an existing security rule, by clicking its name. Remember that modification will be applied to every NIC that the NSG is attached.

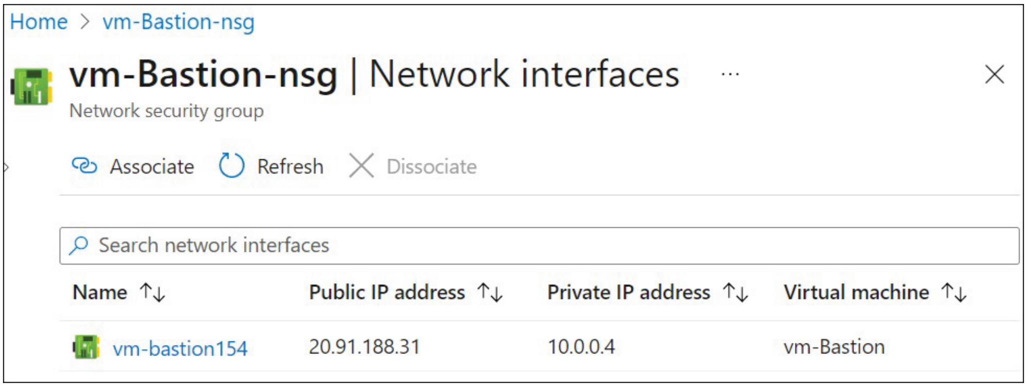


Network security group security rules are evaluated by priority using the combination of source, source port, destination, destination port, and protocol to allow or deny the traffic. A security rules can't have the same priority and direction as an existing rule. You can't delete default security rules, but you can override them with rules that have a higher priority.

Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	141.192.164.229	10.0.0.4	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInB...	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Figure 2-20: *Associated Network Security Group.*

Figure 2-21 shows that the NSG vm-Bastion-nsg is associated with the NIC vm-bastion154.

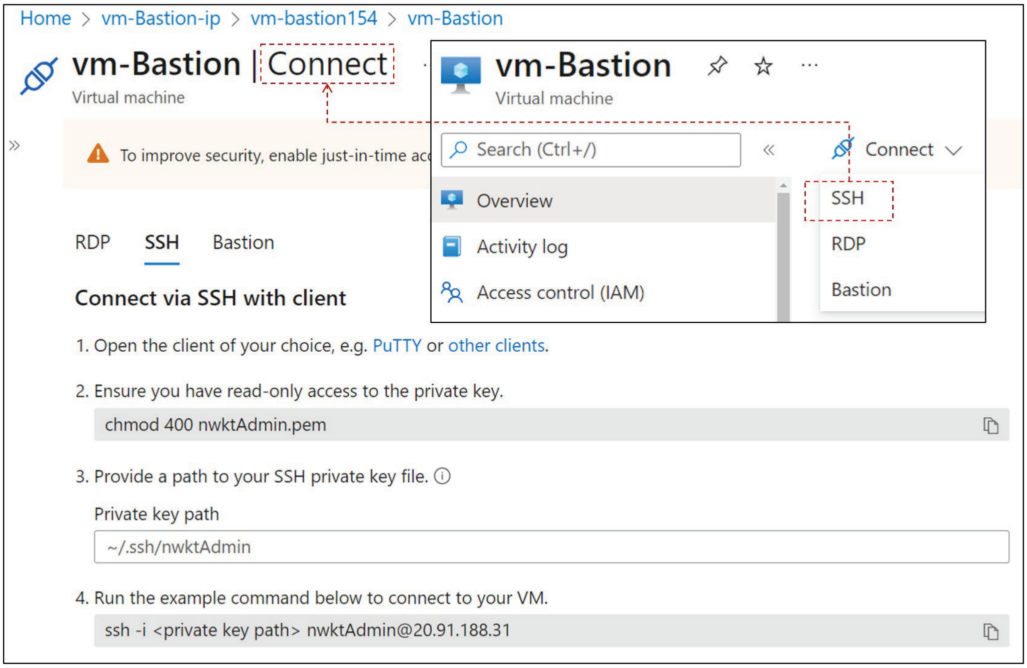


Name	Public IP address	Private IP address	Virtual machine
vm-bastion154	20.91.188.31	10.0.0.4	vm-Bastion

Figure 2-21: *Create Virtual Machine: Public IP Overview.*

Step-2: SSH connection to VM

Select the SSH option from the *Connect* drop-down menu on the VM Overview window to get connection instructions. The command chmod 400 <file name> on the second step gives read-only permission for the file owner, removing all other permissions.



Connect via SSH with client

1. Open the client of your choice, e.g. PuTTY or other clients.
2. Ensure you have read-only access to the private key.

```
chmod 400 nwktAdmin.pem
```

3. Provide a path to your SSH private key file.

Private key path

```
~/.ssh/nwktAdmin
```

4. Run the example command below to connect to your VM.

```
ssh -i <private key path> nwktAdmin@20.91.188.31
```

Figure 2-22: *Create Virtual Machine: Public IP Overview.*

The example below shows the error message when we haven't changed the default PEM file user access rights.

```
C:\01-Toni\Own\01-Azure\***\***>ssh -i AzureNWKT.pem nwktAdmin@20.91.188.31
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions for 'AzureNWKT.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "AzureNWKT.pem": bad permissions
nwktAdmin@20.91.188.31: Permission denied (publickey).
```

Example 2-1: *SSH Connection with Default User Rights.*

Changing PEM File Permission in Windows

To change the file permission in Windows machine, open the file properties and select the *Security* tab. Next, click the *Advanced* button and then *Disable inheritance button* on the Permission tab. Select the *Convert inherited permission into explicit permission on this object*.

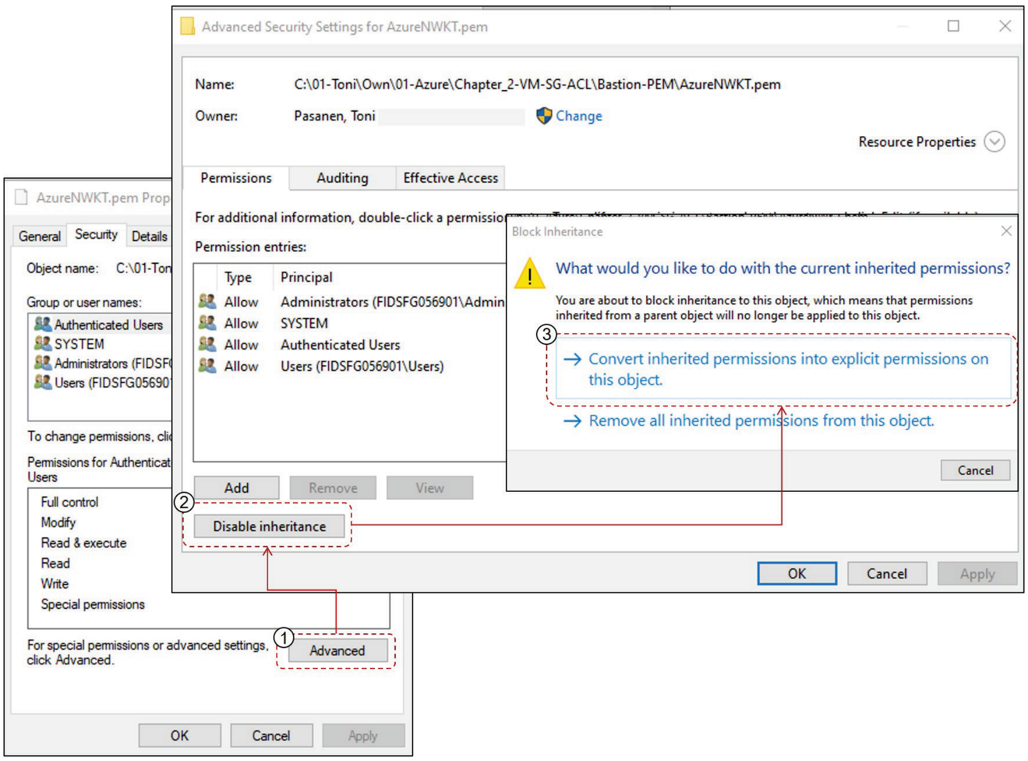

AzureNWKT.pem Prop
General
Security
Details
Object name: C:\01-Ton
Group or user names:
Authenticated Users
SYSTEM
Administrators (FIDSF
Users (FIDSFG05690
To change permissions, cli
Permissions for Authenticat Users
Full control
Modify
Read & execute
Read
Write
Special permissions
For special permissions or advanced settings, click Advanced.
1
Advanced
OK
Cancel
Apply
Advanced Security Settings for AzureNWKT.pem
Name: C:\01-Toni\Own\01-Azure\Chapter_2-VM-SG-ACL\Bastion-PEM\AzureNWKT.pem
Owner: Pasanen, Toni
Change
Resource Properties
Permissions
Auditing
Effective Access
For additional information, double-click a permissio
Permission entries:
Type
Principal
Allow
Administrators (FIDSFG056901\Admin
Allow
SYSTEM
Allow
Authenticated Users
Allow
Users (FIDSFG056901\Users)
Add
Remove
View
2
Disable inheritance
OK
Cancel
Apply
Block Inheritance
What would you like to do with the current inherited permissions?
You are about to block inheritance to this object, which means that permissions inherited from a parent object will no longer be applied to this object.
3
→ Convert inherited permissions into explicit permissions on this object.
→ Remove all inherited permissions from this object.
Cancel

Figure 2-23: *Change PEM File Permission – Step One.*

Then add your user name as file Owner and click the OK button twice.

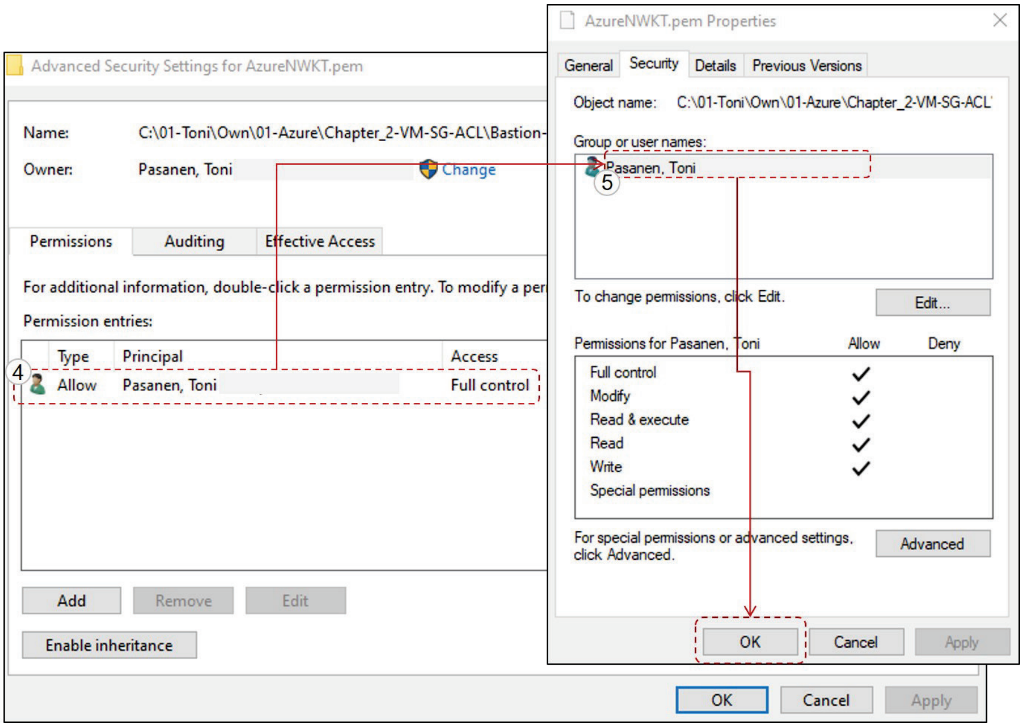


Figure 2-24: *Change PEM File Permission – Step Two.*

Example 2-2 verifies that after modifying PEM file's user rights, we can log in to vm-Bastion.

```
C:\01-Toni\Own\01-Azure\***\***>ssh -i AzureNWKT.pem nwktAdmin@20.91.188.31
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-1022-azure x86_64)

<snipped for brevit>

  System information as of Tue May 10 07:55:08 UTC 2022

  System load:  0.0               Processes:             104
  Usage of /:   4.9% of 28.90GB   Users logged in:       0
  Memory usage: 55%               IPv4 address for eth0: 10.0.0.4
  Swap usage:   0%

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

<snipped for brevity>

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

nwktAdmin@vm-Bastion:~$
nwktAdmin@vm-Bastion:~$
```

Example 2-2: *SSH Connection After User Rights Modifcation.*

Example 2-3 shows how you can see the SSH connection details.

```
nwktAdmin@vm-Bastion:~$ sudo netstat -tpn | grep "ESTABLISHED.*sshd"
tcp        0     36 10.0.0.4:22             141.192.164.229:65383   ESTABLISHED
46776/sshd: nwktAdm
```

Example 2-3: *Verifying the TCP Socket (IP, Protocol, Port) of SSH Connection.*

In order to use the *netstat* command, you need to install *net-tools* to your Linux (example 2-4). You need the elevated privilege to execute the command (sudo = Super User DO).

```
nwktAdmin@vm-Front-1:~$ sudo apt install net-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  net-tools
0 upgraded, 1 newly installed, 0 to remove and 7 not upgraded.
Need to get 196 kB of archives.
After this operation, 864 kB of additional disk space will be used.
Get:1   http://azure.archive.ubuntu.com/ubuntu   focal/main   amd64   net-tools   amd64
1.60+git20180626.aebd88e-1ubuntu1 [196 kB]
Fetched 196 kB in 0s (1808 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 57953 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20180626.aebd88e-1ubuntu1_amd64.deb ...
Unpacking net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Setting up net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Processing triggers for man-db (2.9.1-1) ...
```

Example 2-4: *Install Net-Tools to Linux.*

NSG to Subnet Association

The previous section describes how we deploy a VM and then add the SSH rule to VM's default NSG associated with its Network Interface. This section shows how to create an NSG nsg-front that allows SSH connection from the vm-Bastion's private IP address 10.0.0.4 to all VMs in the subnet 10.0.1.0/24 (snet-frontend). I have deployed a VM vm-Front-1, which I have added to subnet 10.0.1.0/24 (snet-frontend). During the VM 1 setup process, I selected the option *None* for the NIC network security group in the Networking tab (see figure 2-6 to see the Networking tab). This way, the NIC is created without associated NSG.

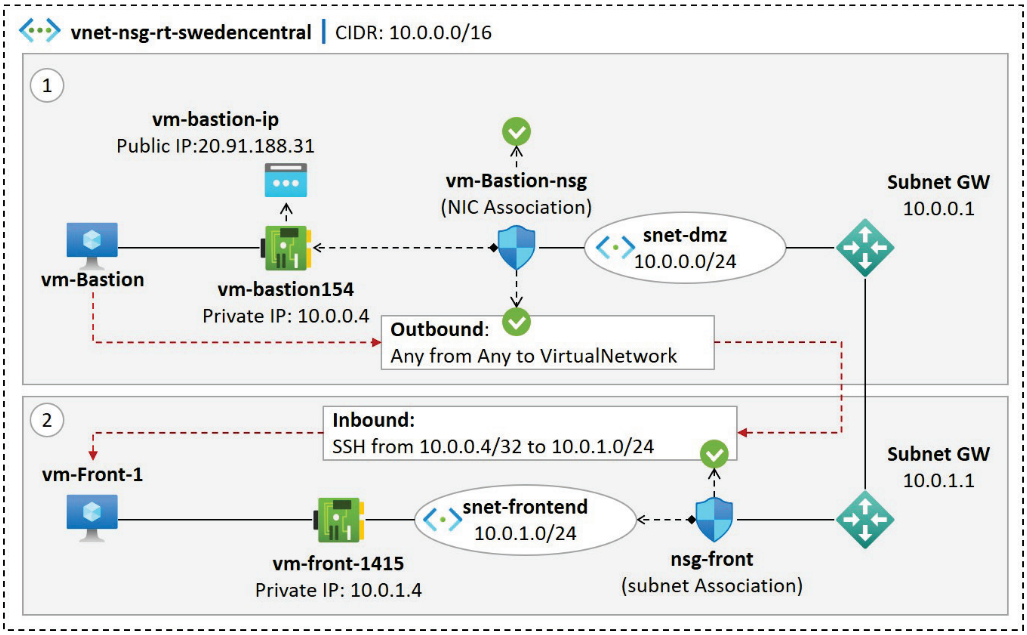


Figure 2-25: *NSG to Subnet Association Overview.*

Step-1: Create New NSG

Select the *Create a resource* from the Azure portal Home view. Then type the *Network security group* on the search field. Next, select the Microsoft Network security group and click the Create button.

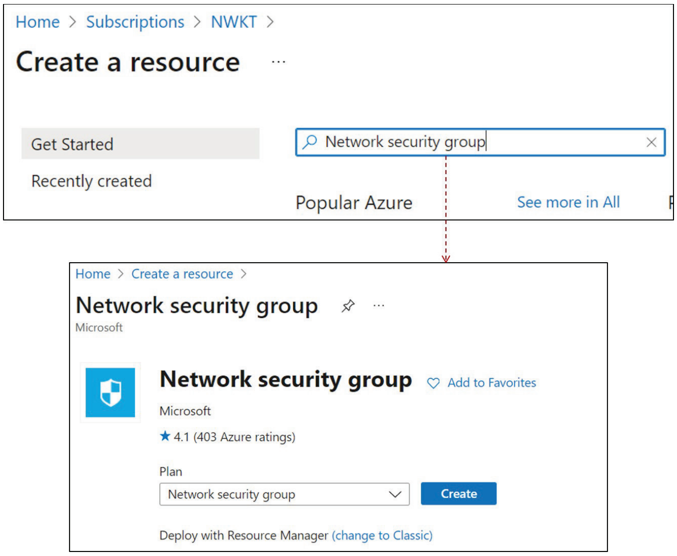
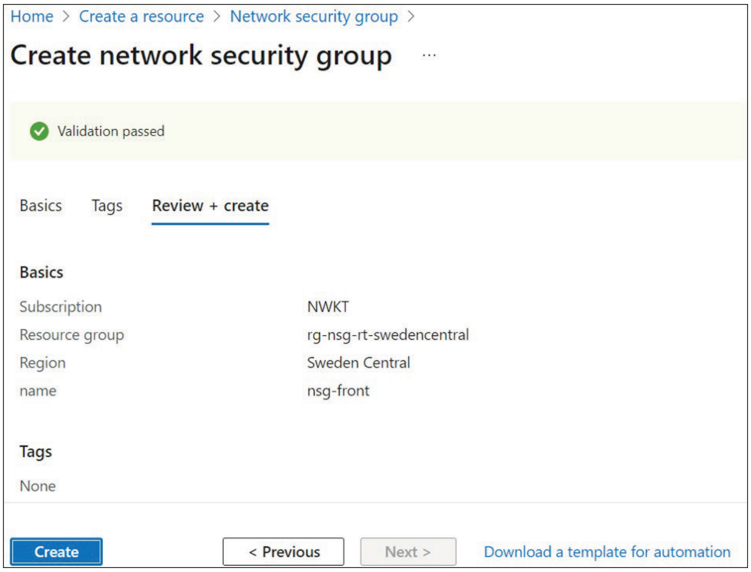


Figure 2-26: *Create Network Security Group.*

Select your subscription, resource group, and region, then give a name to NSG (in the Basic tab). Click the Create button on the *Review + create* tab.


Home > Create a resource > Network security group >
Create network security group ...
Validation passed
Basics
Tags
Review + create
Create
< Previous
Next >
Download a template for automation

Basics

Subscription	NWKT
Resource group	rg-nsg-rt-swedencentral
Region	Sweden Central
name	nsg-front

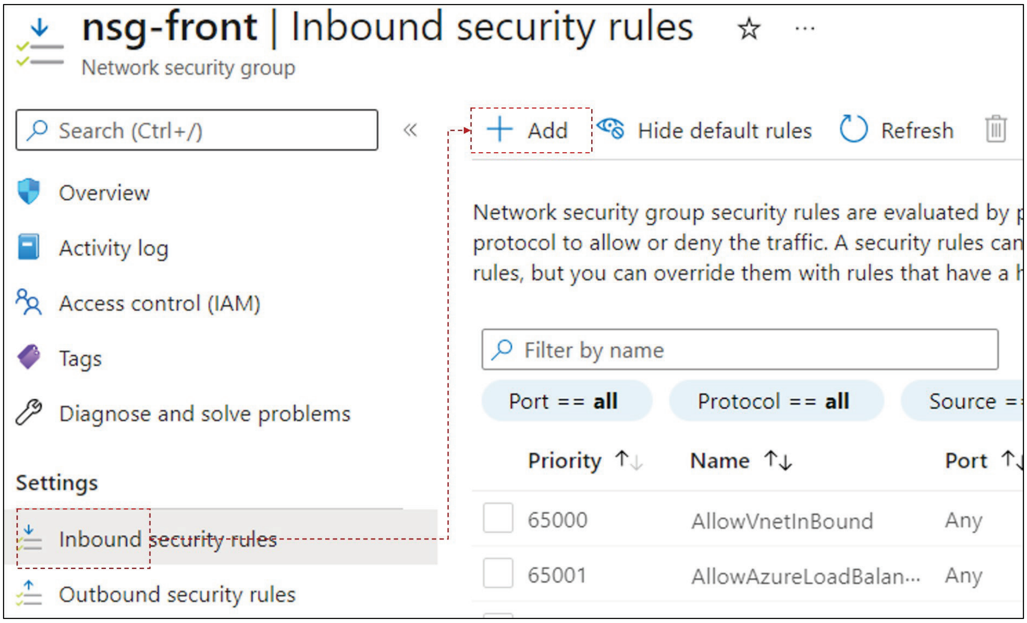
Tags

None

Figure 2-27: *Create Network Security Group.*

Step-2: Add an Inbound Security Rule to NSG

When the NSG deployment process is ready, you can find the NSG from the Resources list. Open the NSG and select the *Inbound security rules* under the *Settings* section on the left pane. Next, create a new policy rule by clicking the +*Add* button.


nsg-front | Inbound security rules

Network security group

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Inbound security rules

Outbound security rules

+ Add | Hide default rules | Refresh

Network security group security rules are evaluated by p
protocol to allow or deny the traffic. A security rules can
rules, but you can override them with rules that have a h

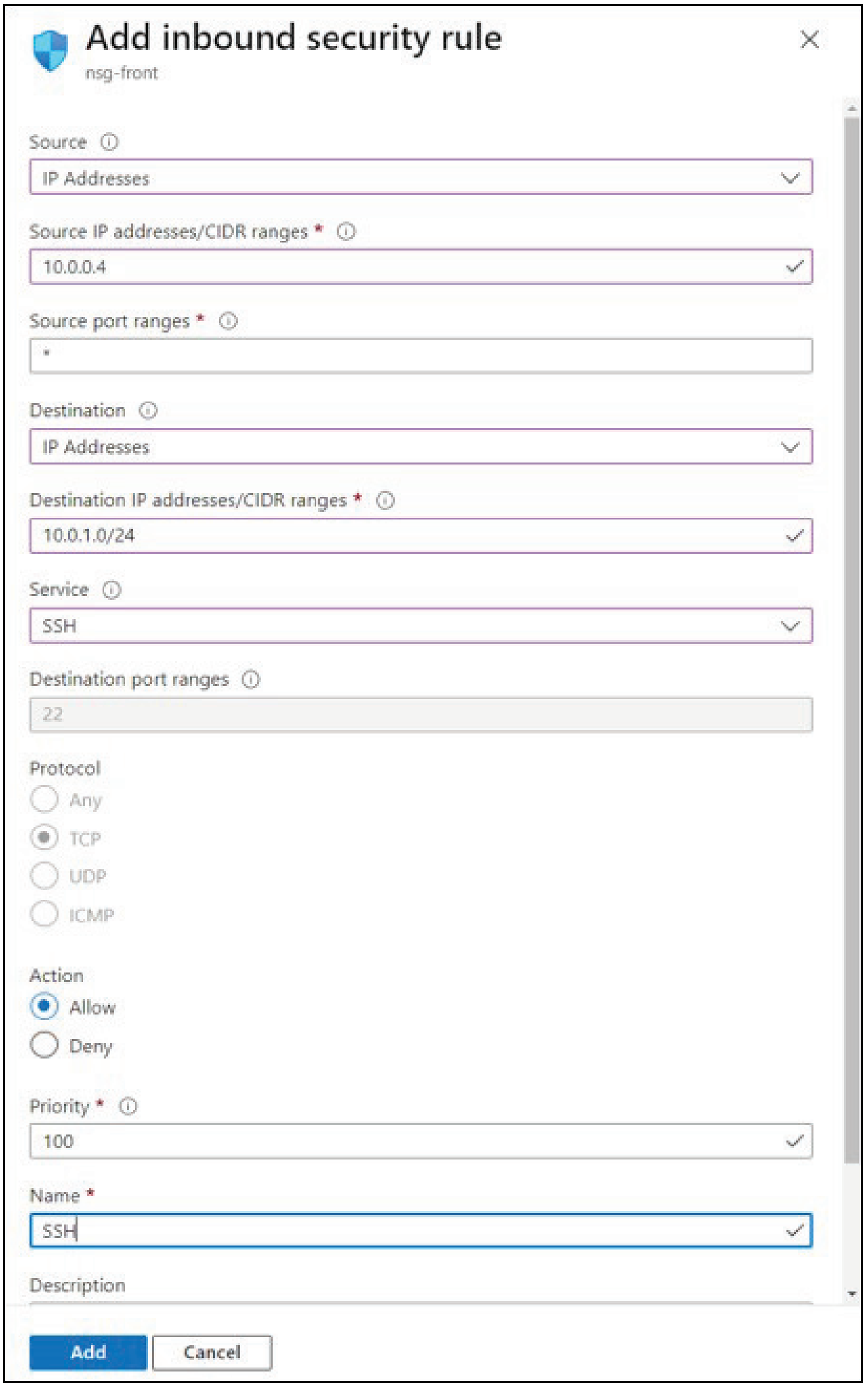
Filter by name

Port == **all** | Protocol == **all** | Source =

Priority	Name	Port
65000	AllowVnetInBound	Any
65001	AllowAzureLoadBalan...	Any

Figure 2-28: *Add a New Inbound Security Rule to NSG.*

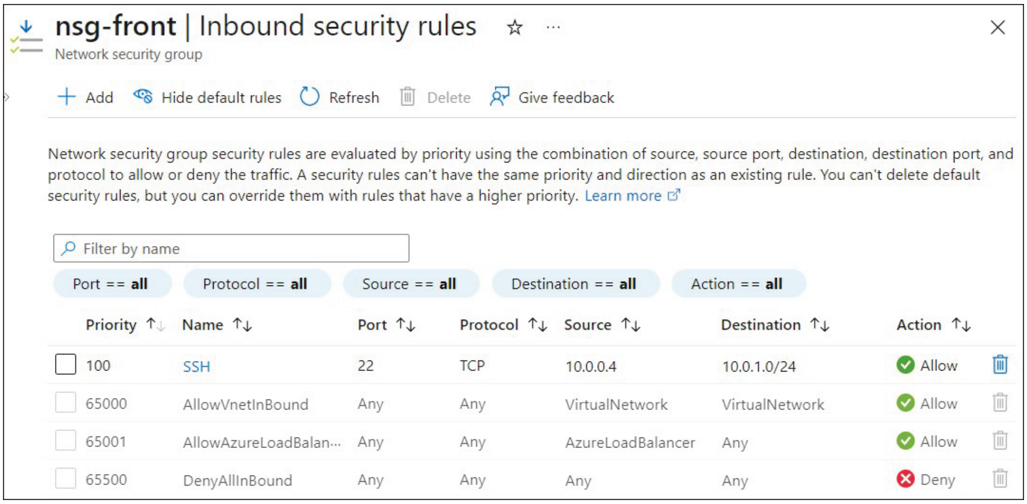
Select the IP address from the *Source* drop-down menu and fill in the IP address of the source VM. The Asterix sign (*) in the *Source port ranges* field means any port. Select the IP address from the *Destination* drop-down menu but instead of defining the host address, use the subnet range. You can use the *Custom* service option and select TCP port 22 for allowing SSH. We have chosen a pre-defined SSH service from the *Service* drop-down list. Choose the *Allow* action. NSG Inbound and Outbound security rules are processed in the priority order, from low to high. The allowed priority values are 100-4096. Give the name to your NSG and add a short description if you like. Click the *Add* button to deploy a new security rule.


Add inbound security rule
nsg-front

Source: IP Addresses
Source IP addresses/CIDR ranges *: 10.0.0.4
Source port ranges *: *
Destination: IP Addresses
Destination IP addresses/CIDR ranges *: 10.0.1.0/24
Service: SSH
Destination port ranges: 22
Protocol: Any [✓ TCP] UDP ICMP
Action: [✓ Allow] Deny
Priority *: 100
Name *: SSH
Description

Figure 2-29: *Add a New Inbound Security Rule to NSG.*

Figure 2-29 shows that we have successfully added a new Inbound security rule to our example NSG, nsg-front.



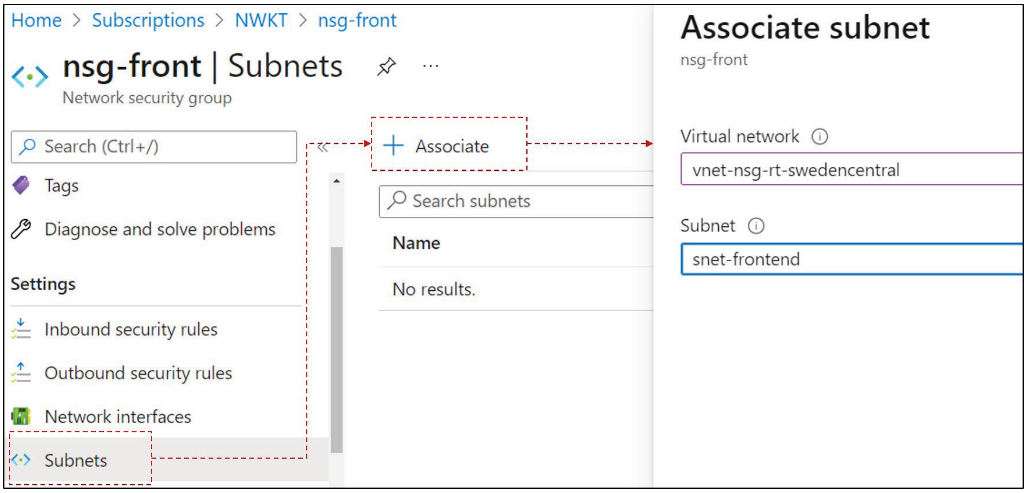
Network security group security rules are evaluated by priority using the combination of source, source port, destination, destination port, and protocol to allow or deny the traffic. A security rules can't have the same priority and direction as an existing rule. You can't delete default security rules, but you can override them with rules that have a higher priority. Learn more

Priority	Name	Port	Protocol	Source	Destination	Action
100	SSH	22	TCP	10.0.0.4	10.0.1.0/24	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalan...	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Figure 2-30: *NSG to Subnet Association.*

Step-3: Associate the NSG to Subnet

Navigate to the nsg-front main view and select the *Subnet* option under the *Settings* section. Click the *+Associate* button. Select your VNet and subnet in the *Associate* window. Then click the *Ok* button.

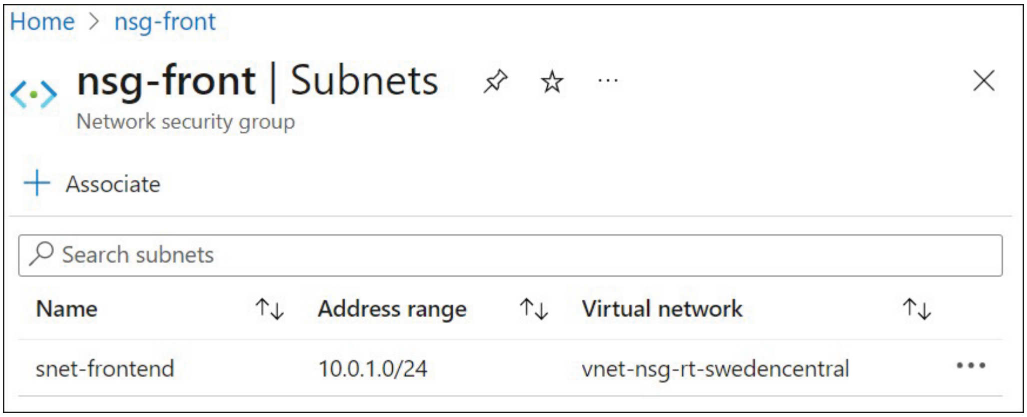


Virtual network: vnet-nsg-rt-swedencentral

Subnet: snet-frontend

Figure 2-31: *NSG to Subnet Association.*

Figures 2-32 and 2-33 verify that the NSG is associated with the subnet 10.0.1.0/24 but not with any NIC.


Home > nsg-front

nsg-front | Subnets

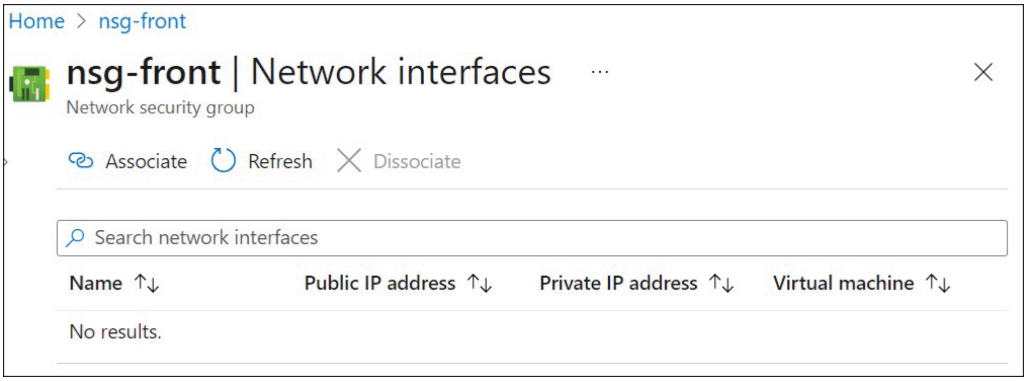
Network security group

+ Associate

Search subnets

Name	Address range	Virtual network
snet-frontend	10.0.1.0/24	vnet-nsg-rt-swedencentral

Figure 2-32: *NSG to Subnet Association.*


Home > nsg-front

nsg-front | Network interfaces

Network security group

Associate Refresh Dissociate

Search network interfaces

Name	Public IP address	Private IP address	Virtual machine
No results.			

Figure 2-33: *NSG to NIC Association.*

Because we want to test the SSH connection from vm-bastion to vm-front-1, we need to copy the vm-front-1's private key to vm-Bastion. Example 2-5 shows how you can do it by using SCP. In our example, we copy the *vm-Front-1_key.pem* file (private key) into vm-front-1's directory /home/<user name>. You also need to use the private key of vm-Bastion to copy the file. This is not a recommended solution. In reality, you should never store a pem file in the VM that uses it for authentication.

```
C:\01-Toni\Own\01-Azure\Chapter_2-VM-SG-ACL\Bastion-PEM>scp -i AzureNWKT.pem
vm-Front-1_key.pem nwktAdmin@20.91.188.31:/home/nwktAdmin
vm-Front-1_key.pem
```

Example 2-5: *Private Key Copy to vm-Bastion.*

Example 2-6 shows an error notification when the user rights of vm-Front-1_key.pem file are too open.

```
nwktAdmin@vm-Bastion:~$ ssh -i vm-Front-1_key.pem nwktAdmin@10.0.1.4
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0664 for 'vm-Front-1_key.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "vm-Front-1_key.pem": bad permissions
nwktAdmin@10.0.1.4: Permission denied (publickey).
nwktAdmin@vm-Bastion:~$
```

Example 2-6: *Error Message when File Permission is too Open.*

Example 2-7 illustrates how to set the permission as read-only for a user and remove all other permissions using *the chmod 400 <fie name>* command. After the change, you can successfully login to from vm-Bastion to vm-Front-1.

```
nwktAdmin@vm-Bastion:~$ chmod 400 vm-Front-1_key.pem

nwktAdmin@vm-Bastion:~$ ssh -i vm-Front-1_key.pem nwktAdmin@10.0.1.4
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-1023-azure x86_64)
<snipped >
  System information as of Wed May 11 17:24:24 UTC 2022

  System load:  0.08              Processes:             105
  Usage of /:   4.8% of 28.90GB   Users logged in:       0
  Memory usage: 55%               IPv4 address for eth0: 10.0.1.4
  Swap usage:   0%

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

nwktAdmin@vm-Front-1:~$
```

Example 2-7: *Changing the File Permission in Linux and Login to Remote VM.*

Application Security Group

We can create a logical group of VMs by associating them with the same *Application Security Group (ASG)*. The ASG can then be used as a source or destination in NSG security rules. In our example, we have two ASGs, asg-Back (associated with VMs 10.0.2.4-6) and asg-Back#2 (associated with VMs 10.0.2.7-9). The first ASG (asg-Back) is used as a destination in the security rule on the NSG nsg-Back that allows ICMP from VM vm-Front-1. The second ASG (asg-Back#2) is used as a destination in the security rule on the same NSG nsg-Back that allows ICMP from VM vm-Bastion.

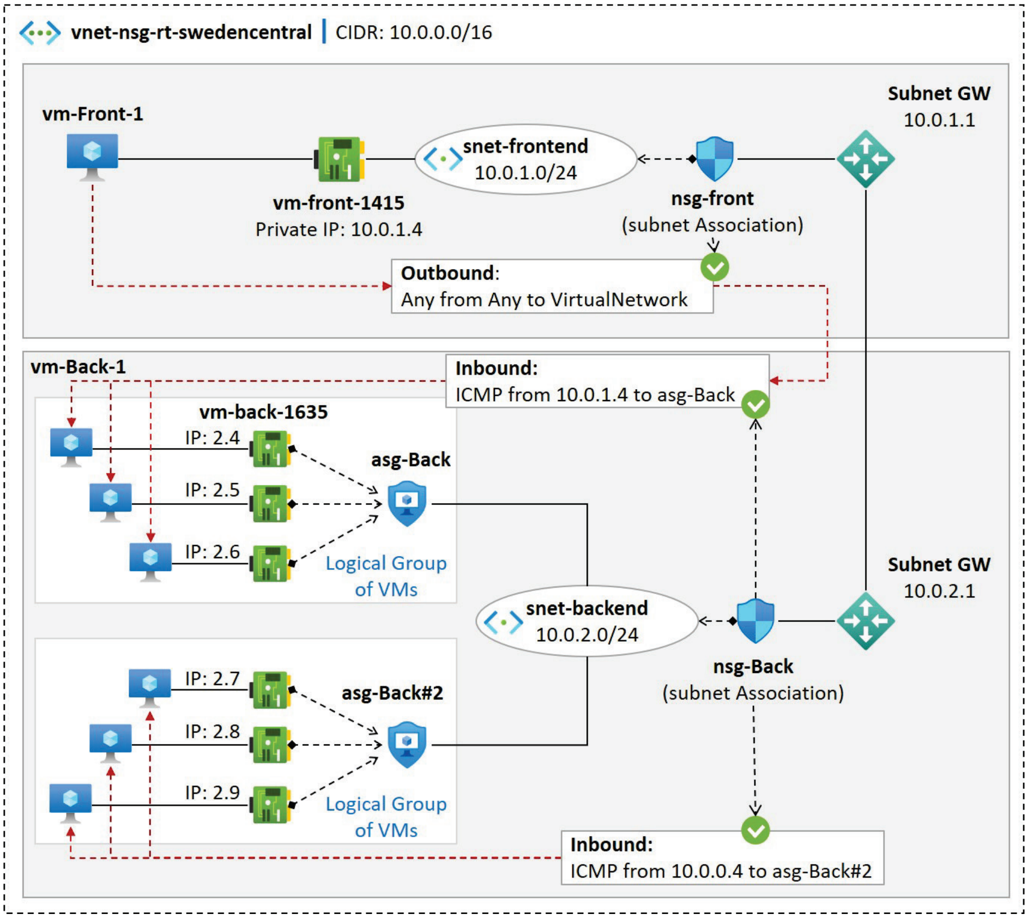


Figure 2-34: *NIC to ASG to NSG Associations.*

Step-1: Create Application Security Group

Select the Create a resource from the Azure portal home view (figure 2-35). Then type the Application security *group* on the search field. Next, select the *Microsoft Application security group* and click the Create button. During the process, we attach the ASG to the resource group under your subscription and define the location just as we did with other Azure resources (figure 2-36).

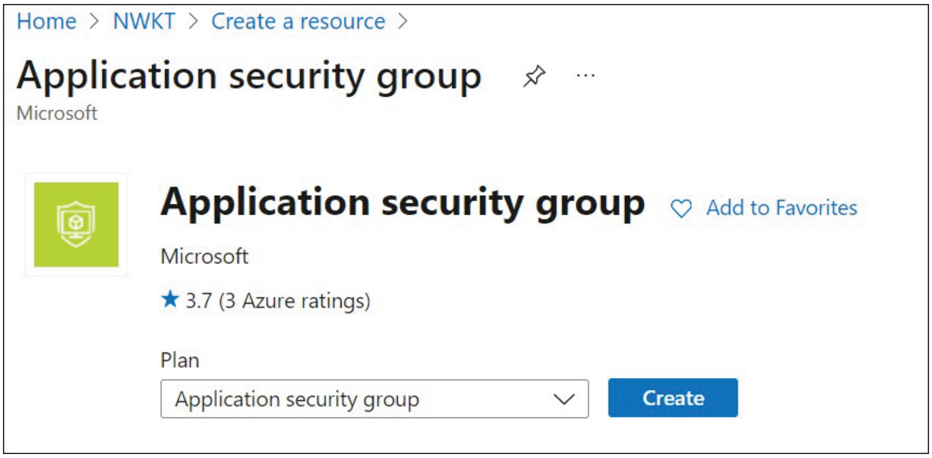
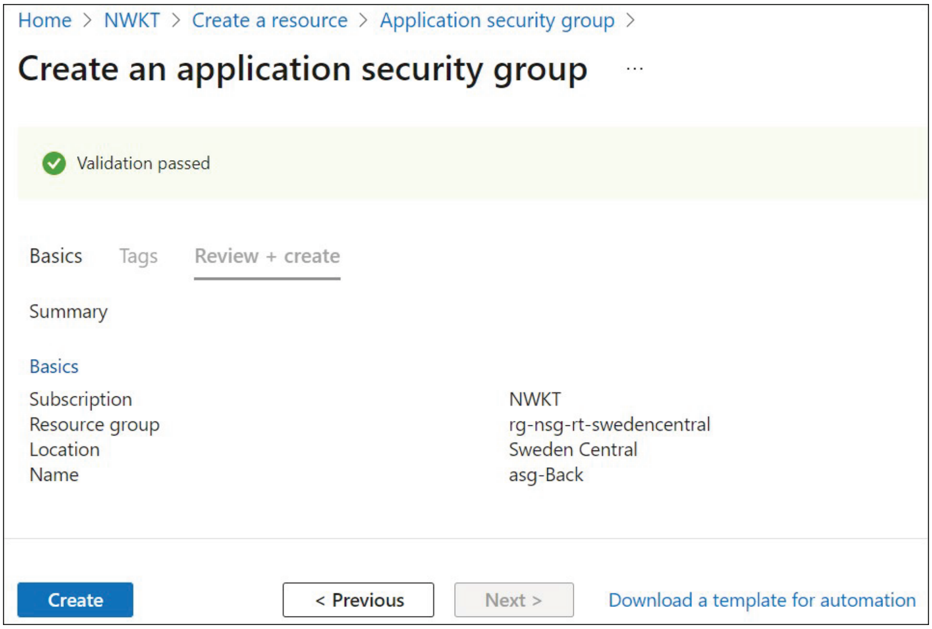

Home > NWKT > Create a resource >
Application security group
Microsoft
Application security group
Add to Favorites
Microsoft
★ 3.7 (3 Azure ratings)
Plan
Application security group
Create

Figure 2-35: *Create Application Security group.*


Home > NWKT > Create a resource > Application security group >

Create an application security group

Validation passed

Basics Tags Review + create

Summary

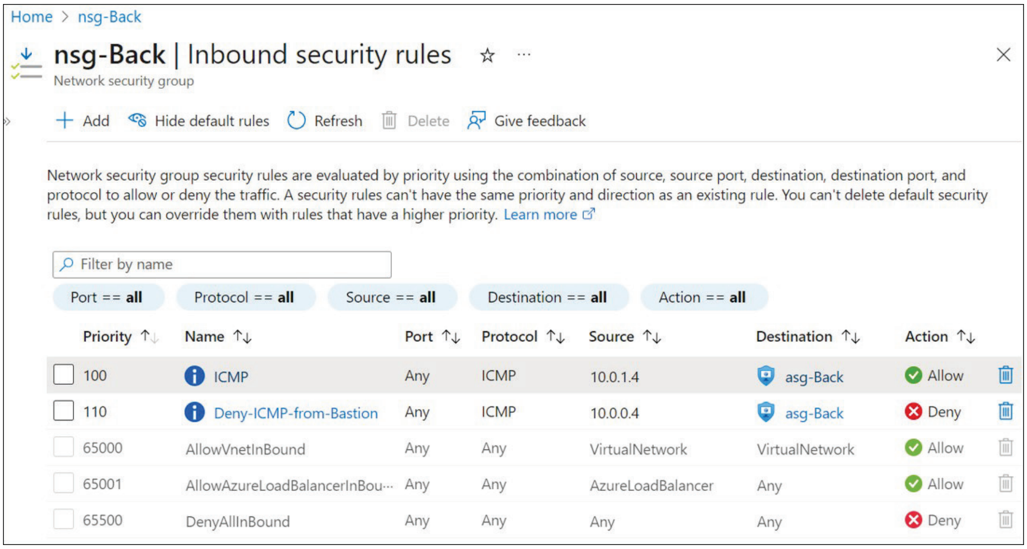
Basics

Subscription	NWKT
Resource group	rg-nsg-rt-swedencentral
Location	Sweden Central
Name	asg-Back

Figure 2-36: *Create Application Security group.*

Step-2: Add a Security Rule into NSG

Figure 2-37 shows the updated inbound security rule in the NSG nsg-Back. There are two new rules. The first rule allows an ICMP from 10.0.1.4 (vm-Front-1) to asg-Back, while the second rule denies the ICMP from 10.0.0.4 (vm-Bastion).



Network security group security rules are evaluated by priority using the combination of source, source port, destination, destination port, and protocol to allow or deny the traffic. A security rules can't have the same priority and direction as an existing rule. You can't delete default security rules, but you can override them with rules that have a higher priority. Learn more

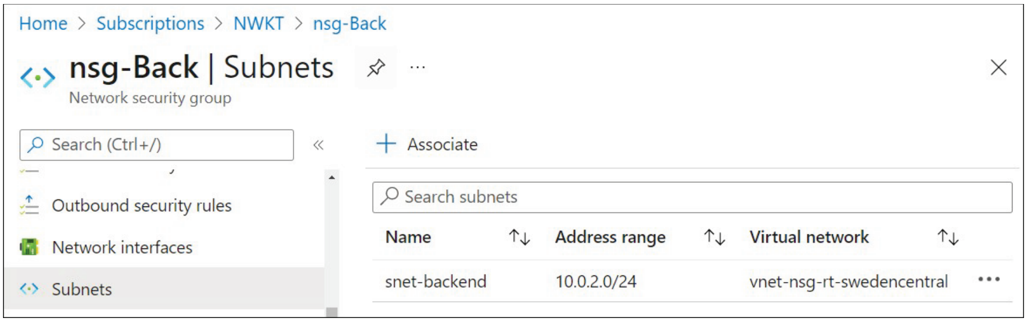
Filter by name

Port == **all** Protocol == **all** Source == **all** Destination == **all** Action == **all**

Priority	Name	Port	Protocol	Source	Destination	Action
100	ICMP	Any	ICMP	10.0.1.4	asg-Back	Allow
110	Deny-ICMP-from-Bastion	Any	ICMP	10.0.0.4	asg-Back	Deny
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBou···	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Figure 2-37: *Updated Inbound Security Rule in NSG.*

Figure 2-38 verifies that the nsg-Back is associated with the subnet 10.0.0.0/24 (snet-Back).

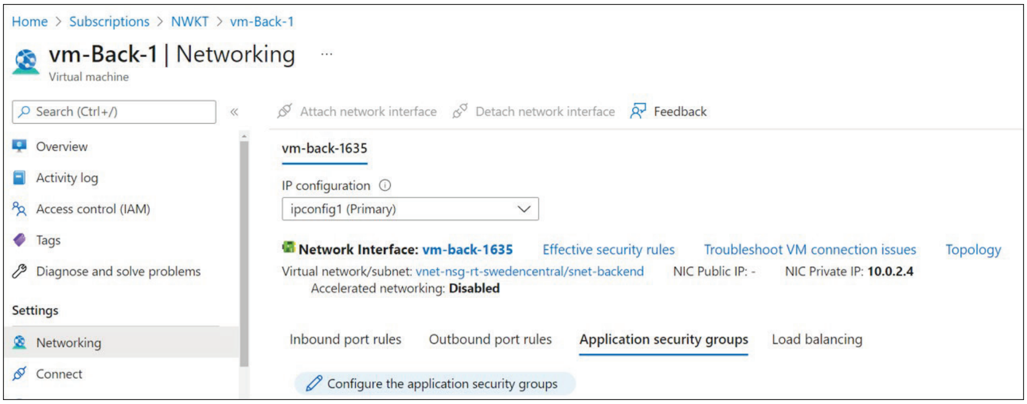


Name	Address range	Virtual network
snet-backend	10.0.2.0/24	vnet-nsg-rt-swedencentral

Figure 2-38: *NSG Subnet Association.*

Step-3: Associate VM's NIC with ASG

Select your VM and choose the *Networking* option under the *Setting* section. Open the *Application security rule* tab.


Home > Subscriptions > NWKT > vm-Back-1

vm-Back-1 | Networking
Virtual machine

vm-back-1635

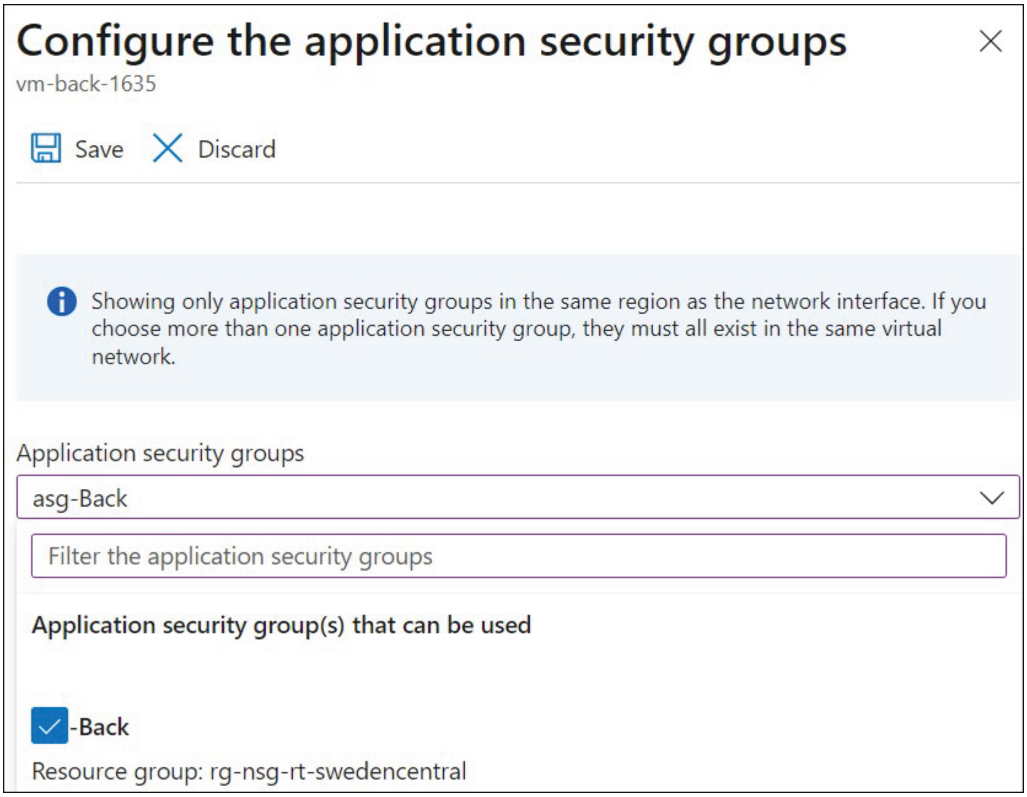
IP configuration: ipconfig1 (Primary)

Network Interface: vm-back-1635 Effective security rules Troubleshoot VM connection issues Topology
Virtual network/subnet: vnet-nsg-rt-swedencentral/snet-backend NIC Public IP: - NIC Private IP: 10.0.2.4
Accelerated networking: Disabled

Inbound port rules
Outbound port rules
Application security groups
Load balancing
Configure the application security groups

Figure 2-39: *VM's NIC Association with an NSG.*

Select your ASG from the drop-down menu and click the Save button.


Configure the application security groups
vm-back-1635

Showing only application security groups in the same region as the network interface. If you choose more than one application security group, they must all exist in the same virtual network.

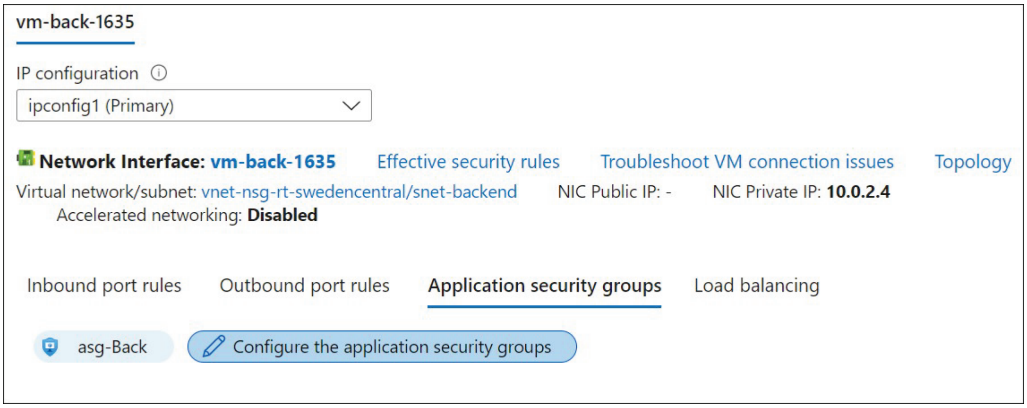
Application security groups: asg-Back

Application security group(s) that can be used

✓ -Back
Resource group: rg-nsg-rt-swedencentral

Figure 2-40: *VM's NIC Association with an NSG.*

Figure 2-41 verifies that the ASG asg-Back is now associated with the NIC.



vm-back-1635

IP configuration

ipconfig1 (Primary)

Network Interface: vm-back-1635 Effective security rules Troubleshoot VM connection issues Topology

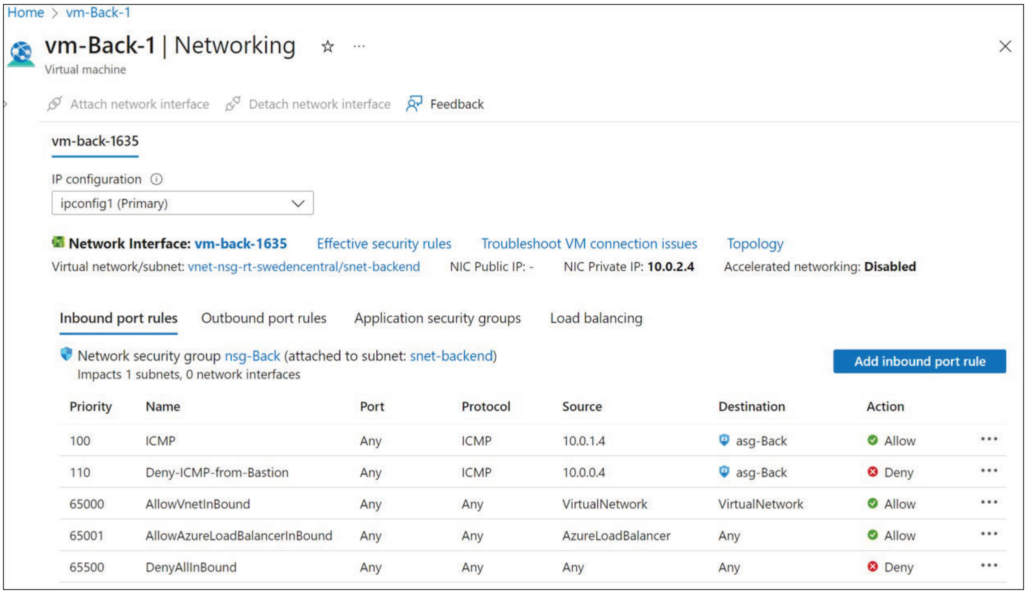
Virtual network/subnet: vnet-nsg-rt-swedencentral/snet-backend NIC Public IP: - NIC Private IP: **10.0.2.4**
Accelerated networking: **Disabled**

Inbound port rules Outbound port rules **Application security groups** Load balancing

asg-Back Configure the application security groups

Figure 2-41: *VM's NIC Association with an NSG.*

After NIC to ASG association, we can see security rules of NSG nsg-BACK.



Home > vm-Back-1

vm-Back-1 | Networking

Virtual machine

Attach network interface Detach network interface Feedback

vm-back-1635

IP configuration

ipconfig1 (Primary)

Network Interface: vm-back-1635 Effective security rules Troubleshoot VM connection issues Topology

Virtual network/subnet: vnet-nsg-rt-swedencentral/snet-backend NIC Public IP: - NIC Private IP: **10.0.2.4** Accelerated networking: **Disabled**

Inbound port rules Outbound port rules Application security groups Load balancing

Network security group nsg-Back (attached to subnet: snet-backend)
Impacts 1 subnets, 0 network interfaces

Add inbound port rule

Priority	Name	Port	Protocol	Source	Destination	Action
100	ICMP	Any	ICMP	10.0.1.4	asg-Back	Allow
110	Deny-ICMP-from-Bastion	Any	ICMP	10.0.0.4	asg-Back	Deny
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Figure 2-42: *VM's NIC Association with an NSG.*

Step-4: Test Connection

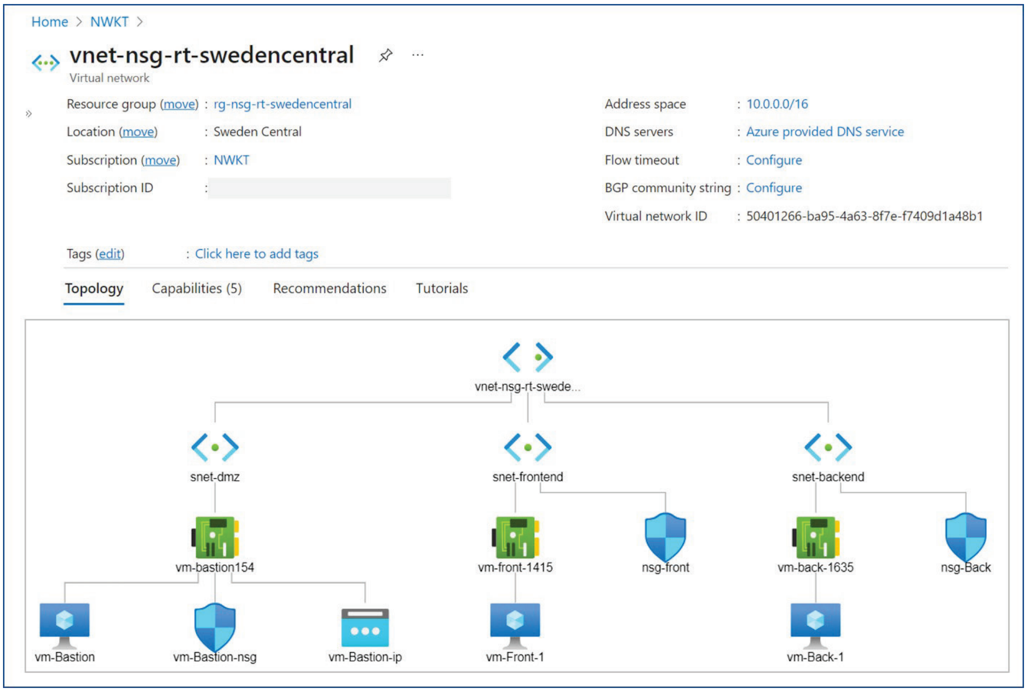
Example 2-8 shows that ping from vm-Bastion to 10.0.2.4 (vm-Back) doesn't work, while the ping from vm-Front-1 works.

```
nwktAdmin@vm-Bastion:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
^C
--- 10.0.2.4 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7149ms

nwktAdmin@vm-Front-1:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=1.54 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=1.04 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.912 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=1.23 ms
^C
--- 10.0.2.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3011ms
rtt min/avg/max/mdev = 0.912/1.181/1.542/0.236 ms
```

Example 2-8: *Ping Test from vm-Bastion and vm-Front-1 to vm-Back.*

Resources View

Figure 2-43 illustrates the Azure-generated overview of our example deployment. On the top of the figure, there is Virtual Network vnet-nsg-rt-swedencentral, where we have subnets snet-dmz, snet-frontend, and snet-backend. The Virtual Machine vm-Bastion is connected to snet-dmz with the Network Interface vm-bastion154. The Network Security Group vm-Bastion-nsg and the Public IP address vm-Bastion-ip are associated with the NIC vm-bastion154. Virtual Machines vm-Front-1 and vm-Back-1 don't have an NSG associated with their NIC. However, they are secured with the NSG attached to their respective subnets. Note that the topology view doesn't show the Application Security Rules.


Home > NWKT >

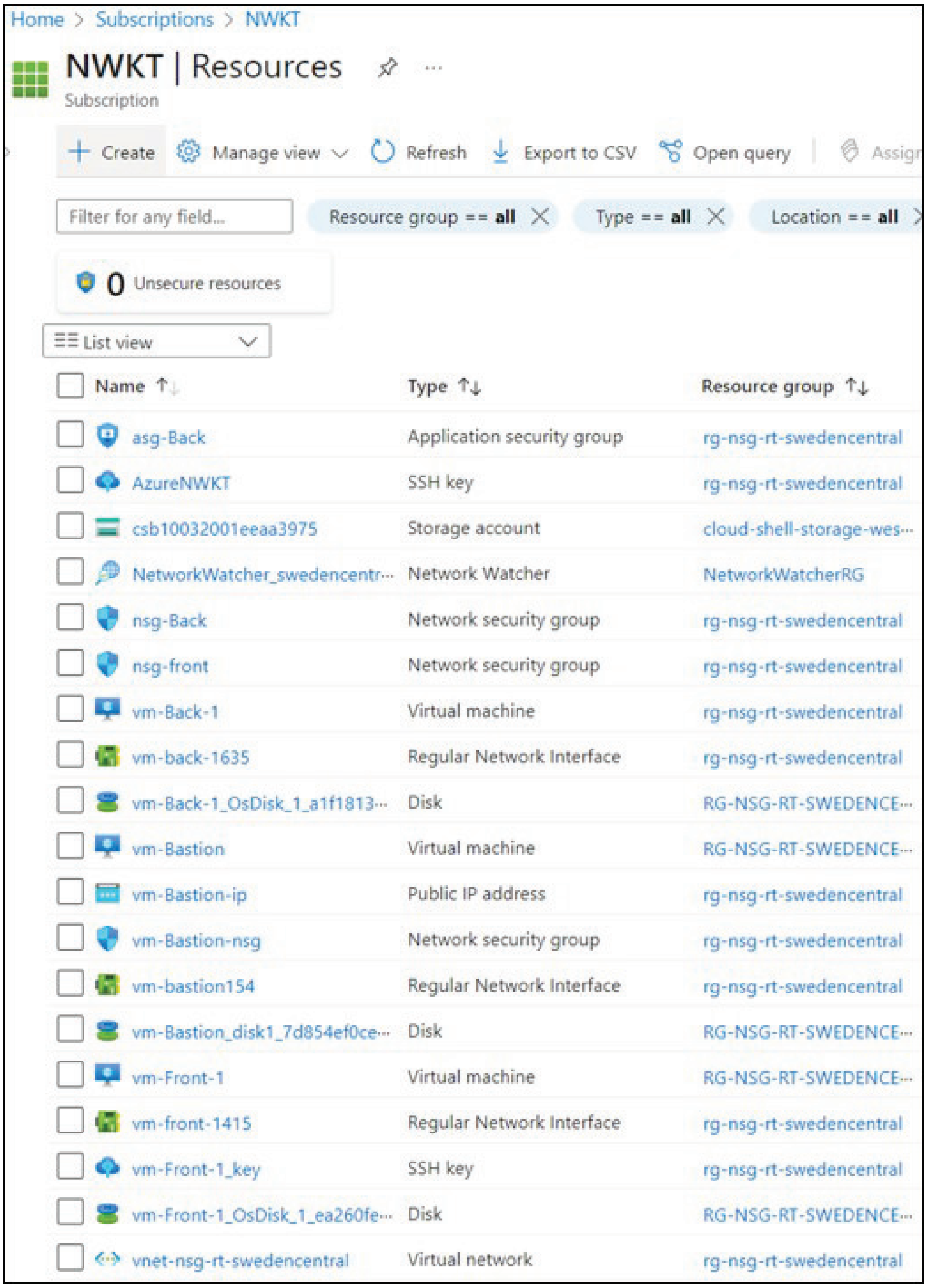
vnet-nsg-rt-swedencentral
Virtual network

Resource group (move)	: rg-nsg-rt-swedencentral		Address space	: 10.0.0.0/16
Location (move)	: Sweden Central		DNS servers	: Azure provided DNS service
Subscription (move)	: NWKT		Flow timeout	: Configure
Subscription ID	:		BGP community string	: Configure
			Virtual network ID	: 50401266-ba95-4a63-8f7e-f7409d1a48b1

Tags (edit) : Click here to add tags

Figure 2-43: *Azure Generated Topology View.*

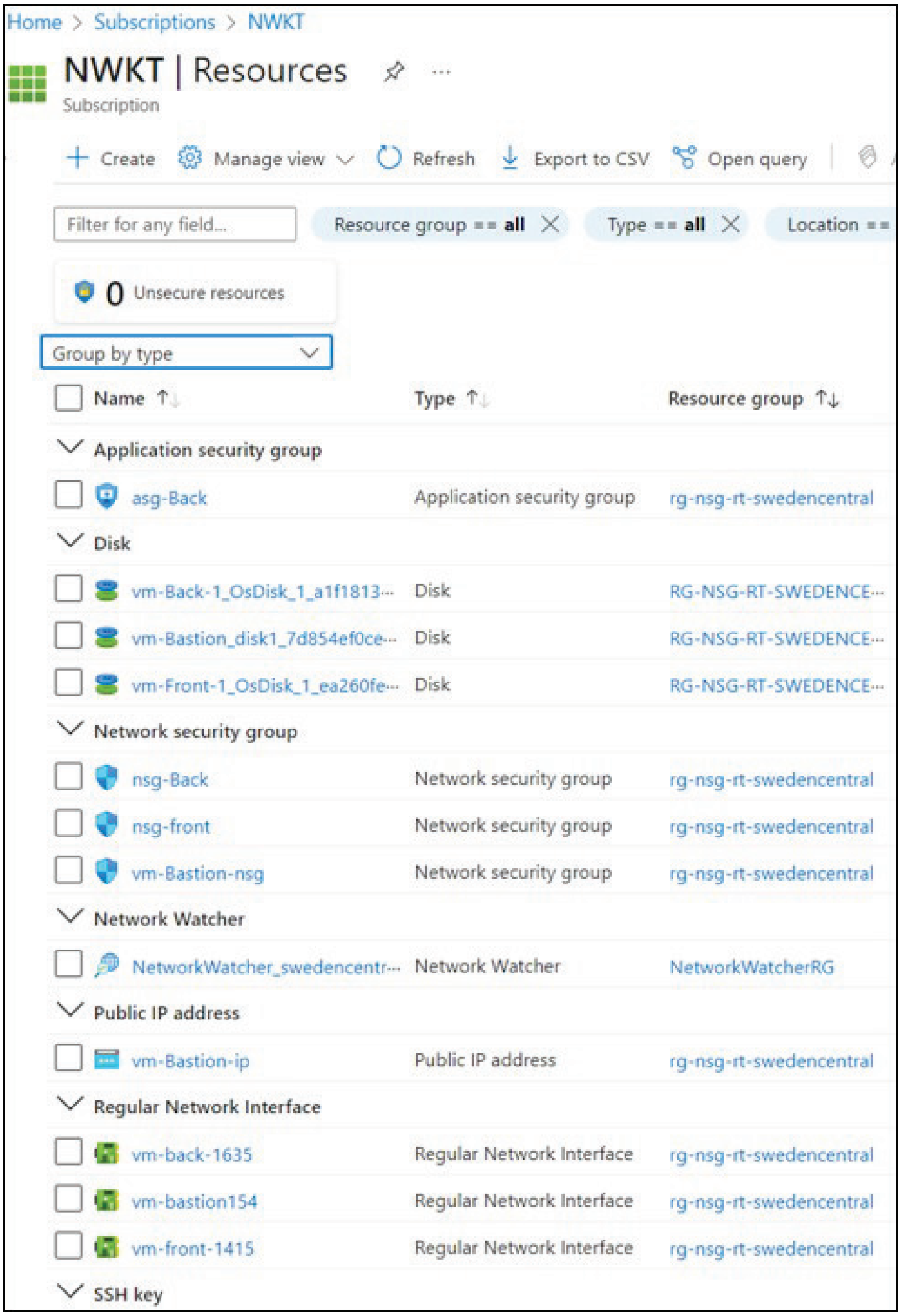
Figure 2-44 shows all resources we have created in this chapter. You can modify the list view in several ways. You can arrange them based on the Name, Type, Resource group, Location, or Subscription.



Name	Type	Resource group
asg-Back	Application security group	rg-nsg-rt-swedencentral
AzureNWKT	SSH key	rg-nsg-rt-swedencentral
csb10032001eeaa3975	Storage account	cloud-shell-storage-wes···
NetworkWatcher_swedencentr···	Network Watcher	NetworkWatcherRG
nsg-Back	Network security group	rg-nsg-rt-swedencentral
nsg-front	Network security group	rg-nsg-rt-swedencentral
vm-Back-1	Virtual machine	rg-nsg-rt-swedencentral
vm-back-1635	Regular Network Interface	rg-nsg-rt-swedencentral
vm-Back-1_OsDisk_1_a1f1813···	Disk	RG-NSG-RT-SWEDENCE···
vm-Bastion	Virtual machine	RG-NSG-RT-SWEDENCE···
vm-Bastion-ip	Public IP address	rg-nsg-rt-swedencentral
vm-Bastion-nsg	Network security group	rg-nsg-rt-swedencentral
vm-bastion154	Regular Network Interface	rg-nsg-rt-swedencentral
vm-Bastion_disk1_7d854ef0ce···	Disk	RG-NSG-RT-SWEDENCE···
vm-Front-1	Virtual machine	RG-NSG-RT-SWEDENCE···
vm-front-1415	Regular Network Interface	rg-nsg-rt-swedencentral
vm-Front-1_key	SSH key	rg-nsg-rt-swedencentral
vm-Front-1_OsDisk_1_ea260fe···	Disk	RG-NSG-RT-SWEDENCE···
vnet-nsg-rt-swedencentral	Virtual network	rg-nsg-rt-swedencentral

Figure 2-44: *Unfiltered Resource View.*

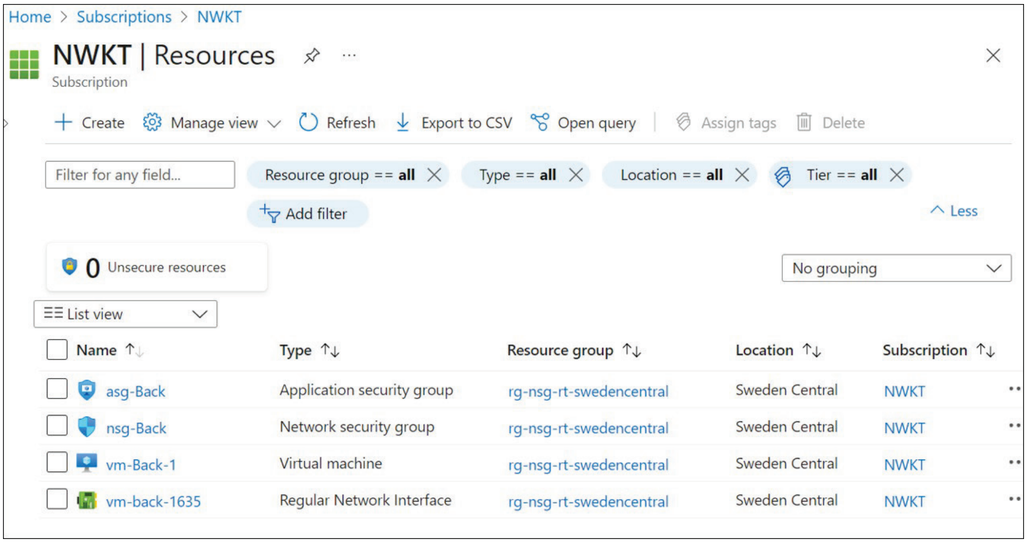
You can also group resource as an example based on their resource type (figure 2-45).



Name	Type	Resource group
Application security group		
asg-Back	Application security group	rg-nsg-rt-swedencentral
Disk		
vm-Back-1_OsDisk_1_a1f1813···	Disk	RG-NSG-RT-SWEDENCE···
vm-Bastion_disk1_7d854ef0ce···	Disk	RG-NSG-RT-SWEDENCE···
vm-Front-1_OsDisk_1_ea260fe···	Disk	RG-NSG-RT-SWEDENCE···
Network security group		
nsg-Back	Network security group	rg-nsg-rt-swedencentral
nsg-front	Network security group	rg-nsg-rt-swedencentral
vm-Bastion-nsg	Network security group	rg-nsg-rt-swedencentral
Network Watcher		
NetworkWatcher_swedencentr···	Network Watcher	NetworkWatcherRG
Public IP address		
vm-Bastion-ip	Public IP address	rg-nsg-rt-swedencentral
Regular Network Interface		
vm-back-1635	Regular Network Interface	rg-nsg-rt-swedencentral
vm-bastion154	Regular Network Interface	rg-nsg-rt-swedencentral
vm-front-1415	Regular Network Interface	rg-nsg-rt-swedencentral
SSH key		

Figure 2-45: *Resources Grouped Based on the Type.*

Tags are also useful for filtering resources. In figure 2-45, I have attached the tag Tier to all resources related to the backend application.

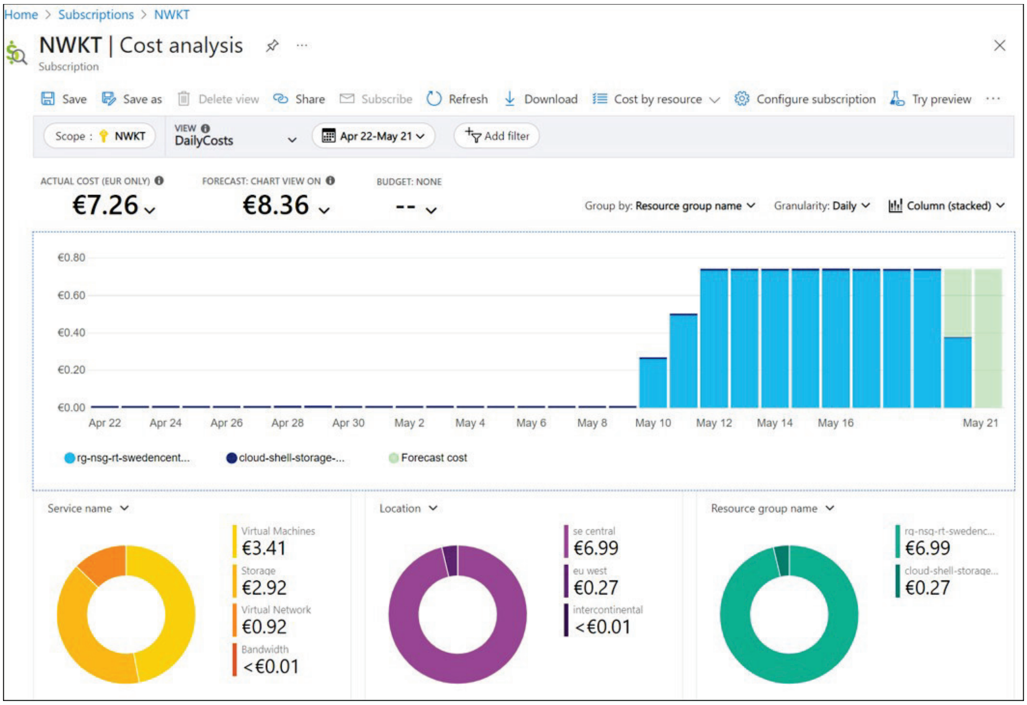


Name	Type	Resource group	Location	Subscription
asg-Back	Application security group	rg-nsg-rt-swedencentral	Sweden Central	NWKT
nsg-Back	Network security group	rg-nsg-rt-swedencentral	Sweden Central	NWKT
vm-Back-1	Virtual machine	rg-nsg-rt-swedencentral	Sweden Central	NWKT
vm-back-1635	Regular Network Interface	rg-nsg-rt-swedencentral	Sweden Central	NWKT

Figure 2-45: *Resource View Filtered with Tags.*

Pricing

This is just an informative section for those readers who want to test NSGs in Azure. The figure below shows the cost of our example environment is only been 7.26 euros in the last 30 days.



ACTUAL COST (EUR ONLY)	FORECAST: CHART VIEW ON	BUDGET: NONE
€7.26	€8.36	--

● rg-nsg-rt-swedencent... ● cloud-shell-storage-... ● Forecast cost

Service name		Location		Resource group name	
Virtual Machines	€3.41	se central	€6.99	rg-nsg-rt-swedenc...	€6.99
Storage	€2.92	eu west	€0.27	cloud-shell-storage...	€0.27
Virtual Network	€0.92	intercontinental	<€0.01		
Bandwidth	<€0.01				

Figure 2-46: *Pricing.*

References

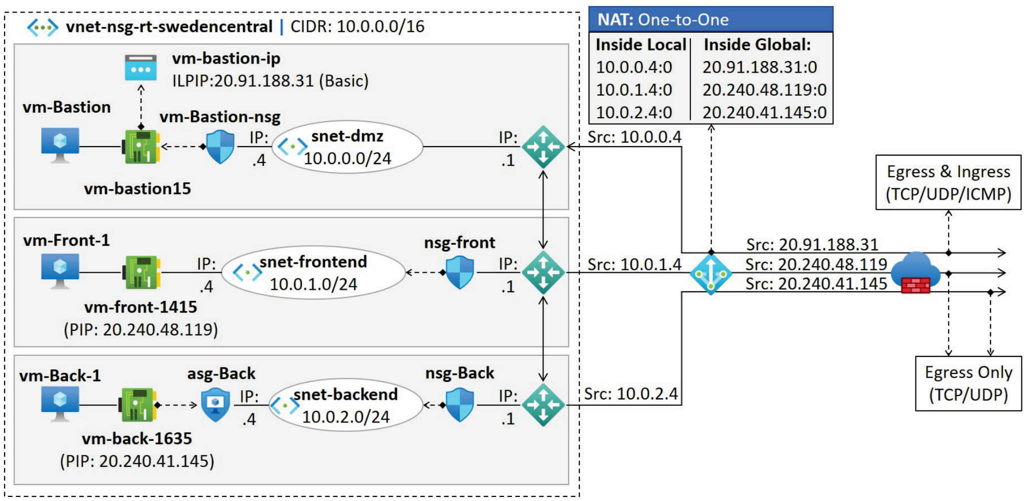
[1] Network security groups:
https://docs.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview

[2] Azure pricing
https://azure.microsoft.com/en-us/pricing/

Chapter 3: Internet Access with VM-Specific Public IP

Introduction

In chapter two, we created a VM vm-Bastion and associated a Public IP address to its attached NIC vm-bastion154. The Public IP addresses associated with VM's NIC are called *Instance Level Public IP (ILPIP)*. Then we added a security rule to the existing NSG vm-Bastion-nsg, which allows an inbound SSH connection from the external host. Besides, we created VMs vm-front-1 and vm-Back-1 without public IP address association. However, these two VMs have an egress Internet connection because Azure assigns *Outbound Access IP (OPIP)* addresses for VMs which we haven't allocated an ILPIP (vm-Front-1: 20.240.48.199 and vm-Back-1- 20.240.41.145). The Azure portal does not list these IP addresses in the Azure portal VM view. Note that neither user-defined nor Azure-allocated Public IP addresses are not configured as NIC addresses. Instead, Azure adds them as a One-to-One entry to the NAT table (chapter 15 introduces a NAT service in detail). Figure 3-1 shows how the source IP address of vm-Bastion is changed from 10.0.1.4 to 20.91.188.31 when traffic is forwarded to the Internet. The source IP address of the Internet traffic from vm-Front-1 and vm-Back-1 will also be translated in the same way. The traffic policy varies based on the IP address assignment mechanism. The main difference is that external hosts can initiate connection only with VMs with an ILPIP. Besides, these VMs are allowed to use TCP/UDP/ICMP, while VMs with the Azure assigned public IP address can only use TCP or UDP but not ICMP.



Inside Local	Inside Global:
10.0.0.4:0	20.91.188.31:0
10.0.1.4:0	20.240.48.119:0
10.0.2.4:0	20.240.41.145:0

Figure 3-1: *Overview of the Azure Internet Access.*

Public IP Address for the Internet Access

The Public IP address allows Azure resources to communicate with hosts located on the Internet. For example, when we assign a Public IP address to VM's NIC, we allow hosts from the Internet to initiate connections with VM. This naturally requires a security rule in NSG protecting VM, which we permit inbound connection. If we don't attach a Public IP address to VM, Azure will allocate a Public IP address that allows only Outbound traffic to the Internet. Figure 3-2 shows the example resources to which we can assign a Public IP address. Neither ILPIP addresses nor Azure-assigned public IP addresses are routable within your VNet (ILPIP and OPIP associated with VM NAT Rule on NAT Gateway). Each VMs has an implicit, default outbound access to the Internet because Azure assigns a default *Outbound Access IP address (OPIP)*. The OPIP addresses are not owned by customers and they may change. Microsoft recommends disabling the default outbound access (assign ILPIP, use NAT GW or LB).

Public IP Allocation Method

We can assign a Public IP address to Azure resources using ***the Static*** or ***Dynamic*** method. The static Public IP address is bound to resource during the resource creation process. The statically allocated public IP address doesn't change when we restart VM. It is only released when we delete its associated resource. Dynamically allocated Public IP addresses are released when we stop or restart the associated VM. It means that the IP address may be allocated to another resource and can't be reassigned to our original resource. Figure 3-2 above shows the supported allocation method for IPv4 and IPv6 addresses.

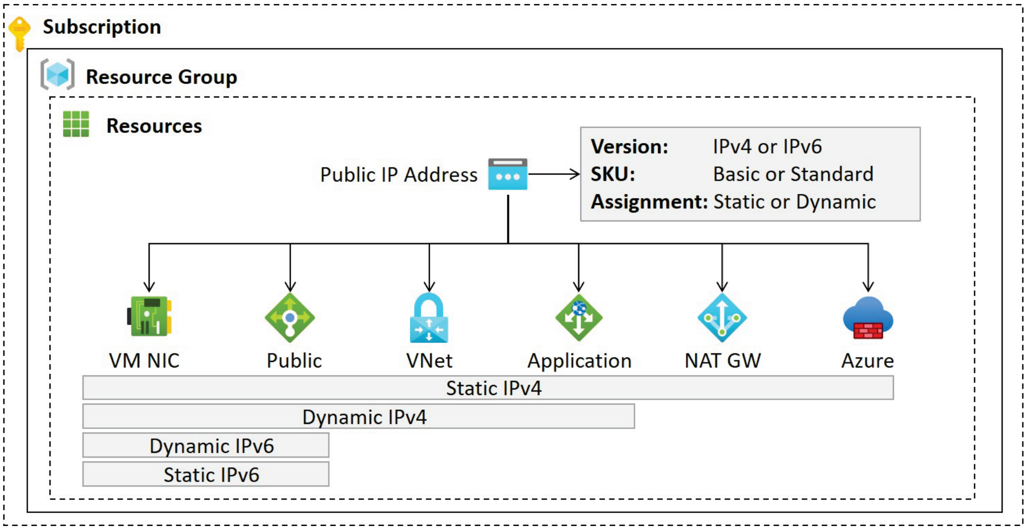


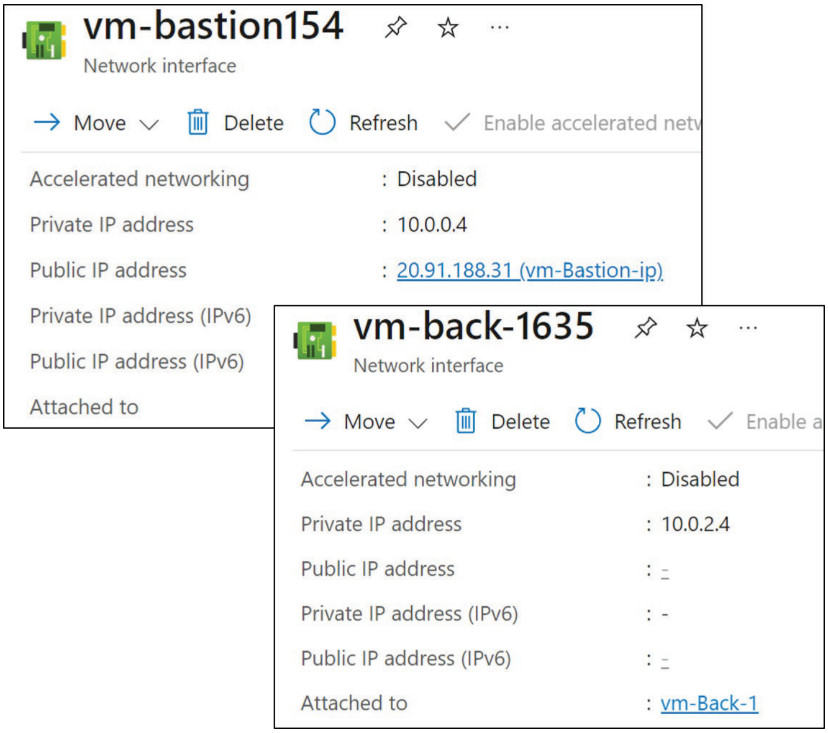
Figure 3-2: *IP Allocation Methods.*

Stock-Keeping Unit (SKU)

The Stock-Keeping Unit (SKU) has two options, ***Standard*** and ***Basic***. The Standard SKU allows only Static assignment, while the Basic SKU also supports a dynamic allocation method. We can set the default TCP or HTTP connection idle timeout between 4 - 30 minutes (the default idle timeout is four minutes) for a connection established by VM. The flow-idle timeout for connections initiated by an external source is fixed at four minutes. From the security perspective, the Standard SKU is closed by default (requires "allow" security rule in NSG), while the Basic SKU is open (optional NSG). Only Standard IP support zone redundancy (requires a region with three zones), routing preference, and cross-region Load Balancers.

Public IP Verification

Figure 3-3 shows that the Azure Portal lists only the user-defined ILPIP in the NIC-specific overview window. It doesn't list the Azure-allocated PIP on this view. We can resolve these egress-only Public IP addresses from the VM's CLI. Examples 3-1, 3-2, and 3-3 show how we can do it by using the *nslookup* command, where we request our IP information from the external DNS server.



vm-bastion154
Network interface

Accelerated networking	: Disabled
Private IP address	: 10.0.0.4
Public IP address	: 20.91.188.31 (vm-Bastion-ip)
Private IP address (IPv6)	
Public IP address (IPv6)	
Attached to	

vm-back-1635
Network interface

Accelerated networking	: Disabled
Private IP address	: 10.0.2.4
Public IP address	: -
Private IP address (IPv6)	: -
Public IP address (IPv6)	: -
Attached to	: vm-Back-1

Figure 3-3: *Network Interface IP Addressing.*

```
nwktAdmin@vm-Bastion:~$ nslookup myip.opendns.com resolver1.opendns.com
Server:         resolver1.opendns.com
Address:        208.67.222.222#53

Non-authoritative answer:
Name:   myip.opendns.com
Address: 20.91.188.31
```

Example 3-1: *Public IP Address Verification: vm-Bastion.*

```
nwktAdmin@vm-Front-1:~$ nslookup myip.opendns.com resolver1.opendns.com
Server:         resolver1.opendns.com
Address:        208.67.222.222#53

Non-authoritative answer:
Name:   myip.opendns.com
Address: 20.240.48.119
```

Example 3-2: *Public IP Address Verification: vm-Front-1.*

```
nwktAdmin@vm-Back-1:~$ nslookup myip.opendns.com resolver1.opendns.com
Server:         resolver1.opendns.com
Address:        208.67.222.222#53

Non-authoritative answer:
Name:   myip.opendns.com
Address: 20.240.41.145
```

Example 3-3: *Public IP Address Verification: vm-Back-1.*

Internet Outbound Traffic Testing

The user assigned ILPIPs support inbound and outbound connections using TCP, UDP, and ICMP, while the Azure associated PIPs support only TCP or UDP outbound connections. Example 3-4 verifies that both ICMP and TCP outbound connection work from the VM vm-Bastion.

```
nwktAdmin@vm-Bastion:~$ ping -c 3 nwktimes.blogspot.com
PING blogspot.l.googleusercontent.com (142.250.74.65) 56(84) bytes of data.
64 bytes from arn09s23-in-f1.1e100.net (142.250.74.65): icmp_seq=1 ttl=53 time=5.23 ms
64 bytes from arn09s23-in-f1.1e100.net (142.250.74.65): icmp_seq=2 ttl=53 time=5.33 ms
64 bytes from arn09s23-in-f1.1e100.net (142.250.74.65): icmp_seq=3 ttl=53 time=5.62 ms

--- blogspot.l.googleusercontent.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 5.227/5.393/5.623/0.167 ms

nwktAdmin@vm-Bastion:~$ nmap -p80 nwktimes.blogspot.com
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-26 08:19 UTC
Nmap scan report for nwktimes.blogspot.com (142.250.74.65)
Host is up (0.0047s latency).
Other addresses for nwktimes.blogspot.com (not scanned): 2a00:1450:400f:805::2001
rDNS record for 142.250.74.65: arn09s23-in-f1.1e100.net

PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
```

Example 3-4: *Testing Outbound Internet Connection (ICMP/HTTP) from vm-Bastion.*

Examples 3-5 and 3-6 verifies that only TCP outbound connections work from the vm-Front-1 and vm-Back-1.

```
nwktAdmin@vm-Front-1:~$ ping -c 3 nwktimes.blogspot.com
PING blogspot.l.googleusercontent.com (216.58.207.193) 56(84) bytes of data.
^C
--- blogspot.l.googleusercontent.com ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2025ms

nwktAdmin@vm-Front-1:~$ nmap -p80 nwktimes.blogspot.com
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-26 08:21 UTC
Nmap scan report for nwktimes.blogspot.com (216.58.207.193)
Host is up (0.0044s latency).
Other addresses for nwktimes.blogspot.com (not scanned): 2a00:1450:400f:80b::2001
rDNS record for 216.58.207.193: arn11s04-in-f1.1e100.net

PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.34 seconds
nwktAdmin@vm-Front-1:~$
```

Example 3-5: *Testing Outbound Internet Connection (ICMP/HTTP) from vm-Front-1.*

```
nwktAdmin@vm-Back-1:~$ ping -c 3 nwktimes.blogspot.com
PING blogspot.l.googleusercontent.com (216.58.207.225) 56(84) bytes of data.
^C
--- blogspot.l.googleusercontent.com ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2039ms

nwktAdmin@vm-Back-1:~$ nmap -p80 nwktimes.blogspot.com
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-26 08:16 UTC
Nmap scan report for nwktimes.blogspot.com (216.58.207.225)
Host is up (0.0041s latency).
Other addresses for nwktimes.blogspot.com (not scanned): 2a00:1450:400f:801::2001
rDNS record for 216.58.207.225: arn09s19-in-f1.1e100.net

PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.19 seconds
```

Example 3-6: *Testing Outbound Internet Connection (ICMP/HTTP) from vm-Back-1.*

Public IP Addresses for Azure Internal Communication

Virtual Machines communicate with other VMs within and between Virtual networks using private IP addresses. For the Internet connection, VMs use either user-assigned ILPIP or Azure-associated Public IP addresses. However, there are a lot of other connection requirements depending on the role of the VM. The ILPIP addresses and Azure assigned PIP addresses are used for translating VMs' private IP addresses to Internet routable public IP addresses. These addresses identify our VMs only after traffic flows are sent through the NAT Gateway. This means that we cannot use these IP addresses internally to establish connections between internal resources. Example 3-7 verifies that we cannot ping either vm-Back-1 PIP 20.240.41.145 or vm-Bastion own ILPIP 20.91.188.31 from vm-Bastion.

```
nwktAdmin@vm-Back-1:~$ ping 20.240.41.145
PING 20.240.41.145 (20.240.41.145) 56(84) bytes of data.
^C
--- 20.240.41.145 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5119ms

nwktAdmin@vm-Bastion:~$ ping 20.240.41.145
PING 20.240.41.145 (20.240.41.145) 56(84) bytes of data.
^C
--- 20.240.41.145 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3074ms
```

Example 3-7: *Testing Internal Connection Using ILPIP and Azure PIP.*

To allow Azure internal communication between resources in Virtual Networks and Azure services, Azure assigns public IP addresses to VMs, which identifies them internally. Let's call these public IP addresses AzPIP (this totally unofficial abbreviation). You can check the Azure internal Public IP address bound to VM by using the command *dig TXT +short o-o.myaddr.l.google.com.* Example 3-8 shows the AzPIP addresses bound to our example VMs.

```
nwktAdmin@vm-Bastion:~$ dig TXT +short o-o.myaddr.l.google.com
"51.12.224.75"

nwktAdmin@vm-Front-1:~$ dig TXT +short o-o.myaddr.l.google.com
"51.12.96.72"

nwktAdmin@vm-Back-1:~$ dig TXT +short o-o.myaddr.l.google.com
"51.12.97.72"
```

Example 3-8: *Azure Internal Public IP Addresses.*

Note that if you use a name server specification with the command *dig -4 TXT +short o-o.myaddr.l.google.com @ns1.google.com* you will get an ILPIP for an answer.

```
nwktAdmin@vm-Bastion:~$ dig -4 TXT +short o-o.myaddr.l.google.com @ns1.google.com
"20.91.188.31"
```

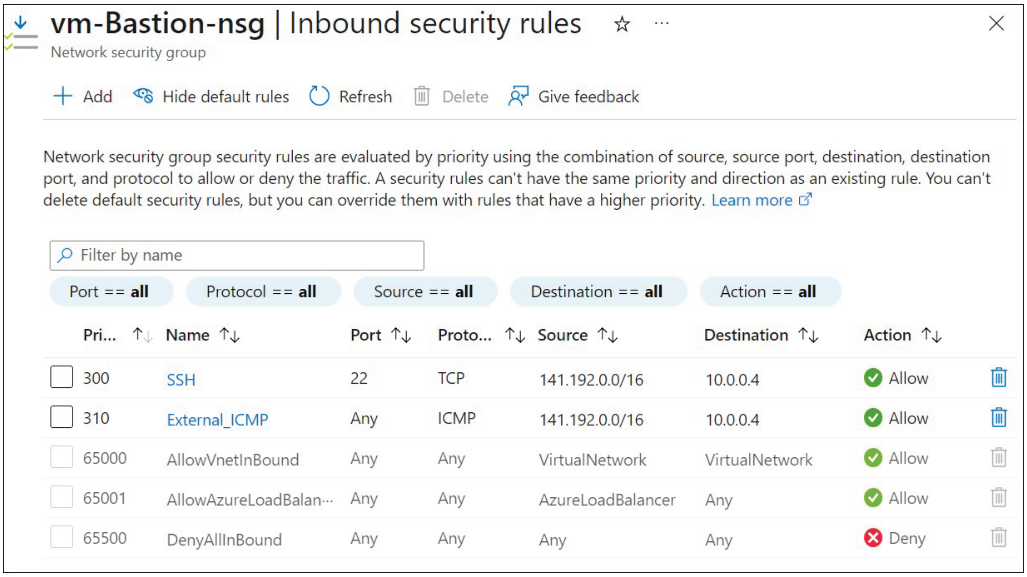
Example 3-8: *Azure Instance Level Public IP Addresses.*

Example 3-9 verifies that the ping from vm-Bastion to vm-Front-1 and vm-Back-1 works using the AzPIP addresses.

```
nwktAdmin@vm-Bastion:~$ ping 51.12.96.72
PING 51.12.96.72 (51.12.96.72) 56(84) bytes of data.
64 bytes from 51.12.96.72: icmp_seq=1 ttl=121 time=2.00 ms
64 bytes from 51.12.96.72: icmp_seq=2 ttl=121 time=2.02 ms
^C
--- 51.12.96.72 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.999/2.011/2.024/0.012 ms

nwktAdmin@vm-Bastion:~$ ping 51.12.97.72
PING 51.12.97.72 (51.12.97.72) 56(84) bytes of data.
64 bytes from 51.12.97.72: icmp_seq=1 ttl=121 time=1.72 ms
64 bytes from 51.12.97.72: icmp_seq=2 ttl=121 time=6.09 ms
^C
--- 51.12.97.72 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.719/3.903/6.088/2.184 ms
```

Example 3-9: *Testing Internal Connection Using AzPIP.*

Be aware that the AzPIP addresses are reachable from the Internet if the NSG security rule allows it. Figure 3-4 shows the inbound security rules of the NSG vm-Bastion-nsg. It allows ICMP traffic from the external subnet 141.192.0.0/16 to vm-Bastion. We need to use private IP addresses in security rules for inbound traffic because the NAT is done before the NSG process.



Network security group security rules are evaluated by priority using the combination of source, source port, destination, destination port, and protocol to allow or deny the traffic. A security rules can't have the same priority and direction as an existing rule. You can't delete default security rules, but you can override them with rules that have a higher priority. Learn more

Pri...	Name	Port	Proto...	Source	Destination	Action
300	SSH	22	TCP	141.192.0.0/16	10.0.0.4	Allow
310	External_ICMP	Any	ICMP	141.192.0.0/16	10.0.0.4	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalan···	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Figure 3-4: *Inbound Security Rules.*

Example 3-10 verifies that we can ping both vm-Bastion's public IP addresses. However, the SSH connection does not work.

```
C:\01-Toni>ping -n 2 20.91.188.31

Pinging 20.91.188.31 with 32 bytes of data:
Reply from 20.91.188.31: bytes=32 time=17ms TTL=46
Reply from 20.91.188.31: bytes=32 time=17ms TTL=46

Ping statistics for 20.91.188.31:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 17ms, Maximum = 17ms, Average = 17ms

C:\01-Toni >ping -n 2 51.12.224.75

Pinging 51.12.224.75 with 32 bytes of data:
Reply from 51.12.224.75: bytes=32 time=16ms TTL=109
Reply from 51.12.224.75: bytes=32 time=17ms TTL=109

Ping statistics for 51.12.224.75:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 16ms, Maximum = 17ms, Average = 16ms
```

Example 3-10: *Testing Inbound Connection from Internet to ILPIP and AzPIP.*

Example 3-11 is shows the partial list IP addresses used in Azure Cloud Swedencentral. You can study Azure IP addressing scheme using Azure Search IP service [11].

```
    {
      "name": "AzureCloud.swedencentral",
      "id": "AzureCloud.swedencentral",
      "properties": {
        "changeNumber": 9,
        "region": "swedencentral",
        "regionId": 76,
        "platform": "Azure",
        "systemService": "",
        "addressPrefixes": [
          "13.105.75.208/28",
          <snipped for brevity>
          "20.91.128.0/17",=> Matches vm-Bastion
          <snipped for brevity>
          "51.12.24.0/21",
          "51.12.32.0/19",
          "51.12.64.0/19",
          "51.12.96.0/21",=> Matches to vm-Front-1 and vm-Back-1
          "51.12.104.32/27",
          "51.12.128.0/21",
          "51.12.144.0/20",
          "51.12.208.0/20",
          "51.12.224.0/19",=> Matches to vm-Bastion
          "51.107.176.0/20",
          "52.101.75.0/24",
          "52.101.80.0/22",
          "52.102.163.0/24",
          <snipped for brevity>
          "20.240.0.0/18",=> Matches to vm-Front-1 and vm-Back-1
        ],
        "networkFeatures": [
          "API",
          "NSG"
        ]
      }
    },
```

Example 3-11: *Azure IP Subnets Associated with Swedencentral.*

References

[1] What is Virtual Network NAT?
https://docs.microsoft.com/en-us/azure/virtual-network/nat-gateway/nat-overview

[2] Chapter 15. Nmap Reference Guide
https://nmap.org/book/man.html

[3] Default outbound access in Azure
https://docs.microsoft.com/en-us/azure/virtual-network/ip-services/default-outbound-access

[4] Upgrade a public IP address using the Azure portal
https://docs.microsoft.com/en-us/azure/virtual-network/ip-services/public-ip-upgrade-portal

[5] Azure Instance Metadata Service (Windows)
https://docs.microsoft.com/en-us/azure/virtual-machines/windows/instance-metadata-service?tabs=windows

[6] Getting Started
https://ipinfo.io/developers

[7] IP addresses used by Azure Monitor
https://docs.microsoft.com/en-us/azure/azure-monitor/app/ip-addresses

[8] Azure IP Ranges and Service Tags – Public Cloud
https://www.microsoft.com/en-us/download/details.aspx?id=56519

[9] Public IP addresses
https://docs.microsoft.com/en-us/azure/virtual-network/ip-services/public-ip-addresses

[10] Version 203 of Cloud Public
https://azureipranges.azurewebsites.net/

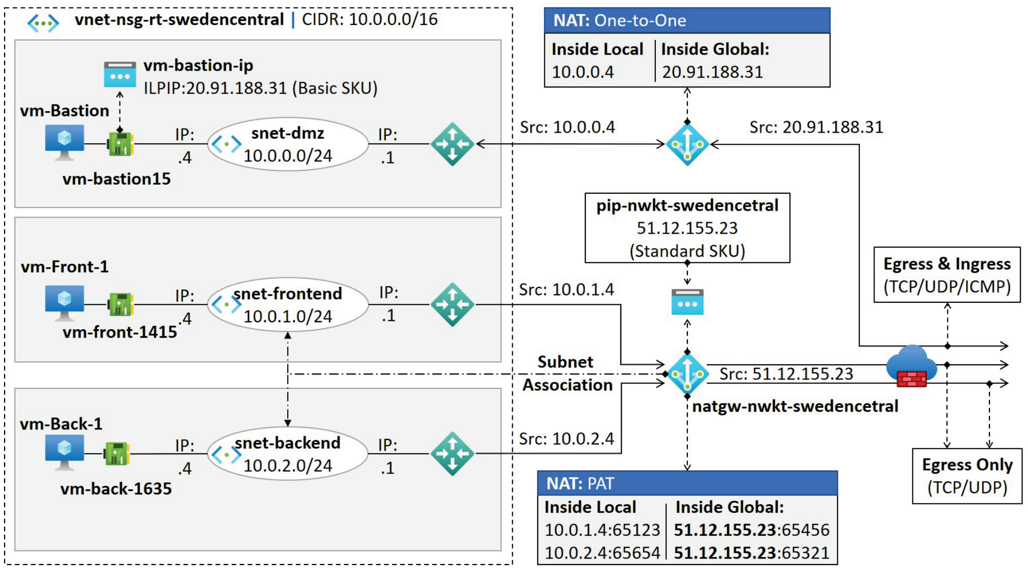
[11] Search for IP (Azure IP Ranges)
https://azureipranges.azurewebsites.net/SearchFor

[12] Search for IP (Azure IP Ranges)
https://azure.microsoft.com/en-us/pricing/details/bandwidth/

Chapter 4: Virtual Network NAT Service - NAT Gateway

Introduction

This chapter explains how VMs without ILPIP can use Azure *Virtual Network NAT service* provided by the Azure *NAT Gateway* for the Internet connection. NAT Gateway is a managed distributed service, which allows an egress-only Internet connection from VMs using TCP and UDP transport layer protocols. The NAT service is produced by Azure's cloud-scale Loadbalancer and NAT solution called Ananta*). We start by creating a public IP resource *pip-nwkt-swedencentral* and a NAT gateway resource *natgw-nwkt-swedencentral*. Then we bind the public IP to NAT GW and attach it to subnets 10.0.1.0/24 (snet-frontend) and 10.0.2.0/24 (snet-backend) in VNet vnet-nsg-rt-swedencentral. As the last step, we verify that everything works as expected.



