Microsoft

Implementing Microsoft Azure Infrastructure Solutions

SECOND EDITION

Exam Ref 70-533

Rick Rainey
Michael Washam
Dan Patrick
Steve Ross

Microsoft

Exam Ref 70-533 Implementing Microsoft Azure Infrastructure Solutions

2nd Edition

Rick Rainey
Michael Washam
Dan Patrick
Steve Ross

Exam Ref 70-533 Implementing Microsoft Azure Infrastructure Solutions, 2nd Edition

Published with the authorization of Microsoft Corporation by:
Pearson Education, Inc.

ISBN-13: 978-1-5093-0648-0
ISBN-10: 1-5093-0648-X

Library of Congress Control Number: TK
2 17

Editor-in-Chief	Greg Wiegand
Senior Acquisitions Editor	Laura Norman
Development Editor	Troy Mott
Managing Editor	Sandra Schroeder
Senior Project Editor	Tracey Croom
Editorial Production	Backstop Media
Copy Editor	Christina Rudloff
Indexer	Julie Grady
Proofreader	Christina Rudloff
Technical Editor	Tim Warner
Cover Designer	Twist Creative, Seattle

Contents at a glance

	Introduction	*xiii*
	Preparing for the exam	*xvii*
CHAPTER 1	Design and implement Azure App Service Web Apps	1
CHAPTER 2	Create and manage Compute Resources	57
CHAPTER 3	Design and implement a storage strategy	155
CHAPTER 4	Implement Virtual Networks	199
CHAPTER 5	Design and deploy ARM templates	293
CHAPTER 6	Manage Azure Security and Recovery Services	335
CHAPTER 7	Manage Azure Operations	415
CHAPTER 8	Manage Azure Identities	469
	Index	*515*

This page intentionally left blank

Contents

Introduction **xiii**

Organization of this book xiv

Microsoft certifications xiv

Acknowledgments xiv

Microsoft Virtual Academy xv

Quick access to online references xv

Errata, updates, & book support xv

We want to hear from you xvi

Stay in touch xvi

Preparing for the exam *xvii*

Chapter 1 Design and implement Azure App Service Web Apps **1**

Skill 1.1: Deploy web apps 2

Create an App Service Plan 2

Create a web app 6

Define deployment slots 8

Swap deployment slots 11

Deploy an application 14

Migrate a web app to separate App Service Plan 15

Skill 1.2: Configure web apps 16

Configuring application settings 17

Configure a custom domain for a web app 20

Configure SSL certificates 22
Configuring handler mappings 26
Configuring virtual applications and directories 27

Skill 1.3: Configure diagnostics, monitoring, and analytics. 27
Enabling application and web server diagnostics 27
Retrieving diagnostic logs 29
Viewing streaming logs 32
Monitor web app resources 33
Monitor App Service Plan resources 34
Monitor availability, performance, and usage 35
Monitor Azure services 39
Configure backup 40

Skill 1.4: Configure web apps for scale and resilience 42
Scale up or down an app service plan 42
Scale app service instances manually 43
Scale app service instances using Autoscale 44
Configure Azure Traffic Manager 47

Thought experiment. 53

Thought experiment answers 53

Chapter summary 54

Chapter 2 Create and manage Compute Resources 57

Skill 2.1: Deploy workloads on Azure Resource Manager (ARM) virtual machines (VMs). 58
Identify and run workloads in VMs 58
Create virtual machines 60
Connecting to virtual machines 79

Skill 2.2: Perform configuration management. 82
PowerShell Desired State Configuration 83
Using the custom script extension 88
Enable remote debugging 91

Skill 2.3: Design and implement VM Storage. 93
Virtual machine storage overview 93
Operating system images 95

Virtual machine disk caching 98
Planning for storage capacity 100
Disk encryption 104
Using the Azure File Service 106

Skill 2.4: Monitor ARM VMs. 110
Monitoring options in Azure 111
Configuring Azure diagnostics 112
Configuring alerts 119

Skill 2.5: Manage ARM VM availability 124
Configure availability zones 124
Configure availability sets 125

Skill 2.6 Scale ARM VMs. 129
Change VM sizes 130
Deploy and configure VM scale sets (VMSS) 132

Skill 2.7 Manage containers with Azure Container Services (ACS). 138
Configure for open-source tooling 138
Create and manage container images 139
Implement Azure Container Registry 142
Deploy a Kubernetes cluster in ACS 145
Manage containers with Azure Container Services (ACS) 146
Scale applications using Docker Swarm, DC/OS, or Kubernetes 148
Migrate container workloads to and from Azure 149
Monitor Kubernetes by using Microsoft Operations Management Suite (OMS) 149

Though experiment 151

Thought experiment answers 151

Chapter summary 152

Chapter 3 Design and implement a storage strategy 155

Skill 3.1: Implement Azure Storage blobs and files. 155
Manage blob storage 156
Using the async blob copy service 164
Configuring the Content Delivery Network 170
Configuring custom domains for storage and CDN 176

Skill 3.2: Manage access 178
Manage storage account keys 178
Creating, and using, shared access signatures 180
Using a stored access policy 182
Virtual Network Service Endpoints 183

Skill 3.3: Configure diagnostics, monitoring, and analytics 185
Configuring Azure Storage Diagnostics 185
Analyzing diagnostic data 187
Enabling monitoring and alerts 189

Skill 3.4: Implement storage encryption 190
Encrypt data using Azure Storage Service Encryption (SSE) 191
Implement encryption and role based access control with Azure Data Lake Store 192

Thought experiment. 195

Thought experiment answers 196

Chapter summary 196

Chapter 4 Implement Virtual Networks 199

Skill 4.1: Configure Virtual Networks. 199
Create a Virtual Network (VNet) 200
Design subnets 208
Gateway subnets 210
Setup DNS at the Virtual Network level 211
User Defined Routes (UDRs) 214
Connect VNets using VNet peering 217
Implement Application Gateway 232

Skill 4.2: Design and implement multi-site or hybrid network connectivity 239
Choose the appropriate solution between ExpressRoute, Site-to-Site and Point-to-Site 240
Choose the appropriate gateway 246
Identify network prerequisites 247
Implement Virtual Network peering service chaining 248
Configure Virtual Network and Multi-Site Virtual Networks 251

Skill 4.3: Configure ARM VM Networking. 260

Configure Private Static IP Addresses 261

Public IP Address 264

DNS at the Network Interface (NIC) Level 268

Network Security Groups (NSGs) 270

User Defined Routes (UDR) with IP Forwarding 274

External and Internal load balancing with HTTP and TCP health probes 275

Direct Server Return 283

Design and Implement Application Gateway (App Gateway) 285

Skill 4.4: Design and implement a communication strategy 287

Leverage Site-to-Site (S2S) VPN to connect to an on-premises infrastructure 287

Implement Hybrid Connections to access data sources on-premises 288

Thought experiment. 289

Thought experiment answers 290

Chapter summary 290

Chapter 5 Design and deploy ARM templates 293

Skill 5.1: Implement ARM templates 294

Author ARM templates 294

Deploy an ARM template 308

Skill 5.2: Control access 310

Leverage service principals with ARM authentication 311

Set management policies 315

Lock resources 319

Skill 5.3: Design role-based access control (RBAC) 322

Implement Azure RBAC standard roles 322

Design Azure RBAC custom roles 328

Thought experiment. 330

Thought experiment answers 330

Chapter summary 331

Chapter 6 Manage Azure Security and Recovery Services 335

Skill 6.1: Manage data protection and security compliance 335

Create and import encryption keys with Key Vault 336

Automate tasks for SSL/TLS Certificates 342

Prevent and respond to security threats with Azure Security Center 346

Configure single sign-on with SaaS applications using federation and password based 363

Add users and groups to applications 369

Revoke access to SaaS applications 370

Configure federation with public consumer identity providers such as Facebook and Google 371

Skill 6.2: Implement recovery services. 378

Create a Recovery Services vault 378

Backup and restore data 383

Use of snapshots 387

Geo-replication for recovery 392

Implement DR as service, Deploy ASR agent, ASR Configuration & best practices 393

Thought experiment 412

Thought experiment answer. 413

Chapter summary 413

Chapter 7 Manage Azure Operations 415

Skill 7.1: Enhance cloud management with automation 415

Implement PowerShell runbooks 416

Integrate Azure Automation with Web Apps 428

Create and manage PowerShell Desired State Configurations (DSC) 432

Import DSC resources 433

Generate DSC node configurations 434

Monitor and automatically update machine configurations with Azure Automation DSC 436

Skill 7.2: Collect and analyze data generated by resources in cloud and on-premises environments 441

Collect and search across data sources from multiple systems 441

Build custom visualizations 452

Visualize Azure resources across multiple subscriptions 456

Transform Azure activity data and managed resource data into an insight with flexible search queries 457

Monitor system updates and malware status 459

Track server configuration changes by using Azure Log Analytics 463

Thought experiment. 466

Thought experiment answers 467

Chapter summary 467

Chapter 8 Manage Azure Identities 469

Skill 8.1: Monitor On-Premises Identity Infrastructure and Synchronization Services with Azure AD Connect Health 470

Monitor Sync Engine & Replication 470

Monitor Domain Controllers 473

Setup Email Notifications for Critical Alerts 476

Monitor ADFS proxy and Web Application proxy Servers 477

Generate Utilization Reports 479

Skill 8.2: Manage Domains with Active Directory Domain Services. 479

Implement Azure AD Domain Services 480

Join Azure virtual machines to a Domain 482

Securely Administer Domain-joined virtual machines by using Group Policy 484

Migrate On-premises Apps to Azure 485

Handle Traditional Directory-aware Apps along with SaaS Apps 485

Skill 8.3: Integrate with Azure Active Directory (Azure AD) 485

Add Custom Domains 485

Implement Azure AD Connect and Single Sign-on with On-premises Windows Server 487

Multi-Factor Authentication (MFA) 488
Config Windows 10 with Azure AD Domain Join 493
Implement Azure AD Integration in Web and Desktop Applications 495
Leverage Microsoft Graph API 496

Skill 8.4: Implement Azure AD B2C and Azure AD B2B 497
Create an Azure AD B2C Directory 497
Register an Application 502
Implement Social Identity Provider Authentication 503
Enable Multi-Factor Authentication (MFA) 506
Set up Self-Service Password Reset 507
Implement B2B Collaboration and Configure Partner Users 508
Integrate with Applications 510

Thought experiment. 512

Thought experiment answers 512

Chapter summary 513

Introduction

The 70-533 exam focuses on Infrastructure as a Service (IaaS) features available in Microsoft Azure (storage, networking, and compute). The exam also covers some of the more common Platform as a Service (PaaS) services that an IT professional will experience in an Azure environment. It covers deploying and configuring virtual machines, virtual machine scale sets, containers, and web apps, with Azure App Services including integration with services like the Content Deployment Network (CDN). This exam also covers the intricacies of networking, including hybrid connectivity with technologies like ExpressRoute, site-to-site VPN, and point-to-site, as well as a broad range of storage related topics, such as choosing the right storage solution, and understanding how to scale and diagnose performance.

Other key capabilities measured by the exam include your ability to author and deploy ARM templates, and automate workloads using a wide variety of services and tools, such as the Azure command line tools, Azure Automation, and implementing monitoring solutions for infrastructure and services deployed in Azure. Security is a key topic that is interwoven throughout the subjects, including topics such as encryption at rest and in transit, as well as identity management with Azure AD and monitoring for security threats with Azure Security Center.

This book is written specifically for IT professionals who want to demonstrate their skills to implement and configure these services in Microsoft Azure.

This book covers every major topic area found on the exam, but it does not cover every exam question. Only the Microsoft exam team has access to the exam questions, and Microsoft regularly adds new questions to the exam, making it impossible to cover specific questions. You should consider this book a supplement to your relevant real-world experience and other study materials. If you encounter a topic in this book that you do not feel completely comfortable with, use the “Need more review?” links you’ll find in the text to find more information and take the time to research and study the topic. Great information is available on MSDN, TechNet, and in blogs and forums.

Organization of this book

This book is organized by the "Skills measured" list published for the exam. The "Skills measured" list is available for each exam on the Microsoft Learning website: *https://aka.ms/examlist*. Each chapter in this book corresponds to a major topic area in the list, and the technical tasks in each topic area determine a chapter's organization. If an exam covers six major topic areas, for example, the book will contain six chapters.

Microsoft certifications

Microsoft certifications distinguish you by proving your command of a broad set of skills and experience with current Microsoft products and technologies. The exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop, or implement and support, solutions with Microsoft products and technologies both on-premises and in the cloud. Certification brings a variety of benefits to the individual and to employers and organizations.

MORE INFO **ALL MICROSOFT CERTIFICATIONS**

For information about Microsoft certifications, including a full list of available certifications, go to *https://www.microsoft.com/learning*.

Acknowledgments

Rick Rainey. It is a privilege to be a contributing author to such a valuable resource for the IT professional working in Azure. To the reader, it is my hope that the information in this text provides a rich learning experience. Thank you to everyone who has contributed to this second edition. To my family and dearest friends, thank you for your patience and support during this journey.

Michael Washam. Helping others learn about the cloud is always a great experience, and I hope this second edition helps readers learn more about Azure, and of course ultimately help them prepare for passing the exam! I would like to thank my wife Becky for being very patient with me when I take on projects like this, and my co-authors for making this book excellent by passing on their immense technical expertise. In addition to the tech gurus, I would like to thank James Burleson at Opsgility for editing assistance and the rest of the folks at the Opsgility team for being patient during the authoring and editing process. Finally, the editors and reviewers from Pearson provided fantastic support and feedback throughout the process.

Dan Patrick. Writing this book has taught me much more than probably anyone who reads it will ever learn. To Michael Washam, thank you for taking a chance on me. Finally, I want to thank my two girls Stella and Elizabeth, and the love of my life Michelle for being the reason why I continue to learn.

Steve Ross. This was my first foray into authoring a book, and it was quite a learning experience. Many thanks to Michael Washam and Dan Patrick for patiently answering a lot of questions during the process. I'm also thankful for the folks who accomplished the editing and technical reviews, as the work is much improved due to their diligence. Endeavors like this, done "off the side of the desk" require a lot of afterhours work, so thanks to my beautiful wife and kids for putting up with my late nights in the office. Finally, I'm thankful to God for a wonderful career in IT, and for many other kindnesses too numerous to mention. May He be honored in all I do.

Microsoft Virtual Academy

Build your knowledge of Microsoft technologies with free expert-led online training from Microsoft Virtual Academy (MVA). MVA offers a comprehensive library of videos, live events, and more to help you learn the latest technologies and prepare for certification exams. You'll find what you need here:

https://www.microsoftvirtualacademy.com

Quick access to online references

Throughout this book are addresses to webpages that the author has recommended you visit for more information. Some of these addresses (also known as URLs) can be painstaking to type into a web browser, so we've compiled all of them into a single list that readers of the print edition can refer to while they read.

Download the list at *https://aka.ms/examref5332E/downloads*.

The URLs are organized by chapter and heading. Every time you come across a URL in the book, find the hyperlink in the list to go directly to the webpage.

Errata, updates, & book support

We've made every effort to ensure the accuracy of this book and its companion content. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

https://aka.ms/examref533/errata

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support, email Microsoft Press Book Support at *mspinput@microsoft.com*.

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to *https://support.microsoft.com*.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

https://aka.ms/tellpress

We know you're busy, so we've kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*.

Preparing for the exam

Microsoft certification exams are a great way to build your resume and let the world know about your level of expertise. Certification exams validate your on-the-job experience and product knowledge. Although there is no substitute for on-the-job experience, preparation through study and hands-on practice can help you prepare for the exam. We recommend that you augment your exam preparation plan by using a combination of available study materials and courses. For example, you might use the Exam ref and another study guide for your "at home" preparation, and take a Microsoft Official Curriculum course for the classroom experience. Choose the combination that you think works best for you.

Note that this Exam Ref is based on publicly available information about the exam and the author's experience. To safeguard the integrity of the exam, authors do not have access to the live exam.

This page intentionally left blank

CHAPTER 4

Implement Virtual Networks

Azure Virtual Networks (VNet) provide the infrastructure for deploying workloads that require an advanced network configuration. VNets provide support for hybrid network connectivity from Azure to either your on-premises network or to other VNets within Azure regions. The use of VMs in Azure is entirely dependent upon the VNets where deployed. These can be internet-facing applications that are deployed behind either the Azure load balancer for Layer 4 workloads, or the Azure Application Gateway, which can deploy more complex Layer 7 implementations. Azure VNets also provide support for deploying intranet or n-tier workloads using the internal load balancer. Workloads, such as Active Directory, are also enabled in the cloud using features only supported in VNets (such as subnets and static IP addresses). This chapter will focus on VNets as well as how to create and configure them. There is also a focus on key hybrid technologies.