NAT: One-to-One

Inside Local	Inside Global:
10.0.0.4	20.91.188.31

NAT: PAT

Inside Local	Inside Global:
10.0.1.4:65123	**51.12.155.23**:65456
10.0.2.4:65654	**51.12.155.23**:65321

Figure 4-1: *Virtual Network NAT – NAT Gateway.*

*) Ananta instance consists of three main building blocks 1) Ananta Manager, 2) pool of MUXs (ingress load balancing, VIP advertisement, and IP-in-IP encapsulation), and 3) Hosta Agent (SNAT). Chapter 15 introduces Ananta in detail.

Create Public IP address

Azure NAT Gateway requires *a static SKU* public IP address. We create a regional, standard SKU public IP address, which we can use in any Availability Zone (zonal) within the region. Besides, our example IP uses Azure infrastructure for the Internet connection (routing preference).

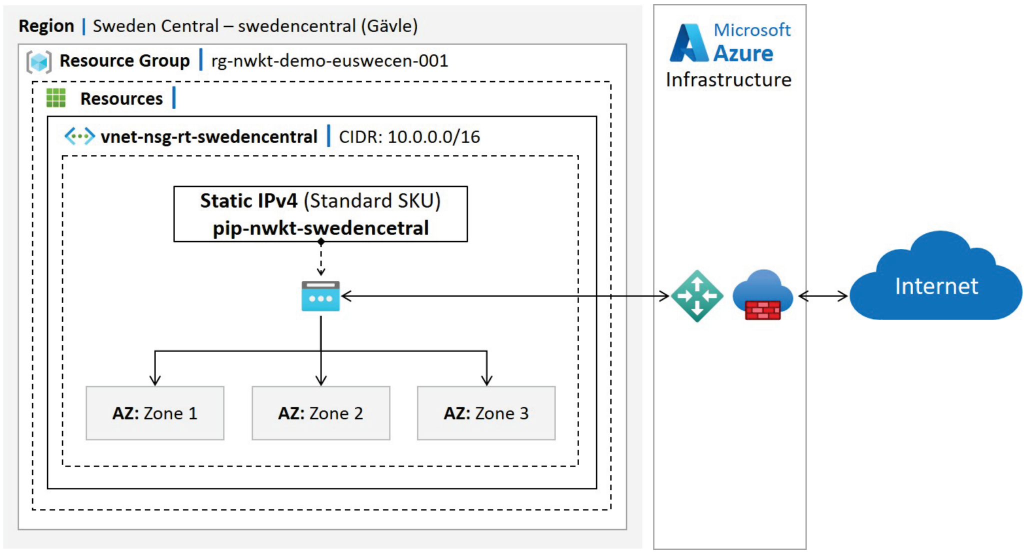
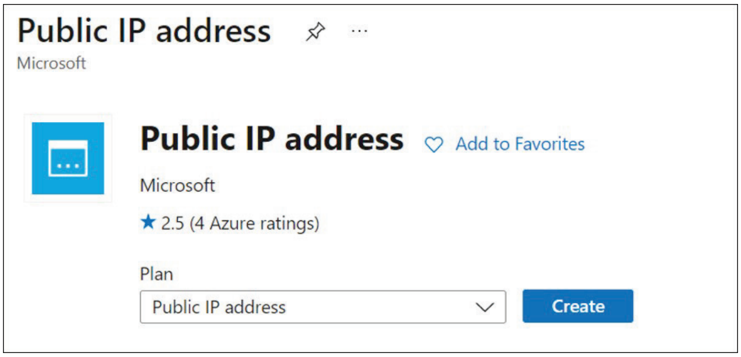

Region | Sweden Central – swedencentral (Gävle)
Resource Group | rg-nwkt-demo-euswecen-001
Resources |
vnet-nsg-rt-swedencentral | CIDR: 10.0.0.0/16
Static IPv4 (Standard SKU)
pip-nwkt-swedencetral
AZ: Zone 1
AZ: Zone 2
AZ: Zone 3
Microsoft Azure
Infrastructure
Internet

Figure 4-2: *Regional, Standard SKU Public IP Address.*

Select the *Create a resource* from the Azure portal Home view. Then type the *Public IP address* on the search field. Next, select the *Microsoft Public IP* address and click the *Create* button.


Public IP address

Microsoft

Public IP address

Add to Favorites

Microsoft

★ 2.5 (4 Azure ratings)

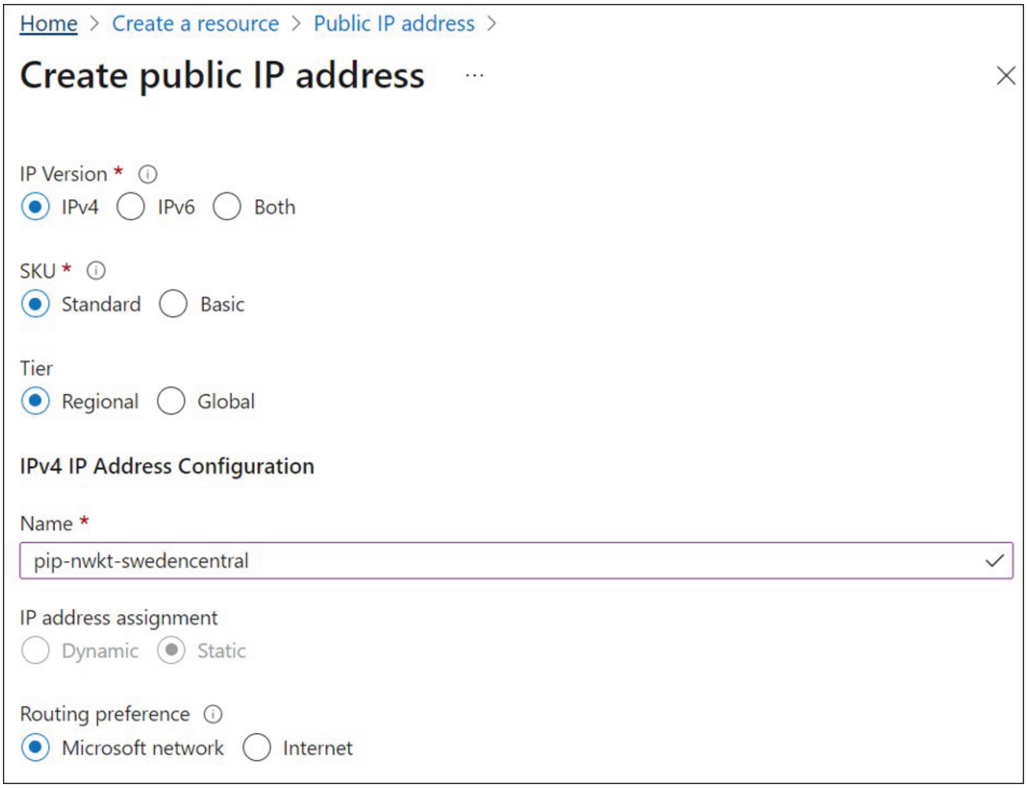
Plan

Public IP address

Create

Figure 4-3: *Create Public IP Address – Phase#1.*

Leave the *IP version*, *SKU*, and *Tier* to their defaults. Give the name of the IP resource. I'm using the naming scheme that describes the resource type, organization, and location. The IP address assignment method for the standard SKU is static, and you can't change it. The *Dynamic* option is available only when the *basic SKU* option is selected. Choose the *Microsoft network* as a Routing preference. This way, the IP address is allocated from the IP range, which uses Microsoft network infrastructure for inbound Internet connection. Next, scroll down and fill in the rest of the information.



IP Version * [✓ IPv4] IPv6 Both

SKU * [✓ Standard] Basic

Tier [✓ Regional] Global

IPv4 IP Address Configuration

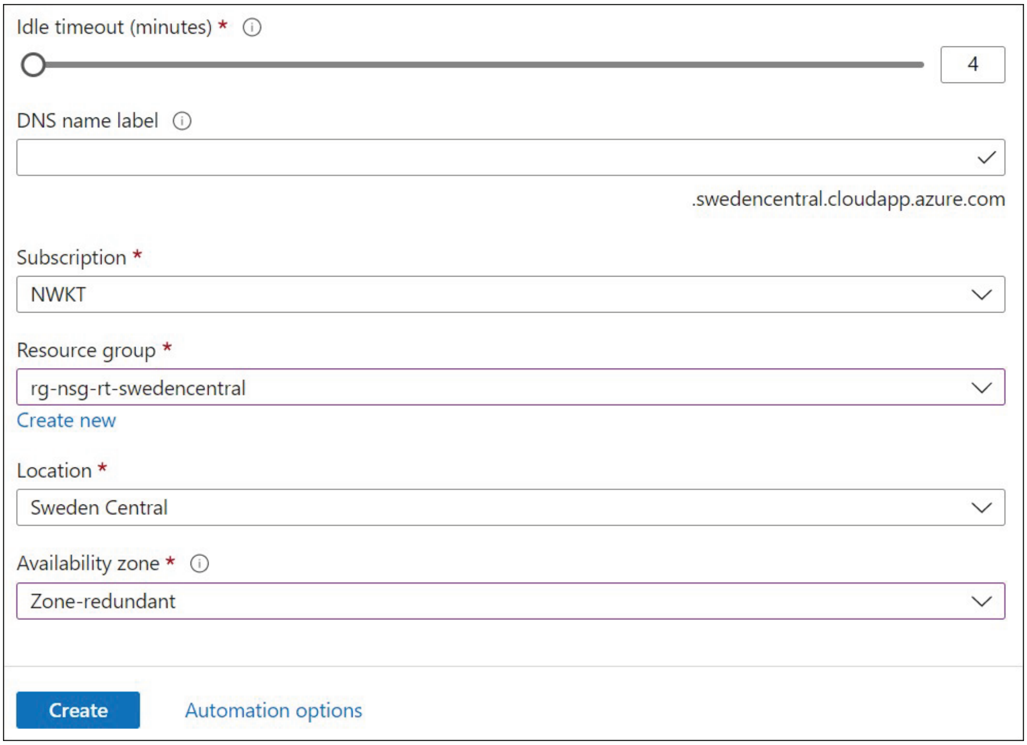
Name *: pip-nwkt-swedencentral

IP address assignment: Dynamic [✓ Static]

Routing preference: [✓ Microsoft network] Internet

Figure 4-4: *Create Public IP Address – Phase#2.1.*

You can adjust the idle timeout between 4 - 30 minutes for flows initiated by internal resources. The inbound timeout is fixed at four minutes. We are assigning this IP address later to NAT Gateway. This is why we don't need to fill in the DNS name label. Select your subscription and resource group. Choose the location and Zone-redundant option. Note that you can associate a Zone-redundant public IP address with a non-zone NAT gateway. Click the *Create* button.


Idle timeout (minutes) * 4

DNS name label .swedencentral.cloudapp.azure.com

Subscription * NWKT

Resource group * rg-nsg-rt-swedencentral

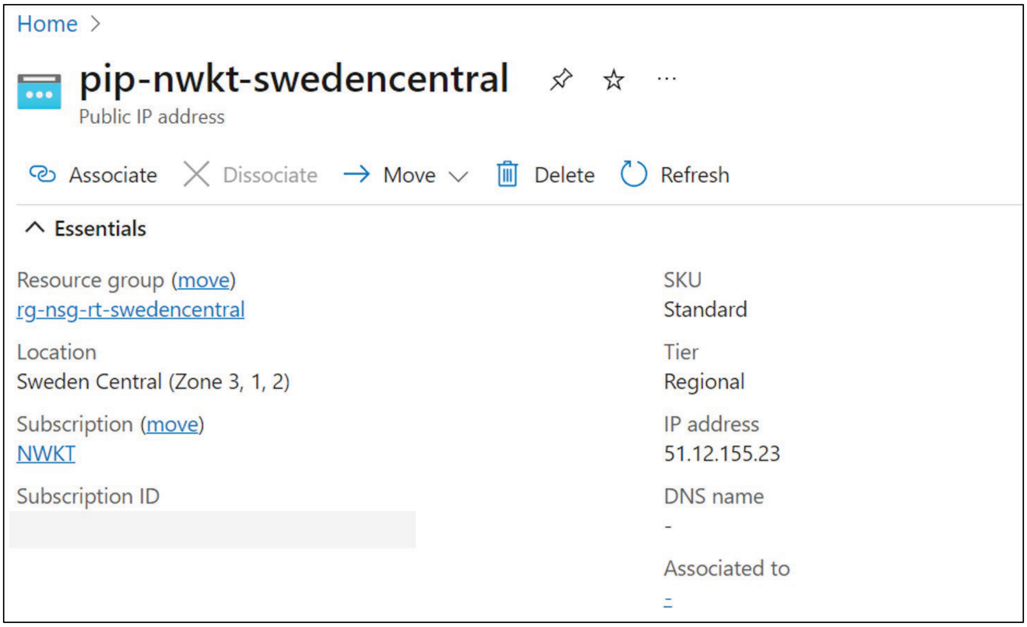
Create new

Location * Sweden Central

Availability zone * Zone-redundant

Figure 4-5: *Create Public IP Address – Phase#2.2.*

Figure 4-6 shows the overview of our new public IP address.



pip-nwkt-swedencentral

Public IP address

Resource group (move) rg-nsg-rt-swedencentral	SKU Standard
Location Sweden Central (Zone 3, 1, 2)	Tier Regional
Subscription (move) NWKT	IP address 51.12.155.23
Subscription ID	DNS name -
	Associated to -

Figure 4-6: *Public IP Address Overview.*

Create NAT Gateway

After creating a public IP address, we launch a NAT gateway. The first steps follow the basic resource deployment processes where we define a Subscription, Resource group, Azure region, and Virtual Network. Then we associate the basic SKU public IP 51.12.155.23 (pip-nwkt-swedencentral) address with the NAT gateway. As the last step, we define the subnets which VMs we want to protect by hiding them behind the same public IP address.

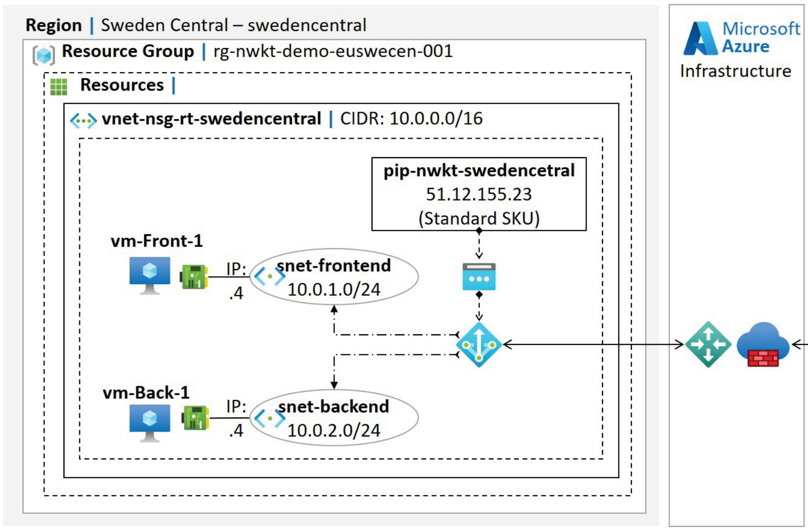
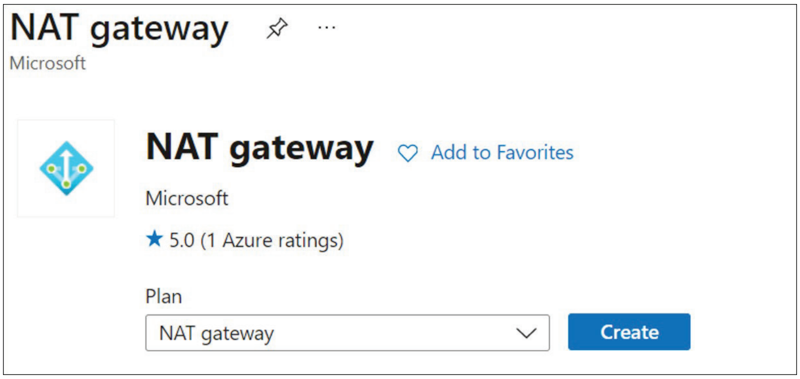


Figure 4-7: *NAT Gateway Overview.*

Select the *Create a resource* from the Azure portal Home view. Then type the *NAT gateway* on the search field. Next, select the Microsoft *NAT Gateway* and click the *Create* button.


NAT gateway

Microsoft

NAT gateway

Add to Favorites

Microsoft

★ 5.0 (1 Azure ratings)

Plan

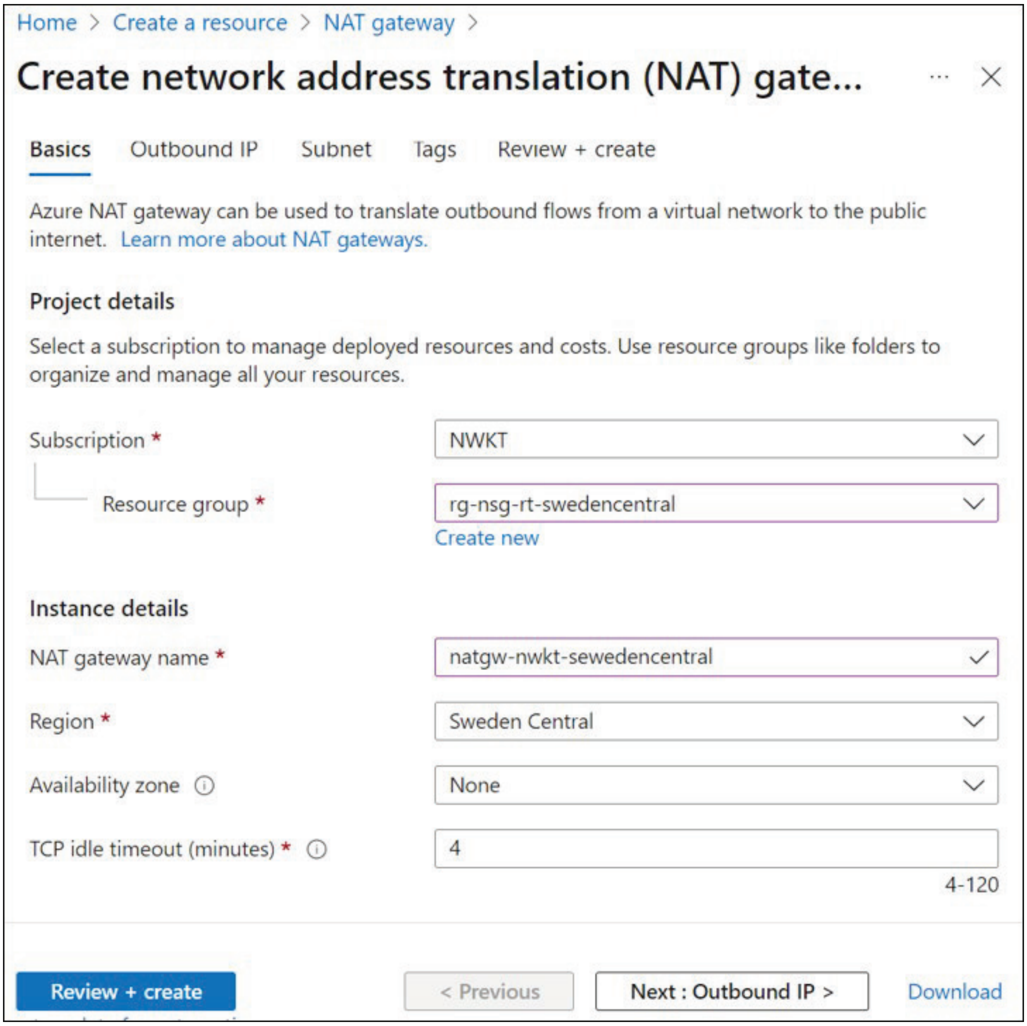
NAT gateway

Create

Figure 4-8: *Create NAT Gateway – Phase#1.*

Basic Settings

Select your Subscription and Resource group from the drop-down menus in the Project details section. Then, name the NAT gateway, and select the region and Availability Zone. We have chosen the *None* option from the Availability zone drop-down menu because the public IP address is zone redundant. You can also adjust the TCP Idle timeout here. Note that it affects only the traffic flows initiated from your internal resources. Click either the *Next: Outbound IP* button or the *Outbound IP* tab to move forward.



Create network address translation (NAT) gate...

Azure NAT gateway can be used to translate outbound flows from a virtual network to the public internet. Learn more about NAT gateways.

Project details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	NWKT
Resource group *	rg-nsg-rt-swedencentral

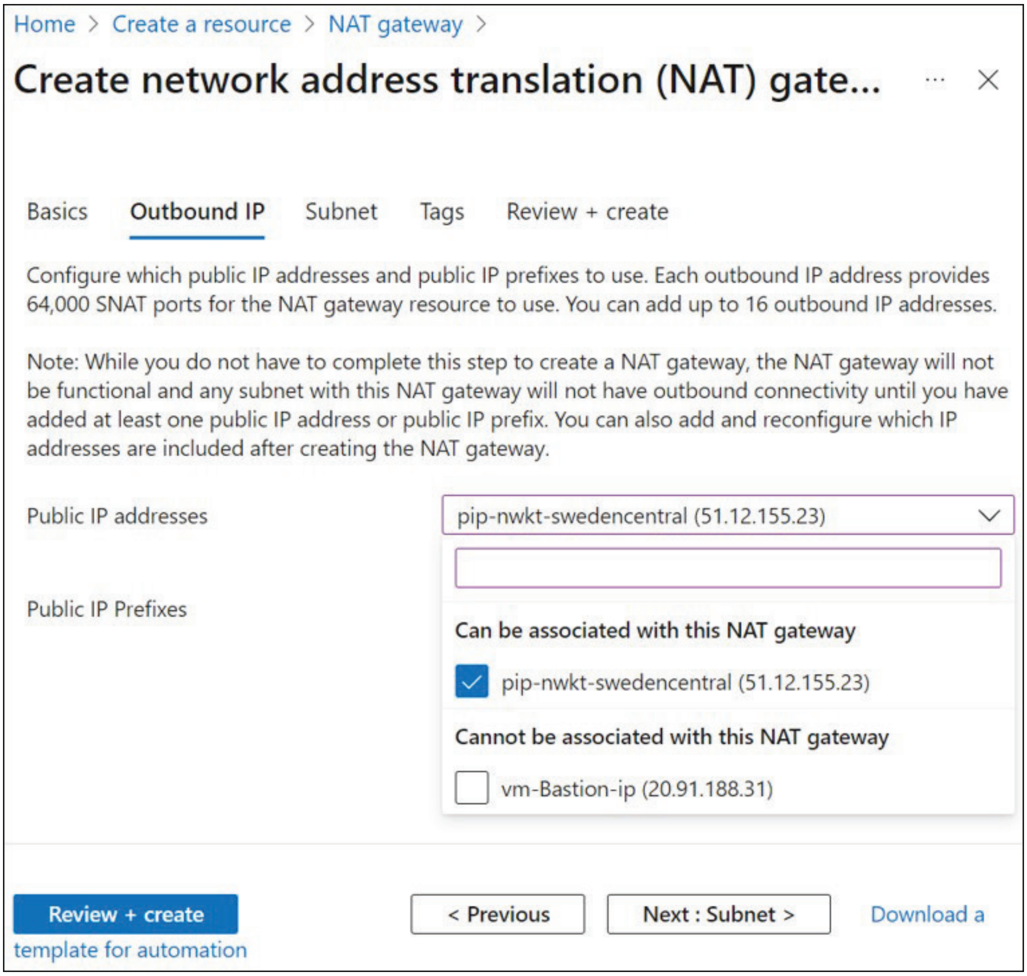
Instance details

NAT gateway name *	natgw-nwkt-sewedencentral
Region *	Sweden Central
Availability zone	None
TCP idle timeout (minutes) *	4 (4-120)

Figure 4-9: *Create NAT Gateway – Basics.*

Outbound IP Address

Select the previously created Public IP address from the drop-down menu. Note that you can't select the public IP associated with vm-Bastion because a) it is currently attached to vm-Bastion, and 2) the Basic SKU is not eligible for NAT Gateway. Click either the *Next: Subnet* button or the *Subnet tab* to proceed.


Home > Create a resource > NAT gateway >
Create network address translation (NAT) gate...
Basics
Outbound IP
Subnet
Tags
Review + create
Review + create
< Previous
Next : Subnet >
Download a template for automation

Configure which public IP addresses and public IP prefixes to use. Each outbound IP address provides 64,000 SNAT ports for the NAT gateway resource to use. You can add up to 16 outbound IP addresses.

Note: While you do not have to complete this step to create a NAT gateway, the NAT gateway will not be functional and any subnet with this NAT gateway will not have outbound connectivity until you have added at least one public IP address or public IP prefix. You can also add and reconfigure which IP addresses are included after creating the NAT gateway.

Public IP addresses: pip-nwkt-swedencentral (51.12.155.23)

Public IP Prefixes

Can be associated with this NAT gateway

✓ pip-nwkt-swedencentral (51.12.155.23)

Cannot be associated with this NAT gateway

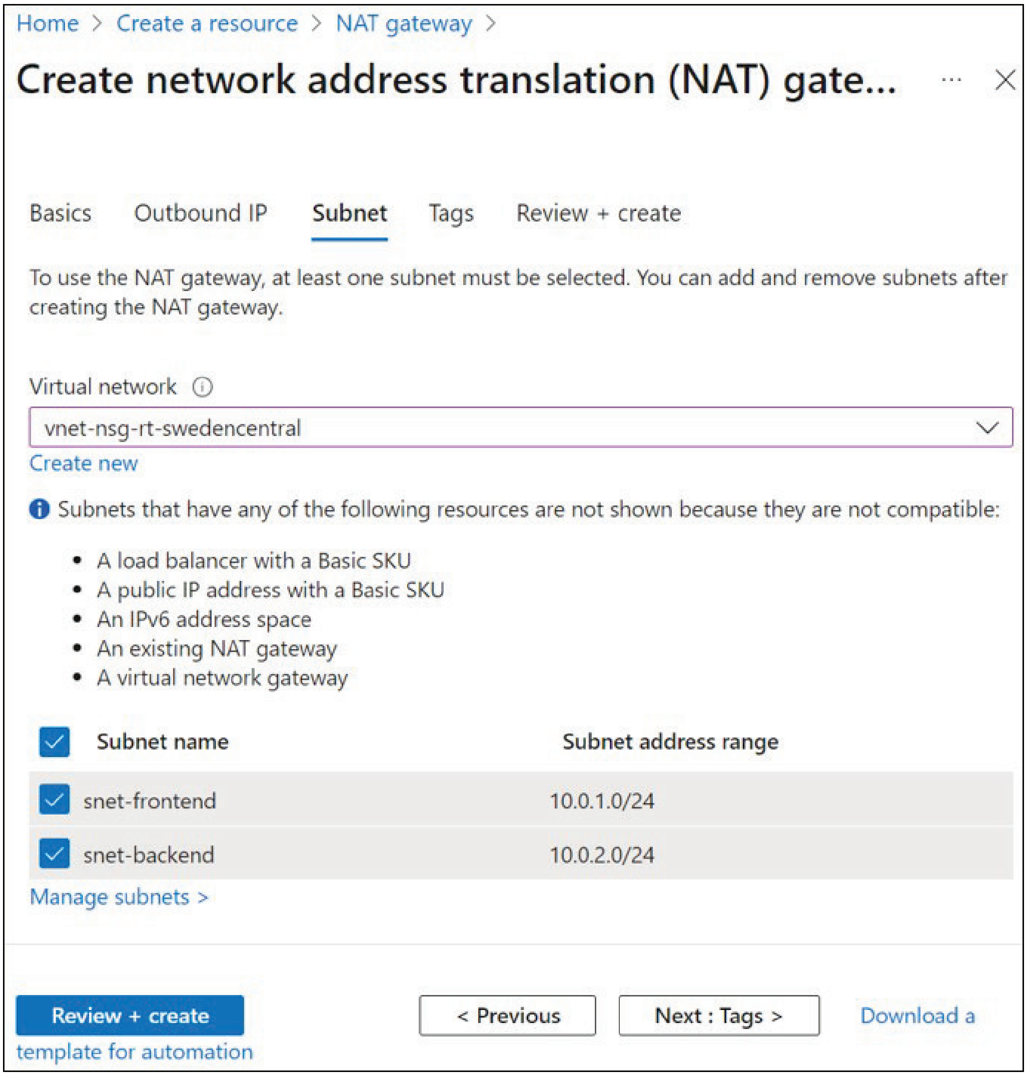
☐ vm-Bastion-ip (20.91.188.31)

Figure 4-10: *Create NAT Gateway – Outbound IP.*

VNet and Subnet Association

Select the Virtual network from the drop-down menu. Then choose the subnets you want to attach to NAT Gateway. Click either the *Next: Review + create* button or select the *Review + create* tab to proceed.

Note! You may have several NAT Gateways within one Virtual Network, which can be associated with different subnets. However, a subnet may have only one attached NAT Gateway.


Home > Create a resource > NAT gateway >
Create network address translation (NAT) gate...
Basics
Outbound IP
Subnet
Tags
Review + create
Create new
Manage subnets >
Review + create
< Previous
Next : Tags >
Download a template for automation

To use the NAT gateway, at least one subnet must be selected. You can add and remove subnets after creating the NAT gateway.

Virtual network: vnet-nsg-rt-swedencentral

Subnets that have any of the following resources are not shown because they are not compatible:

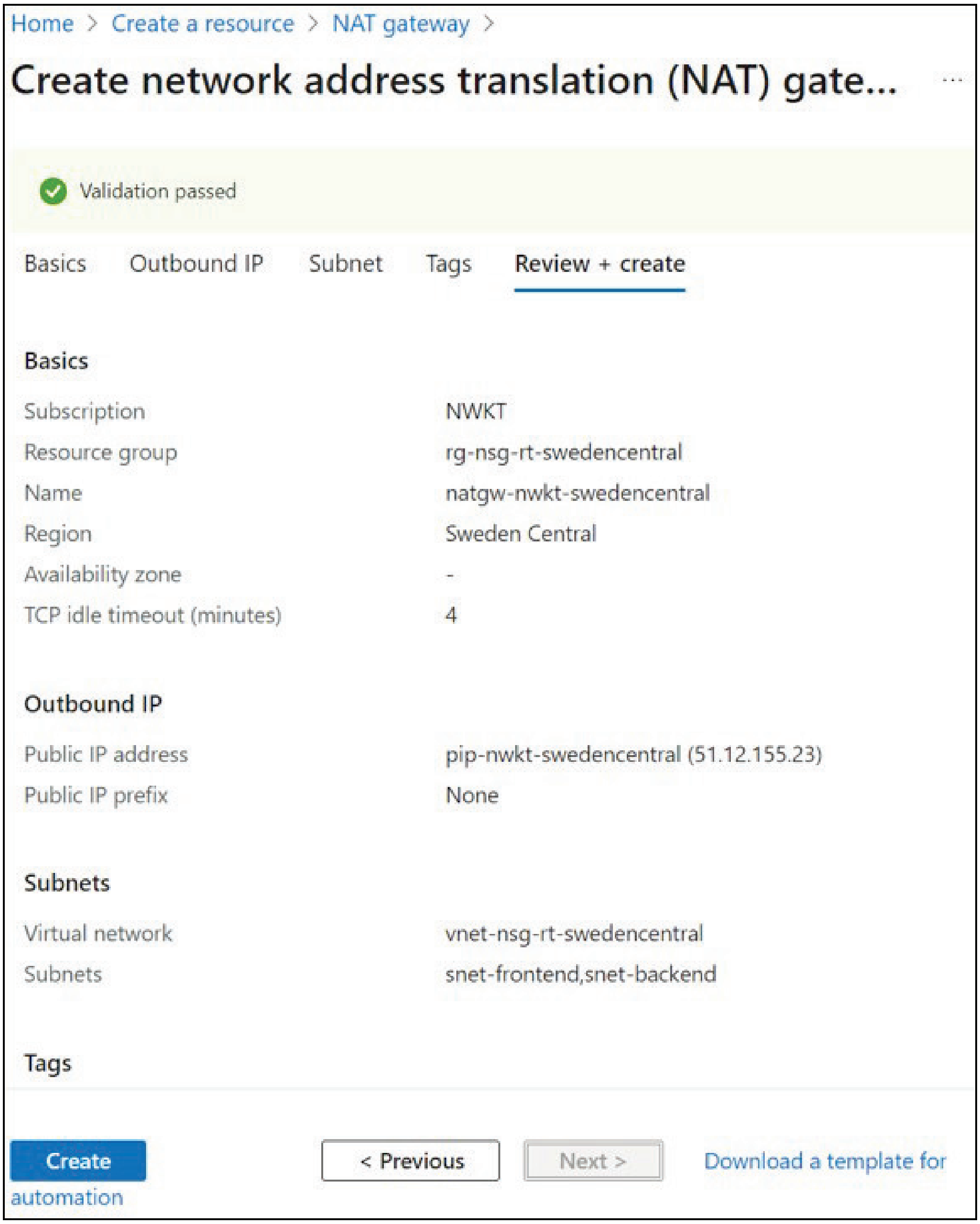
- A load balancer with a Basic SKU
- A public IP address with a Basic SKU
- An IPv6 address space
- An existing NAT gateway
- A virtual network gateway

✓	Subnet name	Subnet address range
✓	snet-frontend	10.0.1.0/24
✓	snet-backend	10.0.2.0/24

Figure 4-11: *Create NAT Gateway – Virtual Network and Subnets.*

Deploying

Azure validates the configuration before you can implement it. Deploy the NAT Gateway by clicking the Create button.



Home > Create a resource > NAT gateway >

Create network address translation (NAT) gate...

Validation passed

Basics | Outbound IP | Subnet | Tags | Review + create

Basics

Subscription	NWKT
Resource group	rg-nsg-rt-swedencentral
Name	natgw-nwkt-swedencentral
Region	Sweden Central
Availability zone	-
TCP idle timeout (minutes)	4

Outbound IP

Public IP address	pip-nwkt-swedencentral (51.12.155.23)
Public IP prefix	None

Subnets

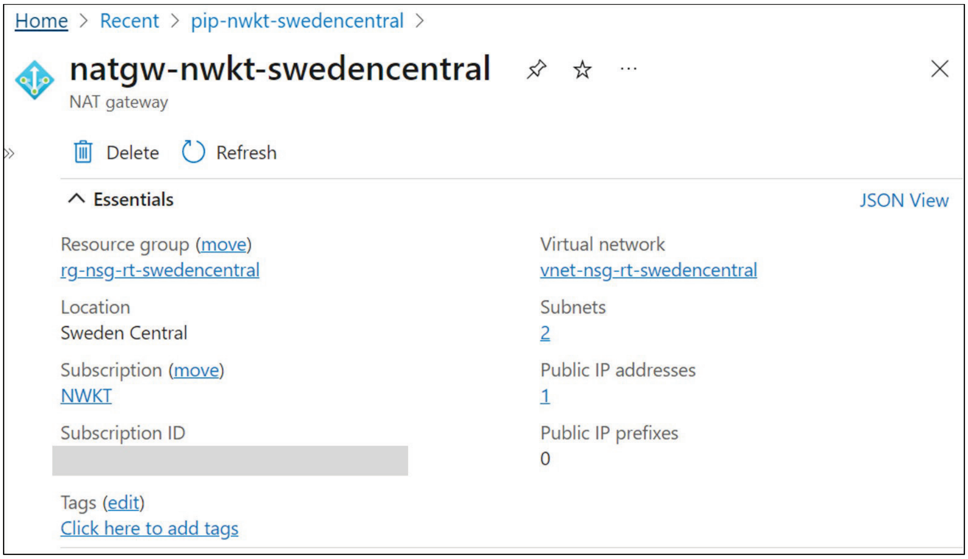
Virtual network	vnet-nsg-rt-swedencentral
Subnets	snet-frontend,snet-backend

Tags

Figure 4-12: *Create NAT Gateway – Review.*

Verification

Navigate to the Azure portal home view and select your NAT gateway from the Resource list. The numbers under the Subnets and Public IP addresses are hyperlinks that lead you to the NAT Gateway settings section.



Home > Recent > pip-nwkt-swedencentral >

natgw-nwkt-swedencentral

NAT gateway

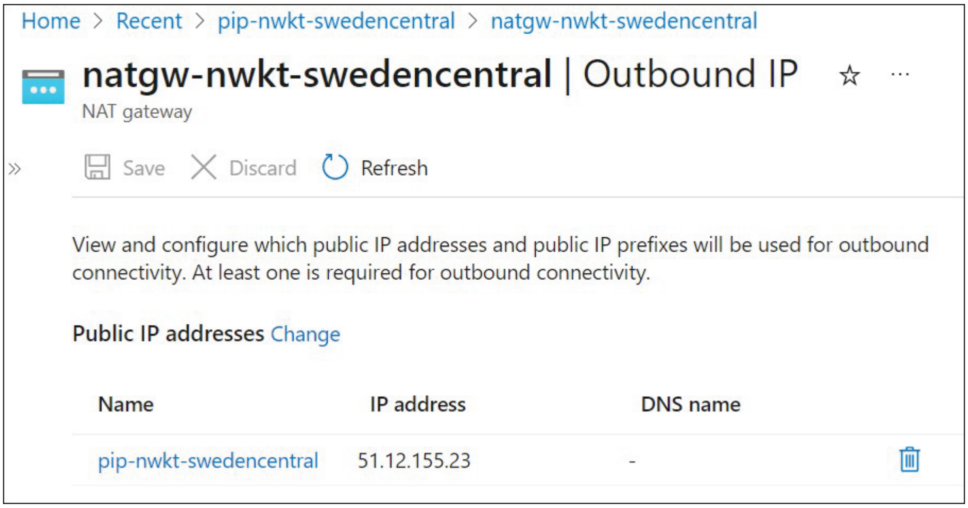
Delete Refresh

Essentials JSON View

Resource group (move) rg-nsg-rt-swedencentral	Virtual network vnet-nsg-rt-swedencentral
Location Sweden Central	Subnets 2
Subscription (move) NWKT	Public IP addresses 1
Subscription ID [redacted]	Public IP prefixes 0
Tags (edit) Click here to add tags	

Figure 4-13: *Nat Gateway - Overview.*

Example 4-14 shows the Outbound IP address bound to the NAT gateway. The name of the IP address is a hyperlink to the Public IP resource. You can navigate to the Public IP address resource by clicking the *pip-nwkt-swedencentral* hyperlink.



Home > Recent > pip-nwkt-swedencentral > natgw-nwkt-swedencentral

natgw-nwkt-swedencentral | Outbound IP

NAT gateway

Save Discard Refresh

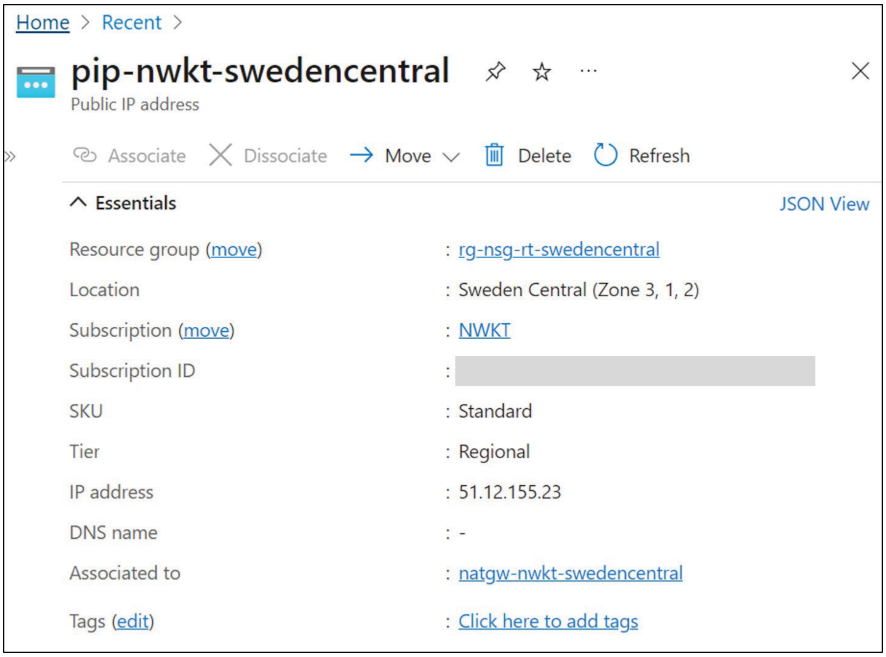
View and configure which public IP addresses and public IP prefixes will be used for outbound connectivity. At least one is required for outbound connectivity.

Public IP addresses Change

Name	IP address	DNS name
pip-nwkt-swedencentral	51.12.155.23	-

Figure 4-14: *Nat Gateway - Overview.*

We can see that the *Associated* field now shows our NAT Gateway resource name.



Resource group (move)	: rg-nsg-rt-swedencentral
Location	: Sweden Central (Zone 3, 1, 2)
Subscription (move)	: NWKT
Subscription ID	: [redacted]
SKU	: Standard
Tier	: Regional
IP address	: 51.12.155.23
DNS name	: -
Associated to	: natgw-nwkt-swedencentral
Tags (edit)	: Click here to add tags

Figure 4-15: *Public IP Address.*

The *Insight* option under the NAT Gateway *Monitoring* section illustrates the visual view of our implementation.

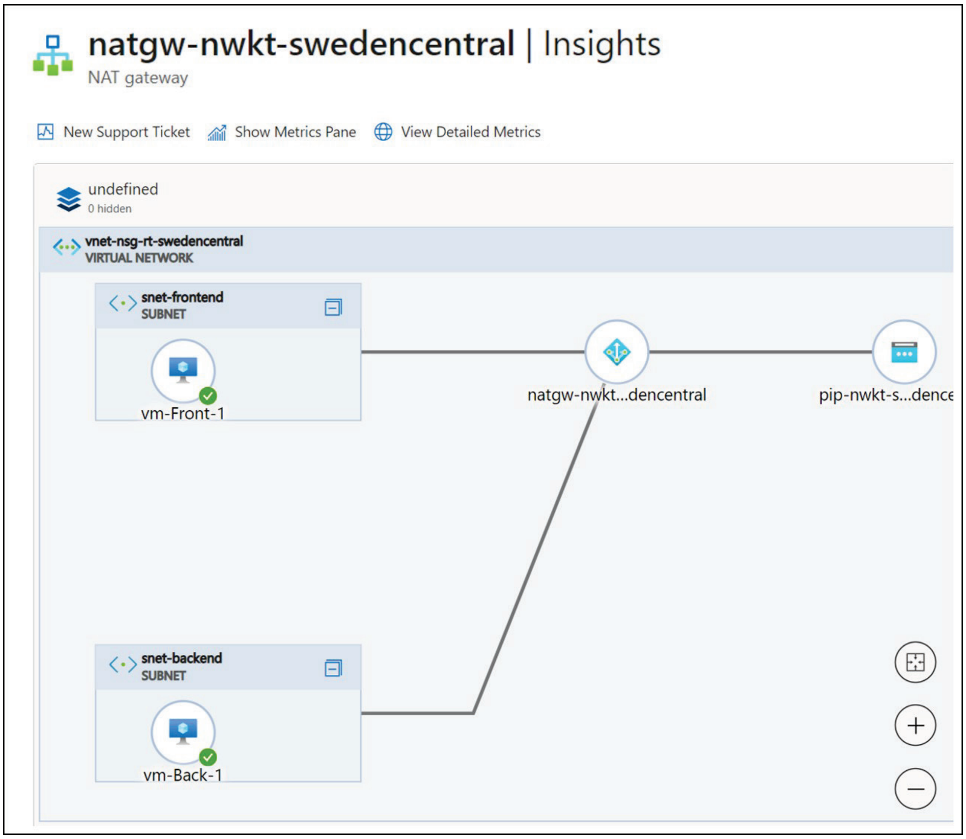


Figure 4-16: *Nat Gateway - Overview.*

The examples below show that the Virtual machine vm-Bastion uses its ILPIP 20.91.188.31 as a source IP address for its Internet traffic. Virtual machines vm-Back-1 and vm-Front-1, in turn, use the same IP address 51.12.155.23 (the Outbound IP of NAT Gateway).

```
nwktAdmin@vm-Bastion:~$ nslookup myip.opendns.com resolver1.opendns.com
Server:         resolver1.opendns.com
Address:        208.67.222.222#53

Non-authoritative answer:
Name:   myip.opendns.com
Address: 20.91.188.31
```
Example 4-1: *Public IP Association with vm-Bastion.*

```
nwktAdmin@vm-Back-1:~$ nslookup myip.opendns.com resolver1.opendns.com
Server:         resolver1.opendns.com
Address:        208.67.222.222#53

Non-authoritative answer:
Name:   myip.opendns.com
Address: 51.12.155.23
```
Example 4-2: *Public IP Association with vm-Back-1.*

```
nwktAdmin@vm-Front-1:~$ nslookup myip.opendns.com resolver1.opendns.com
Server:         resolver1.opendns.com
Address:        208.67.222.222#53

Non-authoritative answer:
Name:   myip.opendns.com
Address: 51.12.155.23
```

Example 4-3: *Public IP Association with vm-Front-1.*

Example 4-4 verifies that HTTP works from the VM vm-Front-1 while ICMP is not allowed.

```
nwktAdmin@vm-Front-1:~$ nmap -p80 nwktimes.blogspot.com
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-29 10:17 UTC
Nmap scan report for nwktimes.blogspot.com (142.250.74.33)
Host is up (0.0057s latency).
Other addresses for nwktimes.blogspot.com (not scanned): 2a00:1450:400f:801::2001
rDNS record for 142.250.74.33: arn09s22-in-f1.1e100.net

PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds

nwktAdmin@vm-Front-1:~$ ping 142.250.74.33 -c3
PING 142.250.74.33 (142.250.74.33) 56(84) bytes of data.
^C
--- 142.250.74.33 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2055ms
```

Example 4-4: *HTTP and ICMP Tests from vm-Front-1.*

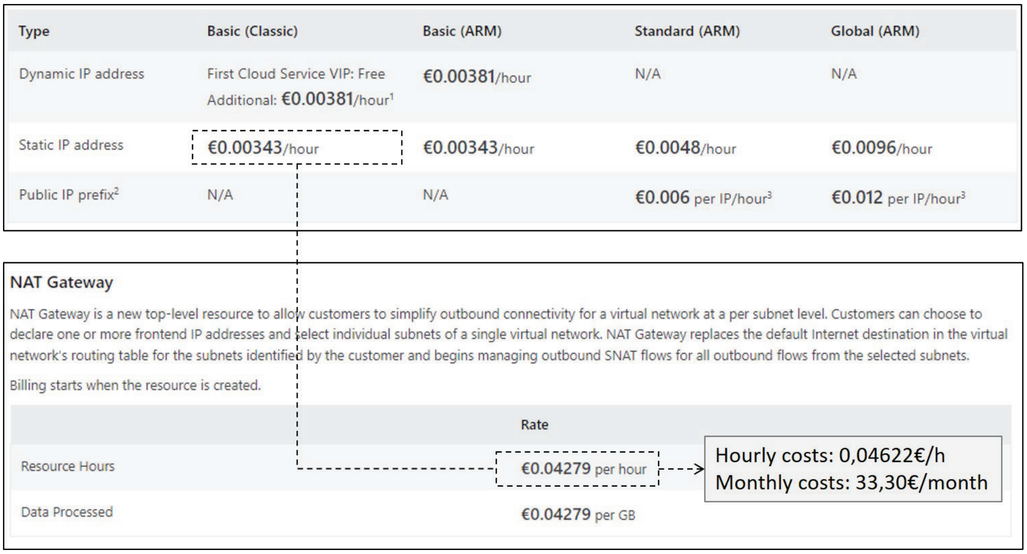
Note that we still can ping the Azure assigned default Outbound Access IP addresses (OPIP) from the external network.

```
C:\01-Toni>ping 51.12.96.72

Pinging 51.12.96.72 with 32 bytes of data:
Reply from 51.12.96.72: bytes=32 time=15ms TTL=109
Reply from 51.12.96.72: bytes=32 time=17ms TTL=109
```

Example 4-5: *External Ping.*

Pricing

Figure 4-16 shows the cost components related to NAT Gateway.



Type	Basic (Classic)	Basic (ARM)	Standard (ARM)	Global (ARM)
Dynamic IP address	First Cloud Service VIP: Free Additional: €0.00381/hour[1]	€0.00381/hour	N/A	N/A
Static IP address	€0.00343/hour	€0.00343/hour	€0.0048/hour	€0.0096/hour
Public IP prefix[2]	N/A	N/A	€0.006 per IP/hour[3]	€0.012 per IP/hour[3]

NAT Gateway

NAT Gateway is a new top-level resource to allow customers to simplify outbound connectivity for a virtual network at a per subnet level. Customers can choose to declare one or more frontend IP addresses and select individual subnets of a single virtual network. NAT Gateway replaces the default Internet destination in the virtual network's routing table for the subnets identified by the customer and begins managing outbound SNAT flows for all outbound flows from the selected subnets.

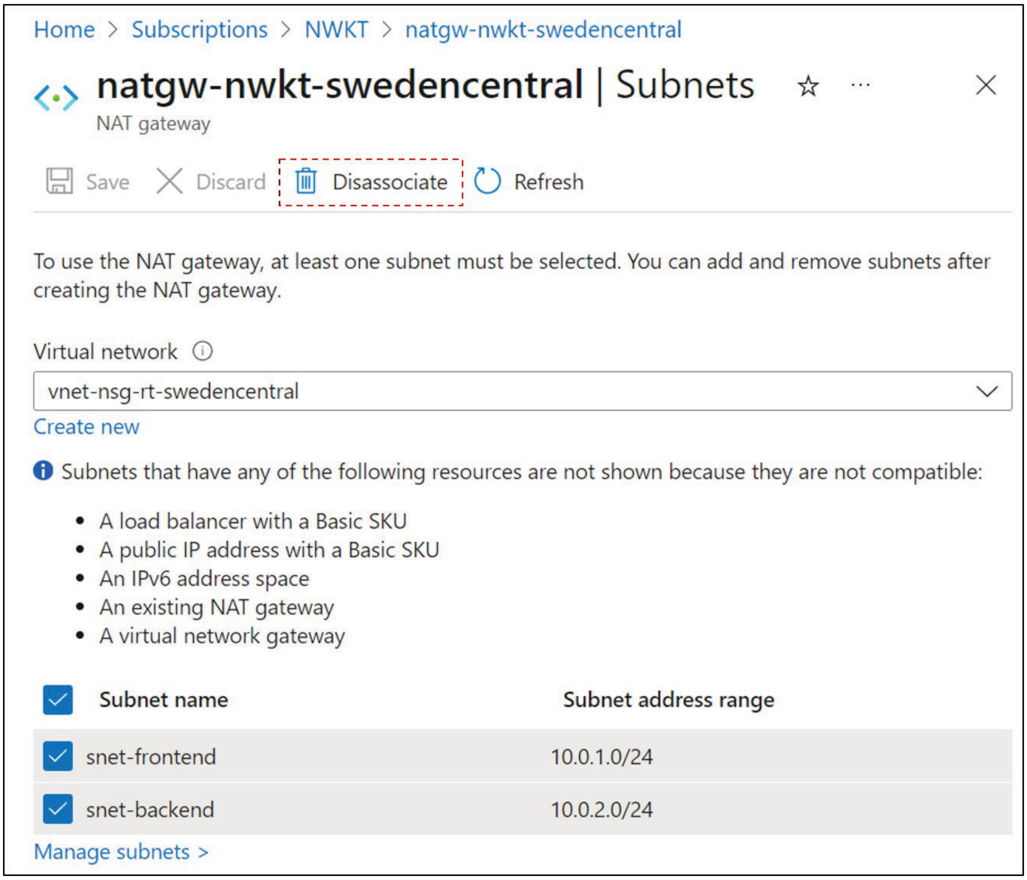
Billing starts when the resource is created.

	Rate
Resource Hours	€0.04279 per hour
Data Processed	€0.04279 per GB

Figure 4-17: *The Cost Structure of a Nat Gateway.*

Delete NAT Gateway

NAT Gateway deletion is a two-step process. First, you have to disassociate subnets. Navigate to the *Subnets* option under the NAT gateway *Settings* section. All associated subnets are selected by default. Click the *Disassociate* option from the horizontal bar.


Home > Subscriptions > NWKT > natgw-nwkt-swedencentral
natgw-nwkt-swedencentral | Subnets
NAT gateway
Save
Discard
Disassociate
Refresh

To use the NAT gateway, at least one subnet must be selected. You can add and remove subnets after creating the NAT gateway.

Virtual network: vnet-nsg-rt-swedencentral

Create new

Subnets that have any of the following resources are not shown because they are not compatible:

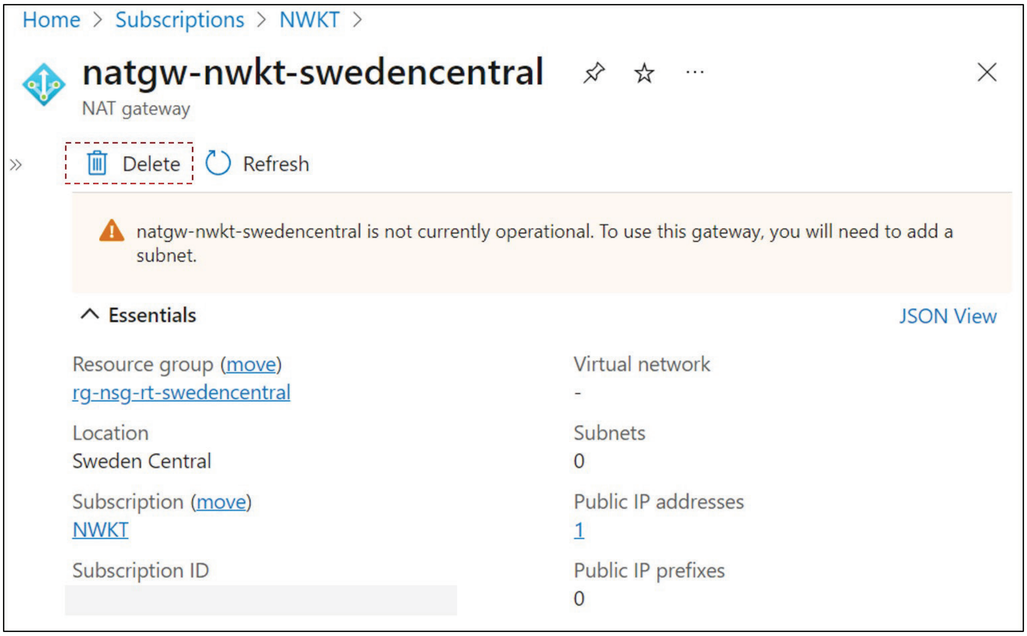
- A load balancer with a Basic SKU
- A public IP address with a Basic SKU
- An IPv6 address space
- An existing NAT gateway
- A virtual network gateway

	Subnet name	Subnet address range
✓	snet-frontend	10.0.1.0/24
✓	snet-backend	10.0.2.0/24

Manage subnets >

Figure 4-18: *Deleting NAT gateway – Subnet Disassociation.*

Next, go to the NAT Gateway *Overview* window after disassociating subnets from it. Then select the *Delete* option. The public IP address bound with NAT Gateway is automatically released but not deleted.



natgw-nwkt-swedencentral is not currently operational. To use this gateway, you will need to add a subnet.

Resource group (move)	Virtual network
rg-nsg-rt-swedencentral	-
Location	Subnets
Sweden Central	0
Subscription (move)	Public IP addresses
NWKT	1
Subscription ID	Public IP prefixes
	0

Figure 4-19: *Deleting NAT gateway – Subnet Disassociation.*

References

[1] Manage a public IP address with a NAT gateway
https://docs.microsoft.com/en-us/azure/virtual-network/ip-services/configure-public-ip-nat-gateway

[2] Public IP addresses
https://docs.microsoft.com/en-us/azure/virtual-network/ip-services/public-ip-addresses

[3] What is routing preference?
https://docs.microsoft.com/en-us/azure/virtual-network/ip-services/routing-preference-overview

[4] IP Addresses pricing
https://azure.microsoft.com/en-us/pricing/details/ip-addresses/

[5] Virtual Network pricing
https://azure.microsoft.com/en-us/pricing/details/virtual-network/

Chapter 5: Hybrid Cloud - Site-to-Site VPN

Introduction

A Hybrid Cloud is a model where we split application-specific workloads across the public and private clouds. This chapter introduces Azure's hybrid cloud solution using Site-to-Site (S2S) Active-Standby VPN connection between Azure and on-prem DC. Azure S2S A/S VPN service includes five Azure resources. The first one, *Virtual Network Gateway (VGW),* also called VPN Gateway, consists of two VMs, one in active mode and the other in standby mode. These VMs are our VPN connection termination points on the Azure side, which encrypt and decrypt data traffic. The active VM has a public IP address associated with its Internet side. If the active VM fails, the standby VM takes the active role, and the public IP is associated with it. Active and standby VMs are attached to the special subnet called *Gateway Subnet*. The name of the gateway subnet has to be *SubnetGateway*. The *Local Gateway (LGW)* resource represents the VPN termination point on the on-prem location. Our example LGW is located behind the NAT device. The inside local IP address of LGW is the private IP 192.168.100.18, which the NAT device translates to public IP 91.156.51.38. Because of this, we set our VGW in *ResponderOnly* mode. The last resource is the *Connection* resource. It defines the tunnel type and its termination points. In our example, we are using Site-to-Site (IPSec) tunnels, which are terminated to our VGW and LGW.

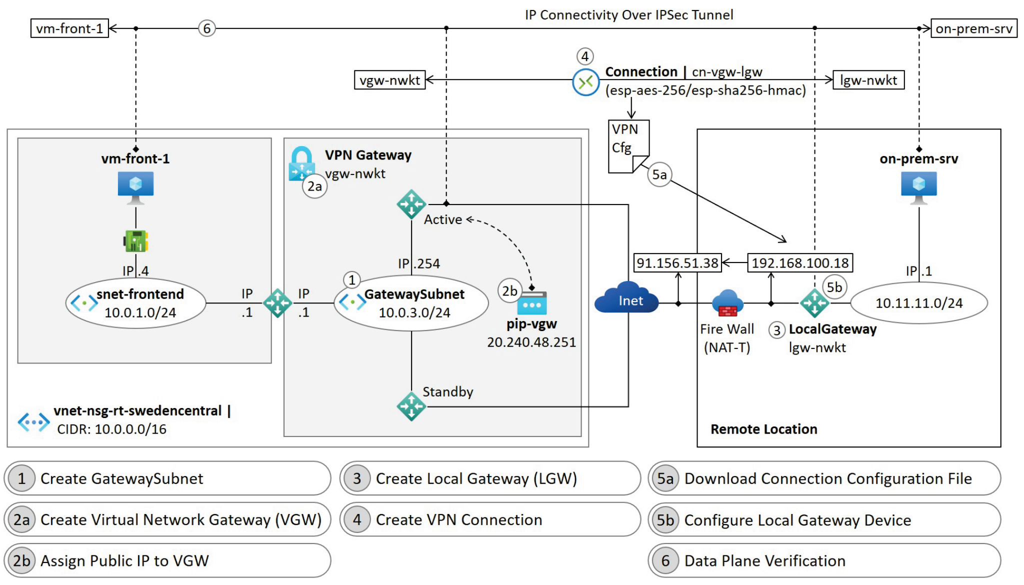
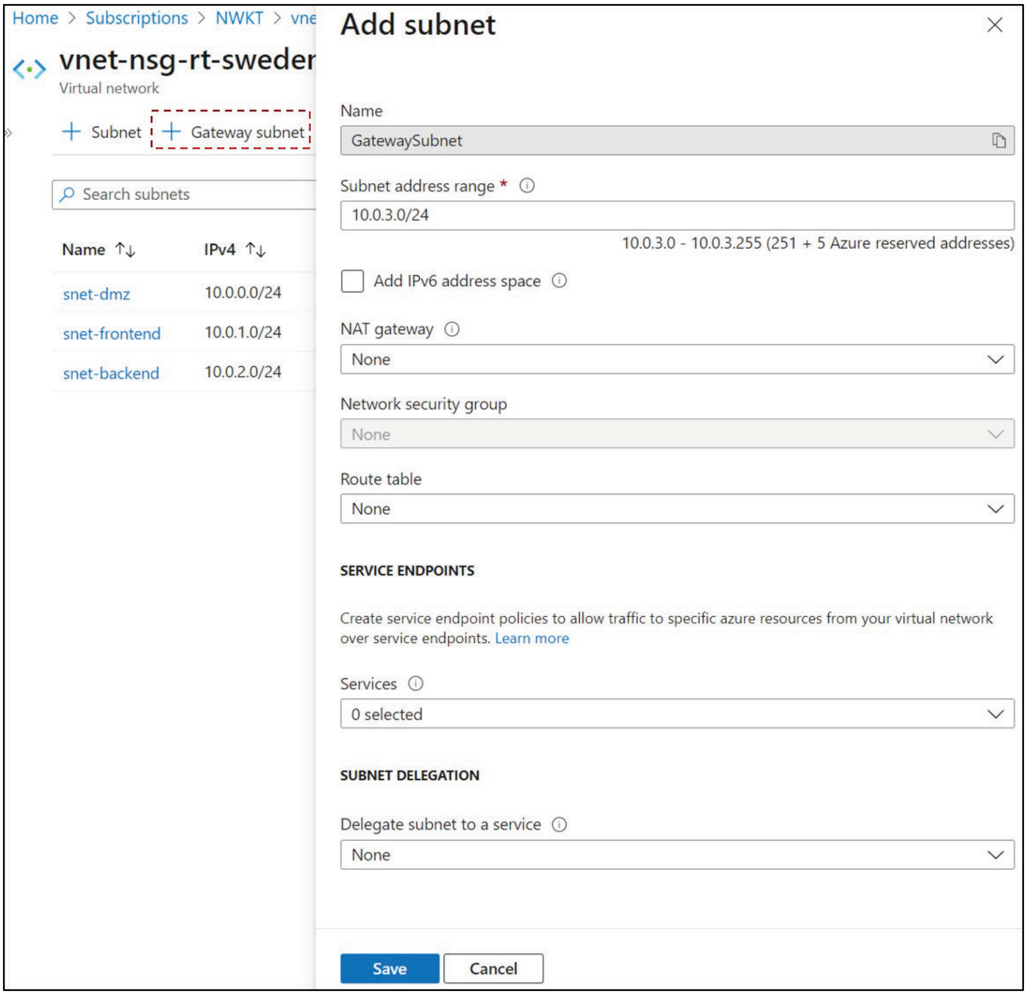


Figure 5-1: *Active-Standby Site-to-Site VPN Overview.*

Create GatewaySubnet

Before we create VGW, we need to deploy a GatewaySubnet. Navigate to *Virtual Network* resource in the Azure portal. Choose the *Subnets* option under the *Settings* section, and click the *Gateway subnet* option. The name *GatewaySubnet* is prefilled, and you can't change it. Azure proposes the next available subnet, in our case 10.0.3.0/24. Note that the recommended subnet mask length is either /27 or /28. You can't associate any Network Security Group (NSG) with the GatewaySubnet. Leave all other fields to their default, and click the *Save* button.



Name	IPv4
snet-dmz	10.0.0.0/24
snet-frontend	10.0.1.0/24
snet-backend	10.0.2.0/24

Add subnet

Name: GatewaySubnet

Subnet address range *: 10.0.3.0/24

10.0.3.0 - 10.0.3.255 (251 + 5 Azure reserved addresses)

☐ Add IPv6 address space

NAT gateway: None

Network security group: None

Route table: None

SERVICE ENDPOINTS

Create service endpoint policies to allow traffic to specific azure resources from your virtual network over service endpoints. Learn more

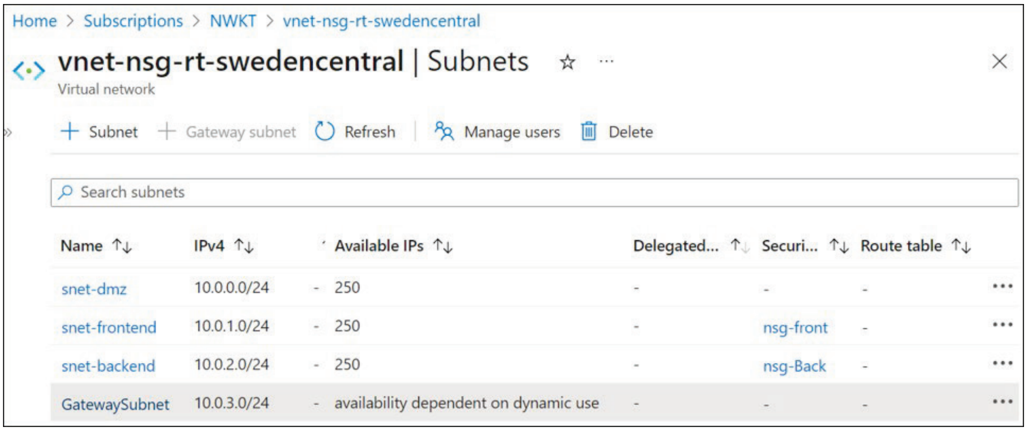
Services: 0 selected

SUBNET DELEGATION

Delegate subnet to a service: None

Figure 5-2: *Create Gateway Subnet for VPN Service.*

Figure 5-3 shows that we have successfully deployed GatewaySubnet.



Home > Subscriptions > NWKT > vnet-nsg-rt-swedencentral

vnet-nsg-rt-swedencentral | Subnets

Virtual network

Name	IPv4	Available IPs	Delegated...	Securi...	Route table
snet-dmz	10.0.0.0/24	- 250	-	-	-
snet-frontend	10.0.1.0/24	- 250	-	nsg-front	-
snet-backend	10.0.2.0/24	- 250	-	nsg-Back	-
GatewaySubnet	10.0.3.0/24	- availability dependent on dynamic use	-	-	-

Figure 5-3: *GatewaySubnet Verification.*

Create Virtual Network Gateway (VGW)

After creating a SubnetGateway, we can launch a VGW and associate it with the previously deployed GatewaySubnet. Because we are using Active-Standby redundancy, we only need one public IP address, which is the IPSec tunnel IP address for VGW. This address is then associated with the active VM. Azure also adds a private IP address to VMs from the GatewaySubnet address space. In our case, it is 10.0.3.254/24 (see example 5-1).

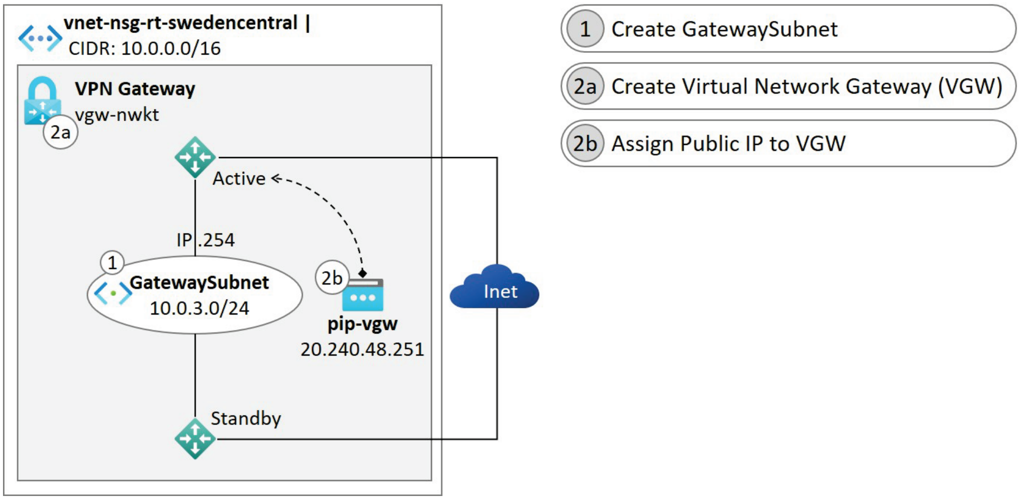


Figure 5-4: *Virtual Network Gateway.*

Select the *Create a resource* from the Azure portal Home view. Then type the *Virtual Network gateway* on the search field. Next, select the Microsoft *Virtual network gateway* and click the *Create* button.

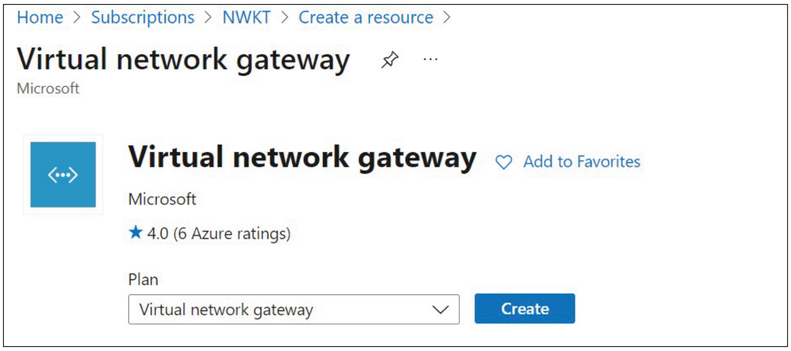
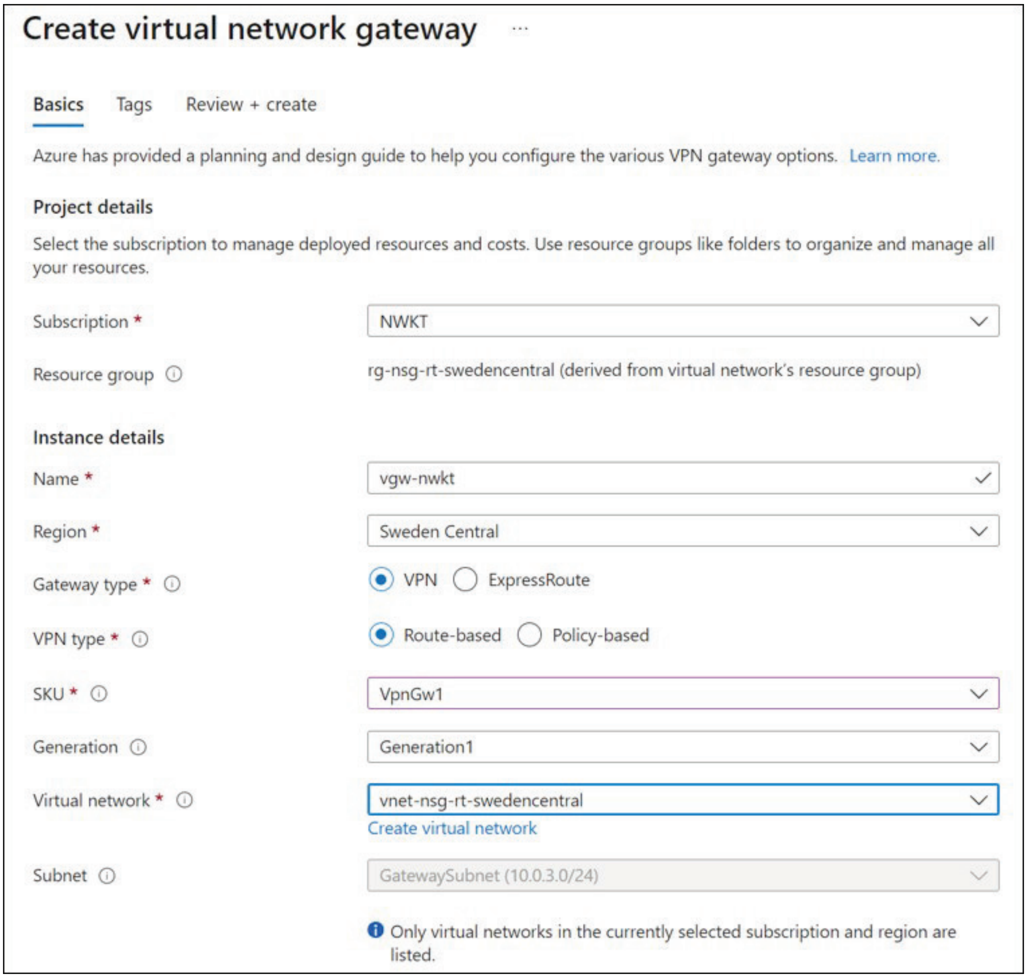


Figure 5-5: *Create VPN Gateway – Step#1.*

First, we select our subscription (NWKT) from the drop-down menu. We don't have to define a resource group because that is derived from the Virtual Network's resource group. Fill in the Name field (vgw-nwkt) and select the Azure region (swedencentral). Choose the *VPN* option Gateway type and *Route based* VPN type. The route-Based supports dynamic routing with BGP. We are using the VpnGw1 SKU (Generation 1), which allows a 650 Mbps transfer rate and 30 Site-to-Site VPN tunnels. Scroll down to the IP section.


Create virtual network gateway

Azure has provided a planning and design guide to help you configure the various VPN gateway options. Learn more.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	NWKT
Resource group	rg-nsg-rt-swedencentral (derived from virtual network's resource group)

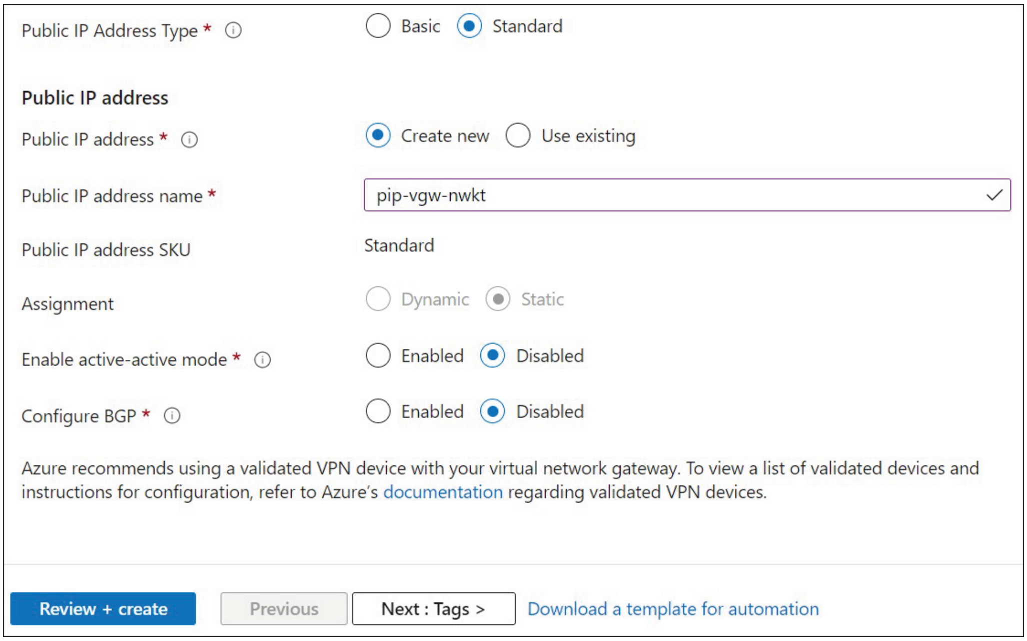
Instance details

Name *	vgw-nwkt
Region *	Sweden Central
Gateway type *	[✓ VPN] ExpressRoute
VPN type *	[✓ Route-based] Policy-based
SKU *	VpnGw1
Generation	Generation1
Virtual network *	vnet-nsg-rt-swedencentral Create virtual network
Subnet	GatewaySubnet (10.0.3.0/24)

Only virtual networks in the currently selected subscription and region are listed.

Figure 5-6: *Create VPN Gateway – Step#2.1.*

We create a new standard public IP address (pip-vgw-nwkt) that will be associated with the active VGW during the deployment process. Leave the *Enable active-active mode* and the *Configure BGP* radio buttons to default. Click the *Review + create* button to proceed.

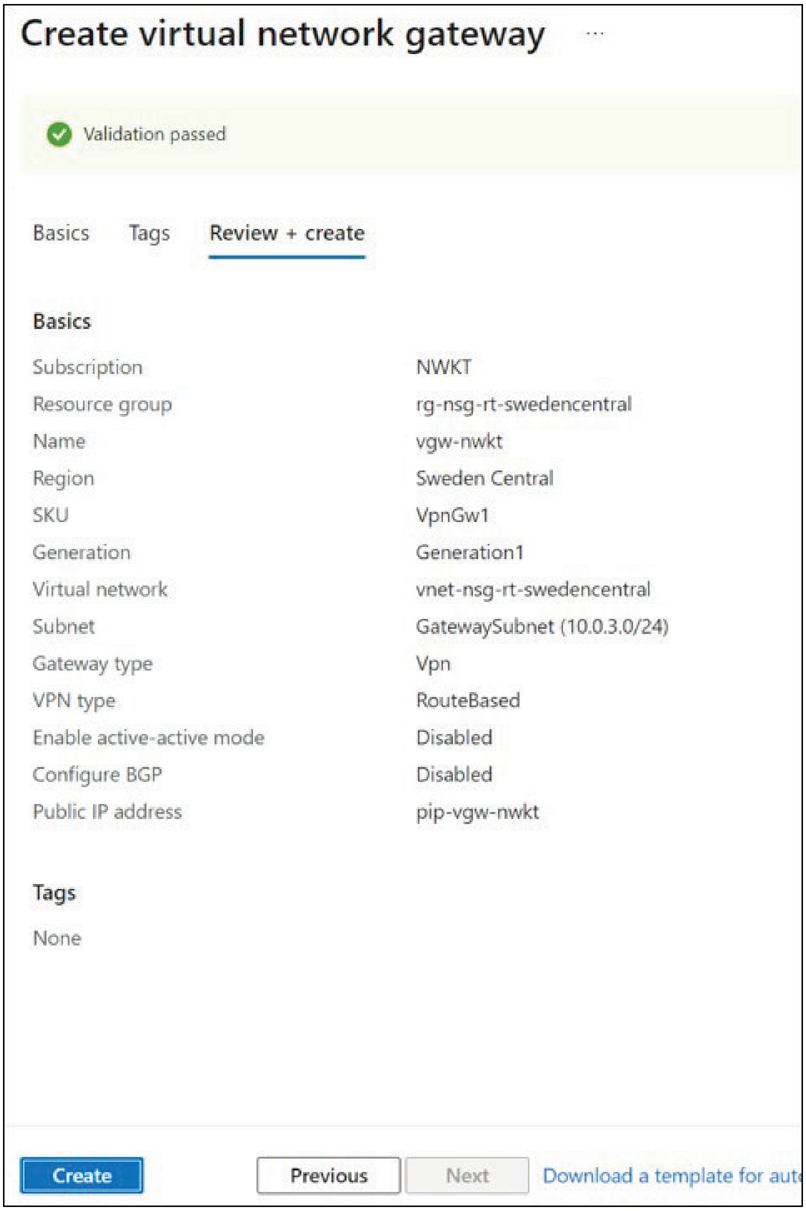


Public IP Address Type *	Basic [✓ Standard]
Public IP address	
Public IP address *	[✓ Create new] Use existing
Public IP address name *	pip-vgw-nwkt
Public IP address SKU	Standard
Assignment	Dynamic [✓ Static]
Enable active-active mode *	Enabled [✓ Disabled]
Configure BGP *	Enabled [✓ Disabled]

Azure recommends using a validated VPN device with your virtual network gateway. To view a list of validated devices and instructions for configuration, refer to Azure's documentation regarding validated VPN devices.

Figure 5-7: *Create VPN Gateway – Step#2.2.*

Figure 5-8 shows the review window. Start the deployment by clicking the Create button. The process may take up to 45 minutes.



Create virtual network gateway

Validation passed

Basics

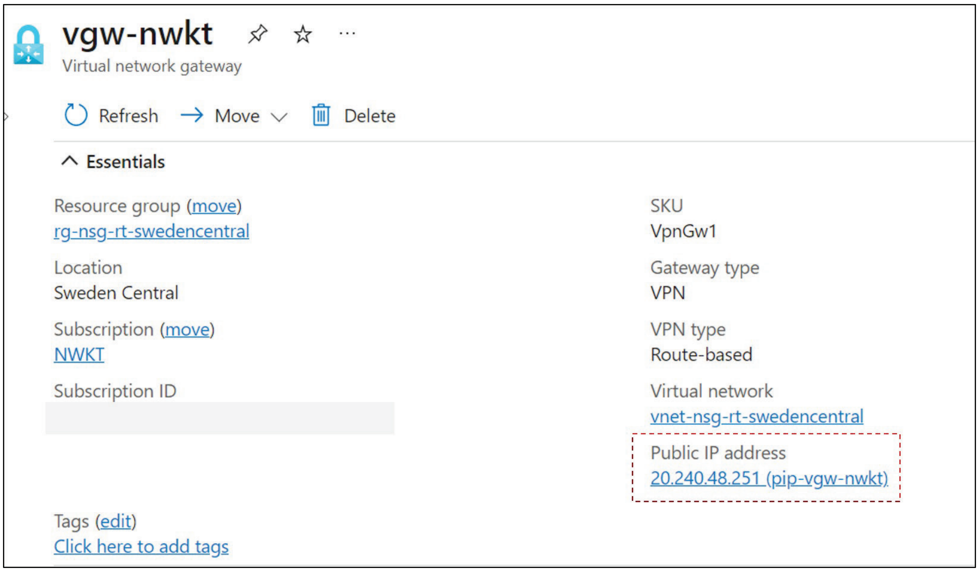
Subscription	NWKT
Resource group	rg-nsg-rt-swedencentral
Name	vgw-nwkt
Region	Sweden Central
SKU	VpnGw1
Generation	Generation1
Virtual network	vnet-nsg-rt-swedencentral
Subnet	GatewaySubnet (10.0.3.0/24)
Gateway type	Vpn
VPN type	RouteBased
Enable active-active mode	Disabled
Configure BGP	Disabled
Public IP address	pip-vgw-nwkt

Tags

None

Figure 5-8: *VPN Gateway – Review.*

Figure 5-9 shows the overview of VPN Gateway vgw-nwkt. You can see that Azure has allocated public IP address 20.240.48.251 to VGW.



Resource group (move)
rg-nsg-rt-swedencentral

Location
Sweden Central

Subscription (move)
NWKT

Subscription ID

SKU
VpnGw1

Gateway type
VPN

VPN type
Route-based

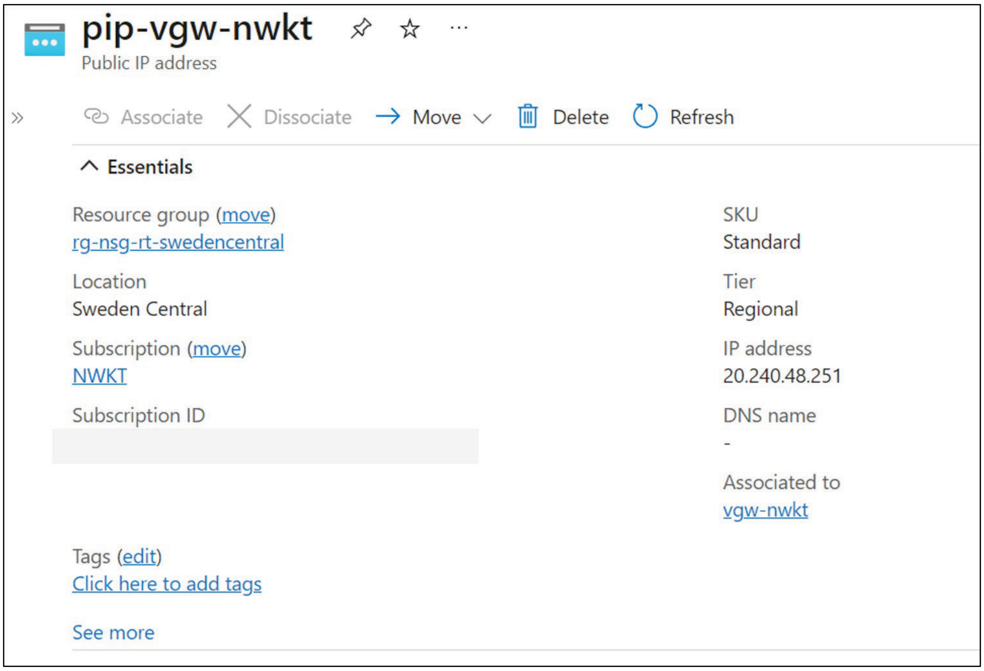
Virtual network
vnet-nsg-rt-swedencentral

Public IP address
20.240.48.251 (pip-vgw-nwkt)

Tags (edit)
Click here to add tags

Figure 5-9: *Virtual Network Gateway Overview (vgw-nwkt).*

Figure 5-9 shows an overview of the public IP address assigned to VGW.



Resource group (move)
rg-nsg-rt-swedencentral

Location
Sweden Central

Subscription (move)
NWKT

Subscription ID

SKU
Standard

Tier
Regional

IP address
20.240.48.251

DNS name
-

Associated to
vgw-nwkt

Tags (edit)
Click here to add tags

See more

Figure 5-10: *VGW's Public IP Address (pip-vgw-nwkt) Overview.*

Figure 5-11 shows that the public IP address *pip-vgw-nwkt* is associated with VPN Gateway *vgw-nwkt*. The VPN Gateway is attached to the subnet *SubnetGateway* on the Virtual Network *vnet-nsg-rt-swedencentral*.

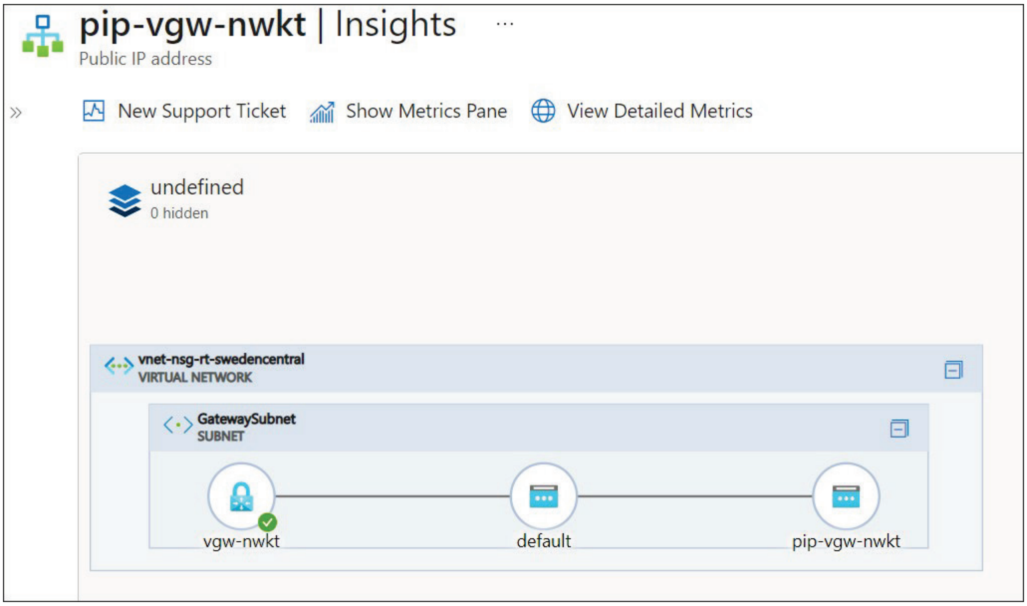


Figure 5-11: *The Insight View of the Public IP (pip-vgw-nwkt).*

You can verify VGW's parameters and configuration using the Azure PowerShell command *Get-AzVirtualNetworkGateway -name [name] -ResourceGroupName [name].*

```
Get-AzVirtualNetworkGateway -name vgw-nwkt -ResourceGroupName rg-nsg-rt-swedencentral

Name                             : vgw-nwkt
ResourceGroupName                : rg-nsg-rt-swedencentral
Location                         : swedencentral
Id                               : /subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworkGateways/vgw-nwkt
Etag                             : W/"d544cf91-ddd8-47ce-9323-64430c73e79e"
ResourceGuid                     : 34ec4c28-e594-4511-923b-37452e822081
ProvisioningState                : Succeeded
Tags                             :
IpConfigurations                 : [
                                     {
                                       "PrivateIpAllocationMethod": "Dynamic",
                                       "Subnet": {
                                         "Id": "/subscriptions/xxx/resourceGroups/rg-
nsg-rt-swedencentral/providers/Microsoft.Network/virtualNetworks/vnet-nsg-rt-
swedencentral/subnets/GatewaySubnet"
                                       },
                                       "PublicIpAddress": {
                                         "Id": "/subscriptions/xxx/resourceGroups/rg-
nsg-rt-swedencentral/providers/Microsoft.Network/publicIPAddresses/pip-vgw-nwkt"
```

```
                                      },
                                      "Name": "default",
                                      "Etag": "W/\"d544cf91-ddd8-47ce-9323-
64430c73e79e\"",
                                      "Id": "/subscriptions/xxx/resourceGroups/rg-nsg-
rt-swedencentral/providers/Microsoft.Network/virtualNetworkGateways/vgw-
nwkt/ipConfigurations/default"
                                    }
                                  ]
GatewayType                     : Vpn
VpnType                         : RouteBased
EnableBgp                       : False
ActiveActive                    : False
GatewayDefaultSite              : null
Sku                             : {
                                    "Capacity": 2,
                                    "Name": "VpnGw1",
                                    "Tier": "VpnGw1"
                                  }
VpnClientConfiguration          : null
BgpSettings                     : {
                                    "Asn": 65515,
                                    "BgpPeeringAddress": "10.0.3.254",
                                    "PeerWeight": 0,
                                    "BgpPeeringAddresses": [
                                      {
                                        "IpconfigurationId":
"/subscriptions/xxx/providers/Microsoft.Network/virtualNetworkGateways/vgw-
nwkt/ipConfigurations/default",
                                        "DefaultBgpIpAddresses": [
                                          "10.0.3.254"
                                        ],
                                        "CustomBgpIpAddresses": [],
                                        "TunnelIpAddresses": [
                                          "20.240.48.251"
                                        ]
                                      }
                                    ]
                                  }
CustomRoutes                    : null
NatRules                        : []
EnableBgpRouteTranslationForNat : False
```

Example 5-1: *Verifying VPN Gateway Settings with Azure PowerShell.*

Create Local Gateway (LGW)

Our remote site VPN device is Cisco CSR1000v. It is behind the Internet FireWall, which hides all the internal addresses behind the public IP address 91.156.51.38 (Port Address Translation/PAT). For this reason, we select the *ResponderOnly* option on the site-to-site VPN connection settings. Note that in this phase, we only create a resource and do not configure the remote VPN device. We can download the remote VPN device configuration file after deploying a Connection resource.

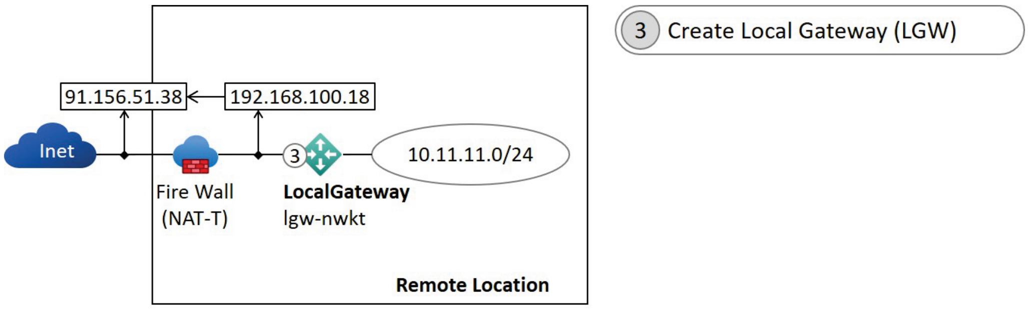
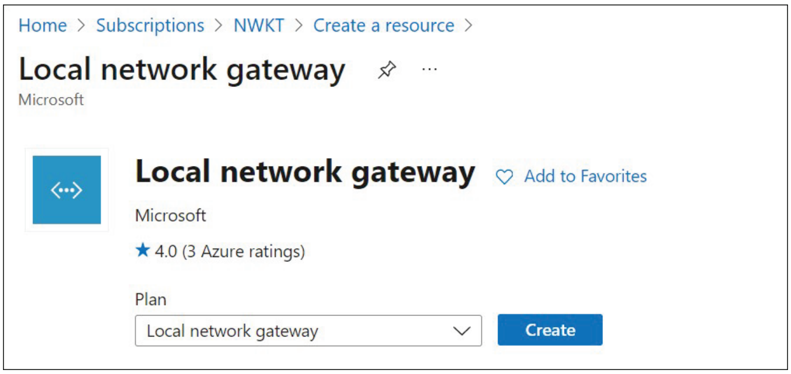


Figure 5-12: *Local Network Gateway.*

Select the *Create a resource* from the Azure portal Home view. Then type the *Local network gateway* on the search field. Next, select the Microsoft *Local network gateway* and click the *Create* button.


Home > Subscriptions > NWKT > Create a resource >

Local network gateway

Microsoft

Local network gateway   Add to Favorites

Microsoft

★ 4.0 (3 Azure ratings)

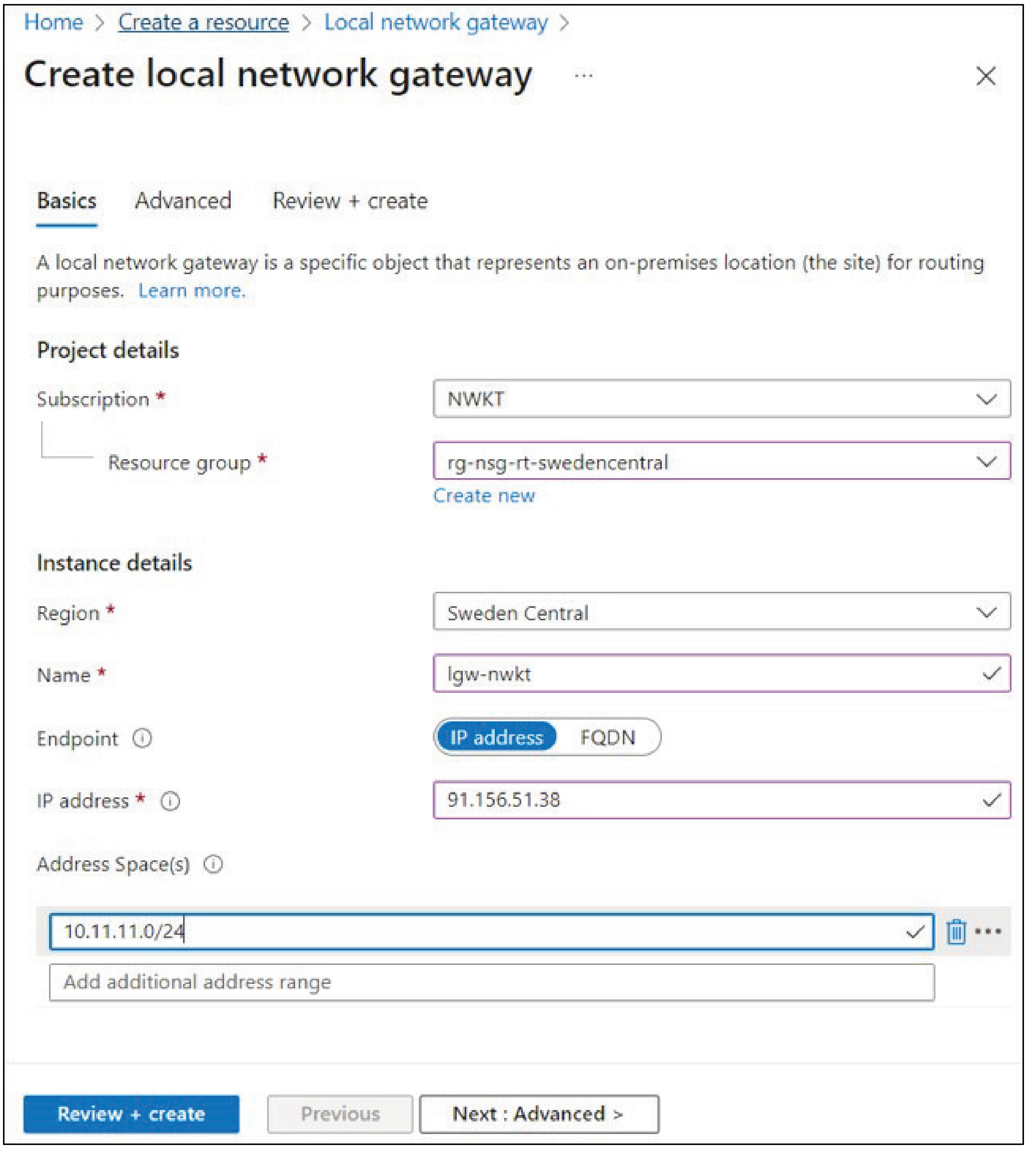
Plan

Local network gateway

Create

Figure 5-13: *Create Local Network Gateway – Step#1.*

Choose your Subscription and Resource group from the drop-down menus under the Project details. Then select Region and the name to LGW. Fill the IP address field with the LGW's Outside Global IP address. There is only subnet 10.11.11.0/24 on our remote site. Add it to IP address Space(s) field. Next, click the *Review + create* button or select the *Review + Create* tab.


Home > Create a resource > Local network gateway >
Create local network gateway
Basics
Advanced
Review + create

A local network gateway is a specific object that represents an on-premises location (the site) for routing purposes. Learn more.

Project details

Subscription *	NWKT
Resource group *	rg-nsg-rt-swedencentral

Instance details

Region *	Sweden Central
Name *	lgw-nwkt
Endpoint	[✓ IP address] FQDN
IP address *	91.156.51.38

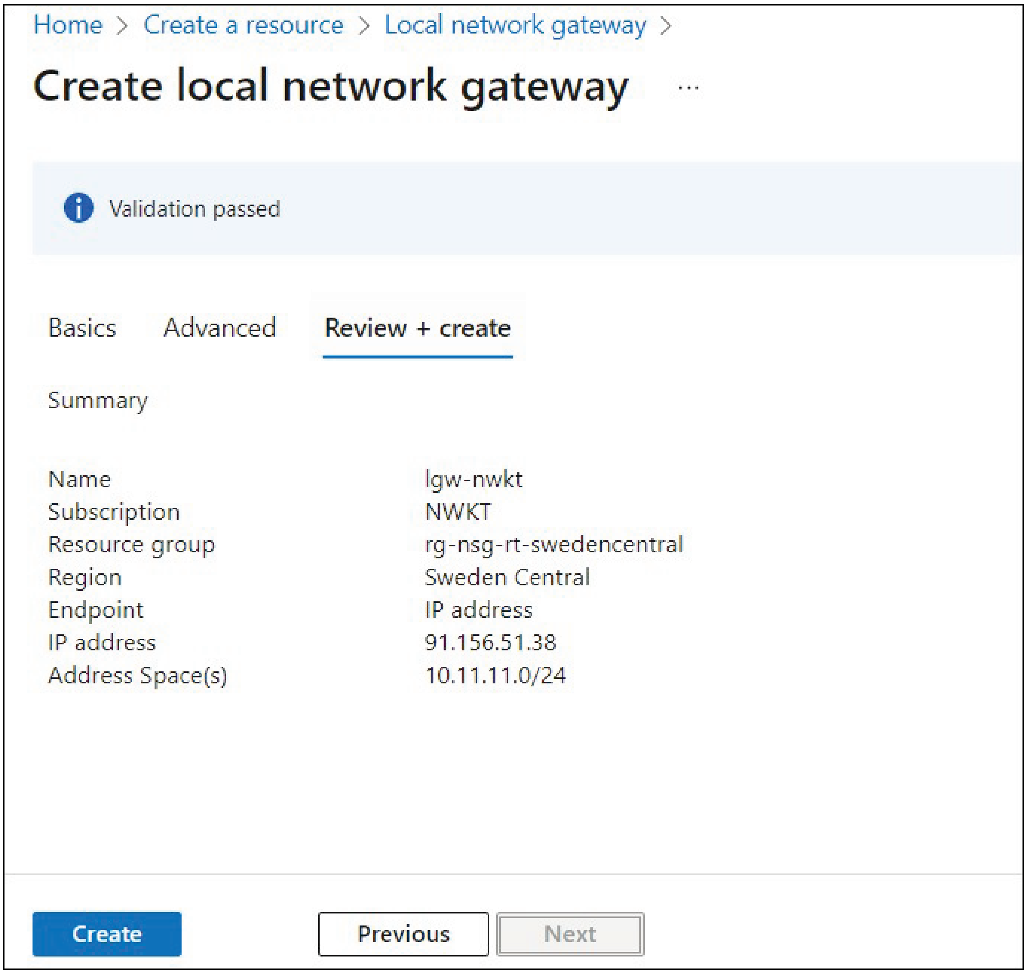
Address Space(s)

10.11.11.0/24

Add additional address range

Figure 5-14: *Create Local Network Gateway – Step#2.*

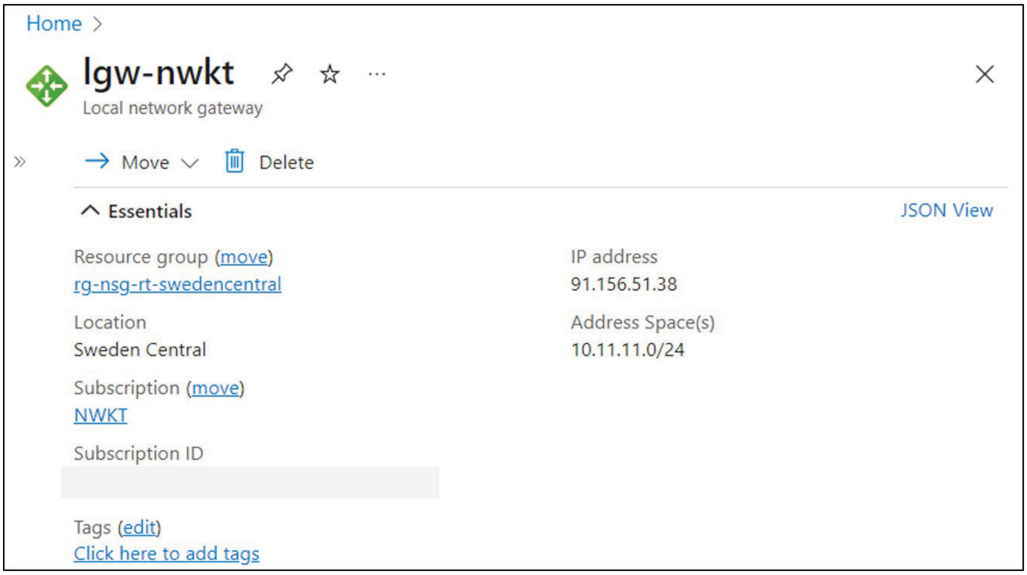
Figure 5-15 shows our LGW setting. Deploy the LGW by clicking the *Create* button. The LGW deployment process is faster than the VGW process because it only generates the configuration to VGW, where it describes the IPSec tunnel remote end IP address and the subnets behind the tunnel.


Home > Create a resource > Local network gateway >
Create local network gateway ...
Validation passed
Basics
Advanced
Review + create
Summary
Create
Previous
Next

Name	lgw-nwkt
Subscription	NWKT
Resource group	rg-nsg-rt-swedencentral
Region	Sweden Central
Endpoint	IP address
IP address	91.156.51.38
Address Space(s)	10.11.11.0/24

Figure 5-15: *Local Gateway – Review.*

Figure 5-16 shows the overview of our LGW lgw-nwkt.



Resource group (move)
rg-nsg-rt-swedencentral

Location
Sweden Central

Subscription (move)
NWKT

Subscription ID

Tags (edit)
Click here to add tags

IP address
91.156.51.38

Address Space(s)
10.11.11.0/24

Figure 5-16: *The Overview of LGW (lgw-nwkt).*

You can verify LGW's parameters and configuration using the Azure PowerShell command *Get-AzLocalNetworkGateway -name [name] -ResourceGroupName [name].*

```
Get-AzLocalNetworkGateway -name lgw-nwkt -ResourceGroupName rg-nsg-rt-swedencentral

Name                     : lgw-nwkt
ResourceGroupName        : rg-nsg-rt-swedencentral
Location                 : swedencentral
Id                       : /subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/localNetworkGateways/lgw-nwkt
Etag                     : W/"76117240-0270-4fe6-863a-8d932d904e43"
ResourceGuid             : 6e2700a1-8952-443c-bea7-6672a729dfb2
ProvisioningState        : Succeeded
Tags                     :
GatewayIpAddress         : 91.156.51.38
Fqdn                     :
LocalNetworkAddressSpace : {
                             "AddressPrefixes": [
                               "10.11.11.0/24"
                             ]
                           }
BgpSettings              : null
```

Example 5-2: *Verifying Local Gateway Settings with Azure PowerShell.*

Create VPN Connection

After deploying VGW and LGW, we create a Site-to-Site Connection resource, which binds VGW and LGW together and defines tunnel setup parameters and data encryption/integrity settings.

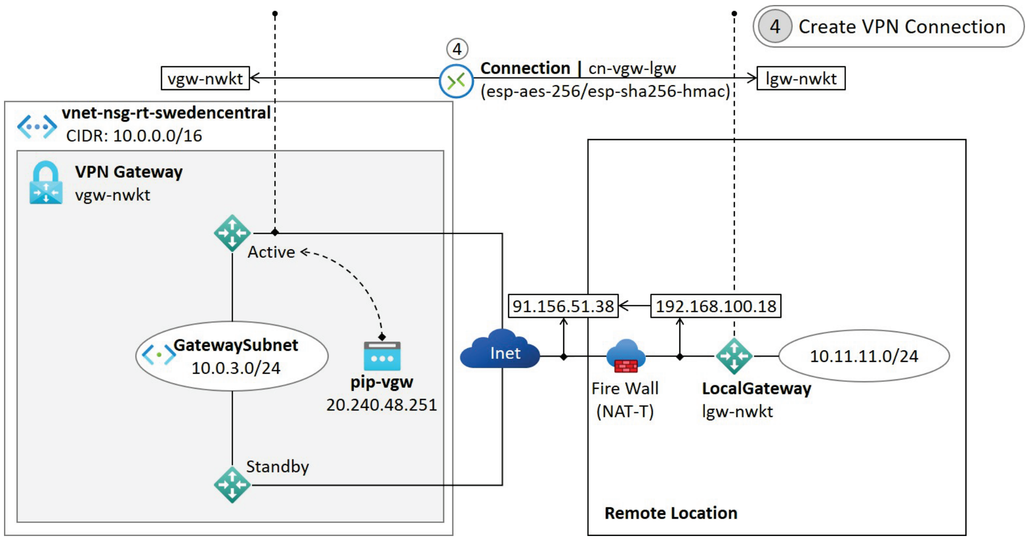


Figure 5-17: *VPN Connection.*

Select the *Create a resource* from the Azure portal Home view. Then type the *Connection* on the search field. Next, select the Microsoft *Connection* and click the *Create* button.

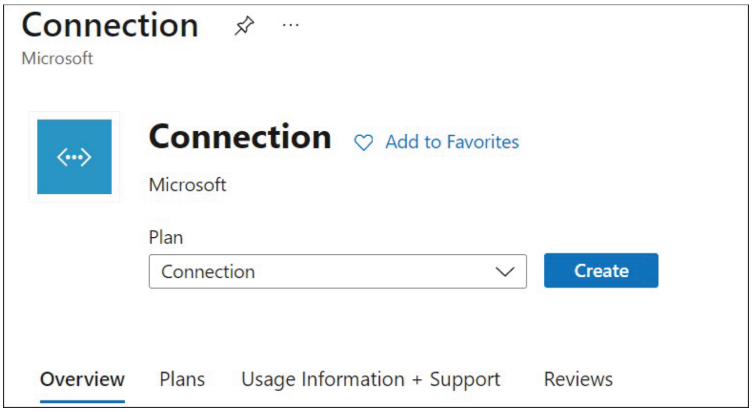
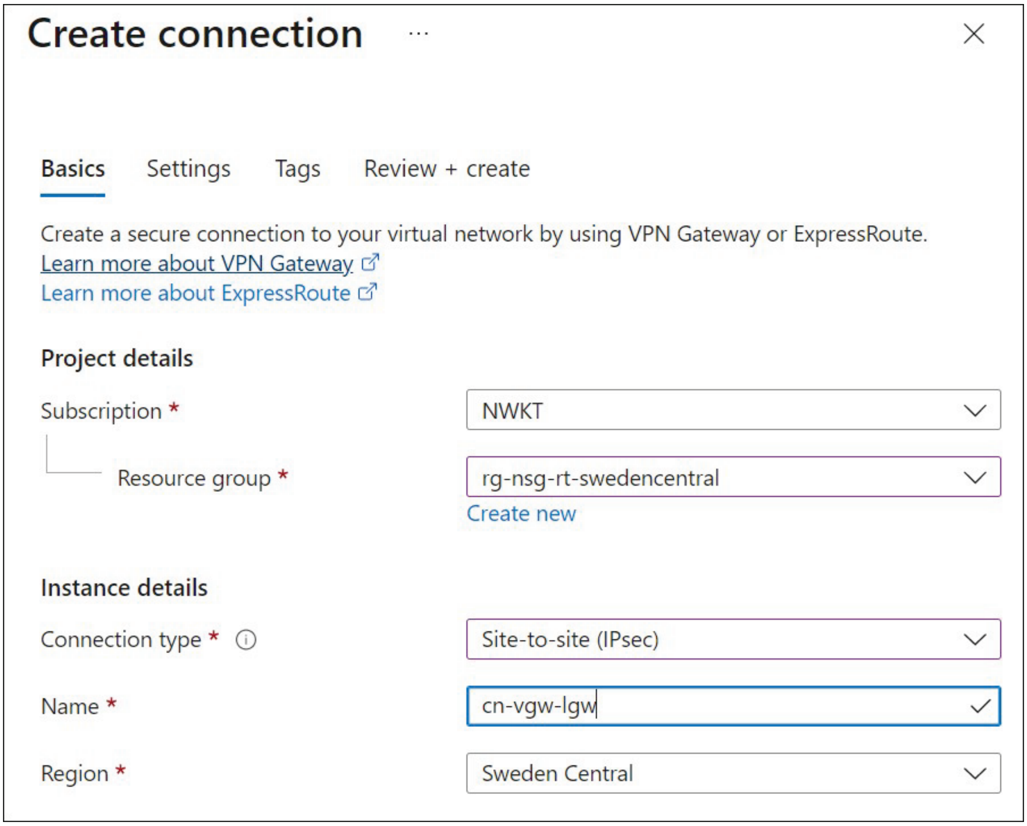


Figure 5-18: *Create Connection – Step#1.*

Select your Subscription and Resource group from the drop-down menus under the Project details section. Choose the *Site-to-Site (IPSec)* from the Connection type drop-down menu. Fill in the *Name* field and select the region from the drop-down menu. Select the *Settings* tab to proceed.


Create connection

Create a secure connection to your virtual network by using VPN Gateway or ExpressRoute.
Learn more about VPN Gateway
Learn more about ExpressRoute

Project details

Subscription * NWKT

Resource group * rg-nsg-rt-swedencentral

Create new

Instance details

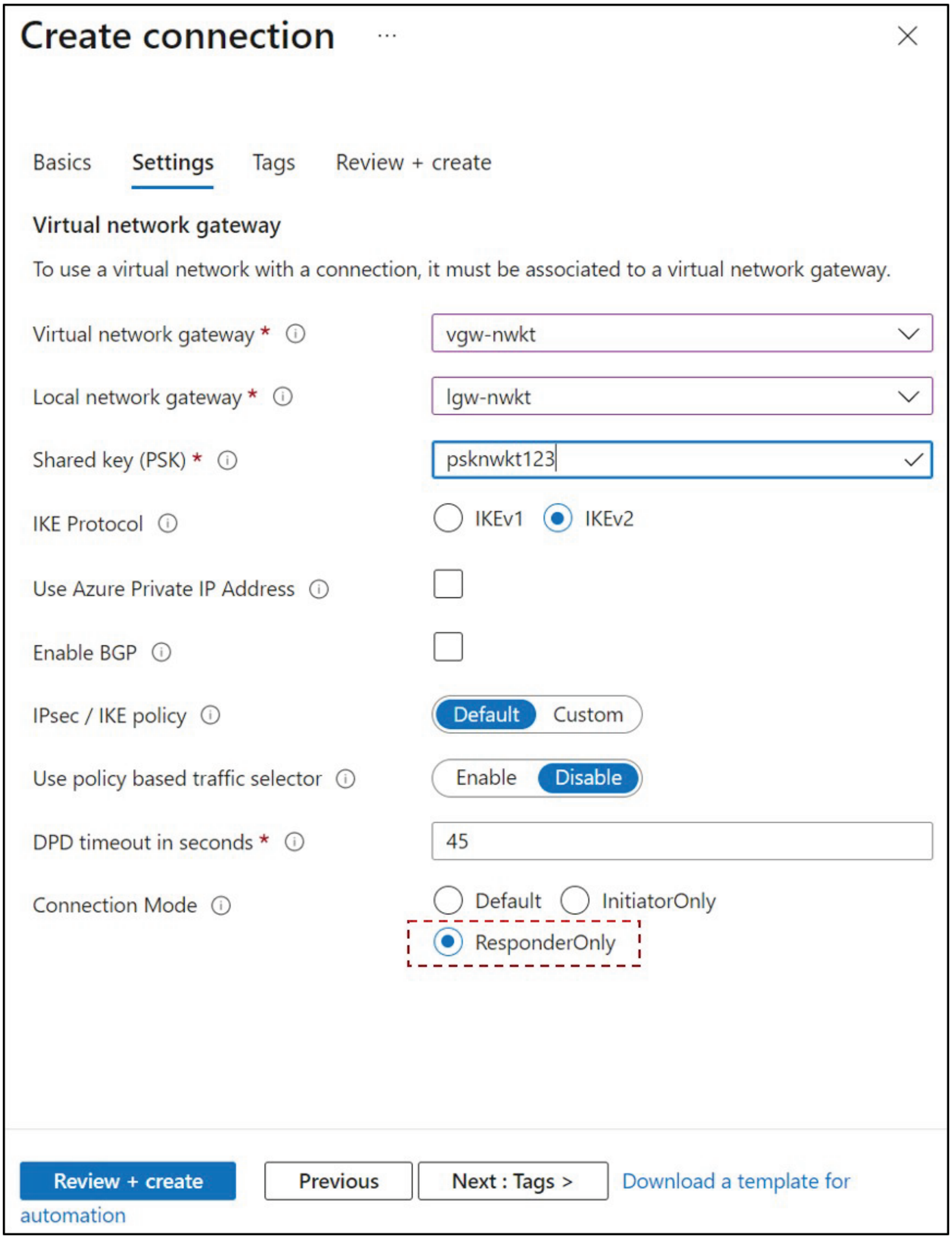
Connection type * Site-to-site (IPsec)

Name * cn-vgw-lgw

Region * Sweden Central

Figure 5-19: *Create Connection – Step#2: Project and Instance Details.*

Choose your VPN tunnel termination resources on the Azure side (vgw-nwkt) and Remote location (lgw-nwkt) from the drop-down menus. Give the Pre-Shared Key. Left the IKE protocol and IPSec/IKE Policy to their default values. Our LGW is behind the NAT Firewall, so we do not want the VGW to start tunnel negotiation. Select the *ResponderOnly* as the Connection mode. Next, click the *Review + create* button or select the *Review + create* tab. Note that we don't have to specify the VNet because we already did it with the VGW.


Create connection

Basics | Settings | Tags | Review + create

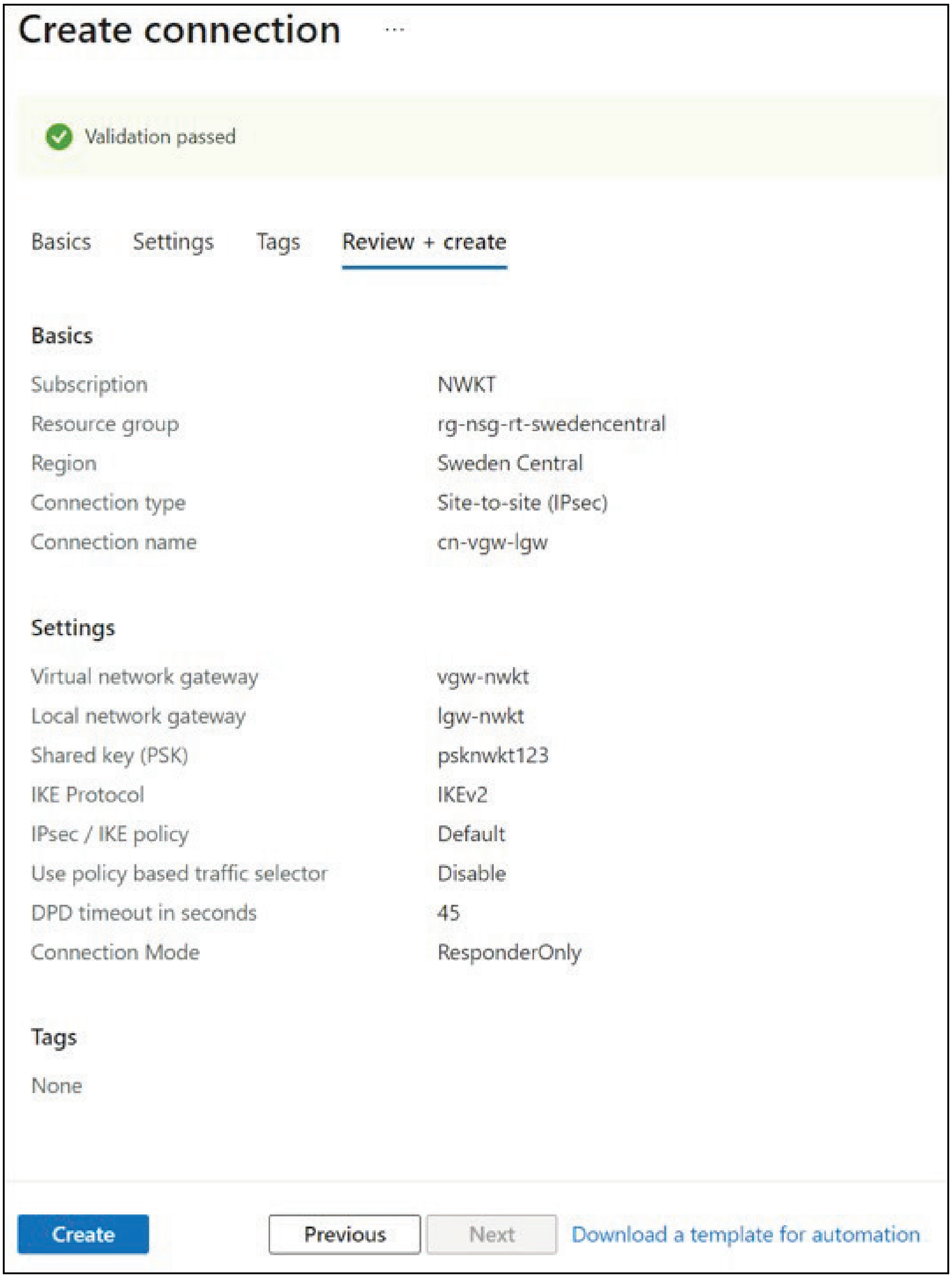
Virtual network gateway

To use a virtual network with a connection, it must be associated to a virtual network gateway.

Virtual network gateway *	vgw-nwkt
Local network gateway *	lgw-nwkt
Shared key (PSK) *	psknwkt123
IKE Protocol	IKEv1 [✓ IKEv2]
Use Azure Private IP Address	☐
Enable BGP	☐
IPsec / IKE policy	[✓ Default] Custom
Use policy based traffic selector	Enable [✓ Disable]
DPD timeout in seconds *	45
Connection Mode	Default InitiatorOnly [✓ ResponderOnly]

Figure 5-20: *Create Connection – Step#3: Project and Instance Details.*

Figure 5-21 shows the connection setting. Deploy the Connection resource by clicking the *Create* button.



Basics

Subscription	NWKT
Resource group	rg-nsg-rt-swedencentral
Region	Sweden Central
Connection type	Site-to-site (IPsec)
Connection name	cn-vgw-lgw

Settings

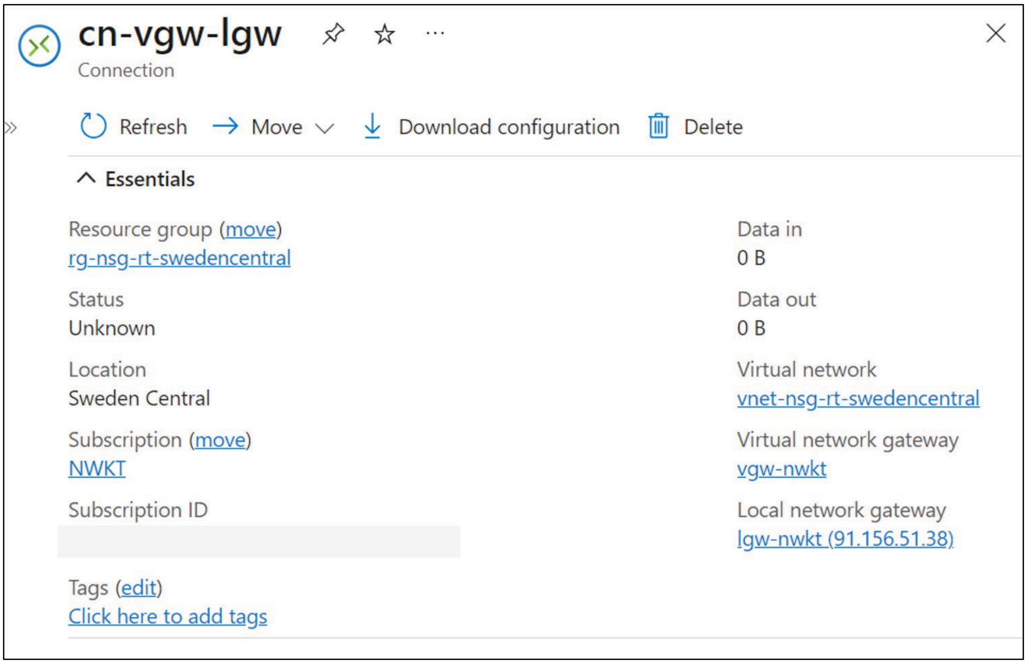
Virtual network gateway	vgw-nwkt
Local network gateway	lgw-nwkt
Shared key (PSK)	psknwkt123
IKE Protocol	IKEv2
IPsec / IKE policy	Default
Use policy based traffic selector	Disable
DPD timeout in seconds	45
Connection Mode	ResponderOnly

Tags

None

Figure 5-21: *Site-to-Site VPN Connection - Review.*

Figure 5-22 shows the overview of the connection cn-vgw-lgw.



cn-vgw-lgw
Connection

Refresh | Move | Download configuration | Delete

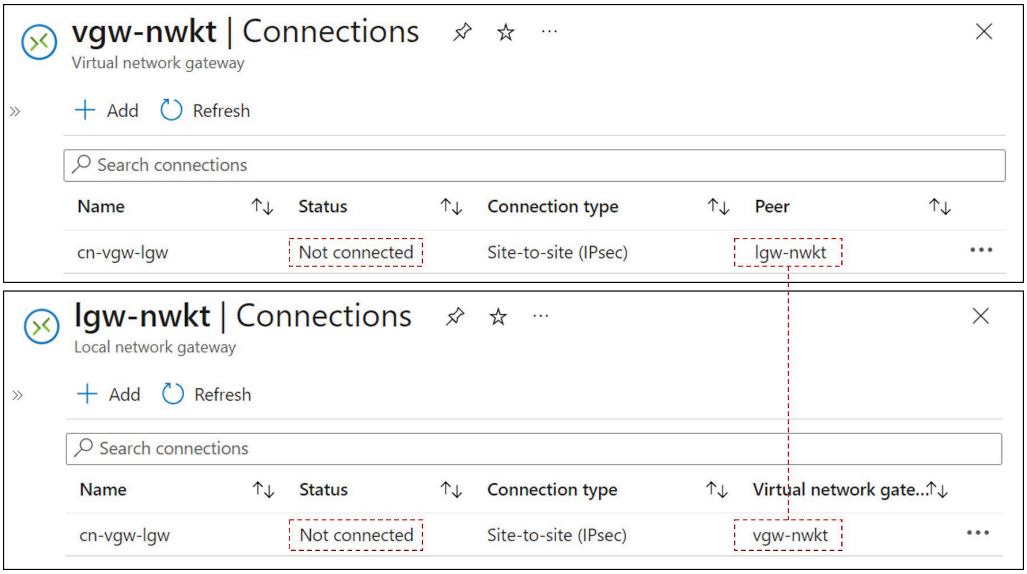
Essentials

Resource group (move)	Data in
rg-nsg-rt-swedencentral	0 B
Status	Data out
Unknown	0 B
Location	Virtual network
Sweden Central	vnet-nsg-rt-swedencentral
Subscription (move)	Virtual network gateway
NWKT	vgw-nwkt
Subscription ID	Local network gateway
	lgw-nwkt (91.156.51.38)

Tags (edit)
Click here to add tags

Figure 5-22: *The Overview of S2S Connection (cn-vgw-lgw).*

Figure 5-23 shows the connections of VGW *vgw-nwkt* and LGW *lgw-nwkt*. The status of the Site-to-Site connection is *Not connected* because we haven't yet configured the LGW.



vgw-nwkt | Connections
Virtual network gateway

Add | Refresh

Search connections

Name	Status	Connection type	Peer
cn-vgw-lgw	Not connected	Site-to-site (IPsec)	lgw-nwkt

lgw-nwkt | Connections
Local network gateway

Add | Refresh

Search connections

Name	Status	Connection type	Virtual network gate...
cn-vgw-lgw	Not connected	Site-to-site (IPsec)	vgw-nwkt

Figure 5-23: *Site-to-Site Connection Status.*

Configure Local Gateway

As the last step of setting up the Site-to-Site connection, we need to configure the remote VPN device. It is a two-step process. First, we need to download the configuration file and adjust some of its parameters. Then we copy the configuration file and paste it to the LGW.

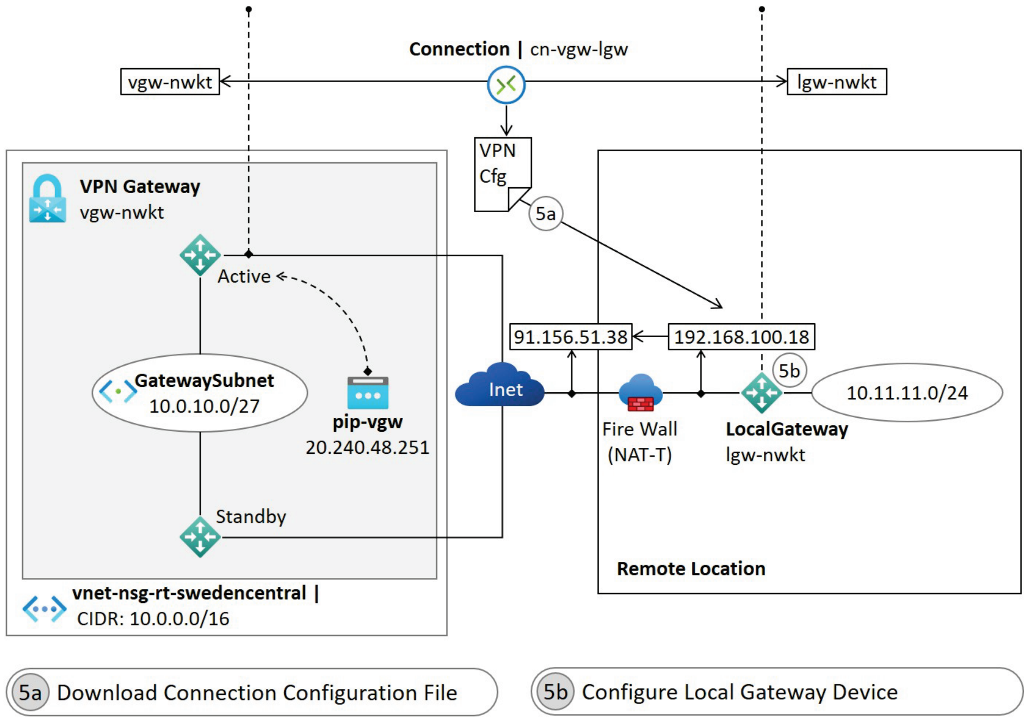
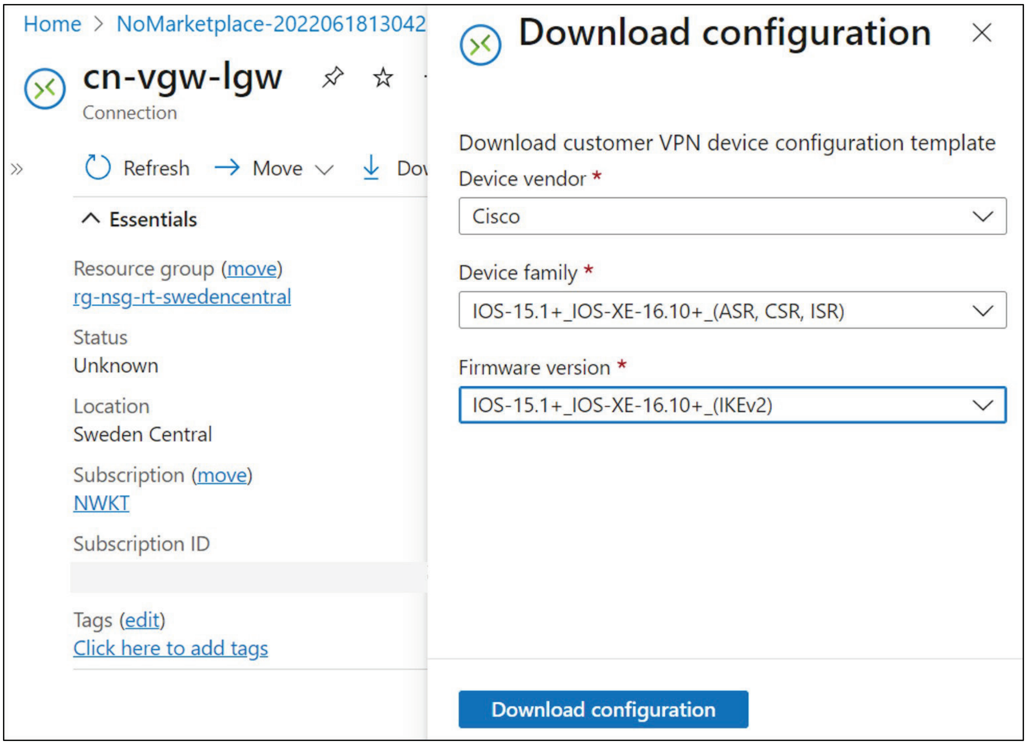


Figure 5-24: *Configuring Local Gateway.*

Download Configuration File

Open the Connection resource overview window and select the Download configuration hyperlink. Select the Device vendor and family as well as the Firmware version. I am using Cisco CSR1000V (IOS-XE 16.06.07)


Home > NoMarketplace-2022061813042

cn-vgw-lgw
Connection

Refresh Move Dov

Essentials

Resource group (move)
rg-nsg-rt-swedencentral

Status
Unknown

Location
Sweden Central

Subscription (move)
NWKT

Subscription ID

Tags (edit)
Click here to add tags

Download configuration

Download customer VPN device configuration template

Device vendor *
Cisco

Device family *
IOS-15.1+_IOS-XE-16.10+_(ASR, CSR, ISR)

Firmware version *
IOS-15.1+_IOS-XE-16.10+_(IKEv2)

Download configuration

Figure 5-25: *Download the Configuration File.*

Example 5-3 on the next three pages shows the complete configuration file for our S2S connection. It has several sections. First, there is instruction on what parameters you should change. For example, you may need to modify an Access-List (ACL) number if your device configuration has an existing ACL with the same number. File also describes the Device information [0], Network parameters [1], IPSec/IKE parameters [2], and BGP parameters [3]. It also shows the ACL, which might be needed for permitting the S2S IPSec tunnel. After ACL instructions, there is the actual device configuration. In our example, we have to change the public IP address 91.156.51.38 LGW to Inside Local private IP address 192.168.100.18 (ike policy, ike profile, and the source IP address of tunnel 11).

```
! Microsoft Corporation
! ---------------------------------------------------------------------------------
! Sample VPN tunnel configuration template for Cisco IOS-based devices [IOS 15.1 or
beyond / IOS-XE 16.10 or beyond]
!
! #################################################################################
! !!! Search for "REPLACE" to find the values that require special
! !!! considerations
! !!!
! !!! (1) ACL/access-list rule numbers
! !!! (2) Tunnel interface number
! !!! (3) Tunnel interface IP address
! !!! (4) BGP routes to advertise (if BGP is enabled)
! !!! (5) BGP peer IP address on the device - loopback interface number
! #################################################################################
!
! [0] Device infomration
!
!   > Device vendor:    Cisco
!   > Device family:    IOS-based (ASR, ISR, CSR)
!   > Firmware version: IOS 15.1 or beyond / IOS-XE 16.10 or beyond
!   > Test platform:    Cisco ISR 2911, version 15.2
!
! [1] Network parameters
!
!   > Connection name:       cn-vgw-lgw
!   > VPN Gateway name:      69fd0046-e6a9-430e-8998-000b505a75f1
!   > Public IP addresses:
!     + Public IP 1:         20.240.48.251
!   > Virtual network address space:
!     + CIDR:10.0.0.0/16, prefix:10.0.0.0, netmask:255.255.0.0, wildcard:0.0.255.255
!   > Local network gateway: lgw-nwkt
!   > On-premises VPN IP:    91.156.51.38
!   > On-premises address prefixes:
!     + CIDR:10.11.11.0/24, prefix:10.11.11.0, netmask:255.255.255.0,
wildcard:0.0.0.255
!
! [2] IPsec/IKE parameters
!
!   > IKE version:             IKEv2
!     + Encryption algorithm:  aes-cbc-256
!     + Integrityalgorithm:    sha1
!     + Diffie-Hellman group:  2
!     + SA lifetime (seconds): 3600
!     + Pre-shared key:        psknwkt123
!     + UsePolicyBasedTS:      False
!
!   > IPsec
!     + Encryption algorithm:  esp-aes 256
!     + Integrity algorithm:   esp-sha256-hmac
!     + PFS Group:             None
!     + SA lifetime (seconds): 3600
!     + SA lifetime (KB):      102400000
!
! [3] BGP parameters - Azure VPN gateway
!
!   > Azure virtual network
!     + Enable BGP:            False
!     + Azure BGP ASN:         VNG_ASN
!   > On-premises network / LNG
!     + On premises BGP ASN:   LNG_ASN
!     + On premises BGP IP:    LNG_BGPIP
```

```
!
! ==============================================================================
! Cisco IOS 15.x+ / IOS-XE 16.10+ IKEv2, route-based (any-to-any)
! ==============================================================================
!
! ACL rules
!
! Some VPN devices require explicit ACL rules to allow cross-premises traffic:
!
! 1. Allow traffic between on premises address ranges and VNet address ranges
! 2. Allow IKE traffic (UDP:500) between on premises VPN devices and Azure VPN gateway
! 3. Allow IPsec traffic (Proto:ESP) between on premises VPN devices and Azure VPN
gateway
! [REPLACE] access-list number: access-list 101

access-list 101 permit ip 10.11.11.0 0.0.0.255 10.0.0.0 0.0.255.255
access-list 101 permit esp host 20.240.48.251 host 91.156.51.38
access-list 101 permit udp host 20.240.48.251 eq isakmp host 91.156.51.38
access-list 101 permit udp host 20.240.48.251 eq non500-isakmp host 91.156.51.38

! ==============================================================================
! Internet Key Exchange (IKE) configuration
! - IKE Phase 1 / Main mode configuration
! - Encryption/integrity algorithms, Diffie-Hellman group, pre-shared key

crypto ikev2 proposal Azure-Ikev2-Proposal
  encryption aes-cbc-256
  integrity sha1
  group 2
  exit

crypto ikev2 policy Azure-Ikev2-Policy
  proposal Azure-Ikev2-Proposal
  match address local 91.156.51.38
  exit

crypto ikev2 keyring cn-vgw-lgw-keyring
  peer 20.240.48.251
    address 20.240.48.251
    pre-shared-key psknwkt123
    exit
  exit

crypto ikev2 profile Azure-Ikev2-Profile
  match address local 91.156.51.38
  match identity remote address 20.240.48.251 255.255.255.255
  authentication remote pre-share
  authentication local pre-share
  lifetime 28800
  dpd 10 5 on-demand
  keyring local cn-vgw-lgw-keyring
  exit

! ------------------------------------------------------------------------------
! IPsec configuration
! - IPsec (or IKE Phase 2 / Quick Mode) configuration
! - Transform Set: IPsec encryption/integrity algorithms, IPsec ESP mode

crypto ipsec transform-set Azure-TransformSet esp-aes 256 esp-sha256-hmac
  mode tunnel
  exit
```

```
crypto ipsec profile Azure-IPsecProfile
  set transform-set Azure-TransformSet
  set ikev2-profile Azure-Ikev2-Profile
  set security-association lifetime seconds 3600
  ! Note: PFS (perfect-forward-secrecy) is an optional feature (commented out)
  !set pfs None
  exit

! ---------------------------------------------------------------------------------
! Tunnel interface (VTI) configuration
! - Create/configure a tunnel interface
! - Configure an APIPA (169.254.x.x) address that does NOT overlap with any
!   other address on this device. This is not visible from the Azure gateway.
! * REPLACE: Tunnel interface numbers and APIPA IP addresses below
! * Default tunnel interface 11 (169.254.0.1) and 12 (169.254.0.2)

int tunnel 11
  ip address 169.254.0.1 255.255.255.255
  tunnel mode ipsec ipv4
  ip tcp adjust-mss 1350
  tunnel source 91.156.51.38
  tunnel destination 20.240.48.251
  tunnel protection ipsec profile Azure-IPsecProfile
  exit


! ---------------------------------------------------------------------------------
! Static routes
! - Adding the static routes to point the VNet prefixes to the IPsec tunnels
! * REPLACE: Tunnel interface number(s), default tunnel 11 and tunnel 12

ip route 10.0.0.0 255.255.0.0 Tunnel 11

! =================================================================================
! Cleanup script
! =================================================================================
!
! [WARNING] This section of the script will cleanup the resources: IPsec/IKE,
! [WARNING] interfaces, routes, access-list. Validate the objects in your
! [WARNING] configuration before applying the script below.
! [REPLACE] Interfaces: Loopback 11, Tunnel 11, Tunnel 12; access-list 101
!
!!
!! no ip route 10.0.0.0 255.255.0.0 Tunnel 11
!!
!!
!! no int tunnel 11
!!
!! no crypto ipsec profile Azure-IPsecProfile
!! no crypto ipsec transform-set Azure-TransformSet
!!
!! no crypto ikev2 profile Azure-Ikev2-Profile
!! no crypto ikev2 keyring cn-vgw-lgw-keyring
!! no crypto ikev2 policy Azure-Ikev2-Policy
!! no crypto ikev2 proposal Azure-Ikev2-Proposal
!!
!! no access-list 101 permit ip 10.11.11.0 0.0.0.255 10.0.0.0 0.0.255.255
!! no access-list 101 permit esp host 20.240.48.251 host 91.156.51.38
!! no access-list 101 permit udp host 20.240.48.251 eq isakmp host 91.156.51.38
!! no access-list 101 permit udp host 20.240.48.251 eq non500-isakmp host 91.156.51.38
```

Example 5-3: *The Configuration File.*

Configure Local Gateway

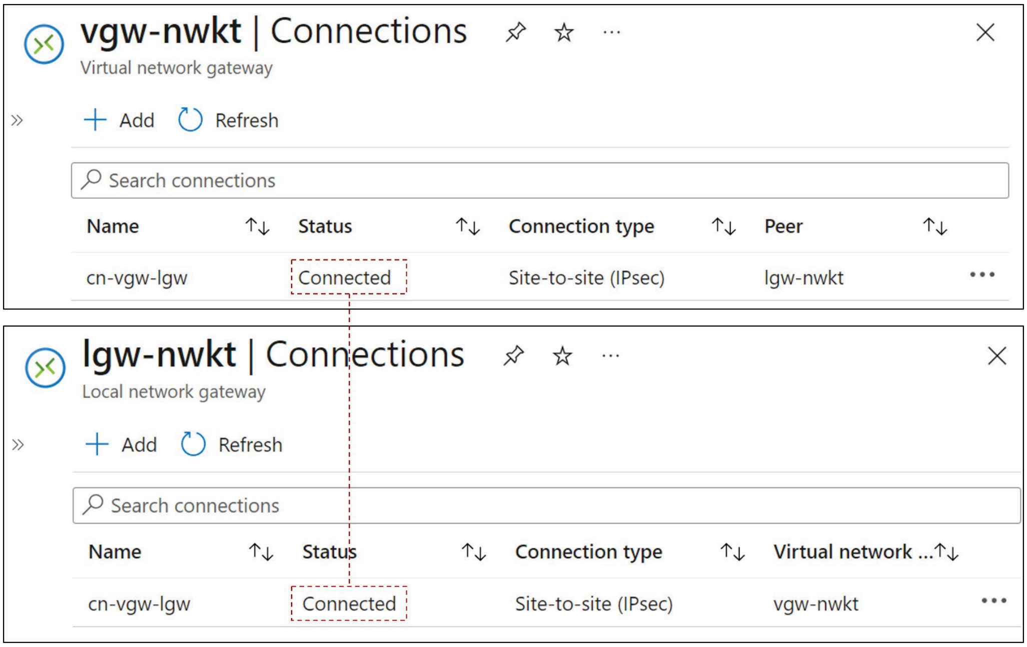
Example 5-4 shows the modified configuration file. I have removed all instructions, comments, and ACL lines. I have also done IP address changes.

```
crypto ikev2 proposal Azure-Ikev2-Proposal
  encryption aes-cbc-256
  integrity sha1
  group 2
  exit

crypto ikev2 policy Azure-Ikev2-Policy
  proposal Azure-Ikev2-Proposal
  match address local 192.168.100.18
  exit

crypto ikev2 keyring cn-vgw-lgw-keyring
  peer 20.240.48.251
    address 20.240.48.251
    pre-shared-key psknwkt123
    exit
  exit

crypto ikev2 profile Azure-Ikev2-Profile
  match address local 192.168.100.18
  match identity remote address 20.240.48.251 255.255.255.255
  authentication remote pre-share
  authentication local pre-share
  lifetime 28800
  dpd 10 5 on-demand
  keyring local cn-vgw-lgw-keyring
  exit

crypto ipsec transform-set Azure-TransformSet esp-aes 256 esp-sha256-hmac
  mode tunnel
  exit

crypto ipsec profile Azure-IPsecProfile
  set transform-set Azure-TransformSet
  set ikev2-profile Azure-Ikev2-Profile
  set security-association lifetime seconds 3600
 exit

int tunnel 11
  ip address 169.254.0.1 255.255.255.255
  tunnel mode ipsec ipv4
  ip tcp adjust-mss 1350
  tunnel source 192.168.100.18
  tunnel destination 20.240.48.251
  tunnel protection ipsec profile Azure-IPsecProfile
  exit

ip route 10.0.0.0 255.255.0.0 Tunnel 11
```

Example 5-4: *The Configuration File of CSR1000v.*

Verification

Figure 5-26 shows that after we have implemented the tunnel configuration to CSR1000v (lgw-nwkt), the status of the connections is now *Connected*.



Name	Status	Connection type	Peer
cn-vgw-lgw	Connected	Site-to-site (IPsec)	lgw-nwkt

Name	Status	Connection type	Virtual network ...
cn-vgw-lgw	Connected	Site-to-site (IPsec)	vgw-nwkt

Figure 5-26: *Site-to-Site Connection Status.*

Example 5-5 on next page shows how you can request connection status using Azure PowerShell command: *az network vpn-connection show --resource-group [name] --name [name]*.

```
az network vpn-connection show --resource-group rg-nsg-rt-swedencentral --
name cn-vgw-lgw
{
  "authorizationKey": null,
  "connectionMode": "ResponderOnly",
  "connectionProtocol": "IKEv2",
  "connectionStatus": "Connected",
  "connectionType": "IPsec",
  "dpdTimeoutSeconds": 45,
  "egressBytesTransferred": 2920,
  "egressNatRules": null,
  "enableBgp": false,
  "etag": "W/\"c805e424-9630-432f-8b38-30f437bd530e\"",
  "expressRouteGatewayBypass": false,
  "gatewayCustomBgpIpAddresses": [],
  "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/connections/cn-vgw-lgw",
  "ingressBytesTransferred": 5440,
  "ingressNatRules": null,
  "ipsecPolicies": [],
  "localNetworkGateway2": {
    "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/localNetworkGateways/lgw-nwkt",
    "resourceGroup": "rg-nsg-rt-swedencentral"
  },
  "location": "swedencentral",
  "name": "cn-vgw-lgw",
  "provisioningState": "Succeeded",
  "resourceGroup": "rg-nsg-rt-swedencentral",
  "resourceGuid": "56d9dcd3-d718-4c68-8ce7-2a139868a9ba",
  "routingWeight": 0,
  "sharedKey": "psknwkt123",
  "tags": {},
  "trafficSelectorPolicies": [],
  "tunnelConnectionStatus": null,
  "type": "Microsoft.Network/connections",
  "useLocalAzureIpAddress": false,
  "usePolicyBasedTrafficSelectors": false,
  "virtualNetworkGateway1": {
    "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworkGateways/vgw-nwkt",
    "resourceGroup": "rg-nsg-rt-swedencentral"
  }
}
```

Example 5-5: *Site-to-Site Tunnel Verification – Azure PowerShell.*

Example below shows that the tunnel 11 interface is up and protected.

```
NWKT-WAN-Edge1#show interfaces tunnel 11
Tunnel11 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 169.254.0.1/32
  MTU 9922 bytes, BW 100 Kbit/sec, DLY 50000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel linestate evaluation up
  Tunnel source 192.168.100.18, destination 20.240.48.251
  Tunnel protocol/transport IPSEC/IP
  Tunnel TTL 255
  Tunnel transport MTU 1422 bytes
  Tunnel transmit bandwidth 8000 (kbps)
  Tunnel receive bandwidth 8000 (kbps)
  Tunnel protection via IPSec (profile "Azure-IPsecProfile")
  Last input never, output never, output hang never
  Last clearing of "show interface" counters 00:20:53
  Input queue: 0/375/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/0 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
```

Example 5-6: *Tunnel Verification on LGW CSR1000v.*

Data Plane Testing

As the last verification, we test the data plane connection pinging between vm-front-1 and the on-prem server.

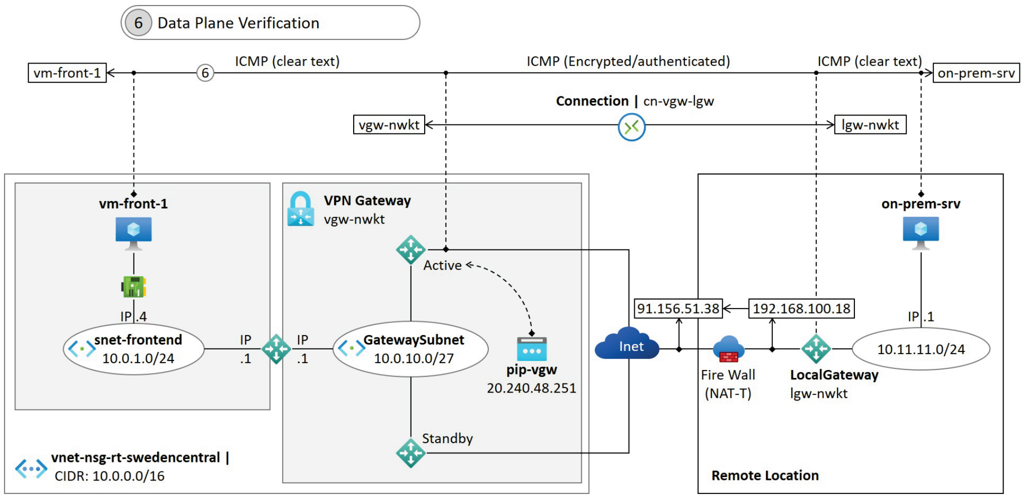


Figure 5-27: *Data Plane Testing.*

```
NWKT-WAN-Edge1#ping 10.0.1.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.1.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 13/13/14 ms
```

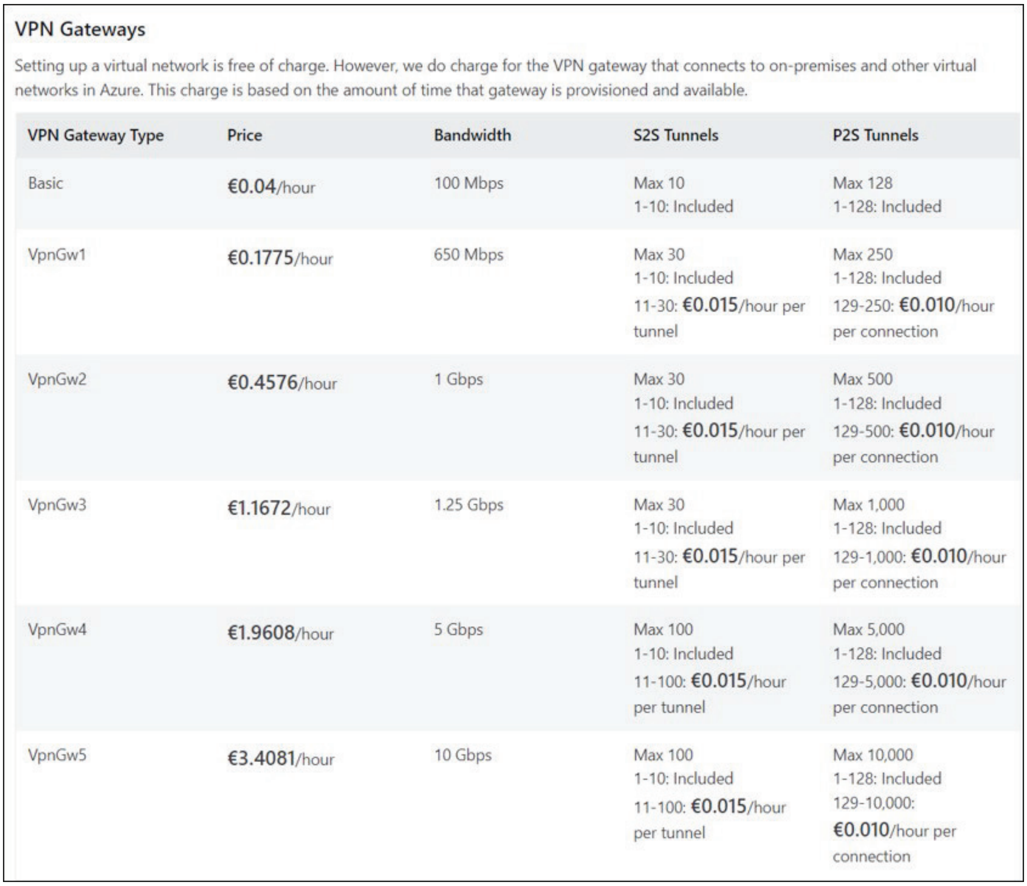
Example 5-7: *Ping from on-prem-srv to vm-front-1.*

```
nwktAdmin@vm-Front-1:~$ ping 10.11.11.1
PING 10.11.11.1 (10.11.11.1) 56(84) bytes of data.
64 bytes from 10.11.11.1: icmp_seq=1 ttl=255 time=30.5 ms
64 bytes from 10.11.11.1: icmp_seq=2 ttl=255 time=13.8 ms
64 bytes from 10.11.11.1: icmp_seq=3 ttl=255 time=13.3 ms
64 bytes from 10.11.11.1: icmp_seq=4 ttl=255 time=13.1 ms
64 bytes from 10.11.11.1: icmp_seq=5 ttl=255 time=13.2 ms
^C
--- 10.11.11.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 13.103/16.779/30.548/6.888 ms
```

Example 5-8: *Ping from vm-front-1 to on-prem-srv.*

Pricing

Figure 5-28 shows the hourly based pricing of Azure VGW. In addition to VGW pricing, you will also be charged based on data transfer.



VPN Gateways

Setting up a virtual network is free of charge. However, we do charge for the VPN gateway that connects to on-premises and other virtual networks in Azure. This charge is based on the amount of time that gateway is provisioned and available.

VPN Gateway Type	Price	Bandwidth	S2S Tunnels	P2S Tunnels
Basic	**€0.04**/hour	100 Mbps	Max 10 1-10: Included	Max 128 1-128: Included
VpnGw1	**€0.1775**/hour	650 Mbps	Max 30 1-10: Included 11-30: **€0.015**/hour per tunnel	Max 250 1-128: Included 129-250: **€0.010**/hour per connection
VpnGw2	**€0.4576**/hour	1 Gbps	Max 30 1-10: Included 11-30: **€0.015**/hour per tunnel	Max 500 1-128: Included 129-500: **€0.010**/hour per connection
VpnGw3	**€1.1672**/hour	1.25 Gbps	Max 30 1-10: Included 11-30: **€0.015**/hour per tunnel	Max 1,000 1-128: Included 129-1,000: **€0.010**/hour per connection
VpnGw4	**€1.9608**/hour	5 Gbps	Max 100 1-10: Included 11-100: **€0.015**/hour per tunnel	Max 5,000 1-128: Included 129-5,000: **€0.010**/hour per connection
VpnGw5	**€3.4081**/hour	10 Gbps	Max 100 1-10: Included 11-100: **€0.015**/hour per tunnel	Max 10,000 1-128: Included 129-10,000: **€0.010**/hour per connection

Figure 5-28: *VPN Gateway Pricing.*

References

[1] VPN Gateway design
https://docs.microsoft.com/en-us/azure/vpn-gateway/design

[2] Create a virtual network with a Site-to-Site VPN connection using CLI
https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-site-to-site-resource-manager-cli

[3] About VPN Gateway configuration settings
https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpn-gateway-settings

[4] Tutorial: Create a site-to-site VPN connection in the Azure portal
https://docs.microsoft.com/en-us/azure/vpn-gateway/tutorial-site-to-site-portal

[5] What is VPN Gateway?
https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpngateways

[6] S. Kelly et al., “Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec”, RFC 4868, May 2007.

[7] C. Kaufman et al., “Internet Key Exchange Protocol Version 2 (IKEv2)”, RFC 7296, October 2014.

[8] VPN Gateway pricing
https://azure.microsoft.com/en-us/pricing/details/vpn-gateway/

[9] Data transferred out of Azure data centers
https://azure.microsoft.com/en-us/pricing/details/bandwidth/

Chapter 6: Hybrid Cloud – Site-to-Site VPN with BGP

Introduction

This chapter shows how to enable dynamic routing using BGP between on-prem DC and Azure VNet. We are still using the Active-Standby Virtual Network Gateway (VGW) solution for simplicity. First, we enable BGP on Virtual Network Gateway (1). Azure assigns an IP address from the GatewaySubnet for BGP peering if we don't define an APIPA (Automatic Private IP Addressing) address. In our example, the IP address for BGP peering is 10.0.3.254. The default Azure BGP Autonomous System Number (ASN) is 65515. Then we activate BGP on Local Network Gateway (2). During the LGW configuration, we define the IP address for BGP peering on the LGW side. In our example, we are using the IP address 192.168.10.1. Besides, we describe the BGP ASN 64777 on the LGW side. After VGW and LGW configuration, we enable BGP on VPN Connection (3) and download the new LGW configuration file (4), which is then implemented with minor adjustments (because our LGW is behind the NAT device). As the last step, we verify the Control Plane and Data Plane operation.

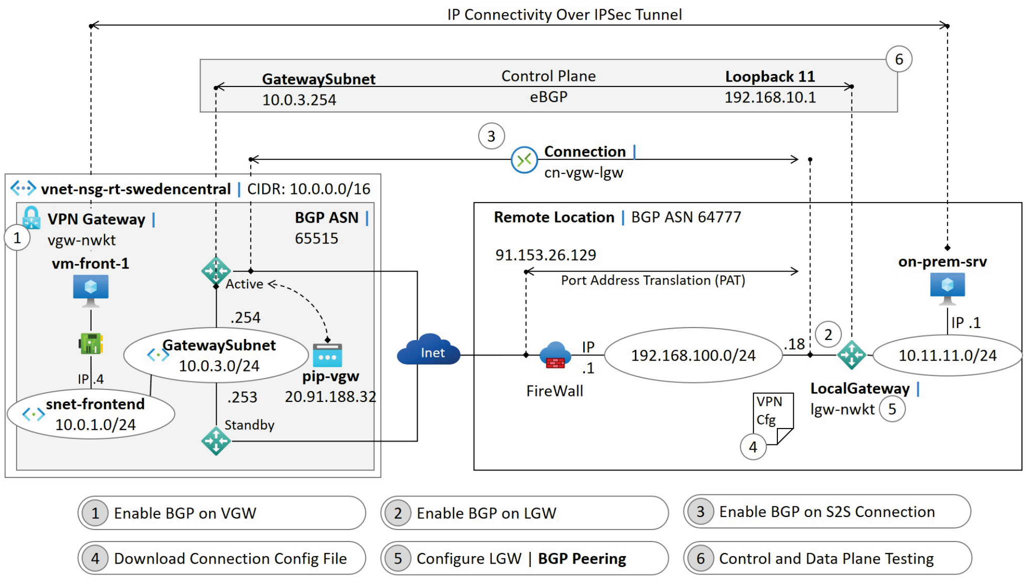
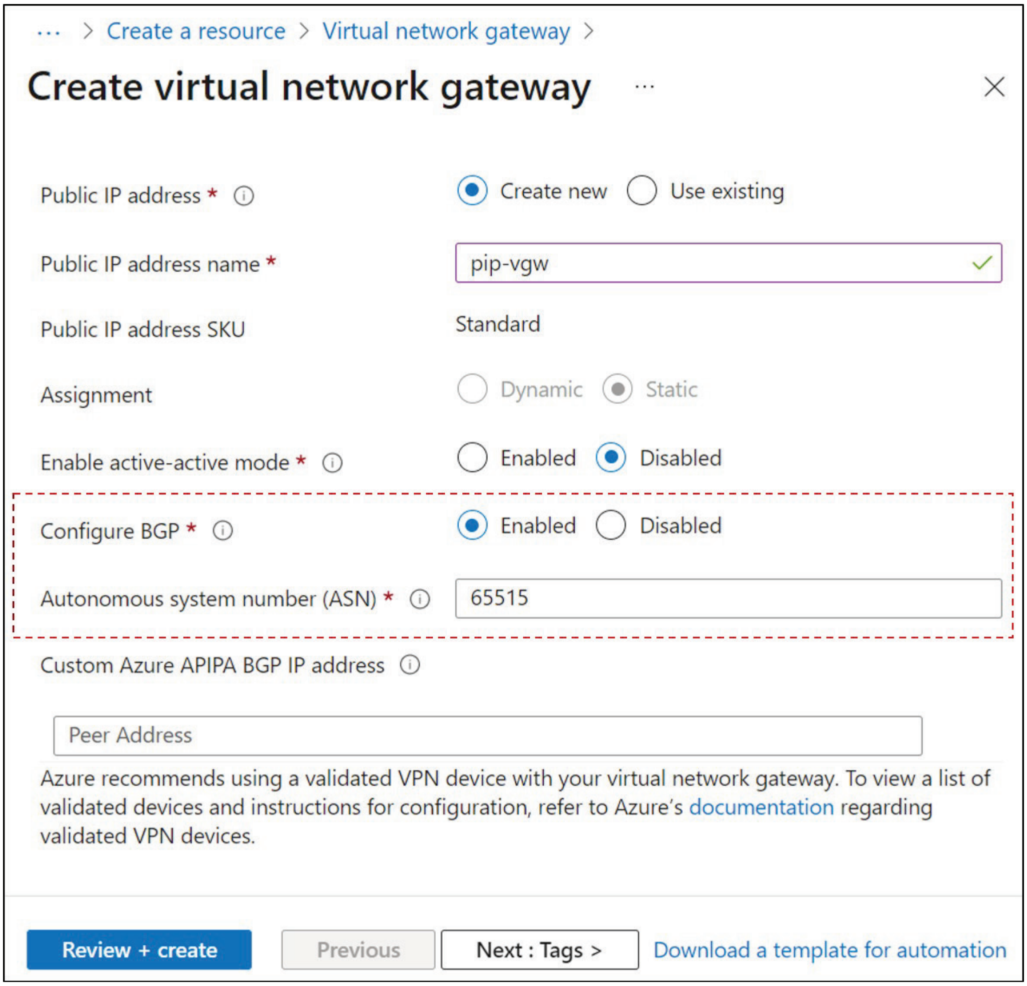


Figure 6-1: *Dynamic Routing Between Site-to-Site VPN with BGP.*

Enable BGP on VGW

Figure 6-2 shows that activating dynamic routing on VGW requires only enabling the BGP and giving the ASN number. If your on-prem VGW is using an APIPA IP address (169.254.0.0/16), you need to define the custom BGP peering IP from the APIPA address range on VGW too. Note that VGW doesn't start BGP peering initiation if the BGP peering use APIPA addresses. If you define the custom APIPA IP address on VGW and LGW uses a non-APIPA IP address, VGW start automatically using the IP private IP address from the GatewaySubnet address space.



Create virtual network gateway

Public IP address * [✓ Create new] Use existing

Public IP address name * pip-vgw

Public IP address SKU Standard

Assignment Dynamic [✓ Static]

Enable active-active mode * Enabled [✓ Disabled]

Configure BGP * [✓ Enabled] Disabled

Autonomous system number (ASN) * 65515

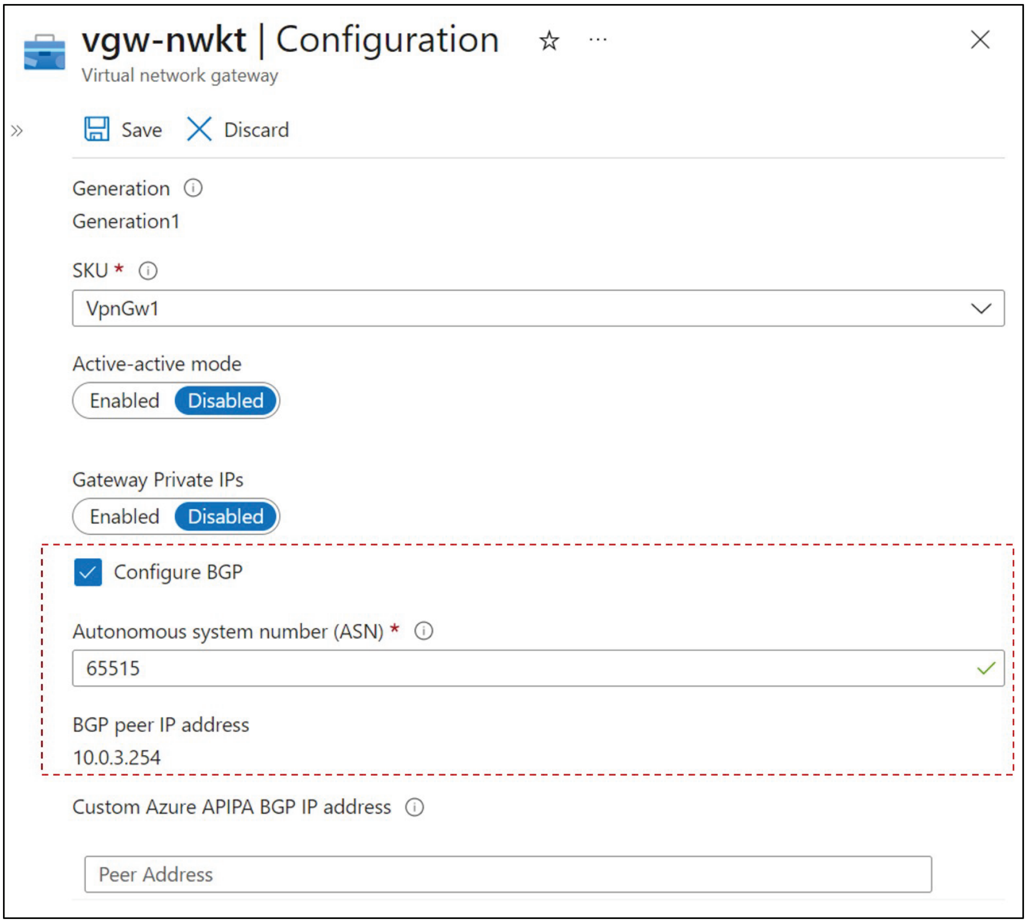
Custom Azure APIPA BGP IP address

Peer Address

Azure recommends using a validated VPN device with your virtual network gateway. To view a list of validated devices and instructions for configuration, refer to Azure's documentation regarding validated VPN devices.

Figure 6-2: *Enabling BGP on VGW.*

Figure 6-3 shows that the Azure has assigned IP address 10.0.3.254 for BGP peering.


vgw-nwkt | Configuration
Virtual network gateway

Generation
Generation1

SKU *
VpnGw1

Active-active mode
Enabled [✓ Disabled]

Gateway Private IPs
Enabled [✓ Disabled]

✓ Configure BGP

Autonomous system number (ASN) *
65515

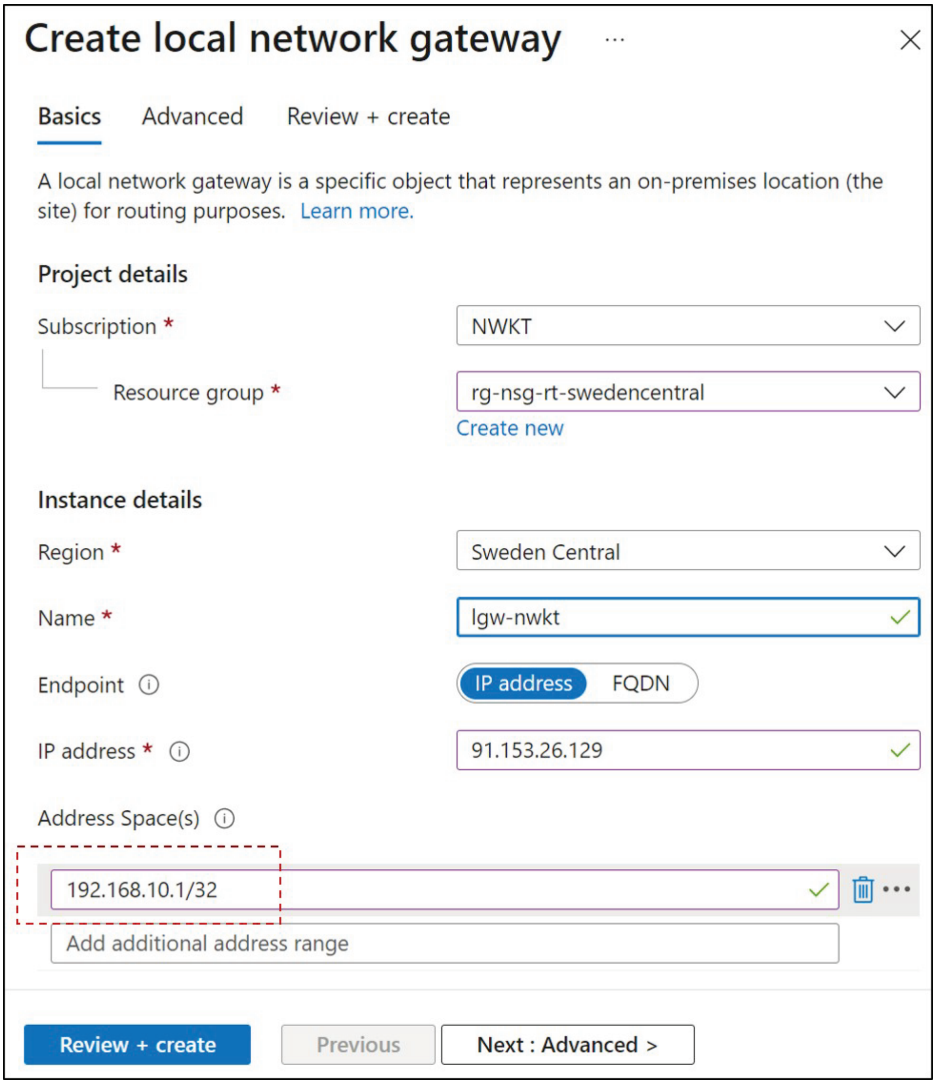
BGP peer IP address
10.0.3.254

Custom Azure APIPA BGP IP address
Peer Address

Figure 6-3: *VGW Configuration.*

Enable BGP on LGW

BGP session initiation starts with the TCP three-way handshake process (SYN, SYN-ACK, and ACK), where BGP peers open a TCP socket for BGP communication. BGP peers then use this socket for exchanging BGP messages (Open, Update, Keepalive, and Notification messages). To start the communication path initiation process, both BGP peers need to know the peer IP address and how to route IP packets to it. By adding the LGW BGP peering IP address in *Address Space(s)* field, we instruct VGW to route IP packets with the destination IP address 192.168.10.1 to VGW.


Create local network gateway

A local network gateway is a specific object that represents an on-premises location (the site) for routing purposes. Learn more.

Project details

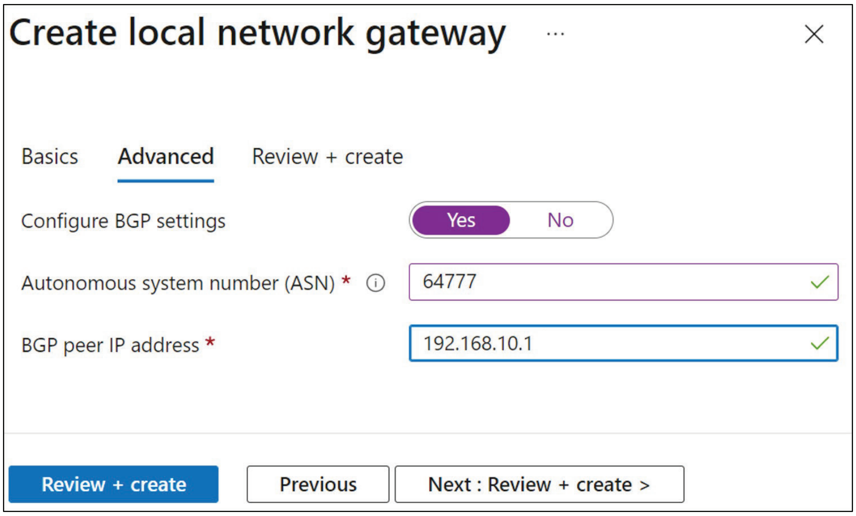
Subscription *: NWKT
Resource group *: rg-nsg-rt-swedencentral
Create new

Instance details

Region *: Sweden Central
Name *: lgw-nwkt
Endpoint: IP address / FQDN
IP address *: 91.153.26.129
Address Space(s): 192.168.10.1/32
Add additional address range

Figure 6-4: *LGW Configuration - Basics.*

Next, we define LGW's BGP ASN (65777) and the IP address for BGP peering (192.168.10.1) in an *Advanced* tab. The BGP peer IP address is also added to the LGW configuration file as Loopback interface 11, which is used as a BGP source address for all BGP messages.


Create local network gateway

Basics | Advanced | Review + create

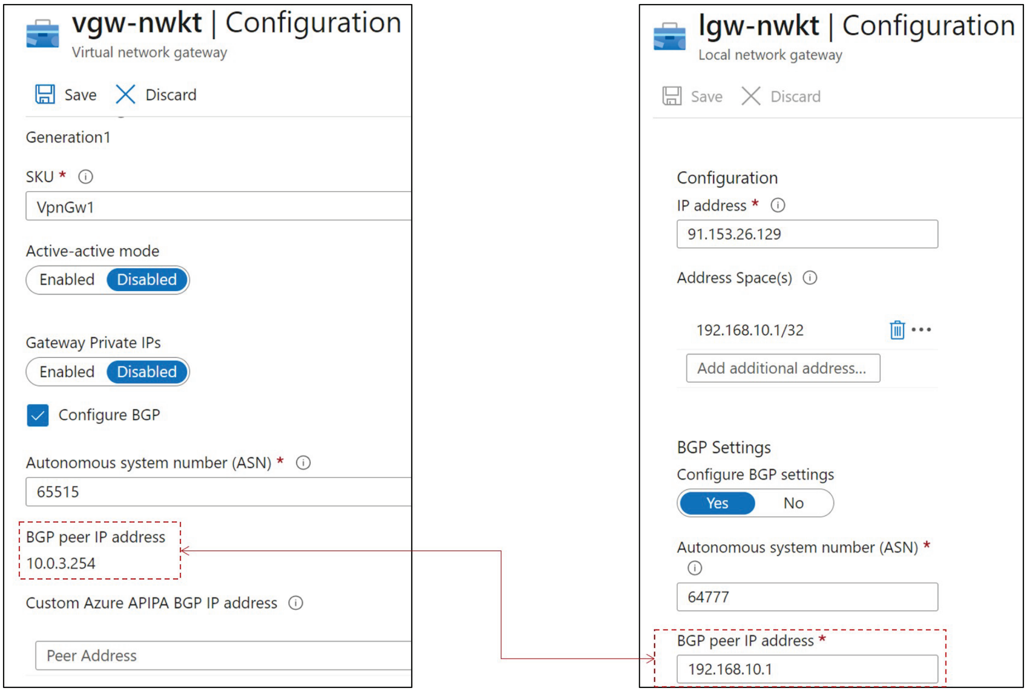
Configure BGP settings: [✓ Yes] No

Autonomous system number (ASN) *: 64777

BGP peer IP address *: 192.168.10.1

Figure 6-5: *LGW Configuration - Advanced.*

Figure 6-6 shows BGP ASNs and peering IP addresses of both VGW and LGW.


vgw-nwkt | Configuration
Virtual network gateway

Generation1

SKU *: VpnGw1

Active-active mode: Enabled [✓ Disabled]

Gateway Private IPs: Enabled [✓ Disabled]

✓ Configure BGP

Autonomous system number (ASN) *: 65515

BGP peer IP address: 10.0.3.254

Custom Azure APIPA BGP IP address: Peer Address

lgw-nwkt | Configuration
Local network gateway

Configuration

IP address *: 91.153.26.129

Address Space(s): 192.168.10.1/32

BGP Settings

Configure BGP settings: [✓ Yes] No

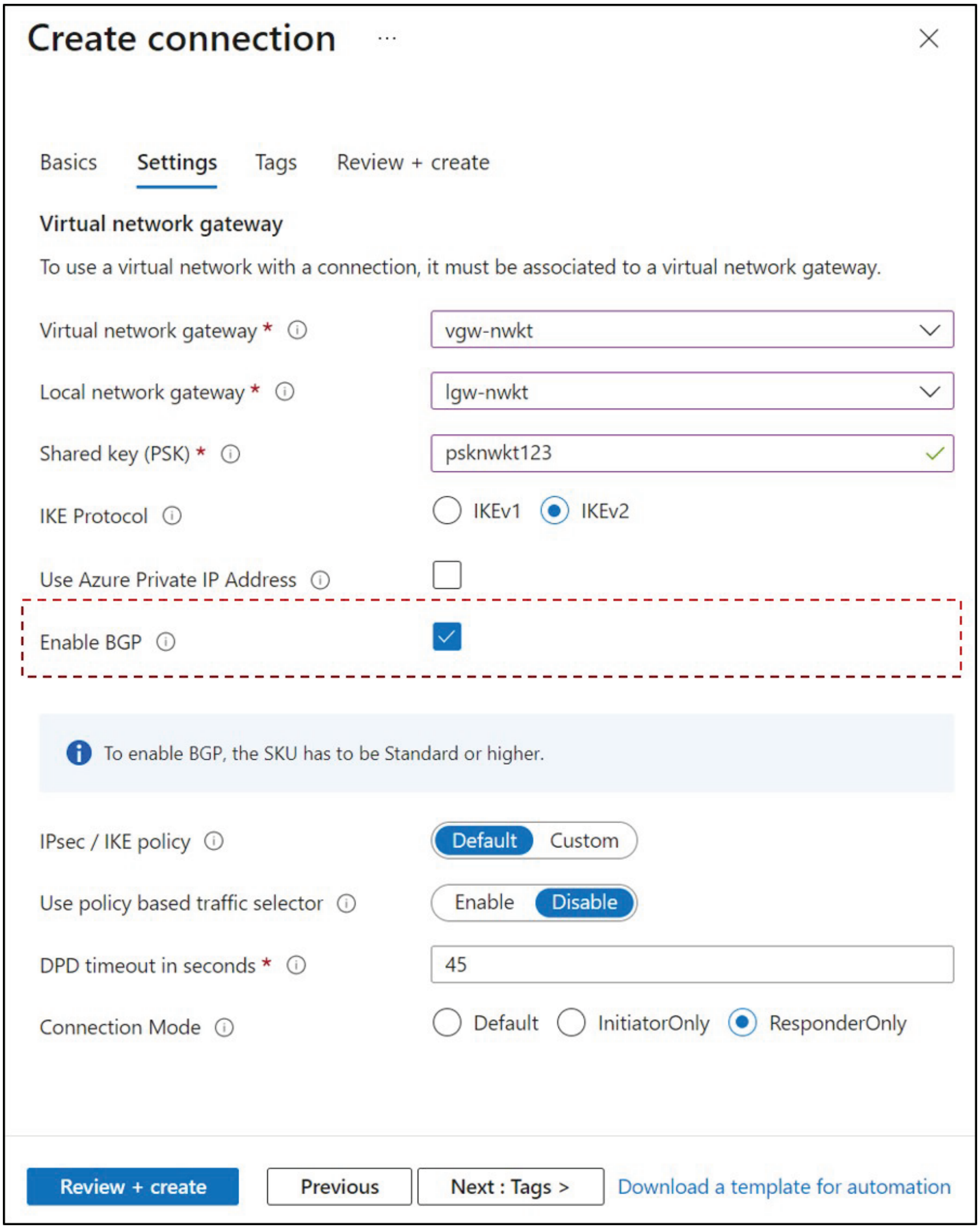
Autonomous system number (ASN) *: 64777

BGP peer IP address *: 192.168.10.1

Figure 6-6: *VGW and LGW Configuration.*

Enable BGP on S2S VPN Connection

After enabling BGP on VGW and LGW, we need to activate it on the VPN connection configuration by simply marking the *Enable BGP* option.



Create connection ...

Basics | **Settings** | Tags | Review + create

Virtual network gateway

To use a virtual network with a connection, it must be associated to a virtual network gateway.

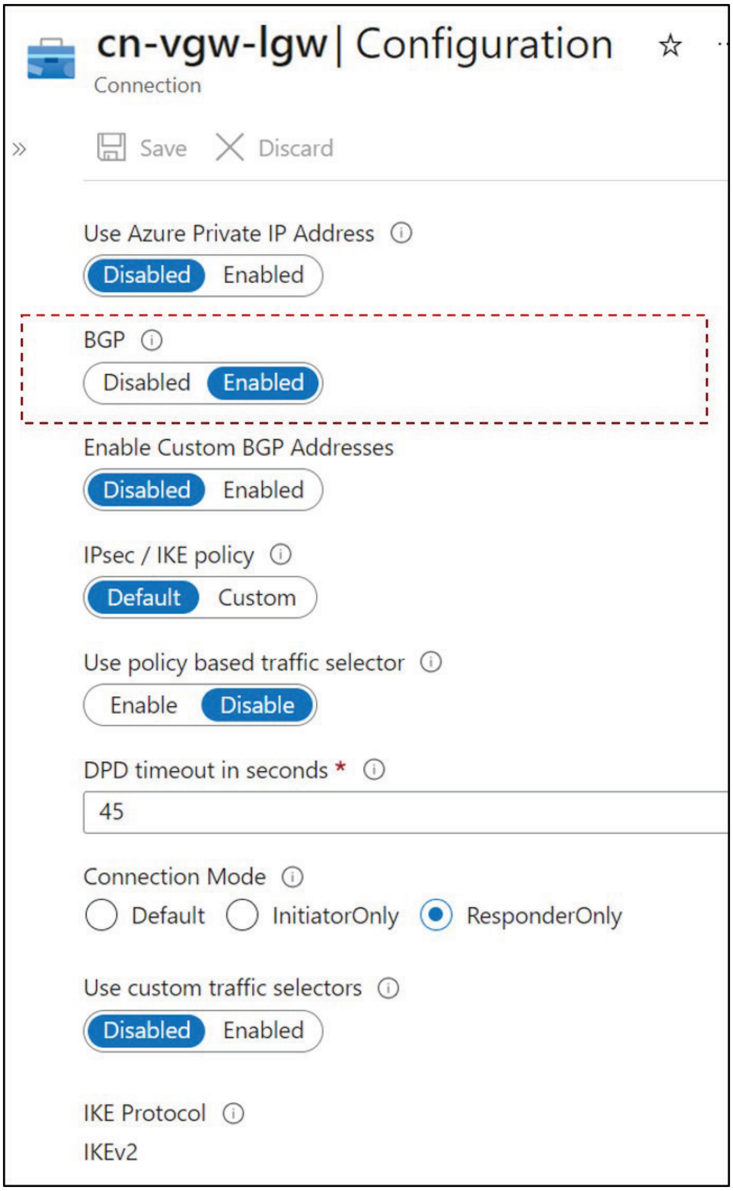
Virtual network gateway *	vgw-nwkt
Local network gateway *	lgw-nwkt
Shared key (PSK) *	psknwkt123 ✓
IKE Protocol	IKEv1 [✓ IKEv2]
Use Azure Private IP Address	☐
Enable BGP	☑

To enable BGP, the SKU has to be Standard or higher.

IPsec / IKE policy	[✓ Default] Custom
Use policy based traffic selector	Enable [✓ Disable]
DPD timeout in seconds *	45
Connection Mode	Default InitiatorOnly [✓ ResponderOnly]

Figure 6-7: *Enabling BGP on VPN Connection.*

The figure below shows the VPN connection configuration.



cn-vgw-lgw | Configuration

Connection

Save Discard

Use Azure Private IP Address: [✓ Disabled] Enabled

BGP: Disabled [✓ Enabled]

Enable Custom BGP Addresses: [✓ Disabled] Enabled

IPsec / IKE policy: [✓ Default] Custom

Use policy based traffic selector: Enable [✓ Disable]

DPD timeout in seconds *: 45

Connection Mode: Default InitiatorOnly [✓ ResponderOnly]

Use custom traffic selectors: [✓ Disabled] Enabled

IKE Protocol: IKEv2

Figure 6-8: *VPN Connection Configuration.*

Configure BGP on LGW

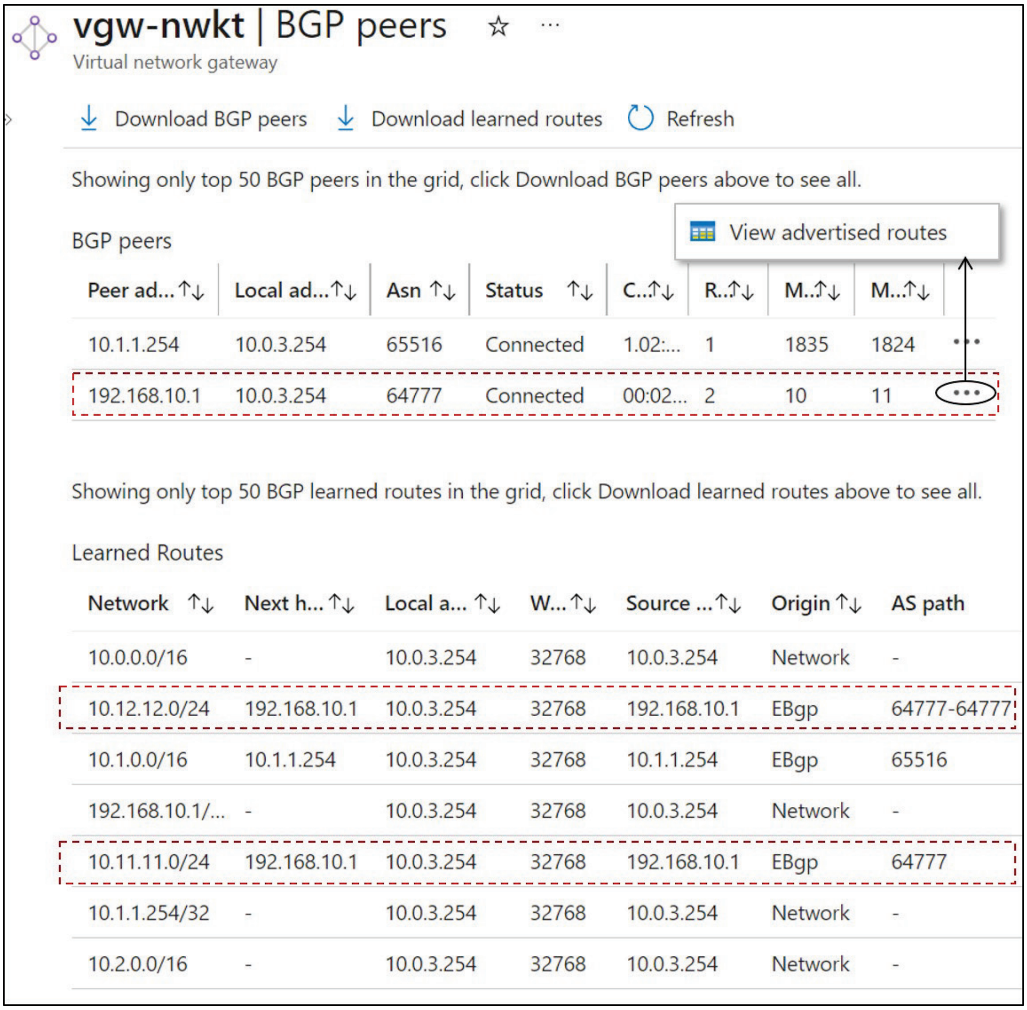
Example 6-1 shows the BGP configuration of our LGW downloaded from the Azure portal. The grayed sections are related to AS-PATH prepending and are not included in the configuration file generated by Azure. Note that the *ebgp-multihop* command changes the default Time to Live (TTL) value of the BGP message's IP header from 1 to 255. The Active-Active VGW does flow-based load-balancing by default. I have added an AS-Path prepend configuration to show how you can affect the path selection process on the VGW side if you have multiple tunnels between VGW and LGW.

```
LGW-NWKT#

interface Loopback11
 ip address 192.168.10.1 255.255.255.255
!
router bgp 64777
 bgp log-neighbor-changes
 neighbor 10.0.3.254 remote-as 65515
 neighbor 10.0.3.254 ebgp-multihop 255
 neighbor 10.0.3.254 update-source Loopback11
 !
 address-family ipv4
  network 10.11.11.0 mask 255.255.255.0
  network 10.12.12.0 mask 255.255.255.0
  neighbor 10.0.3.254 activate
  neighbor 10.0.3.254 route-map AS_Path-Prepend out
 exit-address-family
!
ip route 10.0.3.254 255.255.255.255 Tunnel 11
!
ip prefix-list AS_PATH-Prepend seq 5 permit 10.12.12.0/24
!
!
route-map AS-Path-Prep permit 10
!
route-map AS_Path-Prepend permit 10
 match ip address prefix-list AS_PATH-Prepend
 set as-path prepend 64777
!
route-map AS_Path-Prepend permit 100
!
```

Example 6-1: *The BGP Configuration of LGW.*

Control Plane Verification on VGW

You can verify the BGP peering and routing by navigating to the VGW resource and selecting *BGP Peering* under the Monitoring section. Figure 6-9 shows that the VGW peers with LGW (192.168.10.1) and that VGW has sent 10 BGP messages to LGW and received 11 BGP messages. The figure also verifies that VGW has received BGP Update messages, which describes the Network layer Reachability Information (NLRI) about subnets 10.12.12.0/24 and 10.11.11.0/24 from LGW. The BGP Path Attributes carried within these BGP updates are similar with the difference that the AS Path list to 10.12.12.0/24 is prepended by one (see example 6-1). You can check the advertised routes to LGW by clicking the ". . ." selector at the end of the *BGP Peers* row and selecting *View advertised routes*.


vgw-nwkt | BGP peers

Virtual network gateway

Download BGP peers | Download learned routes | Refresh

Showing only top 50 BGP peers in the grid, click Download BGP peers above to see all.

BGP peers

View advertised routes

Peer ad...	Local ad...	Asn	Status	C...	R...	M...	M...	
10.1.1.254	10.0.3.254	65516	Connected	1.02:...	1	1835	1824	•••
192.168.10.1	10.0.3.254	64777	Connected	00:02...	2	10	11	•••

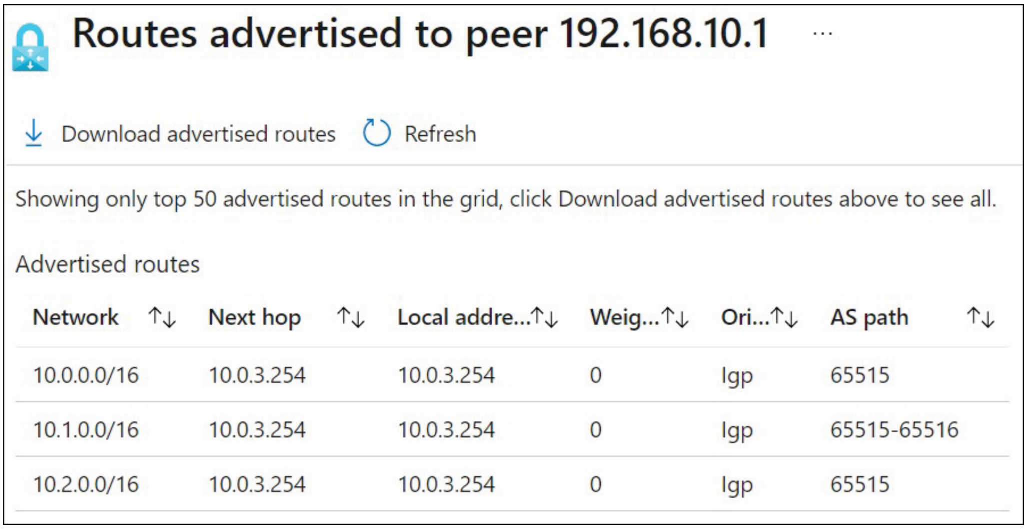
Showing only top 50 BGP learned routes in the grid, click Download learned routes above to see all.

Learned Routes

Network	Next h...	Local a...	W...	Source ...	Origin	AS path
10.0.0.0/16	-	10.0.3.254	32768	10.0.3.254	Network	-
10.12.12.0/24	192.168.10.1	10.0.3.254	32768	192.168.10.1	EBgp	64777-64777
10.1.0.0/16	10.1.1.254	10.0.3.254	32768	10.1.1.254	EBgp	65516
192.168.10.1/...	-	10.0.3.254	32768	10.0.3.254	Network	-
10.11.11.0/24	192.168.10.1	10.0.3.254	32768	192.168.10.1	EBgp	64777
10.1.1.254/32	-	10.0.3.254	32768	10.0.3.254	Network	-
10.2.0.0/16	-	10.0.3.254	32768	10.0.3.254	Network	-

Figure 6-9: *VGW BGP Peers and Learned Routes.*

Figure 6-10 shows that the VGW has advertised three BGP Updates to LGW. The first one on the list describes the BGP NLRI about subnet 10.0.0.0/16, which is the CIDR range associated with the VNet vnet-nsg-rt-swedencentral where we have attached the VGW. The subnets 10.1.0.0/16 and 10.2.0.0/16 belong to VNets, which we use with VNet-to-VNet VPN (chapter 7) and VNet peering (chapter 8).


Routes advertised to peer 192.168.10.1

Showing only top 50 advertised routes in the grid, click Download advertised routes above to see all.

Advertised routes

Network	Next hop	Local addre...	Weig...	Ori...	AS path
10.0.0.0/16	10.0.3.254	10.0.3.254	0	Igp	65515
10.1.0.0/16	10.0.3.254	10.0.3.254	0	Igp	65515-65516
10.2.0.0/16	10.0.3.254	10.0.3.254	0	Igp	65515

Figure 6-10: *VGW Advertised Routes.*

Example 6-2 shows how you can get information about VGW Peers using the Azure PowerShell command:

*Get-AzVirtualNetworkGateway**BgpPeerStatus** -ResourceGroupName [rg name] -VirtualNetworkGatewayName [vgw name].*

```
Get-AzVirtualNetworkGatewayBgpPeerStatus -ResourceGroupName rg-nsg-rt-
swedencentral -VirtualNetworkGatewayName vgw-nwkt

LocalAddress      : 10.0.3.254
Neighbor          : 192.168.10.1
Asn               : 64777
State             : Connected
ConnectedDuration : 00:27:43.3460497
RoutesReceived    : 1
MessagesSent      : 35
MessagesReceived  : 35
```

Example 6-2: *Get-AzVirtualNetworkGateway**BgpPeerStatus**.*

Example 6-3 shows how you can get information about routes advertised by VGW Peers using the Azure PowerShell command:

*Get-AzVirtualNetworkGateway**AdvertisedRoute** -VirtualNetworkGatewayName [vgw name]-ResourceGroupName [rg name] -Peer [BGP peer IP address].*

```
Get-AzVirtualNetworkGatewayAdvertisedRoute -VirtualNetworkGatewayName vgw-
nwkt -ResourceGroupName rg-nsg-rt-swedencentral -Peer 192.168.10.1

LocalAddress : 10.0.3.254
Network      : 10.0.0.0/16
NextHop      : 10.0.3.254
SourcePeer   :
Origin       : Igp
AsPath       : 65515
Weight       : 0
```

Example 6-3: *Get-AzVirtualNetworkGateway**AdvertisedRoute**.*

Example 6-4 shows how you can verify BGP NLRI about learned subnets with their associated BGP Path Attributes by using the Azure PowerShell command:

Get-AzVirtualNetworkGatewayLearnedRoute -ResourceGroupName [rg name] -VirtualNetworkGatewayname [vge name].

```
Get-AzVirtualNetworkGatewayLearnedRoute -ResourceGroupName rg-nsg-rt-
swedencentral -VirtualNetworkGatewayname vgw-nwkt

LocalAddress : 10.0.3.254
Network      : 10.0.0.0/16
NextHop      :
SourcePeer   : 10.0.3.254
Origin       : Network
AsPath       :
Weight       : 32768

LocalAddress : 10.0.3.254
Network      : 192.168.10.1/32
NextHop      :
SourcePeer   : 10.0.3.254
Origin       : Network
AsPath       :
Weight       : 32768

LocalAddress : 10.0.3.254
Network      : 10.11.11.0/24
NextHop      : 192.168.10.1
SourcePeer   : 192.168.10.1
Origin       : EBgp
AsPath       : 64777
Weight       : 32768

LocalAddress : 10.0.3.254
Network      : 10.12.12.0/24
NextHop      : 192.168.10.1
SourcePeer   : 192.168.10.1
Origin       : EBgp
AsPath       : 64777-64777
Weight       : 32768
```

Example 6-4: *Get-AzVirtualNetworkGateway**LearnedRoute**.*

Control Plane Verification on LGW

Example 6-5 shows the BGP peering statistics from the LGW perspective. The status of the BGP neighbor 10.0.3.254 (vgw-nwkt) on BGP AS 65515 is up, and LGW has received one prefix from the peer.

```
LGW-NWKT#show ip bgp summary
BGP router identifier 192.168.10.1, local AS number 64777
BGP table version is 3, main routing table version 3
2 network entries using 496 bytes of memory
2 path entries using 272 bytes of memory
2/2 BGP path/bestpath attribute entries using 560 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1352 total bytes of memory
BGP activity 2/0 prefixes, 2/0 paths, scan interval 60 secs

Neighbor     V   AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.0.3.254   4   65515      18      19        0    0    0 00:13:33        1
```

Example 6-5: *show ip bgp summary.*

Example 6-6 shows the LGW's BGP table. The LGW has installed prefix 10.11.11.0/24 with its associated BGP Path attributes from the BGP Adj-RIB-In table into the BGP Loc-RIB-In table without modification. The prefix 10.0.0.0/16 is valid (the next hop carried within the BGP update is reachable). There is only one path to prefix 10.0.0.0/16, which the BGP process has chosen as the best path.

```
LGW-NWKT#show ip bgp
BGP table version is 3, local router ID is 192.168.10.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>   10.0.0.0/16      10.0.3.254                             0 65515 i
 *>   10.11.11.0/24    0.0.0.0                  0         32768 i
```

Example 6-6: *show ip bgp.*

Example 6-7 shows the prefix 10.0.0.0/16 specific information stored in the BGP Loc-RIB table.

```
LGW-NWKT#show ip bgp 10.0.0.0 255.255.0.0
BGP routing table entry for 10.0.0.0/16, version 2
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 1
  65515
    10.0.3.254 from 10.0.3.254 (10.0.3.254)
      Origin IGP, localpref 100, valid, external, best
      rx pathid: 0, tx pathid: 0x0
```

Example 6-7: *show ip bgp 10.0.0.0 255.255.0.0.*

Example 6-8 verifies that the LGW has imported routing information about prefix 10.0.0.0/16 from the BGP Loc-RIB into the RIB (Routing Information Base = Routing Table).

```
LGW-NWKT#show ip route 10.0.0.0 255.255.0.0
Routing entry for 10.0.0.0/16
  Known via "bgp 64777", distance 20, metric 0
  Tag 65515, type external
  Last update from 10.0.3.254 00:05:55 ago
  Routing Descriptor Blocks:
  * 10.0.3.254, from 10.0.3.254, 00:05:55 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
      Route tag 65515
      MPLS label: none
```

Example 6-8: *show ip route 10.0.0.0 255.255.0.0.*

Example 6-9 shows that the TCP connection state is Established between LGW and VGW.

```
LGW-NWKT#show tcp brief
TCB       Local Address              Foreign Address           (state)
7FF538FDCAF8  192.168.10.1.179          10.0.3.254.50094            ESTAB
```

Example 6-9: *show tcp brief.*

As a last step we can verify that the data plane is ok by pinging the VM vm-Front-1 (10.0.1.4) from the LGW using loopback 1011 IP address 10.11.11.1 as a source.

```
LGW-NWKT#ping 10.0.1.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 13/13/14 ms
```

Example 6-9: *show tcp brief.*

References

[1] How to configure BGP on Azure VPN Gateways
https://docs.microsoft.com/en-us/azure/vpn-gateway/bgp-howto

[2] About BGP with Azure VPN Gateway
https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-bgp-overview

[3] Get-AzVirtualNetworkGatewayBGPPeerStatus
https://docs.microsoft.com/en-us/powershell/module/az.network/get-azvirtualnetworkgatewaybgppeerstatus?view=azps-8.0.0

[4] Get-AzVirtualNetworkGatewayAdvertisedRoute
https://docs.microsoft.com/en-us/powershell/module/az.network/get-azvirtualnetworkgatewayadvertisedroute?view=azps-8.0.0

[5] Get-AzVirtualNetworkGatewayLearnedRoute
https://docs.microsoft.com/en-us/powershell/module/az.network/get-azvirtualnetworkgatewaylearnedroute?view=azps-8.0.0

[6] What Is VPN Gateway
https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpngateways

Chapter 7: VNet-to-VNet VPN

Introduction

In the previous two chapters, we first implemented a VPN Connection between *vgw-nwkt* and *lgw-nwkt* and then established a BGP peering between them. In this chapter, we launch a new VGW, *vgw-spoke-1*, in VNet *vnet-spoke-1* with a CIDR 10.1.0.0/16. Then we configure a secure VNet-to-VNet connection between *vgw-vnet2* and *vgw-nwkt*. Besides, we implement BGP peering between VGWs.

Figure 7-1 illustrates our example topology. To allow bi-directional connection between VMs in VNets, we create two connections, *cn-spoke1-hub-1-1* and *cn-spoke1-hub-1-2* between VGWs. These connections are established over Azure's infrastructure using VGW-specific public IP addresses. We have created VNet-specific GatewaySubnets from where Azure allocates IP addresses for eBGP peering. VGW-based connection architecture supports transitive routing relationships. In our example, the hub *vgw-nwkt* forwards valid routing information received via BGP update message from *lgw-nwkt* to *vgw-vnet2*, and the other way around. This is the default BGP operation between external BGP peers (BGP peers are in a different BGP AS).