Skills covered in this chapter:

- Skill 4.1 Configure Virtual Networks
- Skill 4.2 Design and implement multi-site or hybrid network connectivity
- Skill 4.3 Configure ARM VM Networking
- Skill 4.4 Design and implement a communication strategy

Skill 4.1: Configure Virtual Networks

Configuring an Azure VNet involves network design skills such as specifying the address spaces (IP network ranges), and dividing the network into subnets. Setting up name resolution with DNS at the Virtual Network level is critical when connecting multiple networks together, and VNet peering is one option for this that is covered here. Securing VNets by implementing network security groups, along with the ability to control the routing on VNets applying User Defined Routes, is also covered. These serve as the most important tasks for deployment of VMs into VNets to provide services for your cloud implementation. One such deployment of VMs will be the use of the Azure Application Gateway used for Layer 7 load balancing. Customization of these many services, along with designing the VNet itself, is covered in this skill.

This skill covers how to:

- Create a VNet
- Design subnets
- Setup DNS at the Virtual Network level
- User Defined Routes (UDRs)
- Connect VNets using VNet peering
- Setup network security groups (NSGs)
- Deploy a VM into a VNet
- Implement Application Gateway

Create a Virtual Network (VNet)

Azure VNets enable you to securely connect Azure VMs to each other and to extend your on-premises network to the Azure cloud. A VNet is a representation of your own network in the cloud and is a logical isolation of the Azure cloud dedicated to your subscription.

VNets are isolated from one another allowing you to create separate VNets for development, testing, and production. Although it's not recommended, you can even use the same Classless Inter-Domain Routing (CIDR) address blocks. The best plan is to create multiple VNets that use different CIDR address blocks. This allows you to connect these networks together if you wish. Each VNet can be segmented into multiple subnets. Azure provides internal name resolution for VMs and cloud service role instances connected to a VNet. You can optionally configure a VNet to use your own DNS servers instead of using Azure internal name resolution.

The following are some of the connectivity capabilities that you should understand about VNets:

- **internet connectivity** All Azure Virtual Machines (VMs), connected to a VNet, have access to the internet by default. You can also enable inbound access to specific resources running in a VNet, such as a web server.
- **Azure resource connectivity** Azure resources such as VMs can be connected to the same VNet. The VMs can connect to each other by using private IP addresses, even if they are in different subnets. Azure provides system routes between subnets, VNets, and on-premises networks, so you don't have to configure and manage routes.
- **VNet connectivity** VNets can be connected together, enabling VMs connected to any VNet to communicate with any VM on any other VNet. This can be accomplished by using VNet peering if they are in the same region or by using VPN Gateways if they are in different Azure regions. Traffic from one VNet to another VNet is always secured and never egresses to the internet.

- **On-premises connectivity** VNets can be connected to on-premises networks through private network connections and Azure (ExpressRoute), or through an encrypted Site-to-Site VPN connection over the internet.
- **Traffic filtering (firewall)** VM network traffic can be filtered by using 5-tuple inbound and outbound by source IP address and port, destination IP address and port, and protocol.
- **Routing** You can optionally override Azure's default routing by configuring your own routes, or by using BGP routes through a VPN Gateway.

All VMs connected to a VNet have outbound connectivity to the internet by default. The resource's private IP address is source network address translated (SNAT), to a Public IP address by the Azure infrastructure. You can change the default connectivity by implementing custom routing and traffic filtering.

To communicate inbound to Azure resources from the internet, or to communicate outbound to the internet without SNAT, a resource must be assigned a Public IP address.

Azure VNets can be created by using the Azure portal, Azure PowerShell cmdlets, or the Azure CLI.

Creating a Virtual Network using the Azure Portal

To create a new VNet by using the Azure portal, first click New and then select Networking. Next, click Virtual Network as shown in Figure 4-1.

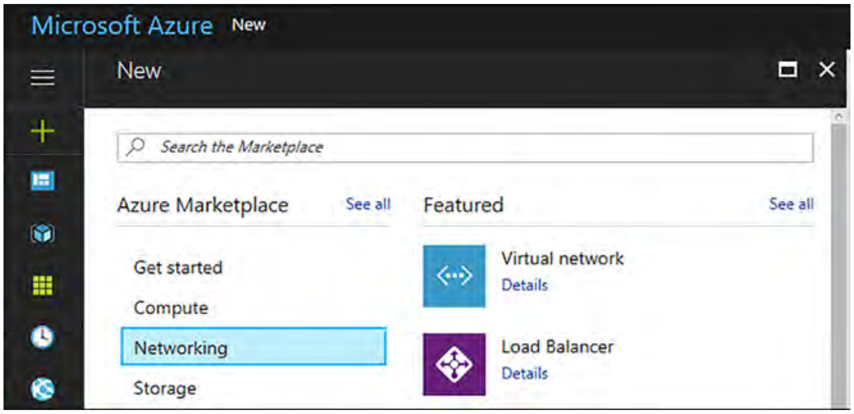
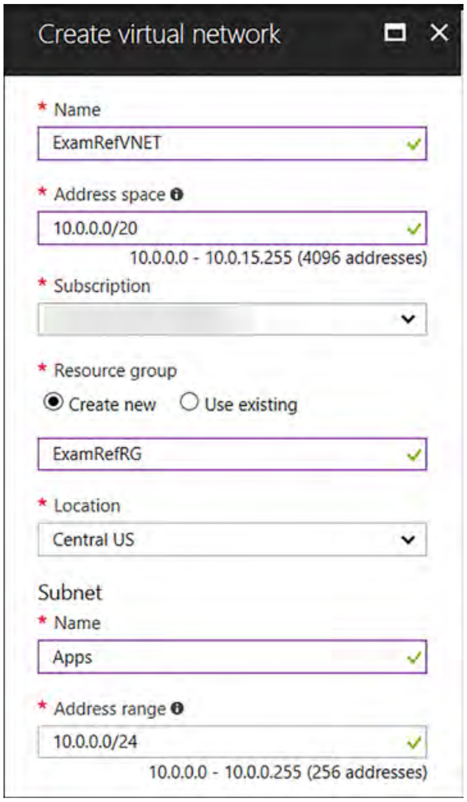


FIGURE 4-1 Creating a Virtual Network using the Azure Portal

The Create Virtual Network blade opens. Here you can provide configuration information about the Virtual Network. This blade requires the following inputs, as shown in Figure 4-2:

- Name of the Virtual Network
- Address Space to be used for the VNet using CIDR notation
- Subscription in which the VNet is created

- The resource group where the VNet is created
- Location for VNet
- Subnet Name for the first subnet in the VNet
- The Address Range of the first Subnet


Create virtual network

* Name
ExamRefVNET

* Address space
10.0.0.0/20
10.0.0.0 - 10.0.15.255 (4096 addresses)

* Subscription

* Resource group
● Create new ○ Use existing
ExamRefRG

* Location
Central US

Subnet
* Name
Apps

* Address range
10.0.0.0/24
10.0.0.0 - 10.0.0.255 (256 addresses)

FIGURE 4-2 Create Virtual Network Blade

The address space is the most critical configuration for a VNet in Azure. This is the IP range for the entire network that will be divided into subnets. The address space can almost be any IP range that you wish (public or private). You can add multiple address spaces to a VNet. To ensure this VNet can be connected to other networks, the address space should never overlap with any other networks in your environment. If a VNet has an address space that overlaps with another Azure VNet or on-premises network, the networks cannot be connected, as the routing of traffic will not work properly.

IMPORTANT **ADDRESS RANGES**

There are some address ranges that cannot be used for VNets.

These include the following:

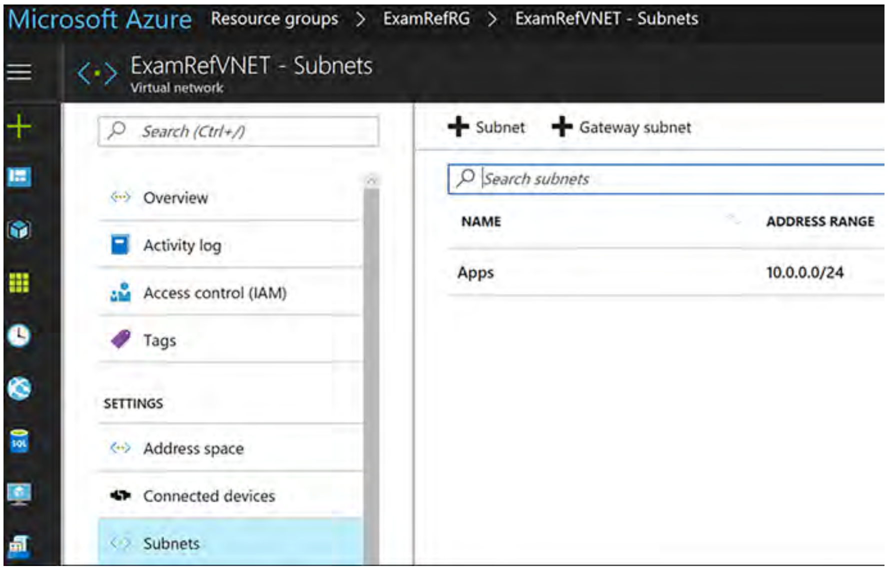
- 224.0.0.0/4 (Multicast)
- 255.255.255.255/32 (Broadcast)
- 127.0.0.0/8 (Loopback)
- 169.254.0.0/16 (Link-local)
- 168.63.129.16/32 (Internal DNS)

MORE INFO **ADDRESS SPACE FROM IP RANGES**

Whether you define your address space as public or private, the address space is only reachable from within that Virtual Network or any other network that you successfully connected to in the VNet. Typically, the address spaces are from IP ranges defined in RFC 1918. This RFC includes IP addresses in the following ranges:

- 10.0.0.0 - 10.255.255.255 (10.0.0.0/8)
- 172.16.0.0 - 172.31.255.255 (172.16.0.0/12)
- 192.168.0.0 - 192.168.255.255 (192.168.0.0/16)

Once the VNet has completed provisioning, you can review the settings using the Azure portal. Notice the Apps subnet has been created as part of the inputs you made as seen in Figure 4-3.

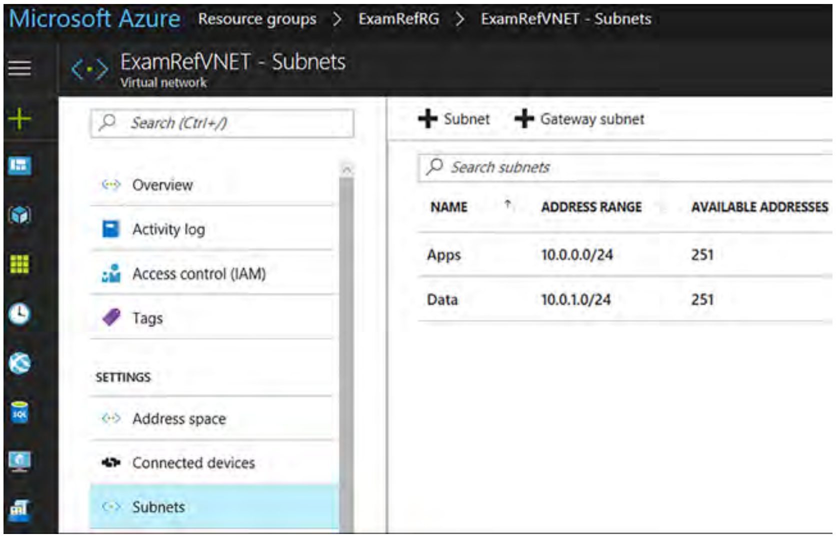


NAME	ADDRESS RANGE
Apps	10.0.0.0/24

FIGURE 4-3 Virtual Network created using the Azure portal

To create another subnet in the ExamRefVNET, click +Subnet on this blade and provide the following inputs, as shown in Figure 4-4:

- Name of the Subnet: Data.
- Address Range: This is the portion of the address space that is made available for the subnet. In the case, it is 10.0.1.0/24.



NAME	ADDRESS RANGE	AVAILABLE ADDRESSES
Apps	10.0.0.0/24	251
Data	10.0.1.0/24	251

FIGURE 4-4 Virtual Network created using the Azure Portal with two subnets Apps and Data

Creating a Virtual Network with PowerShell

Using the Azure PowerShell cmdlets, you can create and configure VNets programmatically. Let's walk through how to create a VNet using PowerShell.

Before you can create or manage any resources in your Azure subscription by using the Azure PowerShell cmdlets, you must log in by executing the Login-AzureRmAccount cmdlet.

```
Login-AzureRmAccount
```

Once the PowerShell session is authenticated to Azure, the first thing needed will be a new resource group. Using the New-AzureRmResourceGroup cmdlet, you can create a new resource group. This cmdlet requires you to specify the resource group name as well as the name of the Azure region. These values are defined in the variables $rgName and $location.

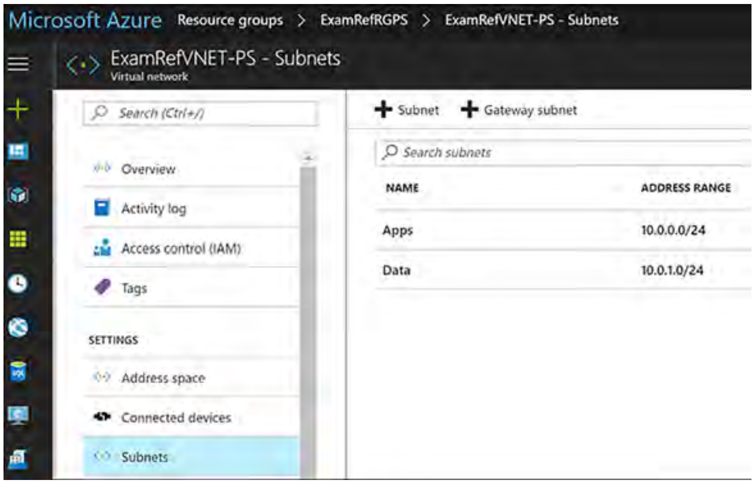
```
$rgName      = "ExamRefRGPS"
$location    = "Central US"
New-AzureRmResourceGroup -Name $rgName -Location $location
```

If you wanted to use an existing resource group you can use the Get-AzureRmResource-Group cmdlet to see if the resource group. You can also use the Get-AzureRmLocation cmdlet to view the list of available regions.

In the code example below, the New-AzureRmVirtualNetworkSubnetConfig cmdlet is used to create two local objects that represent two subnets in the VNet. The VNet is subsequently created with the call to New-AzureRmVirtualNetwork. It is passed the address space of 10.0.0.0/16. You could also pass in multiple address spaces like how the subnets were passed in using an array. Notice how `$subnets = @()` creates an array and then the array is loaded with two different commands using the New-AzureRmVirtualNetworkSubnetConfig cmdlet. When the New-AzureRmVirtualNetwork cmdlet is called in the last command of the script, the two subnets are then populated by the array values that have been loaded in the $subnets.

```
$subnets = @()
$subnet1Name = "Apps"
$subnet2Name = "Data"
$subnet1AddressPrefix = "10.0.0.0/24"
$subnet2AddressPrefix = "10.0.1.0/24"
$vnetAddresssSpace = "10.0.0.0/16"
$VNETName = "ExamRefVNET-PS"
$rgName = "ExamRefRGPS"
$location = "Central US"
$subnets = New-AzureRmVirtualNetworkSubnetConfig -Name $subnet1Name `
                                    -AddressPrefix $subnet1AddressPrefix
$subnets = New-AzureRmVirtualNetworkSubnetConfig -Name $subnet2Name `
                                    -AddressPrefix $subnet2AddressPrefix
$vnet = New-AzureRmVirtualNetwork -Name $VNETName `
                          -ResourceGroupName $rgName `
                          -Location $location `
                          -AddressPrefix $vnetAddresssSpace `
                          -Subnet $subnets
```

Following the completion of the PowerShell script, there should be a new resource group and a new VNet provisioned. In Figure 4-5, you see the VNet ExamRefVNET-PS was created in the ExamRefRGPS resource group. You can now click on the subnets button to view the new App and Data subnets and their address ranges.



NAME	ADDRESS RANGE
Apps	10.0.0.0/24
Data	10.0.1.0/24

FIGURE 4-5 Virtual Network created using PowerShell

Creating a Virtual Network using the Azure CLI

By using the Azure CLI 2.0, you can create and configure VNets by using individual commands or as a part of a script to be run using a bash script or batch file depending upon the platform (bash is supported on Linux, macOS, or Windows Subsystem for Linux).