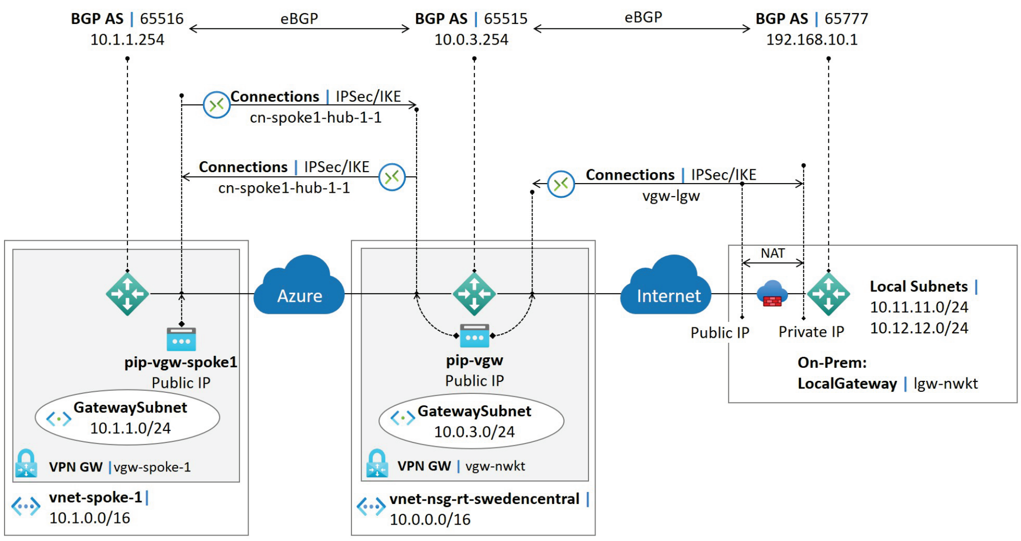
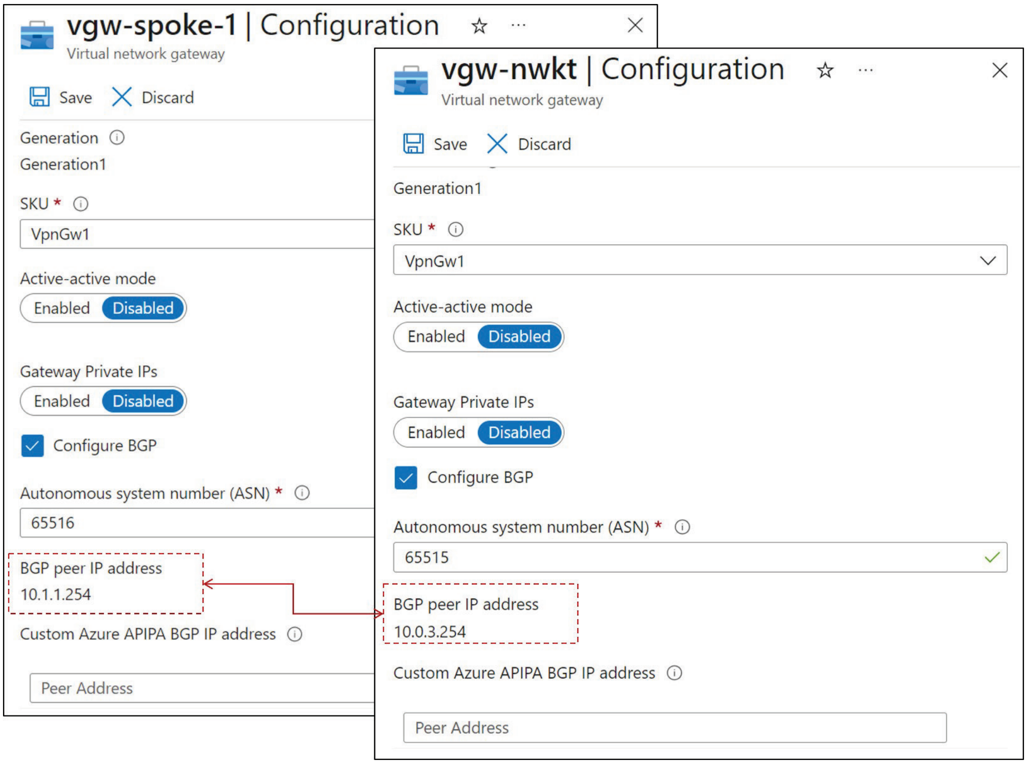


Figure 7-1: *VNet-to-VNet Example Diagram.*

VGW Settings

The figure below shows the configuration of VGWs. We have defined a BGP ASN 65516 for *vgw-spoke-1* and 65515 (default) for *vgw-nwkt*. The BGP peering IP addresses are derived automatically from the GatewaySubnet address range.



vgw-spoke-1 | Configuration
Virtual network gateway

Generation
Generation1
SKU *
VpnGw1
Active-active mode
Enabled [✓ Disabled]
Gateway Private IPs
Enabled [✓ Disabled]
✓ Configure BGP
Autonomous system number (ASN) *
65516
BGP peer IP address
10.1.1.254
Custom Azure APIPA BGP IP address
Peer Address

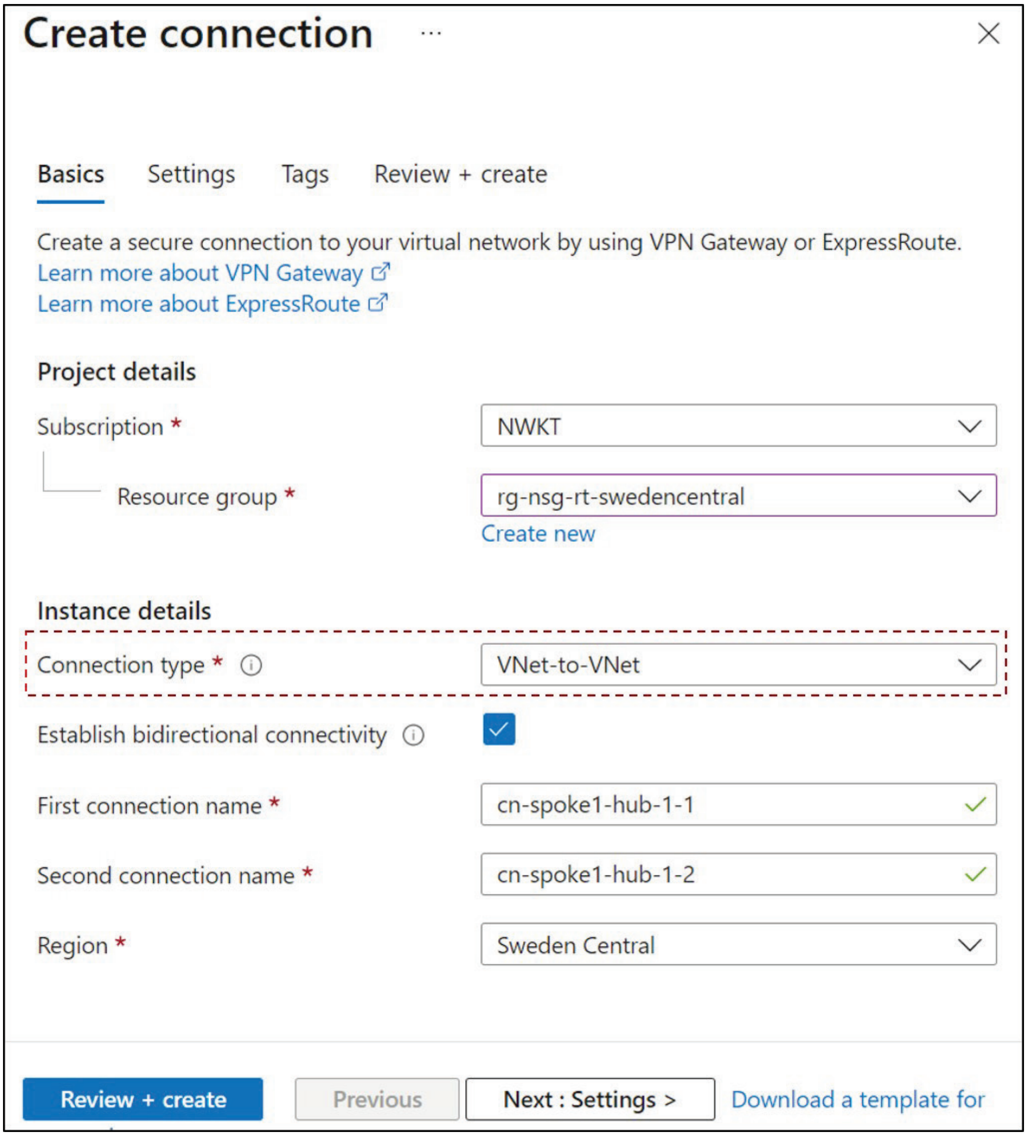
vgw-nwkt | Configuration
Virtual network gateway

Generation1
SKU *
VpnGw1
Active-active mode
Enabled [✓ Disabled]
Gateway Private IPs
Enabled [✓ Disabled]
✓ Configure BGP
Autonomous system number (ASN) *
65515
BGP peer IP address
10.0.3.254
Custom Azure APIPA BGP IP address
Peer Address

Figure 7-2: *VGW Configuration.*

Connection Settings

Select a VNet-to-Vnet connection type from the *Connection type* drop-down menu on the Basics tab. Then mark the *Establish bidirectional connectivity* and fill in the name fields. We use the same resource group and region in this example we have used in previous chapters.



Create a secure connection to your virtual network by using VPN Gateway or ExpressRoute.
Learn more about VPN Gateway
Learn more about ExpressRoute

Project details

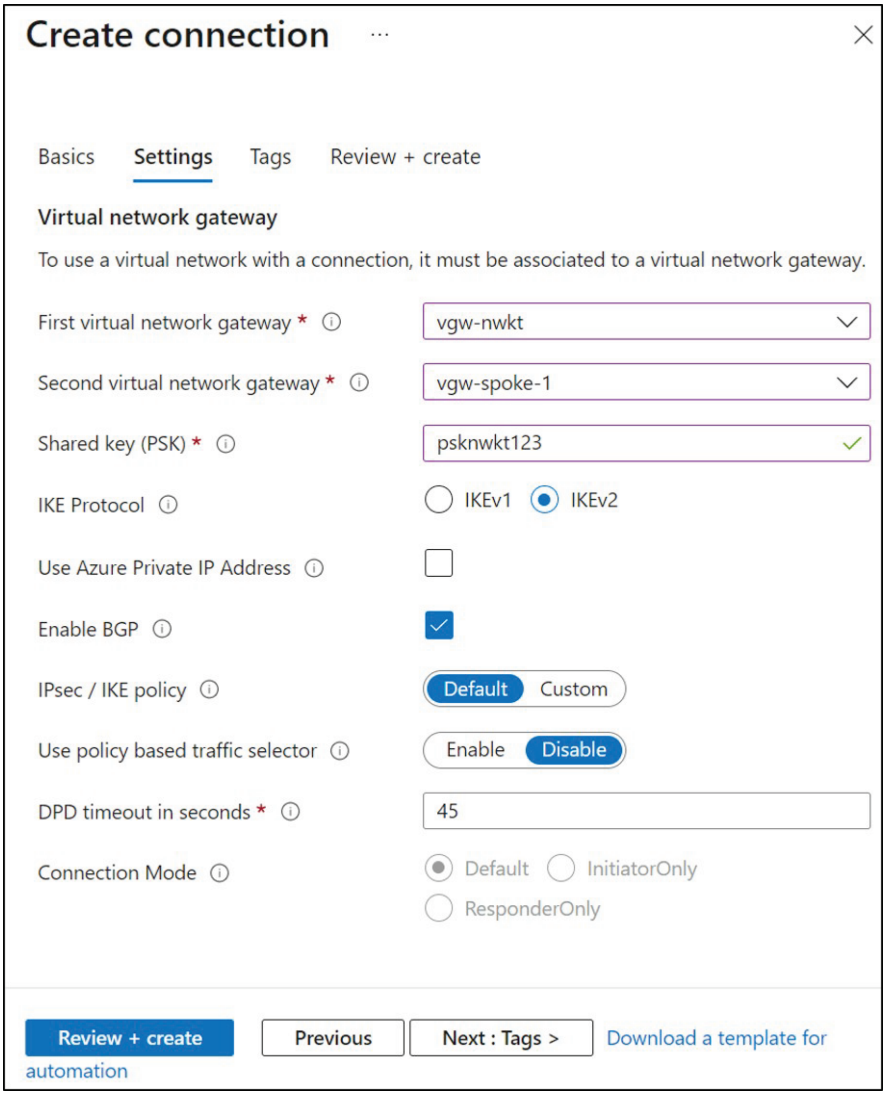
Subscription *	NWKT
Resource group *	rg-nsg-rt-swedencentral
	Create new

Instance details

Connection type *	VNet-to-VNet
Establish bidirectional connectivity	✓
First connection name *	cn-spoke1-hub-1-1
Second connection name *	cn-spoke1-hub-1-2
Region *	Sweden Central

Figure 7-3: *Connection Settings – Connection Type and Name.*

Select VGWs from the drop-down menu and define IKE/IPSec settings.



Create connection

Basics | **Settings** | Tags | Review + create

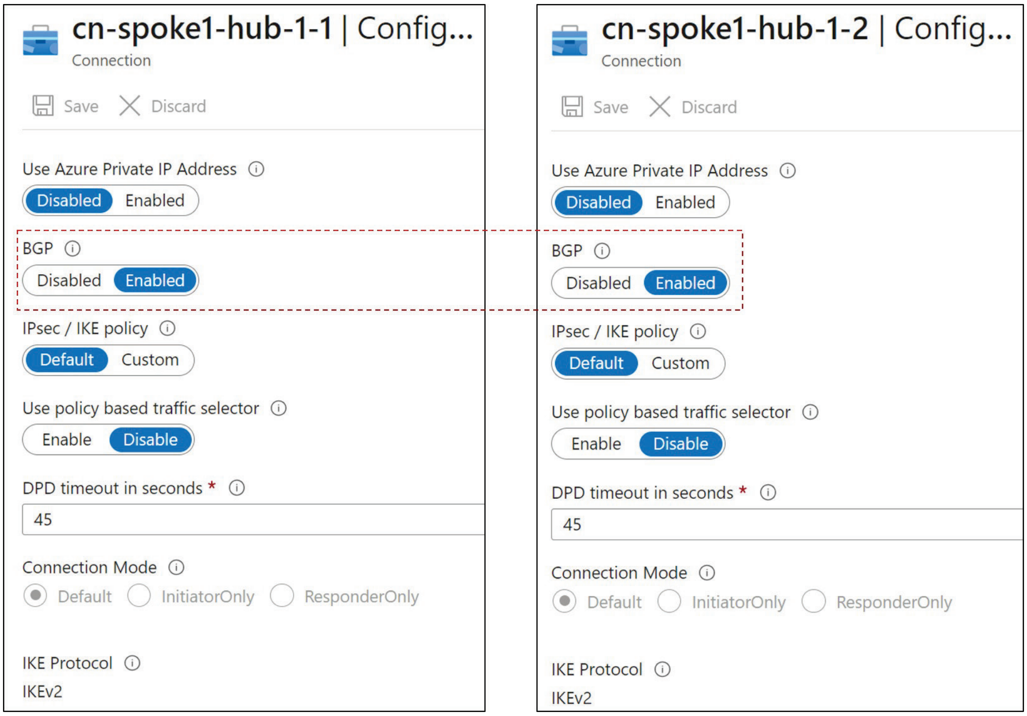
Virtual network gateway

To use a virtual network with a connection, it must be associated to a virtual network gateway.

First virtual network gateway *	vgw-nwkt
Second virtual network gateway *	vgw-spoke-1
Shared key (PSK) *	psknwkt123
IKE Protocol	IKEv1 [✓ IKEv2]
Use Azure Private IP Address	☐
Enable BGP	✓
IPsec / IKE policy	[✓ Default] Custom
Use policy based traffic selector	Enable [✓ Disable]
DPD timeout in seconds *	45
Connection Mode	[✓ Default] InitiatorOnly ResponderOnly

Figure 7-4: *Connection Settings – Define VGWs and Security Settings.*

Figure 7-5 verifies that we have successfully enabled BGP on both connections. The VGW configuration pages shown in figure 7-6 verifies that *vgw-nwkt* and *vgw-spoke-1* are peering over connections *cn-spoke1-hub-1-1* and *cn-spoke1-hub-1-2*.



cn-spoke1-hub-1-1 | Config...
Connection

Save Discard

Use Azure Private IP Address: [✓ Disabled] Enabled

BGP: Disabled [✓ Enabled]

IPsec / IKE policy: [✓ Default] Custom

Use policy based traffic selector: Enable [✓ Disable]

DPD timeout in seconds *: 45

Connection Mode: [✓ Default] InitiatorOnly ResponderOnly

IKE Protocol: IKEv2

cn-spoke1-hub-1-2 | Config...
Connection

Save Discard

Use Azure Private IP Address: [✓ Disabled] Enabled

BGP: Disabled [✓ Enabled]

IPsec / IKE policy: [✓ Default] Custom

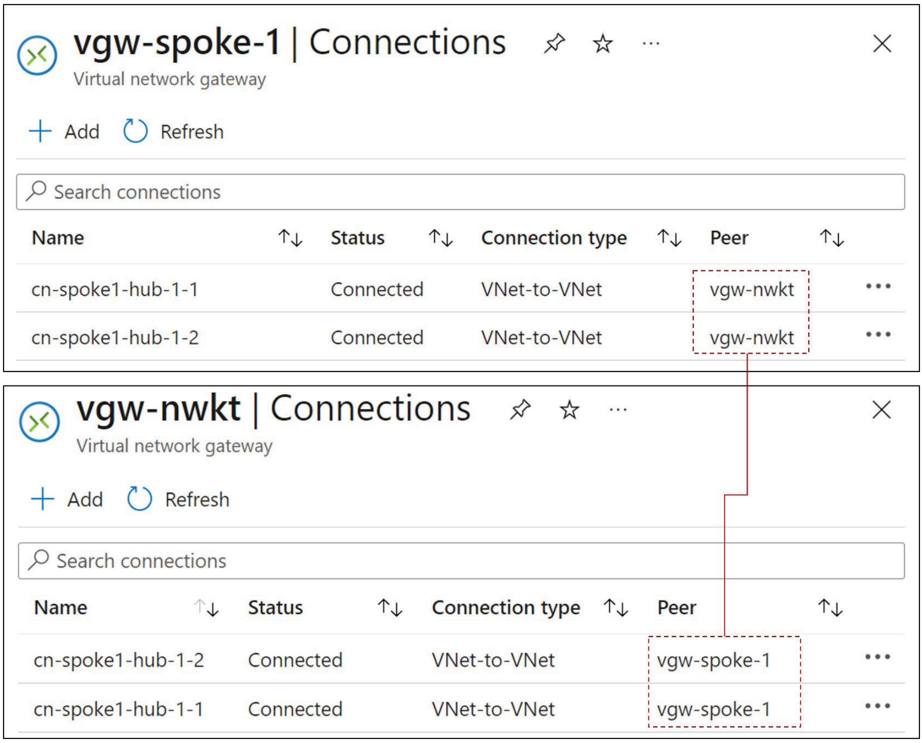
Use policy based traffic selector: Enable [✓ Disable]

DPD timeout in seconds *: 45

Connection Mode: [✓ Default] InitiatorOnly ResponderOnly

IKE Protocol: IKEv2

Figure 7-5: *Connection Configuration.*



vgw-spoke-1 | Connections
Virtual network gateway

Add Refresh

Search connections

Name	Status	Connection type	Peer
cn-spoke1-hub-1-1	Connected	VNet-to-VNet	vgw-nwkt
cn-spoke1-hub-1-2	Connected	VNet-to-VNet	vgw-nwkt

vgw-nwkt | Connections
Virtual network gateway

Add Refresh

Search connections

Name	Status	Connection type	Peer
cn-spoke1-hub-1-2	Connected	VNet-to-VNet	vgw-spoke-1
cn-spoke1-hub-1-1	Connected	VNet-to-VNet	vgw-spoke-1

Figure 7-6: *The VNet-to-Vnet Connection Between VGWs.*

Figure 7-7 shows the Diagram view for both *vnet-spoke-1* and *vnet-nsg-rt-swedencentral*. The connection *cn-spoke1-hub-1-1* is associated with vgw-nwkt and *cn-spoke1-hub-1-2* with *vgw-spoke-1*. Note that I have slightly modified the diagram view by removing unnecessary resources.

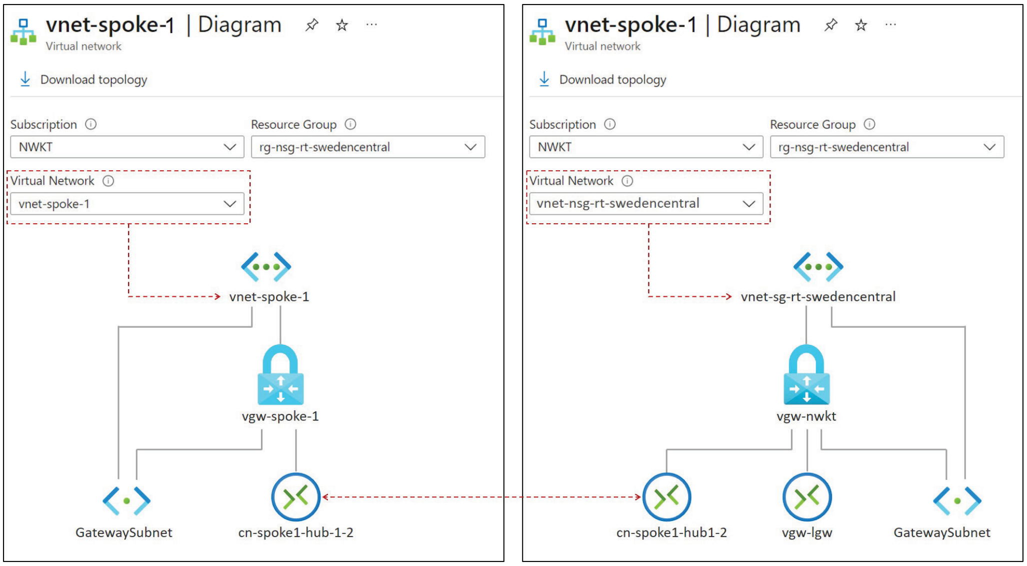


Figure 7-7: *Vnet Diagrams.*

Control Plane Verification

Figure 7-8 shows how BGP Update messages are propagated. The VGW in vnet-spoke-1 sends a BGP Update message about prefix 10.1.0.0/16. It sets the local BGP ASN 65516 in the *AS-Path* Path Attribute (PA) list and BGP Peering IP address 10.1.1.254 as a *Next-Hop* PA. When vgw-nwkt receives the BGP Update message, it stores the information into neighbor-specific BGP *Adj-RIB-In*. Then it validates information by checking that the Next Hop IP address is reachable and that the BGP ASN matches to configured peer BGP ASN. After successful validation, it installs the NLRI into BGP Loc-RIB as a valid and best route. Then it programs the information into the routing table. Next, the BGP process constructs a new BGP Update message and sends it to the neighbor-specific (lgw-nwkt) BGP Adj-RIB-Out table. Before sending the BGP Update message, it adds the local BGP ASN 65515 and the original ASN 65516 to the AS-Path PA list and sets the BGP peering IP address 10.0.3.254 to the Next-Hop PA field. The process recurs in the opposite direction concerning remote location subnets. Note that the AS path related to remote location subnet 10.12.12.0/24 is prepended by one local AS no lgw-nwkt.

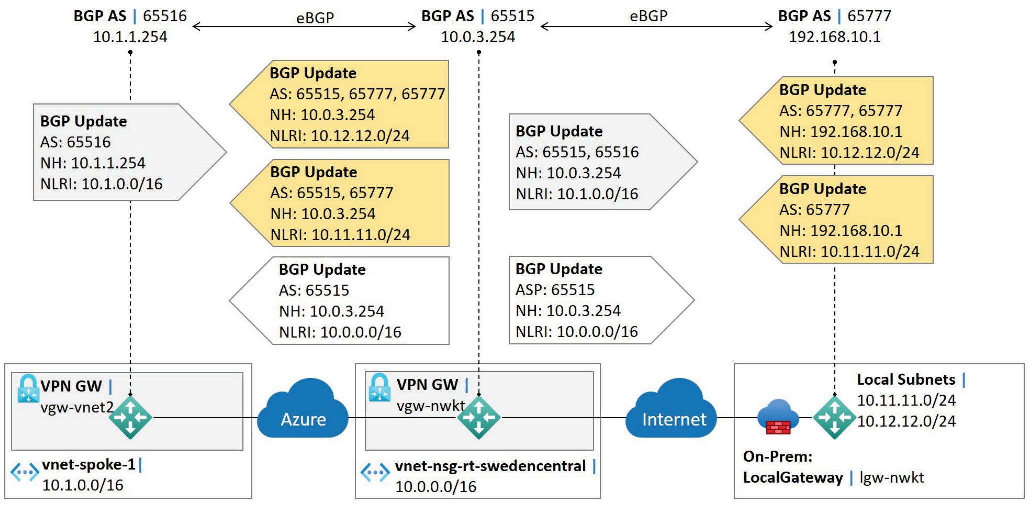
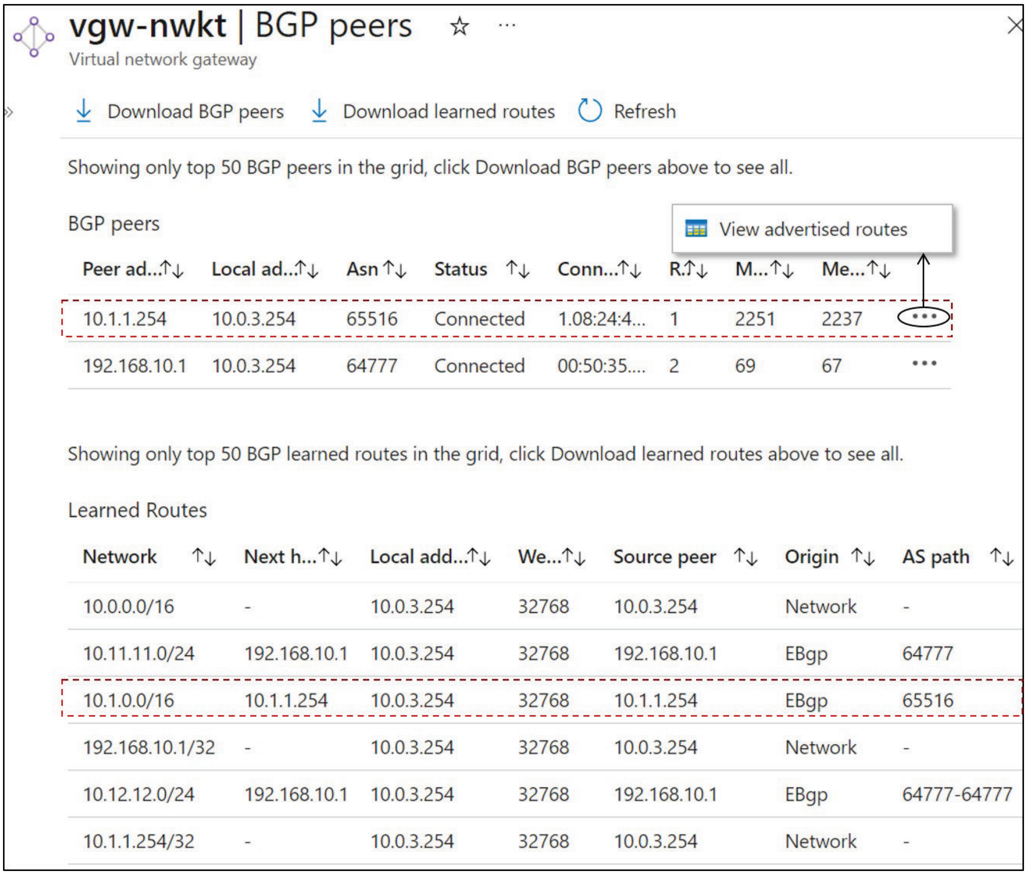


Figure 7-8: *The Propagation of BGP Update Messages.*

The BGP peers window of vgw-nwkt shows BGP peers and routes learned via BGP. The figure below shows that vgw-nwkt is peering with 10.1.1.254 (vgw-spoke-1) and 192.168.10.1 (lgw-nwkt). It has received routing information about 10.1.0.0/16 from vgw-spoke-1 with the AS-Path PA 65516. The LGW lgw-nwkt, in turn, has advertised two local subnets 10.11.11.0/24 (AS-Path 65777) and 10.12.12.0/24 (AS-Path 65777-65777). The local network 10.0.0.0/16 is listed without the Next-Hop address. The BGP peering IP address 192.168.10.1 (vgw-nwkt) is shown without the Next-Hop information because it is routed to the IPSec tunnel to the remote location, not to the BGP peer. Without the route, vgw-nwkt and lgw-nwkt can't establish a BGP peering relationship. You can verify routes that vgw-nwkt has advertised to its peer by clicking the "..." at the end of the BGP peer row.



Showing only top 50 BGP peers in the grid, click Download BGP peers above to see all.

BGP peers

Peer ad...	Local ad...	Asn	Status	Conn...	R.	M...	Me...	
10.1.1.254	10.0.3.254	65516	Connected	1.08:24:4...	1	2251	2237	•••
192.168.10.1	10.0.3.254	64777	Connected	00:50:35....	2	69	67	•••

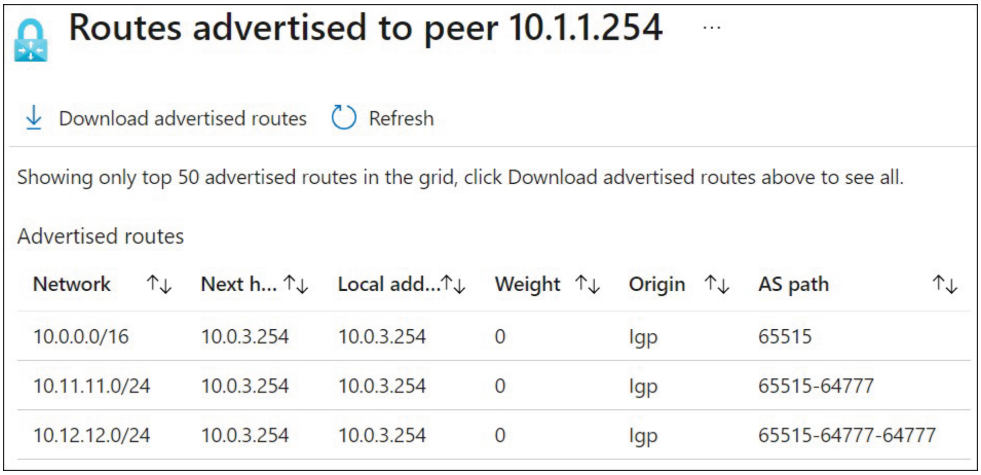
Showing only top 50 BGP learned routes in the grid, click Download learned routes above to see all.

Learned Routes

Network	Next h...	Local add...	We...	Source peer	Origin	AS path
10.0.0.0/16	-	10.0.3.254	32768	10.0.3.254	Network	-
10.11.11.0/24	192.168.10.1	10.0.3.254	32768	192.168.10.1	EBgp	64777
10.1.0.0/16	10.1.1.254	10.0.3.254	32768	10.1.1.254	EBgp	65516
192.168.10.1/32	-	10.0.3.254	32768	10.0.3.254	Network	-
10.12.12.0/24	192.168.10.1	10.0.3.254	32768	192.168.10.1	EBgp	64777-64777
10.1.1.254/32	-	10.0.3.254	32768	10.0.3.254	Network	-

Figure 7-9: *BGP Peers and BGP Routing Information.*

Figure 7-10 shows that vgw-nwkt has advertised its local CIDR 10.0.0.0/16 and prefixes 10.11.11.0/16 and 10.12.12.0/16 learned from lgw-nwkt to vgw-spoke-1.



Routes advertised to peer 10.1.1.254

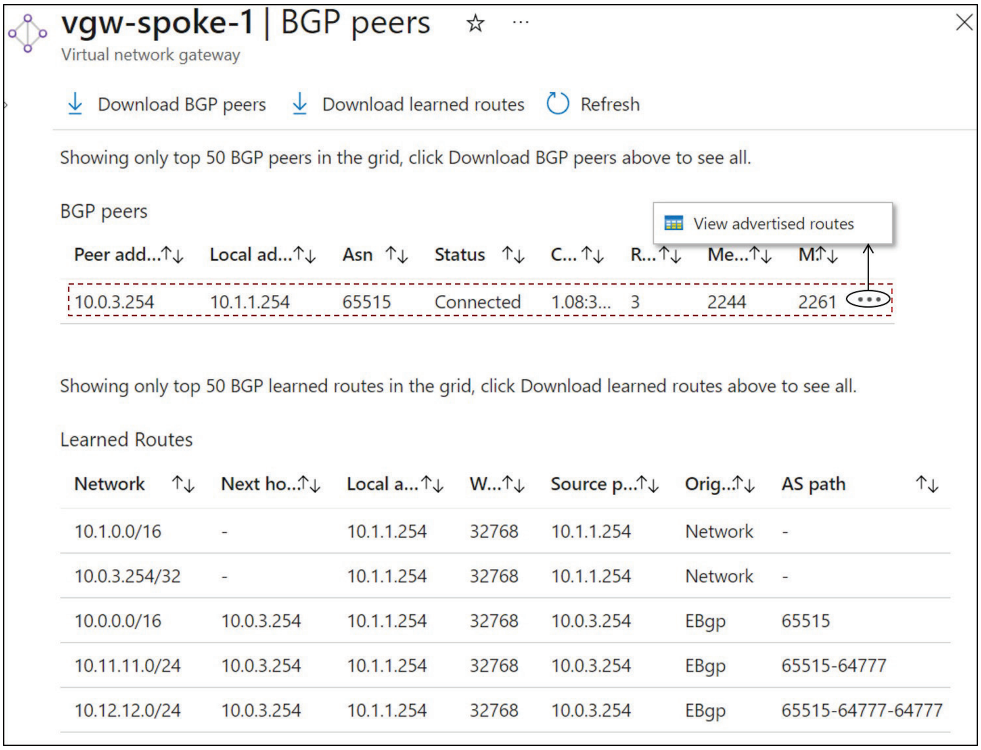
Showing only top 50 advertised routes in the grid, click Download advertised routes above to see all.

Advertised routes

Network	Next h...	Local add...	Weight	Origin	AS path
10.0.0.0/16	10.0.3.254	10.0.3.254	0	Igp	65515
10.11.11.0/24	10.0.3.254	10.0.3.254	0	Igp	65515-64777
10.12.12.0/24	10.0.3.254	10.0.3.254	0	Igp	65515-64777-64777

Figure 7-10: *BGP Route Advertisement.*

Figures 7-11 and 7-12 show the same information from the vgw-spoke-1 perspective.



vgw-spoke-1 | BGP peers

Virtual network gateway

Showing only top 50 BGP peers in the grid, click Download BGP peers above to see all.

BGP peers

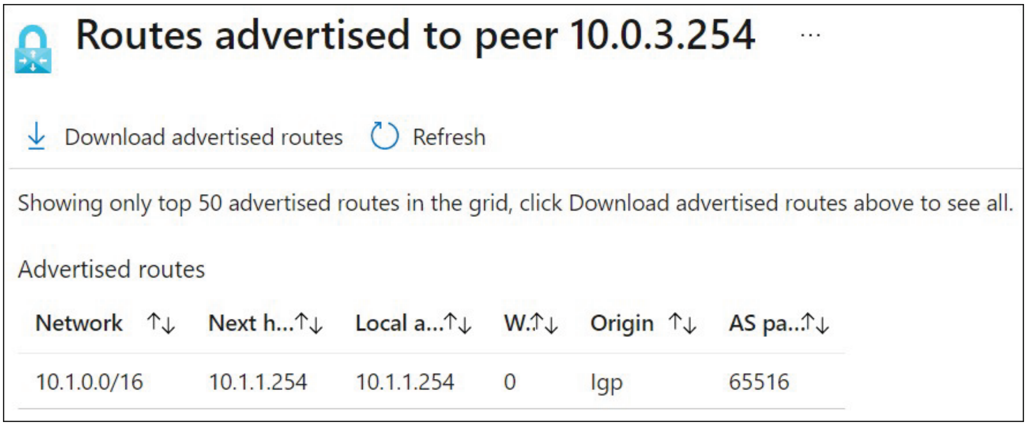
Peer add...	Local ad...	Asn	Status	C...	R...	Me...	M...
10.0.3.254	10.1.1.254	65515	Connected	1.08:3...	3	2244	2261

Showing only top 50 BGP learned routes in the grid, click Download learned routes above to see all.

Learned Routes

Network	Next ho...	Local a...	W...	Source p...	Orig...	AS path
10.1.0.0/16	-	10.1.1.254	32768	10.1.1.254	Network	-
10.0.3.254/32	-	10.1.1.254	32768	10.1.1.254	Network	-
10.0.0.0/16	10.0.3.254	10.1.1.254	32768	10.0.3.254	EBgp	65515
10.11.11.0/24	10.0.3.254	10.1.1.254	32768	10.0.3.254	EBgp	65515-64777
10.12.12.0/24	10.0.3.254	10.1.1.254	32768	10.0.3.254	EBgp	65515-64777-64777

Figure 7-11: *BGP Peers and BGP Routing Information.*


Routes advertised to peer 10.0.3.254

Showing only top 50 advertised routes in the grid, click Download advertised routes above to see all.

Advertised routes

Network	Next h...	Local a...	W.	Origin	AS pa...
10.1.0.0/16	10.1.1.254	10.1.1.254	0	Igp	65516

Figure 7-12: *BGP Route Advertisement.*

Examples 7-1 and 7-2 show the BGP peering status of both VGWs.

```
Get-AzVirtualNetworkGatewayBgpPeerStatus -ResourceGroupName rg-nsg-rt-
swedencentral -VirtualNetworkGatewayName vgw-nwkt

LocalAddress       : 10.0.3.254
Neighbor           : 10.1.1.254
Asn                : 65516
State              : Connected
ConnectedDuration  : 1.09:01:38.4545528
RoutesReceived     : 1
MessagesSent       : 2303
MessagesReceived   : 2285

LocalAddress       : 10.0.3.254
Neighbor           : 192.168.10.1
Asn                : 64777
State              : Connected
ConnectedDuration  : 01:27:27.9069500
RoutesReceived     : 2
MessagesSent       : 113
MessagesReceived   : 113
```

Example 7-1: *BGP Peer Status of vgw-nwkt.*

```
Get-AzVirtualNetworkGatewayBgpPeerStatus -ResourceGroupName rg-nsg-rt-
swedencentral -VirtualNetworkGatewayName vgw-spoke-1

LocalAddress       : 10.1.1.254
Neighbor           : 10.0.3.254
Asn                : 65515
State              : Connected
ConnectedDuration  : 00:01:00.0356311
RoutesReceived     : 3
MessagesSent       : 6
MessagesReceived   : 11
```

Example 7-2: *BGP Peer Status of vgw-spoke-1.*

Examples 7-3 and 7-4 show routes learned from BGP peers.

```
Get-AzVirtualNetworkGatewayLearnedRoute -ResourceGroupName rg-nsg-rt-
swedencentral -VirtualNetworkGatewayname vgw-nwkt

LocalAddress : 10.0.3.254
Network      : 10.0.0.0/16
NextHop      :
SourcePeer   : 10.0.3.254
Origin       : Network
AsPath       :
Weight       : 32768

LocalAddress : 10.0.3.254
Network      : 10.12.12.0/24
NextHop      : 192.168.10.1
SourcePeer   : 192.168.10.1
Origin       : EBgp
AsPath       : 64777-64777
Weight       : 32768

LocalAddress : 10.0.3.254
Network      : 10.1.0.0/16
NextHop      : 10.1.1.254
SourcePeer   : 10.1.1.254
Origin       : EBgp
AsPath       : 65516
Weight       : 32768

LocalAddress : 10.0.3.254
Network      : 192.168.10.1/32
NextHop      :
SourcePeer   : 10.0.3.254
Origin       : Network
AsPath       :
Weight       : 32768

LocalAddress : 10.0.3.254
Network      : 10.11.11.0/24
NextHop      : 192.168.10.1
SourcePeer   : 192.168.10.1
Origin       : EBgp
AsPath       : 64777
Weight       : 32768

LocalAddress : 10.0.3.254
Network      : 10.1.1.254/32
NextHop      :
SourcePeer   : 10.0.3.254
Origin       : Network
AsPath       :
Weight       : 32768
```

Example 7-3: *BGP Learned Routes vgw-nwkt BGP Table.*

```
Get-AzVirtualNetworkGatewayLearnedRoute -ResourceGroupName rg-nsg-rt-
swedencentral -VirtualNetworkGatewayname vgw-spoke-1

LocalAddress : 10.1.1.254
Network      : 10.1.0.0/16
NextHop      :
SourcePeer   : 10.1.1.254
Origin       : Network
AsPath       :
Weight       : 32768

LocalAddress : 10.1.1.254
Network      : 10.0.3.254/32
NextHop      :
SourcePeer   : 10.1.1.254
Origin       : Network
AsPath       :
Weight       : 32768

LocalAddress : 10.1.1.254
Network      : 10.0.0.0/16
NextHop      : 10.0.3.254
SourcePeer   : 10.0.3.254
Origin       : EBgp
AsPath       : 65515
Weight       : 32768

LocalAddress : 10.1.1.254
Network      : 10.12.12.0/24
NextHop      : 10.0.3.254
SourcePeer   : 10.0.3.254
Origin       : EBgp
AsPath       : 65515-64777-64777
Weight       : 32768

LocalAddress : 10.1.1.254
Network      : 10.11.11.0/24
NextHop      : 10.0.3.254
SourcePeer   : 10.0.3.254
Origin       : EBgp
AsPath       : 65515-64777
Weight       : 32768
```
Example 7-4: *BGP Learned Routes- vgw-spoke-1 BGP Table.*

Example 7-5 shows that *lgw-spoke-1* has received a BGP Update about vnet-spoke-1 CIDR 10.1.0.0/16 from vgw-nwkt. The route is valid and it is selected as Best Route.

```
LGW-NWKT#sh ip bgp 10.1.0.0/16
BGP routing table entry for 10.1.0.0/16, version 3
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 1
  65515 65516
    10.0.3.254 from 10.0.3.254 (10.0.3.254)
      Origin IGP, localpref 100, valid, external, best
      rx pathid: 0, tx pathid: 0x0
```

Example 7-5: *lgw-nwkt BGP Table.*

Example 7-6 shows that *lgw-spoke-1* has installed route into the routing table.

```
LGW-NWKT#sh ip route 10.1.0.0 255.255.0.0
Routing entry for 10.1.0.0/16
  Known via "bgp 64777", distance 20, metric 0
  Tag 65515, type external
  Last update from 10.0.3.254 00:49:02 ago
  Routing Descriptor Blocks:
  * 10.0.3.254, from 10.0.3.254, 00:49:02 ago
      Route metric is 0, traffic share count is 1
      AS Hops 2
      Route tag 65515
      MPLS label: none
```

Example 7-6: *lgw-nwkt Routing Table.*

Example 7-7 verifies that also data plane is ok, we have IP connection between remote location and VNet vnet-spoke-1.

```
LGW-NWKT#ping 10.1.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
```

Example 7-7: *lgw-nwkt Routing Table.*

References

[1] How
https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-vnet-vnet-resource-manager-portal

[2] Choosing between Azure VNet Peering and VNet Gateways
https://azure.microsoft.com/en-us/blog/vnet-peering-and-vpn-gateways/

[3] Interoperability in Azure: Test setup
https://docs.microsoft.com/en-us/azure/networking/connectivty-interoperability-preface?toc=%2Fazure%2Fvpn-gateway%2Ftoc.json

Chapter 8: VNet Peering

Introduction

This chapter introduces an Azure VNet Peering solution. VNet peering creates bidirectional IP connections between peered VNets. VNet peering links can be established within and across Azure regions and between VNets under the different Azure subscriptions or tenants. The unencrypted data path over peer links stays within Azure private infrastructure. Consider a software-level solution (or use VGW) if your security policy requires data path encryption. There is no bandwidth limitation in VNet Peering like in VGW, where BW is based on SKU. From the VM perspective, VNet peering gives seamless network performance (bandwidth, latency, delay, and jitter) for Inter-VNet and Intra-VNet traffic. Unlike VGW solution, VNet peering is a non-transitive solution, the routing information learned from one VNet peer is not advertised to another VNet peer. However, we can permit peered VNets (Spokes) to use local VGW (Hub) and route Spoke-to-Spoke data by using a subnet-specific route table (chapter 9 explains the concept in detail). Note that by deploying a VNet peering, we create a bidirectional, always-on IP data path between VNets. However, we can prevent traffic from crossing the link if needed without deleting the peering. Azure uses Virtual Network Service Tags for VNet peering traffic policy.

Figure 8-1 shows our example topology. We create a VNet Peering between *vnet-spoke-2* and *vnet-nsg-rt-swedencentral*. Besides the Inter-VNet connection, our solution allows *vnet-spoke-2* to use *vnet-nsg-rt-swedencentral* as a transit VNet to other peered VNets (which we don't have in this example). We also permit IP connection to/from *vnet-spoke-2* to *vnet-spoke-1* and on-prem location by authorizing vnet-spoke-2 to use *vgw-nwkt* as a transit gateway.

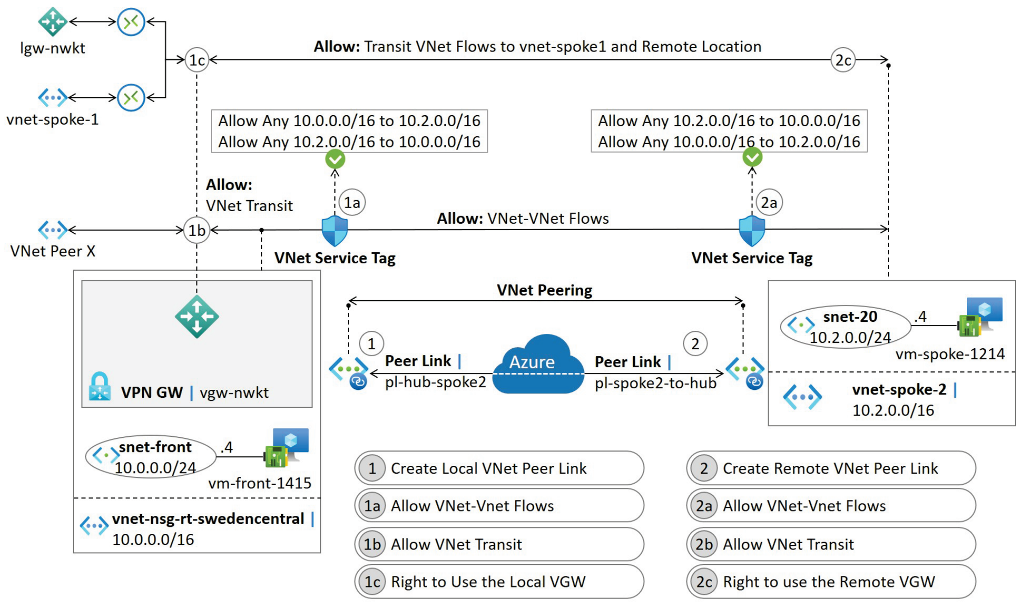
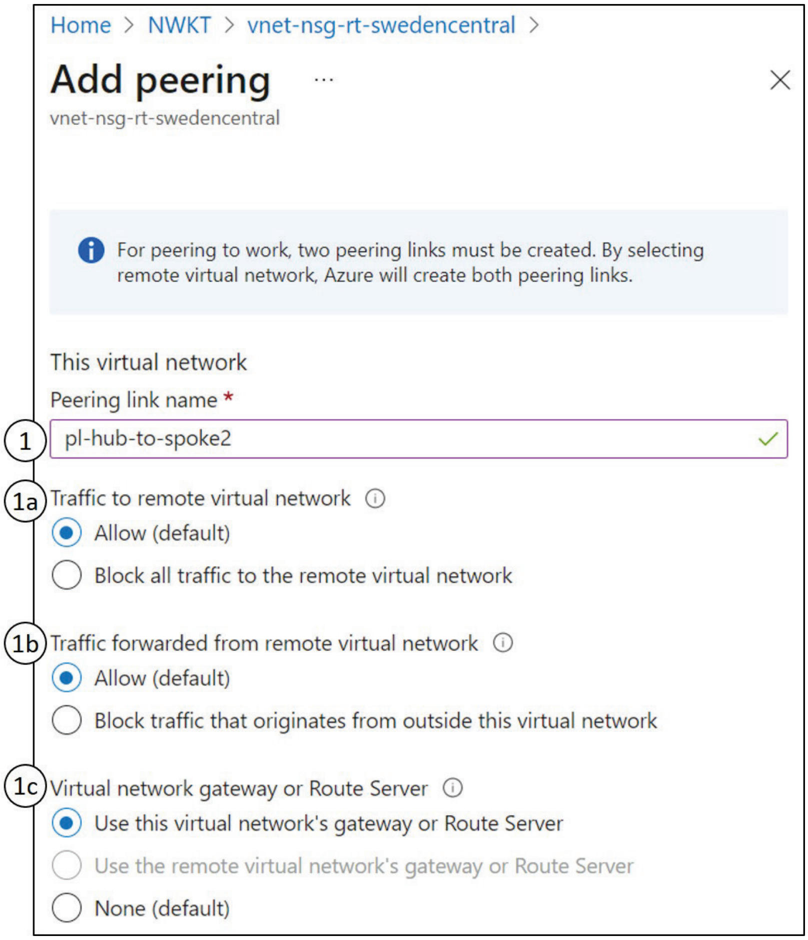

lgw-nwkt
vnet-spoke-1
1c
Allow: Transit VNet Flows to vnet-spoke1 and Remote Location
2c
Allow Any 10.0.0.0/16 to 10.2.0.0/16
Allow Any 10.2.0.0/16 to 10.0.0.0/16
Allow Any 10.2.0.0/16 to 10.0.0.0/16
Allow Any 10.0.0.0/16 to 10.2.0.0/16
Allow:
VNet Transit
1a
2a
Allow: VNet-VNet Flows
VNet Peer X
1b
VNet Service Tag
VNet Service Tag
VNet Peering
1
2
Peer Link | pl-hub-spoke2
Azure
Peer Link | pl-spoke2-to-hub
VPN GW | vgw-nwkt
snet-front 10.0.0.0/24
.4
vm-front-1415
vnet-nsg-rt-swedencentral | 10.0.0.0/16
snet-20 10.2.0.0/24
.4
vm-spoke-1214
vnet-spoke-2 | 10.2.0.0/16
1 Create Local VNet Peer Link
1a Allow VNet-Vnet Flows
1b Allow VNet Transit
1c Right to Use the Local VGW
2 Create Remote VNet Peer Link
2a Allow VNet-Vnet Flows
2b Allow VNet Transit
2c Right to use the Remote VGW

Figure 8-1: *VNet Peering Example Diagram.*

Deploy VNet Peering

Figures 8-2, 8-3, and 8-4 are all part of the same *Add Peering* view. The VNet peering configuration is divided into Local and remote VNet sections. We create VNet peering under the VNet *vnet-nsg-rt-swedencentral* peering configuration. To start the configuration, navigate to the VNet resource page and select *the Peerings* option under the *Settings* section. Then click the **+** *Add* selector on the menu bar. After naming the link (*pl-hub-to-spoke2*) we define the traffic policy. (1a) The *Allow* option under the *Traffic to remote virtual network* modifies the VirtualNetwork service tag and allows traffic to remote VNet. (1b-c) VNet vnet-*nsg-rt-swedencentral* is our Hub VNet, and we permit data flows from remote VNet to other peered VNets and to VNets and remote-sites which VPN connection is terminated to VGW vgw-nwkt on the local VNet. Choose the *Allow* selector under the *Traffic forwarded from remote virtual network* option and *Use this virtual network's gateway or Route Servers* under the *Virtual network gateway or Route Server*.


Home > NWKT > vnet-nsg-rt-swedencentral >

Add peering ...

vnet-nsg-rt-swedencentral

For peering to work, two peering links must be created. By selecting remote virtual network, Azure will create both peering links.

This virtual network

Peering link name *

1 pl-hub-to-spoke2 ✓

1a Traffic to remote virtual network

[✓ Allow (default)]

Block all traffic to the remote virtual network

1b Traffic forwarded from remote virtual network

[✓ Allow (default)]

Block traffic that originates from outside this virtual network

1c Virtual network gateway or Route Server

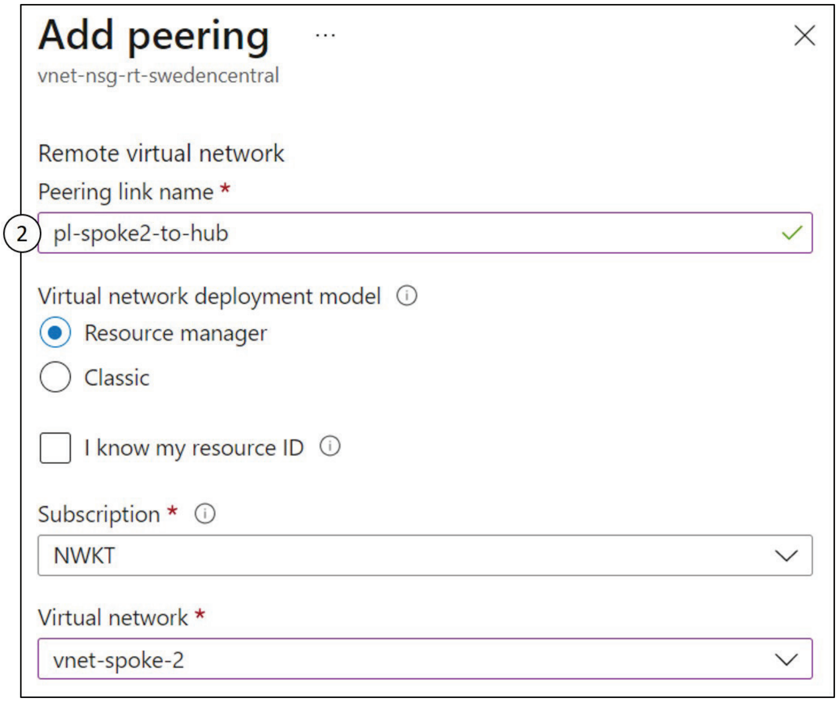
[✓ Use this virtual network's gateway or Route Server]

Use the remote virtual network's gateway or Route Server

None (default)

Figure 8-2: *VNet Peering Deployment – Local VNet.*

Give the name for the Peering link on the remote VNet. We use the default VNet deployment model with the *Resource manager*. The remote VNet is in our subscription, and we have read access to it. That is why we leave the *I know my resource ID* option unchecked. If we don't have read access to remote VNet resources, we need to choose the *I know my resource ID* option. This selection opens a new field where you can add a complete resource ID. You can easily copy the resource ID from the *Overview* window of remote VNet and then select the *JSON view* and copy the resource ID. In our case the full resource ID is: */subscriptions/xxx/resourceGroups/rg-nsg-rt-swedencentral/providers/Microsoft.Network/virtualNetworks/vnet-spoke-2*. Next, Select your subscription and VNet from the drop-down menus.


Add peering
vnet-nsg-rt-swedencentral

Remote virtual network

Peering link name *
(2) pl-spoke2-to-hub ✓

Virtual network deployment model
[● Resource manager]
○ Classic

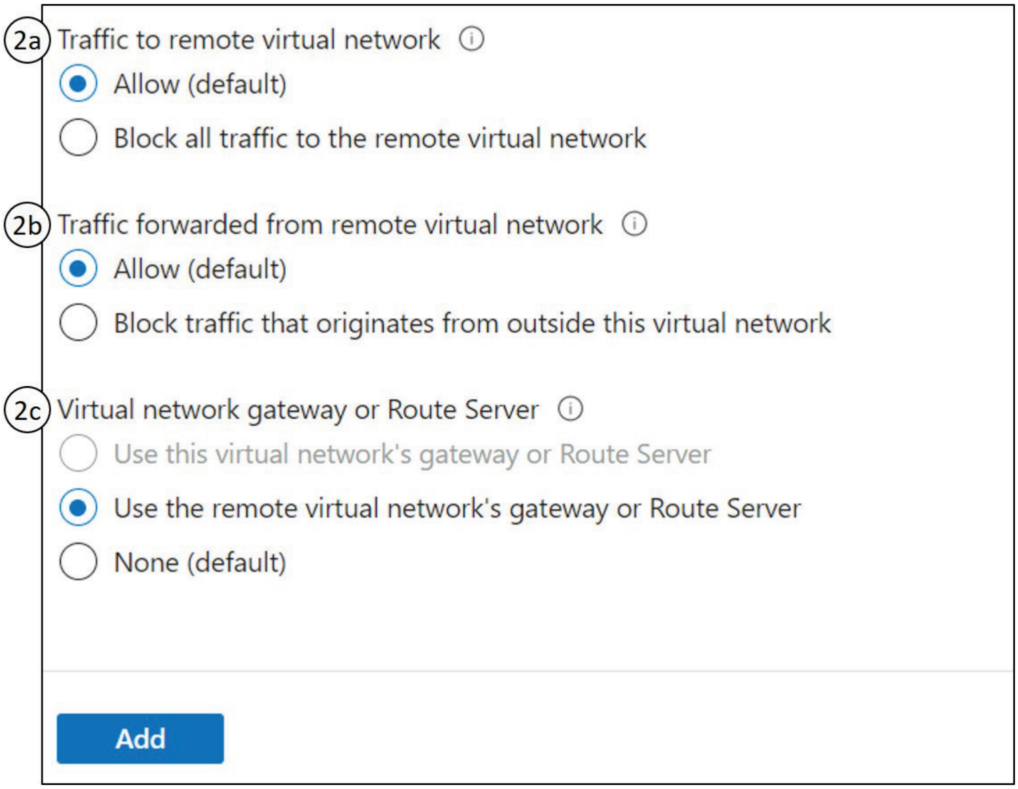
☐ I know my resource ID

Subscription *
NWKT

Virtual network *
vnet-spoke-2

Figure 8-3: *VNet Peering Deployment – Remote VNet.*

The *Traffic to a remote virtual network (2a)* and *Traffic forwarded from a remote virtual network (2b)* options control the same policies as in the case of local VNet. However, in our example, we don't have any additional peering or VPN connection from the vnet-spoke-2, so it's ok to choose the *Block traffic originating from outside this virtual network* option. The last setting Virtual network gateway or Route Server has to be Use the remote virtual network's gateway or Route Server because we want to open a bidirectional data path from vnet-spoke-2 to vnet-spoke-1 and on-prem DC over vgw-nwkt.


2a Traffic to remote virtual network

[✓ Allow (default)] Block all traffic to the remote virtual network

2b Traffic forwarded from remote virtual network

[✓ Allow (default)] Block traffic that originates from outside this virtual network

2c Virtual network gateway or Route Server

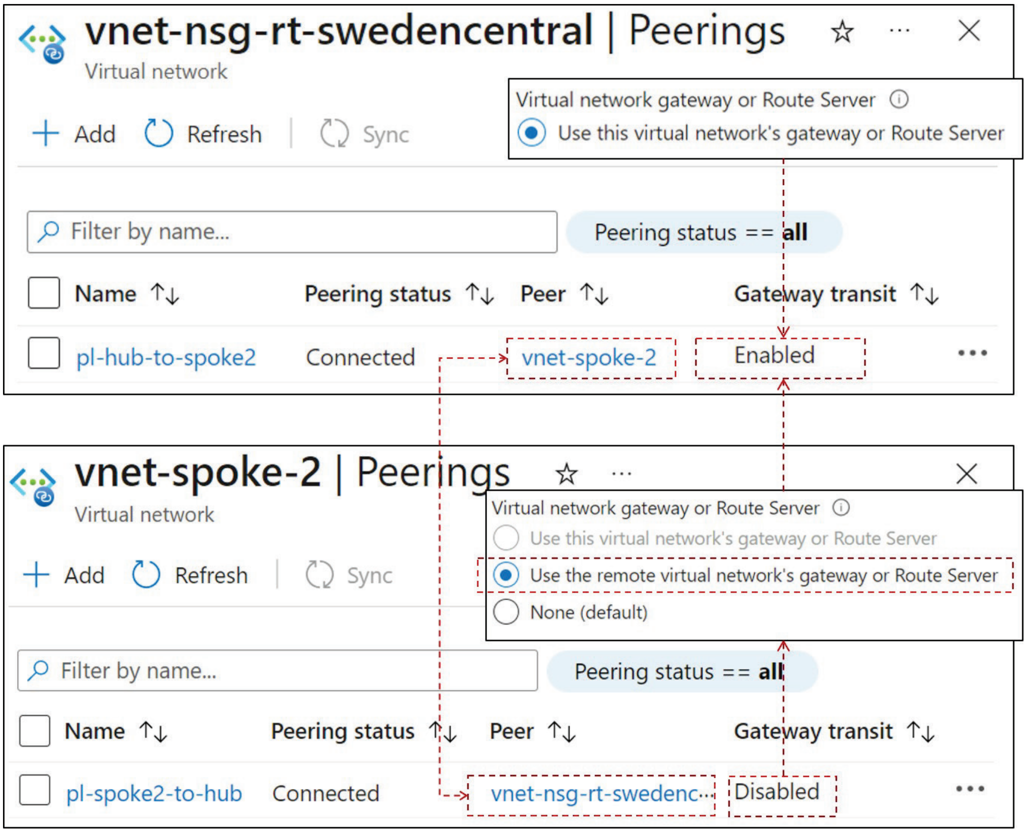
Use this virtual network's gateway or Route Server [✓ Use the remote virtual network's gateway or Route Server] None (default)

Figure 8-4: *VNet Peering Deployment – Remote VNet.*

Verification

VNet Peering

The figure below shows the VNet peering between vnet-nsg-rt-swedencentral (hub VNet) and vnet-spoke-2 (spoke VNet). We have enabled The Gateway transit service on a hub. Besides, we allowed the spoke VNet to use. You can click the Peer link name to navigate the Peer link window.


vnet-nsg-rt-swedencentral | Peerings
Virtual network
Virtual network gateway or Route Server: [✓ Use this virtual network's gateway or Route Server]

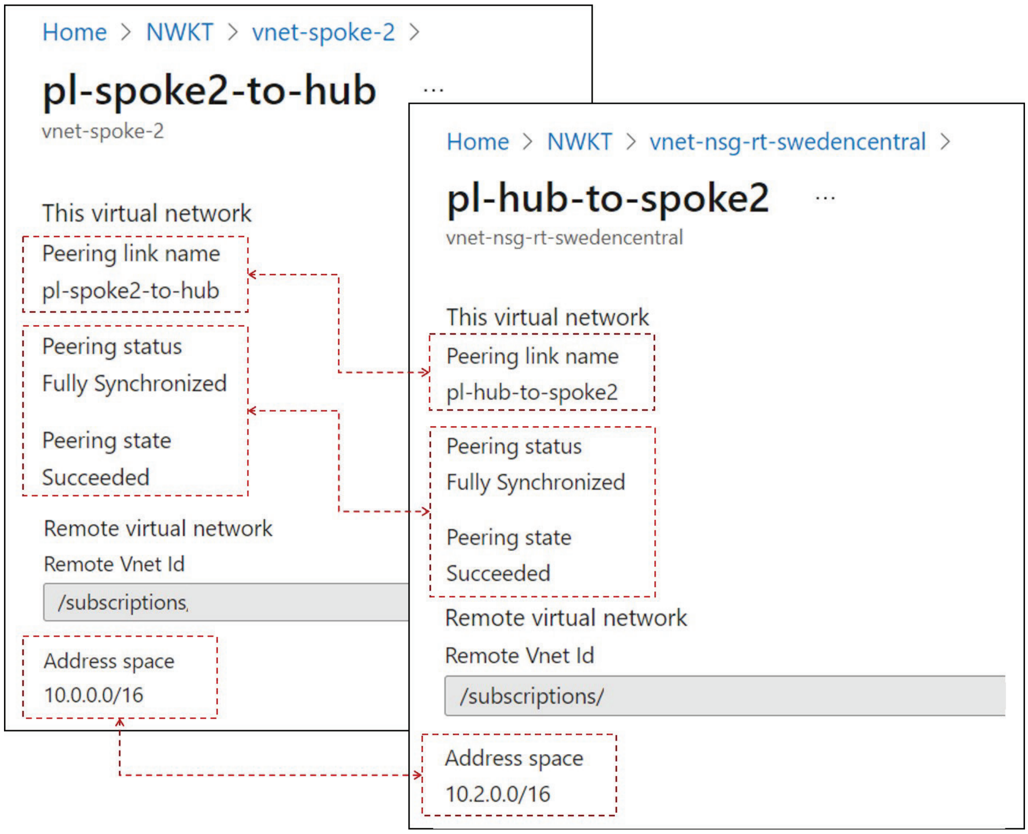
Name	Peering status	Peer	Gateway transit
pl-hub-to-spoke2	Connected	vnet-spoke-2	Enabled

vnet-spoke-2 | Peerings
Virtual network
Virtual network gateway or Route Server: Use this virtual network's gateway or Route Server [✓ Use the remote virtual network's gateway or Route Server] None (default)

Name	Peering status	Peer	Gateway transit
pl-spoke2-to-hub	Connected	vnet-nsg-rt-swedenc…	Disabled

Figure 8-5: *VNet Peering Verification.*

Figure 8-6 shows the Peer link view. We can see that the state of the VNet peering is *Succeeded*, the status is fully synchronized, and the two-way CIDR range exchange has happened.



Home > NWKT > vnet-spoke-2 >

pl-spoke2-to-hub ...

vnet-spoke-2

This virtual network

Peering link name

pl-spoke2-to-hub

Peering status

Fully Synchronized

Peering state

Succeeded

Remote virtual network

Remote Vnet Id

/subscriptions,

Address space

10.0.0.0/16

Home > NWKT > vnet-nsg-rt-swedencentral >

pl-hub-to-spoke2 ...

vnet-nsg-rt-swedencentral

This virtual network

Peering link name

pl-hub-to-spoke2

Peering status

Fully Synchronized

Peering state

Succeeded

Remote virtual network

Remote Vnet Id

/subscriptions/

Address space

10.2.0.0/16

Figure 8-6: *VNet Peering Verification.*

You can also check the VNet peering information using the command *Get-AzVirtualNetworkPeering*. Example 8-1 shows the VNet peering information from the *vnet-spoke-2* perspective, and example 8-2 from the *vnet-nsg-rt-swedencentral* point of view.

```
Get-AzVirtualNetworkPeering

cmdlet Get-AzVirtualNetworkPeering at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
VirtualNetworkName: vnet-spoke-2
ResourceGroupName: rg-nsg-rt-swedencentral

Name                             : pl-Spoke2-toHub
Id                               : /subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworks/vnet-spoke-
2/virtualNetworkPeerings/pl-Spoke2-toHub
Etag                             : W/"f17a45b1-5a5e-44f3-b238-a128f518fa93"
ResourceGroupName                : rg-nsg-rt-swedencentral
VirtualNetworkName               : vnet-spoke-2
PeeringSyncLevel                 : FullyInSync
PeeringState                     : Connected
ProvisioningState                : Succeeded
RemoteVirtualNetwork             : {
                                     "Id": "/subscriptions/xxx/resourceGroups/rg-nsg-
rt-swedencentral/providers/Microsoft.Network/virtualNetworks/vnet-nsg-rt-
swedencentral"
                                   }
AllowVirtualNetworkAccess        : True
AllowForwardedTraffic            : True
AllowGatewayTransit              : False
UseRemoteGateways                : True
RemoteGateways                   : null
PeeredRemoteAddressSpace         : {
                                     "AddressPrefixes": [
                                       "10.0.0.0/16"
                                     ]
                                   }
RemoteVirtualNetworkAddressSpace : {
                                     "AddressPrefixes": [
                                       "10.0.0.0/16"
                                     ]
                                   }
```

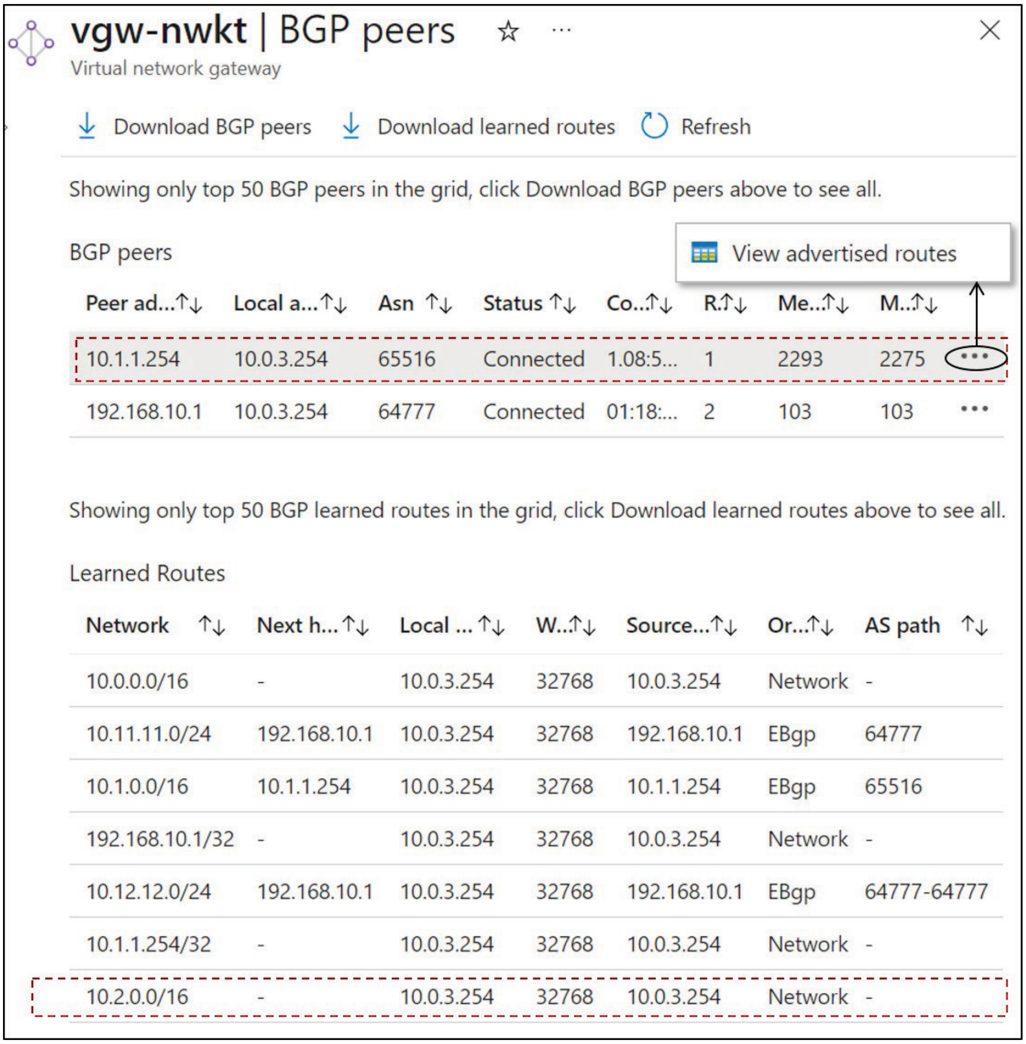
Example 8-1: *The Command "Get-AzVirtualNetworkPeering".*

```
Get-AzVirtualNetworkPeering

cmdlet Get-AzVirtualNetworkPeering at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
VirtualNetworkName: vnet-nsg-rt-swedencentral
ResourceGroupName: rg-nsg-rt-swedencentral

Name                             : pl-Hub-to-Spoke2
Id                               : /subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworks/vnet-nsg-rt-
swedencentral/virtualNetworkPeerings/pl-Hub-to-Spoke2
Etag                             : W/"4b42c147-94a2-408d-af74-1a0eac23b51a"
ResourceGroupName                : rg-nsg-rt-swedencentral
VirtualNetworkName               : vnet-nsg-rt-swedencentral
PeeringSyncLevel                 : FullyInSync
PeeringState                     : Connected
ProvisioningState                : Succeeded
RemoteVirtualNetwork             : {
                                     "Id": "/subscriptions/xxx/resourceGroups/rg-nsg-
rt-swedencentral/providers/Microsoft.Network/virtualNetworks/vnet-spoke-2"
                                   }
AllowVirtualNetworkAccess        : True
AllowForwardedTraffic            : True
AllowGatewayTransit              : True
UseRemoteGateways                : False
RemoteGateways                   : null
PeeredRemoteAddressSpace         : {
                                     "AddressPrefixes": [
                                       "10.2.0.0/16"
                                     ]
                                   }
RemoteVirtualNetworkAddressSpace : {
                                     "AddressPrefixes": [
                                       "10.2.0.0/16"
                                     ]
                                   }
```

Example 8-2: *The Command "Get-AzVirtualNetworkPeering".*

Control Plane - BGP

Figure 8-7 verifies that vgw-nwkt has received the route to CIDR 10.2.0.0/16 and installed it in the BGP table. There is no BGP next-hop path attribute associated with network 10.2.0.0/16, and vgw-nwkt lists its BGP peering address as a local and source address. Based on these, we can assume that VNet peering Control Plane is not based on BGP, and the information is pulled to the BGP table from the RIB (Routing Information Base/Routing Table). In most traditional routers, routes from the RIB can be imported to the BGP table and advertised to BGP peers using the *network* command.



Showing only top 50 BGP peers in the grid, click Download BGP peers above to see all.

BGP peers

Peer ad...	Local a...	Asn	Status	Co...	R.	Me...	M...	
10.1.1.254	10.0.3.254	65516	Connected	1.08:5...	1	2293	2275	•••
192.168.10.1	10.0.3.254	64777	Connected	01:18:...	2	103	103	•••

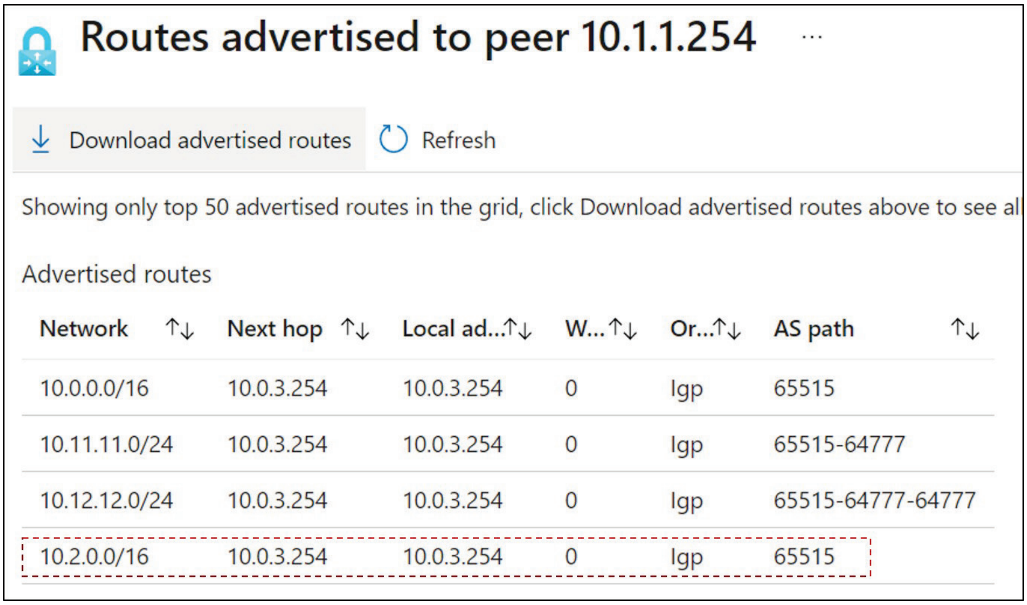
Showing only top 50 BGP learned routes in the grid, click Download learned routes above to see all.

Learned Routes

Network	Next h...	Local ...	W...	Source...	Or...	AS path
10.0.0.0/16	-	10.0.3.254	32768	10.0.3.254	Network	-
10.11.11.0/24	192.168.10.1	10.0.3.254	32768	192.168.10.1	EBgp	64777
10.1.0.0/16	10.1.1.254	10.0.3.254	32768	10.1.1.254	EBgp	65516
192.168.10.1/32	-	10.0.3.254	32768	10.0.3.254	Network	-
10.12.12.0/24	192.168.10.1	10.0.3.254	32768	192.168.10.1	EBgp	64777-64777
10.1.1.254/32	-	10.0.3.254	32768	10.0.3.254	Network	-
10.2.0.0/16	-	10.0.3.254	32768	10.0.3.254	Network	-

Figure 8-7: *Control Plane Verification on VGW.*

Figures 8-8 and 8-9 verifies that vgw-nwkt has sent BGP Update messages about network 10.2.0.0/16 to its BGP peers 10.1.1.254 (vnet-spoke-1) and 192.168.10.1 (lgw-nwkt).



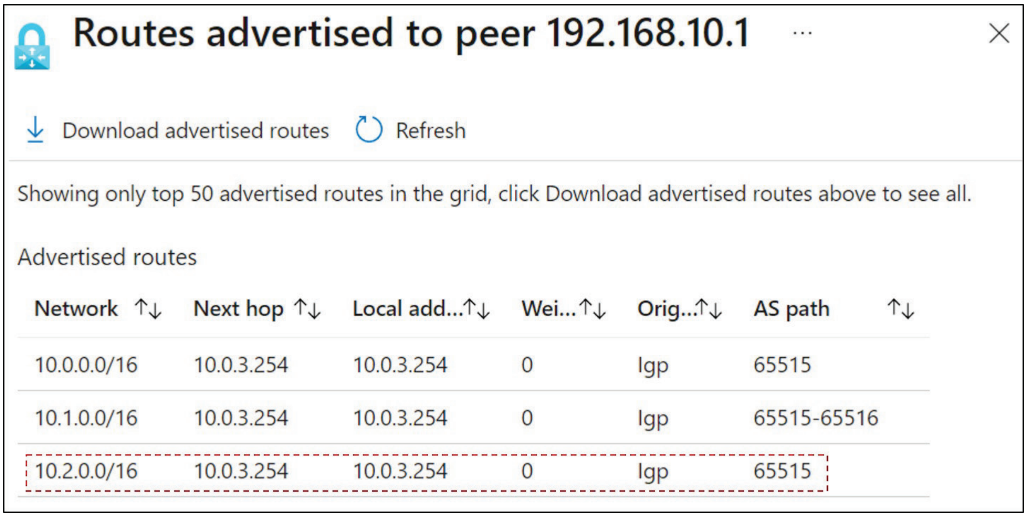
Routes advertised to peer 10.1.1.254

Showing only top 50 advertised routes in the grid, click Download advertised routes above to see al

Advertised routes

Network	Next hop	Local ad...	W...	Or...	AS path
10.0.0.0/16	10.0.3.254	10.0.3.254	0	Igp	65515
10.11.11.0/24	10.0.3.254	10.0.3.254	0	Igp	65515-64777
10.12.12.0/24	10.0.3.254	10.0.3.254	0	Igp	65515-64777-64777
10.2.0.0/16	10.0.3.254	10.0.3.254	0	Igp	65515

Figure 8-8: *Advertised Routes to 10.1.1.254 on vnet-spoke-1.*



Routes advertised to peer 192.168.10.1

Showing only top 50 advertised routes in the grid, click Download advertised routes above to see all.

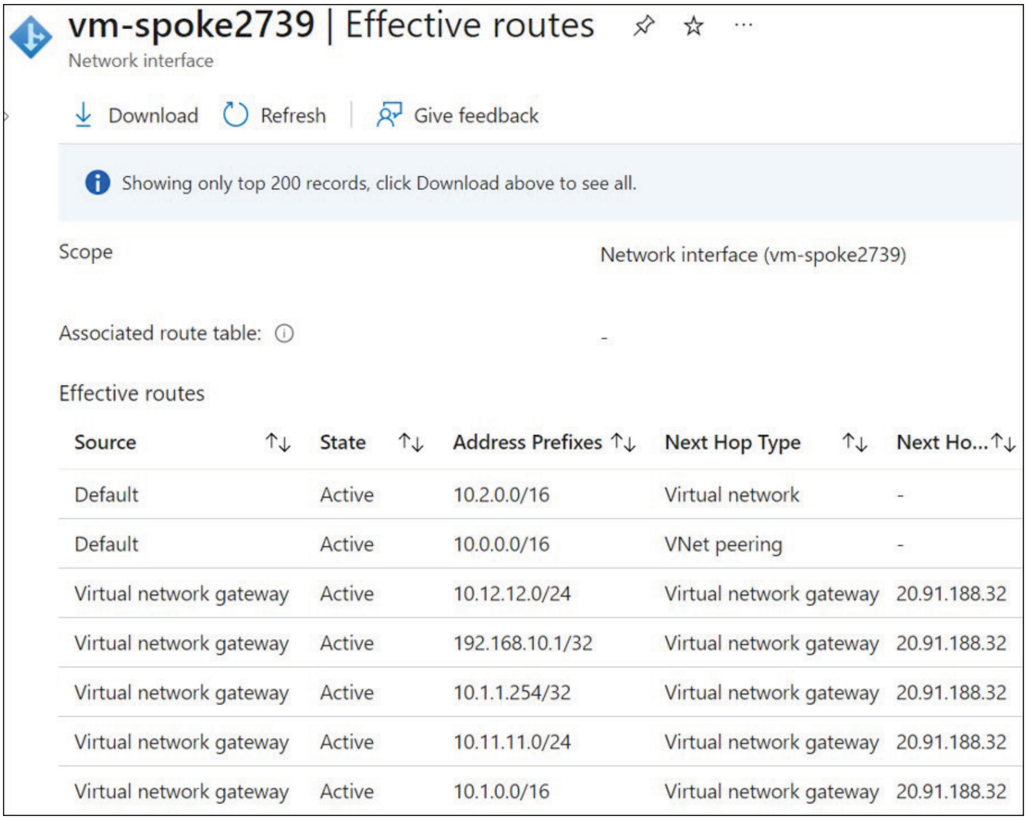
Advertised routes

Network	Next hop	Local add...	Wei...	Orig...	AS path
10.0.0.0/16	10.0.3.254	10.0.3.254	0	Igp	65515
10.1.0.0/16	10.0.3.254	10.0.3.254	0	Igp	65515-65516
10.2.0.0/16	10.0.3.254	10.0.3.254	0	Igp	65515

Figure 8-9: *Advertised Routes to 192.168.10.1 (lgw-nwkt).*

The RIB Associated with NIC vm-spoke2739

Figure 8-10 shows routing entries on NIC vm-spoke2739 (attached to vm-spoke2). CIDR 10.0.0.0/16 has been learned over VNet peering, while all other external subnets are learned via VGW 20.91.188.32 (vgw-nwkt).



Showing only top 200 records, click Download above to see all.

Scope: Network interface (vm-spoke2739)

Associated route table: -

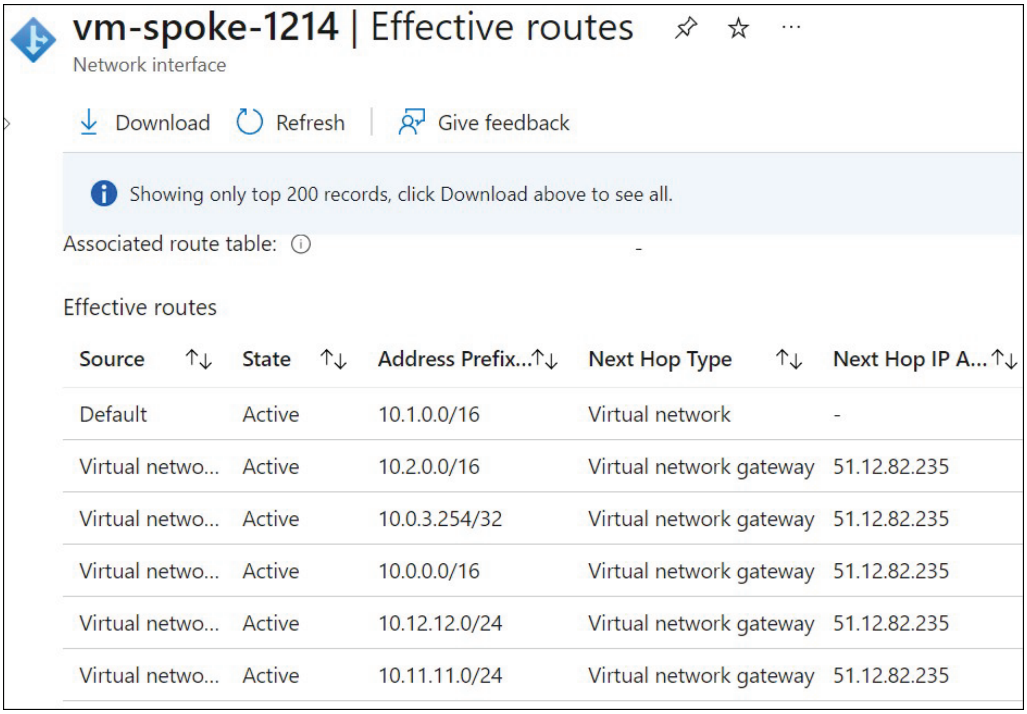
Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Ho...
Default	Active	10.2.0.0/16	Virtual network	-
Default	Active	10.0.0.0/16	VNet peering	-
Virtual network gateway	Active	10.12.12.0/24	Virtual network gateway	20.91.188.32
Virtual network gateway	Active	192.168.10.1/32	Virtual network gateway	20.91.188.32
Virtual network gateway	Active	10.1.1.254/32	Virtual network gateway	20.91.188.32
Virtual network gateway	Active	10.11.11.0/24	Virtual network gateway	20.91.188.32
Virtual network gateway	Active	10.1.0.0/16	Virtual network gateway	20.91.188.32

Figure 8-10: *Routing Entries in RIB Associated with the NIC vm-spoke2739.*

The RIB Associated with NIC vm-spoke1214

Figure 8-10 shows routing entries on NIC vm-spoke1214 (attached to vm-spoke1). All external subnets are learned via VGW 51.12.82.235 (vgw-spoke-1). Remember that vnet-spoke-1 has VPN connection with vgw-nwkt, and they BGP peering address uses public IP addresses though the data path stays in Azure infrastructure.



Showing only top 200 records, click Download above to see all.

Associated route table: -

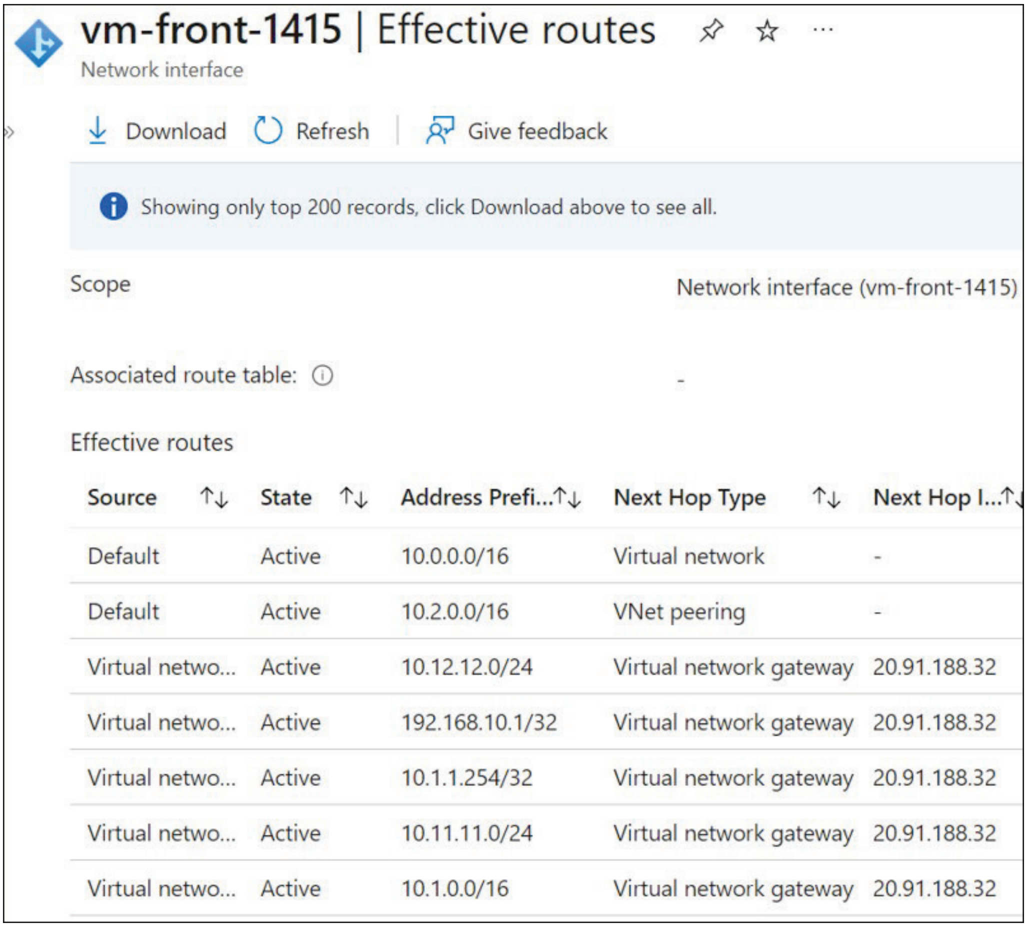
Effective routes

Source	State	Address Prefix...	Next Hop Type	Next Hop IP A...
Default	Active	10.1.0.0/16	Virtual network	-
Virtual netwo...	Active	10.2.0.0/16	Virtual network gateway	51.12.82.235
Virtual netwo...	Active	10.0.3.254/32	Virtual network gateway	51.12.82.235
Virtual netwo...	Active	10.0.0.0/16	Virtual network gateway	51.12.82.235
Virtual netwo...	Active	10.12.12.0/24	Virtual network gateway	51.12.82.235
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	51.12.82.235

Figure 8-11: *Routing Entries in RIB Associated with the NIC vm-spoke1214.*

The RIB Associated with NIC vm-front1415

Figure 8-12 shows routing entries on NIC vm-front1415 (attached to vm-front-1). CIDR 10.2.0.0/16 has been learned over VNet peering, while all other external subnets are learned via VGW 20.91.188.32 (vgw-nwkt).



vm-front-1415 | Effective routes

Network interface

Download Refresh Give feedback

Showing only top 200 records, click Download above to see all.

Scope: Network interface (vm-front-1415)

Associated route table: -

Effective routes

Source	State	Address Prefi...	Next Hop Type	Next Hop I...
Default	Active	10.0.0.0/16	Virtual network	-
Default	Active	10.2.0.0/16	VNet peering	-
Virtual netwo...	Active	10.12.12.0/24	Virtual network gateway	20.91.188.32
Virtual netwo...	Active	192.168.10.1/32	Virtual network gateway	20.91.188.32
Virtual netwo...	Active	10.1.1.254/32	Virtual network gateway	20.91.188.32
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	20.91.188.32
Virtual netwo...	Active	10.1.0.0/16	Virtual network gateway	20.91.188.32

Figure 8-12: *Routing Entries in RIB Associated with the NIC vm-spoke1214.*

Data Plane Verification

Examples 8-3, 8-4, and 8-5 verify that the Data Plane is ok.

```
nwktAdmin@vm-spoke-1:~$ ping 10.2.0.4
PING 10.2.0.4 (10.2.0.4) 56(84) bytes of data.
64 bytes from 10.2.0.4: icmp_seq=1 ttl=64 time=4.22 ms
64 bytes from 10.2.0.4: icmp_seq=2 ttl=64 time=3.33 ms
64 bytes from 10.2.0.4: icmp_seq=3 ttl=64 time=3.69 ms
64 bytes from 10.2.0.4: icmp_seq=4 ttl=64 time=3.46 ms
^C
--- 10.2.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 3.332/3.674/4.216/0.338 ms
nwktAdmin@vm-spoke-1:~$
```

Example 8-3: *Ping from vm-spoke-1 to vm-spoke-2 (10.2.0.4) on vnet-spoke-2.*

```
nwktAdmin@vm-spoke-1:~$ ping 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=3.61 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=4.17 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=5.54 ms
^C
--- 10.0.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 3.612/4.439/5.539/0.809 ms
```

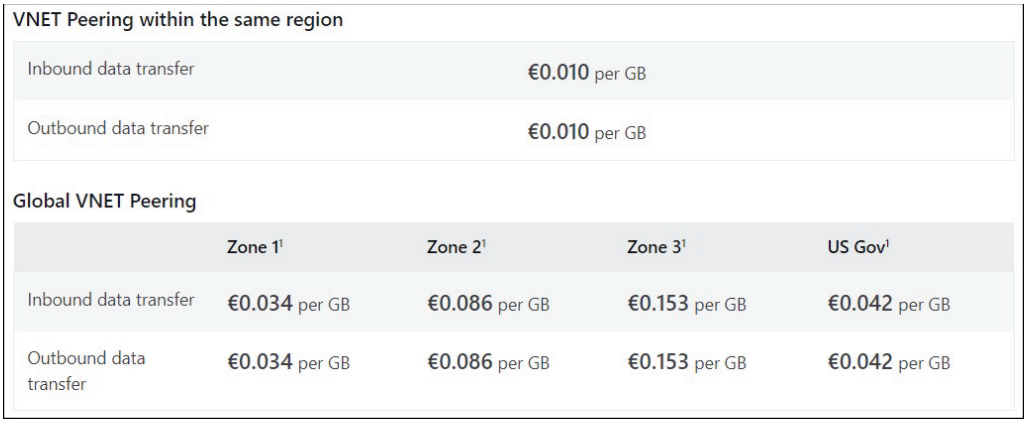
Example 8-4: *Ping from vm-spoke-1 to vm-front (10.0.0.4) on vnet-nsg-rt-swedencentral.*

```
nwktAdmin@vm-spoke-1:~$ ping 10.11.11.1
PING 10.11.11.1 (10.11.11.1) 56(84) bytes of data.
64 bytes from 10.11.11.1: icmp_seq=1 ttl=255 time=15.1 ms
64 bytes from 10.11.11.1: icmp_seq=2 ttl=255 time=13.6 ms
64 bytes from 10.11.11.1: icmp_seq=3 ttl=255 time=13.7 ms
^C
--- 10.11.11.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 13.636/14.146/15.095/0.671 ms
nwktAdmin@vm-spoke-1:~$
```

Example 8-5: *Ping from vm-spoke-1 to host 10.11.11.1 on on on-prem DC.*

Pricing

Figure 8-13 shows the VNet peering pricing. Be aware that Amazon charges Inbound and Outbound at both ends of peered VNets. This means that 1 GB sent across the peer link, costs 0.010€ (Outbound) + 0.010€ (Inbound) = 0.020€.



VNET Peering within the same region

Inbound data transfer	€0.010 per GB
Outbound data transfer	€0.010 per GB

Global VNET Peering

	Zone 1[1]	Zone 2[1]	Zone 3[1]	US Gov[1]
Inbound data transfer	€0.034 per GB	€0.086 per GB	€0.153 per GB	€0.042 per GB
Outbound data transfer	€0.034 per GB	€0.086 per GB	€0.153 per GB	€0.042 per GB

Figure 8-13: *VNet Peering Pricing.*

References

[1] Virtual network peering
https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-peering-overview

[2] Create a transit VNet using VNet peering
https://azure.microsoft.com/en-us/blog/create-a-transit-vnet-using-vnet-peering/

[3] Create, change, or delete a virtual network peering
https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-manage-peering

[4] Virtual network service tags
https://docs.microsoft.com/en-us/azure/virtual-network/service-tags-overview

[5] Get-AzNetworkServiceTag
https://docs.microsoft.com/en-us/powershell/module/az.network/get-aznetworkservicetag?view=azps-8.1.0

[6] Virtual Network pricing
https://azure.microsoft.com/en-us/pricing/details/virtual-network/

Chapter 9: Transit VNet – Hub and Spoke

Introduction

The previous chapter introduced a VNet Peering. VNet Peering alone is a non-transitive solution, which only allows IP connection between peered VNets. However, we can build a design where we configure one of the VNets as a Hub VNet, which has VNet peering with Spoke VNets. Then, we set up a gateway (VGW, Route Server, FireWall...) into the Hub VNet, which routes traffic between Spoke VNets. Besides, we need to create subnet-specific *Route Tables*, where we add User Defined Routes (UDR). Figure 9-1 shows our example topology. We enable IP connectivity between VNets *vnet-west* and *vnet-east* over the Hub VNet *vnet-hub*. The Virtual Network Gateway (vgw-hub) and all Peer links are pre-deployed. Peer link settings, by default, allow traffic flows to and from remote VNet. As the first step, we permit remote VNets use the VGG vgw-hub on the vnet-hub (1a). Then we allow the spoke VNets vnet-west and vnet-east to use the VGW (1b-c). Then we create a Route Table *rt-snet-west* (2a), and attach it to *snet-west* on *vnet-west* (2b). We also add an UDR entry, which routes traffic to destination 10.3.0.0/16 (vnet-east) to remote VGW (2c). The same routing procedure is repeated on vnet-east.

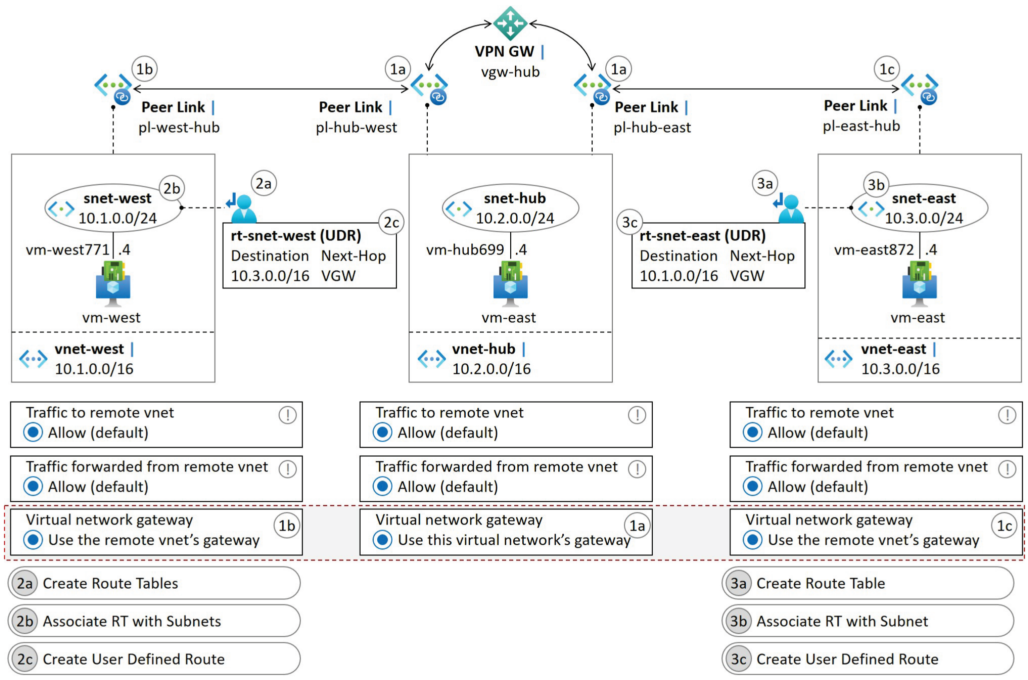


Figure 9-1: *Spoke-to-Spoke Traffic Over Hub VNet.*

Configuration

Create Route Table

Select the *Create a resource* from the Azure portal Home view. Then type the *Route table* on the search field. Next, select the Microsoft *Route table* and click the *Create* button.

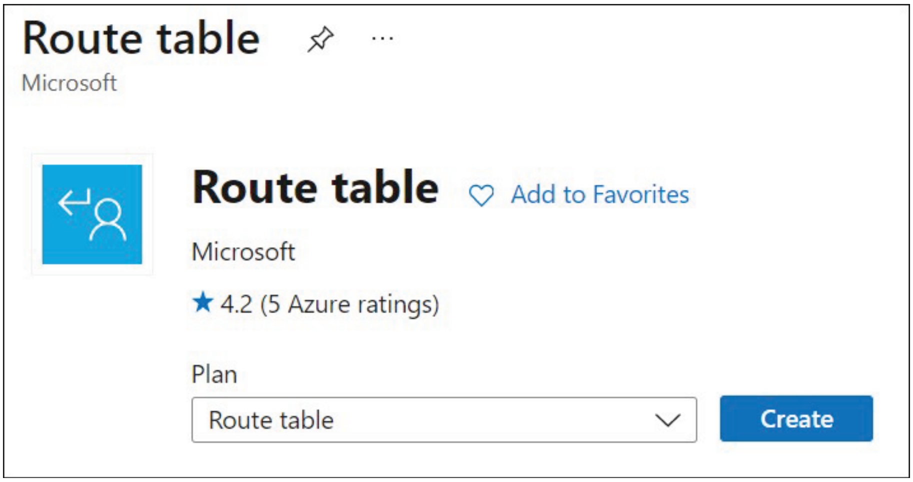
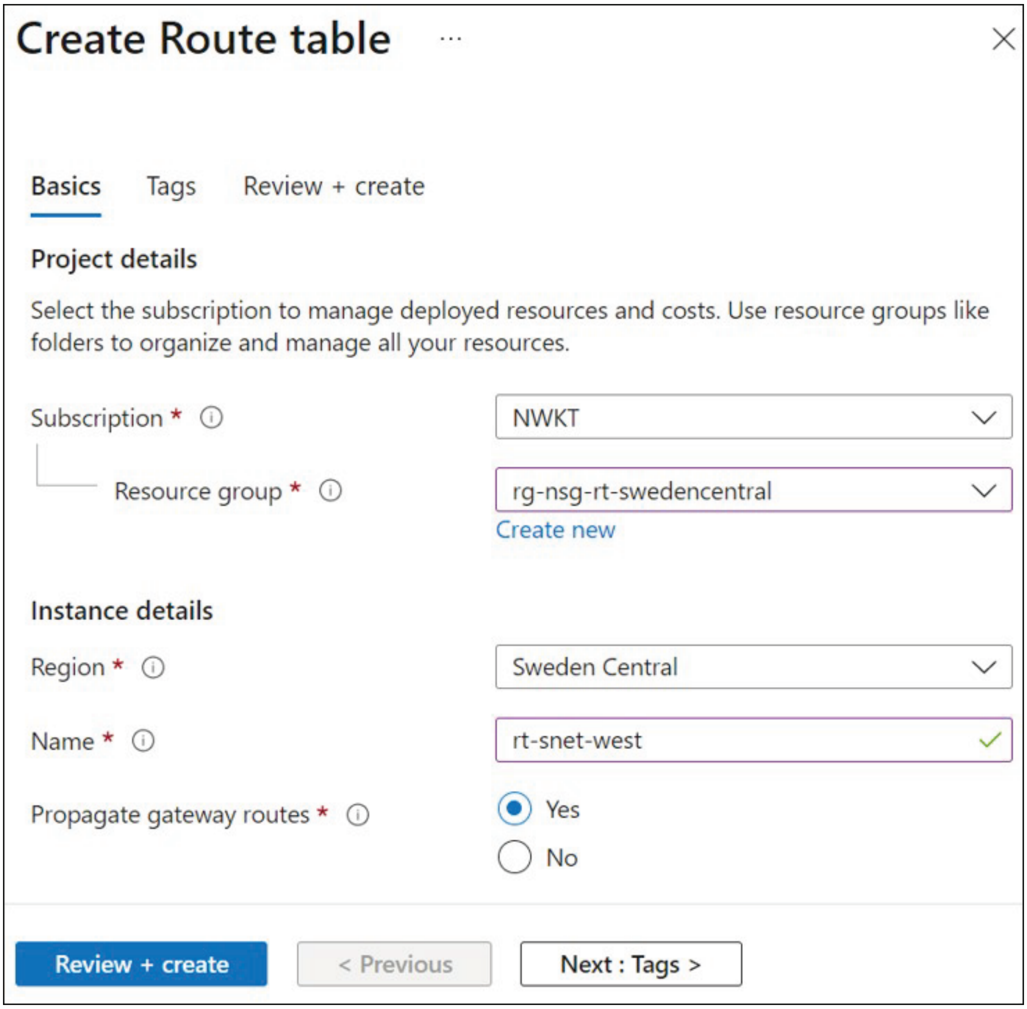


Figure 9-2: *Create Route Table: Step-1.*

First, Select your Subscription and Resource Group. Then choose the region and name the Route Table. Click the Review + Create button to proceed to the preview window. If the validation is ok, deploy the Route Table by clicking the *Create* button.



Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	NWKT
Resource group *	rg-nsg-rt-swedencentral

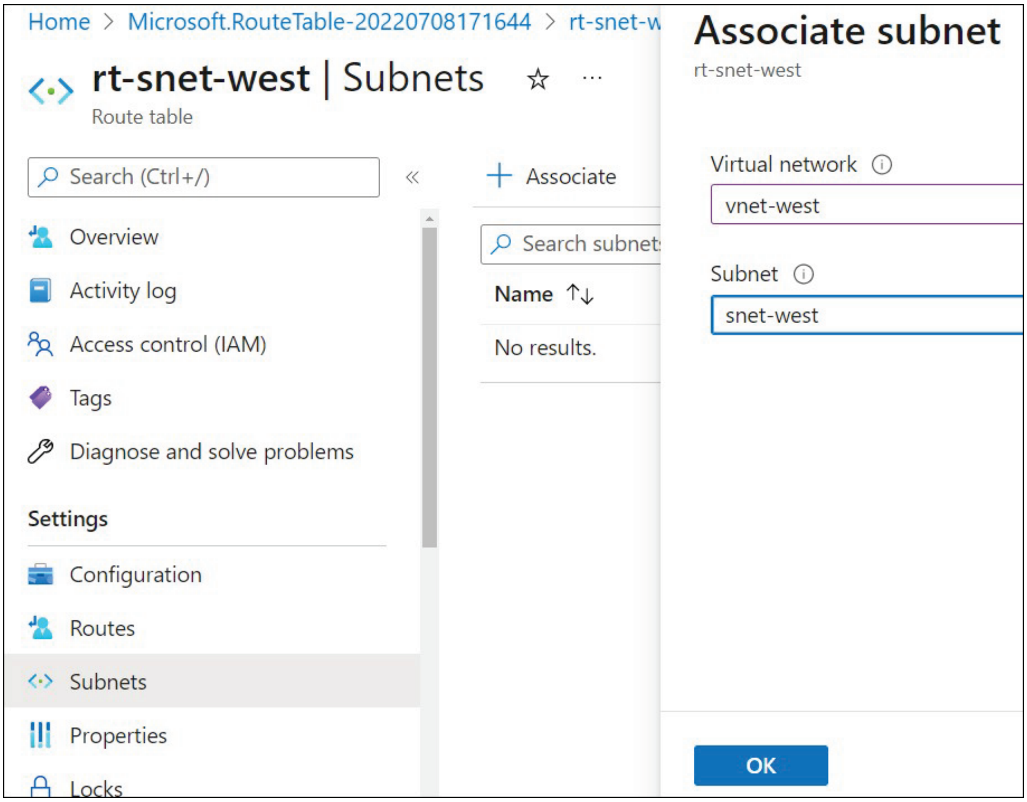
Instance details

Region *	Sweden Central
Name *	rt-snet-west ✓
Propagate gateway routes *	[✓ Yes] No

Figure 9-3: *Create Route Table: Step-3.*

Associate Route Table with Subnet

After the Route Table resource is deployed, you can associate it with the subnet by navigating to the *Route Table* resource page and selecting *Subnets* under the *Settings* section. Click the + *Associate* selector, and choose your VNet and Subnet from the dero-down menu. Then click the *OK* button.


Home > Microsoft.RouteTable-20220708171644 > rt-snet-w

rt-snet-west | Subnets

Route table

Associate subnet

rt-snet-west

Virtual network: vnet-west

Subnet: snet-west

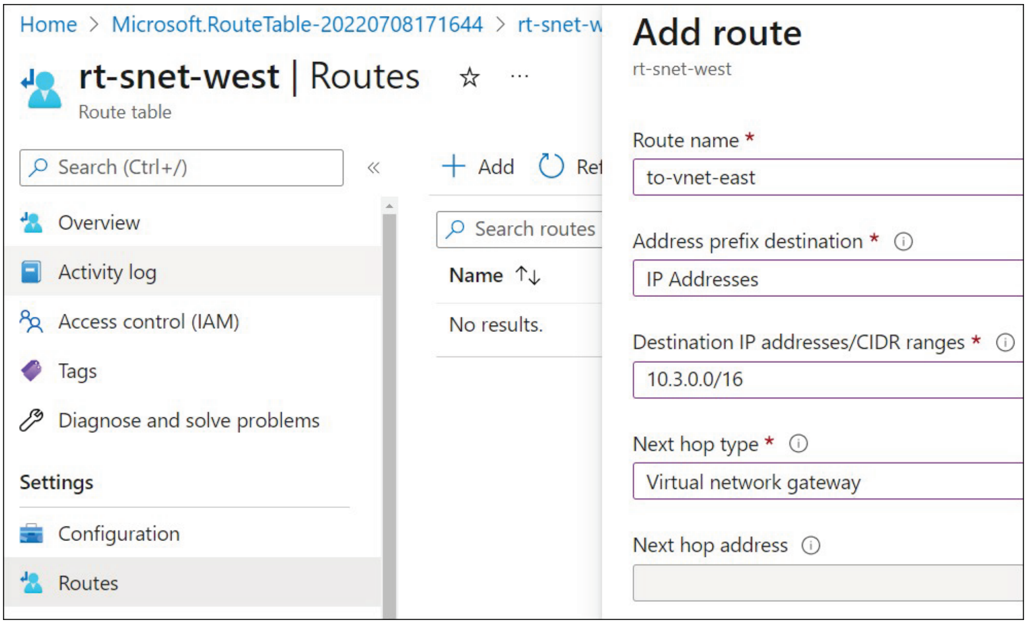
Name ↑↓

No results.

Figure 9-4: *Create Route Table: Step-4 – Route Table-to-Subnet Association.*

Create User Defined Route (UDR)

As the next step, we create a *User Defined Route (UDR)*. Select the Routes option under the Settings section and click the + Add selector. Name the routing entry, and use the *IP Addresses* option as the *Address prefix destination*. Type the destination network to the Destination IP address/CIDR range field and select the next hop type from the drop-down menu. Figure 9-5 shows the UDR configuration steps. Figure 9-6 shows both new Route Tables (rt-snet-west & rt-snet-east) with their respective UDRs.


Home > Microsoft.RouteTable-20220708171644 > rt-snet-w

rt-snet-west | Routes
Route table

Search (Ctrl+/)

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Configuration
Routes

+ Add Ref
Search routes
Name ↑↓
No results.

Add route
rt-snet-west

Route name *
to-vnet-east

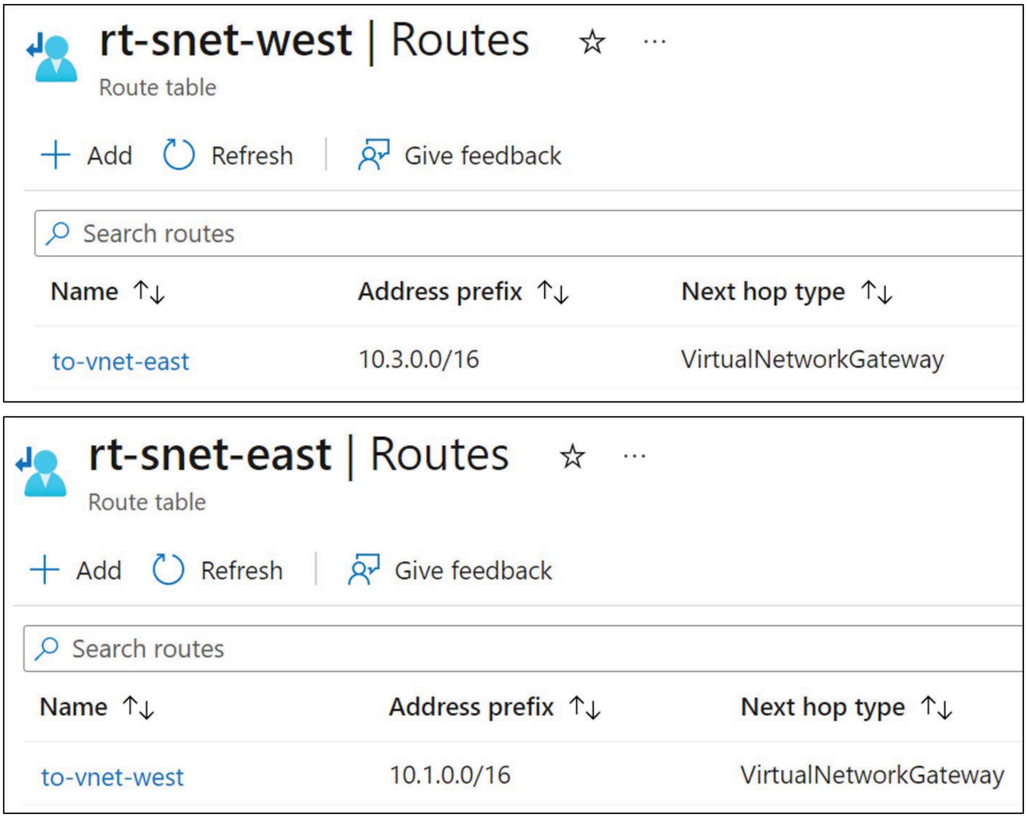
Address prefix destination *
IP Addresses

Destination IP addresses/CIDR ranges *
10.3.0.0/16

Next hop type *
Virtual network gateway

Next hop address

Figure 9-5: *Create Route Table: Step-5 – Configure User Defined Route.*


rt-snet-west | Routes
Route table

Name ↑↓	Address prefix ↑↓	Next hop type ↑↓
to-vnet-east	10.3.0.0/16	VirtualNetworkGateway

rt-snet-east | Routes
Route table

Name ↑↓	Address prefix ↑↓	Next hop type ↑↓
to-vnet-west	10.1.0.0/16	VirtualNetworkGateway

Figure 9-6: *Create Route Table: Verification.*

Control Plane Verification

Example 9-1 verifies that NIC vm-hub699 attached to VM vm-hub has learned the CIDR ranges of both vnet-west (10.1.0.0/16) and vnet-east (10.3.0.0/16) with the next hop VNetPeering.

```
Get-AzEffectiveRouteTable -NetworkInterfaceName vm-hub699 -ResourceGroupName
rg-nsg-rt-swedencentral

Name                         :
DisableBgpRoutePropagation : False
State                        : Active
Source                       : Default
AddressPrefix                : {10.2.0.0/16}
NextHopType                  : VnetLocal
NextHopIpAddress             : {}

Name                         :
DisableBgpRoutePropagation : False
State                        : Active
Source                       : Default
AddressPrefix                : {10.1.0.0/16}
NextHopType                  : VNetPeering
NextHopIpAddress             : {}

Name                         :
DisableBgpRoutePropagation : False
State                        : Active
Source                       : Default
AddressPrefix                : {10.3.0.0/16}
NextHopType                  : VNetPeering
NextHopIpAddress             : {}
```

Example 9-1: *Routing table of the NIC vm-hub699 (vm-hub).*

Example 9-2 shows that NIC vm-west771 attached to VM vm-west has a UDR entry about CIDR ranges of vnet-east (10.3.0.0/16) with the next-hop VirtualNetworkGateway. Example 9-3, in turn, shows that NIC vm-east872 attached to VM vm-east has a UDR entry about CIDR ranges of vnet-west (10.1.0.0/16) with the next-hop VirtualNetworkGateway.

```
Get-AzEffectiveRouteTable -NetworkInterfaceName vm-west771 -ResourceGroupName
rg-nsg-rt-swedencentral
Name                        :
DisableBgpRoutePropagation : False
State                       : Active
Source                      : Default
AddressPrefix               : {10.1.0.0/16}
NextHopType                 : VnetLocal
NextHopIpAddress            : {}

Name                        :
DisableBgpRoutePropagation : False
State                       : Active
Source                      : Default
AddressPrefix               : {10.2.0.0/16}
NextHopType                 : VNetPeering
NextHopIpAddress            : {}

Name                        : to-vnet-east
DisableBgpRoutePropagation : False
State                       : Active
Source                      : User
AddressPrefix               : {10.3.0.0/16}
NextHopType                 : VirtualNetworkGateway
NextHopIpAddress            : {}
```

Example 9-2: *Routing table of the NIC vm-west771 (vm-west).*

```
Get-AzEffectiveRouteTable -NetworkInterfaceName vm-east872 -ResourceGroupName
rg-nsg-rt-swedencentral
Name                        :
DisableBgpRoutePropagation : False
State                       : Active
Source                      : Default
AddressPrefix               : {10.3.0.0/16}
NextHopType                 : VnetLocal
NextHopIpAddress            : {}

Name                        :
DisableBgpRoutePropagation : False
State                       : Active
Source                      : Default
AddressPrefix               : {10.2.0.0/16}
NextHopType                 : VNetPeering
NextHopIpAddress            : {}

Name                        : to-vnet-west
DisableBgpRoutePropagation : False
State                       : Active
Source                      : User
AddressPrefix               : {10.1.0.0/16}
NextHopType                 : VirtualNetworkGateway
NextHopIpAddress            : {}
```

Example 9-3: *Routing table of the NIC vm-east872 (vm-east).*

Data Plane Verification

The last two examples verify that we have IP connectivity between all three VNets.

```
azureuser@vm-west:~$ ping 10.2.0.4
PING 10.2.0.4 (10.2.0.4) 56(84) bytes of data.
64 bytes from 10.2.0.4: icmp_seq=1 ttl=64 time=1.92 ms
64 bytes from 10.2.0.4: icmp_seq=2 ttl=64 time=0.944 ms
^C
--- 10.2.0.4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.944/1.431/1.919/0.487 ms
```

Example 9-4: *Ping from vm-west to 10.2.0.4 (vm-hub) on vnet-hub.*

```
azureuser@vm-west:~$ ping 10.3.0.4
PING 10.3.0.4 (10.3.0.4) 56(84) bytes of data.
64 bytes from 10.3.0.4: icmp_seq=1 ttl=63 time=4.42 ms
64 bytes from 10.3.0.4: icmp_seq=2 ttl=63 time=3.50 ms
64 bytes from 10.3.0.4: icmp_seq=3 ttl=63 time=2.84 ms
^C
--- 10.3.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.844/3.590/4.422/0.647 ms
```

Example 9-5: *Ping from vm-west to 10.3.0.4 (vm-east) on vnet-east.*

Pricing

Figure 9-10 illustrates the cost units related to VNet Peering. VNet Peer Link is free but the ingress/egress fee is 0.010€/GB. This means that sending one GB over one peer link costs 0.02€. If you use Transit VNet, you pay 0.04€ because data flow crosses two Peer Link. Besides, you pay monthly costs for the Virtual Network Gateway or other Virtual Appliance you use for routing. In our example, we use VpnGw2, which monthly cost is 340.2€. Note that prices may change monthly.

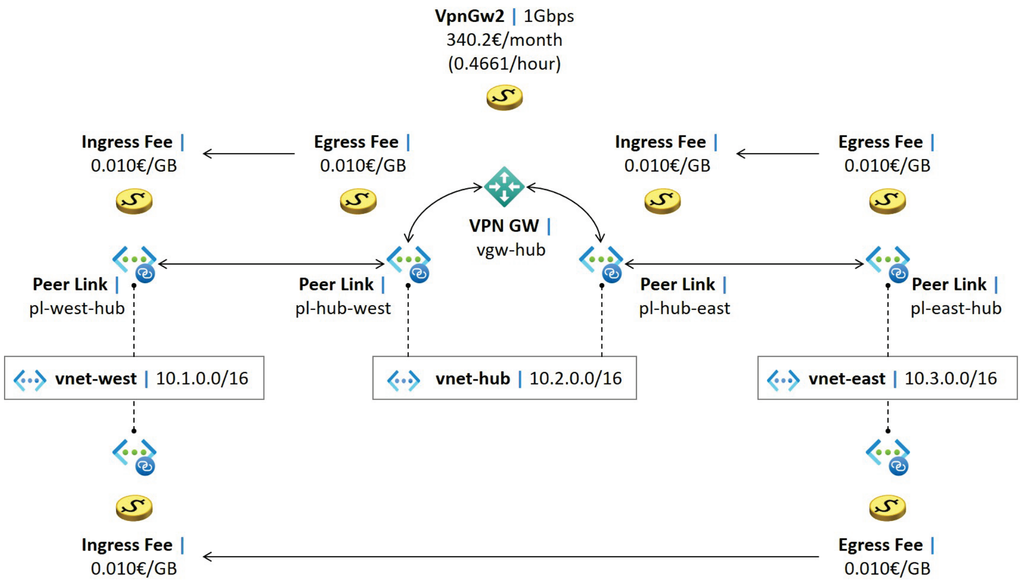


Figure 9-10: *Transit VNet Peering Cost Units.*

References

[1] Create a transit VNet using VNet peering
https://azure.microsoft.com/en-us/blog/create-a-transit-vnet-using-vnet-peering/

[2] VPN Gateway pricing
https://azure.microsoft.com/en-us/pricing/details/vpn-gateway/

Chapter 10: Hybrid Cloud - Routing Studies

Introduction

This chapter discusses route propagations between on-prem DC and Azure Virtual Network in a Hybrid Cloud solution. Figure 10-1 shows our example diagram where we have an on-prem DC connected to Azure over two VPN connections. Local Network Gateway (LGW) *lgw-dc1* has eBGP peering with Virtual Network Gateway *vgw-hub1* (in vnet-hub1). The eBGP peering is established between IP addresses 192.168.10.1 (vgw-dc1) and 10.0.1.254 (vgw-hub1). The LGW *lgw-dc2* has BGP peering with VGW *vgw-hub2* (in vnet-hub2). The eBGP peering is established between IP addresses 192.168.10.2 and 10.1.1.254. VGWs, in turn, are eBGP peers over a VNet-to-Vnet VPN connection. The third VNet, vnet-spoke, has VNet Peering with both hub VNets. It uses vgw-hub1 as a gateway to on-prem DC. Internal BGP sessions in on-prem DC have been established between loopback 1 interfaces, which are advertised using OSPF. Leaf-01 advertises the subnet 10.11.11.0/24 to its iBGP peers lgw-dc1 and lgw-dc2. LGWs process the information received via BGP Update message and generates a new BGP update message for their eBGP peers but not for iBGP peers (BGP loop-prevention mechanism). We have two BGP policies on LGWs. Lgw-dc1 changes the BGP Next-Hop Path Attribute (PA) from the original 10.0.1.254 to its loopback 172.16.1.1 when sending a BGP Update message to its iBGP peers. The same logic applies to lgw-dc2. We have done this because LGWs don’t advertise VGW’s eBGP peering addresses to Leaf-01 and the BGP route validation process verifies that the route to the next hop is installed into the routing table. If the next-hop validation fails, the route is invalid, and Leaf-01 doesn’t install the route to RIB. The second BGP policy defines the outbound policy to eBGP peers, which allows only BGP Updates with an empty AS_PATH list. This policy prevents transit traffic between vnet-hub1 and vnet-hub2 over S2S VPN connections to/from on-prem DC. If you allow on-prem DC to be used as a transit network, remember that the data path goes over two VPN connections, which is not free of charge.

Note! The example solution in this chapter doesn’t describe the best practice model for Hybrid Connection. Its purpose is to introduce the routing interaction between Azure resources and traditional on-prem solutions. I am using a 32-bits BGP ASN 65777 (unreserved at the time of writing) on the customer side in this chapter.

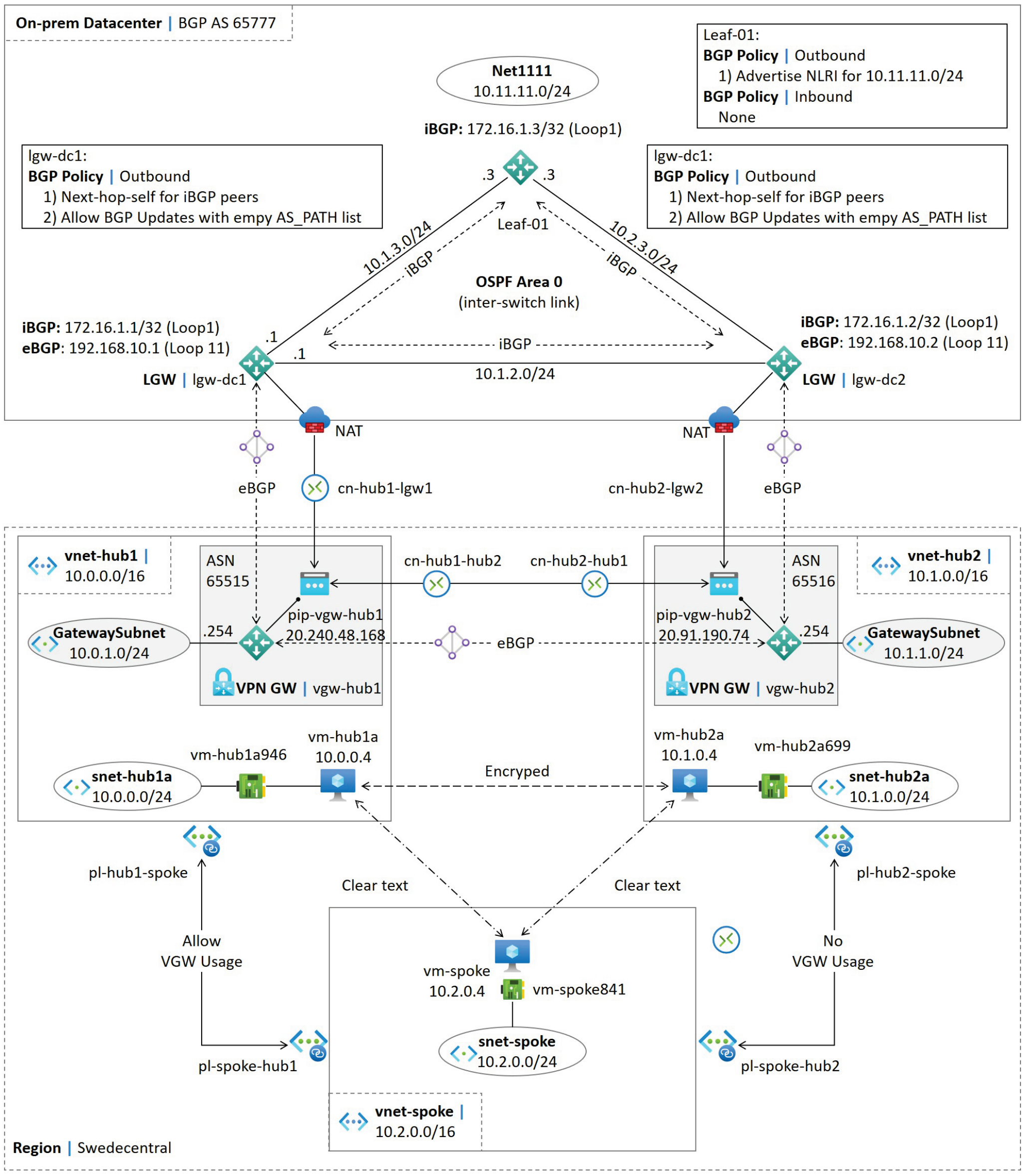


Figure 10-1: *Hybrid Cloud Routing Study Diagram.*

BGP Operation

Figure 10-2 illustrates the BGP route processing flow chart. When a local BGP speaker receives a BGP Update message from one of its BGP peers, the information is stored in the neighbor-specific Adj-RIB-In Pre table without modification or filtering. Then the information is processed based on the Inbound route filter/policy, which may: 1) modify or add values of received path attributes like change the default Local Preference. Routes passing the inbound policy/filter are installed into the Adj-RIB-In Post table. These routes are installed to Loc-RIB if they pass the route validation process, which verifies that the local ASN is not in the AS-Path list and that the advertised next-hop is reachable (the route to next-hop is in RIB). If BGP peering is established over tunnel interfaces, the next hop route points to the tunnel interface, and that interface has to be reachable (recursive next-hop resolution). In our example, this applies to all of our eBGP peerings. BGP speakers may have several BGP peers with the different peering types (iBGP, eBGP, RR-Client, Confederation) and Address Family Information/Subsequent Address Family Information (AFI/SAFI). For example, lgw-dc1 and vgw-hub are eBGP peers exchanges IPv4 Unicast (AFI1/SAFI1) Network Layer Reachability Information (NLRI). All routes from all peers that pass the validation processes are installed to the Loc-RIB and participate in the BGP best path selection process. The best path selection process first compares NLRI's Path Attributes 1) Highest Weight, 2) Highest Local-Preference, 3) Prefer locally originated prefixes, 4) Shortest AS_Path attribute list, 5) prefer IGP < EGP < Incomplete, 6) lowest MED, 7). If Path Attributes are equal, the decision process prefers 8) eBGP over iBGP, (9) the smallest IGP metric to Next-Hop, and 10) prefer the lowest BGP router ID. After selecting the best path, the BGP speaker encodes it to Routing Information Base (RIB) if no other route source with better (lower) Administrative Distance (AD) exists. Then, only the best route is moved to the neighbor-specific Adj-RIB-Out Pre table. Not all routes are valid for all peers. For example, routes learned from iBGP peers are not advertised to other iBGP peers unless they are RR-Clients or if the peering type uses a different AFI/SAFI. Routes are then moved through the policy/filter to the Adj-RIB-Out Post table. The policy/filter may change/add/delete Path Attributes. For example, it can prepend the AS-Path list or add communities. It can also aggregate routes or filter them out. The last policy is executed in the Adj-RIB-Out Post table, which sets the next-hop path attribute when sending a BGP Update message to a peer.

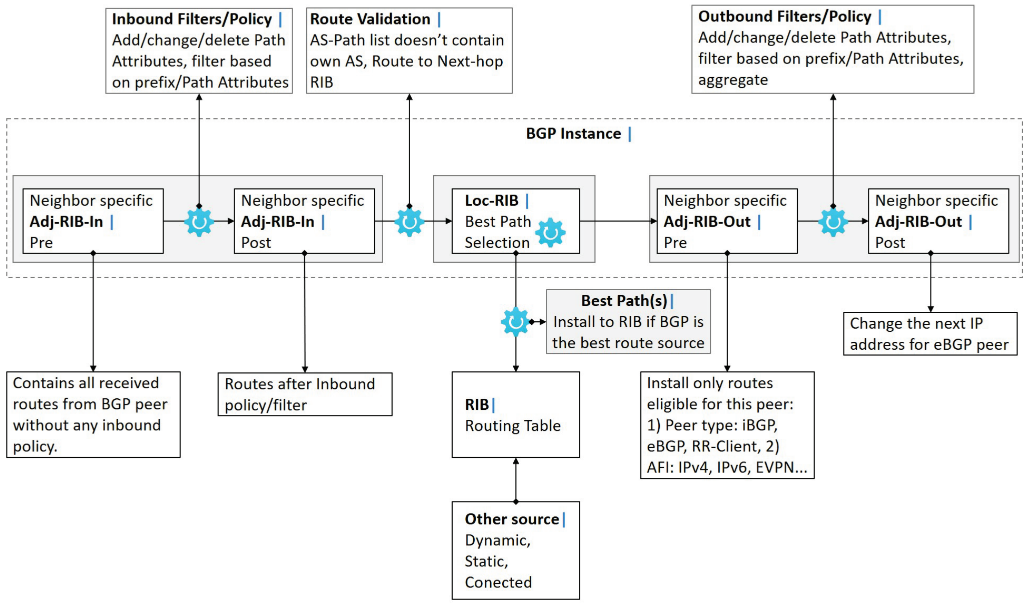
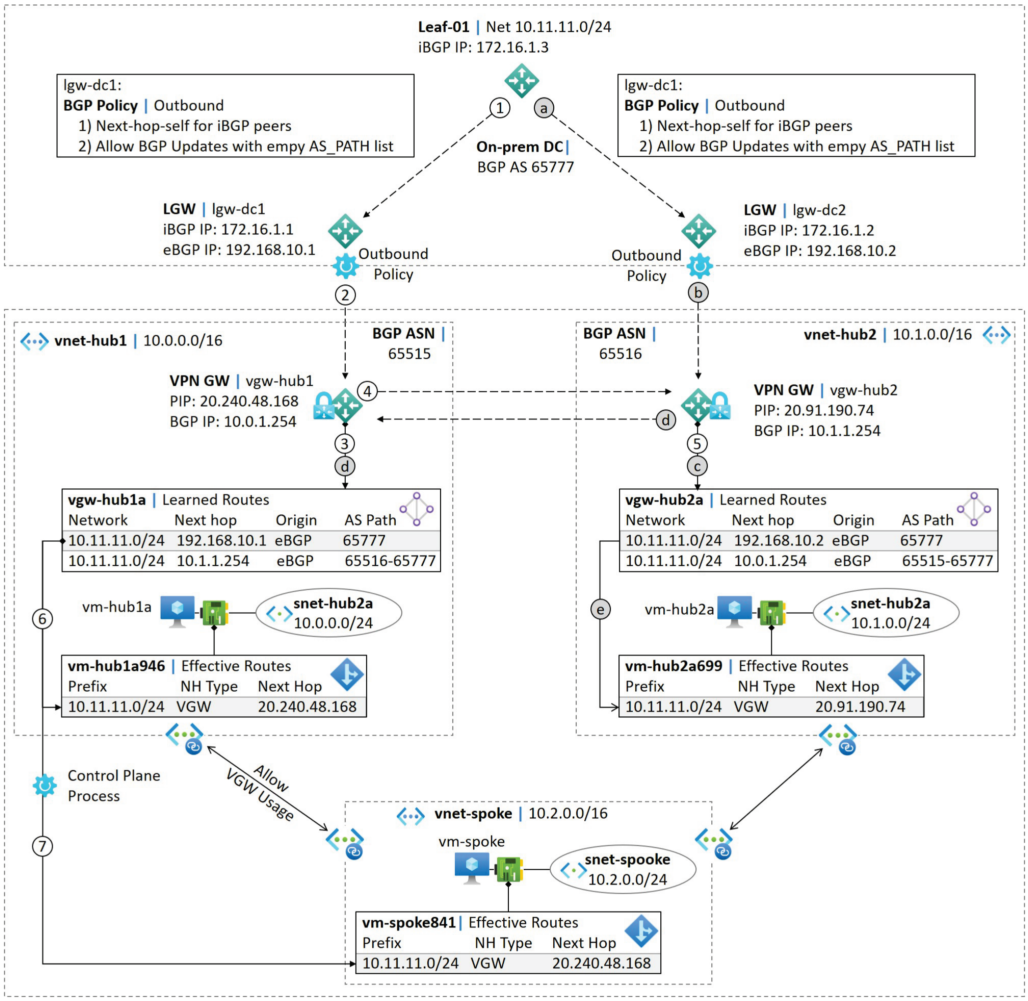

Inbound Filters/Policy |
Add/change/delete Path Attributes, filter based on prefix/Path Attributes
Route Validation |
AS-Path list doesn't contain own AS, Route to Next-hop RIB
Outbound Filters/Policy |
Add/change/delete Path Attributes, filter based on prefix/Path Attributes, aggregate
BGP Instance |
Neighbor specific Adj-RIB-In | Pre
Neighbor specific Adj-RIB-In | Post
Loc-RIB | Best Path Selection
Neighbor specific Adj-RIB-Out | Pre
Neighbor specific Adj-RIB-Out | Post
Best Path(s) | Install to RIB if BGP is the best route source
Change the next IP address for eBGP peer
Contains all received routes from BGP peer without any inbound policy.
Routes after Inbound policy/filter
RIB | Routing Table
Install only routes eligible for this peer: 1) Peer type: iBGP, eBGP, RR-Client, 2) AFI: IPv4, IPv6, EVPN...
Other source | Dynamic, Static, Conected

Figure 10-2: *BGP Processes.*

Routing Process: on-Prem DC Subnet 10.11.11.0/24

Figure 10-3 illustrates the subnet 10.11.11.0/24 route propagation.



lgw-dc1:
BGP Policy | Outbound
1) Next-hop-self for iBGP peers
2) Allow BGP Updates with empy AS_PATH list

lgw-dc1:
BGP Policy | Outbound
1) Next-hop-self for iBGP peers
2) Allow BGP Updates with empy AS_PATH list

vgw-hub1a | Learned Routes

Network	Next hop	Origin	AS Path
10.11.11.0/24	192.168.10.1	eBGP	65777
10.11.11.0/24	10.1.1.254	eBGP	65516-65777

vgw-hub2a | Learned Routes

Network	Next hop	Origin	AS Path
10.11.11.0/24	192.168.10.2	eBGP	65777
10.11.11.0/24	10.0.1.254	eBGP	65515-65777

vm-hub1a946 | Effective Routes

Prefix	NH Type	Next Hop
10.11.11.0/24	VGW	20.240.48.168

vm-hub2a699 | Effective Routes

Prefix	NH Type	Next Hop
10.11.11.0/24	VGW	20.91.190.74

vm-spoke841 | Effective Routes

Prefix	NH Type	Next Hop
10.11.11.0/24	VGW	20.240.48.168

Figure 10-3: *Subnet 10.11.11.0/24 propagation.*

Leaf-01 exports its connected subnet 10.11.11.0/24 from RIB to BGP Adj-RIB-Out tables associated with iBGP peer lgw-dc1 and lgw-dc1 with an empty AS-Path list. It builds a BGP Update message and sends it to both LGWs (1) lgw-dc1 and (a) lgw-dc2 using the same next-hop address 172.16.1.3. Lgw-dc1 installs the received message to Adj-RIB-In, validates it, and copies information to BGP Loc-RIB. Lgw-dc2 runs the same process. LGWs don't exchange BGP Updates received from iBGP peer Leaf-01 between themselves due to BGP built-In lop prevention mechanism (information from one iBGP peer is not propagated to another iBGP, excluding RR-Clients). Both LGWs select the NLRI from Leaf-01 as the best route and install it to RIB. Examples 10-1 and 10-2 show the BGP Loc-RIB and RIB of lgw-dc1 and lgw-dc2.

```
lgw-dc1#sh ip bgp | beg Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>   10.0.0.0/16      10.0.1.254                             0 65515 i
 *>i  10.1.0.0/16      172.16.1.2               0    100      0 65516 i
 *                     10.0.1.254                             0 65515 65516 i
 *>   10.2.0.0/16      10.0.1.254                             0 65515 i
 *>i  10.11.11.0/24    172.16.1.3               0    100      0 i

lgw-dc1#sh ip route bgp | beg 10.0
      10.0.0.0/8 is variably subnetted, 10 subnets, 3 masks
B        10.0.0.0/16 [20/0] via 10.0.1.254, 02:11:09
B        10.1.0.0/16 [200/0] via 172.16.1.2, 00:15:23
B        10.2.0.0/16 [20/0] via 10.0.1.254, 02:11:09
B        10.11.11.0/24 [200/0] via 172.16.1.3, 01:17:06
```

Example 10-1: *lgw-dc1: BGP Loc-RIB and RIB.*

```
lgw-dc2#sh ip bgp | beg Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>i  10.0.0.0/16      172.16.1.1               0    100      0 65515 i
 *                     10.1.1.254                             0 65516 65515 i
 *>   10.1.0.0/16      10.1.1.254                             0 65516 i
 *>i  10.2.0.0/16      172.16.1.1               0    100      0 65515 i
 *                     10.1.1.254                             0 65516 65515 i
 *>i  10.11.11.0/24    172.16.1.3               0    100      0 i

lgw-dc2#sh ip route bgp | beg 10.0
      10.0.0.0/8 is variably subnetted, 10 subnets, 3 masks
B        10.0.0.0/16 [200/0] via 172.16.1.1, 00:16:20
B        10.1.0.0/16 [20/0] via 10.1.1.254, 02:16:00
B        10.2.0.0/16 [200/0] via 172.16.1.1, 00:16:20
B        10.11.11.0/24 [200/0] via 172.16.1.3, 01:17:55
```

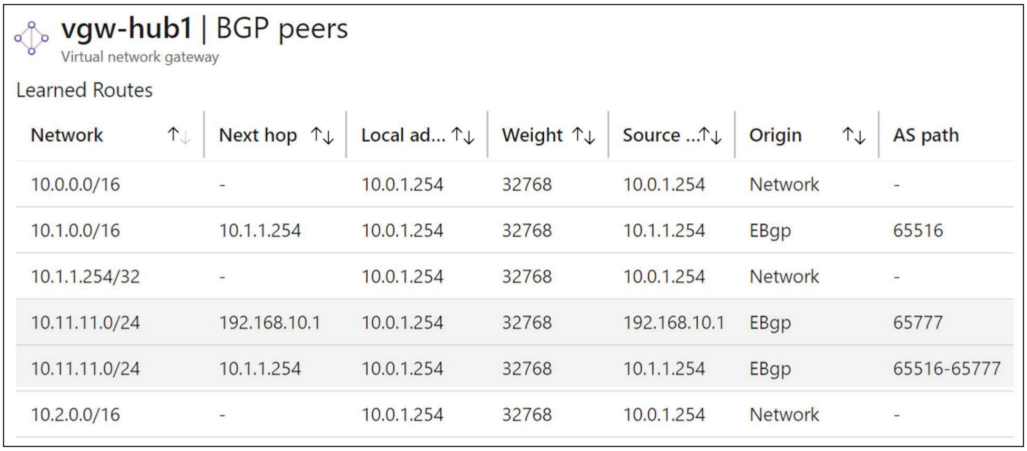
Example 10-2: *Lgw-dc2: BGP Loc-RIB and RIB.*

After installing the best route to RIB, LGWs code it to the neighbor-specific BGP Adj-RIB-Out table. We have configured an outbound BGP policy towards Azure VGWs on both lgw-dc1 and lgw-dc2. The policy permits only prefixes with an empty AS-Path list to be advertised to eBGP peers. Lgw-dc1, as an example, runs the outbound policy before it adds its local ASN 65777 to AS-Path list. Then it sets its eBGP peering IP address to the next-hop field and sends the message to vgw-hub1 (2). Lgw-dc2 repeats the same steps (b). Example 10-3 shows our eBGP peer outbound policy.

```
router bgp 65777
 bgp log-neighbor-changes
 neighbor 10.1.1.254 remote-as 65516
 neighbor 10.1.1.254 ebgp-multihop 255
 neighbor 10.1.1.254 update-source Loopback11
 neighbor 172.16.1.1 remote-as 65777
 neighbor 172.16.1.1 update-source Loopback1
 neighbor 172.16.1.3 remote-as 65777
 neighbor 172.16.1.3 update-source Loopback1
 !
 address-family ipv4
  network 192.168.99.2 mask 255.255.255.255
  neighbor 10.1.1.254 activate
  neighbor 10.1.1.254 route-map BGP-OUT-POLICY out
  neighbor 172.16.1.1 activate
  neighbor 172.16.1.1 next-hop-self
  neighbor 172.16.1.3 activate
  neighbor 172.16.1.3 next-hop-self
 exit-address-family
!
ip route 10.1.1.254 255.255.255.255 Tunnel11
!
ip as-path access-list 1 permit ^$
!
!
!
route-map BGP-OUT-POLICY permit 10
 match as-path 1
```

Example 10-3: *Lgw-dc2: BGP Configuration – Outbound Policy.*

Vgw-hub1 receives the BGP Update message from lgw-dc1 about prefix 10.11.11.0/24 with AS-Path list 65777 and next-hop 192.16.10.1. It validates the information and installs it to the BGP table as an eBGP entry (3). Vgw-hub2 processes the received BGP Update from its eBGP lgw-dc2 in the same way (c). Besides, VGWs exchange routing information about 10.11.11.0/24 because they are eBGP peers (4, b). Vgw-hub1 creates a BGP Update message, prepends the AS-Path list with its local ASN 65515, and sets the next-hop address to its eBGP peering address 10.0.1.254. Then it sends the message to vgw-hub2 (4). Vgw-hub2 validates the received message and installs it to the BGP table as an eBGP entry with AS-Path list 65515-65777. The receiving BGP Update message about prefix 10.11.11.0/24 triggers the BGP best path selection process because vgw-hub2 has an existing eBGP entry via lgw-dc2 in the BGP table. Route from lgw-dc2 wins the best path selection due to the shorter AS-Path list. Azure Control Plane then encodes the best route to the routing table associated with NIC vm-hub2a699 (attached to vm-hub2a in subnet 10.1.0.0/24) with the next-hop 20.91.190.74 (vgw-hub2). The route exchange applies on the opposite direction from vgw-hub2 to vgw-hub1 (d). VNet peering policy between vnet-hub1 and vnet-spoke allows vnet-spoke to use vgw-hub1 as a gateway. Azure Control Plane propagates the routing information to NIC vm-spoke841 (attached to vm-spoke bin subnet 10.2.0.0/24) with the next-hop 20.240.48.168 (vgw-hub1) (7). Now all resources know how to route packets to hosts attached to subnet 10.11.11.0/24 in on-prem DC. Figures 10-4 and 10-5 show VGWs' BGP tables.

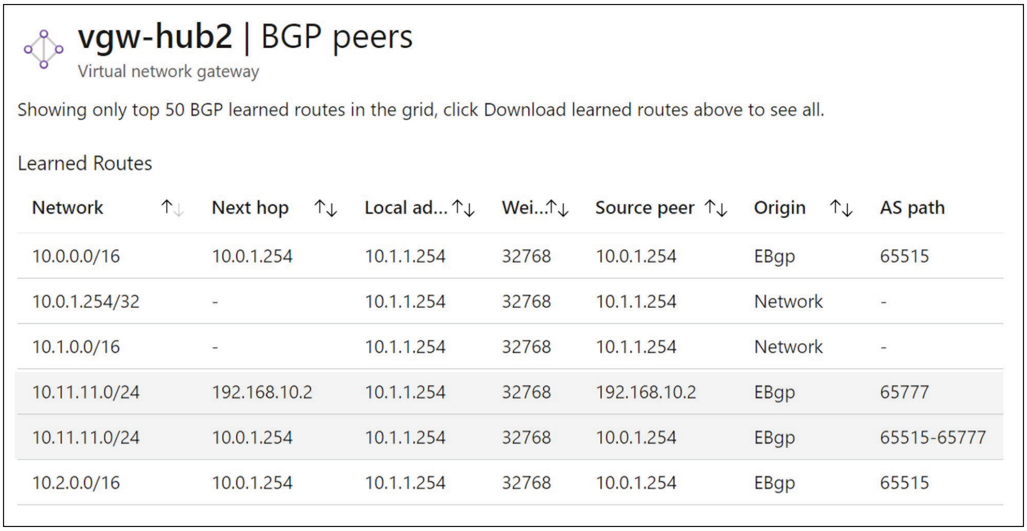


vgw-hub1 | BGP peers
Virtual network gateway

Learned Routes

Network	Next hop	Local ad...	Weight	Source ...	Origin	AS path
10.0.0.0/16	-	10.0.1.254	32768	10.0.1.254	Network	-
10.1.0.0/16	10.1.1.254	10.0.1.254	32768	10.1.1.254	EBgp	65516
10.1.1.254/32	-	10.0.1.254	32768	10.0.1.254	Network	-
10.11.11.0/24	192.168.10.1	10.0.1.254	32768	192.168.10.1	EBgp	65777
10.11.11.0/24	10.1.1.254	10.0.1.254	32768	10.1.1.254	EBgp	65516-65777
10.2.0.0/16	-	10.0.1.254	32768	10.0.1.254	Network	-

Figure 10-4: *Vgw-hub1 BGP Table.*



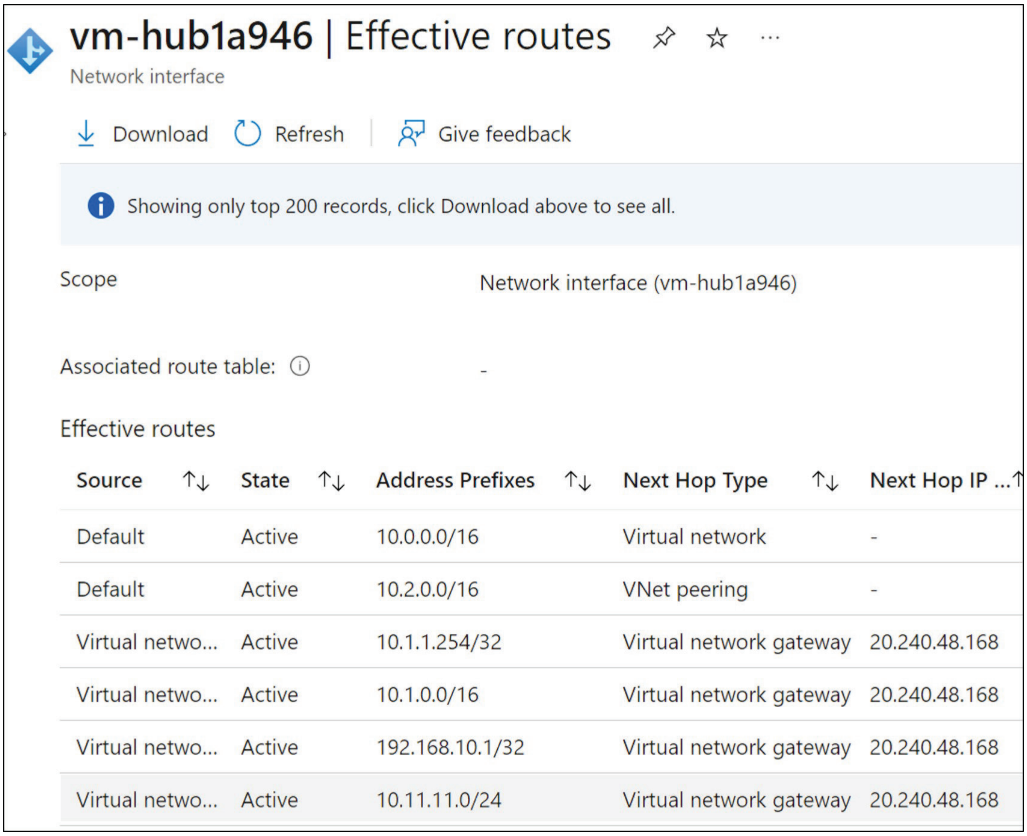
Showing only top 50 BGP learned routes in the grid, click Download learned routes above to see all.

Learned Routes

Network	Next hop	Local ad...	Wei...	Source peer	Origin	AS path
10.0.0.0/16	10.0.1.254	10.1.1.254	32768	10.0.1.254	EBgp	65515
10.0.1.254/32	-	10.1.1.254	32768	10.1.1.254	Network	-
10.1.0.0/16	-	10.1.1.254	32768	10.1.1.254	Network	-
10.11.11.0/24	192.168.10.2	10.1.1.254	32768	192.168.10.2	EBgp	65777
10.11.11.0/24	10.0.1.254	10.1.1.254	32768	10.0.1.254	EBgp	65515-65777
10.2.0.0/16	10.0.1.254	10.1.1.254	32768	10.0.1.254	EBgp	65515

Figure 10-5: *Vgw-hub2 BGP Table.*

Figures 10-6, 10-7, and 10-8 show route tables associated with NICs.



Showing only top 200 records, click Download above to see all.

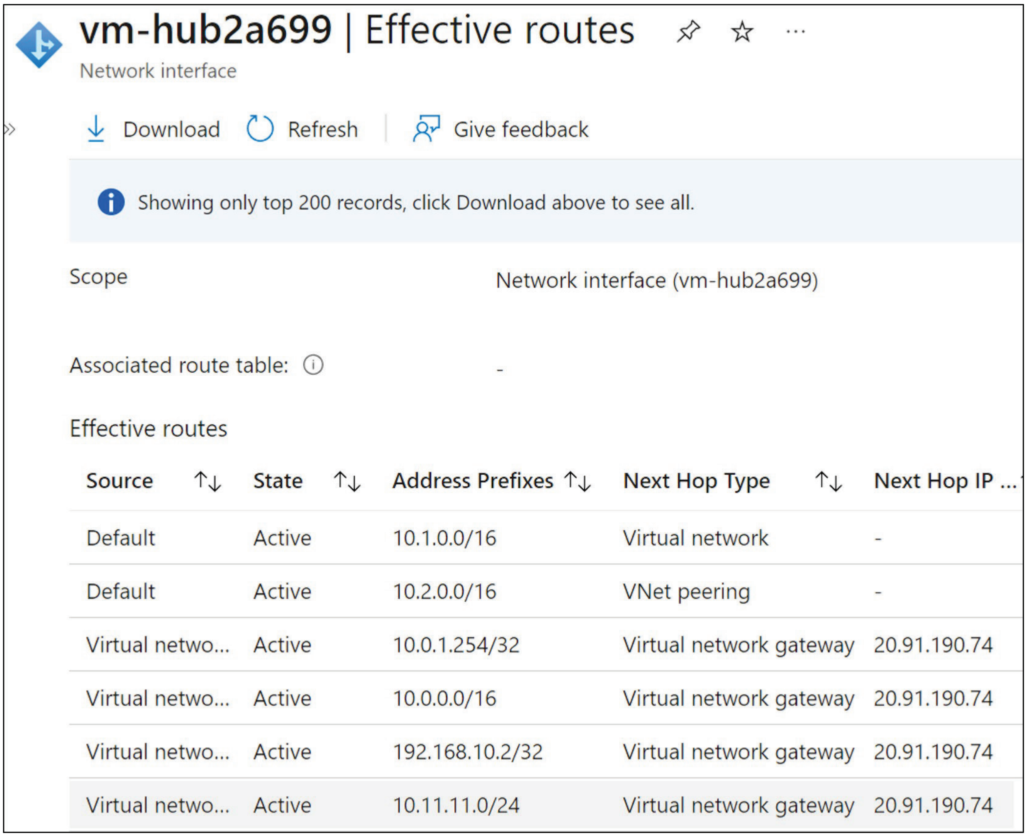
Scope: Network interface (vm-hub1a946)

Associated route table: -

Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Hop IP ...
Default	Active	10.0.0.0/16	Virtual network	-
Default	Active	10.2.0.0/16	VNet peering	-
Virtual netwo...	Active	10.1.1.254/32	Virtual network gateway	20.240.48.168
Virtual netwo...	Active	10.1.0.0/16	Virtual network gateway	20.240.48.168
Virtual netwo...	Active	192.168.10.1/32	Virtual network gateway	20.240.48.168
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	20.240.48.168

Figure 10-6: *Effective Route on vm-hub1a946.*



vm-hub2a699 | Effective routes

Network interface

Download Refresh Give feedback

Showing only top 200 records, click Download above to see all.

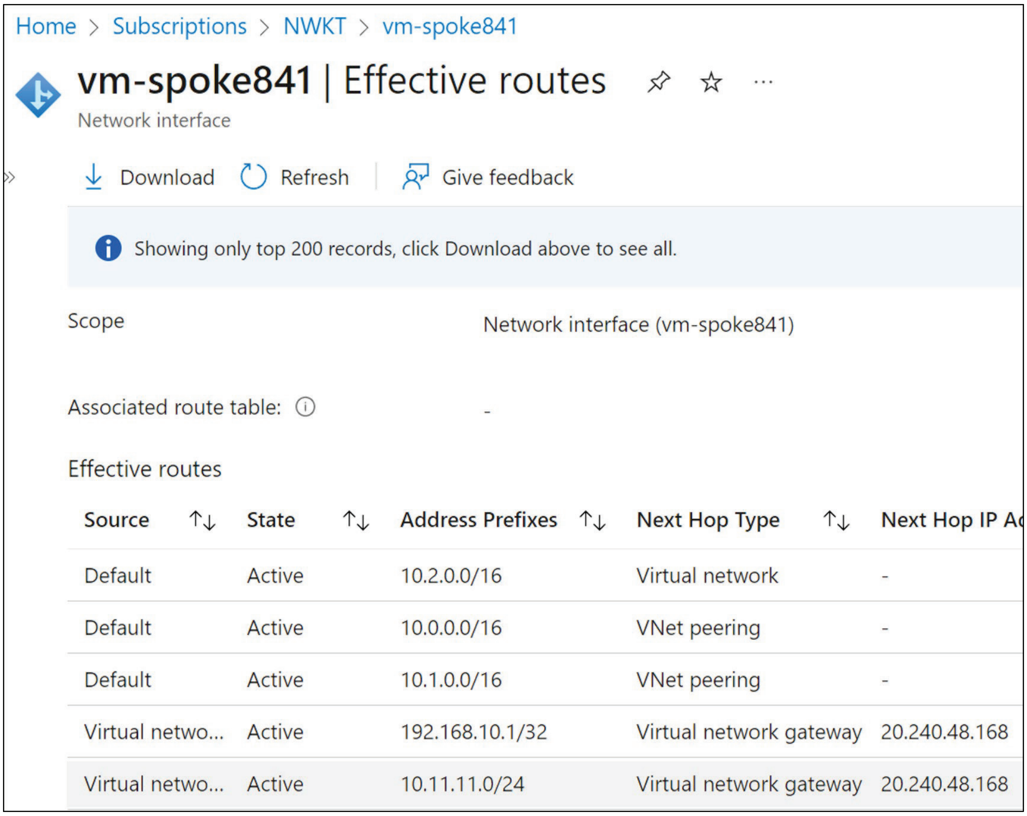
Scope: Network interface (vm-hub2a699)

Associated route table: -

Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Hop IP ...
Default	Active	10.1.0.0/16	Virtual network	-
Default	Active	10.2.0.0/16	VNet peering	-
Virtual netwo...	Active	10.0.1.254/32	Virtual network gateway	20.91.190.74
Virtual netwo...	Active	10.0.0.0/16	Virtual network gateway	20.91.190.74
Virtual netwo...	Active	192.168.10.2/32	Virtual network gateway	20.91.190.74
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	20.91.190.74

Figure 10-7: *Effective Route on vm-hub2a699.*


Home > Subscriptions > NWKT > vm-spoke841
vm-spoke841 | Effective routes
Network interface
Download
Refresh
Give feedback

Showing only top 200 records, click Download above to see all.

Scope: Network interface (vm-spoke841)

Associated route table: -

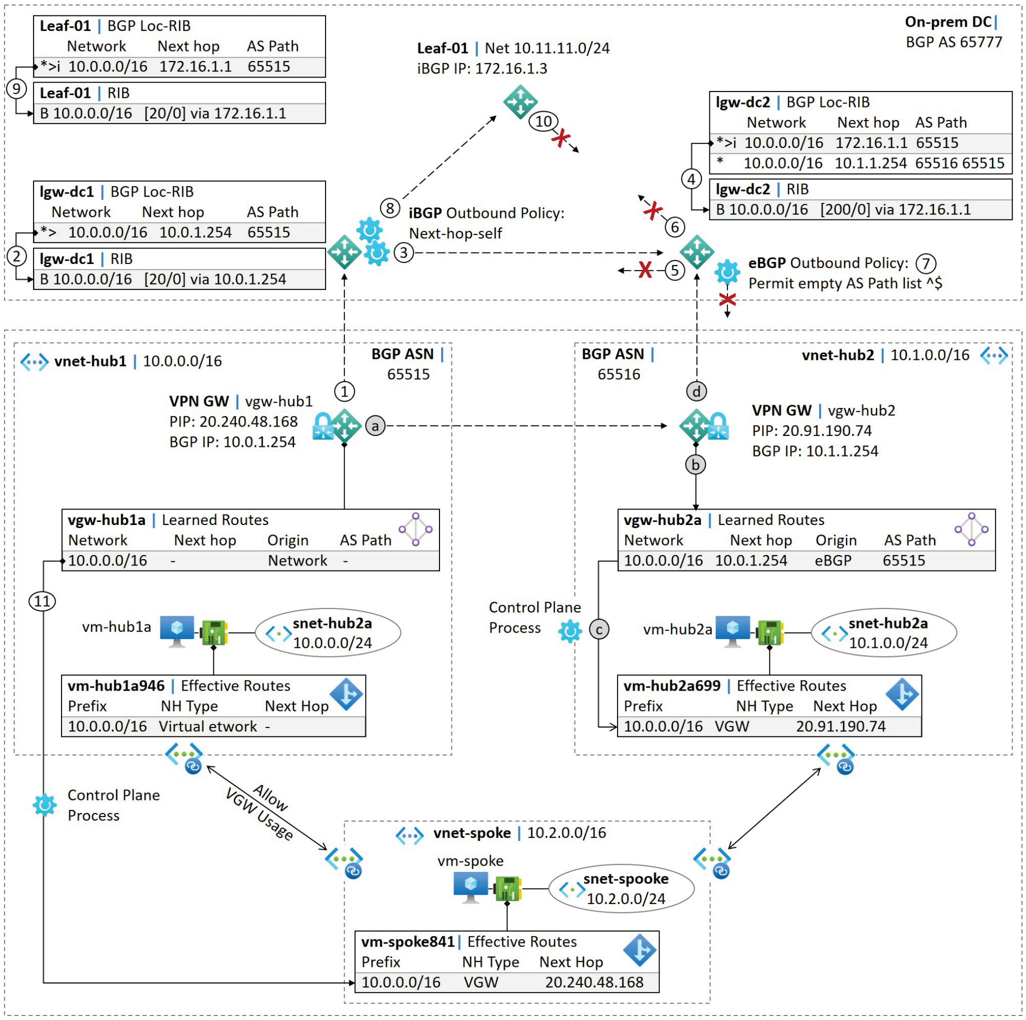
Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Hop IP Ad
Default	Active	10.2.0.0/16	Virtual network	-
Default	Active	10.0.0.0/16	VNet peering	-
Default	Active	10.1.0.0/16	VNet peering	-
Virtual netwo...	Active	192.168.10.1/32	Virtual network gateway	20.240.48.168
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	20.240.48.168

Figure 10-8: *Effective Route on vm-spoke841.*

Routing Process: VNet CIDR 10.0.0.0/16

Figure 10-9 illustrates the subnet 10.0.0.0/16 route propagation.



Leaf-01 | BGP Loc-RIB

	Network	Next hop	AS Path
*>i	10.0.0.0/16	172.16.1.1	65515

Leaf-01 | RIB

B 10.0.0.0/16 [20/0] via 172.16.1.1

lgw-dc1 | BGP Loc-RIB

	Network	Next hop	AS Path
*>	10.0.0.0/16	10.0.1.254	65515

lgw-dc1 | RIB

B 10.0.0.0/16 [20/0] via 10.0.1.254

lgw-dc2 | BGP Loc-RIB

	Network	Next hop	AS Path
*>i	10.0.0.0/16	172.16.1.1	65515
*	10.0.0.0/16	10.1.1.254	65516 65515

lgw-dc2 | RIB

B 10.0.0.0/16 [200/0] via 172.16.1.1

vgw-hub1a | Learned Routes

Network	Next hop	Origin	AS Path
10.0.0.0/16	-	Network	-

vgw-hub2a | Learned Routes

Network	Next hop	Origin	AS Path
10.0.0.0/16	10.0.1.254	eBGP	65515

vm-hub1a946 | Effective Routes

Prefix	NH Type	Next Hop
10.0.0.0/16	Virtual etwork	-

vm-hub2a699 | Effective Routes

Prefix	NH Type	Next Hop
10.0.0.0/16	VGW	20.91.190.74

vm-spoke841 | Effective Routes

Prefix	NH Type	Next Hop
10.0.0.0/16	VGW	20.240.48.168

Figure 10-9: *Subnet 10.0.0.0/24 propagation.*

Figure 10-9 illustrates the vnet-hub1's CIDR 10.0.0.0/16 route propagation. Vgw-hub1 sends BGP Update to its eBGP peers lgw-dc1 (1) and vgw-hub2 (a) using the next-hop IP address 10.0.1.254 and AS-Path list 65515. Vgw-hub2 processes the BGP Update message and eventually installs the NLRI into the BGP table. Azure Control Plane then copies the routing information to all NICs in vnet-hub2 (b). Lgw-dc1 handles the ingress BGP Update in the same way as vgw-hub2. Note that lgw-dc1 installs the route to the RIB with Administrative Distance 20 (the AD for eBGP learned routes).

Because this is the only BGP learned route and there isn't any other route source, lgw-dc1 encodes the routing information to the RIB. Next, both vgw-hub2 and lgw-dc1 send a BGP Update message to lgw-dc2. Vgw-hub2 prepends the AS-Path list with its local ASN 65516 and sets its eBGP peering address 10.1.1.254 to the next-hop field (d). Lgw-dc1 also sends a BGP Update message from its best path to 10.0.0.0/16. We have implemented an outbound policy on lgw-dc1, which changes the next-hop IP address to the local iBGP peering address (3). Without this policy, lgw-dc2 receives the BGP Update from vgw-dc1 with the next-hop 10.0.1.254 (vgw-hub1), and route validation fails. Lgw-dc2 receives BGP Update from vgw-hub2 with AS-Path list 65516-65515 and from lgw-dc1 with AS-Path list 65515. Both received routes are valid. The BGP best path selection process chooses the route from lgw-dc1 because of the shorter AS-Path. This route is installed into RIB (4) with the AD 200 (iBGP AD). Because lgw-dc2 has received the best path from lgw-dc1, it doesn't advertise a route back to it (5). Lgw-dc2 doesn't advertise the route to its iBGP peer Leaf-01 either because it learned the route from another iBGP peer (6). Remember that the BGP speaker only advertises the best route, so route information learned from vgw-hub2 is not advertised to any peer because it is not selected as the best route. In addition, the lgw-dc2 egress policy towards vgw-hub2 permits only routes with an empty AS-Path list, so lgw-dc2 doesn't send BGP Update to vgw-hub2 about prefix 10.0.0.0/6 (7). This policy verifies that on-prem DC will never be used as a transit network for traffic flows between Azure VNets vnet-hub1 and vnet-hub2. Steps 8-10 show the BGP process from the Leaf-01 perspective. Step 11 shows how the Azure control plane copies the CIDR 10.0.0.0/16 to NICs in vnet-spoke because the Vnet peering policy allows vnet-spoke to use vgw-hub1 as a gateway.

The following three examples show the BGP tables and RIBs of on-prem devices. Prefix 10.0.0.0/16 related entries are highlighted.

```
lgw-dc1#sh ip bgp | beg Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>   10.0.0.0/16      10.0.1.254                             0 65515 i
 *>i  10.1.0.0/16      172.16.1.2               0    100      0 65516 i
 *                     10.0.1.254                             0 65515 65516 i
 *>   10.2.0.0/16      10.0.1.254                             0 65515 i
 *>i  10.11.11.0/24    172.16.1.3               0    100      0 i

lgw-dc1#sh ip route bgp | beg 10.0
      10.0.0.0/8 is variably subnetted, 10 subnets, 3 masks
B        10.0.0.0/16 [20/0] via 10.0.1.254, 02:11:09
B        10.1.0.0/16 [200/0] via 172.16.1.2, 00:15:23
B        10.2.0.0/16 [20/0] via 10.0.1.254, 02:11:09
B        10.11.11.0/24 [200/0] via 172.16.1.3, 01:17:06
```

Example 10-4: *lgw-dc1: BGP Loc-RIB and RIB.*

```
lgw-dc2#sh ip bgp | beg Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>i  10.0.0.0/16      172.16.1.1               0    100      0 65515 i
 *                     10.1.1.254                             0 65516 65515 i
 *>   10.1.0.0/16      10.1.1.254                             0 65516 i
 *>i  10.2.0.0/16      172.16.1.1               0    100      0 65515 i
 *                     10.1.1.254                             0 65516 65515 i
 *>i  10.11.11.0/24    172.16.1.3               0    100      0 i

lgw-dc2#sh ip route bgp | beg 10.0
      10.0.0.0/8 is variably subnetted, 10 subnets, 3 masks
B        10.0.0.0/16 [200/0] via 172.16.1.1, 00:16:20
B        10.1.0.0/16 [20/0] via 10.1.1.254, 02:16:00
B        10.2.0.0/16 [200/0] via 172.16.1.1, 00:16:20
B        10.11.11.0/24 [200/0] via 172.16.1.3, 01:17:55
```

Example 10-5: *Lgw-dc2: BGP Loc-RIB and RIB.*

```
Leaf-01#sh ip bgp | beg Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>i  10.0.0.0/16      172.16.1.1               0    100      0 65515 i
 *>i  10.1.0.0/16      172.16.1.2               0    100      0 65516 i
 *>i  10.2.0.0/16      172.16.1.1               0    100      0 65515 i
 *>   10.11.11.0/24    0.0.0.0                  0         32768 i

Leaf-01#sh ip route bgp | beg 10.
      10.0.0.0/8 is variably subnetted, 9 subnets, 3 masks
B        10.0.0.0/16 [200/0] via 172.16.1.1, 00:35:44
B        10.1.0.0/16 [200/0] via 172.16.1.2, 00:35:36
B        10.2.0.0/16 [200/0] via 172.16.1.1, 00:35:44
```

Example 10-6: *Leaf-01: BGP Loc-RIB and RIB.*

Data Plane Test Between on-prem DC and Azure

Example 10-7 verifies that we have IP connections between on-prem DC and Azure Virtual Networks.

```
# ping to vm-hub1a from on-prem DC
Leaf-01#ping 10.0.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/12/13 ms
# ping to vm-hub2a from on-prem DC
Leaf-01#ping 10.1.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/13/19 ms
# ping to vm-spoke from on-prem DC
Leaf-01#ping 10.2.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/14/25 ms
```

Example 10-7: *Leaf-01: Data Plane Testing Using ICMP.*

Azure Internal Data Plane Verification

Example 10-8 verifies that we also have Intra-VNet IP connections in Azure.

```
# ping from vm-spoke to vm-hub1a
azureuser@vm-spoke:~$ ping 10.0.0.4 -c 3
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=1.11 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.915 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.933 ms

--- 10.0.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.915/0.986/1.112/0.088 ms

# ping from vm-spoke to vm-hub2a
azureuser@vm-spoke:~$ ping 10.1.0.4 -c 3
PING 10.1.0.4 (10.1.0.4) 56(84) bytes of data.
64 bytes from 10.1.0.4: icmp_seq=1 ttl=64 time=3.06 ms
64 bytes from 10.1.0.4: icmp_seq=2 ttl=64 time=0.822 ms
64 bytes from 10.1.0.4: icmp_seq=3 ttl=64 time=1.11 ms

--- 10.1.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2008ms
rtt min/avg/max/mdev = 0.822/1.664/3.064/0.996 ms
```

Example 10-8: *vm-spoke: Data Plane Testing Using ICMP.*

References

[1] Y. Rekhter et al., "A Border Gateway Protocol (BGP-4)",
RFC 4271, January 2006.

[2] T. Bates et al., "Multiprotocol Extensions for BGP-4",
RFC 4760, January 2007.

[3] J. Scudder et al., "BGP Monitoring Protocol (BMP)"
RFC 7854, June 2006

[4] T. Evens et al., "Support for Adj-RIB-Out in the BGP Monitoring Protocol (BMP)"
RFC 8671, November 2019

[5] T. Evens et al., "Support for Local RIB in the BGP Monitoring Protocol (BMP)"
RFC 9069, February 2022

Chapter 10 - Appendix A: On-Prem DC – BGP Configuration

Lgw-dc1

```
router bgp 65777
 bgp log-neighbor-changes
 neighbor 10.0.1.254 remote-as 65515
 neighbor 10.0.1.254 ebgp-multihop 255
 neighbor 10.0.1.254 update-source Loopback11
 neighbor 172.16.1.2 remote-as 65777
 neighbor 172.16.1.2 update-source Loopback1
 neighbor 172.16.1.3 remote-as 65777
 neighbor 172.16.1.3 update-source Loopback1
 !
 address-family ipv4
  network 192.168.99.1 mask 255.255.255.255
  neighbor 10.0.1.254 activate
  neighbor 10.0.1.254 route-map BGP-OUT-POLICY out
  neighbor 172.16.1.2 activate
  neighbor 172.16.1.2 next-hop-self
  neighbor 172.16.1.3 activate
  neighbor 172.16.1.3 next-hop-self
 exit-address-family
!
ip route 10.0.1.254 255.255.255.255 Tunnel11
!
ip as-path access-list 1 permit ^$
!
!
!
route-map BGP-OUT-POLICY permit 10
 match as-path 1
```

Lgw-dc2

```
router bgp 65777
 bgp log-neighbor-changes
 neighbor 10.1.1.254 remote-as 65516
 neighbor 10.1.1.254 ebgp-multihop 255
 neighbor 10.1.1.254 update-source Loopback11
 neighbor 172.16.1.1 remote-as 65777
 neighbor 172.16.1.1 update-source Loopback1
 neighbor 172.16.1.3 remote-as 65777
 neighbor 172.16.1.3 update-source Loopback1
 !
 address-family ipv4
  network 192.168.99.2 mask 255.255.255.255
  neighbor 10.1.1.254 activate
  neighbor 10.1.1.254 route-map BGP-OUT-POLICY out
  neighbor 172.16.1.1 activate
  neighbor 172.16.1.1 next-hop-self
  neighbor 172.16.1.3 activate
  neighbor 172.16.1.3 next-hop-self
 exit-address-family
!
ip route 10.1.1.254 255.255.255.255 Tunnel11
!
ip as-path access-list 1 permit ^$
!
!
!
route-map BGP-OUT-POLICY permit 10
 match as-path 1
!
```

Leaf-01

```
router bgp 65777
 bgp log-neighbor-changes
 neighbor 172.16.1.1 remote-as 65777
 neighbor 172.16.1.1 update-source Loopback1
 neighbor 172.16.1.2 remote-as 65777
 neighbor 172.16.1.2 update-source Loopback1
 !
 address-family ipv4
  network 10.11.11.0 mask 255.255.255.0
  network 192.168.99.3 mask 255.255.255.255
  neighbor 172.16.1.1 activate
  neighbor 172.16.1.2 activate
 exit-address-family
!
```

Chapter 10 - Appendix B: Azure – BGP Configuration

Lgw-dc1 JSON View

```
{
    "name": "lgw-dc1",
    "id": "/subscriptions/xxx
    "etag": "W/\"b1c805c5-ef9e-4d6a-95b7-866ed3cdb421\"",
    "type": "Microsoft.Network/localNetworkGateways",
    "location": "swedencentral",
    "properties": {
        "provisioningState": "Succeeded",
        "resourceGuid": "1b066ff9-9f4d-42e3-9ee4-5c841c693b45",
        "localNetworkAddressSpace": {
            "addressPrefixes": [
                "192.168.10.1/32"
            ]
        },
        "gatewayIpAddress": "91.153.26.247",
        "bgpSettings": {
            "asn": 65777,
            "bgpPeeringAddress": "192.168.10.1",
            "peerWeight": 0
        }
    }
}
```

Lgw-dc2 JSON View

```
{
    "name": "lgw-dc2",
    "id": "/subscriptions/,
    "etag": "W/\"f22b72dc-8350-4e83-bc4f-5cf71f6faa70\"",
    "type": "Microsoft.Network/localNetworkGateways",
    "location": "swedencentral",
    "properties": {
        "provisioningState": "Succeeded",
        "resourceGuid": "a40cafcf-68d6-4b22-a01c-5145e7d18b0a",
        "localNetworkAddressSpace": {
            "addressPrefixes": [
                "192.168.10.2/32"
            ]
        },
        "gatewayIpAddress": "91.153.26.247",
        "bgpSettings": {
            "asn": 65777,
            "bgpPeeringAddress": "192.168.10.2",
            "peerWeight": 0
        }
    }
}
```

Vgw-hub1 JSON View

```
{
    "name": "vgw-hub1",
    "id": "/subscriptions/xxx/,
    "etag": "W/\"1849aa5e-fa3b-4243-9269-8ed65127d18c\"",
    "type": "Microsoft.Network/virtualNetworkGateways",
    "location": "swedencentral",
    "tags": {},
    "properties": {
        "provisioningState": "Succeeded",
        "resourceGuid": "96d99253-07f9-486f-b5f0-b677cac2481a",
        "enablePrivateIpAddress": false,
        "ipConfigurations": [
            {
                "name": "default",
                "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworkGateways/vgw-
hub1/ipConfigurations/default",
                "etag": "W/\"1849aa5e-fa3b-4243-9269-8ed65127d18c\"",
                "type": "Microsoft.Network/virtualNetworkGateways/ipConfigura
tions",
                "properties": {
                    "provisioningState": "Succeeded",
                    "privateIPAllocationMethod": "Dynamic",
                    "publicIPAddress": {
                        "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/publicIPAddresses/pip-vgw-hub1"
                    },
                    "subnet": {
                        "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworks/vnet-
hub1/subnets/GatewaySubnet"
                    }
                }
            }
        ],
        "sku": {
            "name": "VpnGw1",
            "tier": "VpnGw1",
            "capacity": 2
        },
        "gatewayType": "Vpn",
        "vpnType": "RouteBased",
        "enableBgp": true,
        "activeActive": false,
        "bgpSettings": {
            "asn": 65515,
            "bgpPeeringAddress": "10.0.1.254",
            "peerWeight": 0,
            "bgpPeeringAddresses": [
                {
                    "ipconfigurationId": "/subscriptions/xxx/resourceGroups/r
g-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworkGateways/vgw-
hub1/ipConfigurations/default",
                    "defaultBgpIpAddresses": [
                        "10.0.1.254"
                    ],
                    "customBgpIpAddresses": [],
                    "tunnelIpAddresses": [
                        "20.240.48.168"
                    ]
                }
            ]
        },
        "remoteVirtualNetworkPeerings": [
            {
                "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworks/vnet-
spoke/virtualNetworkPeerings/pl-spoke-hub1"
            }
        ],
        "vpnGatewayGeneration": "Generation1"
    }
}
```

Vgw-hub2 JSON View

```
{
    "name": "vgw-hub2",
    "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworkGateways/vgw-hub2",
    "etag": "W/\"364e02ca-6347-4325-b532-7cdad3f89c72\"",
    "type": "Microsoft.Network/virtualNetworkGateways",
    "location": "swedencentral",
    "properties": {
        "provisioningState": "Succeeded",
        "resourceGuid": "22759670-21b6-418f-9b2f-57bde50114e3",
        "enablePrivateIpAddress": false,
        "ipConfigurations": [
            {
                "name": "default",
                "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworkGateways/vgw-
hub2/ipConfigurations/default",
                "etag": "W/\"364e02ca-6347-4325-b532-7cdad3f89c72\"",
                "type": "Microsoft.Network/virtualNetworkGateways/ipConfigura
tions",
                "properties": {
                    "provisioningState": "Succeeded",
                    "privateIPAllocationMethod": "Dynamic",
                    "publicIPAddress": {
                        "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/publicIPAddresses/pip-vgw-hub2"
                    },
                    "subnet": {
                        "id": "/subscriptions/xxx/resourceGroups/rg-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworks/vnet-
hub2/subnets/GatewaySubnet"
                    }
                }
            }
        ],
        "sku": {
            "name": "VpnGw1",
            "tier": "VpnGw1",
            "capacity": 2
        },
        "gatewayType": "Vpn",
        "vpnType": "RouteBased",
        "enableBgp": true,
        "activeActive": false,
        "bgpSettings": {
            "asn": 65516,
            "bgpPeeringAddress": "10.1.1.254",
            "peerWeight": 0,
            "bgpPeeringAddresses": [
                {
                    "ipconfigurationId": "/subscriptions/xxx/resourceGroups/r
g-nsg-rt-
swedencentral/providers/Microsoft.Network/virtualNetworkGateways/vgw-
hub2/ipConfigurations/default",
                    "defaultBgpIpAddresses": [
                        "10.1.1.254"
                    ],
                    "customBgpIpAddresses": [],
                    "tunnelIpAddresses": [
                        "20.91.190.74"
                    ]
                }
            ]
        },
        "vpnGatewayGeneration": "Generation1"
    }
}
```

Chapter 11: Virtual WAN Part 1 - S2S VPN and VNet Connections

Introduction

This chapter introduces Azure *Virtual WAN (vWAN)* service. It offers a single deployment, management, and monitoring pane for connectivity services such as Inter-VNet, Site-to-Site VPN, and Express Route. In this chapter, we are focusing on S2S VPN and VNet connections. The Site-to-Site VPN solutions in vWAN differ from the traditional model, where we create resources as an individual components. In the vWAN solution, we only deploy a vWAN resource and manage everything else through its management view. Figure 11-1 illustrates our example topology and deployment order. The first step is to implement a vWAN resource. Then we deploy a *vHub*. It is an Azure-managed VNet to which we assign a CIDR, just like we do with the traditional VNet. We can deploy a vHub as an empty VNet without associating any connection. A vHub deployment process launches a pair of redundant routers, which exchange reachability information with the VNet Gateway router and VGW instances using BGP. We intend to allow Inter-VNet data flows between vnet-swe1, vnet-swe2, and Branch-to-VNet traffic. For Site-to-Site VPN, we deploy VPN Gateway (VGW) into vHub. The VGW started in the vHub creates two *instances*, instance0, and instance1, in active/active mode. We don’t deploy a GatewaySubnet for VGW because Azure handles subnetting and assigns both Public and Private IP addresses to instances. Besides, Azure starts a vHub-specific BGP process and allocates a BGP ASN 65515 to the VGW regardless of the selected S2S routing model (static or dynamic). Note that when we connect VNets and branch site to vHub, the Hub Router exchanges routing information with VNet’s GWs and VGW instance using BGP. After the vHub and VGW deployment, we configure VPN site parameters such as IPsec tunnel endpoint IP address, BGP ASN, and peering IP address for branch device. Then we connect VPN Site to vHub and download the remote device configuration file. The file format is JSON and presents the values/parameters for Site-to-Site VPN and BGP peering but not the device-specific configuration. As a last deployment step, we connect VNets to vHub. The VGW in vHub is associated with a default Route Table (RT), and VNets are associated with none by default. During the connection setup, we need to associate VNets also to default RT. When everything is in place, we verify that each component has the necessary routing information and that the IP connectivity is ok.

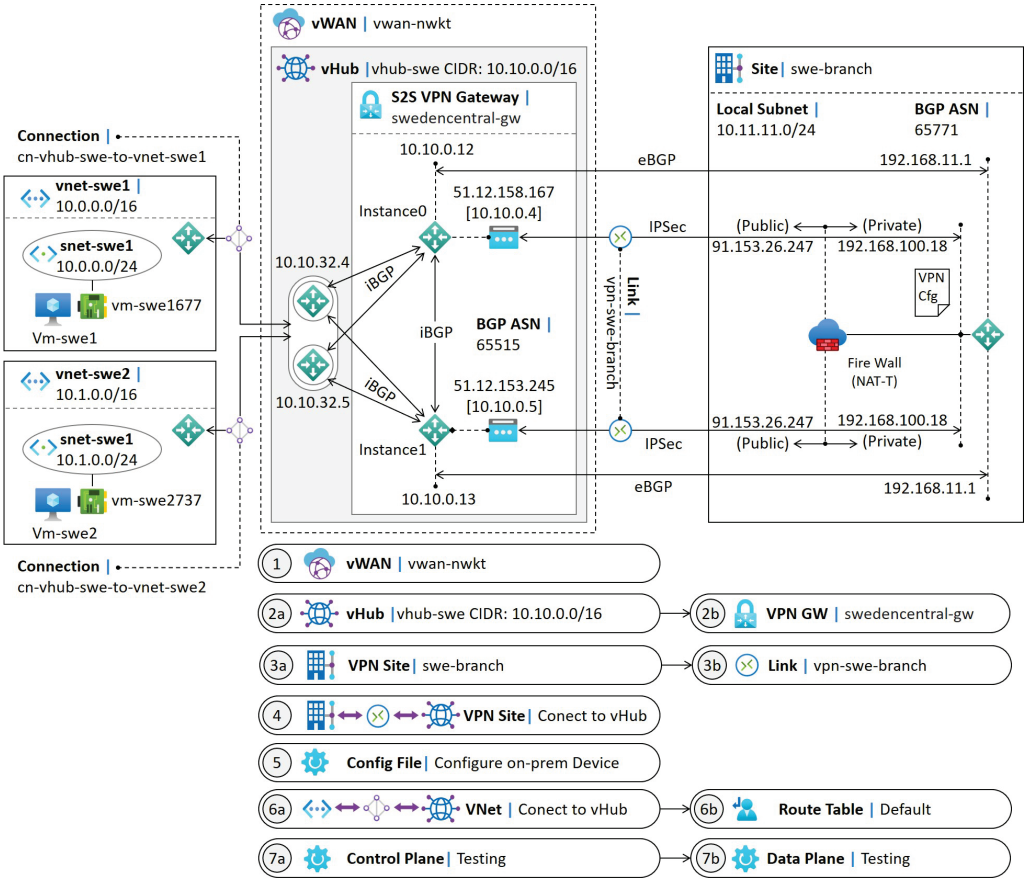


Figure 11-1: *vWAN Diagram.*

Create Virtual WAN (vWAN)

This section shows how to deploy a standard vWAN that supports ExpressRoute, User VPN, Site-to-Site VPN, and VNet-to-Vnet. Besides, Standard vWAN supports an Inter-Vnet connection through the vHub and a full-mesh Inter-vHub architecture. Note that a Basic SKU supports only a Site-to-Site VPN.

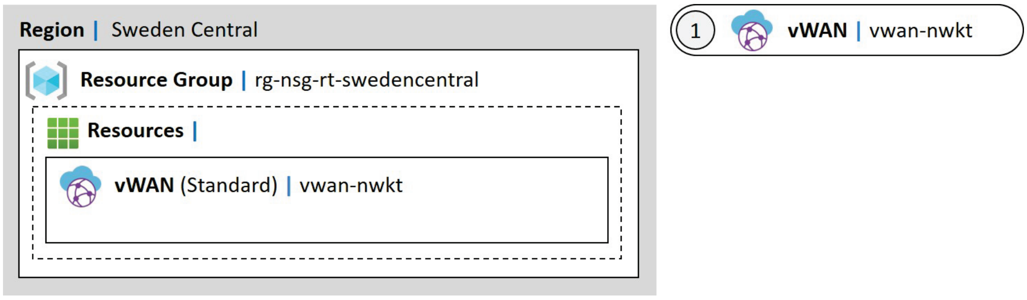
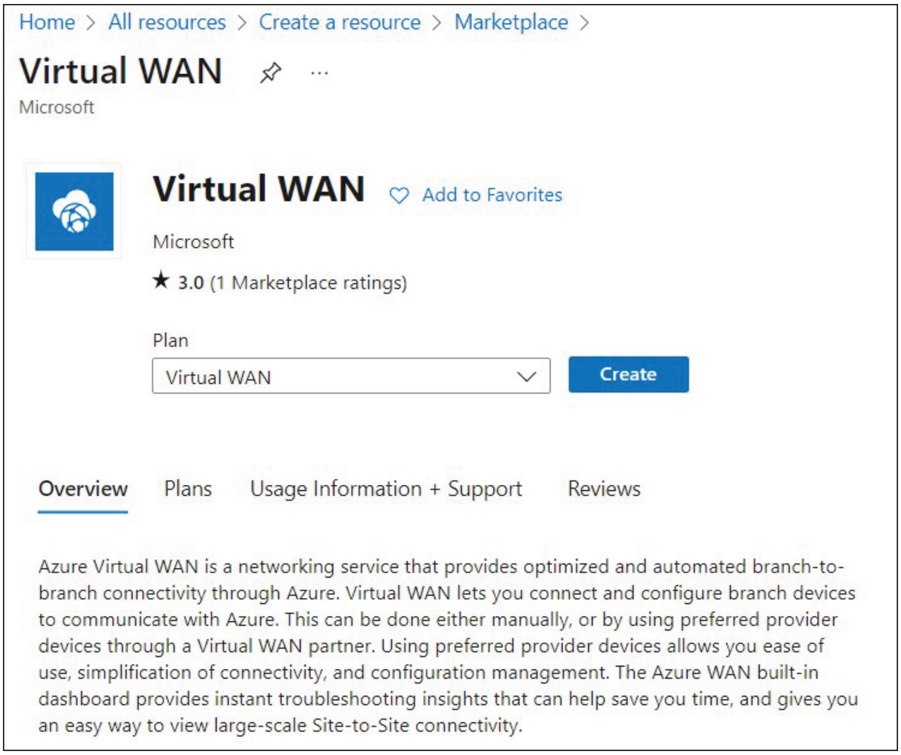

Region | Sweden Central
Resource Group | rg-nsg-rt-swedencentral
Resources |
vWAN (Standard) | vwan-nwkt
1
vWAN | vwan-nwkt

Figure 11-2: *Virtual WAN.*

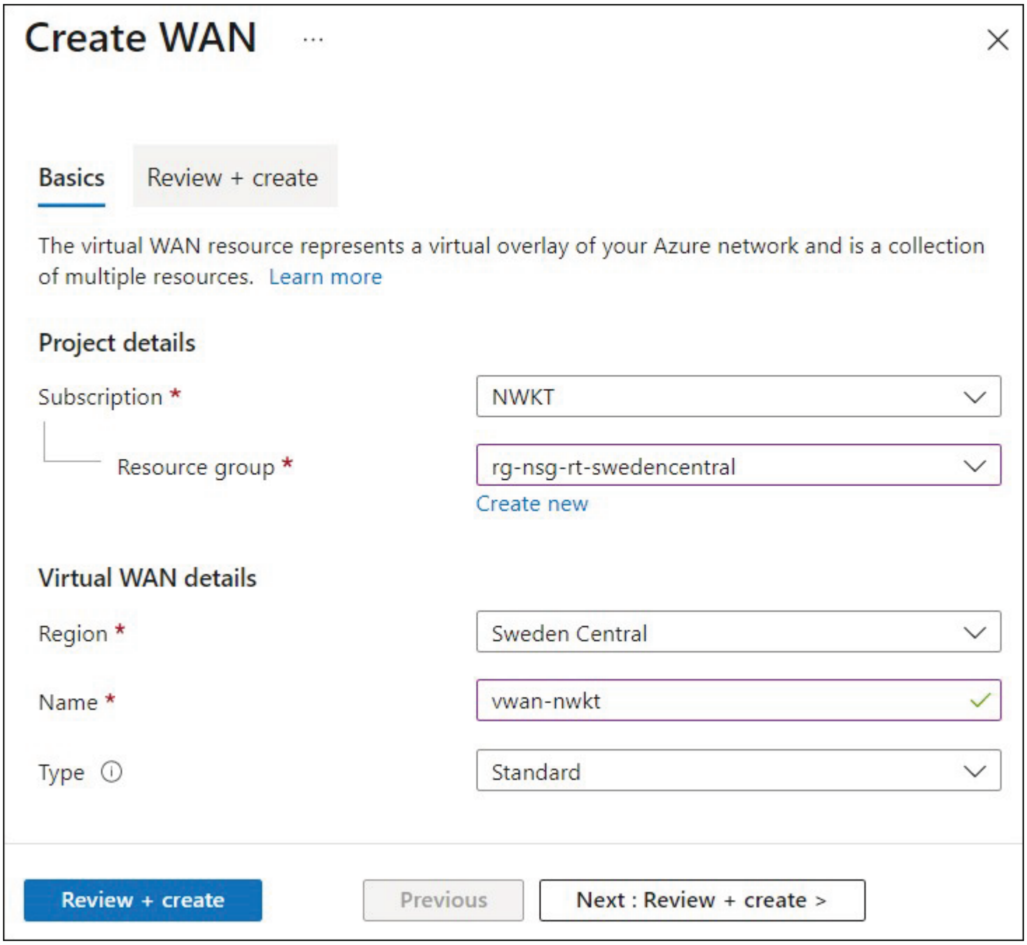
Select the *Create a resource* from the Azure portal Home view. Then type the *Virtual WAN* on the search field. Next, select the Microsoft *Virtual WAN* and click the *Create* button.


Home > All resources > Create a resource > Marketplace >
Virtual WAN
Microsoft
Virtual WAN
Add to Favorites
Microsoft
★ 3.0 (1 Marketplace ratings)
Plan
Virtual WAN
Create
Overview
Plans
Usage Information + Support
Reviews

Azure Virtual WAN is a networking service that provides optimized and automated branch-to-branch connectivity through Azure. Virtual WAN lets you connect and configure branch devices to communicate with Azure. This can be done either manually, or by using preferred provider devices through a Virtual WAN partner. Using preferred provider devices allows you ease of use, simplification of connectivity, and configuration management. The Azure WAN built-in dashboard provides instant troubleshooting insights that can help save you time, and gives you an easy way to view large-scale Site-to-Site connectivity.

Figure 11-3: *Deploy Virtual WAN (vWAN) – Phase#1: Starting Point.*

First, select your Subscription and its Resource group from the drop-down menus. Then pick up the Azure Region where you want to launch vWAN. These values are used in every resource deployed into vWAN. Next, fill in the name field and choose the Standard SKU from the drop-down menu. Verify the configuration and click the *Create* button in the *Review + create* window (not shown in the figure).



The virtual WAN resource represents a virtual overlay of your Azure network and is a collection of multiple resources. Learn more

Project details

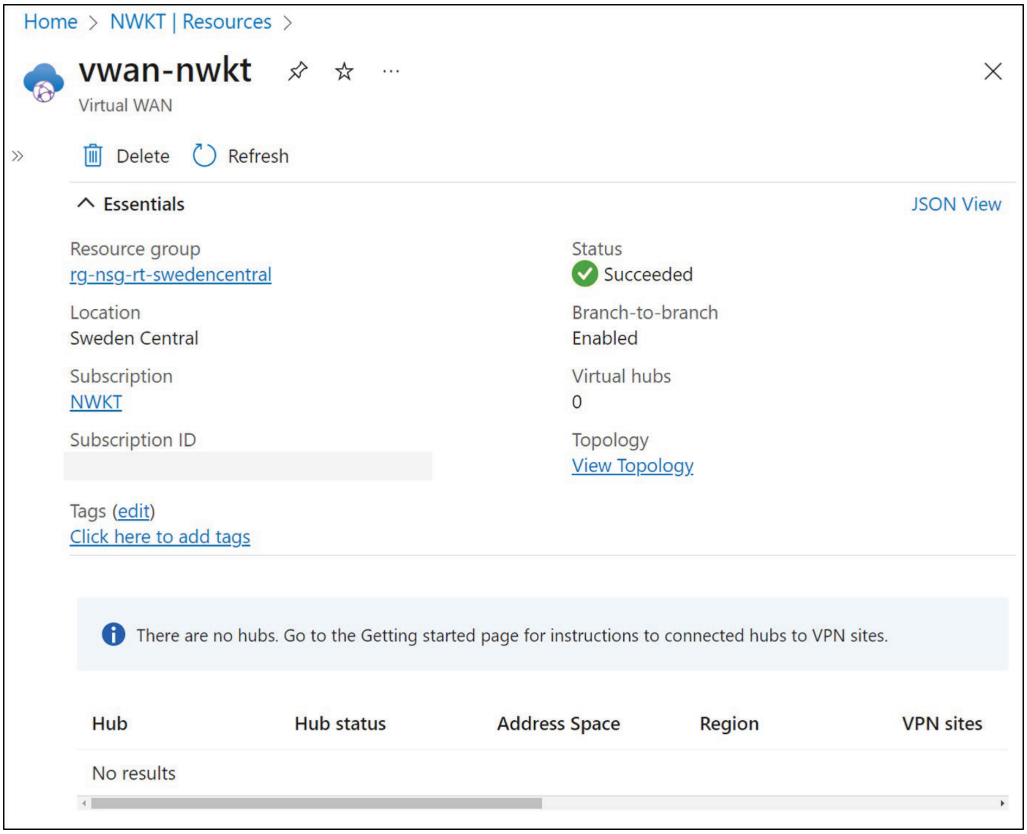
Subscription *	NWKT
Resource group *	rg-nsg-rt-swedencentral
	Create new

Virtual WAN details

Region *	Sweden Central
Name *	vwan-nwkt ✓
Type ⓘ	Standard

Figure 11-4: *Deploy Virtual WAN (vWAN) – Phase#2 - Configuration.*

Figure 11-5 shows the vWAN resource we just created. There is no vHub under vWAN at this phase.


Home > NWKT | Resources >

vwan-nwkt
Virtual WAN

Delete | Refresh

Essentials | JSON View

Resource group	Status
rg-nsg-rt-swedencentral	✓ Succeeded
Location	Branch-to-branch
Sweden Central	Enabled
Subscription	Virtual hubs
NWKT	0
Subscription ID	Topology
	View Topology

Tags (edit)
Click here to add tags

There are no hubs. Go to the Getting started page for instructions to connected hubs to VPN sites.

Hub	Hub status	Address Space	Region	VPN sites
No results				

Figure 11-5: *Deploy Virtual WAN (vWAN) – Phase#3 - Verification.*

Create Virtual Hub with S2S VPN GW

Next, we launch a vHub. Figure 11-6 shows the result after we deployed our example vHub. We assign a CIDR range 10.10.0.0/16 to vHub, where Azure assigns two private IP addresses to both instances. Instance 0 gets Private IP addresses: 1) 10.10.0.4 (tunnel interface), and 2) 10.10.0.12 (BGP peering). Besides, Instance 0 gets a Public IP address 51.12.158.167 for the IPSec tunnel. The same IP addressing schema is used with Instance 1. vHub deployment launches redundant Hub routers. Azure also assigns IP addresses for BGP peering to Hub Routing Infrastructure Units from the vHub CIDR.

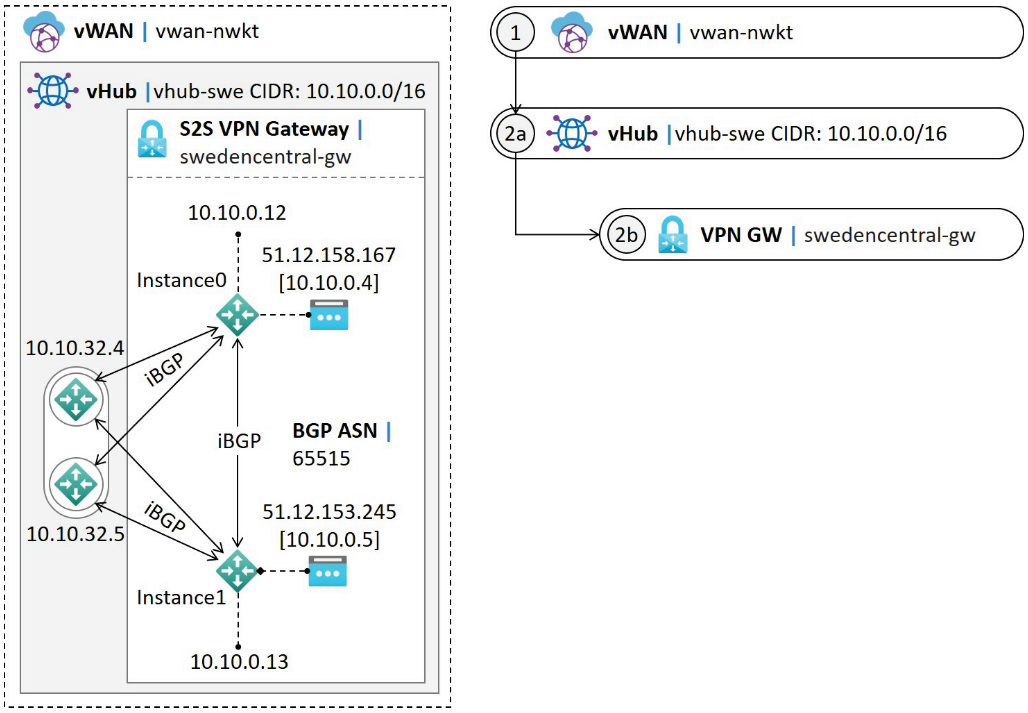
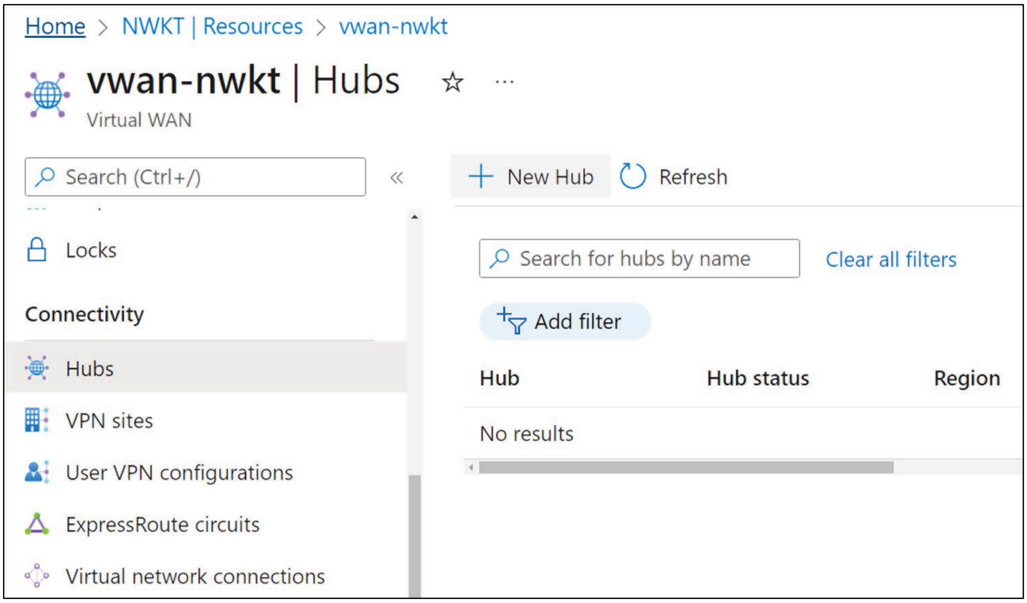


Figure 11-6: *Virtual Hub (vHub).*

To start the vHub deployment, select the Hubs link from the vwan-nwkt *Connectivity* section on the left pane. As the figure shows 11-7, we don't have any vHubs yet. Click the + *New Hub* link to start the vHub deployment.


Home > NWKT | Resources > vwan-nwkt

vwan-nwkt | Hubs
Virtual WAN

Search (Ctrl+/)
Locks
Connectivity
Hubs
VPN sites
User VPN configurations
ExpressRoute circuits
Virtual network connections

+ New Hub   Refresh

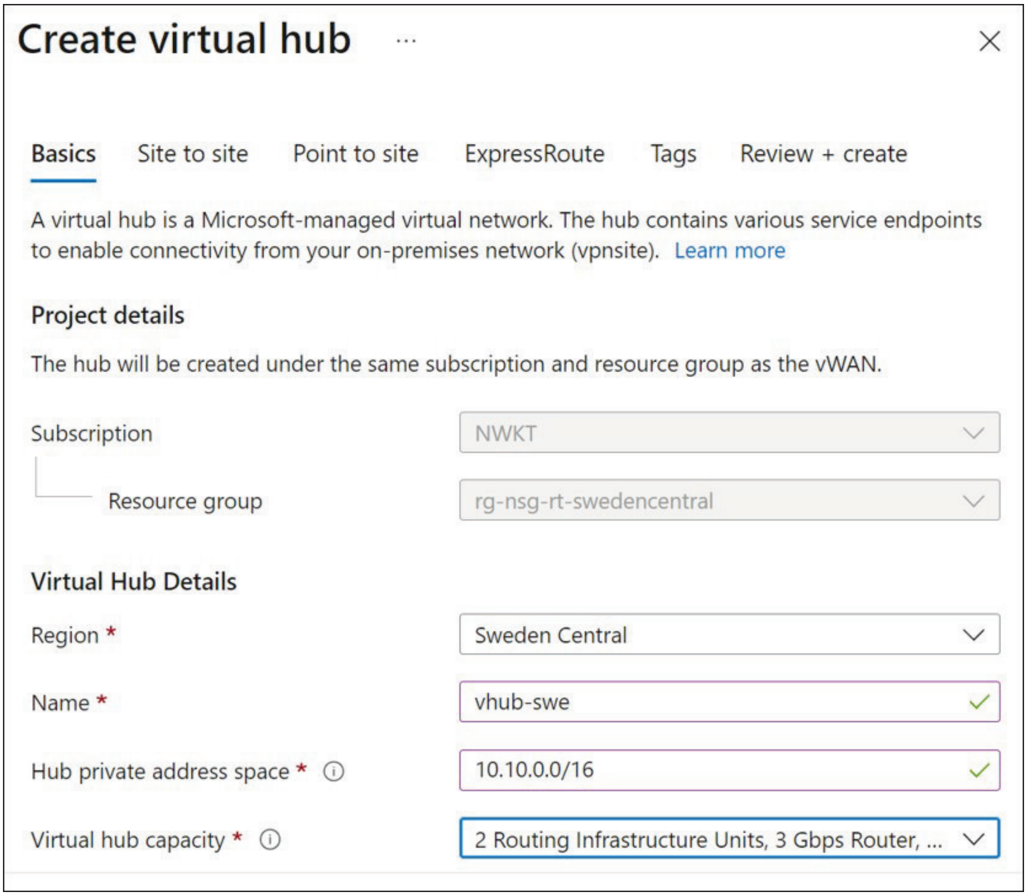
Search for hubs by name   Clear all filters
Add filter

Hub	Hub status	Region
No results		

Figure 11-7: *Virtual Hub Deployment – Phase#1.*

Figure 11-8 shows the Basics tab. Subscription and Resource group are taken from the vWAN setting, and you can't change them. Select the Azure Region and name the vHub. Then, add the CIDR range. The Virtual hub capacity field defines how many Routing Infrastructure Units (RIFs) we deploy to vHub. We have selected the lowest option with two RIFs, which support 3 Gbps throughput and 2000 connected Virtual Machines (VMs). Three RIFs support the same throughput as two RIFs but supported VM count increases from 2000 to 3000. Four RIFs solution supports 4Gbps/4000 VMs, five RIFs solution supports 5Gbps/5000, and so on until we reach the maximum RIF count of 50 with 50Gbps throughput and 50 000 connected VMs.

Note! Virtual hub capacity defines the traffic pattern between RIFs and VNet only. The Gateway scale unit in the Site-to-Site tab describes the aggregated bandwidth between the Site-to-Site Virtual GW and branch over VPN connection. Point to Site and ExpressRoute have dedicated scale units setting.


Create virtual hub

A virtual hub is a Microsoft-managed virtual network. The hub contains various service endpoints to enable connectivity from your on-premises network (vpnsite). Learn more

Project details

The hub will be created under the same subscription and resource group as the vWAN.

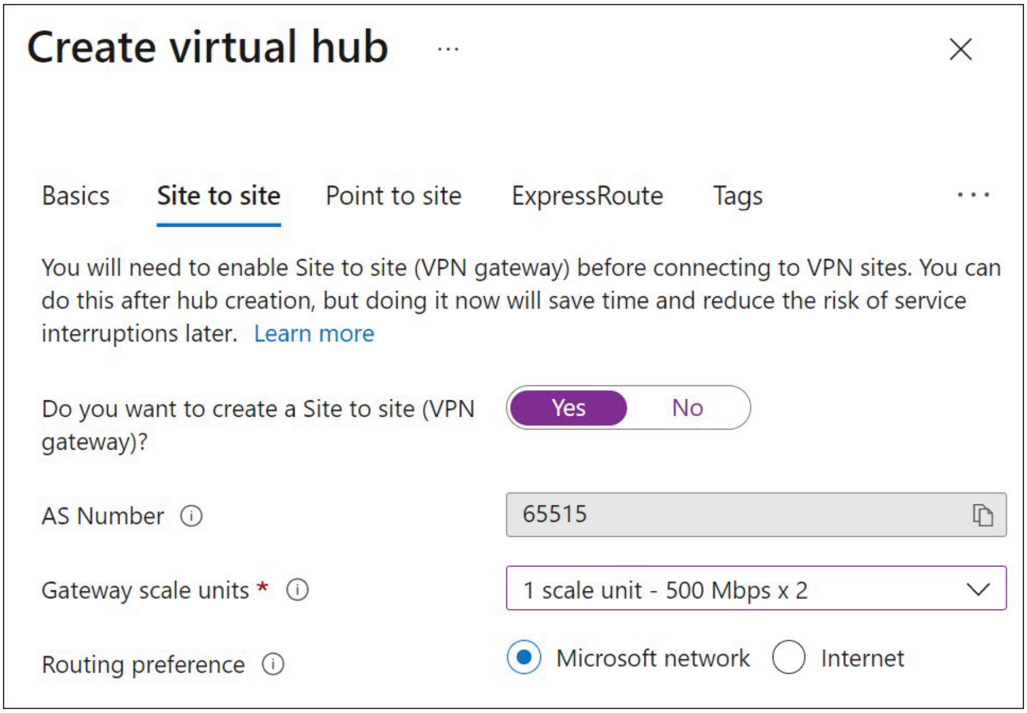
Subscription	NWKT
Resource group	rg-nsg-rt-swedencentral

Virtual Hub Details

Region *	Sweden Central
Name *	vhub-swe
Hub private address space *	10.10.0.0/16
Virtual hub capacity *	2 Routing Infrastructure Units, 3 Gbps Router, ...