***IMPORTANT* AZURE CLI**

The Azure CLI is available on many platforms including Windows, MAC, Linux and even as a Docker container. It is also available in the Azure portal and is called the Azure Cloud Shell. To learn more about installing the Azure CLI 2.0, read this article: *https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest.*

***NOTE* AZURE CLOUD SHELL**

For these examples, use the Azure Cloud Shell. If you are using the Azure CLI installed on your machine or from a Docker container, you first need to log in to Azure by using the `az login` **command and follow the instructions to authenticate your session.**

In this case here, let's walk through each command to create the VNet using the Azure CLI Cloud Shell. To initiate the Azure CLI Cloud Shell, open the Azure portal and then click the CLI symbol along the upper right-hand corner as seen in Figure 4-6.

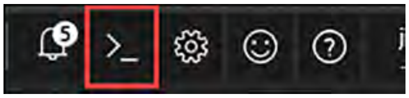
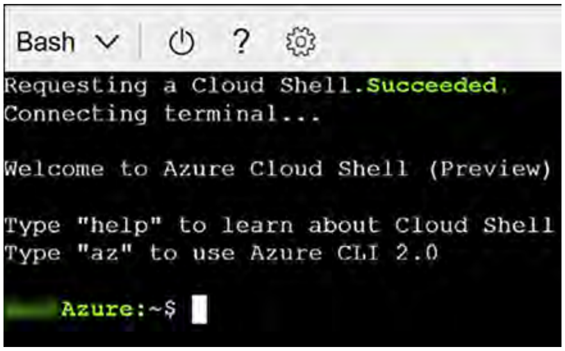


FIGURE 4-6 Azure CLI Cloud Shell

After a few moments, the Cloud Shell will be ready, and you will see an interactive bash prompt. In Figure 4-7, Azure Cloud Shell is ready to use with your subscription.



```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell (Preview)

Type "help" to learn about Cloud Shell
Type "az" to use Azure CLI 2.0

    Azure:~$
```

FIGURE 4-7 Azure CLI Cloud Shell

The first step will be creating a new resource group for the VNet using the Azure CLI. This will be accomplished using the `az group create` command. You will need to specify a location for the resource group. To locate a list of regions that are available for your subscription, you can use the command `az account list-locations`.

```
az group create -n ExamRefRGCLI -l "centralus"
```

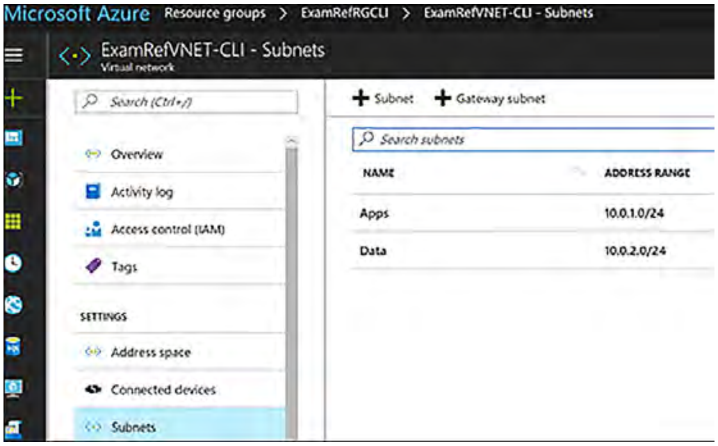
Next, you can create the new VNet using the az network vnet create command.

```
az network vnet create --resource-group ExamRefRGCLI -n ExamRefVNET-CLI
--address-prefixes 10.0.0.0/16 -l "centralus"
```

Then, following the creation of the VNet, create the App and Data subnets. This is accomplished using the az network vnet subnet create command. You will run these commands one at a time for each subnet.

```
az network vnet subnet create --resource-group ExamRefRGCLI --vnet-name
ExamRefVNET-CLI -n Apps --address-prefix 10.0.1.0/24
az network vnet subnet create --resource-group ExamRefRGCLI --vnet-name
ExamRefVNET-CLI -n Data --address-prefix 10.0.2.0/24
```

After running these commands there should be a new resource group named ExamRefRGCLI and the newly provisioned VNet named ExamRefVNET-CLI. In Figure 4-8, you see the ExamREFVNET-CLI, which was created in the ExamRefRGCLI resource group. If you click the Subnets button you will see the new App and Data subnets with the address ranges from the commands entered.



NAME	ADDRESS RANGE
Apps	10.0.1.0/24
Data	10.0.2.0/24

FIGURE 4-8 Virtual Network created using the Azure CLI Cloud Shell

Design subnets

A subnet is a child resource of a VNet, which defines segments of address spaces within a VNets. These are created using CIDR blocks of the address space that was defined for the VNet. NICs can be added to subnets and connected to VMs. This will provide connectivity for various workloads.

The name of a subnet must be unique within that VNet. You cannot change the subnet name subnet following its creation. During the creation of a VNet while using the Azure portal, the requirement is for you to define one subnet, even though a VNet isn't required to have any subnets. In the portal, you can define only one subnet when you create a VNet. You can add more subnets to the VNet later after it has been created. You can create a VNet that has multiple subnets by using Azure CLI or PowerShell.

When creating a subnet, the address range must be defined. The address range of the new subnet must be within the address space you assigned for the VNet. The range that is entered will determine the number of IP Addresses that are part of the subnet.

EXAM TIP

Azure will hold back a total of 5 IP Addresses for every subnet that is created in a VNet. Azure reserves the first and last IP addresses in each subnet like standard IP networks with one for the network identification and the other for broadcast. Azure also holds three additional addresses for internal use starting from the first address in the subnet. For example, if the CIDR range of a subnet has its first IP as .0 then the first useable IP would be .4. So, if the address range was 192.168.1.0/24 then 192.168.1.4 would be the first address assigned to a NIC. Also, the smallest subnet on an Azure VNet would be a CIDR /29. This would provide 3 useable IP Addresses and 5 IP Addresses that Azure would use.

Subnets provide the ability to isolate network traffic between various types of workloads. These are often different types of servers or even tiers of applications. Examples of this could include separating traffic bound for web servers and database servers. These logical segmentations allow for clean separations, so they can be secured and managed. This allows for very precise application of rules securing data traffic as well as how traffic flows into and out of a given set of VMs.

In Azure, the security rules are applied using network security groups, and the traffic flows are controlled using route tables. Designing the subnets should be completed upfront and should be considered while determining the address space. Remember that for each subnet, Azure holds back 5 IP Addresses. If you create a VNet with 10 subnets, you are losing 50 IP addresses to Azure. Careful up-front planning is critical to not causing yourself a shortage of IPs later.

Changes to subnets and address ranges can only be made if there are no devices connected to the subnet. If you wish to make a change to a subnet's address range, you would first have to delete all the objects in that subnet. If the subnet is empty, you can change the range of addresses to any range that is within the address space of the VNet not assigned to any other subnets.

Subnets can be only be deleted from VNets if they are empty. Once a subnet is deleted, the addresses that were part of that address range would be released and available again for use within new subnets that you could create.

Subnets have the following properties: Name, Location, Address range, Network security group, Route table and Users. Table 4-1 discusses each of these properties.

TABLE 4-1 Properties of a Virtual Network Subnet

Property	Description
Name	Subnet name up to 80 characters. May contain letters, numbers, underscores, periods, or hyphens. Must start with a letter or number. Must end with a letter, number, or underscore.
Location	Azure location must be the same as the Virtual Network.
Address Range	Single address prefix that makes up the subnet in CIDR notation. Must be a single CIDR block that is part of one of the VNets address spaces.
Network Security Group	NSGs are essentially firewall rules that can be associated to a subnet. These rules are then applied to all NICs that are attached to that subnet.
Route Table	Route table applied to the subnet that would change the default system routes. These are used to send traffic to destination networks that are different than the routes that Azure uses by default.
Users	Which users have access to use the subnet as a part of Role-Based Access Control.

EXAM TIP

Network security groups (NSGs) can be associated to subnets, individual NICs or both. NSGs associated with subnets will enforce their inbound or outbound rules to all NICs that are connected to the subnet. Inbound rules from a subnet NSG will apply first as packets entering the subnet prior to entering the NIC. Upon enforcement following the NSG in association with the subnet, any NSGs associated with NICs would then be enforced. The opposite is true for outbound rules with the NIC NSG being enforced followed by the subnet NSG, as the traffic is following in the reverse direction. For example, if a VM was running a web server on PORT 80, and it has a NSG associated to the NIC with an inbound Rule that allowed traffic on PORT 80, and the subnet did not, the packets would be discarded without ever reaching the VM. The reason is because the NSG at the subnet level is blocking the traffic on PORT 80 for all NICs.

Gateway subnets

The basis for deploying hybrid clouds is the connection of an on-premises network along with an Azure VNet. This configuration allows clients and servers deployed in Azure to communicate with those in your datacenter and network. To deploy this type of connection, a VPN Gateway needs to be created in Azure. All VPN Gateways must be placed into a special gateway subnet.

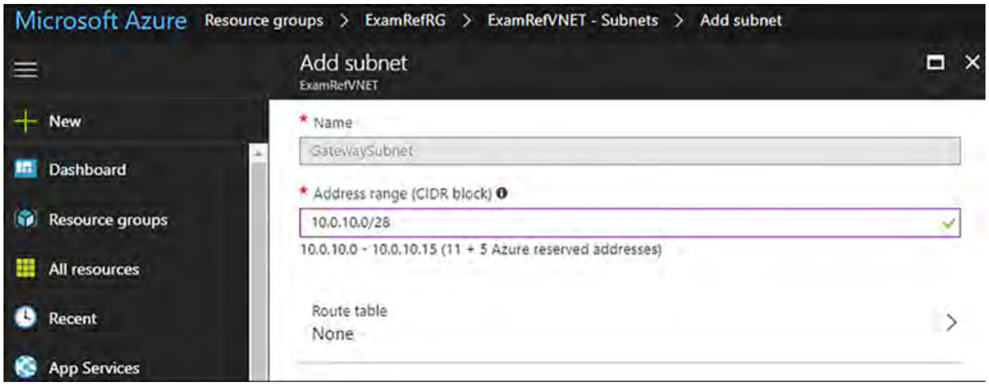
The gateway subnet contains the IP addresses the VPN Gateway VMs and services will use. When you create your VPN Gateway, special Azure managed VMs are deployed to the gateway subnet, and they are configured with the required VPN Gateway settings. Only the VPN Gateways should be deployed to the gateway subnet and its name must be "GatewaySubnet" to work properly.

When you create the gateway subnet, you are required to specify the number of IP addresses available using an address range. The IP addresses in the gateway subnet will be allocated to the gateway VMs and services. It's important to plan ahead because some configurations require more IP addresses than others. For example, if you plan on using ExpressRoute and a Site to Site VPN as a failover, you will need more than just two IPs. You can create a gateway subnet as small as /29, but it's Microsoft's recommendation to create a gateway subnet of /28 or larger (i.e., /28, /27, /26). That way, if you add functionality in the future, you won't have to tear down your gateway. Just delete and recreate the gateway subnet to allow for more IP addresses.

Creating a GatewaySubnet using the Azure portal

The GatewaySubnet can be created using the Azure portal, PowerShell or CLI. Because these aren't created very often they are typically created using the Azure portal.

To create the GatewaySubnet, first open an existing VNet, and move to the subnets under settings. From here, the current subnets can be reviewed. Click +GatewaySubnet, and you will be required to enter the address range for the subnet as seen in Figure 4-9.



Add subnet
ExamRefVNET

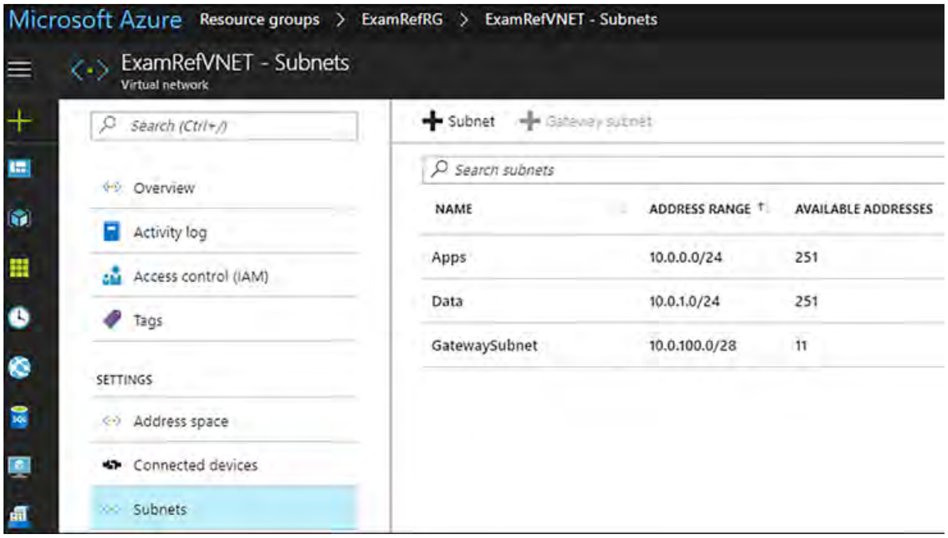
* Name
GatewaySubnet

* Address range (CIDR block)
10.0.10.0/28
10.0.10.0 - 10.0.10.15 (11 + 5 Azure reserved addresses)

Route table
None

FIGURE 4-9 Adding a GatewaySubnet used for VPN Gateways

Once the Gateway subnet is added, the VPN Gateway can be created and placed into this subnet. Many network administrators will create this address range much further away from their subnets in terms of the IP Addressing. Figure 4-10 shows, the GatewaySubnet created using a CIDR block of 10.0.100.0/28. The other subnets are using /24 CIDR blocks for Apps and Data. In this case the GatewaySubnet is 98 subnets away from the others. This is not required, as the GatewaySubnet could be any CIDR address range belonging to the address space of the VNet. This would provide for a continuation of the subnet scheme put in place if the admin wanted to build additional subnets. The next logical subnet would be in the 10.0.2.0/24 and so forth as more are created.



NAME	ADDRESS RANGE	AVAILABLE ADDRESSES
Apps	10.0.0.0/24	251
Data	10.0.1.0/24	251
GatewaySubnet	10.0.100.0/28	11

FIGURE 4-10 GatewaySubnet after being created using the Azure Portal

Setup DNS at the Virtual Network level

The Domain Naming Service (DNS) is critical on all modern networks, and VNets are no different. It is possible for all network communication to be completed by using IP addresses, but it is much simpler to use names that can easily be remembered and do not change. There are two options for providing DNS services to a VNet:

- **Azure Provided DNS** Highly durable and scalable service
- **Customer Managed DNS** Build and deploy your own DNS Servers

By default, Azure DNS is configured when creating a VNet. VMs connected to the VNet use this service for name resolution inside the VNet, as well as on the public internet. Along with resolution of public DNS names, Azure provides internal name resolution for VMs that reside within the same VNet. Within a VNet the DNS suffix is consistent, so the Fully Qualified Domain Name (FQDN), is not needed. This means that VMs on the same VNet using the Azure DNS Server can connect directly via their host names.

Although Azure-provided name resolution does not require any configuration, it is not the appropriate choice for all deployment scenarios. If your needs go beyond what Azure provided DNS can provide, you will need to implement your own Customer Managed DNS Servers. Table 4-2 captures the recommended DNS infrastructure based on various common requirements. Focus in on the scenarios that would require you to deploy your own DNS.

TABLE 4-2 Determining a DNS strategy for an Azure Virtual Network

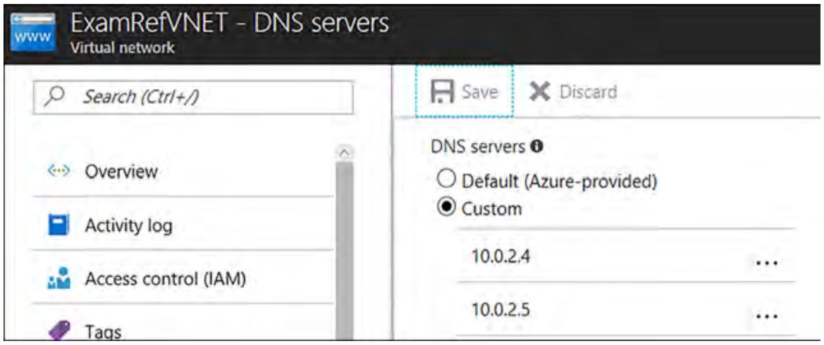
Requirement	Recommended DNS infrastructure
Name resolution between role instances or VMs located in the same cloud service or VNet	Azure provided DNS
Name resolution between role instances or VMs located in different VNets	Customer managed DNS
Resolution of on-premises computer and service names from role instances or VMs in Azure	Customer managed DNS
Resolution of Azure hostnames from on-premises computers	Customer managed DNS
Reverse DNS for internal IPs	Customer managed DNS

When using your own DNS servers, Azure provides the ability to specify multiple DNS servers per VNet. Once in place, this configuration will cause the Azure VMs in the VNet to use your DNS servers for name resolution services. You must restart the VMs for this configuration to update.