Figure 11-8: *Virtual Hub Deployment – Phase#2: Basic Settings.*

We want to create a Site-to-Site VPN connection between Azure and the on-prem branch site, so we answer the *Do you want to...* question with Yes. Azure allocates a BGP ASN number 65515 for S2S VGW, and we can't change it. We have selected one *Gateway scale unit*, which gives 500Mbps to both instances with 1 Gbps total throughput. One scale unit represents 500Mbps for both instances. In other words, one scale unit is 1Gbps. By selecting two scale units, we'll get 2Gbps throughput shared with redundant instances (1 Gbps per instance). This throughput logic applies to client VPN (point-to-site) and ExpressRoute connections. Besides, the VPN Client scale unit also defines the supported VPN client count. Next, select the *Review + Create* tab (not shown in the figure). After successful validation, click the *Create* button.



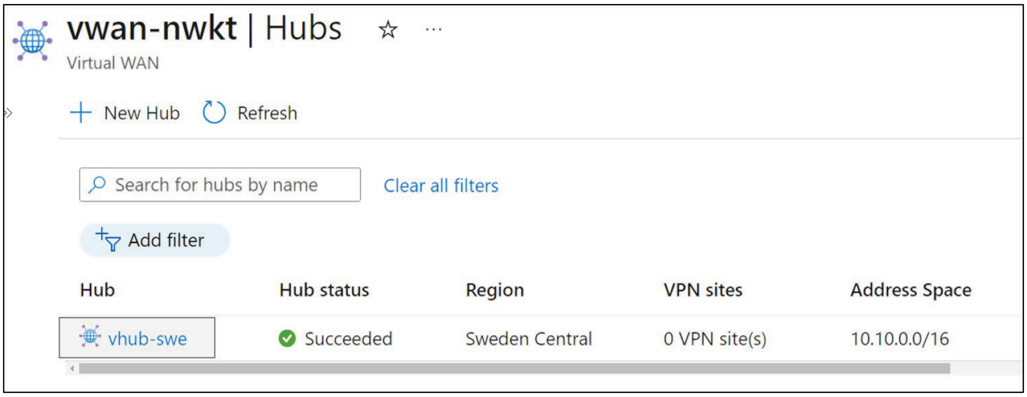
Create virtual hub

You will need to enable Site to site (VPN gateway) before connecting to VPN sites. You can do this after hub creation, but doing it now will save time and reduce the risk of service interruptions later. Learn more

Do you want to create a Site to site (VPN gateway)?	[✓ Yes] No
AS Number	65515
Gateway scale units *	1 scale unit - 500 Mbps x 2
Routing preference	[✓ Microsoft network] Internet

Figure 11-9: *Virtual Hub Deployment – Phase#3: Creating VPN Gateway.*

It takes up to 30 minutes to create vHub with VPN GW. If you don't implement VGW, the vHub deployment time is approximately 5 minutes. After successful deployment, the new vHub appears on the vwan-nwkt Hub list. Hyperlink vhub-swe leads you to the vhub-swe window.



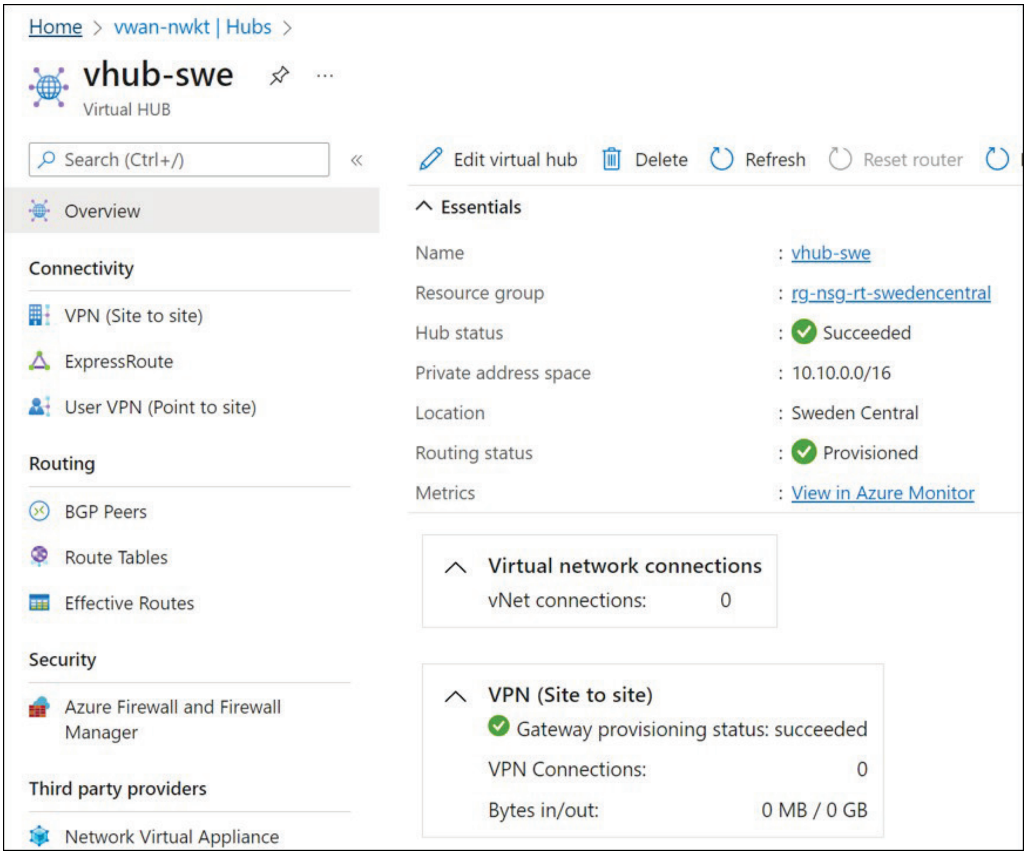
vwan-nwkt | Hubs

Virtual WAN

Hub	Hub status	Region	VPN sites	Address Space
vhub-swe	✓ Succeeded	Sweden Central	0 VPN site(s)	10.10.0.0/16

Figure 11-10: *Virtual Hub Deployment – Phase#4: Verification.*

The *Essentials* section in the *Overview* window shows the basic information like CIDR range and name. It also shows that we have provisioned one Site-to-Site VPN Gateway though we don't have any VPN connection yet. It also shows that there is no VNet connection at this phase.



Name	: vhub-swe
Resource group	: rg-nsg-rt-swedencentral
Hub status	: Succeeded
Private address space	: 10.10.0.0/16
Location	: Sweden Central
Routing status	: Provisioned
Metrics	: View in Azure Monitor

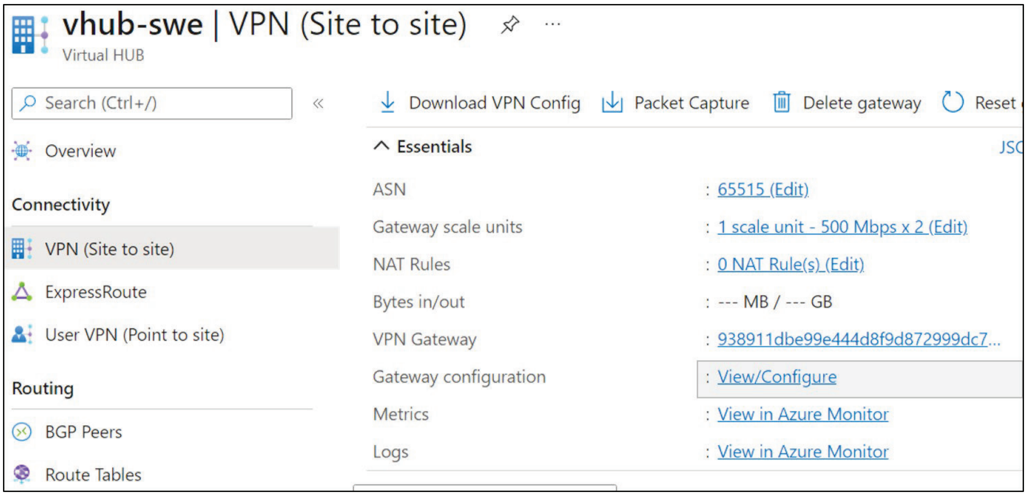
Virtual network connections
vNet connections: 0

VPN (Site to site)
Gateway provisioning status: succeeded
VPN Connections: 0
Bytes in/out: 0 MB / 0 GB

Figure 11-11: *Virtual Hub Overview.*

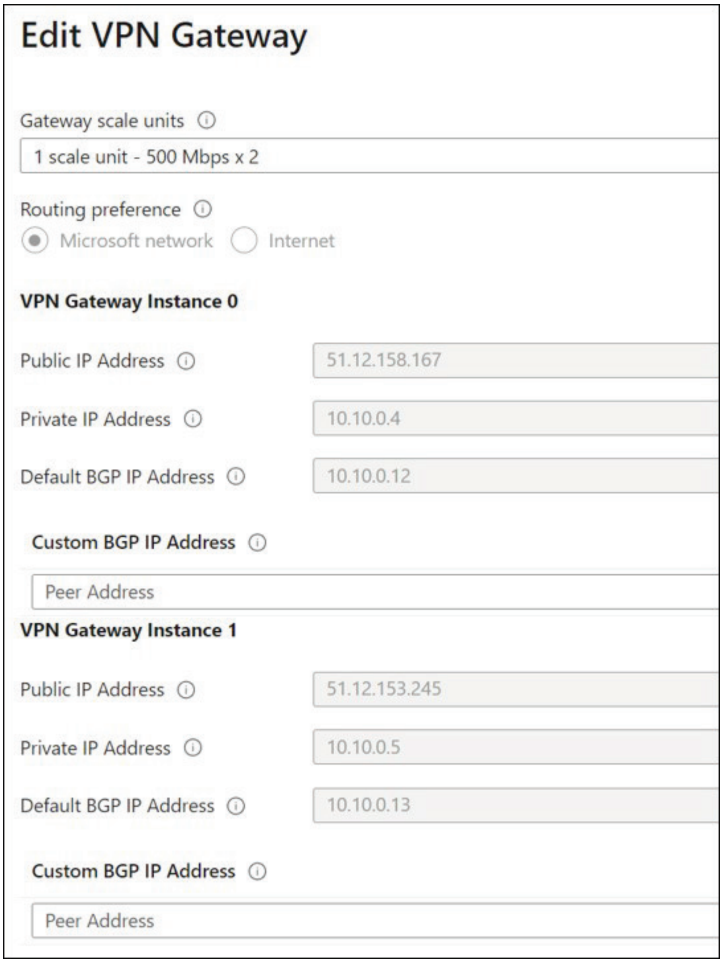
Verifying S2S VPN Gateway

You can enter the Gateway configuration page by selecting the VPN (Site to Site) from the vhub-swe *Connectivity* section in the left pane and then selecting the *View/Configure* hyperlink (figure 11-12). Figure 11-13 shows the configuration page, which shows the IP addresses assigned to VGW. You can adjust a Gateway Scale Unit by selecting the proper option from the drop-down menu.


vhub-swe | VPN (Site to site)
Virtual HUB

ASN	: 65515 (Edit)
Gateway scale units	: 1 scale unit - 500 Mbps x 2 (Edit)
NAT Rules	: 0 NAT Rule(s) (Edit)
Bytes in/out	: --- MB / --- GB
VPN Gateway	: 938911dbe99e444d8f9d872999dc7...
Gateway configuration	: View/Configure
Metrics	: View in Azure Monitor
Logs	: View in Azure Monitor

Figure 11-12: *Verifying VPN GW Configuration - 1.1.*


Edit VPN Gateway

Gateway scale units: 1 scale unit - 500 Mbps x 2

Routing preference: [✓ Microsoft network] Internet

VPN Gateway Instance 0

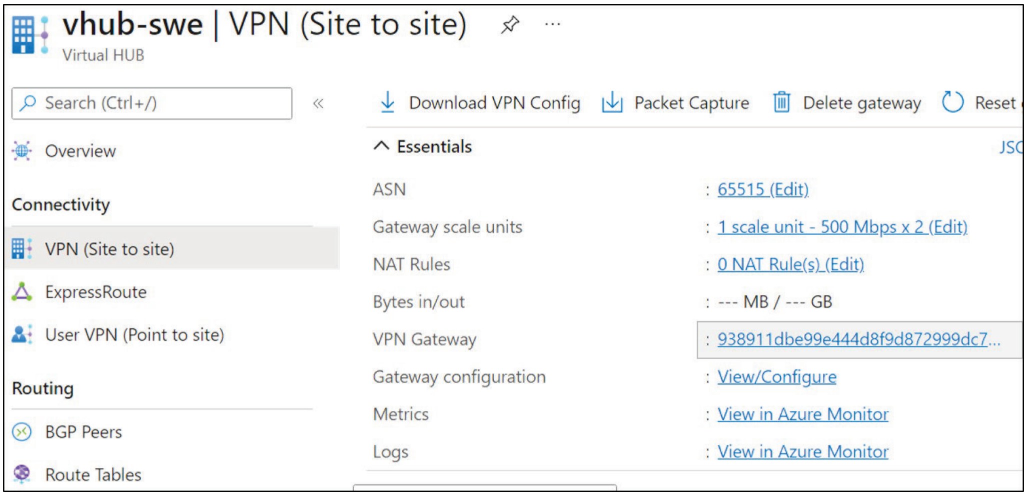
Public IP Address	51.12.158.167
Private IP Address	10.10.0.4
Default BGP IP Address	10.10.0.12
Custom BGP IP Address	Peer Address

VPN Gateway Instance 1

Public IP Address	51.12.153.245
Private IP Address	10.10.0.5
Default BGP IP Address	10.10.0.13
Custom BGP IP Address	Peer Address

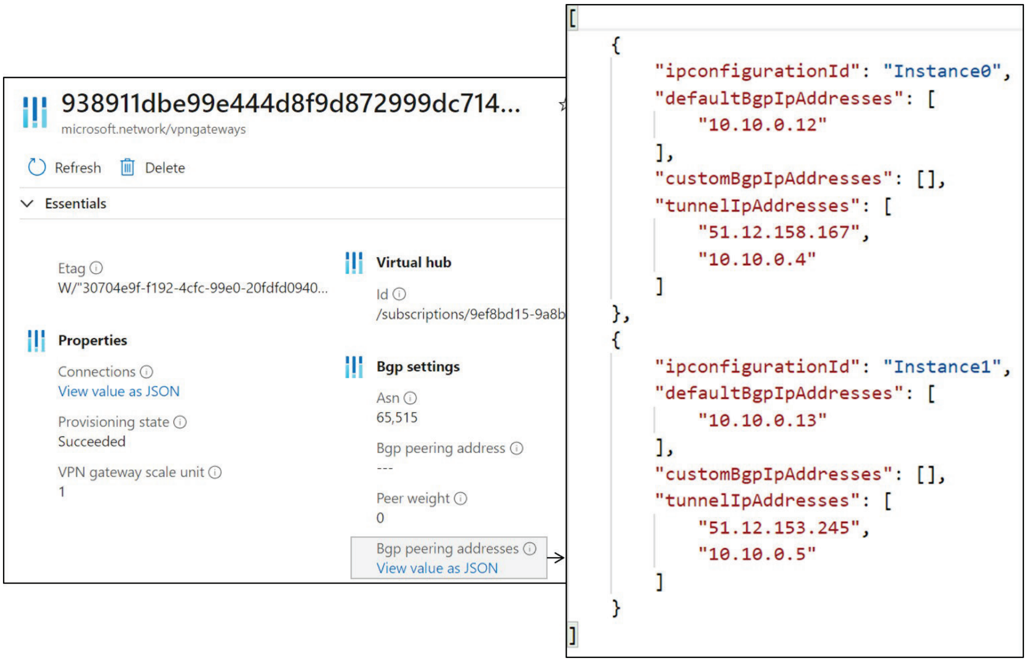
Figure 11-13: *Verifying VPN GW Configuration - 1.2.*

You can also check the VPN GW addressing by selecting the VPN Gateway hyperlink 938911dbe... (figure 11-14). Then, on the VGW window, click the View value as JSON under the BGP peering address field (figure 11-15).



ASN	: 65515 (Edit)
Gateway scale units	: 1 scale unit - 500 Mbps x 2 (Edit)
NAT Rules	: 0 NAT Rule(s) (Edit)
Bytes in/out	: --- MB / --- GB
VPN Gateway	: 938911dbe99e444d8f9d872999dc7...
Gateway configuration	: View/Configure
Metrics	: View in Azure Monitor
Logs	: View in Azure Monitor

Figure 11-14: *Verifying VPN GW Configuration - 2.1.*


938911dbe99e444d8f9d872999dc714...
microsoft.network/vpngateways
Refresh
Delete
Essentials

Etag
W/"30704e9f-f192-4cfc-99e0-20fdfd0940...

Properties
Connections
View value as JSON
Provisioning state
Succeeded
VPN gateway scale unit
1

Virtual hub
Id
/subscriptions/9ef8bd15-9a8b

Bgp settings
Asn
65,515
Bgp peering address

Peer weight
0
Bgp peering addresses
View value as JSON

```
[
    {
        "ipconfigurationId": "Instance0",
        "defaultBgpIpAddresses": [
            "10.10.0.12"
        ],
        "customBgpIpAddresses": [],
        "tunnelIpAddresses": [
            "51.12.158.167",
            "10.10.0.4"
        ]
    },
    {
        "ipconfigurationId": "Instance1",
        "defaultBgpIpAddresses": [
            "10.10.0.13"
        ],
        "customBgpIpAddresses": [],
        "tunnelIpAddresses": [
            "51.12.153.245",
            "10.10.0.5"
        ]
    }
]
```

Figure 11-15: *Verifying VPN GW Configuration - 2.2.*

Create VPN Site

VPN Site configuration defines IPSec tunnel and BGP peering settings between VPN Gateway instances and remote site edge devices. Our branch site edge router Swe-Branch is in the private IP subnet 192.168.100.0/24. The Internet Firewall hides source IP addresses on all data flows towards the Internet. For this reason, we set the VPN Gateway Connection mode to *Responder Only*. This way, the remote end router Swe-Branch always starts the IPSec tunnel negotiation (initiator). VPN gateway instances have BGP peering over IPsec tunnels with the remote router Swe-Branch (192.168.11.1). Note that the BGP peering address of the remote end has to be different than its IPSec tunnel IP address. We define the remote end's BGP peering address and BGP ASN during the VPN Site configuration. Unlike in IPSec tunnel negotiation, VGW instances and remote end routers can initiate BGP peering negotiation. After deploying the VPN Site configuration, we connect the site to vHub. Then we download the remote edge IPSec and BGP parameters, convert them into remote edge device configuration commands, and copy them to the remote end router.

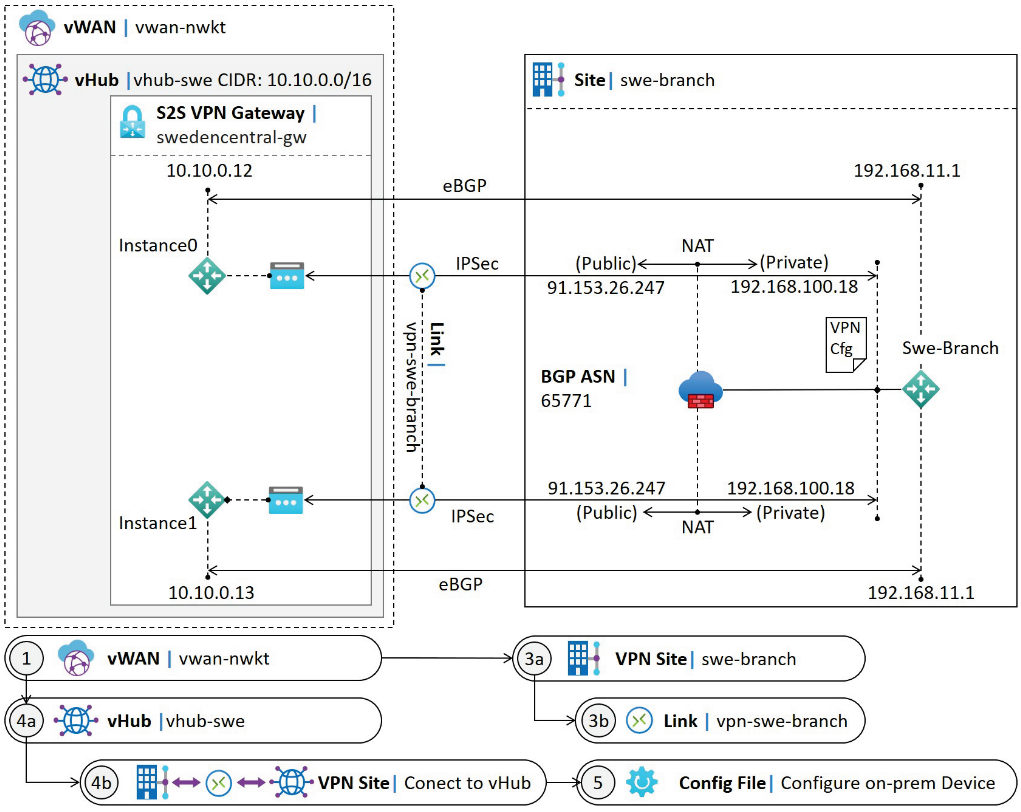


Figure 11-16: *VPN Site Configuration Overview.*

Start the VPN Site deployment by selecting the VPN sites link from the vwan-nwkt *Connectivity* section on the left pane. Click the + *Create site* tab to start the deployment.

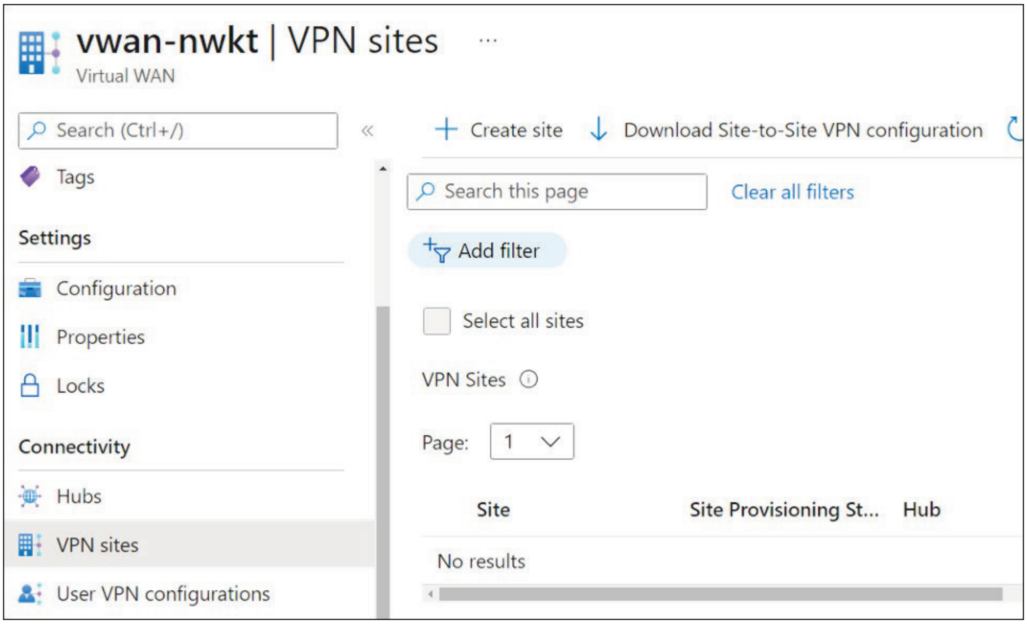
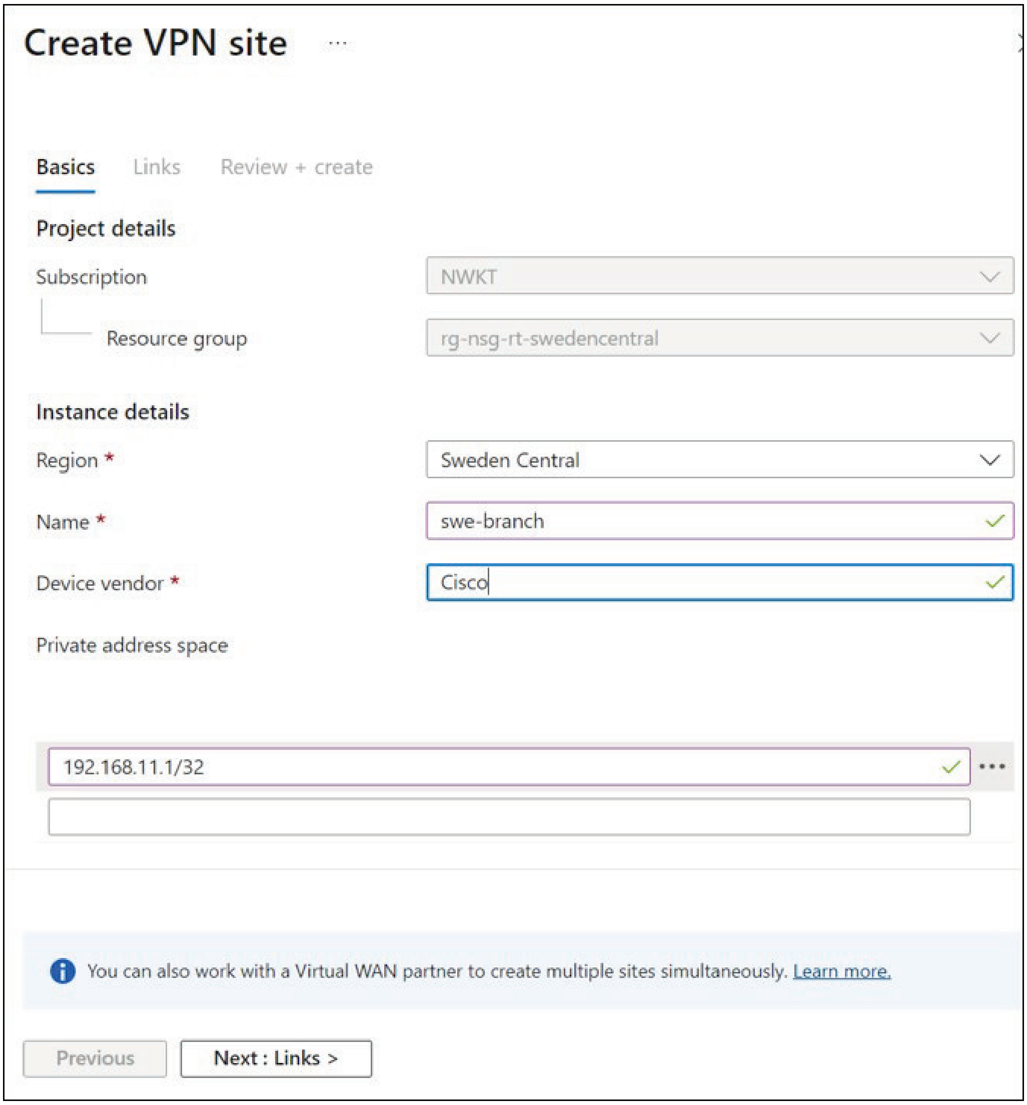


Figure 11-17: *Create VPN Site – Phase#1: Starting Point.*

Figure 11-18 shows the Basics tab. Subscription and Resource group are taken from the vWAN setting. Select the Azure Region. Then, name your VPN site and specify the remote end edge device vendor. The *Private address space* field generates static routes for the VPN site’s local subnets. We only need to define the BGP peering address of Swe-Branch. Other remote sites’ local subnets' reachability information is advertised using BGP.



Project details

Subscription: NWKT

Resource group: rg-nsg-rt-swedencentral

Instance details

Region *: Sweden Central

Name *: swe-branch

Device vendor *: Cisco

Private address space

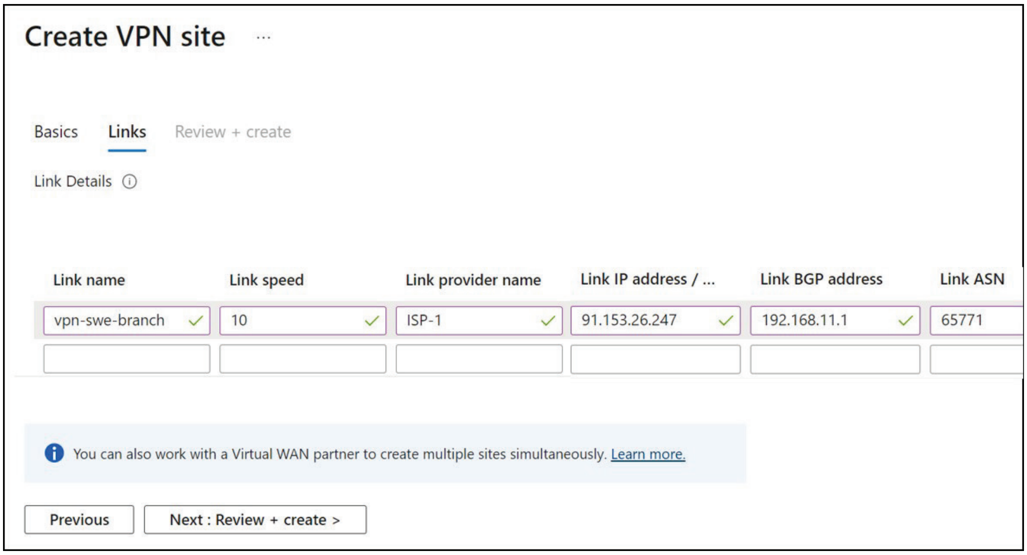
192.168.11.1/32

You can also work with a Virtual WAN partner to create multiple sites simultaneously. Learn more.

Figure 11-18: *Create VPN Site – Phase#2: Basic Information.*

Proceed to the *Link* page. Fill in the Link name, Link Speed, and Link provider fields. The Link IP address defines the Public IP address of Swe-Branch. VGW instances use this IP address for the IPSec tunnel destination address. Remember that the remote end edge router Swe-Branch is behind the NAT Firewall, which translates the original IP address 192.168.100.18 (private) to 91.153.26.247 (public). Link BGP address field defines the remote edge router Swe-Branch's BGP peering address. The last field is the VPN Site BGP ASN.

The following list describes the BGP ASN that you can't use in your VPN site when peering with Azure: 1) Azure Public ASN: 8074, 8075, and 12076, 2) Azure Private ASN: 65515, 65517, and 65518-65520, and 3) Reserved by IANA: 23456, 64496-64511, and 65535-65551. Go to *Review* + *Create* to validate your configuration and start the deployment.


Create VPN site ...

Basics | Links | Review + create

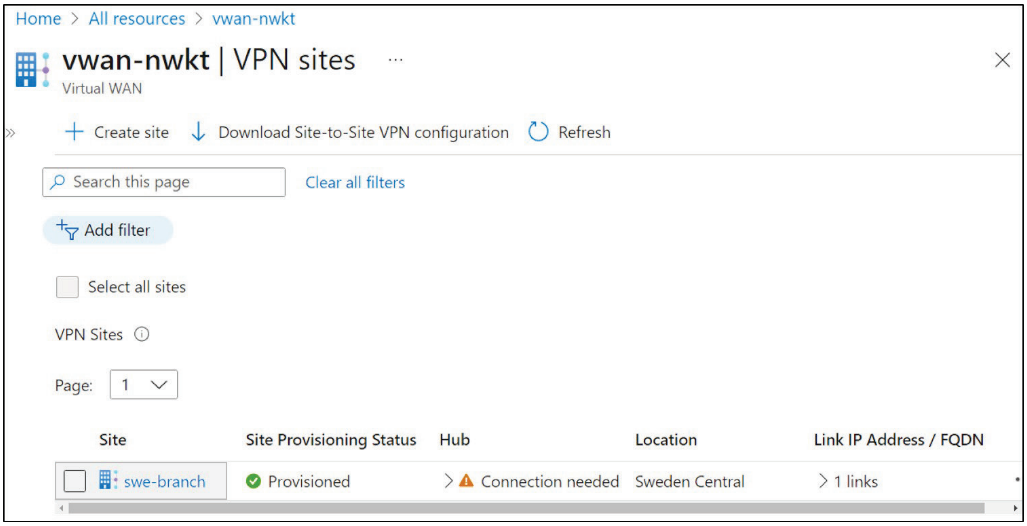
Link Details ⓘ

Link name	Link speed	Link provider name	Link IP address / ...	Link BGP address	Link ASN
vpn-swe-branch ✓	10 ✓	ISP-1 ✓	91.153.26.247 ✓	192.168.11.1 ✓	65771

ⓘ You can also work with a Virtual WAN partner to create multiple sites simultaneously. Learn more.

Figure 11-19: *Create VPN Site – Phase#3: Link Information.*

Figure 11-20 shows that the *Site Provisioning Status* is *Provisioned*. We haven't connected the site to vHub yet, and the status of Hub is *Connection needed*. Click the *swe-site* hyperlink under the *Site* header to navigate to the VPN site page.


Home > All resources > vwan-nwkt

vwan-nwkt | VPN sites ...

Virtual WAN

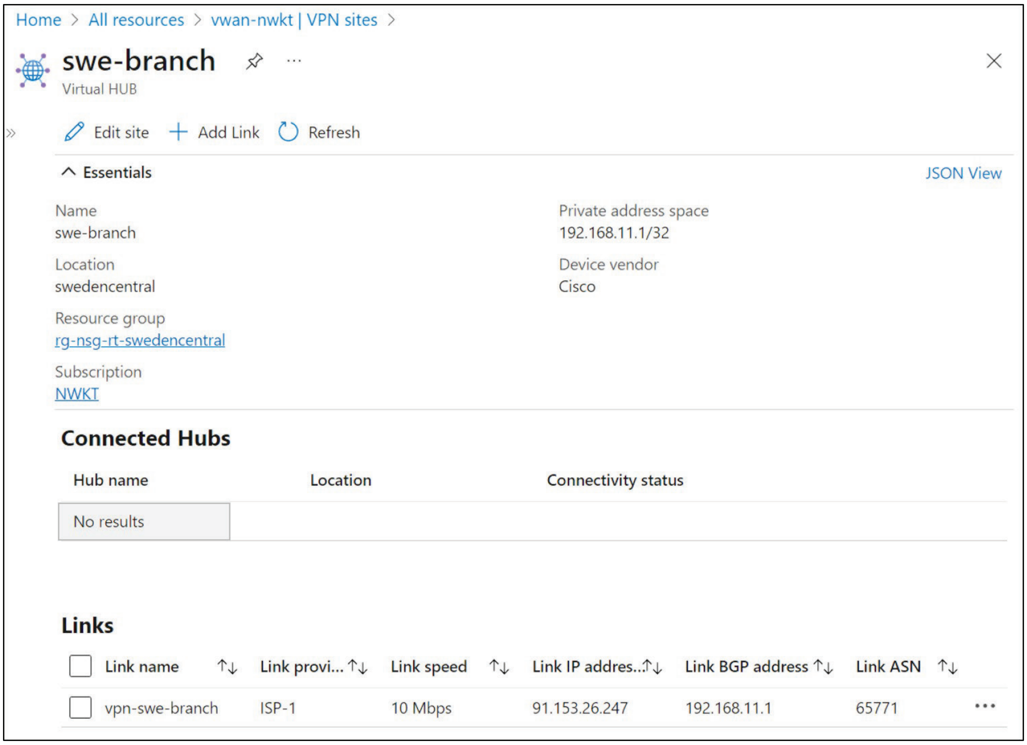
VPN Sites ⓘ

Page: 1

Site	Site Provisioning Status	Hub	Location	Link IP Address / FQDN
swe-branch	✓ Provisioned	> ⚠ Connection needed	Sweden Central	> 1 links

Figure 11-20: *Create VPN Site – Phase#3: Link Information.*

The *Essentials* section shows the VPN Site's basic information, such as Private address space (statically defined local subnets) and the edge device vendor. The *Connected Hubs* section is still empty. Our next step is to build a connection between swe-branch and vHub. The *Links* section describes link configuration.



Name
swe-branch

Location
swedencentral

Resource group
rg-nsg-rt-swedencentral

Subscription
NWKT

Private address space
192.168.11.1/32

Device vendor
Cisco

Connected Hubs

Hub name	Location	Connectivity status
No results		

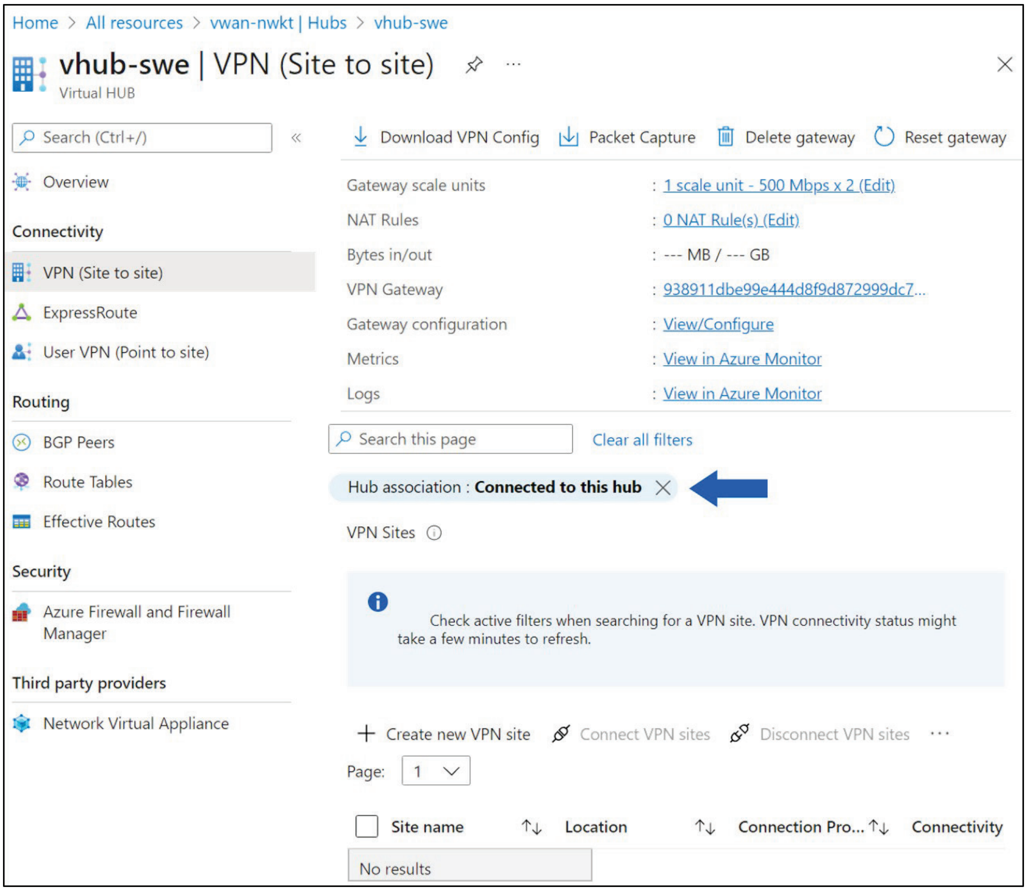
Links

Link name	Link provi...	Link speed	Link IP addres...	Link BGP address	Link ASN
vpn-swe-branch	ISP-1	10 Mbps	91.153.26.247	192.168.11.1	65771

Figure 11-21: *Create VPN Site – Phase#3: Link Information.*

VPN site to vHub connection

Our next task is to connect VPN Site to vHub. First, navigate to the vhub-swe page. If your VPN Site is not visible in the site list, like in figure 11-22, clear the *Hub association* filter. In figure 11-23, we have removed the *Hub associat*ion filter, and then, as figure 11-23 shows, the site swe-branch is in the site list.


Home > All resources > vwan-nwkt | Hubs > vhub-swe

vhub-swe | VPN (Site to site)

Virtual HUB

Gateway scale units	: 1 scale unit - 500 Mbps x 2 (Edit)
NAT Rules	: 0 NAT Rule(s) (Edit)
Bytes in/out	: --- MB / --- GB
VPN Gateway	: 938911dbe99e444d8f9d872999dc7...
Gateway configuration	: View/Configure
Metrics	: View in Azure Monitor
Logs	: View in Azure Monitor

Hub association : **Connected to this hub**

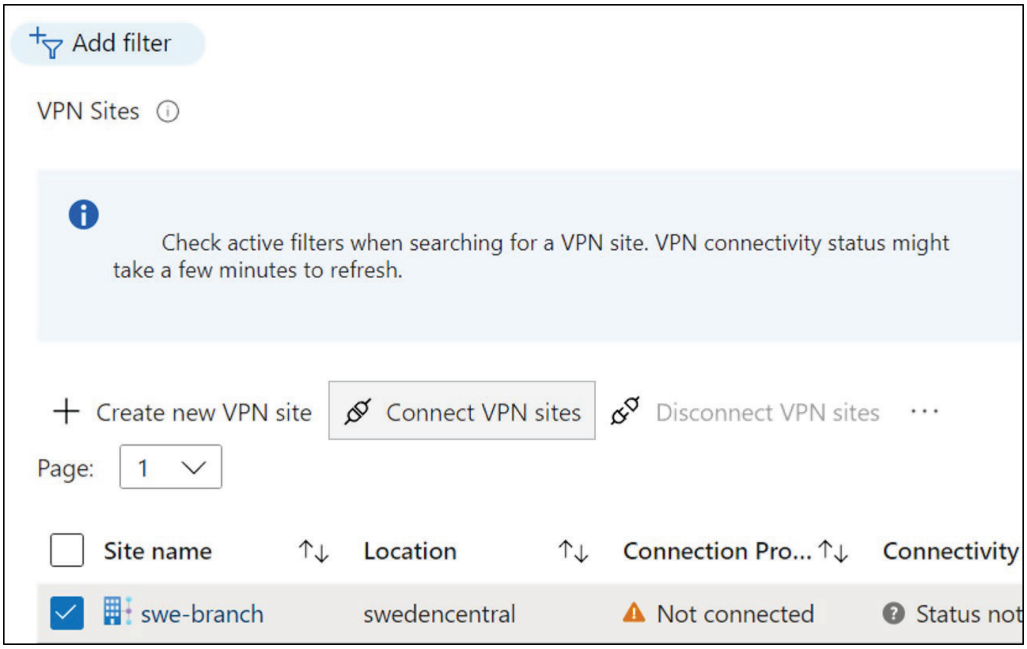
VPN Sites

Check active filters when searching for a VPN site. VPN connectivity status might take a few minutes to refresh.

Site name	Location	Connection Pro...	Connectivity
No results			

Figure 11-22: *Connect VPN Site to vHub – Phase#1: Starting Phase.*

Select your site and choose the *Connect VPN sites* option.

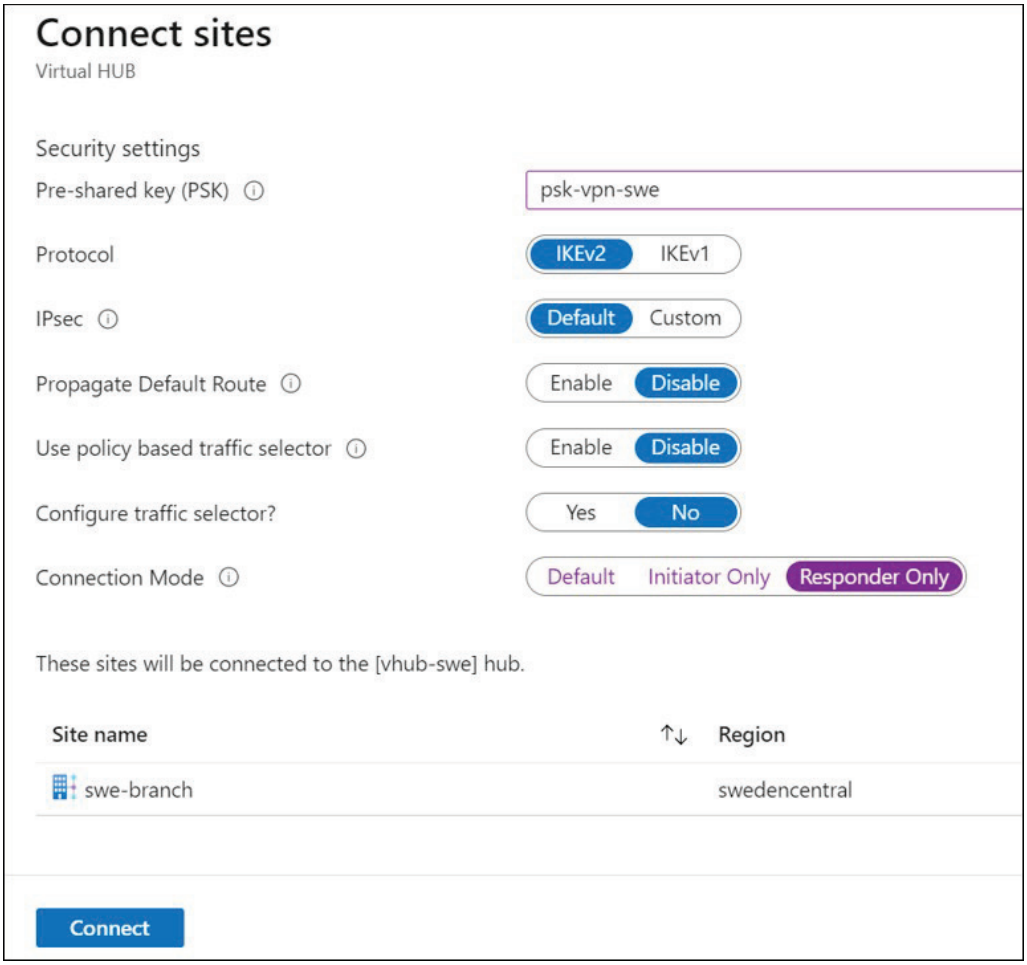


Check active filters when searching for a VPN site. VPN connectivity status might take a few minutes to refresh.

	Site name	Location	Connection Pro...	Connectivity
✓	swe-branch	swedencentral	Not connected	Status not

Figure 11-23: *Connect VPN Site to vHub – Phase#2: Connect VPN Sites.*

Type your pre-shared key in the Per-shared key (PSK) field. Leave all other selectors to their defaults but change the Connection mode from Default to *Responder Only*. Then click the *Connect* button.



Connect sites

Virtual HUB

Security settings

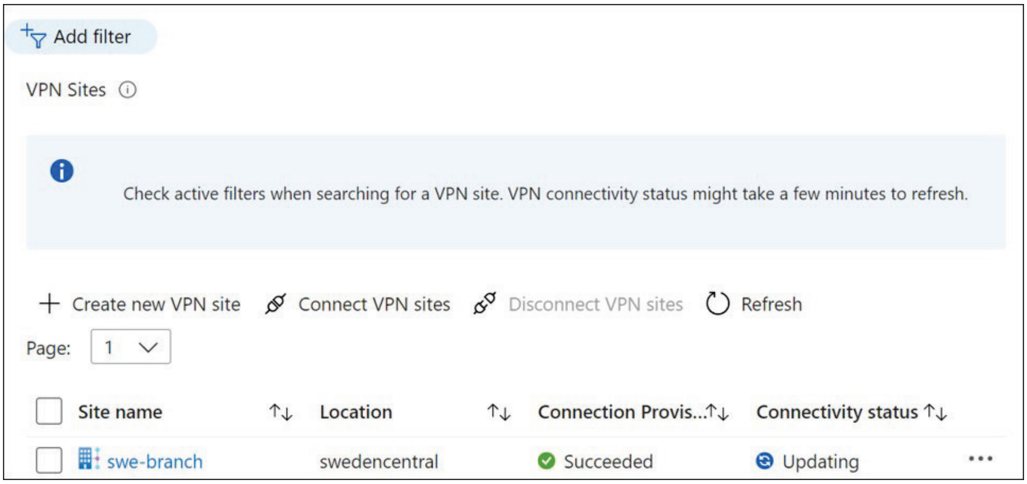
Pre-shared key (PSK)	psk-vpn-swe
Protocol	[✓ IKEv2] IKEv1
IPsec	[✓ Default] Custom
Propagate Default Route	Enable [✓ Disable]
Use policy based traffic selector	Enable [✓ Disable]
Configure traffic selector?	Yes [✓ No]
Connection Mode	Default Initiator Only [✓ Responder Only]

These sites will be connected to the [vhub-swe] hub.

Site name	Region
swe-branch	swedencentral

Figure 11-24: *Connect VPN Site to vHub – Phase#3: Define Connection Settings.*

After the successful deployment process, the *Connection Provisioned* field changes from *Not connected* to *Succeeded*. The Connectivity status is *Updating* because we haven't configured the remote end device yet.

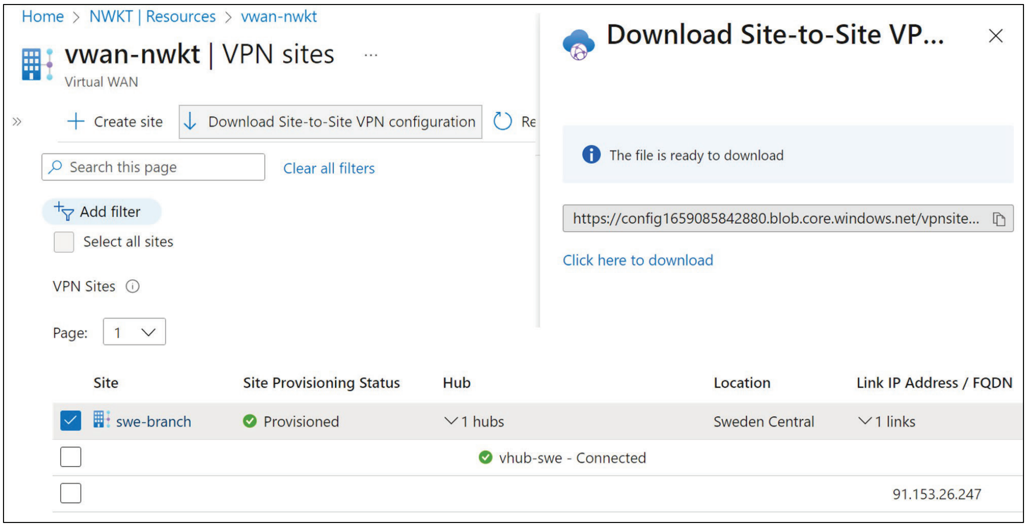


Check active filters when searching for a VPN site. VPN connectivity status might take a few minutes to refresh.

Site name	Location	Connection Provis...	Connectivity status
swe-branch	swedencentral	Succeeded	Updating

Figure 11-25: *Connect VPN Site to vHub – Phase#4: Verification.*

Next, navigate to the vwan-nwkt page and select *VPN sites* under the *Connectivity* section. Then choose your site. Click the *Download Site-to-Site VPN configuration* button. It opens a popup window *Download Site-to-Site VPN Configuration*, where you can download the configuration parameters.



The file is ready to download

https://config1659085842880.blob.core.windows.net/vpnsite...

Site	Site Provisioning Status	Hub	Location	Link IP Address / FQDN
swe-branch	Provisioned	1 hubs	Sweden Central	1 links
		vhub-swe - Connected		
				91.153.26.247

Figure 11-26: *Connect VPN Site to vHub – Phase#5: Download S2S VPN Configuration.*

Verifying S2S VPN Gateway

The configuration parameters file is in JSON format but not displayed as an object-tree structure. I have used a *JSON Viewer plugin* of Notepad ++ to change the original unorganized file to an object tree. You can install the JSON viewer by selecting the *Plugin Admin* option under the *Plugins* tab. Then type the JSON Viewer into the search field. Mark the JSON Viewer and click the *Install* button. Note, you don't have to change the presentation view, you can pick up all the information from the original file too. The converted file is on the next page (example 11-2).

```
[
  {"configurationVersion":{"LastUpdatedTime":"2022-07-
29T09:33:28.7450371Z","Version":"c67b4f2b-07b7-4c02-9c8d-
9fdce7b1d053"},"vpnSiteConfiguration":{"Name":"swe-
branch","IPAddress":"91.153.26.247","BgpSetting":{"Asn":235,"BgpPeeringAddress":"192.1
68.11.1","BgpPeeringAddresses":null,"PeerWeight":32768},"LinkName":"vpn-swe-
branch","Office365Policy":{"BreakOutCategories":{"Optimize":false,"Allow":false,"Defau
lt":false}}},"vpnSiteConnections":[{"hubConfiguration":{"AddressSpace":"10.10.0.0/16",
"Region":"SwedenCentral"},"gatewayConfiguration":{"IpAddresses":{"Instance0":"51.12.15
8.167","Instance1":"51.12.153.245"},"BgpSetting":{"Asn":65515,"BgpPeeringAddresses":{"
Instance0":"10.10.0.12","Instance1":"10.10.0.13"},"CustomBgpPeeringAddresses":{},"Peer
Weight":0}},"connectionConfiguration":{"IsBgpEnabled":true,"PSK":"psk-vpn-
swe","IPsecParameters":{"SADataSizeInKilobytes":102400000,"SALifeTimeInSeconds":3600}}
}]}
]
```

Example 11-1: *Original Configuration File.*

The *vpnSiteConfiguration* object defines the BGP peering IP address (192.168.11.1) of the VPN site edge-router Swe-Branch. Create a Loopback interface to a remote device, to which you assign this address. Then use it as an update-source address in BGP neighbor configurations. Example 11-3 shows the complete Swe-Branch configuration. It is the same configuration file that I used in chapter 6. I have only changed IPSec and BGP configurations. The next object, *VpnSiteConnection*, defines the vHub CIDR range 10.10.0.0/16. You can create a static route that routes the subnet 10.10.0.0/16 to tunnel interfaces. Note that vHub instances advertise reachability information of the subnet 10.10.0.0/16 with BGP over both tunnels. If you are using static routes to 10.10.0.0/16, create two routes, one towards tunnel 11 and the other one towards tunnel 12. Public IP addresses of instance0 and instance1 are used in several configuration sections: 1) ikev2 keyring, 2) ikev2 profile, and 3) tunnel destination. The Pre-Shared Key is used under the ikev2 keyring section. The SA data Size 102400000 is used in the ipsec profile section. BGP peering address is used in BGP peer configuration.

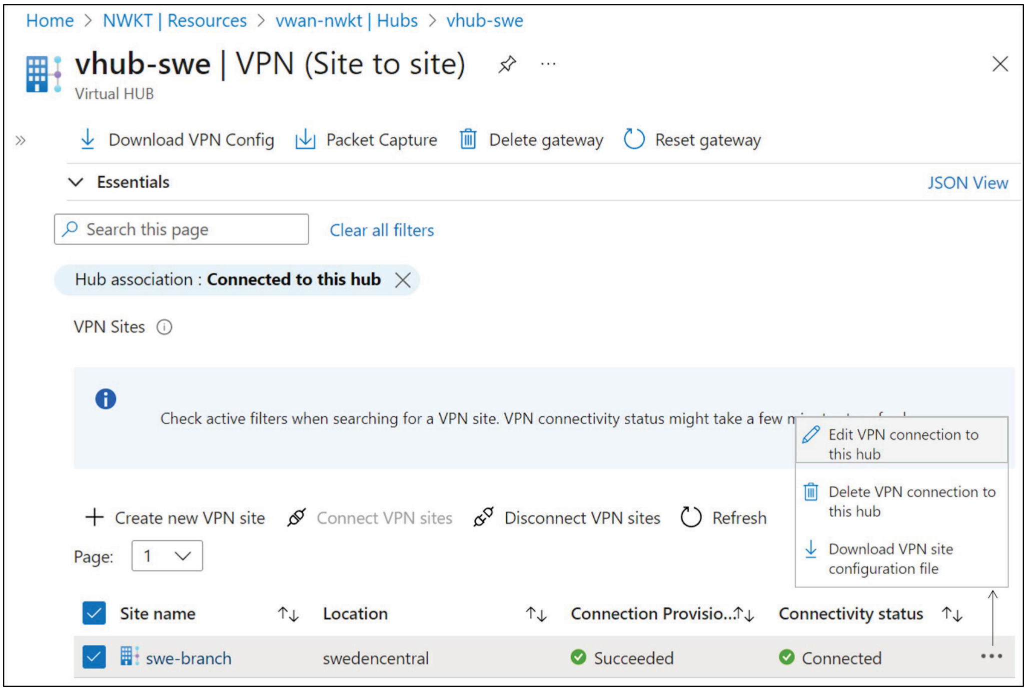
```
[
        {
                "configurationVersion": {
                        "LastUpdatedTime": "2022-07-29T09:33:28.7450371Z",
                        "Version": "c67b4f2b-07b7-4c02-9c8d-9fdce7b1d053"
                },
                "vpnSiteConfiguration": {
                        "Name": "swe-branch",
                        "IPAddress": "91.153.26.247",
                        "BgpSetting": {
                                "Asn": 235,
                                "BgpPeeringAddress": "192.168.11.1",
                                "BgpPeeringAddresses": null,
                                "PeerWeight": 32768
                        },
                        "LinkName": "vpn-swe-branch",
                        "Office365Policy": {
                                "BreakOutCategories": {
                                        "Optimize": false,
                                        "Allow": false,
                                        "Default": false
                                }
                        }
                },
                "vpnSiteConnections": [
                        {
                                "hubConfiguration": {
                                        "AddressSpace": "10.10.0.0/16",
                                        "Region": "Sweden Central"
                                },
                                "gatewayConfiguration": {
                                        "IpAddresses": {
                                                "Instance0": "51.12.158.167",
                                                "Instance1": "51.12.153.245"
                                        },
                                        "BgpSetting": {
                                                "Asn": 65515,
                                                "BgpPeeringAddresses": {
                                                        "Instance0": "10.10.0.12",
                                                        "Instance1": "10.10.0.13"
                                                },
                                                "CustomBgpPeeringAddresses": {},
                                                "PeerWeight": 0
                                        }
                                },
                                "connectionConfiguration": {
                                        "IsBgpEnabled": true,
                                        "PSK": "psk-vpn-swe",
                                        "IPsecParameters": {
                                                "SADataSizeInKilobytes": 102400000,
                                                "SALifeTimeInSeconds": 3600
                                        }
                                }
                        }
                ]
        }
]
```
Example 11-2: *Original Configuration File Changed to Object-Tree Structure.*

```
crypto ikev2 proposal Azure-Ikev2-Proposal
 encryption aes-cbc-256
 integrity sha1
 group 2
!
crypto ikev2 policy Azure-Ikev2-Policy
 match address local 192.168.100.18
 proposal Azure-Ikev2-Proposal
!
crypto ikev2 keyring cn-hub1-dc1-keyring
 peer 51.12.158.167
  address 51.12.158.167
  pre-shared-key psk-vpn-swe
 !
crypto ikev2 keyring cn-hub1-dc1-keyring-2
 peer 51.12.153.245
  address 51.12.153.245
  pre-shared-key psk-vpn-swe

!
crypto ikev2 profile Azure-Ikev2-Profile
 match address local 192.168.100.18
 match identity remote address 51.12.158.167 255.255.255.255
 authentication remote pre-share
 authentication local pre-share
 keyring local cn-hub1-dc1-keyring
 lifetime 28800
 dpd 10 5 on-demand
!
crypto ikev2 profile Azure-Ikev2-Profile-2
 match address local 192.168.100.18
 match identity remote address 51.12.153.245 255.255.255.255
 authentication remote pre-share
 authentication local pre-share
 keyring local cn-hub1-dc1-keyring-2
 lifetime 28800
 dpd 10 5 on-demand
!
crypto ipsec transform-set Azure-TransformSet esp-aes 256 esp-sha256-hmac
 mode tunnel
!
crypto ipsec profile Azure-IPsecProfile
 set security-association lifetime kilobytes 102400000
 set transform-set Azure-TransformSet
 set ikev2-profile Azure-Ikev2-Profile
!
crypto ipsec profile Azure-IPsecProfile-2
 set security-association lifetime kilobytes 102400000
 set transform-set Azure-TransformSet
 set ikev2-profile Azure-Ikev2-Profile-2
!
interface Loopback11
 description ** eBGP Azure-Hub-swe **
 ip address 192.168.11.1 255.255.255.255
!
```

```
interface Tunnel11
 ip address 169.254.0.1 255.255.255.255
 ip tcp adjust-mss 1350
 tunnel source 192.168.100.18
 tunnel mode ipsec ipv4
 tunnel destination 51.12.158.167
 tunnel protection ipsec profile Azure-IPsecProfile
!
interface Tunnel12
 ip address 169.254.0.2 255.255.255.255
 ip tcp adjust-mss 1350
 tunnel source 192.168.100.18
 tunnel mode ipsec ipv4
 tunnel destination 51.12.153.245
 tunnel protection ipsec profile Azure-IPsecProfile-2
!
router bgp 65771
 bgp log-neighbor-changes
 neighbor 10.10.0.12 remote-as 65515
 neighbor 10.10.0.12 ebgp-multihop 255
 neighbor 10.10.0.12 update-source Loopback11
 neighbor 10.10.0.13 remote-as 65515
 neighbor 10.10.0.13 ebgp-multihop 255
 neighbor 10.10.0.13 update-source Loopback11
 !
 address-family ipv4
  network 10.11.11.0 mask 255.255.255.0
  neighbor 10.10.0.12 activate
  neighbor 10.10.0.13 activate
 exit-address-family
!
ip route 10.10.0.12 255.255.255.255 Tunnel11
ip route 10.10.0.13 255.255.255.255 Tunnel12
```

Example 11-3: *Swe-Branch Configuration.*

Figure 11-27 verifies that the *Connectivity status* is now *Connected*. If you need to edit, delete, or change the connection parameters, click the three dots selector [...] at the end of the row.


Home > NWKT | Resources > vwan-nwkt | Hubs > vhub-swe

vhub-swe | VPN (Site to site)

Virtual HUB

Download VPN Config · Packet Capture · Delete gateway · Reset gateway

Essentials · JSON View

Search this page · Clear all filters

Hub association : **Connected to this hub**

VPN Sites

Check active filters when searching for a VPN site. VPN connectivity status might take a few m

Edit VPN connection to this hub

Delete VPN connection to this hub

Download VPN site configuration file

Create new VPN site · Connect VPN sites · Disconnect VPN sites · Refresh

Page: 1

Site name	Location	Connection Provisio...	Connectivity status
swe-branch	swedencentral	Succeeded	Connected

Figure 11-27: *VPN Site Connection Verification.*

Example 11-4 verifies that BGP Peerings between VGW instances and Swe-Branch are up, and Swe-Branch has received one route from both instances.

```
Swe-Branch#sh ip bgp summ
BGP router identifier 192.168.11.1, local AS number 65771
BGP table version is 5, main routing table version 5
2 network entries using 496 bytes of memory
3 path entries using 408 bytes of memory
2/2 BGP path/bestpath attribute entries using 560 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1488 total bytes of memory
BGP activity 2/0 prefixes, 3/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.10.0.12      4        65515     125     120        5    0    0 01:46:58        1
10.10.0.13      4        65515     101      99        5    0    0 01:26:15        1
```

Example 11-4: *BGP Peering Verification on Swe-Branch.*

Example 11-5 shows that Swe-Branch has received reachability information of 10.10.0.0/16 from both instances.

```
Swe-Branch#sh ip bgp
BGP table version is 5, local router ID is 192.168.11.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *    10.10.0.0/16     10.10.0.13                             0 65515 i
 *>                    10.10.0.12                             0 65515 i
 *>   10.11.11.0/24    0.0.0.0                  0         32768 i
```

Example 11-5: *BGP Routes on Swe-Branch.*

VNet to vHub connection

As the last step, we connect our example VNets to vHub. Because VPN Site routes are installed into the Default Route table (RT), we associate VNet with the same RT.

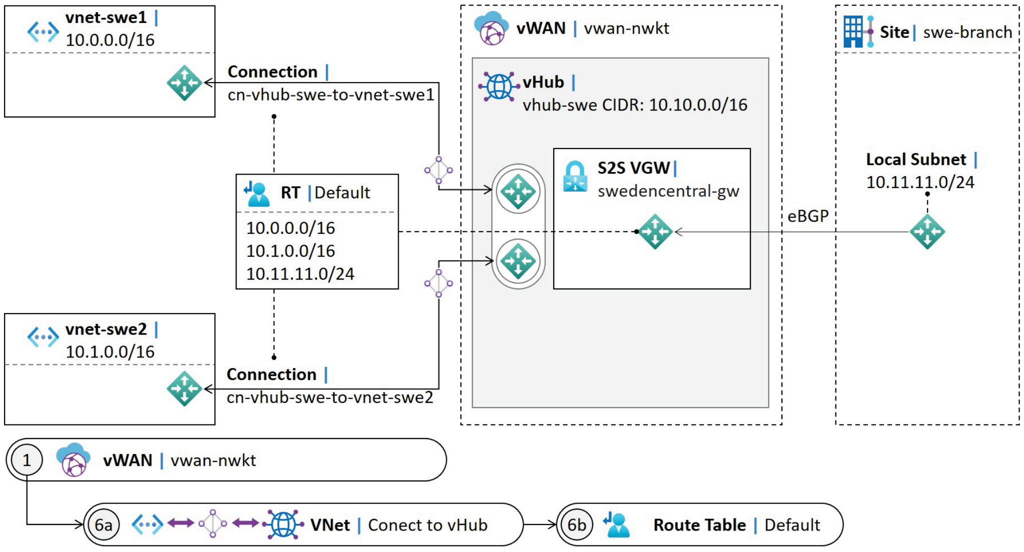
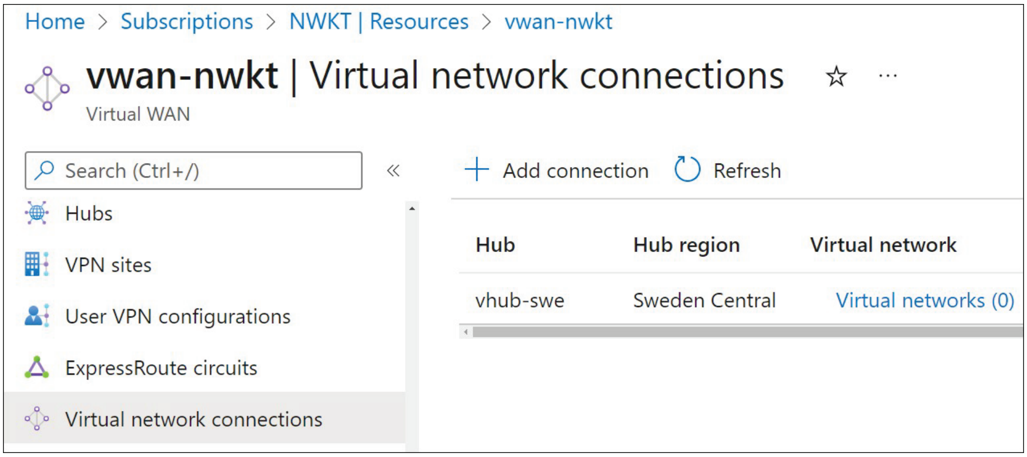


Figure 11-28: *VNet to vHub connection.*

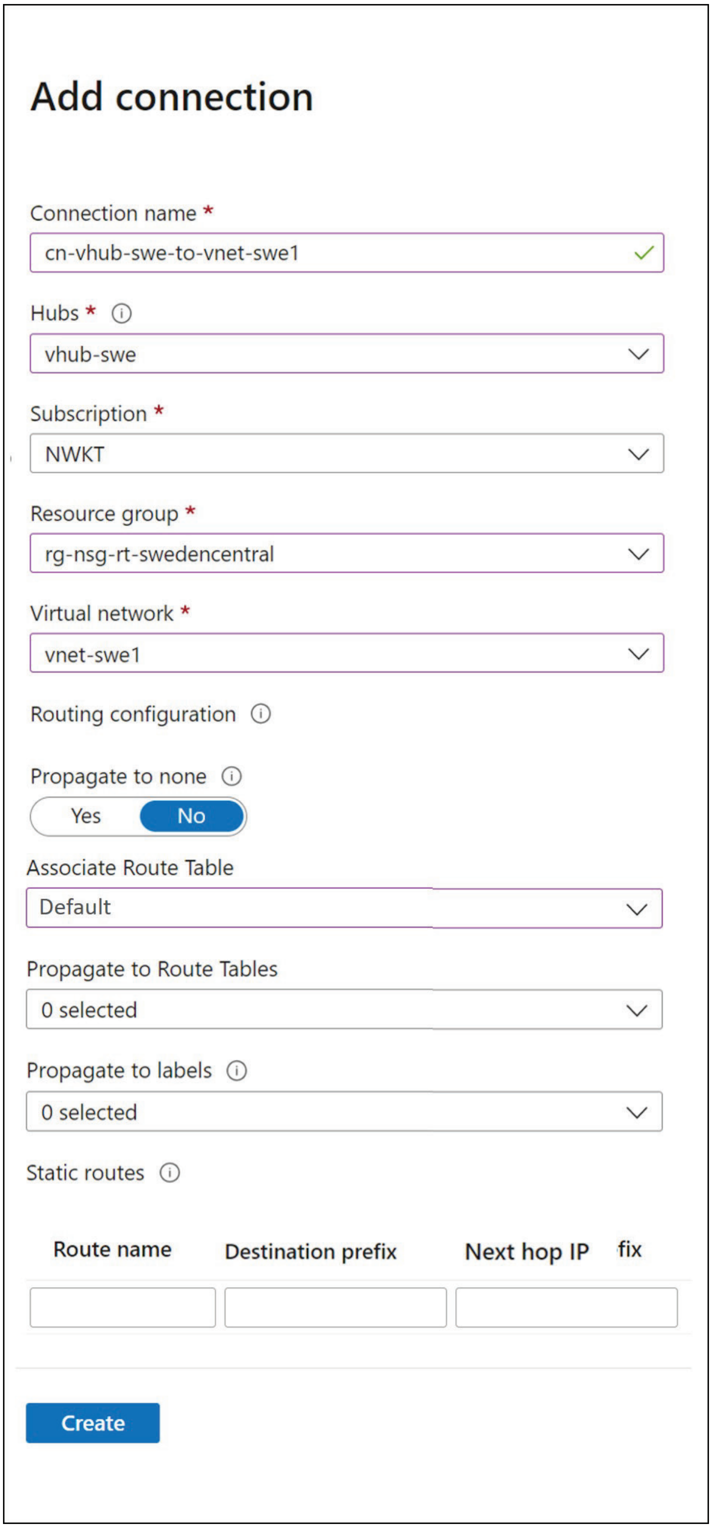
On the vwan-nwkt page, select the *Virtual Network Connections* option under the *Connectivity* section in the left pane. Click the + *Add Connection* button.


Home > Subscriptions > NWKT | Resources > vwan-nwkt
vwan-nwkt | Virtual network connections
Virtual WAN
Search (Ctrl+/)
Hubs
VPN sites
User VPN configurations
ExpressRoute circuits
Virtual network connections
Add connection
Refresh

Hub	Hub region	Virtual network
vhub-swe	Sweden Central	Virtual networks (0)

Figure 11-29: *Deploy VNet to vHub connection – Starting Point.*

In the *Add connection* window (figure 11-30), name the connection, and select your vhub-swe and vnet-swe1. Select also your Subscription and Resource group. Scroll down, and choose the *Default* option from the *Associate Route Table* drop-down menu. Leave all other settings in their defaults. Click the *Create* button to proceed. Create a connection also for vnet-swe2.



Add connection

Connection name *
cn-vhub-swe-to-vnet-swe1

Hubs *
vhub-swe

Subscription *
NWKT

Resource group *
rg-nsg-rt-swedencentral

Virtual network *
vnet-swe1

Routing configuration

Propagate to none
Yes [✓ No]

Associate Route Table
Default

Propagate to Route Tables
0 selected

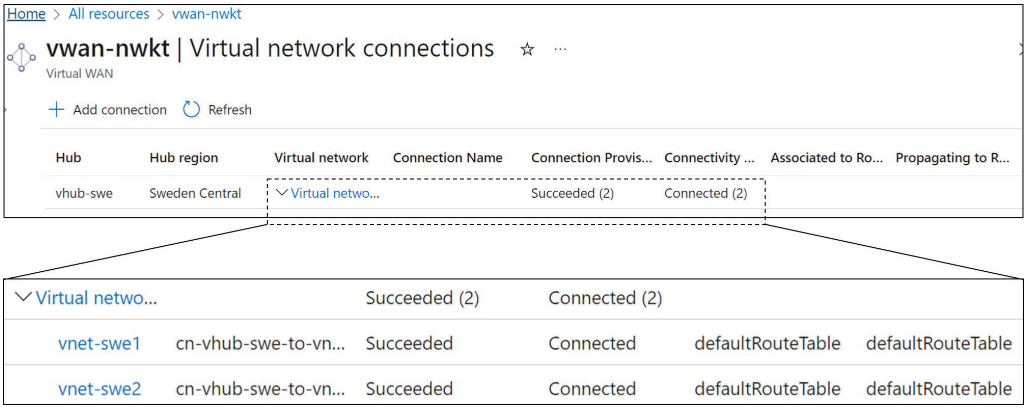
Propagate to labels
0 selected

Static routes

Route name	Destination prefix	Next hop IP

Figure 11-30: *Deploy VNet to vHub connection – Deployment.*

After successful deployment, you should see both VNets in the *vwan-nwkt | Virtual Networks connection* window. You can expand the Virtual Network hyperlink to see connection-specific status information. We can see that both connections are associated with the Default RT.


Home > All resources > vwan-nwkt

vwan-nwkt | Virtual network connections

Virtual WAN

+ Add connection   Refresh

Hub	Hub region	Virtual network	Connection Name	Connection Provis...	Connectivity ...	Associated to Ro...	Propagating to R...
vhub-swe	Sween Central	Virtual netwo...		Succeeded (2)	Connected (2)		

Virtual netwo...		Succeeded (2)	Connected (2)		
vnet-swe1	cn-vhub-swe-to-vn...	Succeeded	Connected	defaultRouteTable	defaultRouteTable
vnet-swe2	cn-vhub-swe-to-vn...	Succeeded	Connected	defaultRouteTable	defaultRouteTable

Figure 11-31: *VNet to vHub connection - Verification.*

The *vwan-nwkt | Insight* page in figure 11-32 gives a view of our vWAN setup.

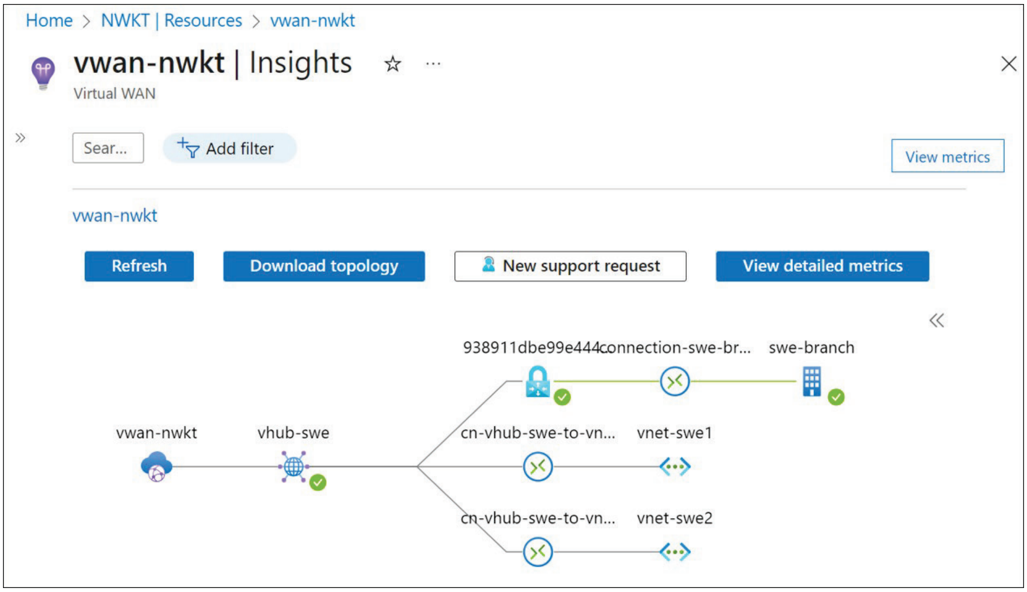


Figure 11-32: *VNet to vHub connection.*

Control Plane verification

Figure 11-33 illustrates a simplified view and IP addressing scheme of our example implementation. We verify the Control Plane operation and check how CIDR ranges 10.0.0.0/16 (vnet-swe1) & 10.1.0.0/16 (vnet-swe2), and VPN Site's local subnet 10.11.11.0/24 are propagated. We first verify effective routes of vNIC associated with VMs, and then we do the same checking from the vHub. Next, we verify what reachability information VGW has exchanged with its BGP peers. As a last Control Plane verification, we check the BGP table and RIB of Swe-Branch.

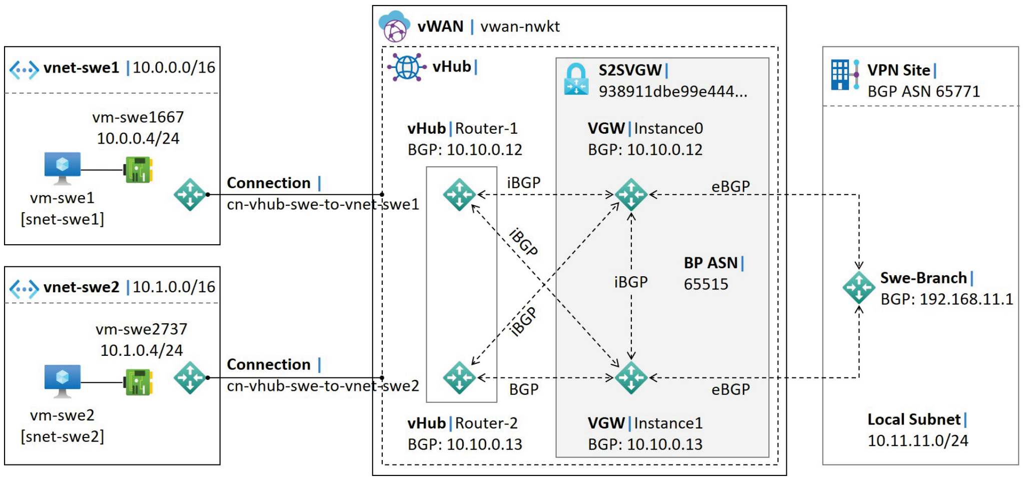
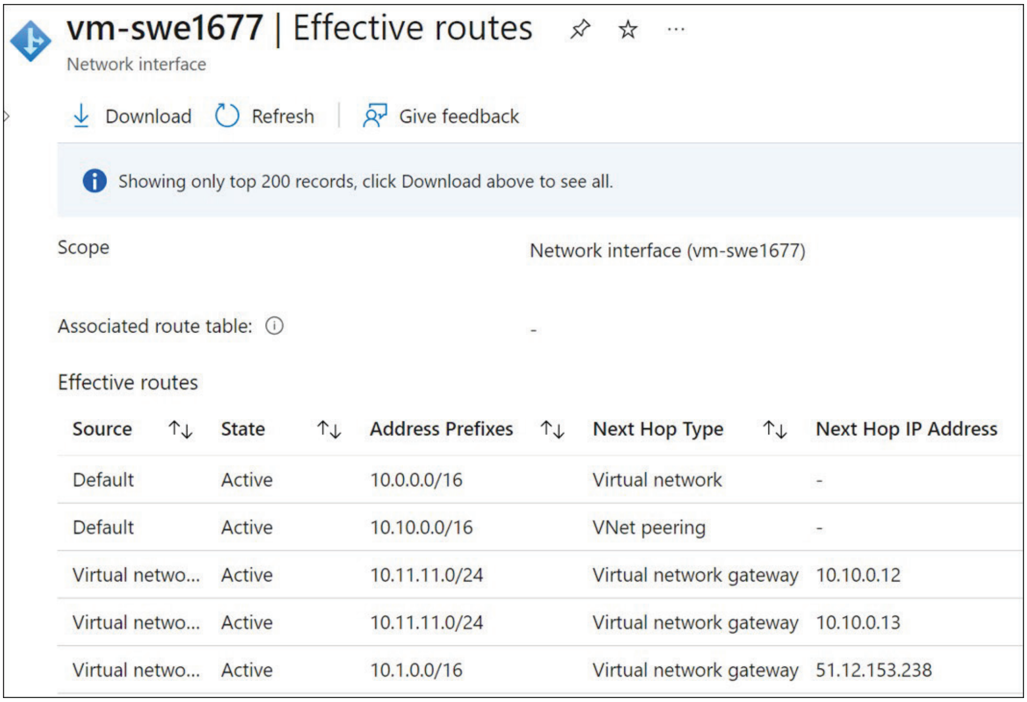


Figure 11-33: *Routing Components.*

VNet Route Table

Figure 11-34 shows the routes learned and used by vNIC vm-swe1667 attached to vn-swe1 in VNet vnet-swe1. The route to vHub CIDR 10.10.0.0/16 is learned over VNet Peering. VNet peering was introduced in chapter 8. The last routing entry describes the CIDR range associated with vnet-swe2. It is learned from the VGW with the next hop public IP 51.12.153.238. This connection is equivalent to VNet-to-VNet VPN explained in chapter 7. The vNIC vm-swe1667 has learned two routes to VPN site local subnet 10.11.11.0/24 from both vHub routers. Figure 11-35 shows the routing information from the vm-swe2737 perspective.



Showing only top 200 records, click Download above to see all.

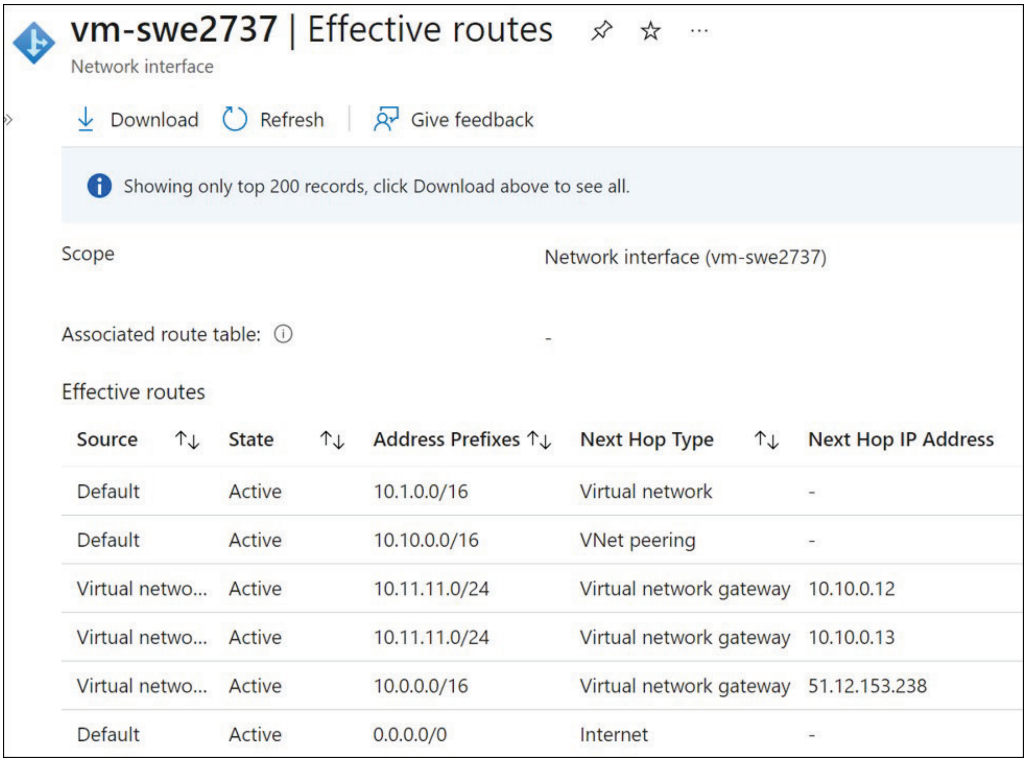
Scope: Network interface (vm-swe1677)

Associated route table: -

Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Hop IP Address
Default	Active	10.0.0.0/16	Virtual network	-
Default	Active	10.10.0.0/16	VNet peering	-
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	10.10.0.12
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	10.10.0.13
Virtual netwo...	Active	10.1.0.0/16	Virtual network gateway	51.12.153.238

Figure 11-34: *Effective Routes: vm-swe1667 on VNet vnet-swe1.*



Showing only top 200 records, click Download above to see all.

Scope: Network interface (vm-swe2737)

Associated route table: -

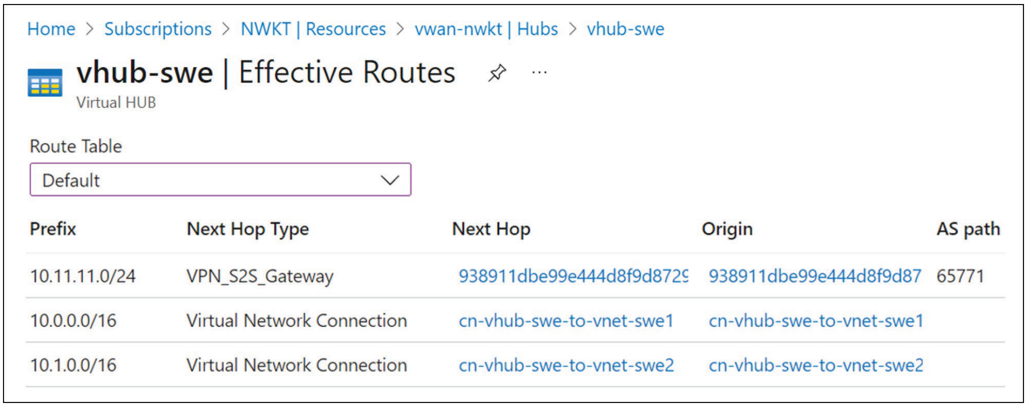
Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Hop IP Address
Default	Active	10.1.0.0/16	Virtual network	-
Default	Active	10.10.0.0/16	VNet peering	-
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	10.10.0.12
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	10.10.0.13
Virtual netwo...	Active	10.0.0.0/16	Virtual network gateway	51.12.153.238
Default	Active	0.0.0.0/0	Internet	-

Figure 11-35: *Effective Routes: vm-swe2337 on VNet vnet-swe2.*

Virtual Hub Route Table

Figure 11-36 shows the effective routes of vHub vhub-swe. It has learned the VPN site swe-branch local subnet 10.11.11.0/24 from the VGW 938911db...swedencentral-gw with AS Path 65771 (VPN Site BGP ASN). Because there are no added ASNs in AS Path list, we can assume that the Hub router and VGW Instances are iBGP peers. VNet CIDR ranges 10.0.0.0/16 and 10.1.0.0/16 are learned from their relative VNet peering connections.

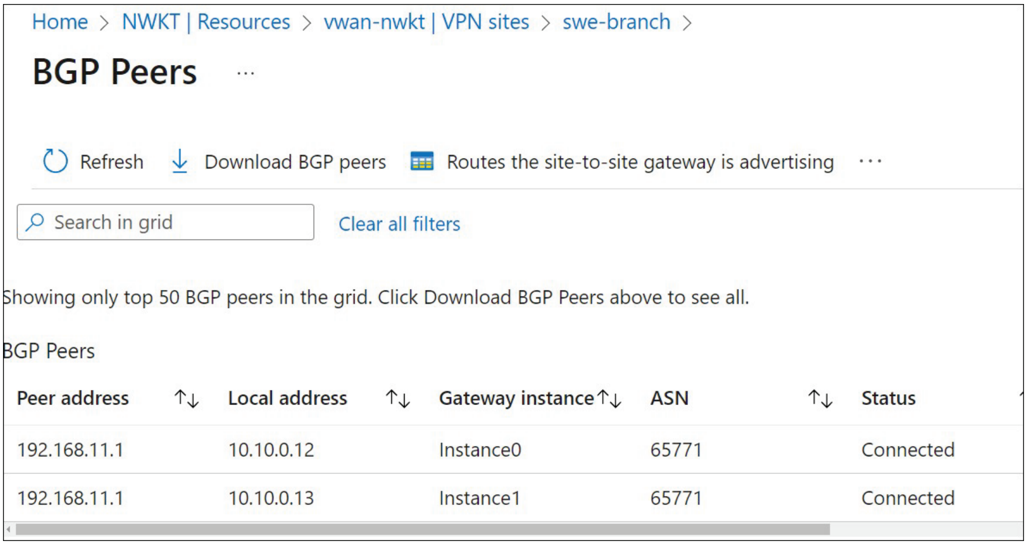

Home > Subscriptions > NWKT | Resources > vwan-nwkt | Hubs > vhub-swe
vhub-swe | Effective Routes
Virtual HUB
Route Table
Default

Prefix	Next Hop Type	Next Hop	Origin	AS path
10.11.11.0/24	VPN_S2S_Gateway	938911dbe99e444d8f9d8729	938911dbe99e444d8f9d87	65771
10.0.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe1	cn-vhub-swe-to-vnet-swe1	
10.1.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe2	cn-vhub-swe-to-vnet-swe2	

Figure 11-36: *Effective Routes: vhub-swe.*

VPN Gateway Routing

Figure 11-37 shows only VGW's external BGP peers. It has two active instances, which both peers with VPN Site edge router Swe-Branch.


Home > NWKT | Resources > vwan-nwkt | VPN sites > swe-branch >
BGP Peers
Refresh
Download BGP peers
Routes the site-to-site gateway is advertising
Search in grid
Clear all filters

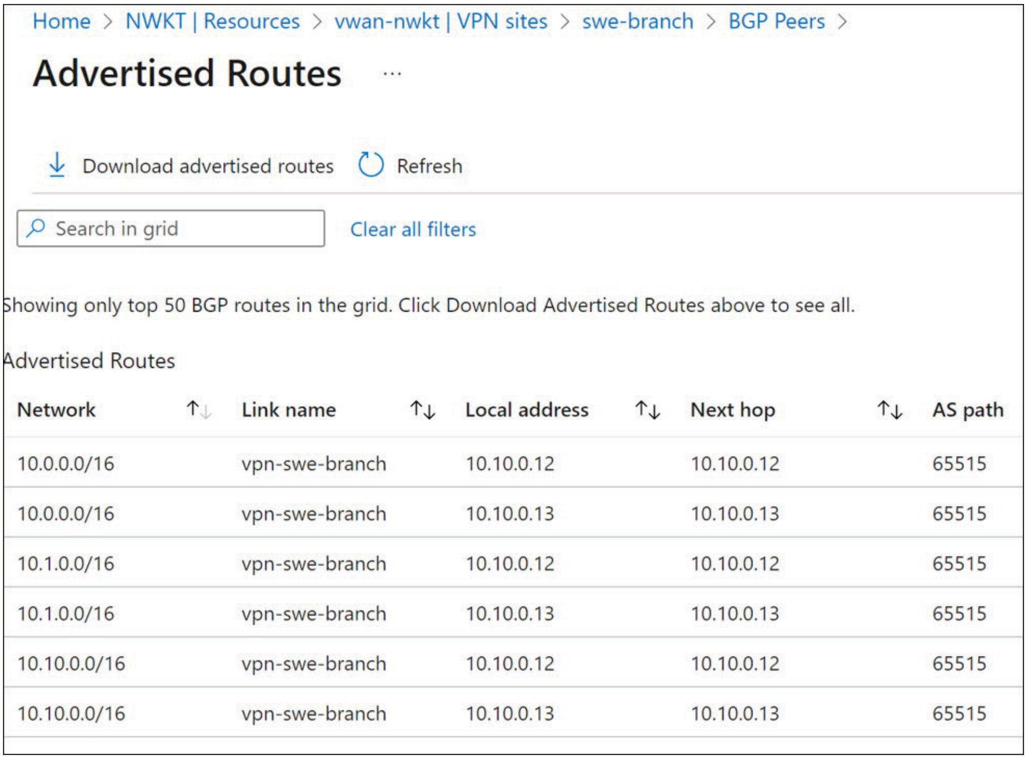
Showing only top 50 BGP peers in the grid. Click Download BGP Peers above to see all.

BGP Peers

Peer address	Local address	Gateway instance	ASN	Status
192.168.11.1	10.10.0.12	Instance0	65771	Connected
192.168.11.1	10.10.0.13	Instance1	65771	Connected

Figure 11-37: *VPN Gateway external BGP Peering.*

Figure 11-38 verifies that VGW instances advertise all of our Azure CIDR ranges to router Swe-Branch on the remote site.


Home > NWKT | Resources > vwan-nwkt | VPN sites > swe-branch > BGP Peers >
Advertised Routes
Download advertised routes
Refresh
Search in grid
Clear all filters

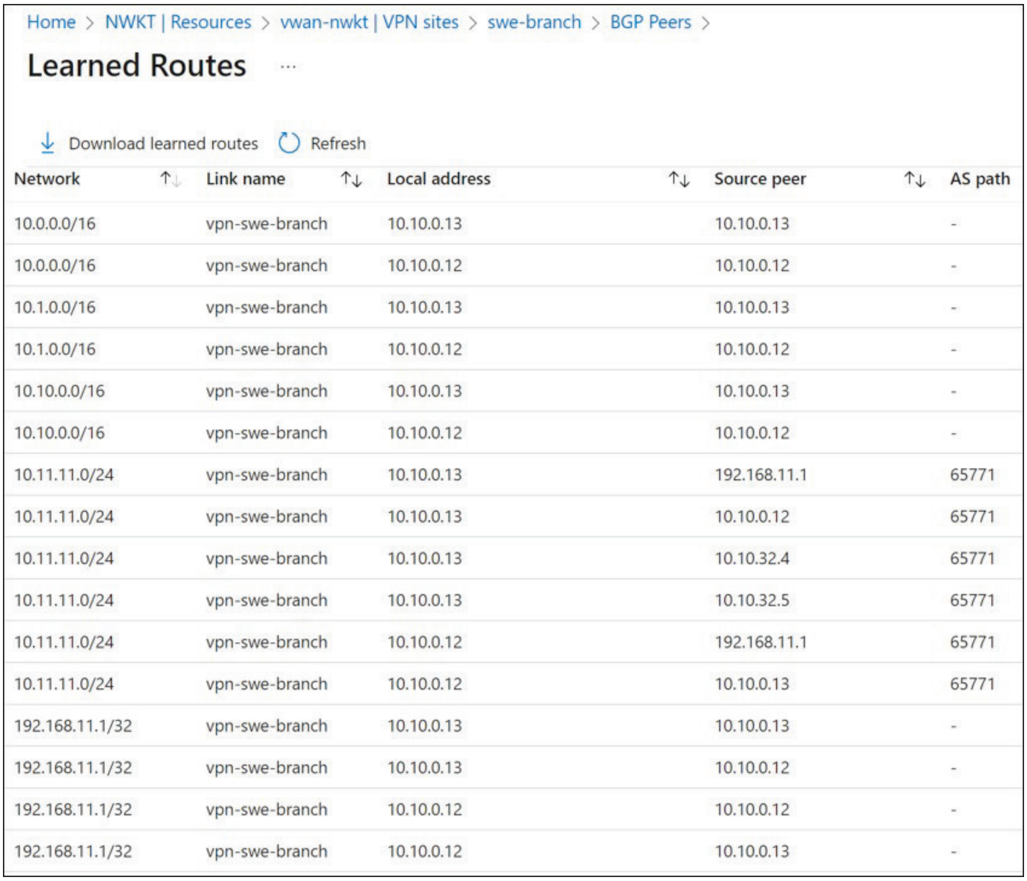
Showing only top 50 BGP routes in the grid. Click Download Advertised Routes above to see all.

Advertised Routes

Network	Link name	Local address	Next hop	AS path
10.0.0.0/16	vpn-swe-branch	10.10.0.12	10.10.0.12	65515
10.0.0.0/16	vpn-swe-branch	10.10.0.13	10.10.0.13	65515
10.1.0.0/16	vpn-swe-branch	10.10.0.12	10.10.0.12	65515
10.1.0.0/16	vpn-swe-branch	10.10.0.13	10.10.0.13	65515
10.10.0.0/16	vpn-swe-branch	10.10.0.12	10.10.0.12	65515
10.10.0.0/16	vpn-swe-branch	10.10.0.13	10.10.0.13	65515

Figure 11-38: *VPN Gateway: Advertised Routes.*

Figure 11-39 shows routes that VGW has learned. The next hop for all CIDR ranges on the Azure side is shown as local routes, where the *Local address* and *Source peer* are instance-specific IP addresses. The Swe-Branch's BGP peering IP address 192.168.11.1 is listed four times at the end of the list. The IP address 192.18.11.1 is filled into the *Private address space* field during the VPN Site configuration (see figure 11-18). Besides both instances having installed the route as a Local route, they have also learned it from each other. The VPN Site local subnet 10.11.11.0/16 is listed six times. First, Instance0 has learned it from 192.168.11.1 (Swe-Branch). Besides, it has received the reachability information from both Hub routers (10.10.32.4 & 10.10.32.5) and instance1 (10.10.0.13). Instance1, in turn, has learned routes only from 192.168.11.1 (Swe-Branch) and instance0 (10.10.0.12). The reason might be that Hub routers have selected the route received from instance1 (10.10.0.13) as the best route and haven't advertised it back to instance1 (default BGP behavior). I said "might" because the Azure documentation doesn't describe the internal routing in detail. Instance0 naturally advertises the route to Instance1 because its best route is via Swe-Branch.



Home > NWKT | Resources > vwan-nwkt | VPN sites > swe-branch > BGP Peers >

Learned Routes ...

Download learned routes   Refresh

Network	Link name	Local address	Source peer	AS path
10.0.0.0/16	vpn-swe-branch	10.10.0.13	10.10.0.13	-
10.0.0.0/16	vpn-swe-branch	10.10.0.12	10.10.0.12	-
10.1.0.0/16	vpn-swe-branch	10.10.0.13	10.10.0.13	-
10.1.0.0/16	vpn-swe-branch	10.10.0.12	10.10.0.12	-
10.10.0.0/16	vpn-swe-branch	10.10.0.13	10.10.0.13	-
10.10.0.0/16	vpn-swe-branch	10.10.0.12	10.10.0.12	-
10.11.11.0/24	vpn-swe-branch	10.10.0.13	192.168.11.1	65771
10.11.11.0/24	vpn-swe-branch	10.10.0.13	10.10.0.12	65771
10.11.11.0/24	vpn-swe-branch	10.10.0.13	10.10.32.4	65771
10.11.11.0/24	vpn-swe-branch	10.10.0.13	10.10.32.5	65771
10.11.11.0/24	vpn-swe-branch	10.10.0.12	192.168.11.1	65771
10.11.11.0/24	vpn-swe-branch	10.10.0.12	10.10.0.13	65771
192.168.11.1/32	vpn-swe-branch	10.10.0.13	10.10.0.13	-
192.168.11.1/32	vpn-swe-branch	10.10.0.13	10.10.0.12	-
192.168.11.1/32	vpn-swe-branch	10.10.0.12	10.10.0.12	-
192.168.11.1/32	vpn-swe-branch	10.10.0.12	10.10.0.13	-

Figure 11-39: *VPN Gateway: Learned Routes.*

Branch Control Plane

Example 11-6 verifies that the router Swe-Branch has received CIDR ranges 10.0.0.0/16 (vnet-swe1), 10.1.0.0/16 (vnet-swe2), and 10.10.0.0/16 (vhub-swe) from both BGP peers. Swe-Branch has selected routes received from 10.10.0.12 (VGW instance1) as the best route due to the lowest BGP peer Router-Id (RID). The BGP path selection process is explained on page 189 (chapter 10).

```
Swe-Branch#sh ip bgp
BGP table version is 5, local router ID is 192.168.11.1
<snipped for brevity>
      Network          Next Hop            Metric LocPrf Weight Path
 *     10.0.0.0/16      10.10.0.13                             0 65515 i
 *>                     10.10.0.12                             0 65515 i
 *     10.1.0.0/16      10.10.0.13                             0 65515 i
 *>                     10.10.0.12                             0 65515 i
 *>    10.10.0.0/16     10.10.0.12                             0 65515 i
 *                      10.10.0.13                             0 65515 i
 *>    10.11.11.0/24    0.0.0.0                  0         32768 i
```

Example 11-6: *Swe-Branch: BGP Table Entry about CIDR range 10.0.0.0/16.*

Examples 11-7 and 11-8 show detailed information about BGP entries concerning CIDR ranges 10.0.0.0/16 and 10.1.0.0/16. The BGP RIDs are highlighted.

```
Swe-Branch#sh ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/16, version 4
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  65515
    10.10.0.13 from 10.10.0.13 (10.10.0.13)
      Origin IGP, localpref 100, valid, external
      rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  65515
    10.10.0.12 from 10.10.0.12 (10.10.0.12)
      Origin IGP, localpref 100, valid, external, best
      rx pathid: 0, tx pathid: 0x0
```

Example 11-7: *Swe-Branch: BGP Table Entry about CIDR range 10.0.0.0/16*

```
Swe-Branch#sh ip bgp 10.1.0.0
BGP routing table entry for 10.1.0.0/16, version 5
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  65515
    10.10.0.13 from 10.10.0.13 (10.10.0.13)
      Origin IGP, localpref 100, valid, external
      rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  65515
    10.10.0.12 from 10.10.0.12 (10.10.0.12)
      Origin IGP, localpref 100, valid, external, best
      rx pathid: 0, tx pathid: 0x0
```

Example 11-8: *Swe-Branch: BGP Table Entry about CIDR range 10.1.0.0/16*

Example 11-9 shows the RIB of Swe-Branch. It verifies that Swe-Branch has installed routes learned from the BGW instance1 into the RIB.

```
Swe-Branch#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is 192.168.100.1 to network 0.0.0.0

S*    0.0.0.0/0 [254/0] via 192.168.100.1
      10.0.0.0/8 is variably subnetted, 7 subnets, 3 masks
B        10.0.0.0/16 [20/0] via 10.10.0.12, 01:35:35
B        10.1.0.0/16 [20/0] via 10.10.0.12, 00:01:32
B        10.10.0.0/16 [20/0] via 10.10.0.12, 02:06:44
S        10.10.0.12/32 is directly connected, Tunnel11
S        10.10.0.13/32 is directly connected, Tunnel12
C        10.11.11.0/24 is directly connected, Loopback1111
L        10.11.11.1/32 is directly connected, Loopback1111
      169.254.0.0/32 is subnetted, 2 subnets
C        169.254.0.1 is directly connected, Tunnel11
C        169.254.0.2 is directly connected, Tunnel12
      192.168.11.0/32 is subnetted, 1 subnets
C        192.168.11.1 is directly connected, Loopback11
      192.168.100.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.100.0/24 is directly connected, GigabitEthernet1
L        192.168.100.18/32 is directly connected, GigabitEthernet1
```

Example 11-9: *Swe-Branch: Routing Table Entry.*

Data Plane verification

The following four examples prove that we have a working Inter-VNet connection between vnet-swe1 and vnet-swe2 and a VNet-to-VPN Site connection.

```
azureuser@vm-swe1:~$ ping 10.1.0.4
PING 10.1.0.4 (10.1.0.4) 56(84) bytes of data.
64 bytes from 10.1.0.4: icmp_seq=1 ttl=63 time=4.33 ms
64 bytes from 10.1.0.4: icmp_seq=2 ttl=63 time=3.05 ms
64 bytes from 10.1.0.4: icmp_seq=3 ttl=63 time=2.12 ms
64 bytes from 10.1.0.4: icmp_seq=4 ttl=63 time=3.44 ms
64 bytes from 10.1.0.4: icmp_seq=5 ttl=63 time=2.25 ms
^C
--- 10.1.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 2.118/3.039/4.328/0.811 ms
```

Example 11-10: *Ping from vm-swe1 (10.0.0.4) to vm-swe2 (10.1.0.4).*

```
azureuser@vm-swe1:~$ ping 10.11.11.1
PING 10.11.11.1 (10.11.11.1) 56(84) bytes of data.
64 bytes from 10.11.11.1: icmp_seq=1 ttl=255 time=13.6 ms
64 bytes from 10.11.11.1: icmp_seq=2 ttl=255 time=12.3 ms
64 bytes from 10.11.11.1: icmp_seq=3 ttl=255 time=12.6 ms
64 bytes from 10.11.11.1: icmp_seq=4 ttl=255 time=12.7 ms
64 bytes from 10.11.11.1: icmp_seq=5 ttl=255 time=13.2 ms
^C
--- 10.11.11.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 12.273/12.870/13.563/0.448 ms
azureuser@vm-swe1:~$
```

Example 11-11: *Ping from vm-swe1 (10.0.0.4) to Swe-Branch Local Subnet (10.11.11.1).*

```
Swe-Branch#ping 10.0.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/12/13 ms
```

Example 11-12: *Ping from Swe-Branch (10.11.11.1) to vm-swe1 (10.0.0.4).*

```
Swe-Branch#ping 10.1.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/12/13 ms
```

Example 11-13: *Ping from Swe-Branch (10.11.11.1) to vm-swe2 (10.1.0.4).*

Pricing

Figure 11-40 illustrates the cost component related to vWAN. There are links to Azure pricing pages in the References section. In our example environment hourly charged costs units are: 1) Standard vHub with two Routing Infrastructure Units (0,248€/hr + 0,198€/hr = 0,0446€/hr), 2) Site-to-Site VPN GW with one Scale Unit (0,357€/hr), and 3) one VPN Site-to-Site Connection Unit (0,050€/hr). This gives us one of our fees of 0,853€/hr = 614,16€/Month. Besides, data transfer is charged based on processed gigabytes.