You can alter the DNS Servers configuration for a VNet using the Azure portal, PowerShell or Azure CLI.

Configuring VNet Custom DNS Settings using the Azure portal

To configure the DNS Servers using the Azure portal, open the VNet that will be configured and click DNS Servers, as seen in Figure 4-11. Select Custom and then input the IP Addresses of your DNS Servers that have been configured and click Save.



DNS servers
Default (Azure-provided) [✓ Custom]
10.0.2.4
10.0.2.5

FIGURE 4-11 Custom DNS Servers configured using the Azure Portal

Configuring VNet Custom DNS Settings using PowerShell

When creating a new VNet, you can specify the customer DNS settings configuration using PowerShell. The New-AzureRmVirtualNetwork cmdlet with the `-DNSServer` (as a part of the command) will create a new VNet with the DNS Servers already specified. This command would assume that your resource group was already created.

```
$subnets = @()
$subnet1Name = "Apps"
$subnet2Name = "Data"
$subnet1AddressPrefix = "10.0.0.0/24"
$subnet2AddressPrefix = "10.0.1.0/24"
$vnetAddresssSpace = "10.0.0.0/16"
$VNETName = "ExamRefVNET-PS"
$rgName = "ExamRefRGPS"
$location = "Central US"

$subnets = New-AzureRmVirtualNetworkSubnetConfig -Name $subnet1Name `
                                    -AddressPrefix $subnet1AddressPrefix
$subnets = New-AzureRmVirtualNetworkSubnetConfig -Name $subnet2Name `
                                    -AddressPrefix $subnet2AddressPrefix
$vnet = New-AzureRmVirtualNetwork -Name $VNETName `
                                  -ResourceGroupName $rgName `
                                  -Location $location `
                                  -AddressPrefix $vnetAddresssSpace `
                                  -DNSServer 10.0.0.4,10.0.0.5 `
                                  -Subnet $subnet
```

Configuring VNet Custom DNS settings using the Azure CLI

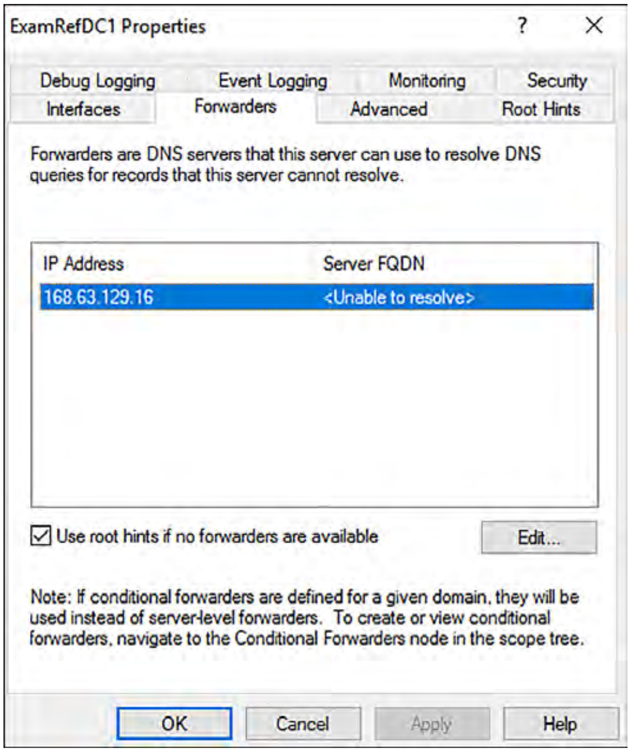
To create a VNet using the Azure CLI and specify custom DNS Servers, you will need to add the `–dns-servers` argument when using both `az network vnet create` `commands`.

```
az network vnet create --resource-group ExamRefRGCLI -n ExamRefVNET-CLI
--address-prefixes 10.0.0.0/16 --dns-servers 10.0.0.4 10.0.0.5 -l "centralus"
```

IMPORTANT **VNET SETTINGS**

When you change your VNet settings to point to your Customer Managed DNS Servers, you will need to restart each of the VMs in that particular VNet. All IP addresses and settings in Azure are provided via the Azure DHCP Servers. The only way to make sure the change is picked up by the OS on the VMs is to reboot. When the VM reboots, it will re-acquire the IP Address and the new DNS Settings will be in place

Customer-managed DNS servers within a VNet can forward DNS queries to Azure's recursive resolvers to resolve hostnames within that VNet. For example, you could use a Domain Controller (DC), running in Azure to respond to DNS queries for its domains, and forward all other queries to Azure. This allows your VMs to see both on-premises resources (via the DC), and Azure-provided hostnames (via the forwarder). Access to Azure's recursive resolvers is provided via the virtual IP 168.63.129.16. If you promote a Domain Controller that is running as VM in Azure, it will automatically pick up the Azure DNS resolver as a Forwarder. In Figure 4-12, you see a Domain Controller deployed to an Azure VNet acting as a DNS Server for that VNet. The Forwarder is set to the address of the Azure recursive resolver.



Forwarders are DNS servers that this server can use to resolve DNS queries for records that this server cannot resolve.

IP Address	Server FQDN
168.63.129.16	<Unable to resolve>

☑ Use root hints if no forwarders are available

Note: If conditional forwarders are defined for a given domain, they will be used instead of server-level forwarders. To create or view conditional forwarders, navigate to the Conditional Forwarders node in the scope tree.

FIGURE 4-12 DNS Forwarder configuration on a Domain Controller running in a Virtual Network

User Defined Routes (UDRs)

Azure VMs that are added to a VNet can communicate with each other over the network automatically. Even if they are in different subnets or attempting to gain access to the internet, there are no configurations required by you as the administrator. Unlike typical networking, you will not need to specify a network gateway, even though the VMs are in different subnets. This is also the case for communication from the VMs to your on-premises network when a hybrid connection from Azure to your datacenter has been established.

This ease of setup is made possible by what is known as system routes. System routes define how IP traffic flows in Azure VNets. The following are the default system routes that Azure will use and provide for you:

- Within the same subnet
- Subnet to another subnet within a VNet
- VMs to the internet
- A VNet to another VNet through a VPN Gateway
- A VNet to another VNet through VNet peering (Service Chaining)
- A VNet to your on-premises network through a VPN Gateway

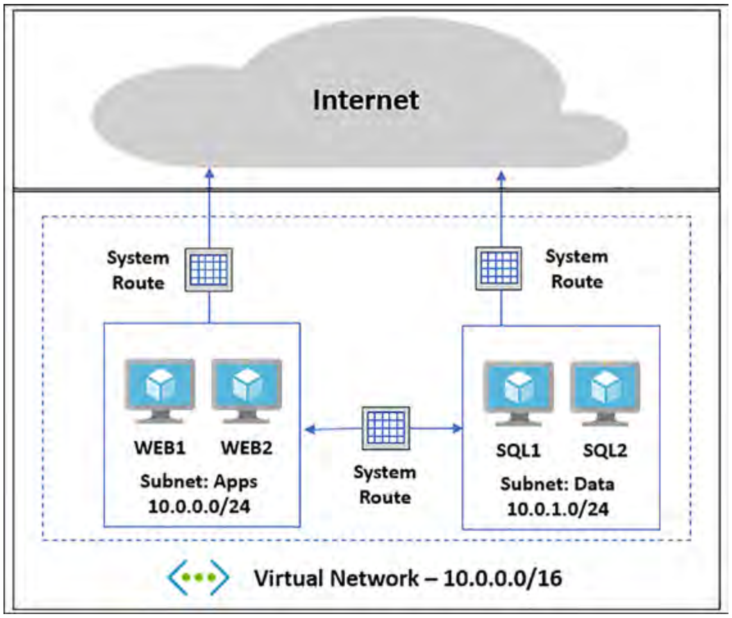


FIGURE 4-13 N-Tier Application Deployed to Azure VNet using system routes

Figure 4-13 shows an example of how these system routes make it easy to get up and running. System routes provide for most typical scenarios by default, but there are use cases where you will want to control the routing of packets.

One of the scenarios is when you want to send traffic through a virtual appliance such as a third-party load balancer, firewall or router deployed into your VNet from the Azure Marketplace.

To make this possible, you must create User Defined Routes (UDRs). These UDRs specify the next hop for packets flowing to a specific subnet through your appliance instead of following the system routes. As seen in Figure 4-14, by using the UDR, traffic will be routed through the device to the destination.

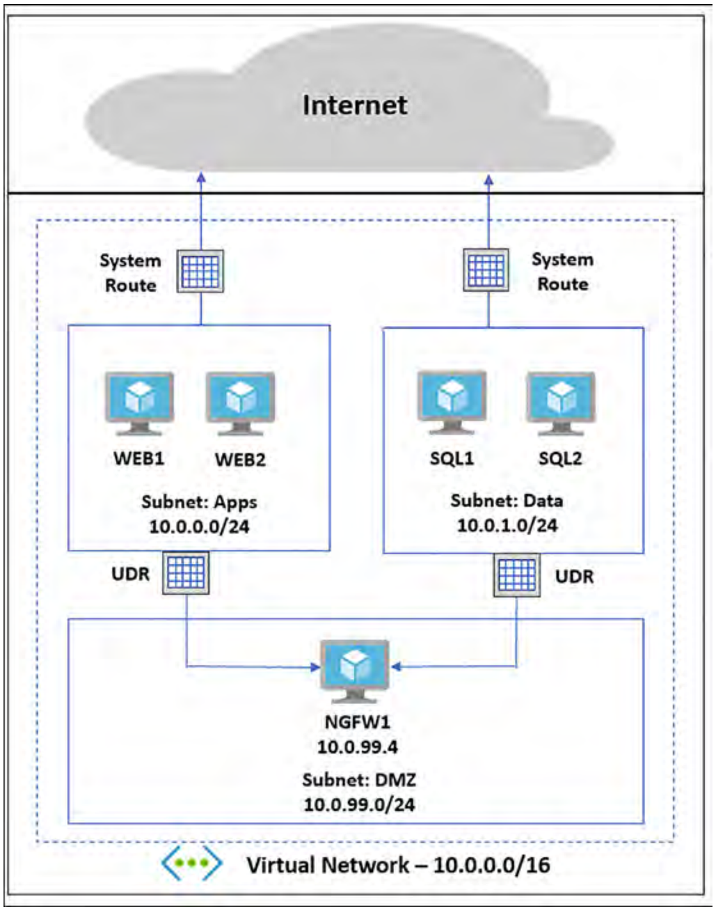
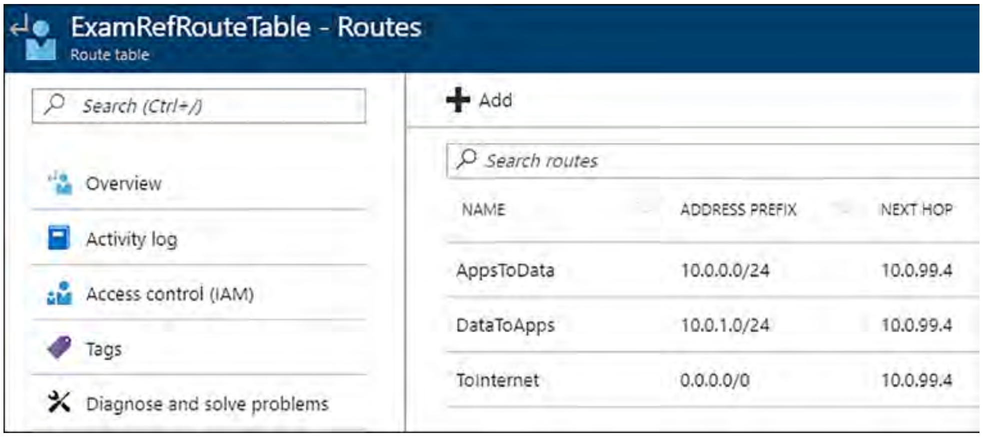


FIGURE 4-14 N-Tier Application Deployed with a Firewall using User Defined Routes

EXAM TIP

You can have multiple route tables, and the same route table can be associated to one or more subnets. Each subnet can only be associated to a single route table. All VMs in a subnet use the route table associated to that subnet.

Figure 4-15 shows a UDR that has been created to allow for traffic to be directed to a virtual appliance. In this case, it would be a Firewall running as a VM in Azure in the DMZ subnet.



NAME	ADDRESS PREFIX	NEXT HOP
AppsToData	10.0.0.0/24	10.0.99.4
DataToApps	10.0.1.0/24	10.0.99.4
ToInternet	0.0.0.0/0	10.0.99.4

FIGURE 4-15 User Defined Route forcing network traffic through firewall

Connect VNets using VNet peering

VNet peering allows you to connect two Azure Resource Manager (ARM), VNets located in the same region together without complex integration. The peered VNets must have non-overlapping IP address spaces. The address spaces in either VNet cannot be added to or deleted from a VNet once a VNet is peered with another VNet.

Once peered, the VNets appear as one network and all VMs in the peered VNets can communicate with each other directly. The traffic between VMs is routed through the Microsoft backbone network in the same way that traffic is routed between the VM in the same VNet through private IP addresses.

You can create a VNet peering using the Azure portal, PowerShell or Azure CLI. Figure 4-16 shows VNETA and VNETB, and they are both in the North Central Azure region. These two VNets will be used to describe how to create VNet peerings.

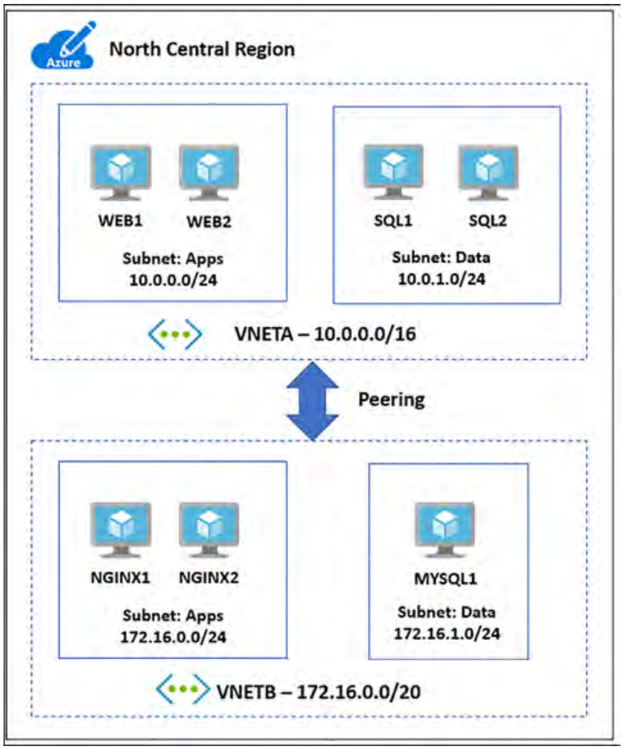
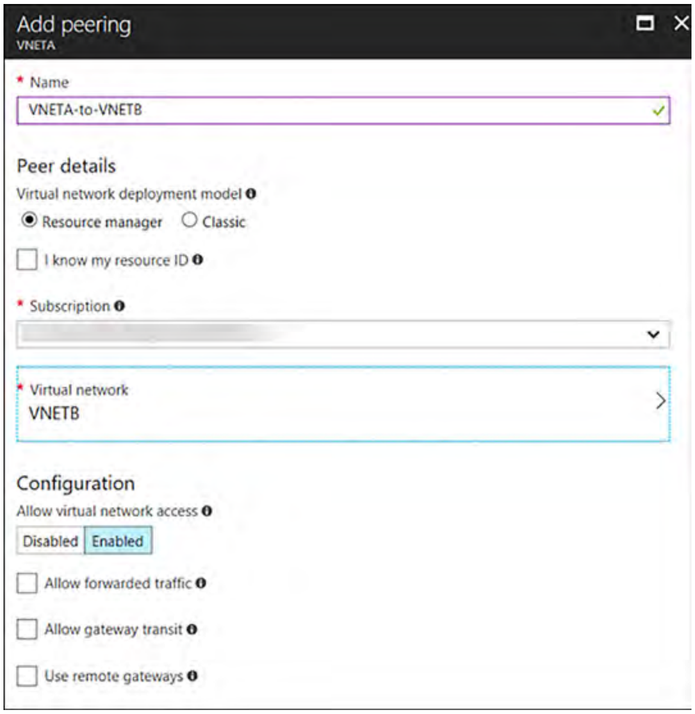


FIGURE 4-16 VNet peering between two networks in the North Central Region

Creating a VNet peering using the Azure portal

The VNets you wish to peer must already be created to establish a VNet peering. To create a new VNet peering from VNETA to VNETB as shown in Figure 4-17, connect to the Azure portal and locate VNETA. Once this is located under Settings, click peerings, and then select +Add. This will load the Add peering blade. Use the following inputs to connect a standard VNet peering:

- **Name** VNETA-to-VNETB
- **Peer Details** Resource Manager (leave the I Know My Resource ID unchecked)
- **Subscription** Select the Subscription for VNETB
- **Virtual Network** Choose VNETB
- **Configuration** Enabled (leave the remaining three boxes unchecked for this simple VNet Peering)


Add peering
VNETA

* Name
VNETA-to-VNETB

Peer details
Virtual network deployment model
(●) Resource manager ( ) Classic

☐ I know my resource ID

* Subscription

* Virtual network
VNETB

Configuration
Allow virtual network access
Disabled [✓ Enabled]

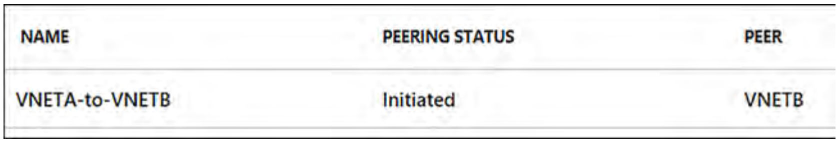
☐ Allow forwarded traffic

☐ Allow gateway transit

☐ Use remote gateways

FIGURE 4-17 Adding peering from VNETA to VNETB using the Portal

Once this process has been completed, the VNet peering will appear in the portal along with the initiation of peering status, as seen in Figure 4-18. To complete the VNet peering, you will follow the same steps on VNETB.