Inter-VNet data transfer includes three charging point: 1) VNet Egress fee (0,010€/GB), 2) vHub Data Processing (0,010€/GB) and VNet Ingress fee (0,010€/GB). Ten Giga Bytes of Inter-VNet data costs 0,03€.

VNet to VPN-Site data transfer fees includes 1) VNet Egress fee (0,010€), and Internet Egress from Standard Outbound Zone1 (0,0865€/GB). Ten Giga Bytes of VNet to VPN Site costs 0,0791€/GB. Note that the Internet Egress data is monitored on monthly bases.

1) The first 100 GB are free		
2) the next 10TB = 0,0865€/GB	=> 10 000GB x 0,0865€	= 865€
3) the next 40 TB = 0,0821€/GB	=> 40 000GB x 0,0821€	= 3284€
4) the next 100TB = 0, 0692€/GB	=> 100 000GB x 0,0692€	= 6920€
5) the next 350TB = 0,0495€/GB	=> 350 000GB x 0, 0495€	= 13 860€

The figures are taken from the Azure web page at August 8, 2022 (Germany West Central). Be aware of pricing changes).

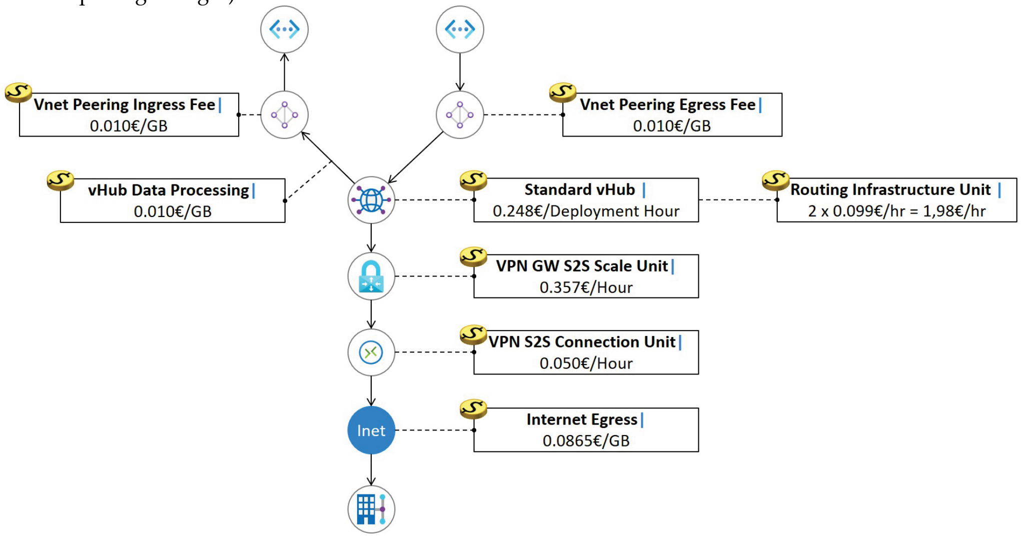


Figure 11-40: *Pricing Components of vWAN.*

References

[1] What is Azure Virtual WAN?
https://docs.microsoft.com/en-us/azure/virtual-wan/virtual-wan-about

[2] About virtual hub settings
https://docs.microsoft.com/en-us/azure/virtual-wan/hub-settings

[3] About Virtual WAN gateway settings
https://docs.microsoft.com/en-us/azure/virtual-wan/gateway-settings

[4] Azure path selection across multiple ISP links
https://docs.microsoft.com/en-us/azure/virtual-wan/path-selection-multiple-links

[5] Virtual hub routing preference (Preview)
https://docs.microsoft.com/en-us/azure/virtual-wan/about-virtual-hub-routing-preference

[6] Scenario: BGP peering with a virtual hub
https://docs.microsoft.com/en-us/azure/virtual-wan/scenario-bgp-peering-hub

[7] Virtual WAN pricing
https://azure.microsoft.com/en-us/pricing/details/virtual-wan/

[8] About Virtual WAN pricing
https://docs.microsoft.com/en-us/azure/virtual-wan/pricing-concepts

[9] Bandwidth pricing
https://azure.microsoft.com/en-us/pricing/details/bandwidth/

[10] Virtual Network pricing
https://azure.microsoft.com/en-us/pricing/details/virtual-network/

[11] How is Virtual WAN different from an Azure virtual network gateway?
https://docs.microsoft.com/en-us/azure/virtual-wan/virtual-wan-faq#how-is-virtual-wan-different-from-an-azure-virtual-network-gateway

Chapter 11 - Appendix A: Swe-Branch Configuration

```
Swe-Branch#sh run
!
enable password cisco
!
crypto ikev2 proposal Azure-Ikev2-Proposal
 encryption aes-cbc-256
 integrity sha1
 group 2
!
crypto ikev2 policy Azure-Ikev2-Policy
 match address local 192.168.100.18
 proposal Azure-Ikev2-Proposal
!
crypto ikev2 keyring cn-hub1-dc1-keyring
 peer 51.12.158.167
  address 51.12.158.167
  pre-shared-key psk-vpn-swe
 !
crypto ikev2 keyring cn-hub1-dc1-keyring-2
 peer 51.12.153.245
  address 51.12.153.245
  pre-shared-key psk-vpn-swe

!
crypto ikev2 profile Azure-Ikev2-Profile
 match address local 192.168.100.18
 match identity remote address 51.12.158.167 255.255.255.255
 authentication remote pre-share
 authentication local pre-share
 keyring local cn-hub1-dc1-keyring
 lifetime 28800
 dpd 10 5 on-demand
!
crypto ikev2 profile Azure-Ikev2-Profile-2
 match address local 192.168.100.18
 match identity remote address 51.12.153.245 255.255.255.255
 authentication remote pre-share
 authentication local pre-share
 keyring local cn-hub1-dc1-keyring-2
 lifetime 28800
 dpd 10 5 on-demand
!
crypto ipsec transform-set Azure-TransformSet esp-aes 256 esp-sha256-hmac
 mode tunnel
!
crypto ipsec profile Azure-IPsecProfile
 set security-association lifetime kilobytes 102400000
 set transform-set Azure-TransformSet
 set ikev2-profile Azure-Ikev2-Profile
!
crypto ipsec profile Azure-IPsecProfile-2
 set security-association lifetime kilobytes 102400000
 set transform-set Azure-TransformSet
 set ikev2-profile Azure-Ikev2-Profile-2
!
interface Loopback11
 description ** eBGP Azure-Hub-swe **
 ip address 192.168.11.1 255.255.255.255
!
```

```
interface Tunnel11
 ip address 169.254.0.1 255.255.255.255
 ip tcp adjust-mss 1350
 tunnel source 192.168.100.18
 tunnel mode ipsec ipv4
 tunnel destination 51.12.158.167
 tunnel protection ipsec profile Azure-IPsecProfile
!
interface Tunnel12
 ip address 169.254.0.2 255.255.255.255
 ip tcp adjust-mss 1350
 tunnel source 192.168.100.18
 tunnel mode ipsec ipv4
 tunnel destination 51.12.153.245
 tunnel protection ipsec profile Azure-IPsecProfile-2
!
router bgp 65771
 bgp log-neighbor-changes
 neighbor 10.10.0.12 remote-as 65515
 neighbor 10.10.0.12 ebgp-multihop 255
 neighbor 10.10.0.12 update-source Loopback11
 neighbor 10.10.0.13 remote-as 65515
 neighbor 10.10.0.13 ebgp-multihop 255
 neighbor 10.10.0.13 update-source Loopback11
 !
 address-family ipv4
  network 10.11.11.0 mask 255.255.255.0
  neighbor 10.10.0.12 activate
  neighbor 10.10.0.13 activate
 exit-address-family
!
ip route 10.10.0.12 255.255.255.255 Tunnel11
ip route 10.10.0.13 255.255.255.255 Tunnel12
```

Chapter 12: Virtual WAN Part 2 –VNet Segmentation

Introduction

VNets and VPN/ExpressRoute connections are associated with vHub's Default Route Table, which allows both VNet-to-VNet and VNet-to-Remote Site IP connectivity. This chapter explains how we can isolate vnet-swe3 from vnet-swe1 and vnet-swe2 using VNet-specific vHub Route Tables (RT), still allowing VNet-to-VPN Site connection. As a first step, we create a Route Table *rt-swe12* to which we associate VNets vnet-swe1 and vnet-swe2. Next, we deploy a Route Table *rt-swe3* for vnet-swe3. Then we propagate routes from these RTs to Default RT but not from rt-swe12 to rt-swe3 and vice versa. Our VPN Gateway is associated with the Default RT, and the route to remote site subnet 10.11.11.0/24 is installed into the Default RT. To achieve bi-directional IP connectivity, we also propagate routes from the Default RT to rt-swe-12 and rt-swe3. As the last step, we verify both Control Plane operation and Data Plane connections.

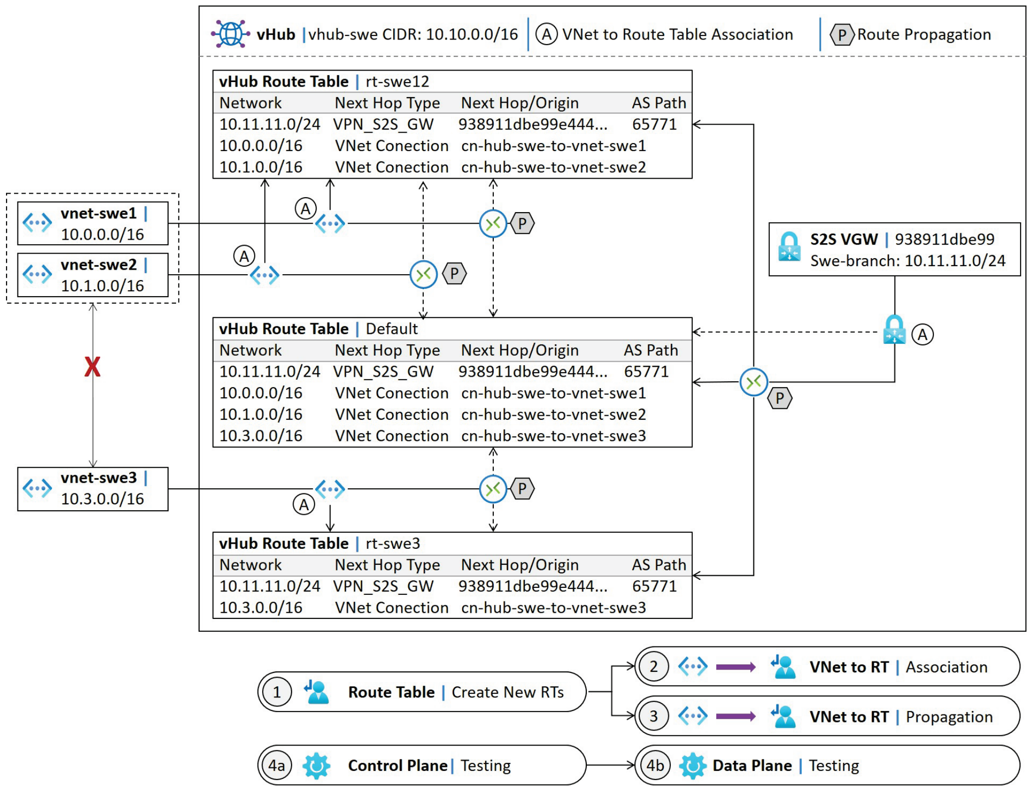
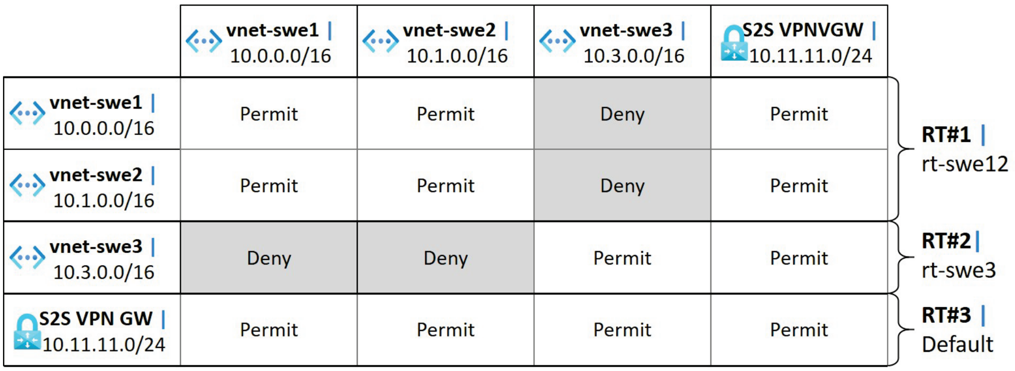

vHub | vhub-swe CIDR: 10.10.0.0/16
A VNet to Route Table Association
P Route Propagation
vHub Route Table | rt-swe12
Network | Next Hop Type | Next Hop/Origin | AS Path
10.11.11.0/24 | VPN_S2S_GW | 938911dbe99e444... | 65771
10.0.0.0/16 | VNet Conection | cn-hub-swe-to-vnet-swe1
10.1.0.0/16 | VNet Conection | cn-hub-swe-to-vnet-swe2
vnet-swe1 | 10.0.0.0/16
vnet-swe2 | 10.1.0.0/16
S2S VGW | 938911dbe99
Swe-branch: 10.11.11.0/24
vHub Route Table | Default
10.11.11.0/24 | VPN_S2S_GW | 938911dbe99e444... | 65771
10.0.0.0/16 | VNet Conection | cn-hub-swe-to-vnet-swe1
10.1.0.0/16 | VNet Conection | cn-hub-swe-to-vnet-swe2
10.3.0.0/16 | VNet Conection | cn-hub-swe-to-vnet-swe3
X
vnet-swe3 | 10.3.0.0/16
vHub Route Table | rt-swe3
10.11.11.0/24 | VPN_S2S_GW | 938911dbe99e444... | 65771
10.3.0.0/16 | VNet Conection | cn-hub-swe-to-vnet-swe3
1 Route Table | Create New RTs
2 VNet to RT | Association
3 VNet to RT | Propagation
4a Control Plane | Testing
4b Data Plane | Testing

Figure 12-1: *Virtual Network Segmentation.*

Figure 12-2 illustrates our traffic matrix. We can associate connection resources having identical connection requirements with one RT. The connection policy for vnet-swe1 and vnet-2 is the same, and we permit traffic between them and traffic to/from the branch site but not with isolated vnet-swe3. All three VNets are allowed to communicate with the on-prem site.

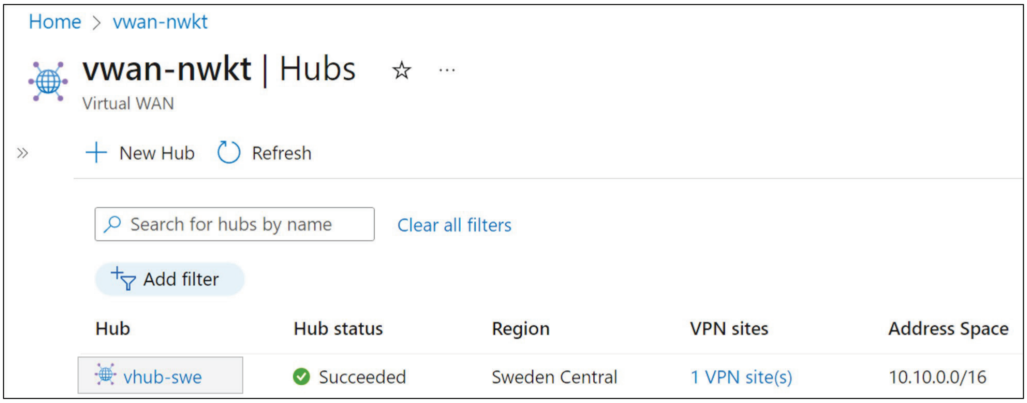


	vnet-swe1 | 10.0.0.0/16	vnet-swe2 | 10.1.0.0/16	vnet-swe3 | 10.3.0.0/16	S2S VPNVGW | 10.11.11.0/24	
vnet-swe1 | 10.0.0.0/16	Permit	Permit	Deny	Permit	RT#1 | rt-swe12
vnet-swe2 | 10.1.0.0/16	Permit	Permit	Deny	Permit	
vnet-swe3 | 10.3.0.0/16	Deny	Deny	Permit	Permit	RT#2 | rt-swe3
S2S VPN GW | 10.11.11.0/24	Permit	Permit	Permit	Permit	RT#3 | Default

Figure 12-2: *VNet Segmentation: Traffic Matrix.*

Default Route Table

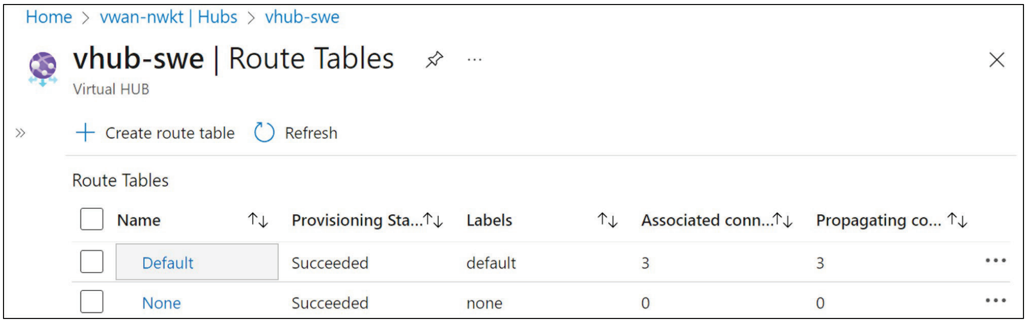
At this phase, we haven't deployed VNet-specific RTs, and all connection recourses are associated with the Default RT. We can manage vHub RTs by first navigating to the vwan-nwkt page and selecting the *Hub*s option. Then, click the vhub-swe hyperlink under the Hub section (figure 12-3).



Hub	Hub status	Region	VPN sites	Address Space
vhub-swe	Succeeded	Sweden Central	1 VPN site(s)	10.10.0.0/16

Figure 12-3: *vHub's the Default Route Table.*

Select the Default RT to view its Association and Propagation settings.


Home > vwan-nwkt | Hubs > vhub-swe

vhub-swe | Route Tables

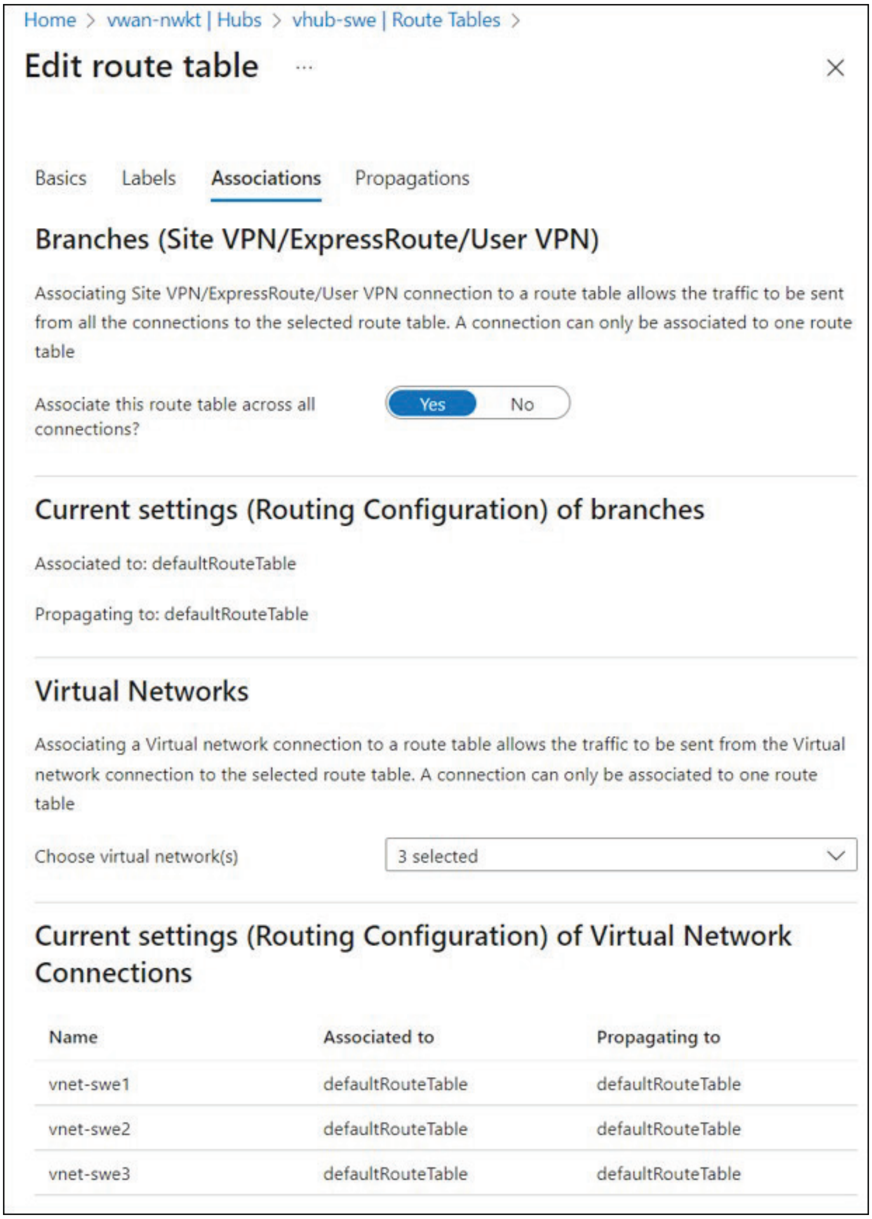
Virtual HUB

Route Tables

Name	Provisioning Sta...	Labels	Associated conn...	Propagating co...
Default	Succeeded	default	3	3
None	Succeeded	none	0	0

Figure 12-4: *vHub's the Default Route Table.*

Figure 11-5 shows the Association tab of the Default Route Table. There are four sections in it. The selector button in the *Branches (Site VPN/ExpressRoute/User VPN)* section is set to Yes. It associates all branch connections with this RT. The *Current settings (Routing Configuration) of branches* shows that branch routes are associated with and propagated to defaultRouteTable. The third section, *Virtual Networks,* has a drop-down menu where we can select VNets that we want to associate with the defaultRouteTable. The "*Current settings (Routing Configuration) of Virtual Network Connections"* section verifies that vnet-swe1, swe-swe2, and swe-vnet3 are all associated with the defaultRouteTable, where all routes learned are propagated.



Home > vwan-nwkt | Hubs > vhub-swe | Route Tables >

Edit route table ...

Basics Labels **Associations** Propagations

Branches (Site VPN/ExpressRoute/User VPN)

Associating Site VPN/ExpressRoute/User VPN connection to a route table allows the traffic to be sent from all the connections to the selected route table. A connection can only be associated to one route table

Associate this route table across all connections? [✓ Yes] No

Current settings (Routing Configuration) of branches

Associated to: defaultRouteTable

Propagating to: defaultRouteTable

Virtual Networks

Associating a Virtual network connection to a route table allows the traffic to be sent from the Virtual network connection to the selected route table. A connection can only be associated to one route table

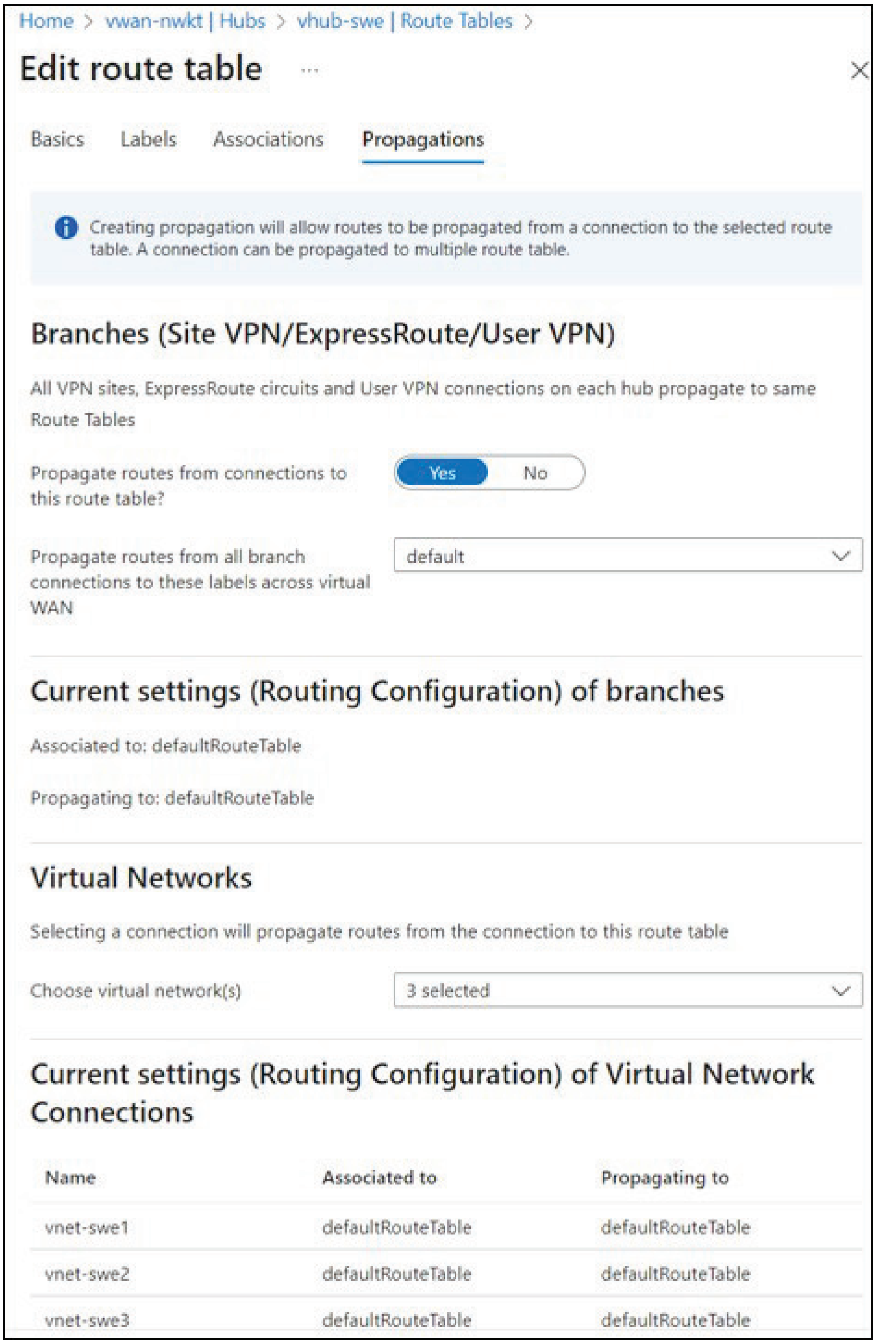
Choose virtual network(s): 3 selected

Current settings (Routing Configuration) of Virtual Network Connections

Name	Associated to	Propagating to
vnet-swe1	defaultRouteTable	defaultRouteTable
vnet-swe2	defaultRouteTable	defaultRouteTable
vnet-swe3	defaultRouteTable	defaultRouteTable

Figure 12-5: *Default Route Table Associations.*

Figure 11-6 shows the *Propagation* tab of the Default Route Table. The *Current settings (Routing Configuration) of branches* and *Current settings (Routing Configuration) of Virtual Network Connections* sections show the same association/propagation information we saw on the Association page. The selector related to route propagation from *Branches (Site VPN/ExpressRoute/User VPN)* connection is set to yes. By doing this, we propagate branch routes learned via Virtual GW/ExpressRoute into this RT. We can use labels for propagating routes across vHubs. At this phase, we only have the default label. By selecting all of our three VNets from the *Choose Virtual Network(s)* drop-down menu, we propagate VNet-specific routes to the Default Route Table.


Home > vwan-nwkt | Hubs > vhub-swe | Route Tables >

Edit route table

Creating propagation will allow routes to be propagated from a connection to the selected route table. A connection can be propagated to multiple route table.

Branches (Site VPN/ExpressRoute/User VPN)

All VPN sites, ExpressRoute circuits and User VPN connections on each hub propagate to same Route Tables

Propagate routes from connections to this route table? [✓ Yes] No

Propagate routes from all branch connections to these labels across virtual WAN: default

Current settings (Routing Configuration) of branches

Associated to: defaultRouteTable

Propagating to: defaultRouteTable

Virtual Networks

Selecting a connection will propagate routes from the connection to this route table

Choose virtual network(s): 3 selected

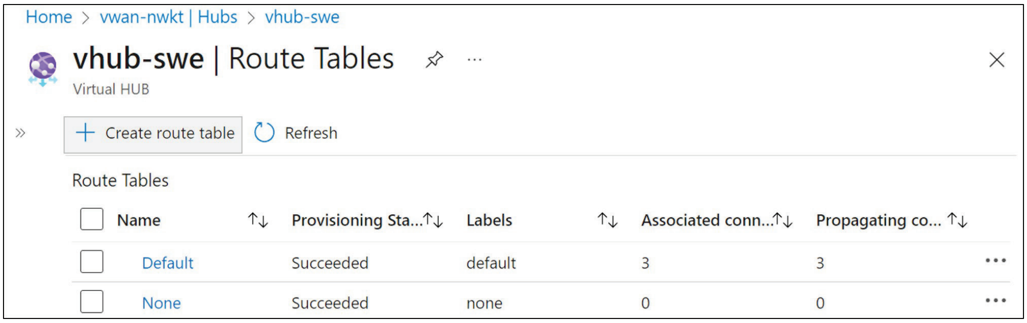
Current settings (Routing Configuration) of Virtual Network Connections

Name	Associated to	Propagating to
vnet-swe1	defaultRouteTable	defaultRouteTable
vnet-swe2	defaultRouteTable	defaultRouteTable
vnet-swe3	defaultRouteTable	defaultRouteTable

Figure 12-6: *Default Route Table Propagation.*

Create New Route Table

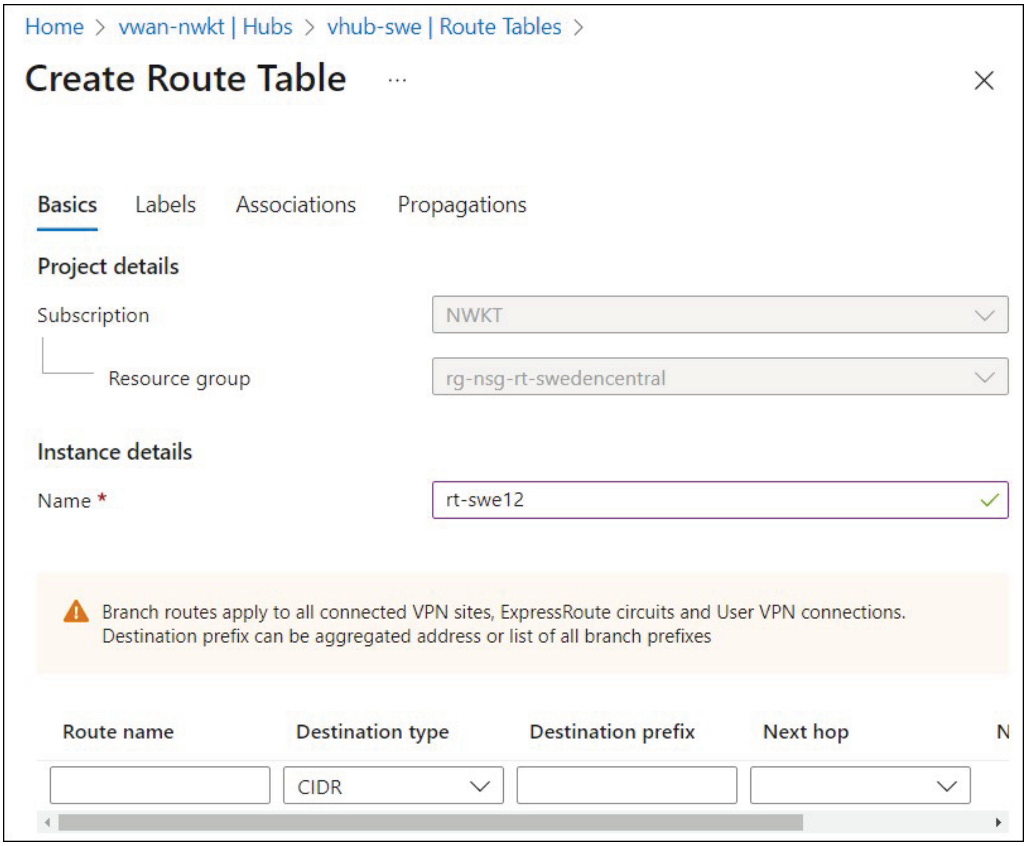
First, select the + *Create route table* from the vhub-swe *Route Tables* page.



Name	Provisioning Sta...	Labels	Associated conn...	Propagating co...
Default	Succeeded	default	3	3
None	Succeeded	none	0	0

Figure 12-7: *Create New Route Table*

Fill in the Basic information and name the Route table. We associate vnet-swe1 and vnet-swe2 with this RT, which is why the name rt-swe12. To proceed, select the Associations tab.



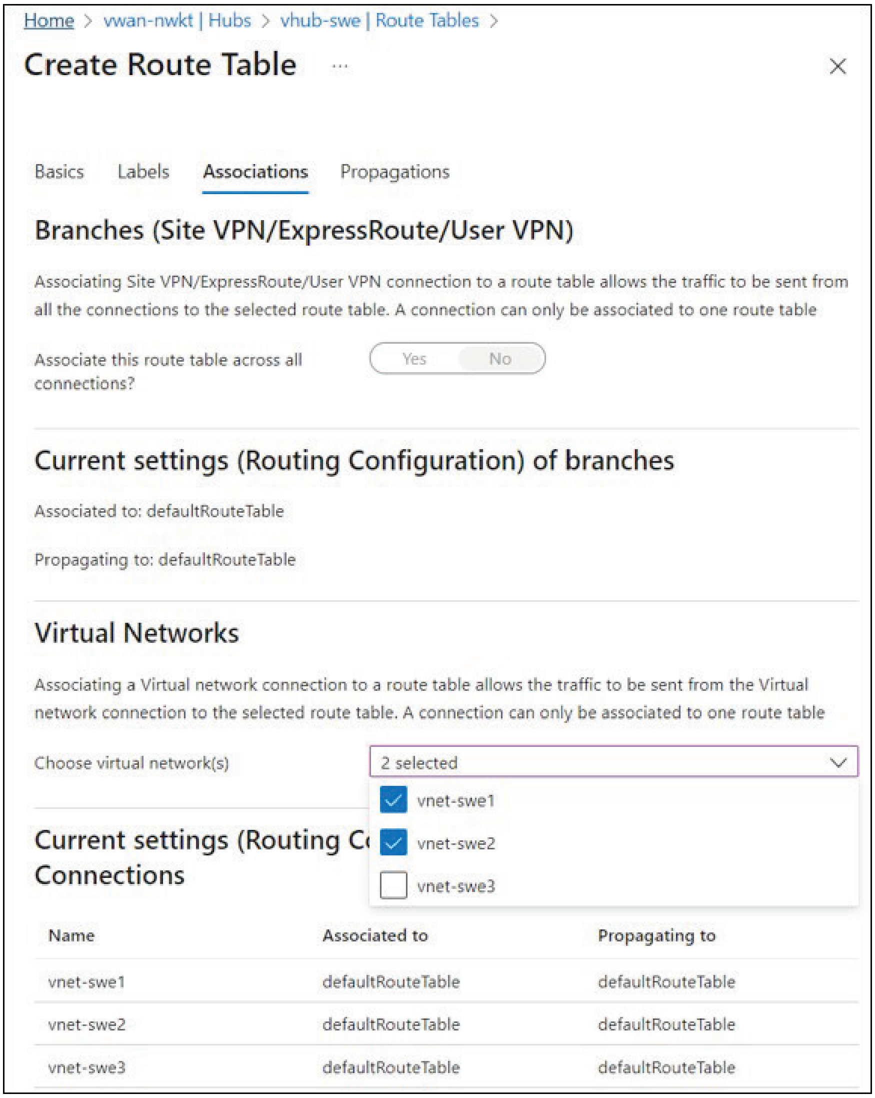
Subscription: NWKT
Resource group: rg-nsg-rt-swedencentral
Name *: rt-swe12

Branch routes apply to all connected VPN sites, ExpressRoute circuits and User VPN connections. Destination prefix can be aggregated address or list of all branch prefixes

Destination type: CIDR

Figure 12-8: *Create New Route Table: Basic Information.*

First, note that the *Associate this route table across connections* selector under the *Branches (Site VPN/ExpressRoute/User VPN)* section is unavailable because branch connections have to associate with the Default Route Table. Open the *Choose virtual network* drop-down menu and select vnets-swe1 and vnet-swe2 from the list. Then, proceed to the *Propagation*s tab.


Home > vwan-nwkt | Hubs > vhub-swe | Route Tables >

Create Route Table

Basics | Labels | Associations | Propagations

Branches (Site VPN/ExpressRoute/User VPN)

Associating Site VPN/ExpressRoute/User VPN connection to a route table allows the traffic to be sent from all the connections to the selected route table. A connection can only be associated to one route table

Associate this route table across all connections? Yes No

Current settings (Routing Configuration) of branches

Associated to: defaultRouteTable

Propagating to: defaultRouteTable

Virtual Networks

Associating a Virtual network connection to a route table allows the traffic to be sent from the Virtual network connection to the selected route table. A connection can only be associated to one route table

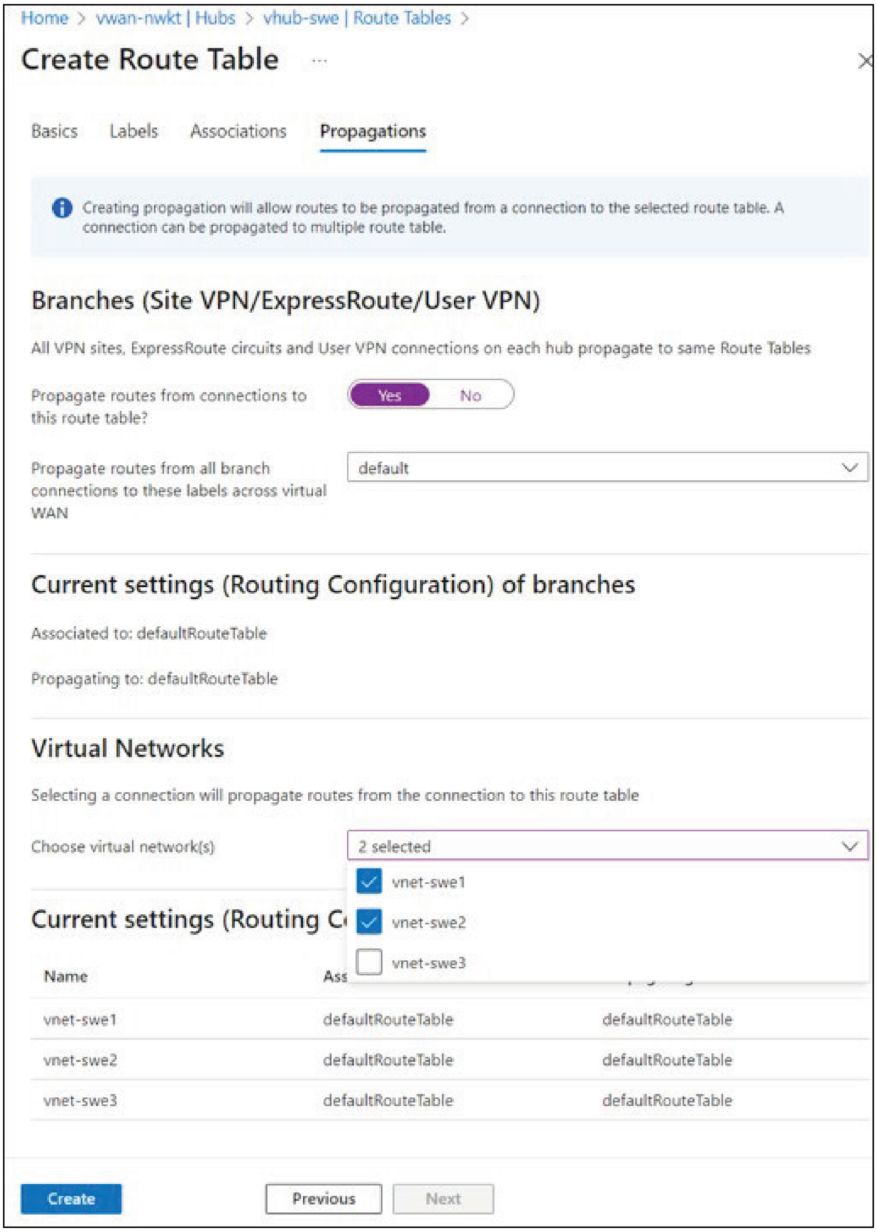
Choose virtual network(s): 2 selected
- ✓ vnet-swe1
- ✓ vnet-swe2
- ☐ vnet-swe3

Current settings (Routing C
Connections

Name	Associated to	Propagating to
vnet-swe1	defaultRouteTable	defaultRouteTable
vnet-swe2	defaultRouteTable	defaultRouteTable
vnet-swe3	defaultRouteTable	defaultRouteTable

Figure 12-9: *Create New Route Table: VNet Association.*

Set the *Propagate routes from connections to this route table* sliding bar to *Yes* (the default option is *No*). This way, the routing information from the Default RT, having routing information related to VPN/ExpressRoute connections, is propagated to RT rt-swe12. Next, propagate routes related to vnet-swe1 and vnet-swe2 by selecting VNets from the Choose virtual network(s) drop-down menu. Then click the *Create* button.



Create Route Table

Creating propagation will allow routes to be propagated from a connection to the selected route table. A connection can be propagated to multiple route table.

Branches (Site VPN/ExpressRoute/User VPN)

All VPN sites, ExpressRoute circuits and User VPN connections on each hub propagate to same Route Tables

Propagate routes from connections to this route table? [✓ Yes] No

Propagate routes from all branch connections to these labels across virtual WAN: default

Current settings (Routing Configuration) of branches

Associated to: defaultRouteTable

Propagating to: defaultRouteTable

Virtual Networks

Selecting a connection will propagate routes from the connection to this route table

Choose virtual network(s): 2 selected

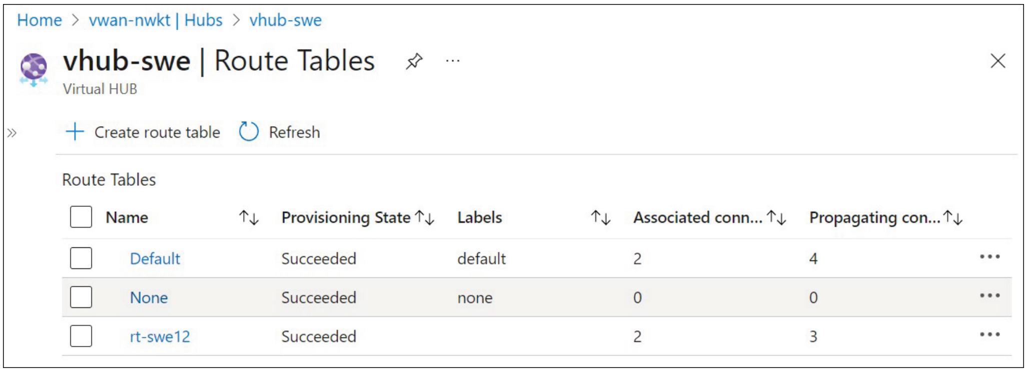
- ✓ vnet-swe1
- ✓ vnet-swe2
- ☐ vnet-swe3

Current settings (Routing C

Name	Ass	
vnet-swe1	defaultRouteTable	defaultRouteTable
vnet-swe2	defaultRouteTable	defaultRouteTable
vnet-swe3	defaultRouteTable	defaultRouteTable

Figure 12-10: *Create New Route Table: Route Propagation.*

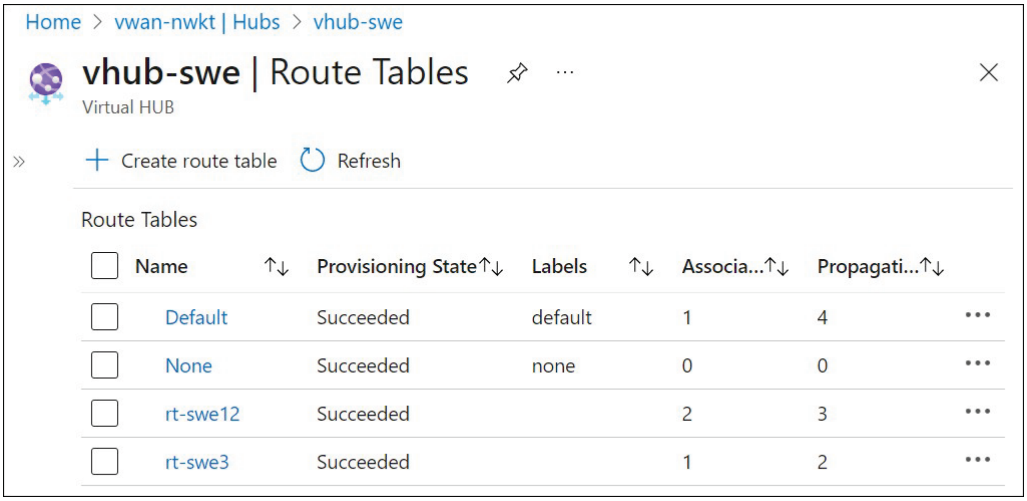
Figure 12-11 shows the new Route Table rt-swe12 on the Route Table of vhub-swe. The *Association* column verifies that it has two associated connections. The *Propagations connection* column, in turn, shows that routes from three connections are installed to this RT.



Name	Provisioning State	Labels	Associated conn...	Propagating con...
Default	Succeeded	default	2	4
None	Succeeded	none	0	0
rt-swe12	Succeeded		2	3

Figure 12-11: *Create New Route Table: Verification.*

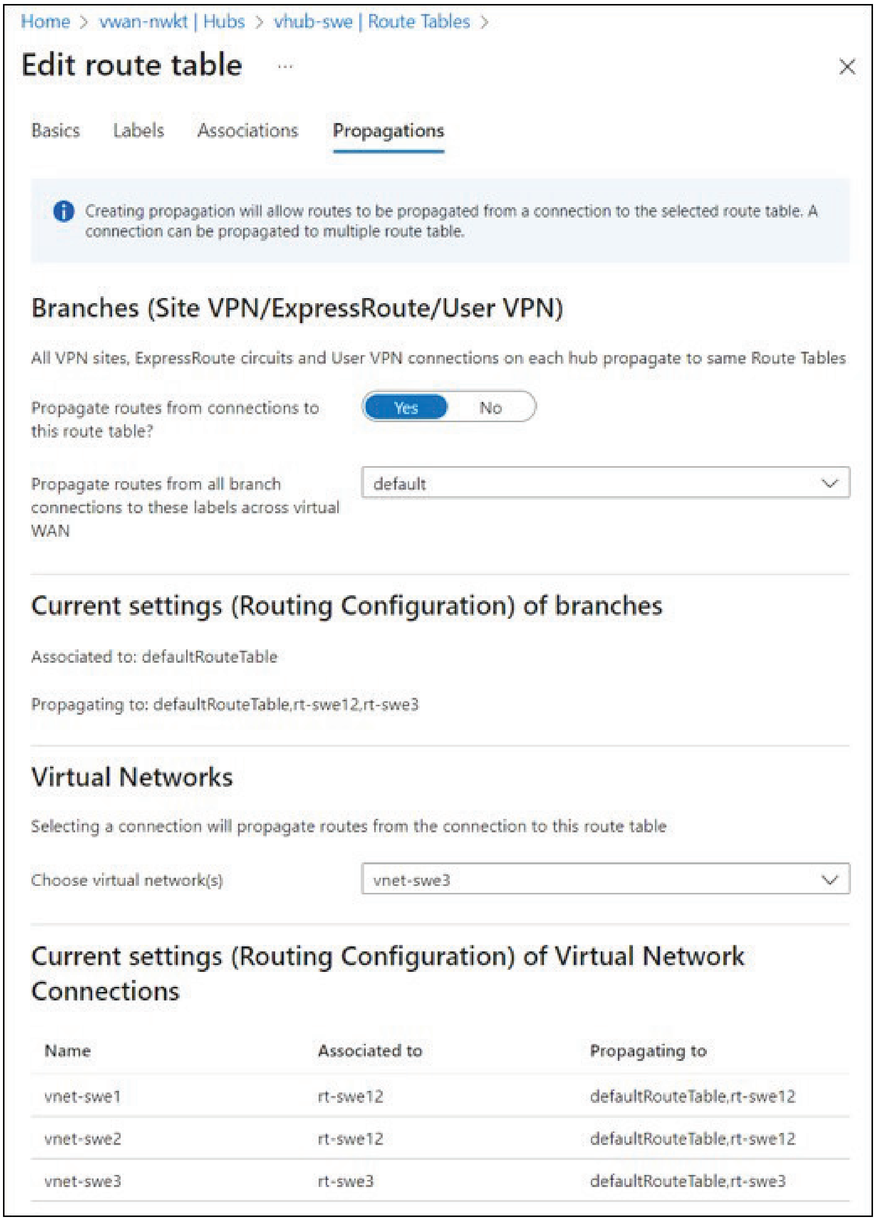
Figure 12-12 shows the Route Table list after we have created a dedicated RT for vnet-swe3. It has one association (vnet-swe3), and routes from the Default RT and rt-swe3 are propagated to RT. The Default RT has now only one association (Site-to-Site VPN). However, routes from vnet-swe1, vnet-swe2, vnet-swe3, and routing information related to Site-to-Site VPN are propagated to Default RT.



Name	Provisioning State	Labels	Associa...	Propagati...
Default	Succeeded	default	1	4
None	Succeeded	none	0	0
rt-swe12	Succeeded		2	3
rt-swe3	Succeeded		1	2

Figure 12-12: *Create New Route Table: Route Propagation.*

You can verify the connection association and route propagation by selecting one of the RTs and selecting a *Propagation* (or *Associations*) tab. The *Current setting (Routing Configuration) of branches* section verifies that VPN connections are associated with the Default RT and routing information learned via this connection (from VGW) is propagated to all three RTs (Default, rt-swe12, and rt-swe3). The *Current setting (Routing Configuration) of Virtual Networks* section shows that vnet-swe1 and vnet-swe2 are associated with the same RT rt-swe12, and routes are propagated to the Default RT and rt-swe-12. It also verifies that vnet-swe3 is associated with the RT rt-swe3, and routing information is installed to the Default RT and rt-swe3.



Creating propagation will allow routes to be propagated from a connection to the selected route table. A connection can be propagated to multiple route table.

Branches (Site VPN/ExpressRoute/User VPN)

All VPN sites, ExpressRoute circuits and User VPN connections on each hub propagate to same Route Tables

Propagate routes from connections to this route table? [✓ Yes] No

Propagate routes from all branch connections to these labels across virtual WAN: default

Current settings (Routing Configuration) of branches

Associated to: defaultRouteTable

Propagating to: defaultRouteTable,rt-swe12,rt-swe3

Virtual Networks

Selecting a connection will propagate routes from the connection to this route table

Choose virtual network(s): vnet-swe3

Current settings (Routing Configuration) of Virtual Network Connections

Name	Associated to	Propagating to
vnet-swe1	rt-swe12	defaultRouteTable,rt-swe12
vnet-swe2	rt-swe12	defaultRouteTable,rt-swe12
vnet-swe3	rt-swe3	defaultRouteTable,rt-swe3

Figure 12-13: *Route Association and Propagation Verification.*

Control Plane Verification

Figure 12-14 shows IP addressing scheme, connections, and other routing components related to this chapter. I've added the figure to make it easier to map destination IP subnets and next-hop installed into routing tables.

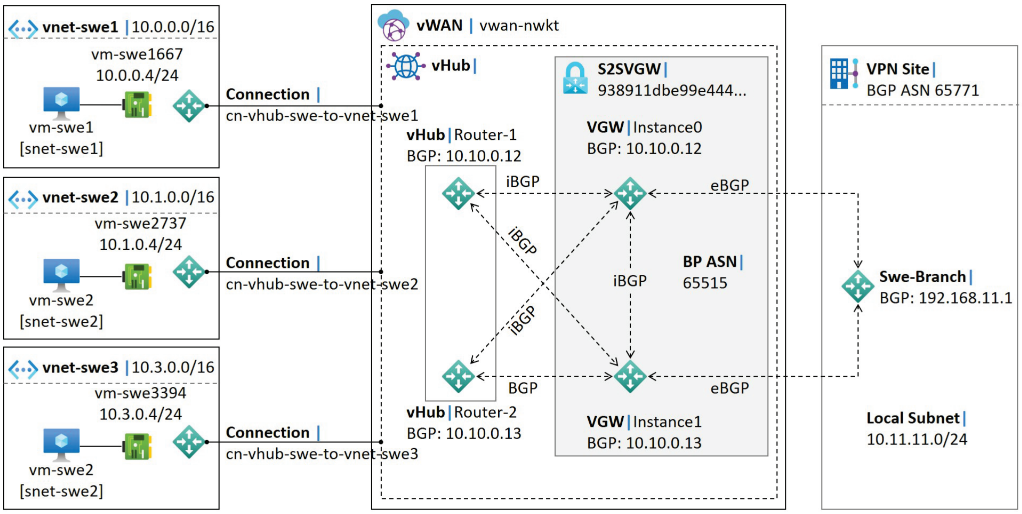
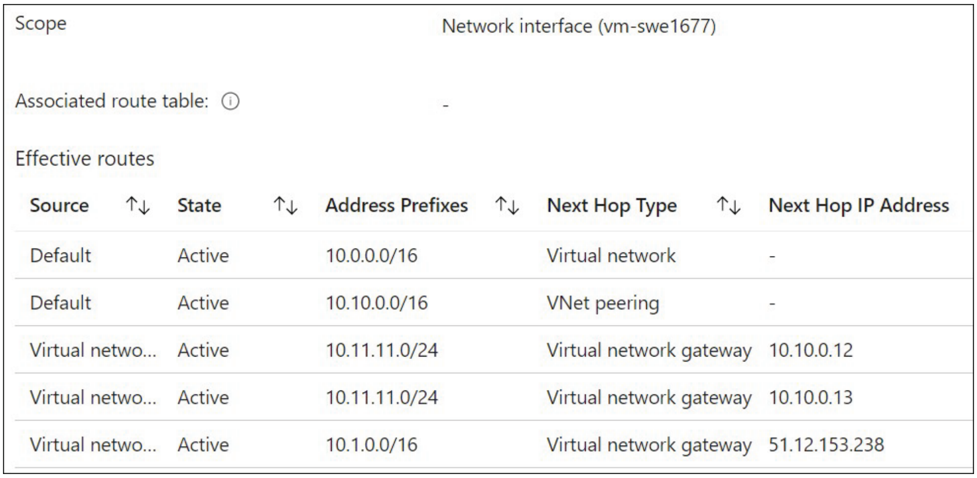


Figure 12-14: *Topology View.*

Figure 12-15 shows the effective routes programmed into NICs vm-swe1677 attached to VM vm-swe1 on VNet vnet-swe1. It has a route to 10.1.0.0/16 (vnet-swe2) but not to 10.3.0.0/16 (vnet-swe3) via VGW with the next-hop 51.12.153.238 (VGW's PIP, not shown in the figure). It also has learned subnet 10.11.11.0/24 (subnet on Branch Site) from both VGW instances. vHub CIDR is learned over the VNet Peering connection.



Scope: Network interface (vm-swe1677)

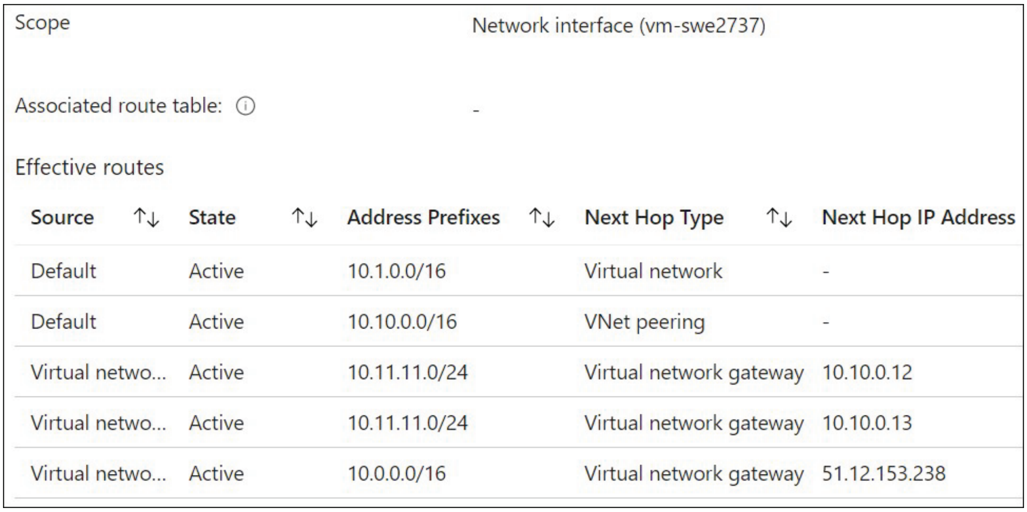
Associated route table: ⓘ -

Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Hop IP Address
Default	Active	10.0.0.0/16	Virtual network	-
Default	Active	10.10.0.0/16	VNet peering	-
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	10.10.0.12
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	10.10.0.13
Virtual netwo...	Active	10.1.0.0/16	Virtual network gateway	51.12.153.238

Figure 12-15: *NIC vm-swe1677 Effective Routes.*

Figure 12-16 verifies that the NIC vm-swe2737 has identical routing information with vm-swe1667 in its route table.



Scope — Network interface (vm-swe2737)

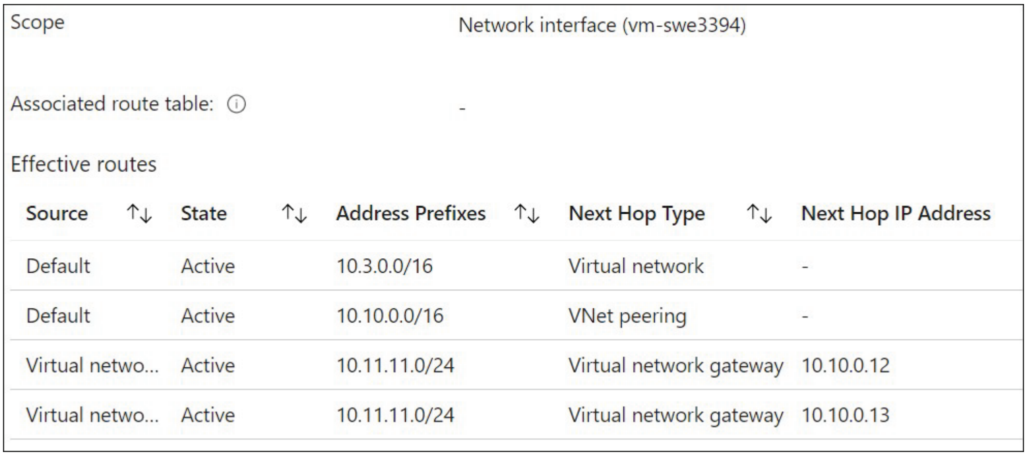
Associated route table: -

Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Hop IP Address
Default	Active	10.1.0.0/16	Virtual network	-
Default	Active	10.10.0.0/16	VNet peering	-
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	10.10.0.12
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	10.10.0.13
Virtual netwo...	Active	10.0.0.0/16	Virtual network gateway	51.12.153.238

Figure 12-16: *NIC vm-swe2737 Effective Routes.*

Figure 12-17 verifies that vm-swe3394 has route to neither 10.0.0.0/16 (vnet-swe1) nor 10.1.0.0/16 (vnet-swe2). Though, it has a route to 10.11.11.0/16 (Branch) and 10.10.0.0/16 (vHub).



Scope — Network interface (vm-swe3394)

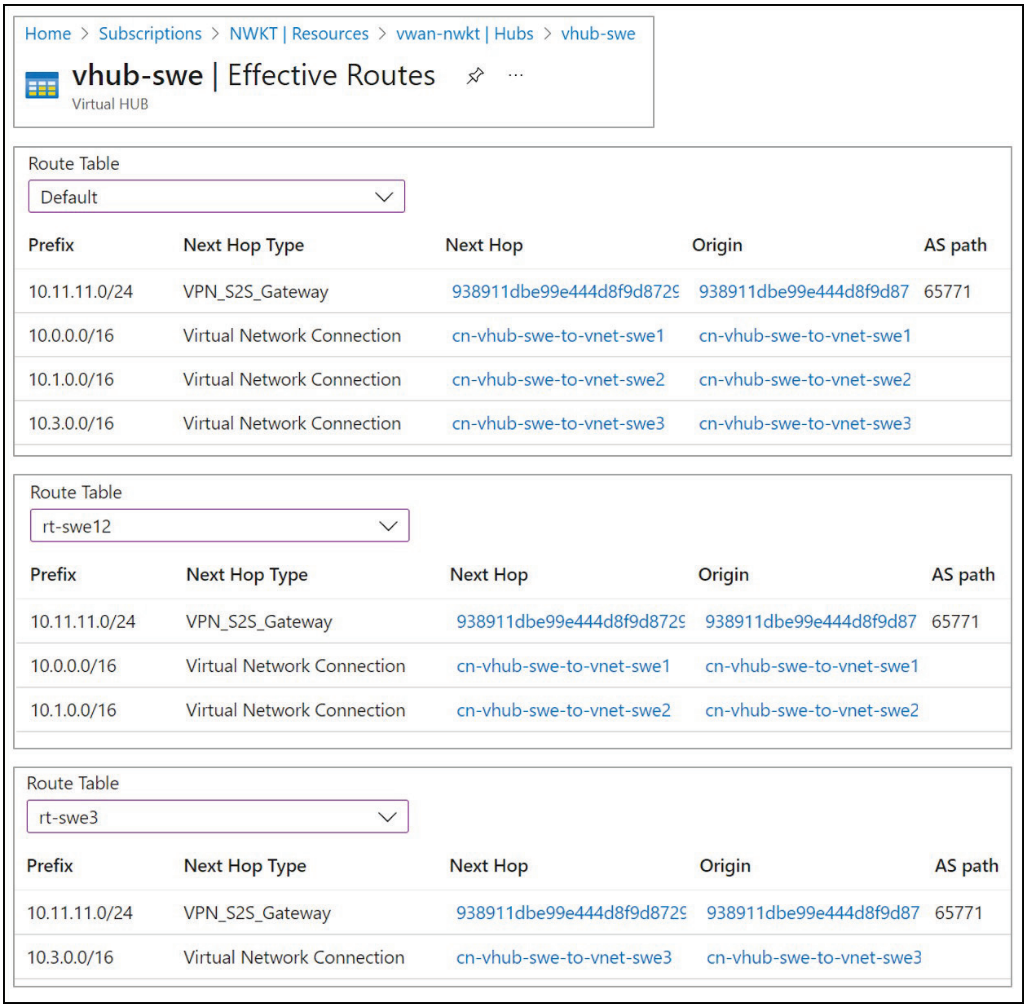
Associated route table: -

Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Hop IP Address
Default	Active	10.3.0.0/16	Virtual network	-
Default	Active	10.10.0.0/16	VNet peering	-
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	10.10.0.12
Virtual netwo...	Active	10.11.11.0/24	Virtual network gateway	10.10.0.13

Figure 12-17: *NIC vm-swe3394 Effective Routes.*

Figure 12-18 shows the routing information of all three RTs from the vHub perspective. The Default Route Table has route to VNets (vnet-swe1, vnet-swe2, and vnet-swe3) and Branch site. The second route table, rt-swe12, has routes to 10.0.0.0/16 (vnet-swe1), 10.1.0.0/16 (vnet-swe2), and 10.11.11.0/24 (VPN Site). The last route table, rt-swe3, has routes to 10.3.0.0/16 (vnet-swe3) and 10.11.11.0/24 (VPN Site) but not CIDR ranges of vnet-swe1 and vnet-swe2.


Home > Subscriptions > NWKT | Resources > vwan-nwkt | Hubs > vhub-swe

vhub-swe | Effective Routes

Virtual HUB

Route Table: Default

Prefix	Next Hop Type	Next Hop	Origin	AS path
10.11.11.0/24	VPN_S2S_Gateway	938911dbe99e444d8f9d8729	938911dbe99e444d8f9d87	65771
10.0.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe1	cn-vhub-swe-to-vnet-swe1	
10.1.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe2	cn-vhub-swe-to-vnet-swe2	
10.3.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe3	cn-vhub-swe-to-vnet-swe3	

Route Table: rt-swe12

Prefix	Next Hop Type	Next Hop	Origin	AS path
10.11.11.0/24	VPN_S2S_Gateway	938911dbe99e444d8f9d8729	938911dbe99e444d8f9d87	65771
10.0.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe1	cn-vhub-swe-to-vnet-swe1	
10.1.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe2	cn-vhub-swe-to-vnet-swe2	

Route Table: rt-swe3

Prefix	Next Hop Type	Next Hop	Origin	AS path
10.11.11.0/24	VPN_S2S_Gateway	938911dbe99e444d8f9d8729	938911dbe99e444d8f9d87	65771
10.3.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe3	cn-vhub-swe-to-vnet-swe3	

Figure 12-18: *Virtual Hub Routes.*

Example 12-1 verifies that the router Swe-Branch on the remote site has learned all three VNet CIDR Ranges and the CIDR range of vHub from both of its eBGP peers instance0 and instance1.

```
Swe-Branch#sh ip bgp
BGP table version is 6, local router ID is 192.168.11.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *    10.0.0.0/16      10.10.0.13                             0 65515 i
 *>                    10.10.0.12                             0 65515 i
 *    10.1.0.0/16      10.10.0.13                             0 65515 i
 *>                    10.10.0.12                             0 65515 i
 *    10.3.0.0/16      10.10.0.13                             0 65515 i
 *>                    10.10.0.12                             0 65515 i
 *    10.10.0.0/16     10.10.0.13                             0 65515 i
 *>                    10.10.0.12                             0 65515 i
 *>   10.11.11.0/24    0.0.0.0                  0         32768 i
```

Example 12-1: *BGP Table of Swe-Branch.*

Reachability information is also installed from BGP table to Routing Table.

```
Swe-Branch#sh ip route bgp
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is 192.168.100.1 to network 0.0.0.0

      10.0.0.0/8 is variably subnetted, 8 subnets, 3 masks
B        10.0.0.0/16 [20/0] via 10.10.0.12, 02:14:11
B        10.1.0.0/16 [20/0] via 10.10.0.12, 06:56:07
B        10.3.0.0/16 [20/0] via 10.10.0.12, 02:17:23
B        10.10.0.0/16 [20/0] via 10.10.0.12, 06:56:07
```

Example 12-2: *Routing Table of Swe-Branch.*

Data Plane Testing

Example 12-3 verifies that we have IP connectivity from VPN Site swe-branch to all thre VNets.

```
Swe-Branch#ping 10.0.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 11/11/12 ms

Swe-Branch#ping 10.1.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/13/15 ms

Swe-Branch#ping 10.3.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/12/14 ms
```

Example 12-3: *Ping from Swe-Branch (10.11.11.1) to Virtual Machines .*

Example 12-4 shows that we can ping vm-swe2 (10.1.0.4) from the vm-swe1 (10.0.0.4). However, we can’t ping vm-swe3 (10.3.0.4).

```
azureuser@vm-swe1:~$ ping 10.1.0.4
PING 10.1.0.4 (10.1.0.4) 56(84) bytes of data.
64 bytes from 10.1.0.4: icmp_seq=1 ttl=63 time=3.31 ms
64 bytes from 10.1.0.4: icmp_seq=2 ttl=63 time=2.17 ms
64 bytes from 10.1.0.4: icmp_seq=3 ttl=63 time=2.10 ms
64 bytes from 10.1.0.4: icmp_seq=4 ttl=63 time=1.65 ms
^C
--- 10.1.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 1.653/2.306/3.311/0.612 ms


azureuser@vm-swe1:~$ ping 10.3.0.4
PING 10.3.0.4 (10.3.0.4) 56(84) bytes of data.
^C
--- 10.3.0.4 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5113ms
```

Example 12-4: *Ping from vm-swe1 to vm-swe2 (10.1.0.4) and vm-swe2 (10.3.0.4).*

References

[1] Scenario: Isolating VNets
https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-udr-overview

Chapter 13: Virtual WAN Part III - Global Transit Network

Introduction

This chapter introduces a global transit network architecture based on Azure vWAN. Figure 13-1 illustrates our example vWAN topology where we have vHubs in Sweden (Swedencentral) and Germany (Germany West Central). It allows cross-regional, any-to-any connections between Virtual Networks and Branches. We have deployed vWAN (vwan-nwkt) and connected the VPN Site (swe-branch) and three VNets to its vHub (vhub-swe). In this chapter, we first create a new vHub (vhub-ger) and VPN Site (*ger-branch)* in the Azure Germany region. Then we connect vnet-ger1 to vHub. The vHub-to-vHub interconnect doesn't require our actions because Azure creates an Inter-Regional vHub connection between (full-mesh). When our deployment is ready, we verify the Control Plane operation by checking that each resource has the required routes in its Route Table. As the last step, we test regional and cross-regions Branch-to-VNet, and VNet-to-VNet connections.

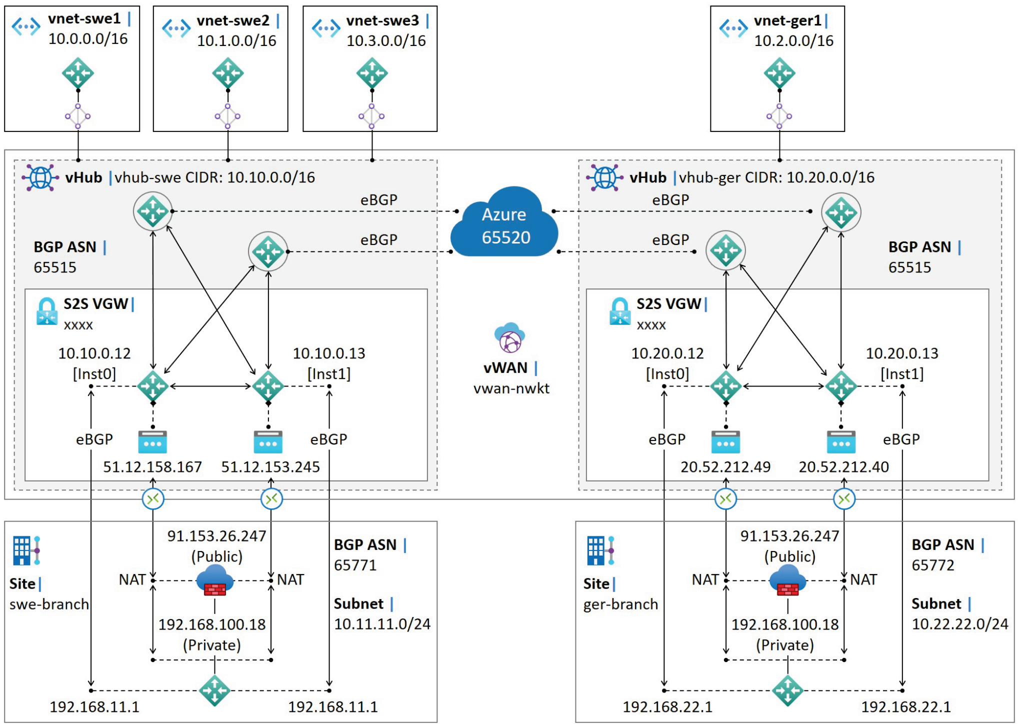
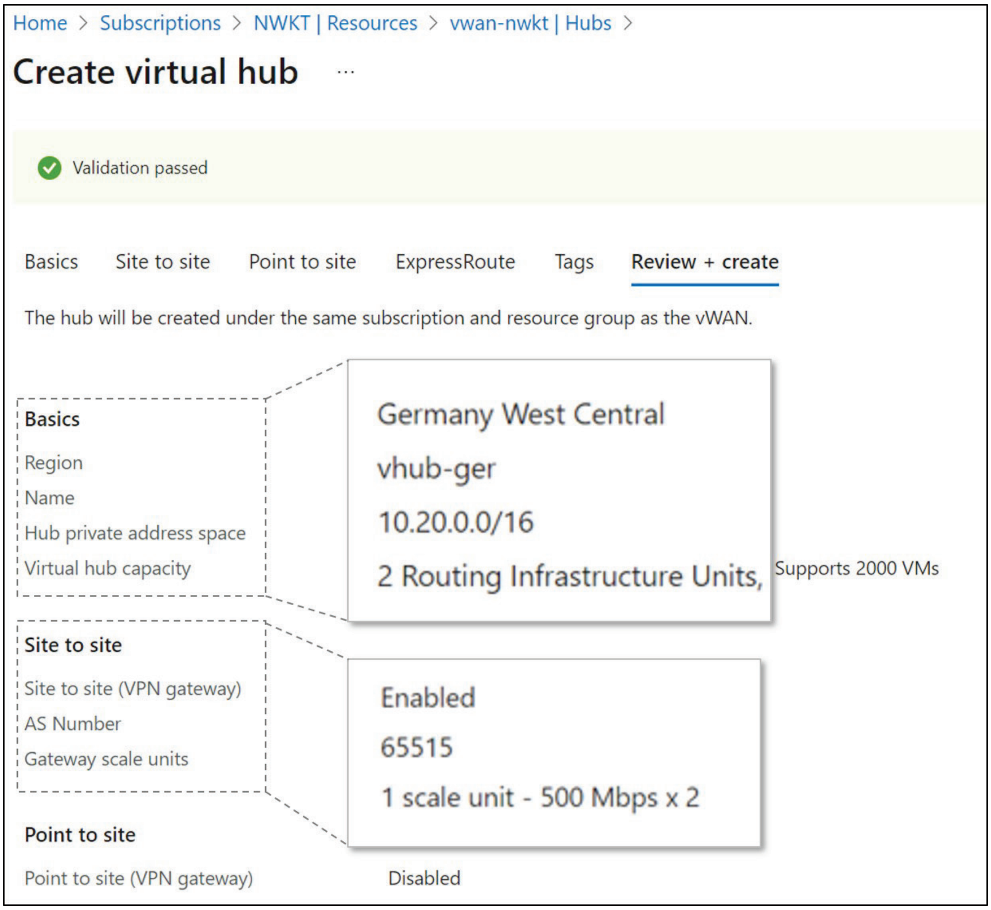


Figure 13-1: *Global Transit Network Architecture.*

Create a New vHub: vhub-ger

We deploy our new vHub and its connections (VPN Site and VNet) in the same way as in chapter 11. Figure 13-2 shows the validation page before deployment (I have zoomed out the output). The CIDR range is 10.20.0.0/16, and the Azure auto-generated BGP ASN is the same 65515 that we have in vhub-swe.



The hub will be created under the same subscription and resource group as the vWAN.

Basics

Region	Germany West Central
Name	vhub-ger
Hub private address space	10.20.0.0/16
Virtual hub capacity	2 Routing Infrastructure Units, Supports 2000 VMs

Site to site

Site to site (VPN gateway)	Enabled
AS Number	65515
Gateway scale units	1 scale unit - 500 Mbps x 2

Point to site

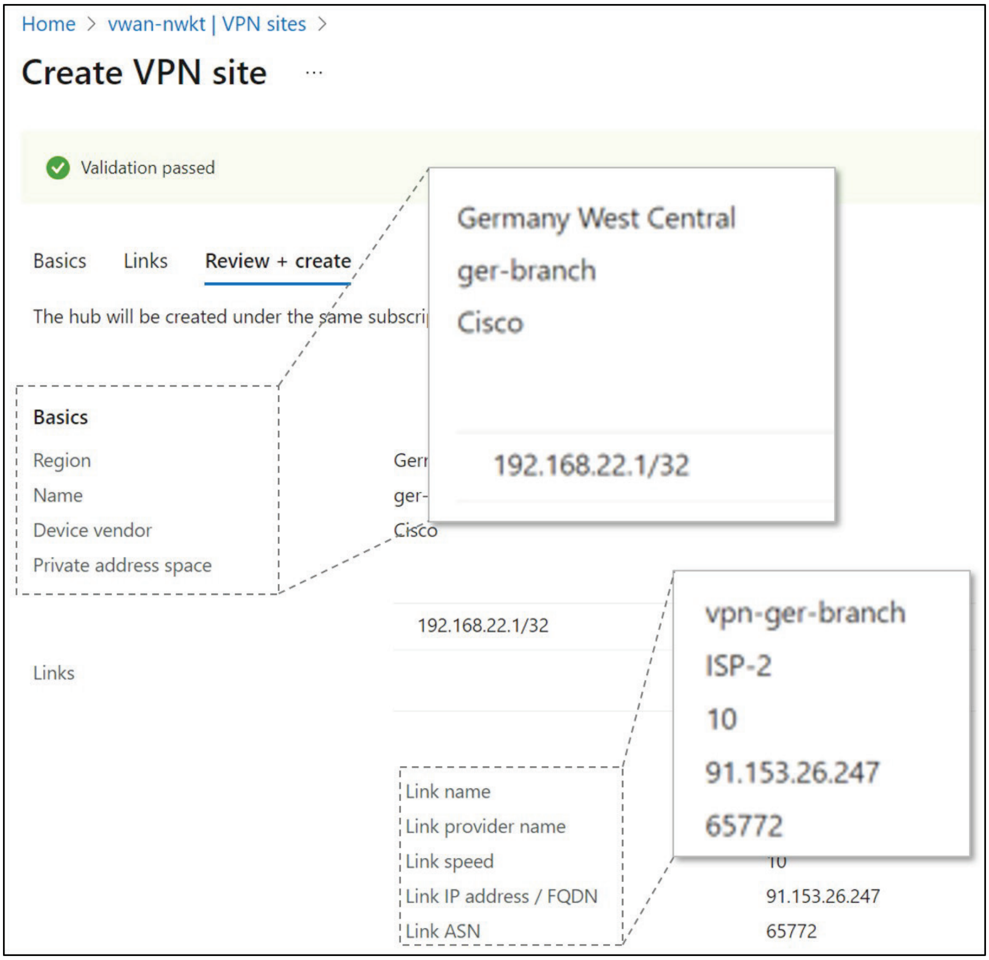
Point to site (VPN gateway)	Disabled

Figure 13-2: *Review of vHub vhub-ger.*

Create a New VPN Site: ger-branch

Figure 13-3 shows the Review page of your new VPN Site ger-branch. Its public IP address for IPSec tunnels is 91.153.26.247. The BGP peering IP address is 192.168.22.1, and the BGP ASN is 65772.

Note! The PIP is the same in both swe-branch and ger-branch because I run both VPN sites as virtual machines on the same server with one Internet connection.


Home > vwan-nwkt | VPN sites >

Create VPN site ...

Validation passed

Basics Links Review + create

The hub will be created under the same subscri

Basics

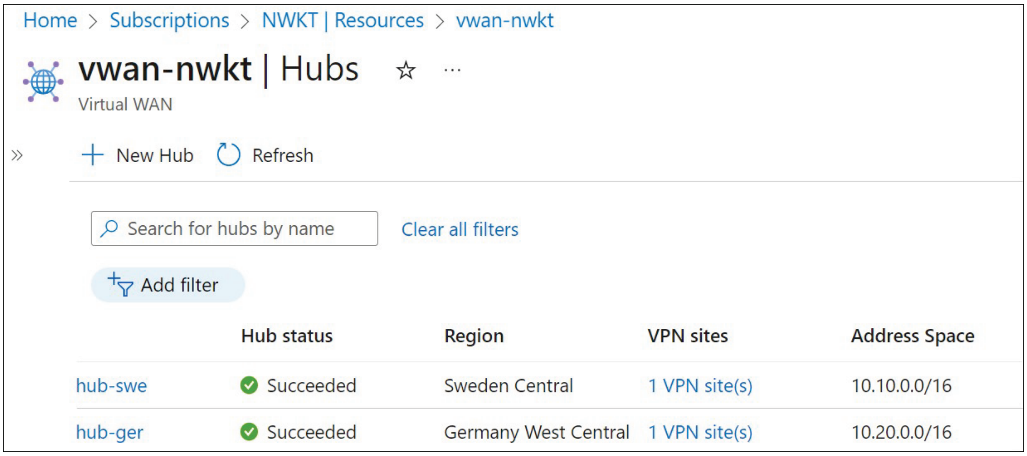
Region	Germany West Central
Name	ger-branch
Device vendor	Cisco
Private address space	192.168.22.1/32

Links

Link name	vpn-ger-branch
Link provider name	ISP-2
Link speed	10
Link IP address / FQDN	91.153.26.247
Link ASN	65772

Figure 13-3: *Review of the VPN Site.*

After deploying a new vHub *vhub-ger*, and its connection to VPN Site *ger-branch* and VNet *vnet-ger1*, we can verify that our deployment is succeeded from the *vwan-nwkt* page. Figure 13-4 shows that we have successfully deployed both vHubs in our vWAN with one connected VPN Site.


Home > Subscriptions > NWKT | Resources > vwan-nwkt

vwan-nwkt | Hubs

Virtual WAN

New Hub | Refresh

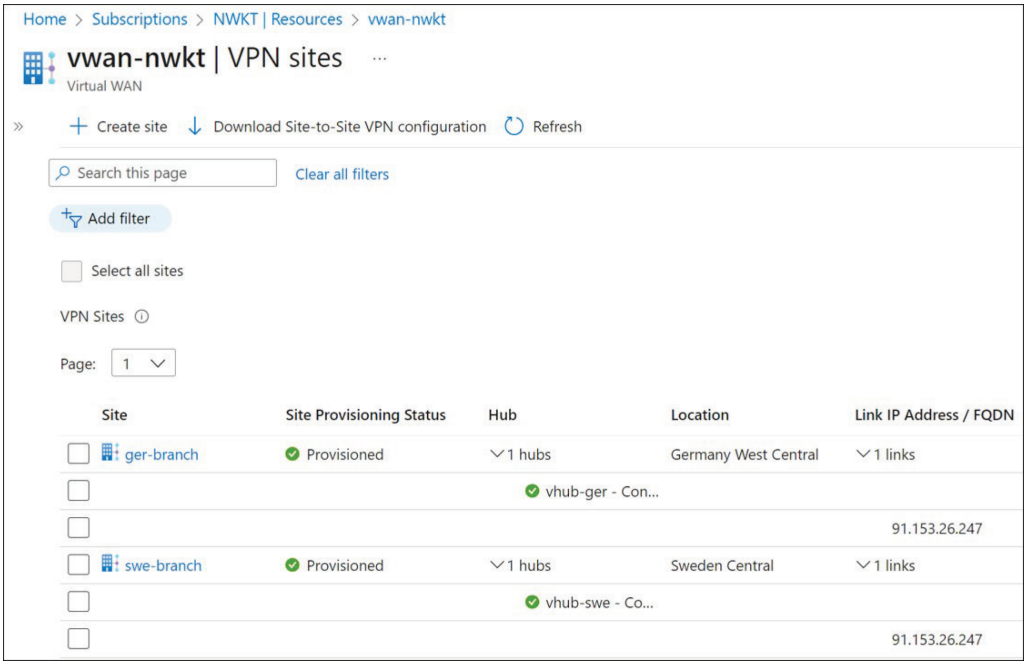
Search for hubs by name | Clear all filters

Add filter

	Hub status	Region	VPN sites	Address Space
hub-swe	✓ Succeeded	Sweden Central	1 VPN site(s)	10.10.0.0/16
hub-ger	✓ Succeeded	Germany West Central	1 VPN site(s)	10.20.0.0/16

Figure 13-4: *Virtual Hubs Deployed to vWAN.*

Figure 13-5, in turn, shows that both VPN Sites are provisioned and connected to their regional vHubs.


Home > Subscriptions > NWKT | Resources > vwan-nwkt

vwan-nwkt | VPN sites

Virtual WAN

Create site | Download Site-to-Site VPN configuration | Refresh

Search this page | Clear all filters

Add filter

Select all sites

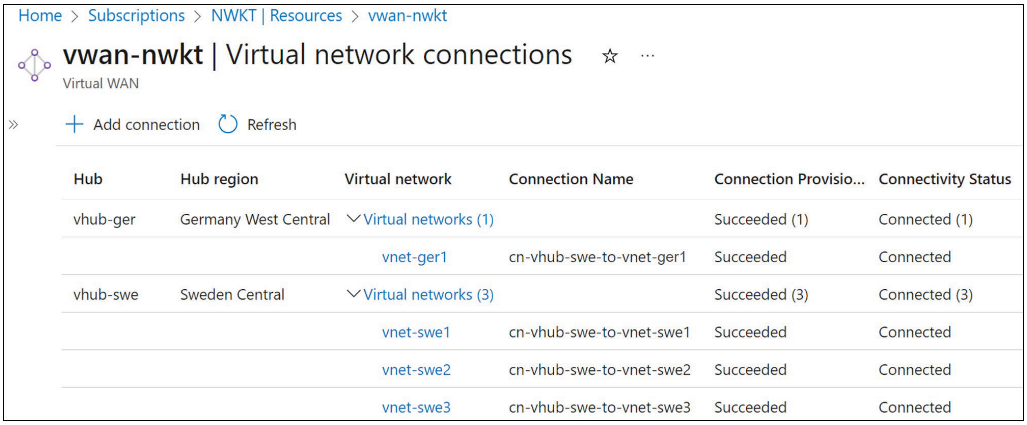
VPN Sites

Page: 1

Site	Site Provisioning Status	Hub	Location	Link IP Address / FQDN
ger-branch	✓ Provisioned	1 hubs	Germany West Central	1 links
		✓ vhub-ger - Con...		
				91.153.26.247
swe-branch	✓ Provisioned	1 hubs	Sweden Central	1 links
		✓ vhub-swe - Co...		
				91.153.26.247

Figure 13-5: *VPN Sites Deployed to vWAN.*

Figure 13-6 shows all our VNets in Sweden and Germany with their vHub connections.


Home > Subscriptions > NWKT | Resources > vwan-nwkt

vwan-nwkt | Virtual network connections

Virtual WAN

+ Add connection   Refresh

Hub	Hub region	Virtual network	Connection Name	Connection Provisio...	Connectivity Status
vhub-ger	Germany West Central	Virtual networks (1)		Succeeded (1)	Connected (1)
		vnet-ger1	cn-vhub-swe-to-vnet-ger1	Succeeded	Connected
vhub-swe	Sweden Central	Virtual networks (3)		Succeeded (3)	Connected (3)
		vnet-swe1	cn-vhub-swe-to-vnet-swe1	Succeeded	Connected
		vnet-swe2	cn-vhub-swe-to-vnet-swe2	Succeeded	Connected
		vnet-swe3	cn-vhub-swe-to-vnet-swe3	Succeeded	Connected

Figure 13-6: *VNets Deployed to vWAN.*

Control Plane verification

Figure 13-7 illustrates the simplified view of our example vWAN deployment. You can use it as a reference picture when we study routing in this section.

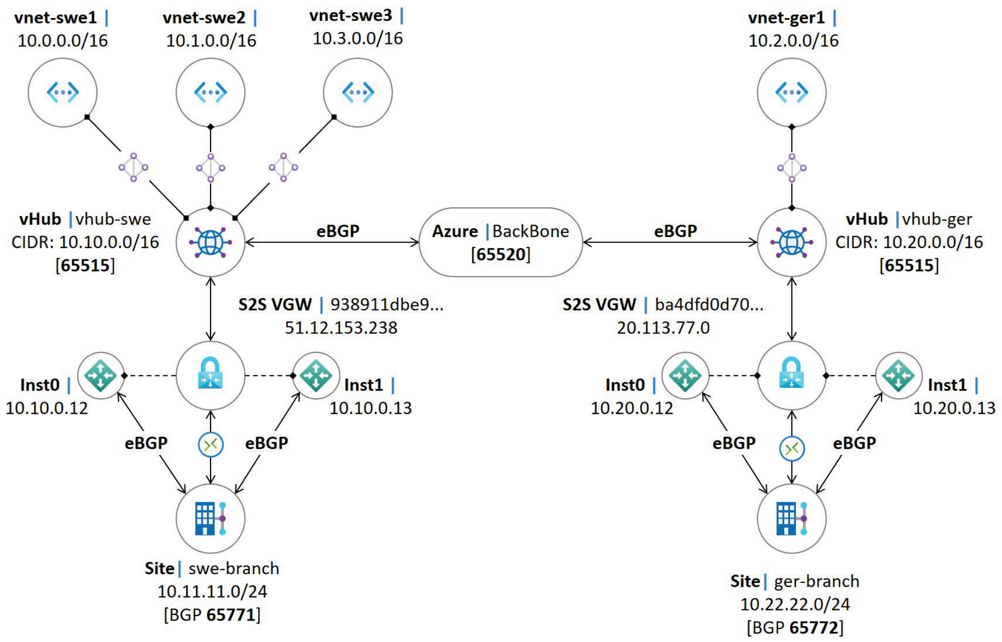
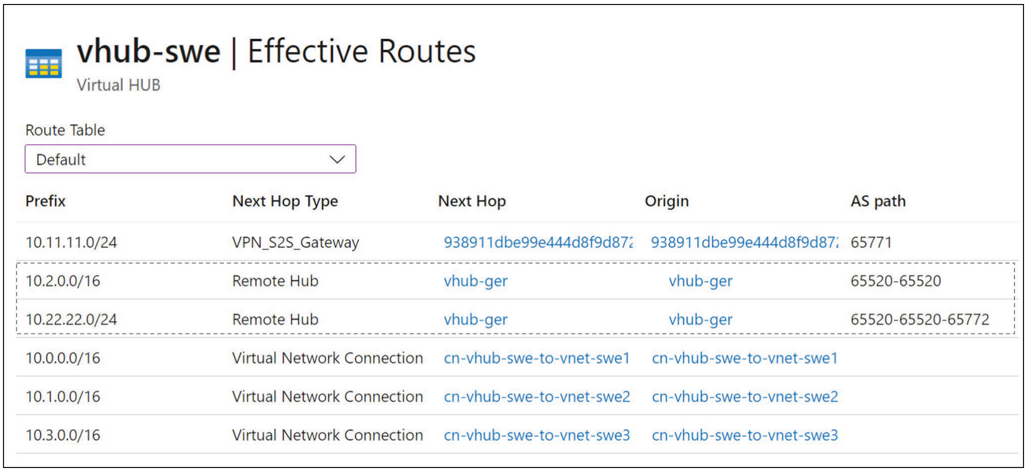

vnet-swe1 | 10.0.0.0/16
vnet-swe2 | 10.1.0.0/16
vnet-swe3 | 10.3.0.0/16
vnet-ger1 | 10.2.0.0/16
vHub | vhub-swe CIDR: 10.10.0.0/16 [65515]
eBGP
Azure | BackBone [65520]
eBGP
vHub | vhub-ger CIDR: 10.20.0.0/16 [65515]
S2S VGW | 938911dbe9... 51.12.153.238
S2S VGW | ba4dfd0d70... 20.113.77.0
Inst0 | 10.10.0.12
Inst1 | 10.10.0.13
Inst0 | 10.20.0.12
Inst1 | 10.20.0.13
eBGP
eBGP
eBGP
eBGP
Site | swe-branch 10.11.11.0/24 [BGP 65771]
Site | ger-branch 10.22.22.0/24 [BGP 65772]

Figure 13-7: *Reference Topology for Routing Studies.*

vHub Effective Routes

Figures 13-8 and 13-9 show the routing from the vhub-swe and vhub-ger perspectives. The information in the figures is self-descriptive, so there's no need to go through each line. However, way pay attention to cross-region routing information. Because both vhub-swe and vhub-ger use the same BGP ASN 65515, Azure Backbone routers override the original ASN from inter-vHub BGP Update messages. This is one way to get around the BGP routing loop prevention rule, which ignores the BGP Updates from external peers carrying the local BGP ASN in its AS-Path list.

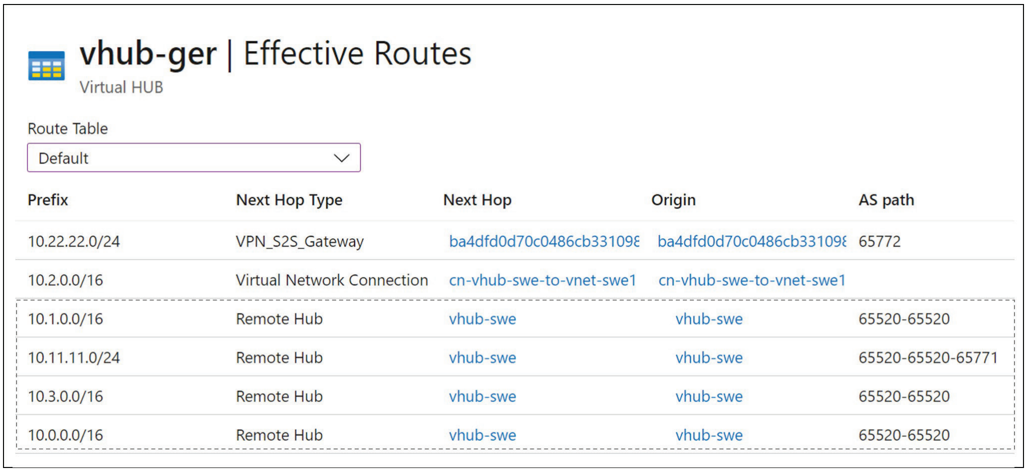


vhub-swe | Effective Routes
Virtual HUB

Route Table: Default

Prefix	Next Hop Type	Next Hop	Origin	AS path
10.11.11.0/24	VPN_S2S_Gateway	938911dbe99e444d8f9d87	938911dbe99e444d8f9d87	65771
10.2.0.0/16	Remote Hub	vhub-ger	vhub-ger	65520-65520
10.22.22.0/24	Remote Hub	vhub-ger	vhub-ger	65520-65520-65772
10.0.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe1	cn-vhub-swe-to-vnet-swe1	
10.1.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe2	cn-vhub-swe-to-vnet-swe2	
10.3.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe3	cn-vhub-swe-to-vnet-swe3	

Figure 13-8: *vhub-swe Effective Routes.*



vhub-ger | Effective Routes
Virtual HUB

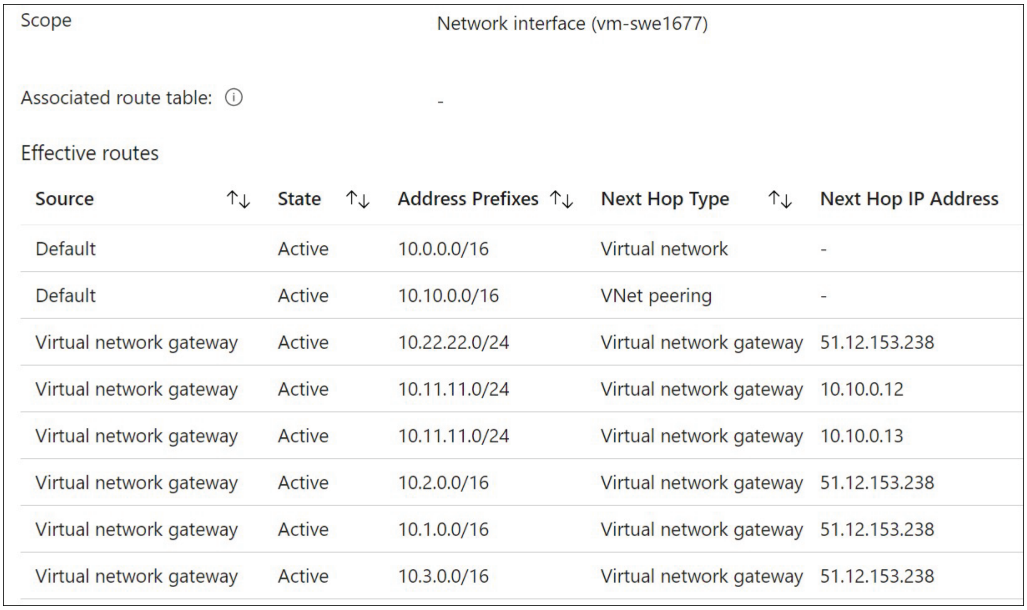
Route Table: Default

Prefix	Next Hop Type	Next Hop	Origin	AS path
10.22.22.0/24	VPN_S2S_Gateway	ba4dfd0d70c0486cb33109	ba4dfd0d70c0486cb33109	65772
10.2.0.0/16	Virtual Network Connection	cn-vhub-swe-to-vnet-swe1	cn-vhub-swe-to-vnet-swe1	
10.1.0.0/16	Remote Hub	vhub-swe	vhub-swe	65520-65520
10.11.11.0/24	Remote Hub	vhub-swe	vhub-swe	65520-65520-65771
10.3.0.0/16	Remote Hub	vhub-swe	vhub-swe	65520-65520
10.0.0.0/16	Remote Hub	vhub-swe	vhub-swe	65520-65520

Figure 13-9: *vhub-ger Effective Routes.*

vNIC Effective Routes

The following four figures illustrate the vNIC routes. Virtual NICs learn VNets and cross-region VPN site routing information from the VGW. The vNIC vm-swe1667 uses the next-hop IP addresses 51.12.153.238 for all VNets and cross-region Branch routes (10.22.22.0/16). Besides, it uses the next-hop IP addresses 10.10.0.12 (VGW int0) and 10.10.0.13 (VGW inst1) is used for regional Branch (10.11.11.0/16) route. The same routing applies to all other vNIC.