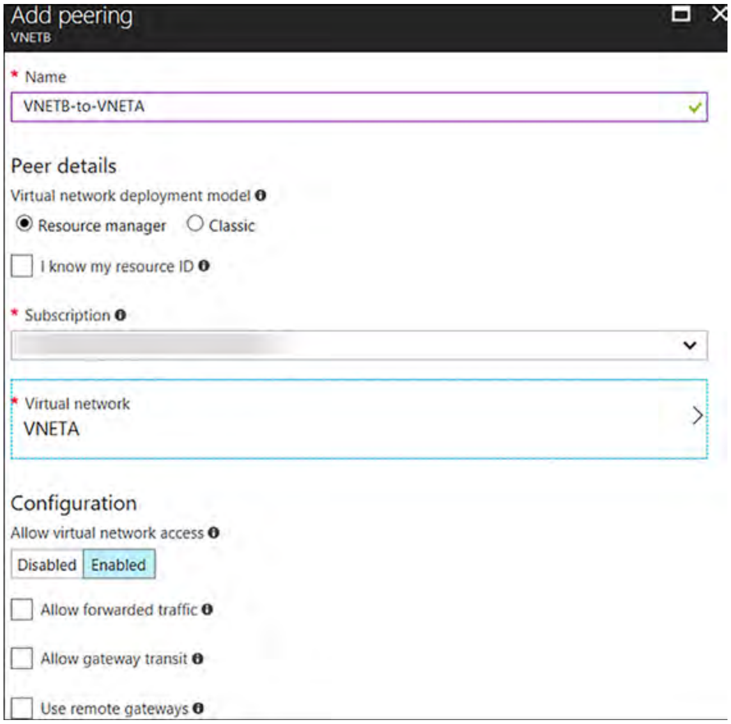
NAME	PEERING STATUS	PEER
VNETA-to-VNETB	Initiated	VNETB

FIGURE 4-18 VNETA-to-VNETB Peering showing as Initiated in the Azure Portal

Now, in the portal, complete the same steps using VNETB. Open VNETB in the Azure portal and click peerings. Next, click +Add and complete the Add peering blade by using the following inputs, as shown in Figure 4-19:

- **Name** VNETB-to-VNETA
- **Peer Details** Resource Manager (Leave the I Know My Resource ID checkbox unchecked)
- **Subscription** Select the Subscription for VNETA

- **Virtual Network** Choose VNETA
- **Configuration** Enabled (leave the other three boxes unchecked for this simple peering)


Add peering
VNETB

* Name
VNETB-to-VNETA ✓

Peer details
Virtual network deployment model
● Resource manager ○ Classic

☐ I know my resource ID

* Subscription

* Virtual network
VNETA

Configuration
Allow virtual network access
Disabled [✓ Enabled]

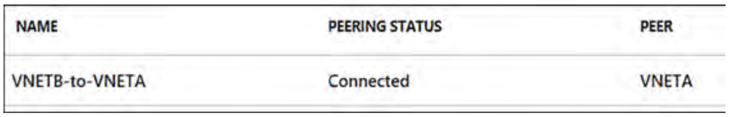
☐ Allow forwarded traffic

☐ Allow gateway transit

☐ Use remote gateways

FIGURE 4-19 Adding peering from VNETB to VNETA using the Portal

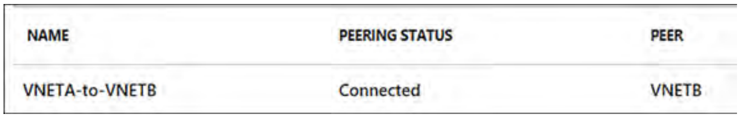
Once the portal has completed the provisioning of the VNet Peering, it will appear in the peering of VNETB and show as Connected with a peer of VNETA, as seen in Figure 4-20. Now the two VNets: VNETA and VNETB are peers, and VMs on these networks can see each other. They are accessible, as if this was one Virtual Network.



NAME	PEERING STATUS	PEER
VNETB-to-VNETA	Connected	VNETA

FIGURE 4-20 VNETB-to-VNETA Peering showing as Connected in the Azure Portal

In Figure 4-21, the peering blade of VNETA shows the peering status VNETA-to-VNETB is also as Connected to VNETB.



NAME	PEERING STATUS	PEER
VNETA-to-VNETB	Connected	VNETB

FIGURE 4-21 VNETA-to-VNETB Peering showing as Connected in the Azure Portal

Creating a VNet peering using PowerShell

When creating a new VNet peering using PowerShell, you will first leverage the Get-AzureRm-VirtualNetwork cmdlet to assign information about the VNETA and VNETB into two variables. Using the Add-AzureRmVirtualNetworkPeering, create the VNet peerings on both VNets just as you did in the portal. Upon completion, the VNet peering will provision, and move to a connected peering status. You can use the Get-AzureRmVirtualNetworkPeering cmdlet to verify the peering status of the VNets.

```
# Load VNETA and VNETB into Variables
$vneta = Get-AzureRmVirtualNetwork `
-Name "VNETA" `
-ResourceGroupName "VNETARG"

$vnetb = Get-AzureRmVirtualNetwork `
-Name "VNETB" `
-ResourceGroupName "VNETBRG"

# Peer VNETA to VNETB.
Add-AzureRmVirtualNetworkPeering `
  -Name 'VNETA-to-VNETB' `
  -VirtualNetwork $vneta `
  -RemoteVirtualNetworkId $vnetb.Id

# Peer VNETB to VNETA.
Add-AzureRmVirtualNetworkPeering `
  -Name 'VNETA-to-VNETB'
  -VirtualNetwork $vnetb `
  -RemoteVirtualNetworkId $vneta.Id

#Check on the Peering Status
  Get-AzureRmVirtualNetworkPeering `
  -ResourceGroupName VNETARG `
  -VirtualNetworkName VNETA `
  | Format-Table VirtualNetworkName, PeeringState
```

Creating VNet Peering using the Azure CLI

To create a VNet peering using the Azure CLI, you will need to use the az network vnet show command to get the Resource ID of each VNet. Next, you will use the az network vnet peering create command to create each of the VNet peerings. Upon successfully running these commands, you can use the az network vnet peering list command to see the peering status as Connected.

```
# Get the Resource IDs for VNETA and VNETB.
az network vnet show --resource-group VNETAResourceGroupName --name VNETA
--query id --out tsv
az network vnet show --resource-group VNETBResourceGroupName --name VNETB
 --query id --out tsv

# Peer VNETB to VNET: the output from the Command to find the Resource ID
 for VNETA & VNETB is used with the --remote-vnet-id argument
az network vnet peering create --name VNETA-to-VNETB --resource-group VNETARG
 --vnet-name VNETA --allow-vnet-access --remote-vnet-id /subscriptions/
11111111-1111-1111-1111-111111111111/resourceGroups/VNETBRG/
providers/Microsoft.Network/virtualNetworks/VNETB

# Peer VNETB to VNETA. the output from the Command to find the Resource ID for
 VNETA is used with the --remote-vnet-id argument
az network vnet peering create --name VNETB-to-VNETA --resource-group
VNETBRG --vnet-name VNETB --allow-vnet-access --remote-vnet-id /subscriptions/
11111111-1111-1111-1111-111111111111/resourceGroups/VNETARG/providers/
Microsoft.Network/virtualNetworks/VNETA

#To See the Current State of the Peering
az network vnet peering list --resource-group VNETARG --vnet-name VNETA -o table
```

Setup Network Security Groups (NSGs)

A network security group (NSG) is a networking filter (firewall) containing a list of security rules, which when applied allow or deny network traffic to resources connected to Azure VNets. These rules can manage both inbound and outbound traffic. NSGs can be associated to subnets and/or individual network interfaces attached to ARM VMs and to classic VMs. Each NSG has the following properties regardless of where it is associated:

- NSG Name
- Azure region where the NSG is located
- Resource group
- Rules that define whether inbound or outbound traffic is allowed or denied

EXAM TIP

When an NSG is associated to a subnet, the rules apply to all resources connected to the subnet. Traffic can further be restricted by also associating an NSG to a VM or NIC. NSGs that are associated to subnets are said to be filtering "North/South" traffic, meaning packets flow into and out of a subnet. NSGs that are associated to network interfaces are said to be filtering "East/West" traffic, or how the VMs within the subnet connect to each other.

NSG Rules

NSG Rules are the mechanism defining traffic the administrator is looking to control. Table 4-3 captures the important information to understand about NSG Rules.

TABLE 4-3 NSG properties

PROPERTY	DESCRIPTION	CONSTRAINTS	CONSIDERATIONS
Name	Name of the rule	Must be unique within the region. Must end with a letter, number, or underscore. Cannot exceed 80 characters.	You can have several rules within an NSG, so make sure you follow a naming convention that allows you to identify the function of your rule.
Protocol	Protocol to match for the rule	TCP, UDP, or *	Using * as a protocol includes ICMP (East-West traffic only), as well as UDP and TCP, can reduce the number of rules you need. This is a very broad approach, so it’s recommended that you only use when necessary.
Source port range	Source port range to match for the rule	Single port number from 1 to 65535, port range (example: 1-65535), or * (for all ports)	Source ports could be ephemeral. Unless your client program is using a specific port, use * in most cases. Try to use port ranges as much as possible to avoid the need for multiple rules. Multiple ports or port ranges cannot be grouped by a comma.
Destination port range	Destination port range to match for the rule	Single port number from 1 to 65535, port range (example: 1-65535), or * (for all ports)	Try to use port ranges as much as possible to avoid the need for multiple rules. Multiple ports or port ranges cannot be grouped by a comma.
Source address prefix	Source address prefix or tag to match for the rule	Single IP address (example: 10.10.10.10), IP subnet (example: 192.168.1.0/24), Tag or * (for all addresses)	Consider using ranges, default tags, and * to reduce the number of rules.
Destination ad-dress prefix	Destination ad-dress prefix or tag to match for the rule	Single IP address (example: 10.10.10.10), IP subnet (example: 192.168.1.0/24), Tag or * (for all addresses)	Consider using ranges, default tags, and * to reduce the number of rules.
Direction	Direction of traffic to match for the rule	Inbound or outbound	Inbound and outbound rules are pro-cessed separately, based on direction.

PROPERTY	DESCRIPTION	CONSTRAINTS	CONSIDERATIONS
Priority	Rules are checked in the order of priority. Once a rule applies, no more rules are tested for matching.	Unique Number between 100 and 4096. Uniqueness is only within this NSG.	Consider creating rules jumping priorities by 100 for each rule to leave space for new rules you might create in the future.
Access	Type of access to apply if the rule matches	Allow or deny	Keep in mind that if an allow rule is not found for a packet, the packet is dropped.

Default Rules

All NSGs have a set of default rules, as shown in Table 4-5 and Table 4-6. These default rules cannot be deleted, but since they have the lowest possible priority, they can be overridden by the rules that you create. The lower the number, the sooner it will take precedence.

The default rules allow and disallow traffic as follows:

- **Virtual network** Traffic originating and ending in a Virtual Network is allowed both in inbound and outbound directions.
- **internet O**utbound traffic is allowed, but inbound traffic is blocked.
- **Load balancer** Allow Azure's load balancer to probe the health of your VMs and role instances. If you are not using a load balanced set, you can override this rule.

TABLE 4-5 Default Inbound Rules

Name	Priority	Source IP	Source Port	Destination IP	Protocol	Access
AllowVNetInBound	65000	VirtualNetwork	*	VirtualNetwork	*	Allow
AllowAzureLoad BalancerInBound	65001	AzureLoadBalancer	*	*	*	Allow
DenyAllInBound	65500	*	*	*	*	Deny

TABLE 4-6 Default Outbound Rules

Name	Priority	Source IP	Source Port	Destination IP	Protocol	Access
AllowVNetOutBound	65000	VirtualNetwork	*	VirtualNetwork	*	Allow
AllowinternetOutBound	65001	*	*	internet	*	Allow
DenyAllOutBound	65500	*	*	*	*	Deny

EXAM TIP

NSG Rules are enforced based on their Priority. Priority values start from 100 and go to 4096. Rules will be read and enforced starting with 100 then 101, 102 etc., until all rules have been evaluated in this order. Rules with the priority "closest" to 100 will be enforced first. For example, if you had an inbound rule that allowed TCP traffic on Port 80 with a priority of 250 and another that denied TCP traffic on Port 80 with a priority of 125, the NSG rule of deny would be put in place. This is because the "deny rule", with a priority of 125 is closer to 100 than the "allow rule", containing a priority of 250.

Default Tags

Default tags are system-provided identifiers to address a category of IP addresses. You can use default tags in the source address prefix and destination address prefix properties of any rule.

There are three default tags you can use:

- **VirtualNetwork (Resource Manager) (VIRTUAL_NETWORK for classic)** This tag includes the Virtual Network address space (CIDR ranges defined in Azure) all connected on-premises address spaces and connected Azure VNets (local networks).
- **AzureLoadBalancer (Resource Manager) (AZURE_LOADBALANCER for classic)** This tag denotes Azure's infrastructure load balancer. The tag translates to an Azure datacenter IP where Azure's health probes originate.
- **internet (Resource Manager) (INTERNET for classic)** This tag denotes the IP address space that is outside the Virtual Network and reachable by public internet. The range includes the Azure owned public IP space.

MORE INFORMATION **MICROSOFT AZURE DATACENTER IP RANGES**

The Microsoft Azure Datacenter IP Ranges can be downloaded at this link: *https://www.microsoft.com/en-us/download/details.aspx?id=41653*. This file contains the IP address ranges (including compute, SQL, and storage ranges) used in the Microsoft Azure datacenters. An updated file is posted weekly that reflects the currently deployed ranges and any upcoming changes to the IP ranges. New ranges appearing in the file are not used in the datacenters for at least one week. By downloading this file once a week, network admins have an updated method to correctly identify services running in Azure. This is often used to whitelist Azure services on corporate firewalls.

Associating NSGs

NSGs are used to define the rules of how traffic is filtered for your IaaS deployments in Azure. NSGs by themselves are not implemented until they are "associated", with a resource in Azure. NSGs can be associated to ARM network interfaces (NIC), which are associated to the VMs, or subnets.

For NICs associated to VMs, the rules are applied to all traffic to/from that Network Interface where it is associated. It is possible to have a multi-NIC VM, and you can associate the same or different NSG to each Network Interface. When NSGs are applied to subnets, rules are applied to traffic to/from all resources connect to that subnet.

EXAM TIP

Understanding the effective rules of NSGs is critical for the exam. Security rules are applied to the traffic by priority in each NSG in the following order:

- **NSG applied to subnet** If a subnet NSG has a matching rule to deny traffic, the packet is dropped.
- **NSG applied to NIC** If VM\NIC NSG has a matching rule that denies traffic, packets are dropped at the VM\NIC, even if a subnet NSG has a matching rule that allows traffic.

Outbound traffic:

- **NSG applied to NIC** If a VM\NIC NSG has a matching rule that denies traffic, packets are dropped.
- **NSG applied to subnet** If a subnet NSG has a matching rule that denies traffic, packets are dropped, even if a VM\NIC NSG has a matching rule that allows traffic.

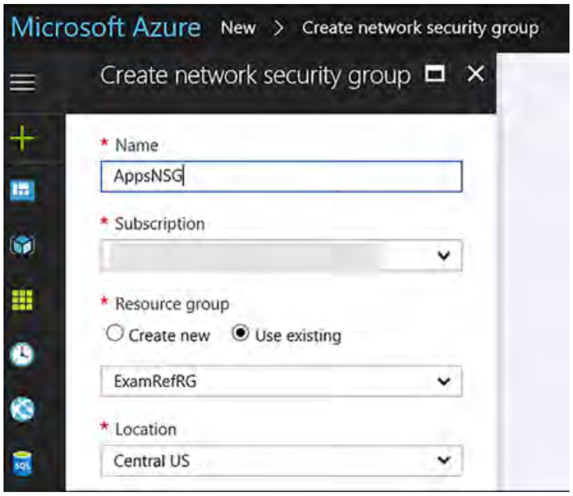
Configuring and Associating NSGs with Subnets

NSGs are a bit different than other types of resources in Azure given they are first created. You must add rules to them (inbound or outbound), and they must be associated to have the desired effect of filtering traffic based on those rules. Remember that NSGs with no rules will have the six default rules covered earlier in this section.

NSGs can be configured and associated with subnets using the Azure portal, PowerShell or the Azure CLI.

Creating an NSG and associating with a subnet using the Azure portal

To create a NSG using the portal, first click New, then Networking, and select network security group. Once the Create Network Security Group blade loads you will need to provide a Name, the Subscription where your resources are located, the resource group for the NSG and the Location (this must be the same as the resources you wish to apply the NSG). In Figure 4-22, the NSG will be created to allow HTTP traffic into the Apps subnet and be named AppsNSG.


Microsoft Azure New > Create network security group

Create network security group

* Name
AppsNSG

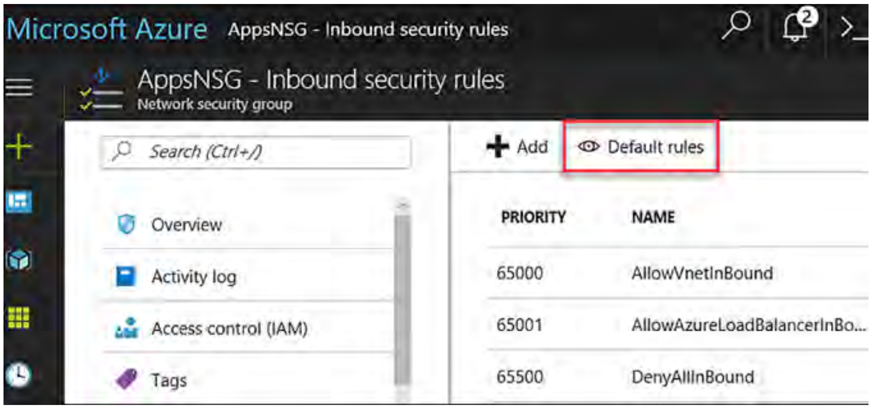
* Subscription

* Resource group
Create new [✓ Use existing]
ExamRefRG

* Location
Central US

FIGURE 4-22 Creating a Network Security Group using the Azure Portal

After AppsNSG is created, the portal opens the Overview blade. Here, you see that the NSG has been created, but there are no inbound or outbound security rules beyond the default rules. In Figure 4-23, the Inbound Security Rules blade of the AppsNSG is shown.


Microsoft Azure AppsNSG - Inbound security rules

AppsNSG - Inbound security rules
Network security group

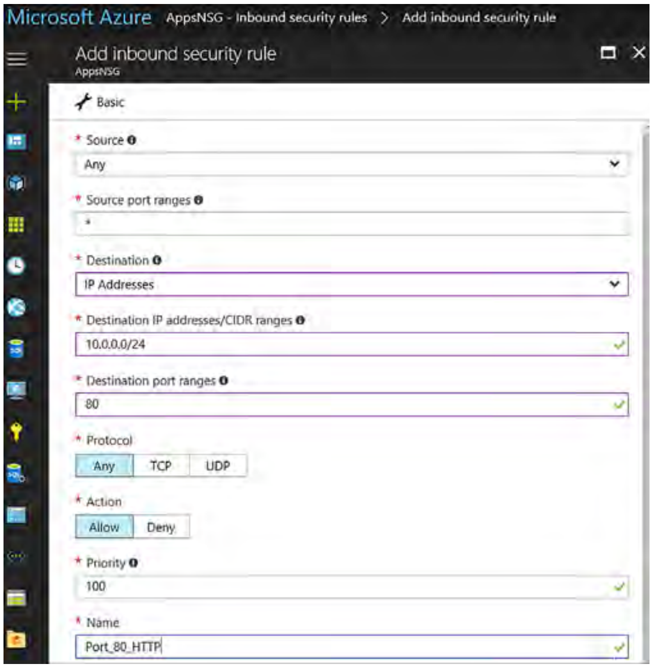
PRIORITY	NAME
65000	AllowVnetInBound
65001	AllowAzureLoadBalancerInBo...
65500	DenyAllInBound

FIGURE 4-23 The Inbound Security Rules showing only the Default Rules

The next step is to create the inbound rule for HTTP. Under the settings area, click on Inbound Security Rules link. The next step will be to click +Add to allow HTTP traffic on Port 80 into the Apps subnet. In the Add inbound security rule blade, configure the following items, and click OK as seen in Figure 4-24.

- **Source** Any
- **Source** Port Ranges *
- **Destination** IP Addresses
- **Destination IP Addresses/CIDR Ranges** The Apps Subnet: 10.0.0.0/24

- **Destination Port Ranges** 80
- **Protocol** TCP
- **Action** Allow
- **Priority** 100
- **Name** Port 80_HTTP


Microsoft Azure AppsNSG - Inbound security rules > Add inbound security rule

Add inbound security rule
AppsNSG

Basic

* Source: Any

* Source port ranges: *

* Destination: IP Addresses

* Destination IP addresses/CIDR ranges: 10.0.0.0/24

* Destination port ranges: 80

* Protocol: [✓ Any] TCP UDP

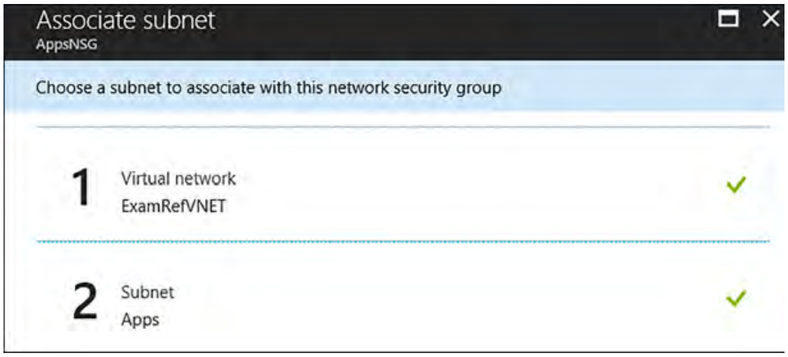
* Action: [✓ Allow] Deny

* Priority: 100

* Name: Port_80_HTTP

FIGURE 4-24 Adding an Inbound Rule to allow HTTP traffic

Once the portal is saved the inbound rule, it will appear in the portal. Review your rule to ensure it has been created correctly. This NSG with its default rules and the newly created inbound rule named Port_80_HTTP are not filtering any traffic. It has yet to be associated with a subnet or a Network Interface, so the rules are currently not in effect. The next task will be to associate it with the Apps subnet. In the Azure portal / Settings, click subnets button, and click +Associate. The portal will ask for two configurations: "Name of the Virtual Network" and the "Name of the subnet". In Figure 4-25, the VNet ExamRefVNET and subnet Apps has been selected.


Associate subnet
AppsNSG

Choose a subnet to associate with this network security group

1 Virtual network
ExamRefVNET ✓

2 Subnet
Apps ✓

FIGURE 4-25 The AppsNSG has been associated with the Apps Subnet

After being saved, the rules of the NSG are now being enforced for all network interfaces that are associated with this subnet. This means that TCP traffic on Port 80 is allowed for all VMs that are connected to this subnet. Of course, you need to have a webserver VM configured and listening on Port 80 to respond, but with this NSG, you have opened the ability for Port 80 traffic to flow to the VMs in this subnet from any other subnet in the world.

IMPORTANT **NSGS**

Remember that NSGs can be associated with network interfaces as well as subnets. For example, if a webserver is connected to this Apps subnet and it didn't have an NSG associated with its network interface, the traffic would be allowed. If the VM had an NSG associated to its network interface, an inbound rule configured exactly like the PORT_80_HTTP rule created here would be required to allow the traffic through. To learn how to work with NSGs associated to network interfaces in Skill 4.3 Configure ARM VM Networking.

Creating an NSG and associating with a subnet using PowerShell

To create an NSG and configure the rules by using PowerShell, you need to use the New-AzureRmNetworkSecurityRuleConfig and New-AzureRmNetworkSecurityGroup PowerShell cmdlets together. In this example, it's assumed that you have run the Login-AzureRmAccount command and have already created a resource group and the Vnet (created from the earlier example of creating a Vnet by using PowerShell).

```
#Build a new Inbound Rule to Allow TCP Traffic on Port 80 to the Subnet
$rule1 = New-AzureRmNetworkSecurityRuleConfig -Name PORT_HTTP_80 `
                                              -Description "Allow HTTP" `
                                              -Access Allow `
                                              -Protocol Tcp `
                                              -Direction Inbound `
                                              -Priority 100 `
                                              -SourceAddressPrefix * `
                                              -SourcePortRange * `
                                              -DestinationAddressPrefix 10.0.0.0/24 `
                                              -DestinationPortRange 80
```

```
$nsg = New-AzureRmNetworkSecurityGroup –ResourceGroupName ExamRefRGPS `
                                       -Location centralus `
                                       -Name "AppsNSG" `
                                       -SecurityRules $rule1
```

After the NSG has been created along with the inbound rule, next you need to associate this with the subnet to control the flow of network traffic using this filter. To achieve this goal, you need to use Get-AzureRmVirtualNetwork and the Set-AzureRmVirtualNetworkSubnetConfig. After the configuration on the subnet has been set, use Set-AzureRmVirtualNetwork to save the configuration in Azure.

```
#Associate the Rule with the Subnet Apps in the Virtual Network ExamRefVNET-PS
$vnet = Get-AzureRmVirtualNetwork –ResourceGroupName ExamRefRGPS –Name ExamRefVNET-PS

Set-AzureRmVirtualNetworkSubnetConfig –VirtualNetwork $vnet `
                                      -Name Apps `
                                      -AddressPrefix 10.0.0.0/24 `
                                      -NetworkSecurityGroup $nsg
Set-AzureRmVirtualNetwork –VirtualNetwork $vnet
```

Creating an NSG and associating with a subnet using the Azure CLI

Creating a NSG using the CLI is a multi-step process just as it was with the portal and PowerShell. The `az network nsg create` command will first be used to create the NSG. Upon creation of the NSG, will be the rule where we will again allow Port 80 to the subnet. This is created using the `az network nsg rule create` command. Upon creation, this will be associated with the Apps subnet on the VNet using the `az network vnet subnet update` command.

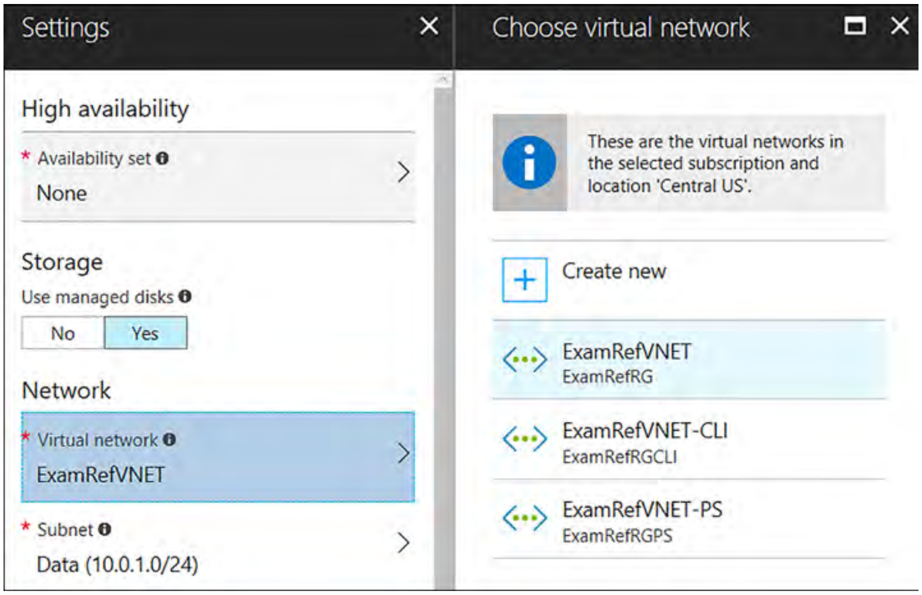
```
# Create the NSG

az network nsg create --resource-group ExamRefRGCLI --name AppsNSG

# Create the NSG Inbound Rule allowing TCP traffic on Port 80

az network nsg rule create --resource-group ExamRefRGCLI --name PORT_HTTP_80
--nsg-name AppsNSG --direction Inbound --priority 100 --access Allow
--source-address-prefix "*" --source-port-range "*" --destination-address-prefix "*"
 --destination-port-range "80" --description "Allow HTTP" --protocol TCP

# Associate the NSG with the ExamRefVNET-CLI Apps Subnet

az network vnet subnet update --resource-group ExamRefRGCLI
--vnet-name ExamRefVNET-CLI --name Apps --network-security-group AppsNSG
```

Deploy a VM into a Virtual Network

VMs can only be deployed into Virtual Networks. VNets provide the networking capabilities that make it possible to benefit from the services of the VM you deploy. The Azure portal allows you to create a VM in an existing Virtual Network, requiring you to specify the subnet. Another option is to create a new VNet while you are creating the new VM.

After a VM is deployed into a VNet, it cannot be moved to another VNet (without deleting it and re-creating), so it is important to consider carefully which VNet the VM should be deployed to during creation.

When creating a VM using the Azure portal, the VNets available to you are filtered to only those in the same subscription and region where you are creating the VM. As seen in Figure 4-26, the selection of the VNet is on Step 3 of creating a VM in the portal after the Basics such as the Name, Location and resource group are selected along with the Size of the VM in Step 2.


Settings

High availability
* Availability set: None

Storage
Use managed disks: No [✓ Yes]

Network
* Virtual network: ExamRefVNET
* Subnet: Data (10.0.1.0/24)

Choose virtual network

These are the virtual networks in the selected subscription and location 'Central US'.

Create new

ExamRefVNET
ExamRefRG

ExamRefVNET-CLI
ExamRefRGCLI

ExamRefVNET-PS
ExamRefRGPS

FIGURE 4-26 The Virtual Network is selected from a list of those in the same subscription and Azure Region.

EXAM TIP

Availability sets are used to inform the Azure fabric that two or more of your VMs are providing the same workload and thus should not be susceptible to the same fault or update domains. If you select an availability set during the creation of a VM in the portal, you can only deploy your VM to the VNet where the other VMs are deployed and the option to create a new VNet is removed.

After you select the VNet where the VM is connected, you are required to specify the subnet. In Figure 4-27, the subnet choices are only those that are within the VNet that was selected previously.

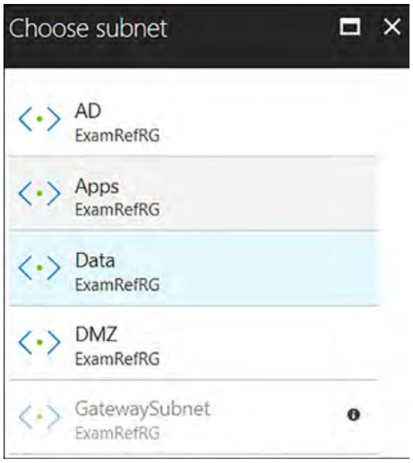


FIGURE 4-27 A subnet is required to be selected when provisioning a VM into a VNet

EXAM TIP

VMs cannot be moved from one VNet to another without deleting and recreating them, but it is possible to move the subnet where a VM is located within the same VNet. This is done by changing the IP configuration of the NIC. If the VM has a static IP address this must be changed to dynamic prior to the move. This is due to the static IP address being outside the address range of the new subnet.