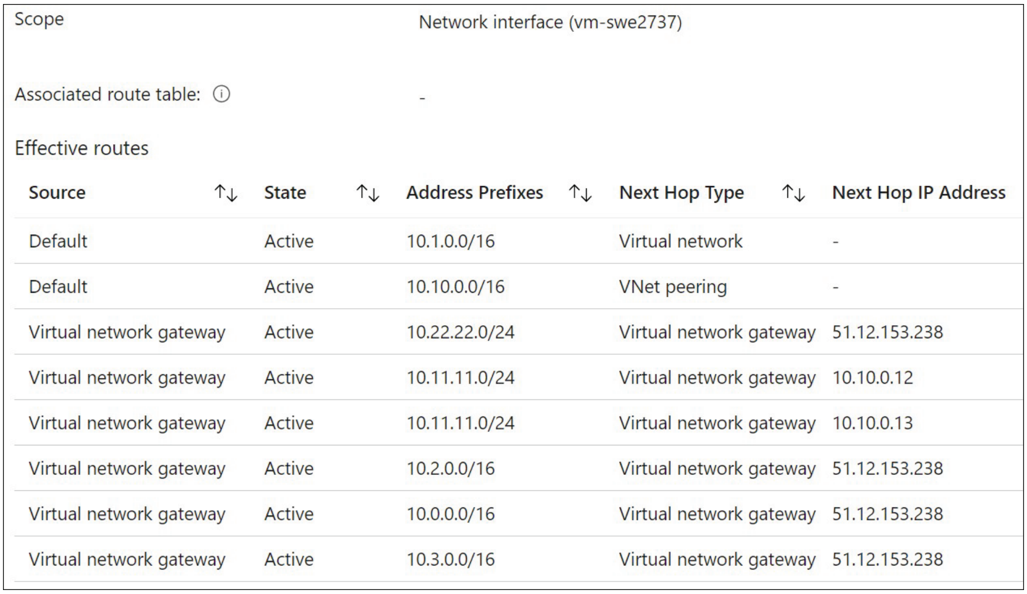


Scope	Network interface (vm-swe1677)
Associated route table: ⓘ	-

Effective routes

Source ↑↓	State ↑↓	Address Prefixes ↑↓	Next Hop Type ↑↓	Next Hop IP Address
Default	Active	10.0.0.0/16	Virtual network	-
Default	Active	10.10.0.0/16	VNet peering	-
Virtual network gateway	Active	10.22.22.0/24	Virtual network gateway	51.12.153.238
Virtual network gateway	Active	10.11.11.0/24	Virtual network gateway	10.10.0.12
Virtual network gateway	Active	10.11.11.0/24	Virtual network gateway	10.10.0.13
Virtual network gateway	Active	10.2.0.0/16	Virtual network gateway	51.12.153.238
Virtual network gateway	Active	10.1.0.0/16	Virtual network gateway	51.12.153.238
Virtual network gateway	Active	10.3.0.0/16	Virtual network gateway	51.12.153.238

Figure 13-10: *Routing Information Installed into vm-swe1667 (vm-swe1 on vnet-swe1).*

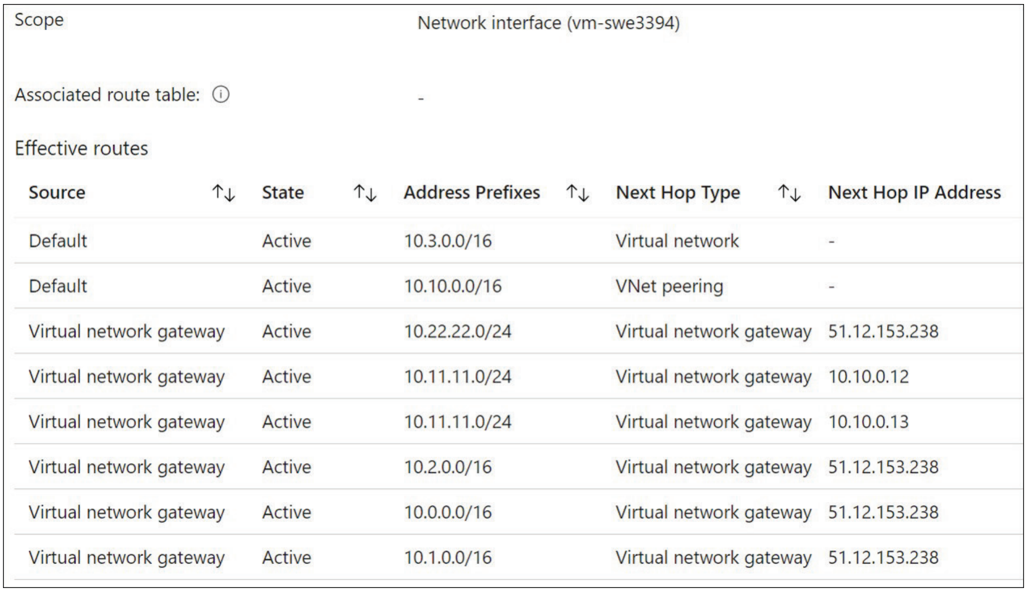


Scope	Network interface (vm-swe2737)
Associated route table:	-

Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Hop IP Address
Default	Active	10.1.0.0/16	Virtual network	-
Default	Active	10.10.0.0/16	VNet peering	-
Virtual network gateway	Active	10.22.22.0/24	Virtual network gateway	51.12.153.238
Virtual network gateway	Active	10.11.11.0/24	Virtual network gateway	10.10.0.12
Virtual network gateway	Active	10.11.11.0/24	Virtual network gateway	10.10.0.13
Virtual network gateway	Active	10.2.0.0/16	Virtual network gateway	51.12.153.238
Virtual network gateway	Active	10.0.0.0/16	Virtual network gateway	51.12.153.238
Virtual network gateway	Active	10.3.0.0/16	Virtual network gateway	51.12.153.238

Figure 13-11: *Routing Information Installed into vm-swe2737 (vm-swe2 on vnet-swe2).*

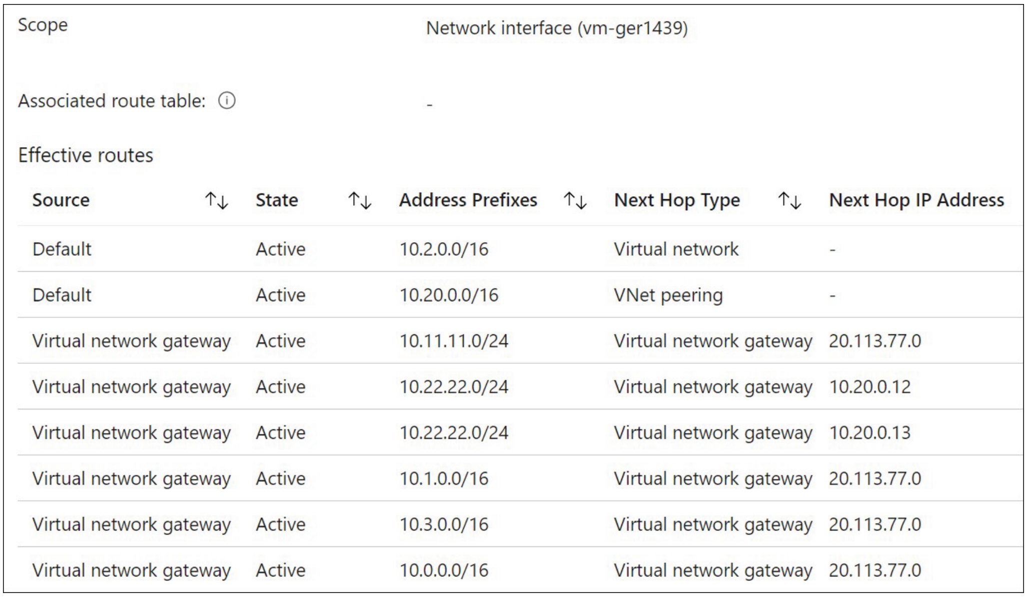


Scope	Network interface (vm-swe3394)
Associated route table:	-

Effective routes

Source	State	Address Prefixes	Next Hop Type	Next Hop IP Address
Default	Active	10.3.0.0/16	Virtual network	-
Default	Active	10.10.0.0/16	VNet peering	-
Virtual network gateway	Active	10.22.22.0/24	Virtual network gateway	51.12.153.238
Virtual network gateway	Active	10.11.11.0/24	Virtual network gateway	10.10.0.12
Virtual network gateway	Active	10.11.11.0/24	Virtual network gateway	10.10.0.13
Virtual network gateway	Active	10.2.0.0/16	Virtual network gateway	51.12.153.238
Virtual network gateway	Active	10.0.0.0/16	Virtual network gateway	51.12.153.238
Virtual network gateway	Active	10.1.0.0/16	Virtual network gateway	51.12.153.238

Figure 13-12: *Routing Information Installed into vm-swe3394 (vm-swe3 on vnet-swe3).*



Scope	Network interface (vm-ger1439)
Associated route table: ⓘ	-

Effective routes

Source ↑↓	State ↑↓	Address Prefixes ↑↓	Next Hop Type ↑↓	Next Hop IP Address
Default	Active	10.2.0.0/16	Virtual network	-
Default	Active	10.20.0.0/16	VNet peering	-
Virtual network gateway	Active	10.11.11.0/24	Virtual network gateway	20.113.77.0
Virtual network gateway	Active	10.22.22.0/24	Virtual network gateway	10.20.0.12
Virtual network gateway	Active	10.22.22.0/24	Virtual network gateway	10.20.0.13
Virtual network gateway	Active	10.1.0.0/16	Virtual network gateway	20.113.77.0
Virtual network gateway	Active	10.3.0.0/16	Virtual network gateway	20.113.77.0
Virtual network gateway	Active	10.0.0.0/16	Virtual network gateway	20.113.77.0

Figure 13-13: *Routing Information Installed into vm-ger1439 (vm-ger1 on vnet-ger1).*

Branch Site Routing

Example 13-1 shows that Ger-Branch has received six routes from its BGP peer 10.20.0.12 (ints0) and 10.20.0.13 (ints1). Example 13-2 shows that Swe-Branch has also received six routes from its BGP peer 10.10.0.12 (ints0) and 10.10.0.13 (ints1).

```
Ger-Branch#sh ip bgp summ | beg Neighbor
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
10.20.0.12      4        65515     117     118        8    0    0 01:37:00        6
10.20.0.13      4        65515     118     117        8    0    0 01:37:03        6
```

Example 13-1: *BGP Peering of Ger-Branch.*

```
Swe-Branch#sh ip bgp summ | beg Neighbor
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
10.10.0.12      4        65515     234     234        8    0    0 03:18:02        6
10.10.0.13      4        65515     234     236        8    0    0 03:17:56        6
```

Example 13-2: *BGP Peering of Swe-Branch.*

The following two examples show BGP tables branch routers Ger-Branch and Swe-Branch. Routes learned from the Instance0 are selected as the best routes due to the BGP Router-Id.

```
Ger-Branch#sh ip bgp | beg Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>   10.0.0.0/16      10.20.0.12                             0 65515 65520 65520 e
 *                     10.20.0.13                             0 65515 65520 65520 e
 *>   10.1.0.0/16      10.20.0.12                             0 65515 65520 65520 e
 *                     10.20.0.13                             0 65515 65520 65520 e
 *    10.2.0.0/16      10.20.0.13                             0 65515 i
 *>                    10.20.0.12                             0 65515 i
 *>   10.3.0.0/16      10.20.0.12                             0 65515 65520 65520 e
 *                     10.20.0.13                             0 65515 65520 65520 e
 *>   10.11.11.0/24    10.20.0.12                             0 65515 65520 65520 65771 e
 *                     10.20.0.13                             0 65515 65520 65520 65771 e
 *>   10.20.0.0/16     10.20.0.12                             0 65515 i
 *                     10.20.0.13                             0 65515 i
 *>   10.22.22.0/24    0.0.0.0                  0         32768 i
```
Example 13-3: *BGP Table of Ger-Branch.*

```
Swe-Branch#sh ip bgp | beg Network
     Network          Next Hop            Metric LocPrf Weight Path
 *    10.0.0.0/16      10.10.0.13                             0 65515 i
 *>                    10.10.0.12                             0 65515 i
 *    10.1.0.0/16      10.10.0.13                             0 65515 i
 *>                    10.10.0.12                             0 65515 i
 *    10.2.0.0/16      10.10.0.13                             0 65515 65520 65520 e
 *>                    10.10.0.12                             0 65515 65520 65520 e
 *    10.3.0.0/16      10.10.0.13                             0 65515 i
 *>                    10.10.0.12                             0 65515 i
 *    10.10.0.0/16     10.10.0.13                             0 65515 i
 *>                    10.10.0.12                             0 65515 i
 *>   10.11.11.0/24    0.0.0.0                  0         32768 i
 *    10.22.22.0/24    10.10.0.13                             0 65515 65520 65520 65772 e
 *>                    10.10.0.12                             0 65515 65520 65520 65772 e
```
Example 13-4: *BGP Table of Swe-Branch.*

Examples 13-5 and 13-6 show the BGP learned routes in the RIB of branch routers. Only the best ones are installed in the RIB.

```
Swe-Branch#sh ip route bgp | beg Gate
Gateway of last resort is 192.168.100.1 to network 0.0.0.0

      10.0.0.0/8 is variably subnetted, 12 subnets, 3 masks
B        10.0.0.0/16 [20/0] via 10.10.0.12, 03:19:27
B        10.1.0.0/16 [20/0] via 10.10.0.12, 03:19:27
B        10.2.0.0/16 [20/0] via 10.10.0.12, 00:50:10
B        10.3.0.0/16 [20/0] via 10.10.0.12, 03:19:27
B        10.10.0.0/16 [20/0] via 10.10.0.12, 03:19:27
B        10.22.22.0/24 [20/0] via 10.10.0.12, 01:38:43
```
Example 13-5: *Routing Table of Ger-Branch.*

```
Ger-Branch#sh ip route bgp | beg Gate
Gateway of last resort is 192.168.100.1 to network 0.0.0.0

      10.0.0.0/8 is variably subnetted, 12 subnets, 3 masks
B        10.0.0.0/16 [20/0] via 10.20.0.12, 01:39:02
B        10.1.0.0/16 [20/0] via 10.20.0.12, 01:39:02
B        10.2.0.0/16 [20/0] via 10.20.0.12, 00:51:28
B        10.3.0.0/16 [20/0] via 10.20.0.12, 01:39:02
B        10.11.11.0/24 [20/0] via 10.20.0.12, 01:39:02
B        10.20.0.0/16 [20/0] via 10.20.0.12, 01:39:02
Ger-Branch#
```

Example 13-6: *Routing Table of Swe-Branch.*

BGP Multipathing

Azure does BGP flow-based multipathing by default. Example 13-7 shows how we can deploy BGP multipathing in Swe-Branch. The command *maximum-paths eibgp 4* allow BGP to use four paths for flow-based load-balancing (applies to both iBGP and eBGP learned routes).

```
Swe-Branch#sh run | sec router bgp
router bgp 65771
 bgp log-neighbor-changes
 neighbor 10.10.0.12 remote-as 65515
 neighbor 10.10.0.12 ebgp-multihop 255
 neighbor 10.10.0.12 update-source Loopback11
 neighbor 10.10.0.13 remote-as 65515
 neighbor 10.10.0.13 ebgp-multihop 255
 neighbor 10.10.0.13 update-source Loopback11
 !
 address-family ipv4
  network 10.11.11.0 mask 255.255.255.0
  neighbor 10.10.0.12 activate
  neighbor 10.10.0.13 activate
  maximum-paths eibgp 4
 exit-address-family
Swe-Branch#
```

Example 13-7: *BGP Multi-Path Configuration of Swe-Branch.*

After enabling BGP Multipath, the routing table of Swe-Branch has two valid next-hop for all Azure CIDR ranges.

```
Swe-Branch#sh ip route bgp | beg Gate
Gateway of last resort is 192.168.100.1 to network 0.0.0.0

      10.0.0.0/8 is variably subnetted, 12 subnets, 3 masks
B        10.0.0.0/16 [20/0] via 10.10.0.13, 00:01:43
                     [20/0] via 10.10.0.12, 00:01:43
B        10.1.0.0/16 [20/0] via 10.10.0.13, 00:01:43
                     [20/0] via 10.10.0.12, 00:01:43
B        10.2.0.0/16 [20/0] via 10.10.0.13, 00:01:43
                     [20/0] via 10.10.0.12, 00:01:43
B        10.3.0.0/16 [20/0] via 10.10.0.13, 00:01:43
                     [20/0] via 10.10.0.12, 00:01:43
B        10.10.0.0/16 [20/0] via 10.10.0.13, 00:01:43
                      [20/0] via 10.10.0.12, 00:01:43
B        10.22.22.0/24 [20/0] via 10.10.0.13, 00:01:43
                       [20/0] via 10.10.0.12, 00:01:43
```

Example 13-8: *Swe-Branch's Routing Table After Enabling BGP Multipathing.*

Data Plane verification

Intra-Region Branch-to-VNet

This section verifies that we have regional IP connections between VPN-Site and VNets.

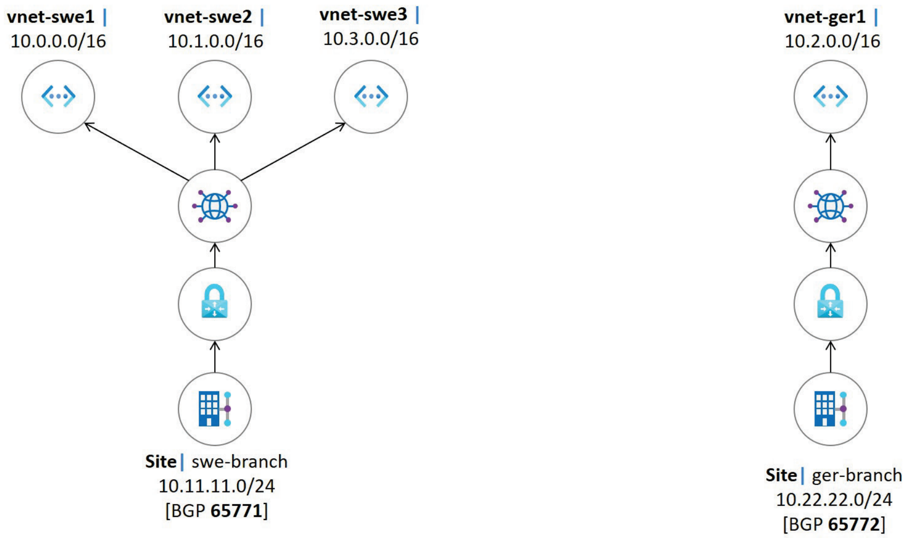


Figure 13-14: *Intra-Region Branch-to-VNet Connection.*

```
Swe-Branch#ping 10.0.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 11/11/12 ms
```

Example 13-9: *Ping from Swe-Branch to 10.0.0.4 (vm-swe1 in vnet-swe1).*

```
Swe-Branch#ping 10.1.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 11/12/14 ms
```

Example 13-10: *Ping from Swe-Branch to 10.1.0.4 (vm-swe2 in vnet-swe2).*

```
Swe-Branch#ping 10.3.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 11/11/13 ms
```

Example 13-11: *Ping from Swe-Branch to 10.3.0.4 (vm-swe3 in vnet-swe3).*

```
Ger-Branch#ping 10.2.0.4 source 10.22.22.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.22.22.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 27/28/30 ms
```

Example 13-12: *Ping from Ger-Branch to 10.2.0.4 (vm-ger1 in vnet-ger1).*

Inter-Region Branch-to-VNet (Branch-to-Hub-to-Hub-Vnet)

This section verifies that we have an Inter-Region IP connection between VPN-Site and VNets. The figure below illustrates this from the ger-branch perspective.

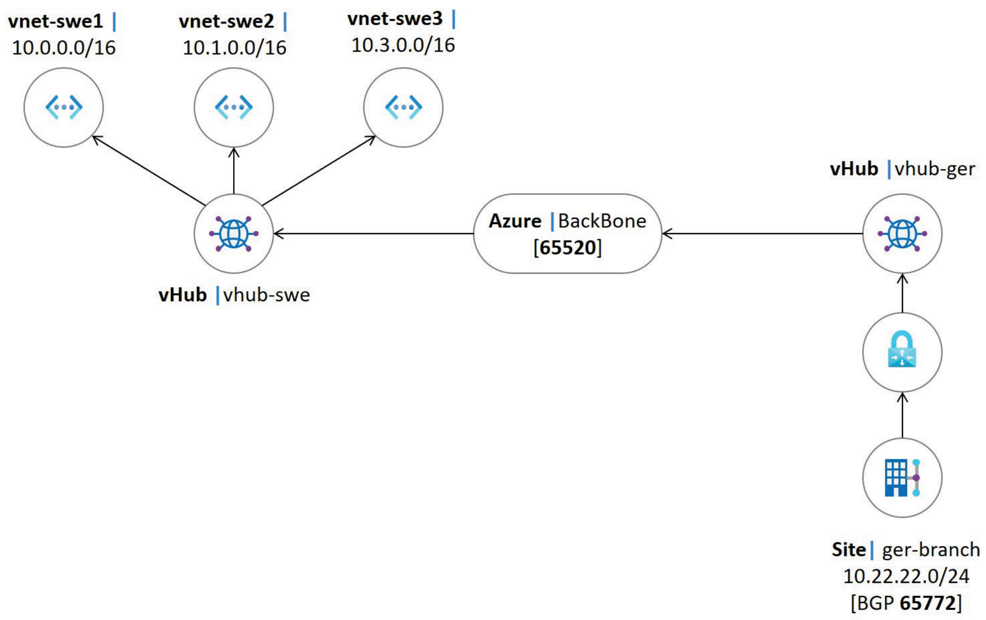


Figure 13-15: *Inter-Region Branch-to-VNet Connection.*

```
Ger-Branch#ping 10.0.0.4 source 10.22.22.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.22.22.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 52/54/57 ms
```

Example 13-13: *Ping from Ger-Branch to 10.0.0.4 (vm-swe1 in vnet-swe1).*

```
Ger-Branch#ping 10.1.0.4 source 10.22.22.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.22.22.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 51/51/52 ms
```

Example 13-14: *Ping from Ger-Branch to 10.1.0.4 (vm-swe2 in vnet-swe2).*

```
Ger-Branch#ping 10.3.0.4 source 10.22.22.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.22.22.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 55/55/56 ms
```

Example 13-15: *Ping from Ger-Branch to 10.3.0.4 (vm-swe3 in vnet-swe3).*

```
Swe-Branch#ping 10.2.0.4 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.0.4, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 38/39/41 ms
```

Example 13-16: *Ping from Swe-Branch to 10.2.0.4 (vm-ger1 in vnet- ger1).*

Inter-Region Branch-to-Branch (Branch-to-Hub-to-Hub-Branch)

This section verifies that we have an Inter-Region Branch-to-Branch IP connection.

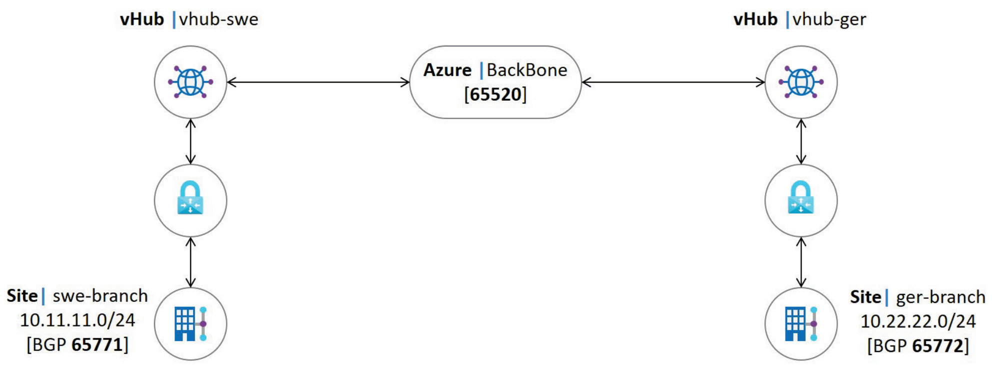


Figure 13-16: *Inter-Region Branch-to-Branch Connection.*

```
Swe-Branch#ping 10.22.22.1 source 10.11.11.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.22.22.1, timeout is 2 seconds:
Packet sent with a source address of 10.11.11.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 65/67/68 ms
```

Example 13-17: *Ping from Swe-Branch to 10.22.22.1 (Ger-Branch).*

```
Ger-Branch#ping 10.11.11.1 source 10.22.22.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.11.11.1, timeout is 2 seconds:
Packet sent with a source address of 10.22.22.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 65/68/77 ms
```

Example 13-18: *Ping from Ger-Branch to 10.11.11.1 (Swe-Branch).*

Intra-Region VNet-to-VNet

This section verifies that we have a Regional VNet-to-VNet IP connection.

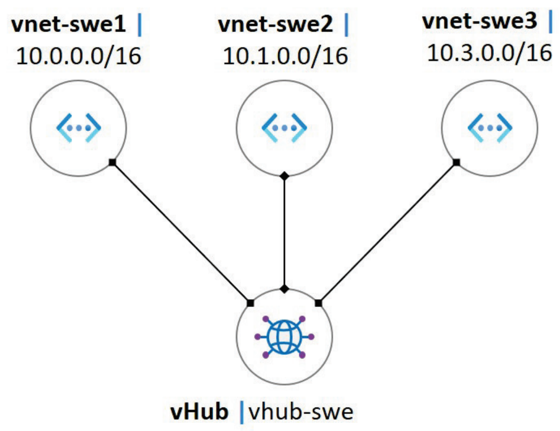


Figure 13-17: *Intra-Region Vnet-to-Vnet Connection.*

```
azureuser@vm-swe1:~$ ping -c 3 10.1.0.4
PING 10.1.0.4 (10.1.0.4) 56(84) bytes of data.
64 bytes from 10.1.0.4: icmp_seq=1 ttl=63 time=3.24 ms
64 bytes from 10.1.0.4: icmp_seq=2 ttl=63 time=2.28 ms
64 bytes from 10.1.0.4: icmp_seq=3 ttl=63 time=2.08 ms

--- 10.1.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.081/2.534/3.243/0.507 ms
```

Example 13-19: *Ping from vm-swe1 to 10.1.0.4 (vm-swe2 in vnet-swe2).*

```
azureuser@vm-swe1:~$ ping -c 3 10.3.0.4
PING 10.3.0.4 (10.3.0.4) 56(84) bytes of data.
64 bytes from 10.3.0.4: icmp_seq=1 ttl=63 time=2.84 ms
64 bytes from 10.3.0.4: icmp_seq=2 ttl=63 time=2.54 ms
64 bytes from 10.3.0.4: icmp_seq=3 ttl=63 time=2.99 ms

--- 10.3.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 2.541/2.792/2.993/0.188 ms
```

Example 13-20: *Ping from vm-swe1 to 10.3.0.4 (vm-swe3 in vnet-swe3).*

Inter-Region VNet-to-VNet (VNet-to-Hub-to-Hub-Vnet)

This section verifies that we have an Inter-Region VNet-to-VNet IP connection.

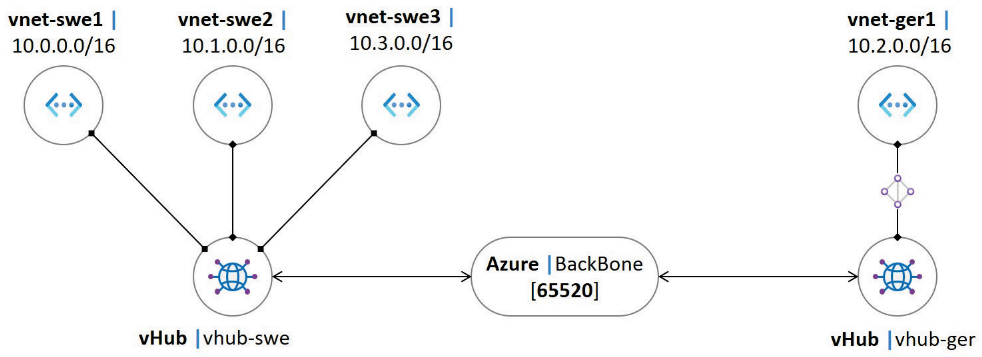


Figure 13-18: *Inter-Region Vnet-to-Vnet Connection.*

```
azureuser@vm-swe1:~$ ping -c 3 10.2.0.4
PING 10.2.0.4 (10.2.0.4) 56(84) bytes of data.
64 bytes from 10.2.0.4: icmp_seq=1 ttl=62 time=29.3 ms
64 bytes from 10.2.0.4: icmp_seq=2 ttl=62 time=28.0 ms
64 bytes from 10.2.0.4: icmp_seq=3 ttl=62 time=27.6 ms

--- 10.2.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 27.638/28.298/29.289/0.713 ms
```

Example 13-20: *Ping from vm-swe1 to 10.2.0.4 (vm-ger1 in vnet- ger1).*

References

[1] Global transit network architecture and Virtual WAN
https://docs.microsoft.com/en-us/azure/virtual-wan/virtual-wan-global-transit-network-architecture

[2] Virtual hub routing preference (Preview)
https://docs.microsoft.com/en-us/azure/virtual-wan/about-virtual-hub-routing-preference

Chapter 13 - Appendix A: Ger-Branch Configuration

```
Ger-Branch#sh run
Building configuration...
!
<snipped>
!
version 16.6
<snipped>
!
redundancy
!
crypto ikev2 proposal Azure-Ikev2-Proposal
 encryption aes-cbc-256
 integrity sha1
 group 2
!
crypto ikev2 policy Azure-Ikev2-Policy
 match address local 192.168.100.19
 proposal Azure-Ikev2-Proposal
!
crypto ikev2 keyring cn-hub1-dc1-keyring
 peer 20.52.212.49
  address 20.52.212.49
  pre-shared-key psk-ger-vpn
 !
!
crypto ikev2 keyring cn-hub1-dc1-keyring-2
 peer 20.52.212.40
  address 20.52.212.40
  pre-shared-key psk-ger-vpn
 !
!
!
crypto ikev2 profile Azure-Ikev2-Profile
 match address local 192.168.100.19
 match identity remote address 20.52.212.49 255.255.255.255
 authentication remote pre-share
 authentication local pre-share
 keyring local cn-hub1-dc1-keyring
 lifetime 28800
 dpd 10 5 on-demand
!
crypto ikev2 profile Azure-Ikev2-Profile-2
 match address local 192.168.100.19
 match identity remote address 20.52.212.40 255.255.255.255
 authentication remote pre-share
 authentication local pre-share
 keyring local cn-hub1-dc1-keyring-2
 lifetime 28800
 dpd 10 5 on-demand
!
```

```
!
!
crypto ipsec transform-set Azure-TransformSet esp-aes 256 esp-sha256-hmac
 mode tunnel
!
crypto ipsec profile Azure-IPsecProfile
 set security-association lifetime kilobytes 102400000
 set transform-set Azure-TransformSet
 set ikev2-profile Azure-Ikev2-Profile
!
crypto ipsec profile Azure-IPsecProfile-2
 set security-association lifetime kilobytes 102400000
 set transform-set Azure-TransformSet
 set ikev2-profile Azure-Ikev2-Profile-2
!
interface Loopback22
 description ** eBGP Azure-Hub-swe **
 ip address 192.168.22.1 255.255.255.255
!
interface Loopback2222
 description ** LocalSnet **
 ip address 10.22.22.1 255.255.255.0
!
interface Tunnel11
 ip address 169.254.0.1 255.255.255.255
 ip tcp adjust-mss 1350
 tunnel source 192.168.100.19
 tunnel mode ipsec ipv4
 tunnel destination 20.52.212.49
 tunnel protection ipsec profile Azure-IPsecProfile
!
interface Tunnel12
 ip address 169.254.0.2 255.255.255.255
 ip tcp adjust-mss 1350
 tunnel source 192.168.100.19
 tunnel mode ipsec ipv4
 tunnel destination 20.52.212.40
 tunnel protection ipsec profile Azure-IPsecProfile-2
!
interface GigabitEthernet1
 ip address dhcp
 negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet2
 no ip address
 shutdown
 negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet3
 ip address 10.2.3.2 255.255.255.0
 negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet4
 no ip address
 shutdown
 negotiation auto
 no mop enabled
 no mop sysid
!
router bgp 65772
 bgp log-neighbor-changes
 neighbor 10.20.0.12 remote-as 65515
 neighbor 10.20.0.12 ebgp-multihop 255
 neighbor 10.20.0.12 update-source Loopback22
 neighbor 10.20.0.13 remote-as 65515
 neighbor 10.20.0.13 ebgp-multihop 255
 neighbor 10.20.0.13 update-source Loopback22
 !
 address-family ipv4
  network 10.22.22.0 mask 255.255.255.0
  neighbor 10.20.0.12 activate
  neighbor 10.20.0.13 activate
  maximum-paths eibgp 4
 exit-address-family
!
<snipped>
!
ip http client source-interface GigabitEthernet1
ip route 10.20.0.12 255.255.255.255 Tunnel11
ip route 10.20.0.13 255.255.255.255 Tunnel12
!
<snipped>
```

Chapter 14: ExpressRoute

Introduction

ExpressRoute (ER) is an Azure service that offers a logical circuit between *Customer Edge (CE)* router(s) and the pair of *Microsoft Enterprise Edge (MSEE)* devices. MSEEs are in one of the Azure selected *Co-locations (Colo) - Carrier Neutral Facilities* having a connection straight to the Microsoft Backbone network and *ExpressRoute Provider (ERP)*. We can deploy an ExpressRoute circuit between *Customer Edge (CE)* and MSEE directly (*ExpressRouteDirect*) if the CE device is in Colo. If we don't have existence in Colo, we can use one of the Azure ExpressRoute Partner's infrastructures as a transport network between on-prem CE and MSEE. In this model, ERP provides a layer 2 circuit (point-to-point) between the CE-facing interface and to MSEE-facing port through its *ExpressRoute Provider Edge (ERPE)* devices. We use this circuit to establish BGP sessions between CE(s) and MSEEs. We can also use an any-to-any (IPVPN) solution where we create eBGP sessions with ERPE devices over an MPLS network. The *ExpressRoute Provider* point-to-point model is the main focus of this chapter. The other peering model, Microsoft peering, offers connection to Microsoft services (Office 365 and Dynamics 365).

Figure 14-1 illustrates our ExpressRoute setup having two ExpressRoute customers. Beetle Co. uses the ExpressRoute (*Standard SKU*) for connecting the Stockholm office to vnet-beetle in the Germany West Central region. Bailey Co., in turn, uses the ExpressRoute (*Local SKU*) for accessing vnet-bailey in Sweden Central region. They both use the same Colo in Stockholm. The Local SKU allows access to resources in the region closest to Colo. The Standard SKU, in turn, gives access to regions within the same geographical area, meaning that Beetle Co. is allowed to connect to resources deployed in Europe. Bailey Co., in turn, has access to VNets only in the Sweden Central region. The notable benefit of using a Local SKU is that Microsoft doesn't charge you for the egress data. Note that the ingress fees are always free for both SKUs. The third SKU, *Premium SKU,* gives global access to Azure regions. It also increases the routing table limit to 10000 routes with private peering. Besides, by selecting Premium SKU, you can add 20 – 100 VNet links to the ExpressRoute circuit (depending on the circuit size). Note that you can change between Standard and Premium but not from Premium/Standard to Local. The Layer 1 physical structure for the ExpressRoute circuit is simple. On the Microsoft side, there are two active MSEEs, which both have a fiberoptic connection to ERPE devices. In our example, MSEE1 is connected to ERPE1, and MSEE2 is connected to ERPE2.

Beetle Co. has a single-homed redundant connection from CE-Beetle to ERPE1 (via e0/0.200) and ERPE2 (via e0/1.200). The reason for using sub-interfaces in two physical interfaces is that we can a) add a VLAN-id to Ethernet headers and b) use the same VLAN-Id for both connections (we only get one C-Tag). Note that you can't associate the same VLAN-Id to two sub-interfaces created under the same physical interface. Beetle Co. has a dual-homed redundant connection (CE-Bailey1 to ERPE1 and CE-Bailey2 to ERPE2.

Note! Getting an SLA for your ExpressRoute circuit, you need two active BGP sessions with Azure MSEEs. Besides, Azure accepts 200 prefixes per BGP session (applies to Private and Microsoft peering). MSEE tears down the BGP peering if the maximum prefix limit is exceeded. The maximum prefixes for all of your BGP peering are 4000 with a Standard circuit and 10 000 with a Premium circuit add-on.

The layer 2 logical connection over the physical infrastructure uses VLAN tagging. During the ExpressRoute circuit deploying process, we define a VLAN-Id, which we set to the Ethernet header's 802.1Q tag filed when sending IP packets towards ERPE devices. Both our example customers use the same VLAN-id 200. The customer VLAN-Id is called *C-Tag*. Connections from the ERPE 's customer-facing interface to MSEE devise use *Q-in-Q tunneling* specified in 802.1ad standard. When we deploy an ExpressRoute circuit, Azure generates a circuit-specific *S-Tag*. It defines the outer VLAN-Id associated with the ExpressRoute circuit. In our example, Azure has allocated S-Tag 2 to Beetle and S-Tag 3 to Bailey. The ERPE devices are only aware of S-Tags. For example, when CE-Beetle starts the BGP initiation process with MSEE1, it uses the C-Tag 200 (VLAN-Id 200) on the 802.1Q Tag field in the Ethernet header of the BGP Open message. When ERPE1 receives the message from the CE-Beetle, it doesn't check the C-Tag but adds an S-Tag (VLAN-Id 2) to the Ethernet 802.1ad tag field and forwards the dual-tagged message to MSEE1 based on the destination Mac address. When MSEE1 sends its BGP messages to CE Beetle, it uses C-tag as an inner VLAN-Id and S-Tag as an outer VLAN-Id in the Ethernet header. When the ERPE1 receives the messages, it does the MAC address lookup from the VLAN 2 Mac-address table. Then, before forwarding the message to CE-Beetle, it removes the outer S-Tag (VLAN-Id 2), leaving the inner C-Tag (VLAN-Id 200) untouchable. This is the simplified operation of Q-in-Q tunneling.

The next question is how the ExpressRoute Provider knows which devices and configurations are needed for ER circuit. First, ERPs have their *ExpressRoute management subscription*. When we deploy an ExpressRoute circuit, Azure creates a shadow resource called *expressRouteCrossConnections* under the ERP's ER management subscription. Besides, Azure creates a circuit-specific 128 bits *Service-*

Key (S-Key). The S-Key is metadata associated with the circuit describing the circuit-specific information such as Location, MSEE physical interfaces, S-Tag (Q-in-Q VLAN-Id), Bandwidth, and the provision state of the ExpressRoute circuit. Next, after deploying the connection, the customer sends the S-Key to the provider, or the provider can use a REST API call to get the S-Key based on the ER circuit name. Then, using the S-Key, ExpressRoute Provider uses a REST API GET command to get the data associated with the S-Key. After configuring ERPE devices, the provider enables the circuit with a REST API call using a PUT command, which sets the *serviceProviderProvisioningState* to Provisioned. After that, the ExpressRoute circuit is enabled, and the customer can configure BGP peering with MSEEs. We will take a closer look at the whole process in the coming sections.

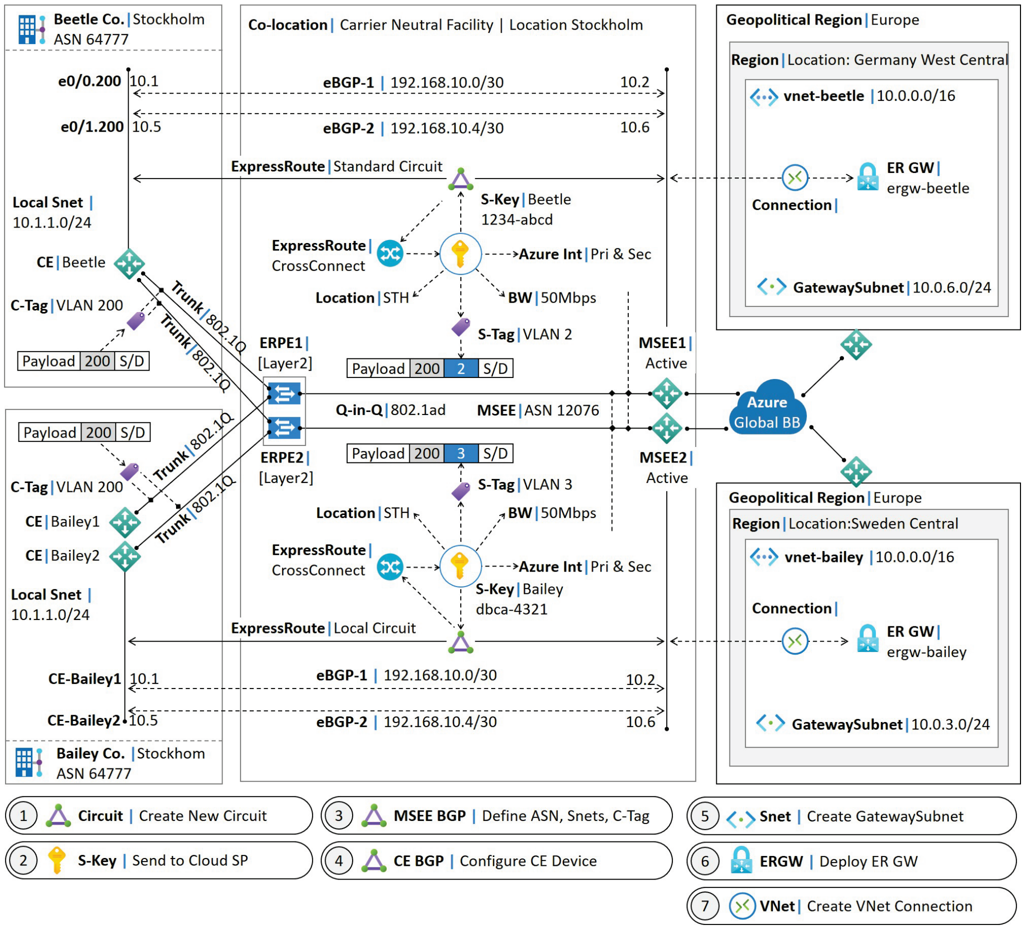

Beetle Co. | Stockholm
ASN 64777
e0/0.200 10.1
e0/1.200 10.5
Local Snet | 10.1.1.0/24
CE | Beetle
C-Tag | VLAN 200
Payload 200 S/D
Co-location | Carrier Neutral Facility | Location Stockholm
eBGP-1 | 192.168.10.0/30 10.2
eBGP-2 | 192.168.10.4/30 10.6
ExpressRoute | Standard Circuit
S-Key | Beetle 1234-abcd
ExpressRoute | CrossConnect
Azure Int | Pri & Sec
Location | STH
BW | 50Mbps
S-Tag | VLAN 2
Trunk | 802.1Q
ERPE1 | [Layer2]
Payload 200 2 S/D
MSEE1 | Active
Q-in-Q | 802.1ad
MSEE | ASN 12076
ERPE2 | [Layer2]
Payload 200 3 S/D
MSEE2 | Active
S-Tag | VLAN 3
Location | STH
BW | 50Mbps
ExpressRoute | CrossConnect
Azure Int | Pri & Sec
S-Key | Bailey dbca-4321
ExpressRoute | Local Circuit
Payload 200 S/D
C-Tag | VLAN 200
CE | Bailey1
CE | Bailey2
Local Snet | 10.1.1.0/24
CE-Bailey1 10.1
CE-Bailey2 10.5
eBGP-1 | 192.168.10.0/30 10.2
eBGP-2 | 192.168.10.4/30 10.6
Bailey Co. | Stockhom
ASN 64777
Geopolitical Region | Europe
Region | Location: Germany West Central
vnet-beetle | 10.0.0.0/16
ER GW | ergw-beetle
Connection |
GatewaySubnet | 10.0.6.0/24
Azure Global BB
Geopolitical Region | Europe
Region | Location:Sweden Central
vnet-bailey | 10.0.0.0/16
Connection |
ER GW | ergw-bailey
GatewaySubnet | 10.0.3.0/24
1 Circuit | Create New Circuit
2 S-Key | Send to Cloud SP
3 MSEE BGP | Define ASN, Snets, C-Tag
4 CE BGP | Configure CE Device
5 Snet | Create GatewaySubnet
6 ERGW | Deploy ER GW
7 VNet | Create VNet Connection

Figure 14-1: *ExpressRoute Example Topology.*

Create a New ExpressRoute Circuit

The figure below illustrates the simplified view of tasks related to ExpressRoute Circuit deployment steps: (1) Customer creates a new ExpressRoute circuit. (2) Microsoft generates the circuit-specific Service-Key. (3) Customer sends the S-key to ExpressRoute Provider. (4) Provider configures ERPE devices based on the S-key metadata and sets the circuit state to Provisioned.

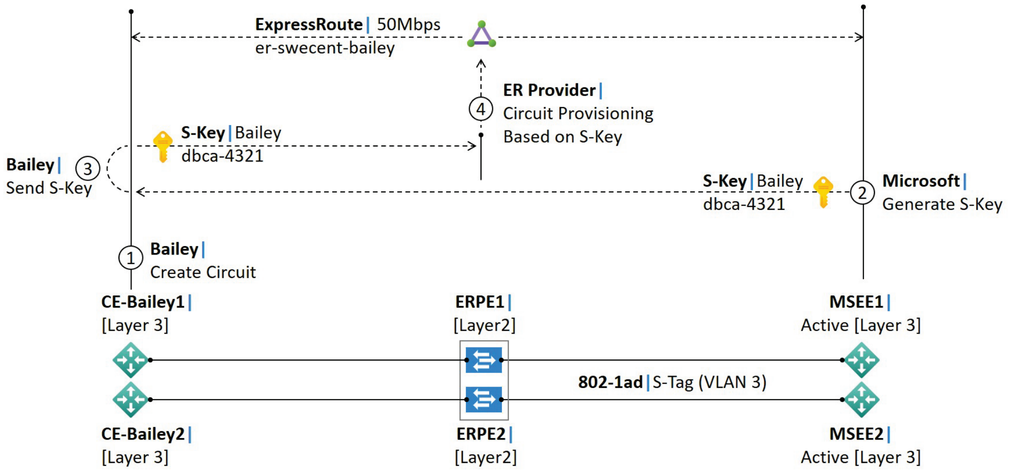
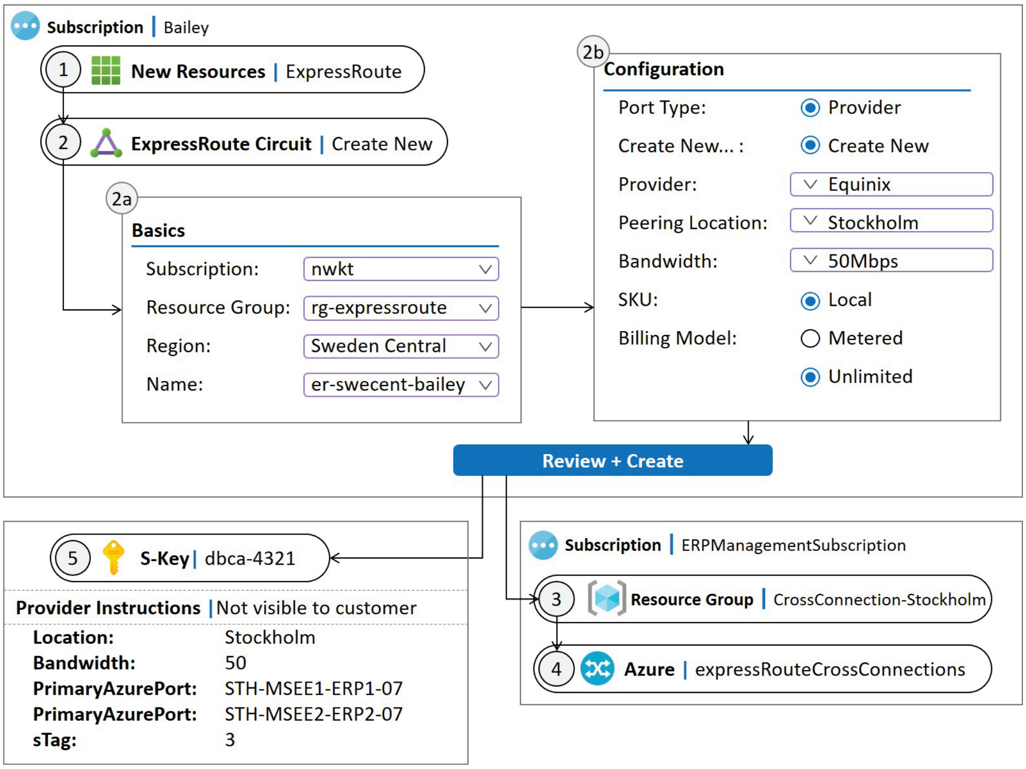


Figure 14-2: *ExpressRoute Circuit Deployment Processes.*

Figure 14-3 illustrates the configuration ExpressRoute circuit deploying process. Select the Create a resource from the Azure portal Home view. Then type the *ExpressRoute* on the search field. Next, select the *Microsoft ExpressRoute* option and click the *Create* button. Select your Subscription and Resource Group from the drop-down menus. Then, on the *Configuration* tab, select the *Provider* option as a *Port Type* (the other option, *Direct*, is not shown in the figure). Select the *Create New* radio button and choose the *provider*, *location*, and *bandwidth* from the drop-down menus. Then, select *Local SKU* (other options are *Standard* and *Premium*). The *Metered* billing model has a fixed, monthly base price besides the egress fee. The base price is related to circuit bandwidth (faster circuit = higher base price). The Unlimited model, in turn, has a fixed monthly fee, including an unlimited Inbound/Outbound data transfer fee (the faster circuit = higher fixed price). In our example, we have selected the Unlimited billing data plan. Be aware that you can change the billing model from Metered to Unlimited but not from Unlimited to Metered.

During the deployment process, Azure creates a shadow resource *expressRouteCrossConnection* for ER circuit under Azure-defined ERP Resource Group *CrossConnection-Stockholm*. I will explain the shadow resource usage later in the REST API section. Besides, Azure deploys circuit-specific, *Service-Key (S-Key)*. The S-Key Identifier (HEX value) is visible for a customer, while the related metadata information is not. The metadata includes all the necessary circuit-specific parameters in order to ERP can configure parameters to their ERPE devices.


Subscription | Bailey
1 New Resources | ExpressRoute
2 ExpressRoute Circuit | Create New
2a
2b
Review + Create
Subscription | ERPManagementSubscription
3 Resource Group | CrossConnection-Stockholm
4 Azure | expressRouteCrossConnections
5 S-Key | dbca-4321

Basics

Subscription:	nwkt
Resource Group:	rg-expressroute
Region:	Sweden Central
Name:	er-swecent-bailey

Configuration

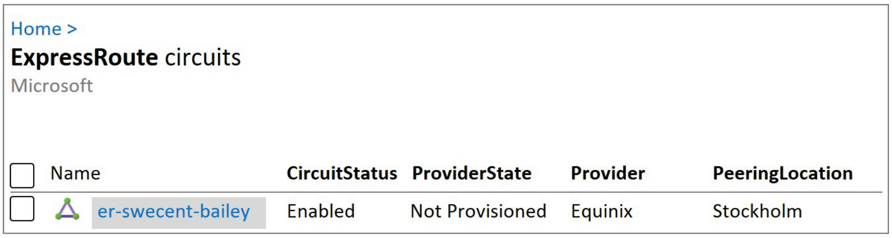
Port Type:	[✓ Provider]
Create New... :	[✓ Create New]
Provider:	Equinix
Peering Location:	Stockholm
Bandwidth:	50Mbps
SKU:	[✓ Local]
Billing Model:	Metered [✓ Unlimited]

Provider Instructions | Not visible to customer

Location:	Stockholm
Bandwidth:	50
PrimaryAzurePort:	STH-MSEE1-ERP1-07
PrimaryAzurePort:	STH-MSEE2-ERP2-07
sTag:	3

Figure 14-3: *Deploy a New ExpressRoute Circuit.*

When the Circuit deployment process is ready, navigate to the Azure portal home window and type the ExpressRoute into the search field at the top of the page. Figure 14-4 shows that we have successfully created circuit *er-swecent-bailey* (Enabled), but ER Provider hasn't yet provisioned it. Next, click the er-swecent-bailey hyperlink.



Home >

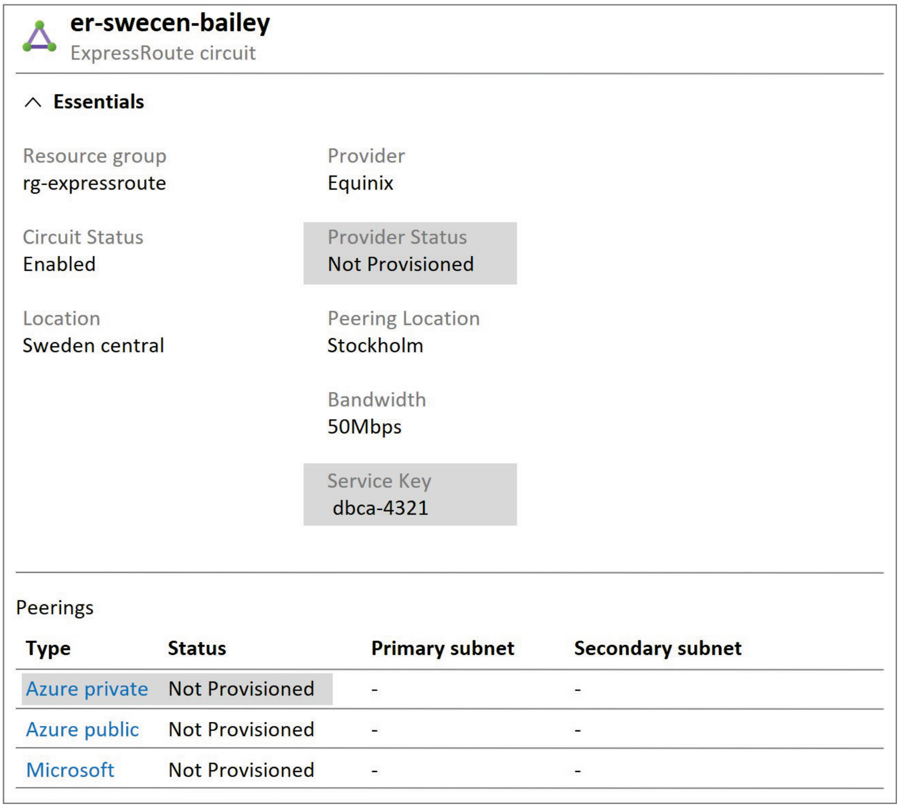
ExpressRoute circuits

Microsoft

	Name	CircuitStatus	ProviderState	Provider	PeeringLocation
☐	er-swecent-bailey	Enabled	Not Provisioned	Equinix	Stockholm

Figure 14-4: *Deployed ExpressRoute Circuit.*

You can find the S-key from the *Essential* section at the *ExpressRoute circuit* window (figure 14-5). Copy the S-key and send it to ERP. Note that both *Provider Status* (L2 Circuit) and *Peering Status* (eBGP peering) are still shown as *Not Provisioned*. Copy the S-key and send it to ERP.



er-swecen-bailey

ExpressRoute circuit

Essentials

Resource group rg-expressroute	Provider Equinix
Circuit Status Enabled	Provider Status Not Provisioned
Location Sweden central	Peering Location Stockholm
	Bandwidth 50Mbps
	Service Key dbca-4321

Peerings

Type	Status	Primary subnet	Secondary subnet
Azure private	Not Provisioned	-	-
Azure public	Not Provisioned	-	-
Microsoft	Not Provisioned	-	-

Figure 14-5: *Deployed ExpressRoute Circuit.*

ERP Circuit Provision

When ExpressRoute Provider gets the customer S-Key, they make a REST API call using the HTTP GET command to get the metadata. Based on the HTTP response, ERP knows the location (Stockholm), circuit Bandwidth (50Mbps), physical interfaces (STHML-MSEE1-ERPE1-07 & STHML-MSEE2-ERPE2-07), and the S-Tag (Q-in-Q VLAN-Id).

```
GET /subscriptions/ ERPManagementSubscription/resourceGroups/CrossConnection-
Stockholm/providers/Microsoft.Network/expressRouteCrossConnections/dbca-4321?api-
version=2018-02-01 HTTP/1.1
------<snipped for brevity>------
HTTP/1.1 200 OK
------<snipped for brevity>------
{
  "name": " dbca-4321",
  "id": "/subscriptions/ ERPManagementSubscription /resourceGroups/ CrossConnection-
Stockholm /providers/Microsoft.Network/expressRouteCrossConnections/dbca-4321",
  "etag": "W/\"xxxx\"",
  "type": "Microsoft.Network/expressRouteCrossConnections",
  "location": "Stockholm",
  "properties": {
    "provisioningState": "Succeeded",
    "expressRouteCircuit": {
      "id": "/subscriptions/Bailey/resourceGroups/rg-expressroute/
             providers/Microsoft.Network/expressRouteCircuits/er-swecent-bailey"
    },
    "peeringLocation": "Stockholm",
    "bandwidthInMbps": 50,
    "serviceProviderProvisioningState": "NotProvisioned",
    "primaryAzurePort": "STHML-MSEE1-ERPE1-07 ",
    "secondaryAzurePort": " STHML-MSEE2-ERPE2-07 ",
    "sTag": 3,
    "peerings": []
  }
}
```

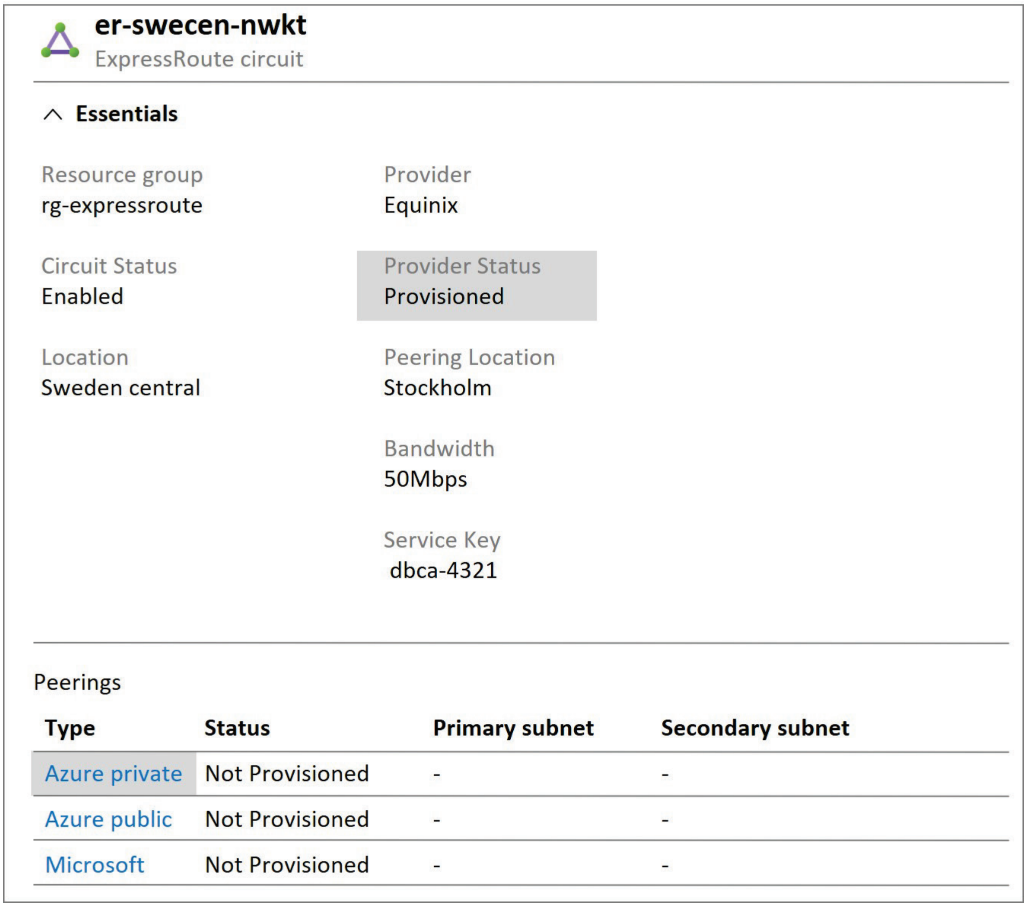
Example 14-1: *REST API: Request an S-Key Metadata.*

After ERP has configured its ERPEs, it can enable the ExpressRoute circuit using the REST API call, which sets the value of the object *serviceProviderProvisionState* to Provisioned. The HTTP response shows a successful result.

```
PUT /subscriptions/ ERPManagementSubscription /resourceGroups/ CrossConnection-
Stockholm /providers/Microsoft.Network/expressRouteCrossConnections/ dbca-4321?api-
version=2018-02-01 HTTP/1.1
------<snipped for brevity>------
{
  "properties": {
    "serviceProviderProvisioningState": "Provisioned",
    "peeringLocation": "Stockholm",
    "expressRouteCircuit": {
      "id": "/subscriptions/ ERPManagementSubscription /resourceGroups/
CrossConnection-Stockholm /providers/Microsoft.Network/ expressRouteCircuits/ dbca-
4321"
    },
    "bandwidthInMbps": 50
  },
  "location": "Stockholm"
}

------<snipped for brevity>------
HTTP/1.1 200 OK
------<snipped for brevity>------

{
  "name": "dbca-4321",
  "id": "id": "/subscriptions/ERPManagementSubscription
/resourceGroups/ CrossConnection-Stockholm /providers/Microsoft.Network/
expressRouteCircuits/ dbca-4321"
,
  "etag": "W/\"xxx\"",
  "type": "Microsoft.Network/expressRouteCrossConnections",
  "location": "Stockholm",
  "properties": {
    "provisioningState": "Updating",
    "expressRouteCircuit": {
      "id": "/subscriptions/Bailey/resourceGroups/rg-expressroute/providers
/Microsoft.Network/expressRouteCircuits/dbca-4321"
    },
    "peeringLocation": "Stockholm",
    "bandwidthInMbps": 50,
    "serviceProviderProvisioningState": "Provisioned",
    "primaryAzurePort": "",
    "secondaryAzurePort": "",
    "sTag": 0,
    "peerings": []
  }
}
```

Example 14-2: *REST API: Circuit Provisioning.*

After ERP has configured ERPE devices and enabled the circuit, the Provider Status in the Azure portal ExpressRoute circuit page is updated to *Provisioned*. Now we have a Layer 2 circuit between CE and MSEE over ERPE devices, which we use for eBGP peering between CE and MSEE. Select the *Azure private* hyperlink from the *Peerings* section to start the BGP configuration.



er-swecen-nwkt
ExpressRoute circuit

^ **Essentials**

Resource group rg-expressroute	Provider Equinix
Circuit Status Enabled	Provider Status Provisioned
Location Sweden central	Peering Location Stockholm
	Bandwidth 50Mbps
	Service Key dbca-4321

Peerings

Type	Status	Primary subnet	Secondary subnet
Azure private	Not Provisioned	-	-
Azure public	Not Provisioned	-	-
Microsoft	Not Provisioned	-	-

Figure 14-6: *Configure BGP Peering.*

Configure eBGP Peering with MSEE

After ER Provider has provisioned the circuit, we can configure eBGP peering between CE and MSEE devices. The ER circuit provides a broadcast domain that allows devices to resolve IP/MAC address binding of remote devices using ARP messages.

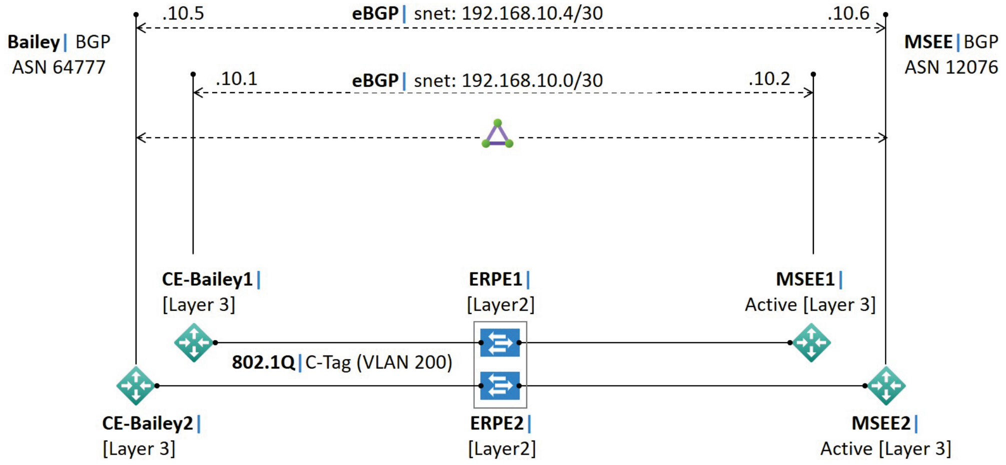
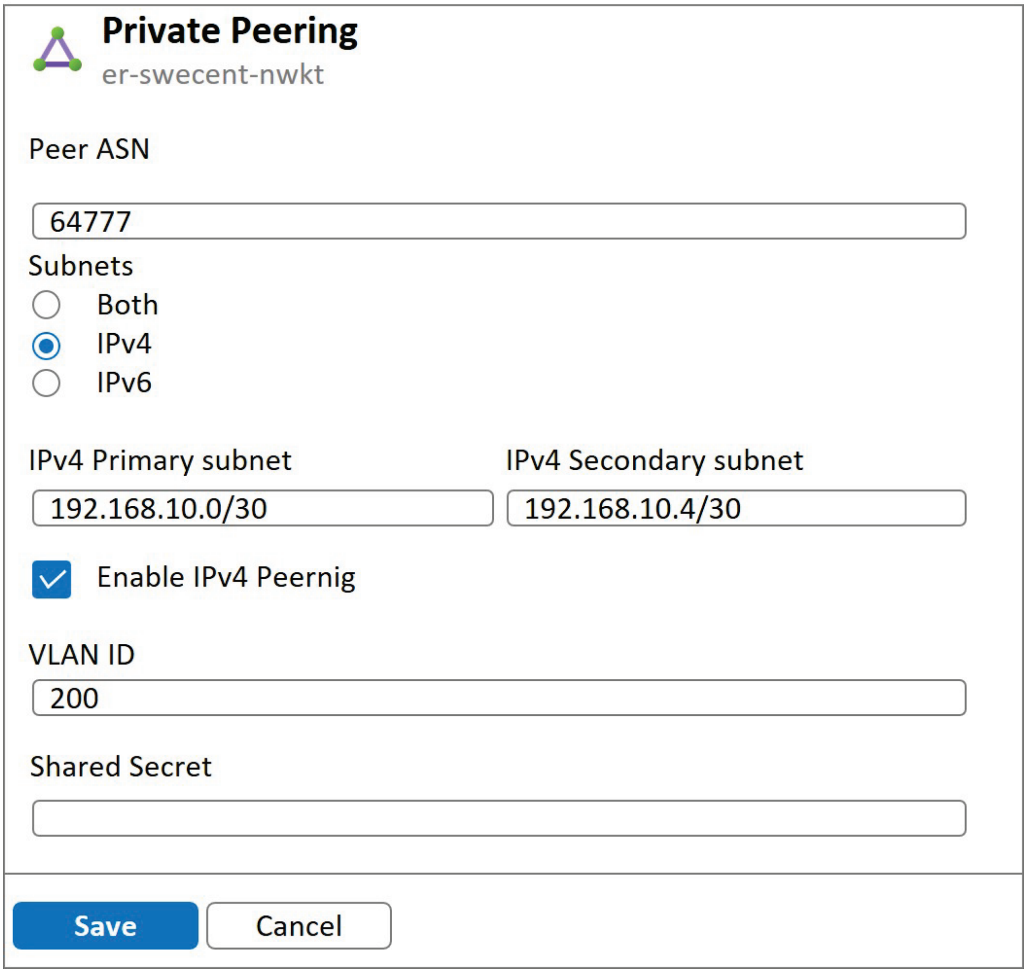


Figure 14-7: *Configure BGP Peering.*

Define your BGP ASN. We can use either public or private ASNs (both 16 and 32 bits ASNs are supported) with a private peering. In this example, we create only IPv4 peering, and you can select the IPv4 option under Subnets. Define your peering subnet (use the subnet mask /30). Define the VLAN-Id, which is the C-Tag. Remember that this VLAN is not visible to your ER Provider. If you want to authenticate BGP messages, fill in your MD5 Shared Secret password. Be aware that the shared secret key field doesn't show the key if you proceed and then navigate back to this view. Assign the first available IP address from the subnet range to your CPE because Microsoft assigns the second usable IP address to its MSEEs. In our example, we bind IP address 192.168.10.1 to CE-Bailey1 and 192.168.10.5 to CE-Bailey2. Remember that though we can use the subnet addresses (.0 & .4) and the broadcast addresses (.3 & .7), Microsoft uses the standard recommendation and doesn't use the for eBGP peerings.



Private Peering
er-swecent-nwkt

Peer ASN
64777

Subnets
- [ ] Both
- [✓] IPv4
- [ ] IPv6

IPv4 Primary subnet
192.168.10.0/30

IPv4 Secondary subnet
192.168.10.4/30

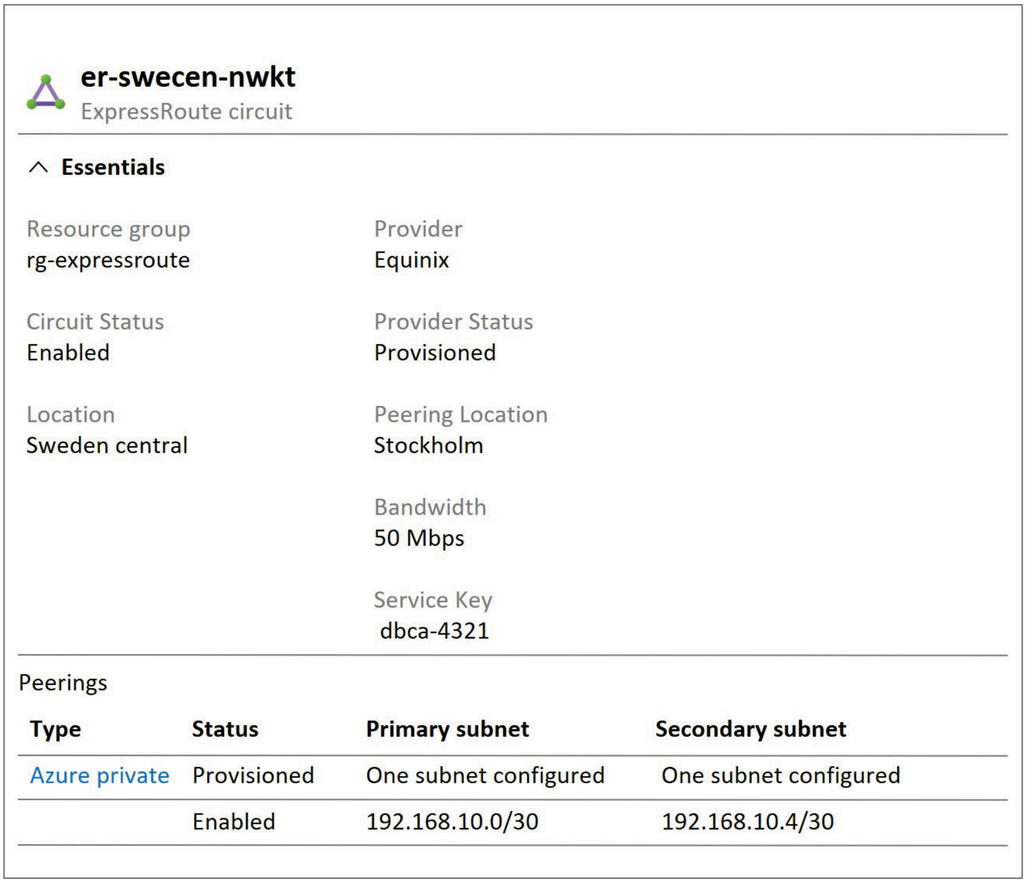
[✓] Enable IPv4 Peernig

VLAN ID
200

Shared Secret

Figure 14-8: *Configure BGP Peering.*

After saving the BGP setting, the peering status changes to *Provisioned*.



er-swecen-nwkt
ExpressRoute circuit

Essentials

Resource group rg-expressroute	Provider Equinix
Circuit Status Enabled	Provider Status Provisioned
Location Sweden central	Peering Location Stockholm
	Bandwidth 50 Mbps
	Service Key dbca-4321

Peerings

Type	Status	Primary subnet	Secondary subnet
Azure private	Provisioned	One subnet configured	One subnet configured
	Enabled	192.168.10.0/30	192.168.10.4/30

Figure 14-9: *Configure BGP Peering - Verification.*

Connect VNet to ExpressRoute Circuit

As the last step, we connect our VNet to the ExpressRoute circuit using an *ExpressRoute Gateway*. It is a three steps process where we first create a *GatewaySubnet*, which is used for ExpressRoute GW IP addressing. Then we deploy an ExpressRoute Gateway. As the last step, we create a VNet connection between the ExpressRoute gateway and ExpressRoute circuit.

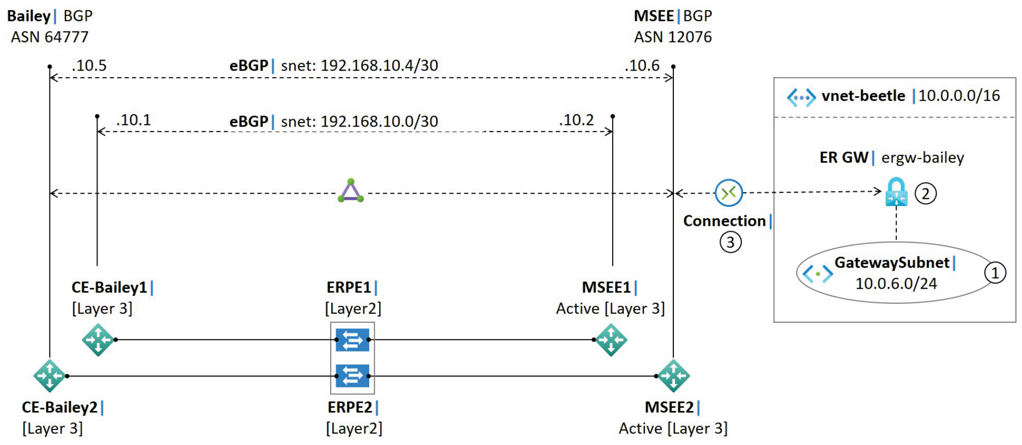
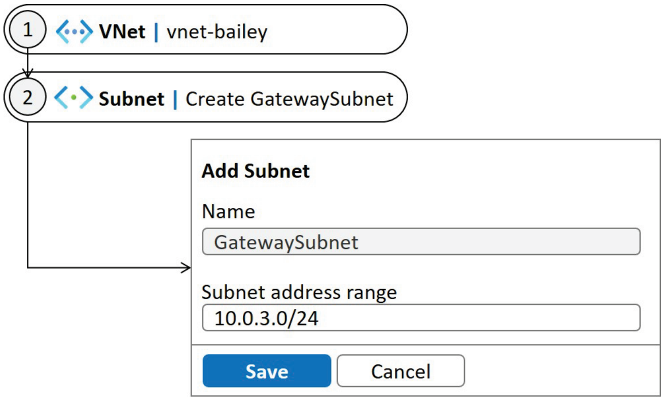


Figure 14-10: *Connect VNet to ExpressRoute Circuit.*

Create GatewaySubnet

Virtual Networks (VNets) are connected to ExpressRoute using ExpressRoute Gateway. The deployment is the same as what we went through in chapter 5. First, we create a GatewaySubnet, which Azure uses for IP addressing for ExpressRoute Gateway. Figure 14-8 illustrates the process. Go to your selected VNet, select Subnet resource, and click the *Create subnet* button (see the section Create Gateway Subnet in chapter 5).



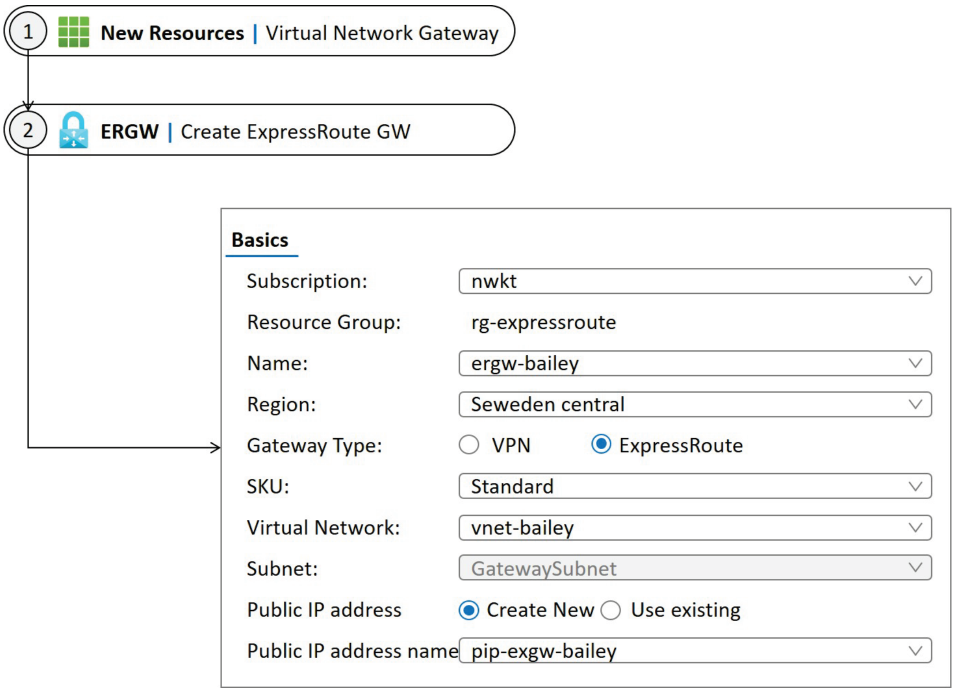
Add Subnet

Name	GatewaySubnet
Subnet address range	10.0.3.0/24

Figure 14-11: *Create GatewaySubnet for ExpressRoute gateway.*

Then create the ExpressRoute Gateway. The only difference compared to VPN Gateway deployment introduced in chapter 5 is that we select the ExpressRoute as a Gateway type.

Configure ExpressRoute Gateway


1
New Resources | Virtual Network Gateway
2
ERGW | Create ExpressRoute GW

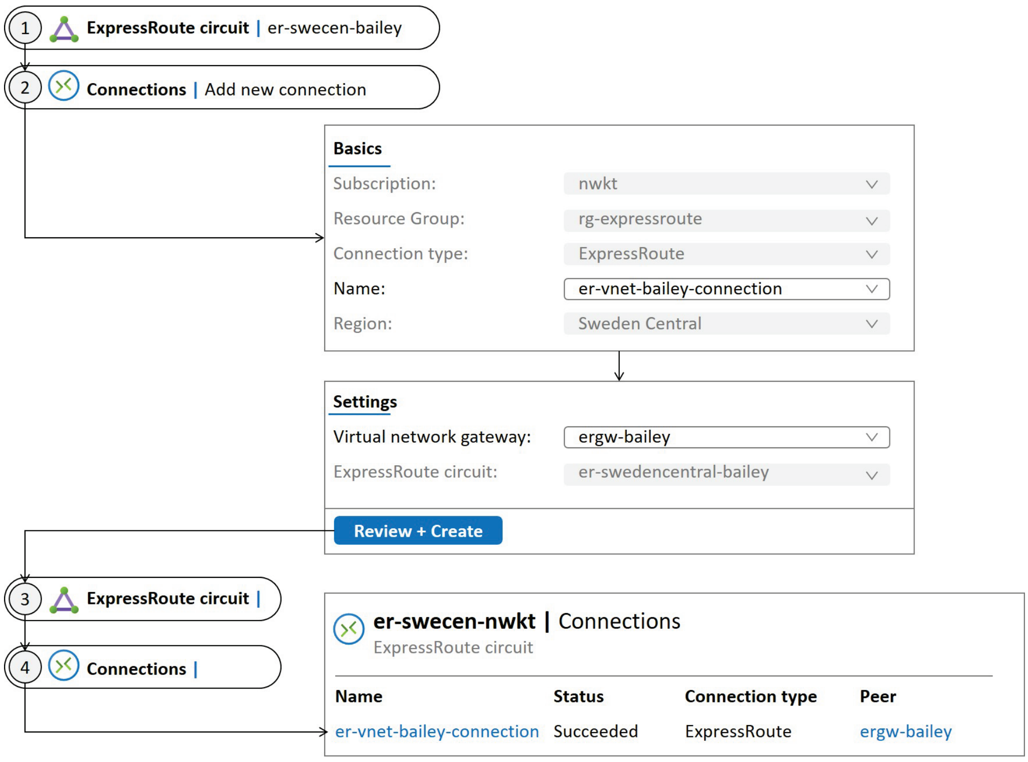
Basics

Subscription:	nwkt
Resource Group:	rg-expressroute
Name:	ergw-bailey
Region:	Seweden central
Gateway Type:	VPN [✓ ExpressRoute]
SKU:	Standard
Virtual Network:	vnet-bailey
Subnet:	GatewaySubnet
Public IP address	[✓ Create New] Use existing
Public IP address name	pip-exgw-bailey

Figure 14-12: *Create ExpressRoute Gateway.*

Connect VNet to Circuit

Virtual Networks (VNets) are connected to ExpressRoute using ExpressRoute Gateway. Navigate to your ExpressRoute circuit page and select Connections under the Settings section. Create a new connection by clicking the +*Add* option at the menu bar. You only need to name the connection in the *Basics* tab. Subscription, Resource Group, Connection type, and region are pre-filled. Next, select your Virtual Network gateway from the drop-down menu. The ExpressRoute circuit name is pre-filled. Click the *Review + Create* button to proceed. After validation and connection deployment, navigate back to the ExpressRoute circuit page, and your new connection should be listed in the connection window.


1
ExpressRoute circuit | er-swecen-bailey
2
Connections | Add new connection
Basics
Settings
Review + Create
3
ExpressRoute circuit |
4
Connections |
er-swecen-nwkt | Connections
ExpressRoute circuit

Subscription:	nwkt
Resource Group:	rg-expressroute
Connection type:	ExpressRoute
Name:	er-vnet-bailey-connection
Region:	Sweden Central
Virtual network gateway:	ergw-bailey
ExpressRoute circuit:	er-swedencentral-bailey

Name	Status	Connection type	Peer
er-vnet-bailey-connection	Succeeded	ExpressRoute	ergw-bailey

Figure 14-13: *Connect VNet to Circuit.*

Appendix A. CE-Bailey Cfg and Show Commands

```
CE-Beetle#
interface Ethernet0/0.200
 encapsulation dot1Q 200
 ip address 192.168.10.1 255.255.255.252
end
!
interface Ethernet0/1.200
 encapsulation dot1Q 200
 ip address 192.168.10.5 255.255.255.252
end
!
router bgp 64777
 bgp log-neighbor-changes
 network 10.1.1.0 mask 255.255.255.0
 neighbor 192.168.10.2 remote-as 12076
 neighbor 192.168.10.6 remote-as 12076
```

```
CE-Beetle#sh ip bgp summ
BGP router identifier 192.168.10.5, local AS number 64777
BGP table version is 3, main routing table version 3
2 network entries using 288 bytes of memory
3 path entries using 252 bytes of memory
2/2 BGP path/bestpath attribute entries using 320 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 884 total bytes of memory
BGP activity 2/0 prefixes, 3/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
192.168.10.2    4        12076      14      14        3    0    0 00:08:35        1
192.168.10.6    4        12076      11      10        3    0    0 00:06:05        1
```

```
CE-Beetle#sh ip bgp
BGP table version is 3, local router ID is 192.168.10.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *    10.0.0.0/16      192.168.10.2             0             0 12076 i
 *>                    192.168.10.6             0             0 12076 i
 *>   10.1.1.0/24      0.0.0.0                  0         32768 i
```

```
CE-Beetle#sh ip route bgp
<snipped>

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 3 subnets, 3 masks
B        10.0.0.0/16 [20/0] via 192.168.10.6, 00:04:47
CE-Beetle#
```

References

[1] Quickstart: Create and modify an ExpressRoute circuit
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-howto-circuit-portal-resource-manager#create

[2] ExpressRoute CrossConnections API development and integration
https://docs.microsoft.com/en-us/azure/expressroute/cross-connections-api-development

[3] Verify ExpressRoute connectivity
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-troubleshooting-expressroute-overview

[4] About ExpressRoute virtual network gateways
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-about-virtual-network-gateways

[5] Tutorial: Connect a virtual network to an ExpressRoute circuit using the Azure portal
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-howto-linkvnet-portal-resource-manager

[6] ExpressRoute SKU scope access
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-faqs#expressroute-sku-scope-access

[7] Dynamic route exchange
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-routing#dynamic-route-exchange

[8] ExpressRoute routing requirements
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-routing

[9] ExpressRoute workflows for circuit provisioning and circuit states
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-workflows

[10] ExpressRoute circuits and peering
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-circuit-peerings

[11] ExpressRoute Local
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-faqs#expressroute-local

[12] Route aggregation and prefix limits
https://docs.microsoft.com/en-us/azure/expressroute/expressroute-routing#route-aggregation-and-prefix-limits

[13] Azure ExpressRoute pricing
https://azure.microsoft.com/en-us/pricing/details/expressroute/

Chapter 15: Load Balancer & NAT - Ananta

Introduction

Chapter 4 introduces the Azure NAT service. In this chapter, we look under the hood of Azure's *Ananta* system that provides NAT service and Loadbalancer functionality. Figure 15-1 shows the simplified architecture of Ananta. *Ananta Manager (AM)* is a highly available *centralized Control Plane* having five replicas, one of which is selected as the Primary/Active one. The primary replica is responsible for the state synchronization between replicas. It also configures *Host Agent (HA)* and *Multiplexers (MUX)* using API. Besides, it monitors the health of each MUX and HAs associated with the Ananta instance. Multiplexer (MUX) pool in the 2nd tier has a set of MUXs (usually eight), which offer a Load Balancing service for inbound traffic. Each MUX gets the *Virtual IP (VIP)* configuration from the AM, which they advertise to upstream routers in 1st tier. Azure Tier 1 routers are BGP multi-path capable and do a flow-based load balancing for inbound data-flows. Besides, MUXs do IP-to-IP encapsulation when forwarding data packets towards VM. They set a *Direct IP (DIP)* of the target VM as the destination address and their own IP address as a source in outer IP header. The third component of Ananta is a *Host Agent (HA)*. It is like a virtual switch within the host Hypervisor's VMSwitch performing NAT functions and packet decapsulation. It gets NAT rules configuration from the AM. In addition, it monitors the state of its VMs.

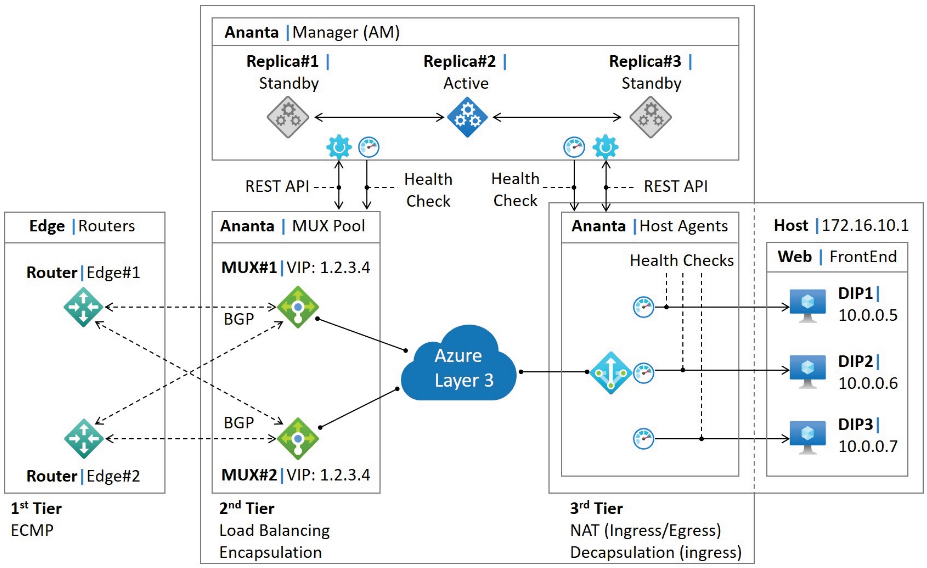
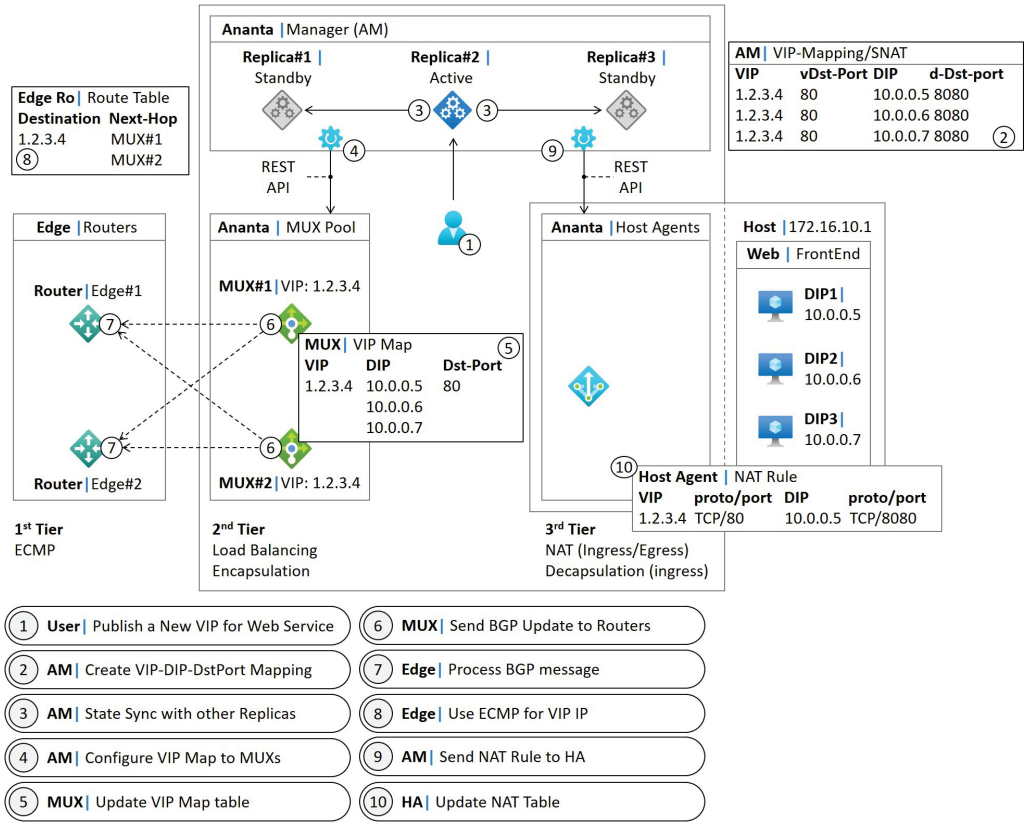


Figure 15-1: *Ananta Instance Building Blocks: AM, MUX, and HA.*

Control Plane

This section shows control plane processes when we publish a new Web service with routable VIP. (1) System administrator creates a Loadbalancer resource for Web service (TCP/80) and assigns a public IP address (Virtual IP/VIP) to it. Our Web service has three front-end Web servers, each having a private IP address (Direct IP/DIP). (2) AM creates VIP-to-DIP/SNAT mapping entries for each DIP. (3) AM shares the information with standby replicas. (4) Next, AM configures the mapping information to each MUX using REST API. (5-8) Every MUX sends a BGP Update message to peering tier 1 routers, which creates a multi-path routing entry in their RIB. (9-10) After configuring MUXs, AM configures the Host Agent hosting the front-end Web servers.



Edge Ro | Route Table

Destination	Next-Hop
1.2.3.4	MUX#1
8	MUX#2

AM | VIP-Mapping/SNAT

VIP	vDst-Port	DIP	d-Dst-port
1.2.3.4	80	10.0.0.5	8080
1.2.3.4	80	10.0.0.6	8080
1.2.3.4	80	10.0.0.7	8080 2

MUX | VIP Map 5

VIP	DIP	Dst-Port
1.2.3.4	10.0.0.5	80
	10.0.0.6	
	10.0.0.7	

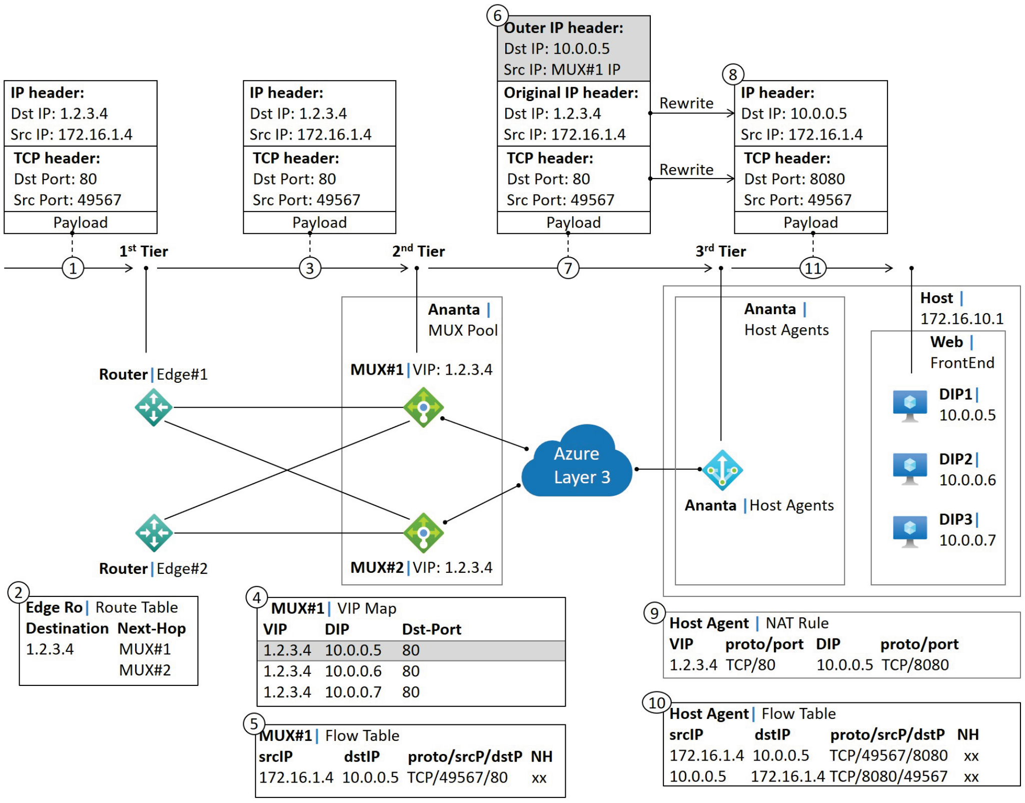
Host Agent | NAT Rule

VIP	proto/port	DIP	proto/port
1.2.3.4	TCP/80	10.0.0.5	TCP/8080

Figure 15-2: *Ananta Control Plane.*

Data Plane for Inbound Traffic: Request

The figure below illustrates processes where an external host requests contents from our example Web page. (1) Tier 1 router Edge#1 receives the first packet with the destination IP address 1.2.3.4, the VIP associated with the web page. (2) Edge#1 has two paths installed into the RIB, so it runs a hash algorithm and selects the one via MUX#1 (3). (4) When MUX#1 receives the packet from Edge#1. The destination IP address is one of the VIP addresses, so it checks associated Direct IP addresses (DIP) from the VIP map table. There are three DIPs, and MUX#1 selects one of them by calculating the hash value from the packet's IP header (src/dstIP) and Transport protocol header (TCP=6, src/dstPort). Based on the hash. It selects a DIP 10.0.0.5. Next, (5) MUX#1 creates a corresponding entry into the flow table. (6) Then, it encapsulates the original packet with a new IP header with the destination IP address 10.0.0.5 (DIP) and uses its IP address as a source IP, and (7) forwards the packet towards the destination based on its routing table. When a Host Agent receives the encapsulated packet, it (8) first removes the outer IP header. Then it (9) rewrites the original destination IP address (VIP > DIP) and destination port based on its NAT rule. Besides, (10) it creates a bi-directional flow entry into the flow table before forwarding the packet to the destination VM.



Edge Ro | Route Table

Destination	Next-Hop
1.2.3.4	MUX#1
	MUX#2

MUX#1 | VIP Map

VIP	DIP	Dst-Port
1.2.3.4	10.0.0.5	80
1.2.3.4	10.0.0.6	80
1.2.3.4	10.0.0.7	80

MUX#1 | Flow Table

srcIP	dstIP	proto/srcP/dstP	NH
172.16.1.4	10.0.0.5	TCP/49567/80	xx

Host Agent | NAT Rule

VIP	proto/port	DIP	proto/port
1.2.3.4	TCP/80	10.0.0.5	TCP/8080

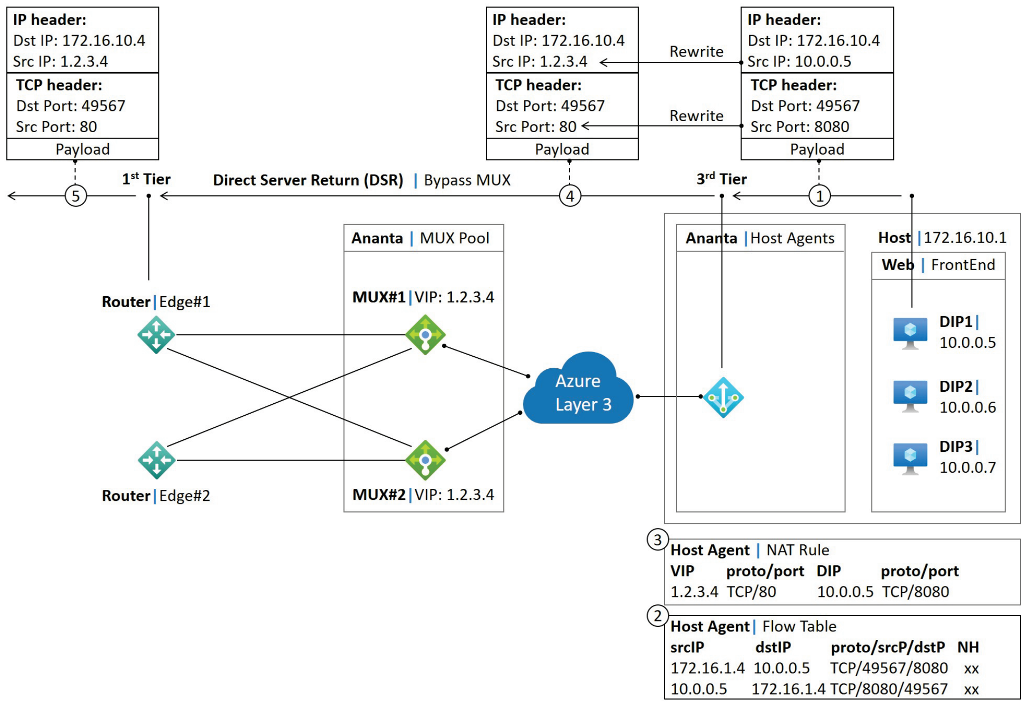
Host Agent | Flow Table

srcIP	dstIP	proto/srcP/dstP	NH
172.16.1.4	10.0.0.5	TCP/49567/8080	xx
10.0.0.5	172.16.1.4	TCP/8080/49567	xx

Figure 15-3: *Inbound Traffic: Request.*

Data Plane for Inbound Traffic: Reply

After processing the received packet, VM with DIP 10.0.0.5 (1) sends a reply message to the external requester. Host Agent notes that (2) its flow table has an entry concerning TCP socket between 172.16.10.4 and 10.0.0.5. Based on its NAT rule table, (3) it rewrites the source DIP with VIP and TCP source port 8080 to 80. Then HA routes the packet towards tier 1 routers bypassing multiplexers.



Host Agent | NAT Rule

VIP	proto/port	DIP	proto/port
1.2.3.4	TCP/80	10.0.0.5	TCP/8080

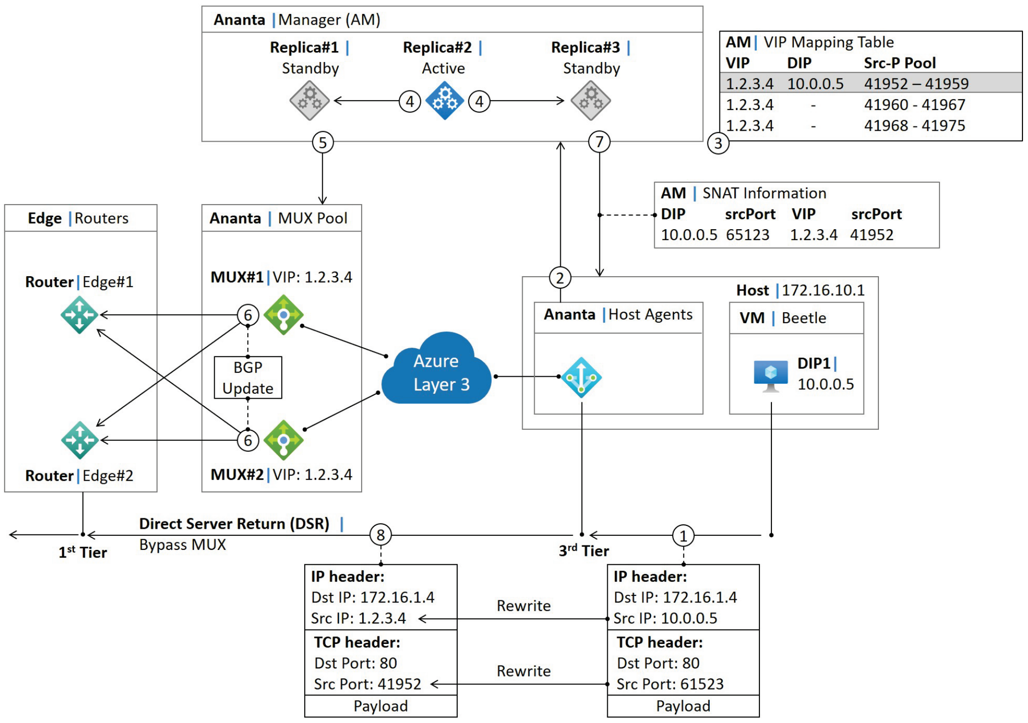
Host Agent | Flow Table

srcIP	dstIP	proto/srcP/dstP	NH
172.16.1.4	10.0.0.5	TCP/49567/8080	xx
10.0.0.5	172.16.1.4	TCP/8080/49567	xx

Figure 15-4: *Inbound Traffic: Reply.*

Data Plane and Control Plane for Outbound Traffic

In figure 15-5, VM beetle starts the http session initiation with an external host 172.16.1.4. (1) It sends the first packet to its gateway. Host Agent intercepts the packet and notices that the source address in the IP header has to be translated to a routable public IP address (PIP), which we have defined in our NAT service (see the configuration from chapter 4). It puts the packet in a queue and requests a VIP address associated with the subnet 10.0.0.0/24 from the Ananta Manager. Besides, it asks for a TCP port to be used as a source port for this connection. AM (3) allocates the VIP 1.2.3.4 and eight source ports 41952-41959 and (4) synchronizes the information with other replicas. Then it (5) configures all MUXs with the VIP-to-DIP-to-srcPort mapping in order to MUXs can add forwarding instructions to their VIP map table and Flow table. Note that though the outbound traffic bypasses MUXs, the return inbound traffic flows go through the MUX pool. (6) MUXs then advertise the VIP to tier 1 BGP, peer speakers. As the last step, (7) AM configures the HA NAT rule table. The reason for allocating eight source ports for one DIP is that VM may open another socket, and HA doesn't have to request a new source port every time VM starts a new connection. After updating the NAT rule table and flow table, HA rewrites the source IP and source port and (8) forwards the packet to the tier 1 router, bypassing the MUX pool.



AM | VIP Mapping Table

VIP	DIP	Src-P Pool
1.2.3.4	10.0.0.5	41952 – 41959
1.2.3.4	-	41960 - 41967
1.2.3.4	-	41968 - 41975

AM | SNAT Information

DIP	srcPort	VIP	srcPort
10.0.0.5	65123	1.2.3.4	41952

After rewrite	Before rewrite
IP header:	**IP header:**
Dst IP: 172.16.1.4	Dst IP: 172.16.1.4
Src IP: 1.2.3.4	Src IP: 10.0.0.5
TCP header:	**TCP header:**
Dst Port: 80	Dst Port: 80
Src Port: 41952	Src Port: 61523
Payload	Payload

Figure 15-5: *Outbound Traffic.*

References

[1] Ananta: Cloud Scale Load Balancing, Parveen Patel et. Al.,
https://conferences.sigcomm.org/sigcomm/2013/papers/sigcomm/p207.pdf

[2] SDN for the Cloud, Albert Greenberg
https://conferences.sigcomm.org/sigcomm/2015/pdf/papers/keynote.pdf