MORE INFORMATION **DEPLOYING VMS TO VIRTUAL NETWORKS**

To learn more about how to deploy VMs to Virtual Networks, refer to Skill 2.1 Deploy Workloads on Azure Resource Manager Virtual Machines.

Implement Application Gateway

Microsoft Azure Application Gateway (App Gateway), is a feature rich dedicated virtual appliance providing application delivery controller (ADC) as a service. This service allows for different types of layer 7 load balancing. App Gateway assists you in optimizing a web farm by offloading CPU intensive SSL termination. It can also provide layer 7 routing capabilities including round robin distribution of incoming traffic, cookie-based session affinity and URL path-based routing.

The App Gateway can host up to 20 websites at the same time, and can be configured as internet facing gateway, internal only gateway, or a combination of both. It also supports the use of Azure services as a backend such as Azure Web Apps and API Gateways.

App Gateway provides health monitoring of backend resources and custom probes to monitor for more specific scenarios. These are used by the App Gateway to know what servers are online and ready for server traffic. This is critical in providing the high-availability required by administrators.

EXAM TIP

App Gateway performs load balancing using a round robin scheme and its Load balancing is accomplished at Layer 7. This means it only handles HTTP(S) and WebSocket traffic. This is different from the Azure load balancer which works at Layer 4 for many different types of TCP and UDP traffic. It can offload SSL Traffic, handle cookie-based session affinity and act as a Web Application Firewall (WAF).

Web Application Firewall (WAF)

A key capability of App Gateway is acting as a web application firewall (WAF). When enabled, the WAF provides protection to web applications from common web vulnerabilities and exploits. These include common web-based attacks such as cross-site scripting, SQL injection attacks and session hijacking.

Cookie-based session affinity

The cookie-based session affinity is a common requirement of many web applications. It is required when your application needs to maintain a user session on the same back-end server. By using gateway-managed cookies, App Gateway will direct all traffic from a user session to the same back-end for processing.

Secure Sockets Layer (SSL) offload

Security is critical for all web applications, and one of the key tasks of web servers is dealing with the overhead of decrypting HTTPS traffic. The App Gateway will take this burden off your web servers by terminating the SSL connection at the Application Gateway and forwarding the request to the server unencrypted. This doesn't mean that the site is now open to hackers as App Gateway will re-encrypt the response before sending it back to the client. It will still be critical to ensure that NSGs are used extensively on the subnets of the VNet where the App Gateway and Web Servers are deployed. This is to ensure the unencrypted traffic can't be seen by others.

End to End SSL

App Gateway supports end to end encryption of traffic. It does this by terminating the SSL connection at the App Gateway and applying routing rules to the traffic. It re-encrypts the packet, and forwards it to the appropriate backend based on the routing rules defined. All responses from the web server go through the same process back to the end user.

URL-based content routing

URL routing allows for directing traffic to different back-end servers based on the content being requested by the user. Typical load balancers will just "spread" the traffic across servers. But with URL routing, you can determine based on the traffic being requested which servers are leveraged. For example, traffic for a folder called /images could be sent to one farm of servers and traffic for folder called /video could be sent to a CDN. This capability reduces the unneeded load on backends that don't serve specific content.

MORE INFORMATION **AZURE APPLICATION GATEWAY**

To read more about the Azure Application Gateway follow this link: *https://docs.microsoft.com/en-us/azure/application-gateway/application-gateway-introduction*.

Application Gateway Sizes

There are three sizes of deployable App Gateways: Small, Medium, and Large. You can create up to 50 application gateways per subscription, and each application gateway can have up to 10 instances each. Each application gateway can consist of 20 http listeners. Table 4-7 shows an average performance throughput for each application gateway instance with SSL offload enabled.

TABLE 4-7 Performance throughput of the App Gateway

Back-end page response	Small	Medium	Large
6K	7.5 Mbps	13 Mbps	50 Mbps
100K	35 Mbps	100 Mbps	200 Mbps

Deployment into Virtual Networks

Application Gateway requires its own subnet. When creating your VNets, ensure you leave enough address space to incorporate one for the App Gateway. The dedicated subnet for your App Gateway will need to be provisioned before creating the App Gateway. Given there is a limit of 50 App Gateways per subscription, this VNet doesn't need to be larger than a /26 CIDR, as this will provide 59 usable Addresses. Also, once an App Gateway is deployed to this subnet, only other App Gateway can be added to it.

Creating an App Gateway using the Azure Portal

This is a three-step process. First is the addition of the dedicated subnet to your VNet. In the example, this subnet was already added and named AppGateway with an address space of 10.0.98.0/26.

Secondly, create the App Gateway using open the Azure portal. As seen in Figure 4-28, New. Then, select Networking followed by Application Gateway.

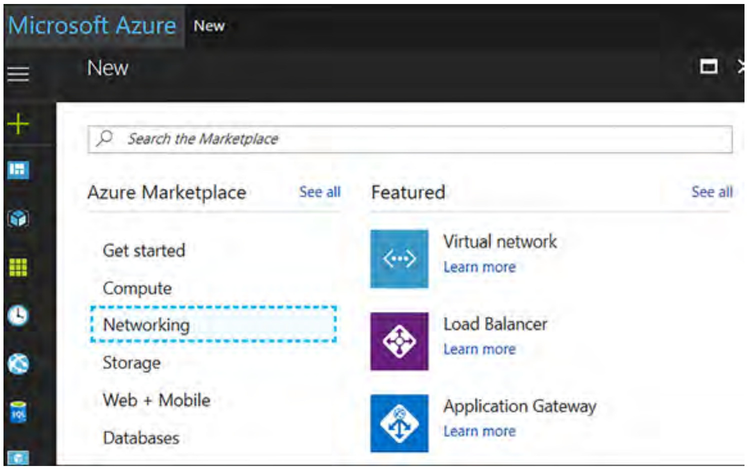
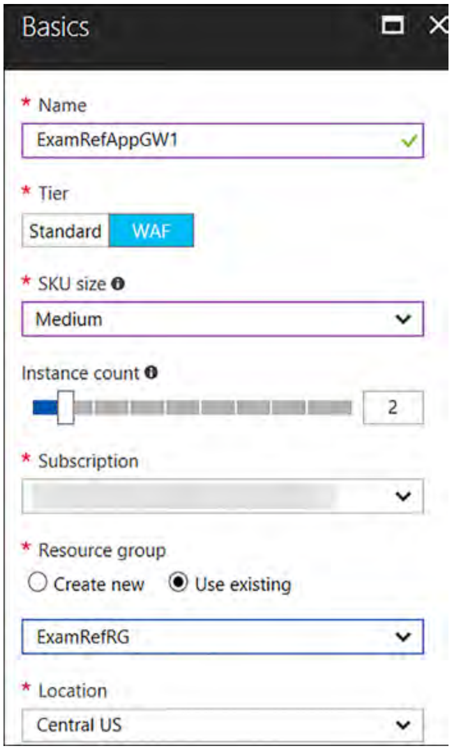


FIGURE 4-28 Creating a New Application Gateway using the Azure Portal

The Create Application Gateway blade opens. Next complete the basics, such as the name, tier (Standard or WAF), size of the App Gateway, and number of instances of the App Gateway, among others, as shown in Figure 4-29.


Basics

* Name: ExamRefAppGW1

* Tier: Standard [✓ WAF]

* SKU size: Medium

Instance count: 2

* Subscription:

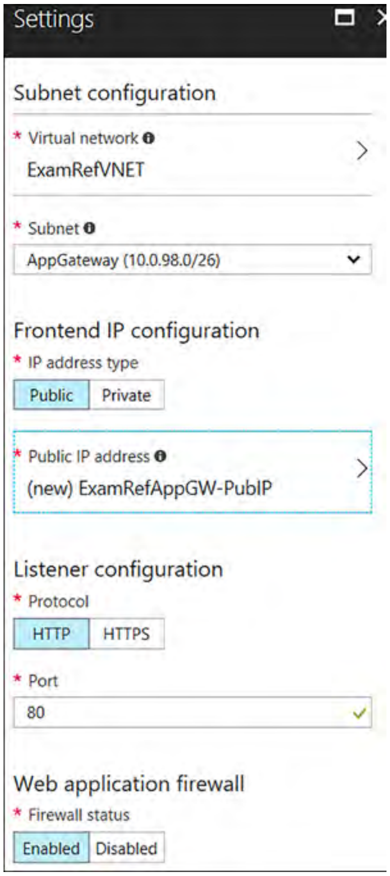
* Resource group: Create new [✓ Use existing]

ExamRefRG

* Location: Central US

FIGURE 4-29 Completed Basics blade for App Gateway

The third step is the Settings blade where critical information is collected in regards to how the App Gateway will be deployed. The first selection is the Virtual Network where it will be deployed. A subnet will be selected next. Remember, the subnet will already have to be created before creating the App Gateway. In Figure 4-30, the App Gateway is being deployed to the ExamRefVNET into a subnet called AppGateway using a /26. Next will be the Frontend IP Configuration (this is important if this App Gateway will be made available to the internet or only the Intranet). Here you can select Public and then create a New Public IP Address. Additional selections will be the Protocol, Port, and WAF configurations.


Settings

Subnet configuration

* Virtual network
ExamRefVNET

* Subnet
AppGateway (10.0.98.0/26)

Frontend IP configuration

* IP address type
[✓ Public] Private

* Public IP address
(new) ExamRefAppGW-PubIP

Listener configuration

* Protocol
[✓ HTTP] HTTPS

* Port
80

Web application firewall

* Firewall status
[✓ Enabled] Disabled

FIGURE 4-30 Completed Settings blade for an internet facing App Gateway

Creating an App Gateway using the PowerShell

Creating an App Gateway using PowerShell is somewhat complex because there are many configurations that are required for this Virtual Appliance. These commands assume that your resource group was already created along with the VNet ExamRefVNET-PS that was created in earlier examples. You can walk through each part of the code to get an idea of how this created.

First you will create a new subnet in the VNet for the App Gateway instances. Remember, this subnet is only to be used by the App Gateway. This code will use the Get-AzureRMVirtualNetwork, Add-AzureRmVirtualNetworksubnetConfig and Set-AzureRmVirtualNetwork cmdlets.

```
# Create a subnet in the ExamRefVNET-PS VNet with the Address Range of 10.0.98.0/26
$vnet = Get-AzureRmVirtualNetwork -ResourceGroupName ExamRefRGPS `
                                  -Name ExamRefVNET-PS
Add-AzureRmVirtualNetworkSubnetConfig -Name AppGateway `
                                      -AddressPrefix "10.0.98.0/26" `
                                      -VirtualNetwork $vnet
Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

The next step is to create the Public IP Address that will be used by the App Gateway. This code uses the New-AzureRMPublicIpAddress cmdlet. It is important to note that you can't use a Static IP Address with the App Gateway.

```
# Create a Public IP address that is used to connect to the application gateway.
$publicip = New-AzureRmPublicIpAddress -ResourceGroupName ExamRefRGPS `
                                       -Name ExamRefAppGW-PubIP `
                                       -Location "Central US" `
                                       -AllocationMethod Dynamic
```

The following commands then used to create the various configurations for the App Gateway. Each of these commands use different cmdlets to load these configurations into variables that are ultimately passed to the New-AzureRmApplicationGateway cmdlet. Upon completion of the App Gateway, the last command will set the WAF configuration.

```
# Create a gateway IP configuration. The gateway picks up an IP address from the
 configured subnet

$vnet = Get-AzureRmvirtualNetwork -Name "ExamRefVNET-PS" -ResourceGroupName
 "ExamRefRGPS"
$subnet = Get-AzureRmVirtualNetworkSubnetConfig -Name "AppGateway" -VirtualNetwork
 $vnet
$gipconfig = New-AzureRmApplicationGatewayIPConfiguration -Name "AppGwSubnet01"
 -Subnet $subnet

# Configure a backend pool with the addresses of your web servers. You could add
 pool members here as well.
$pool = New-AzureRmApplicationGatewayBackendAddressPool -Name "appGatewayBackendPool"

# Configure backend http settings to determine the protocol and port that is used
 when sending traffic to the backend servers.
$poolSetting = New-AzureRmApplicationGatewayBackendHttpSettings -Name
                 "appGatewayBackendHttpSettings" `
                 -Port 80 `
Protocol Http `
 -CookieBasedAffinity Disabled `
 -RequestTimeout 30

# Configure a frontend port that is used to connect to the application gateway
 through the Public IP address
```

```
$fp = New-AzureRmApplicationGatewayFrontendPort -Name frontendport01 `
                                                -Port 80
# Configure the frontend IP configuration with the Public IP address created earlier.
$fipconfig = New-AzureRmApplicationGatewayFrontendIPConfig -Name fipconfig01 `
 -PublicIPAddress $publicip

# Configure the listener.  The listener is a combination of the front-end IP
configuration, protocol, and port
$listener = New-AzureRmApplicationGatewayHttpListener -Name listener01 `
                                                  -Protocol Http `
                                                  -FrontendIPConfiguration $fipconfig `
                                                  -FrontendPort $fp

# Configure a basic rule that is used to route traffic to the backend servers.
$rule = New-AzureRmApplicationGatewayRequestRoutingRule -Name rule1 `
                                                         -RuleType Basic `
                                                         -BackendHttpSettings
$poolSetting `
                                                         -HttpListener $listener `
                                                         -BackendAddressPool $pool

# Configure the SKU for the application gateway, this determines the size and
 whether WAF is used.
$sku = New-AzureRmApplicationGatewaySku -Name "WAF_Medium" `
                                        -Tier "WAF" `
                                        -Capacity 2
# Create the application gateway
New-AzureRmApplicationGateway -Name ExamRefAppGWPS `
                              -ResourceGroupName ExamRefRGPS `
                              -Location "Central US" `
                              -BackendAddressPools $pool `
                              -BackendHttpSettingsCollection $poolSetting `
                              -FrontendIpConfigurations $fipconfig  `
                              -GatewayIpConfigurations $gipconfig `
                              -FrontendPorts $fp `
                              -HttpListeners $listener `
                              -RequestRoutingRules $rule `
                              -Sku $sku `
                              -WebApplicationFirewallConfiguration

# Set WAF Configuration to Enabled
$AppGw = Get-AzureRmApplicationGateway -Name ExamRefAppGWPS -ResourceGroupName
 ExamRefRGPS
Set-AzureRmApplicationGatewayWebApplicationFirewallConfiguration -ApplicationGateway
 $AppGw `
                                                   -Enabled $True `
                                                   -FirewallMode "Detection" `
                                                   -RuleSetType "OWASP" `
                                                   -RuleSetVersion "3.0"
```

Creating the App Gateway using the Azure CLI

You will use the az network command with different arguments. The first step will be to create the AppGateway subnet. Next is the Public IP Address followed by the App Gateway Virtual Appliance. The command to create the App Gateway is quite large, but this is common when using the CLI. After the App Gateway has been provisioned, you will then enable the WAF using the `az network application gateway` command.

```
# Create a subnet for the App Gateway in the ExamRefVNET-CLI VNet with the Address
 Range of 10.0.98.0/26
az network vnet subnet create -g ExamRefRGCLI --vnet-name ExamRefVNET-CLI
                  -n AppGateway --address-prefix 10.0.98.0/26

# Create a Public IP address that is used to connect to the application gateway.
az network public-ip create -g ExamRefRGCLI -n ExamRefAppGW-PubIP

# Create the App gateway named ExamRefAppGWCLI
az network application-gateway create -n "ExamRefAppGWCLI" -g "ExamRefRGCLI"
--vnet-name "ExamRefVNET-CLI" --subnet "AppGateway" --capacity 2 --sku WAF_Medium
 --http-settings-cookie-based-affinity Disabled --http-settings-protocol Http
--frontend-port 80 --routing-rule-type Basic --http-settings-port 80
--public-ip-address "ExamRefAppGW-PubIP"

# Enable the WAF
az network application-gateway waf-config set -g "ExamRefRGCLI" -n "ExamRefAppGWCLI"
 --enabled true --rule-set-type OWASP --rule-set-version 3.0
```

Skill 4.2: Design and implement multi-site or hybrid network connectivity

The term hybrid cloud is used in many ways across the IT industry. Typically, hybrid cloud means to connect your local datacenters to the cloud and run workloads in both locations. This section covers connecting VNets to other VNets, as well as options for connecting your on-premises network to the Azure cloud. There are a range of connection types that can be leveraged for the many scenarios that you as a cloud administrator could face on the exam.

This skill covers how to:

- Choose the appropriate solution between ExpressRoute, Site-to-Site, and Point-to-Site
- Choose the appropriate gateway
- Identity supported devices and software VPN solutions
- Identify network prerequisites
- Configure Virtual Networks and multi-site Virtual Networks
- Implement Virtual Network peering and service chaining

Choose the appropriate solution between ExpressRoute, Site-to-Site and Point-to-Site

When connecting your on-premises network to a VNet, there are three options that provide connectivity for various use cases. These include Point-to-Site virtual private network (VPN), Site-to-Site VPN, and Azure ExpressRoute.

Point-to-Site (P2S) virtual private network (VPN)

This connection type is between a single PC connected to your network and Azure VPN Gateway running over the internet. Sometimes referred to as a "VPN Tunnel," this on-demand connection is initiated by the user and secured by using a certificate. The connection uses the SSTP protocol on port 443 to provide encrypted communication over the internet between the PC and the VNet. The P2S connection is very easy to setup because it requires little or no change to your existing network. The latency for a point-to-site VPN is unpredictable because the traffic traverses the internet.

P2S connections are useful for remote employees or those that only want to establish connectivity when they need it and can disconnect from the Azure VNet when they are finished with their tasks. In Figure 4-31, depicts an example of developers or testers that only need to connect to the Azure VNet when they are writing code or performing tests on their applications.

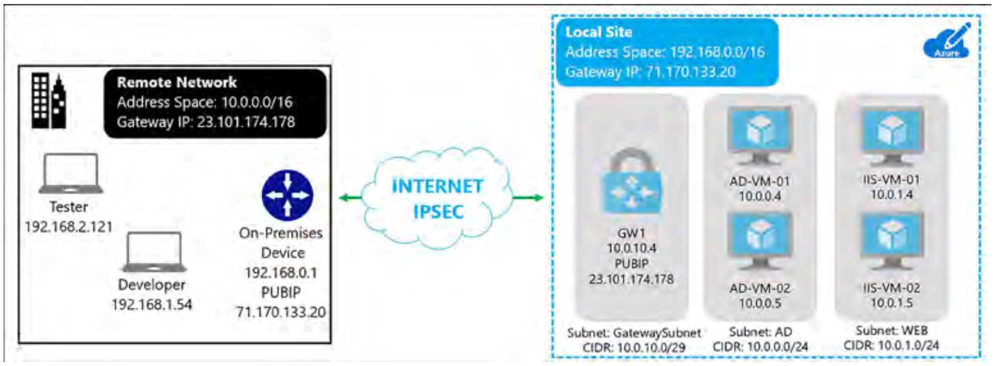


FIGURE 4-31 Remote Developers and Tester connecting to Azure VNet using P2S

Site-to-Site (S2S) VPN

S2S connections are durable methods for building cross-premises and hybrid configurations. S2S connections are established between your VPN on-premises device and an Azure VPN Gateway. This connection type enables any on-premises devices you authorize to access VMs and services that are running in an Azure VNet. The connection is known as an IPSec VPN that provides encrypted network traffic crossing over the internet between your on-premises VPN device and the Azure VPN Gateway. The secure encryption method used for these VPN tunnels is IKEv2. Just as with P2S VPNs, the latency for S2S VPNs is unpredictable because the network connection is an internet connection. The on-premises VPN device is required to have a static Public IP address assigned to it and it cannot be located behind a NAT. An example of a S2S VPN is shown in Figure 4-32.

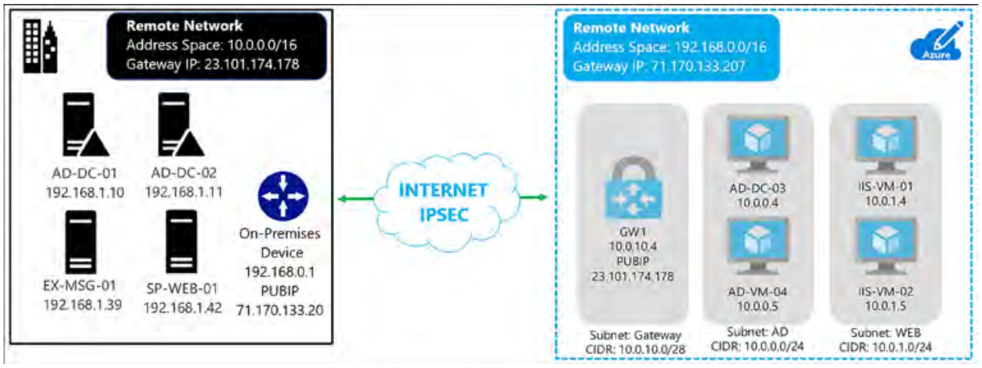


FIGURE 4-32 S2S VPN connection between Azure and On-Premises

There is a variation of this S2S network where you create more than one VPN connection from your VPN Gateway typically connecting to multiple on-premises sites. This is known as a Multi-Site S2S connection. When working with multiple connections, you must use a route-based VPN type. Because each VNet can only have one VPN Gateway, all connections through the gateway share the available bandwidth. In Figure 4-33, you see an example of a network with three sites and two VNets in different Azure regions.

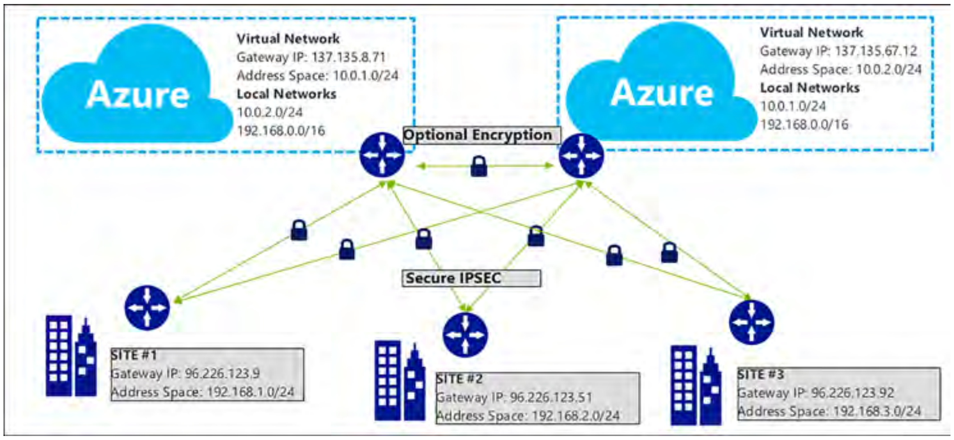


FIGURE 4-33 Multi-Site S2S Network with three locations and two Azure VNets

S2S VPNs should be used for connecting to Azure on a semi-permanent or permanent basis based on your plans for the cloud. These connections are always on and reliable (only as much as your internet connection though). If there are issues with your internet provider or on the public internet infrastructure, your VPN tunnel could go down or run slowly from time to time. Mission critical workloads running in Azure in a hybrid-cloud configuration should use ExpressRoute with a S2S as a backup.

EXAM TIP

P2S connections do not require an on-premises static, public-facing IP address or a VPN device. S2S connections do required a static Public IP address to ensure zero downtime and they must not be behind a NAT. P2S and S2S connections can be initiated through the same Azure VPN Gateway.

Azure ExpressRoute

ExpressRoute lets you connect your on-premises networks into the Microsoft cloud over a private connection hosted by a Microsoft ExpressRoute provider. With ExpressRoute, you can establish connections to Microsoft cloud services, such as Microsoft Azure, Office 365, and Dynamics 365.

ExpressRoute is a secure and reliable private connection. Network traffic does not egress to the internet. The latency for an ExpressRoute circuit is predictable because traffic stays on your provider's network and never touches the internet.

Connectivity can be from a Multiprotocol Label Switching (MPLS), any-to-any IPVPN network, a point-to-point ethernet network, or a virtual cross-connection through a connectivity provider at a co-location facility. Figure 4-34 shows the options for connecting to ExpressRoute.

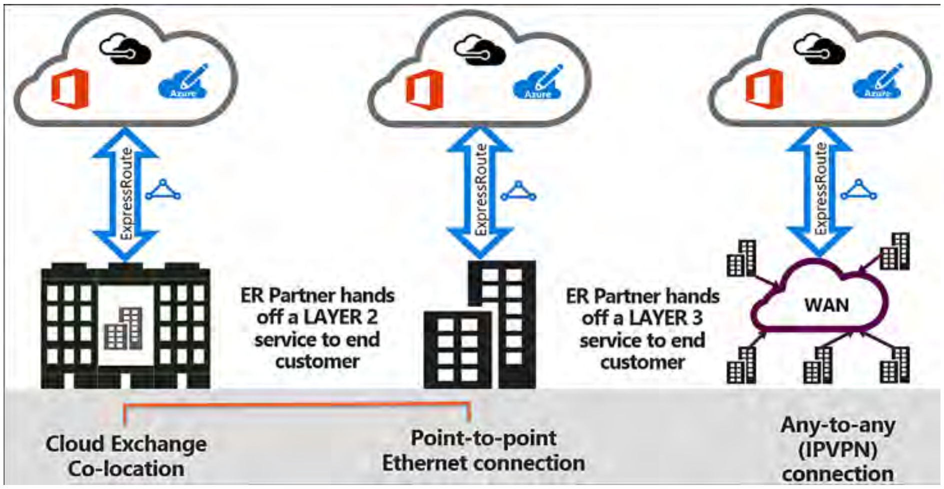

Azure
ExpressRoute
ER Partner hands off a LAYER 2 service to end customer
ER Partner hands off a LAYER 3 service to end customer
WAN
Cloud Exchange Co-location
Point-to-point Ethernet connection
Any-to-any (IPVPN) connection

FIGURE 4-34 Examples of ExpressRoute Circuits

ExpressRoute is only available in certain cities throughout the world, so it is important to check with a provider to determine its availability.

MORE INFORMATION **EXPRESSROUTE**

To locate the cities where ExpressRoute can be used and the providers that support the connection review the following article: *https://docs.microsoft.com/en-us/azure/expressroute/expressroute-locations*.

An ExpressRoute connection does not use a VPN Gateway, although it does use a Virtual Network gateway as part of its required configuration. In an ExpressRoute connection, the Virtual Network gateway is configured with the gateway type 'ExpressRoute,' rather than 'VPN'

To connect to Azure using ExpressRoute, set up and manage the routing or contract with a service provider to manage the connection. ExpressRoute uses the BGP routing protocol to facilitate the connections between the routers on the network. There is support for the use of private and public Autonomous System Numbers (ASN), and public or private IPs depending upon which part of the Microsoft cloud that you are connecting to with ExpressRoute.

EXAM TIP

BGP Routing is supported on S2S VPNs and ExpressRoute. BPG is the routing protocol that the routers use to track each other's networks and helps to facilitate the movement of network traffic from one network to another. Public Autonomous System Numbers (ASN), are globally unique numbers on the internet just like Public IP addresses. They are managed by groups around the world, such as the American Registry for internet Names (ARIN) in the US and others like it.

Each ExpressRoute circuit has two connections to two Microsoft edge routers from your network edge. Microsoft requires dual BGP connections from your edge to each Microsoft edge router. You can choose not to deploy redundant devices or ethernet circuits at your end; however, connectivity providers use redundant devices to ensure that your connections are handed off to Microsoft in a redundant manner. Figure 4-35 shows a redundant connectivity configuration.

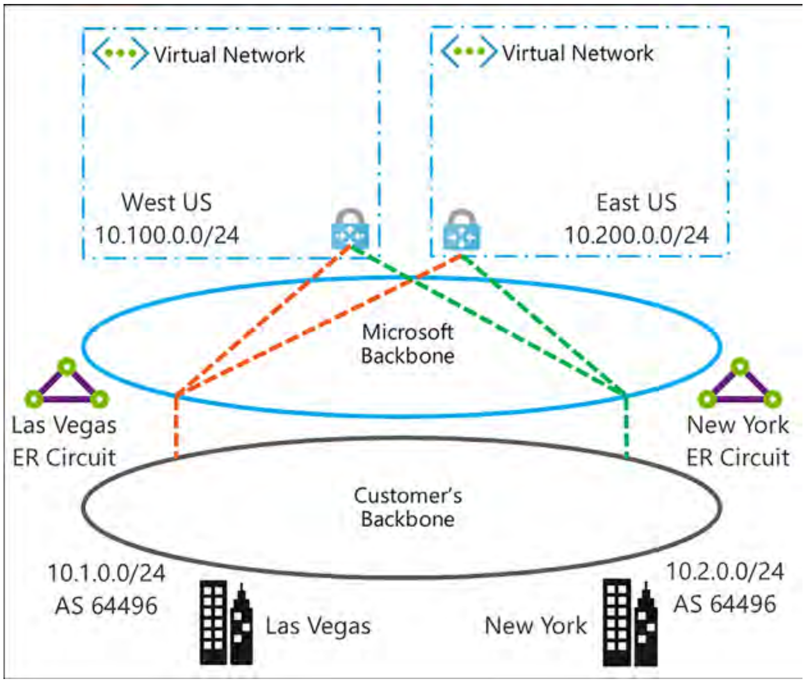


FIGURE 4-35 Multiple Cities Connected to ExpressRoute in Two Azure Regions

As seen in Figure 4-36, ExpressRoute has three routing domains: Azure Public, Azure Private, and Microsoft. They are configured as an active-active load sharing configuration. Azure Services are assigned categories based on what they provide and have different IP Address schemes within the Azure regions.

- **Azure Private Peering** Azure compute services, namely virtual machines (IaaS) and cloud services (PaaS), that are deployed within a Virtual Network can be connected through the private peering domain. The private peering domain is a trusted extension of your core network into Microsoft Azure.
- **Azure Public Peering** Services such as Azure Storage, SQL databases, and websites are offered on Public IP addresses. You can privately connect to services hosted on Public IP addresses, including VIPs of your cloud services, through the public peering routing domain.
- **Microsoft Peering** ExpressRoute provides private network connectivity to Microsoft cloud services. Software as a Service offerings, like Office 365 and Dynamics 365, were created to be accessed securely and reliably via the internet.

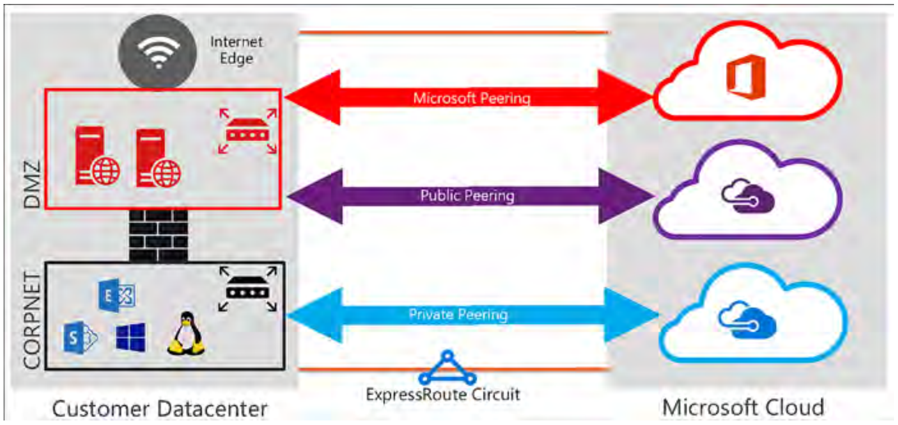


FIGURE 4-36 ExpressRoute Routing Domains

ExpressRoute can be purchased in different speeds and in one of two modes: Metered or Unlimited.

- **Metered** All inbound data transfer is free of charge, and all outbound data transfer is charged based on a pre-determined rate. Users are also charged a fixed monthly port fee (based on high availability dual ports).
- **Unlimited** All inbound and outbound data transfer is free of charge. Users are charged a single fixed monthly port fee (based on high availability dual ports).

EXAM TIP

ExpressRoute supports the following speeds: 50 Mbps, 100 Mbps, 200 Mbps, 500 Mbps, 1 Gbps, 2 Gbps, 5 Gbps, and 10 Gbps.

There is also a premium add-on that can be enabled if your network is global enterprise in nature. The following features are added to your ExpressRoute circuit when the premium add-on is enabled:

- Increased routing table limit from 4000 routes to 10,000 routes for private peering.
- More than 10 VNets can be connected to the ExpressRoute circuit.
- Connectivity to Office 365 and Dynamics 365.
- Global connectivity over the Microsoft core network. You can now link a VNet in one geopolitical region with an ExpressRoute circuit in another region.

EXAM TIP

If the ExpressRoute Premium Add-on is enabled, you can link a VNet created in a different part of the world to your ExpressRoute circuit. For example, you can link a VNet in Europe West to an ExpressRoute circuit created in Silicon Valley. This is also true for your public peering resources. An example of this would be connecting to your SQL Azure database located in Europe West from a circuit in New York.

Choose the appropriate gateway

A VPN Gateway is a type of Virtual Network gateway that sends encrypted traffic over a public connection to an on-premises location. You can also use VPN Gateways to send encrypted traffic between Azure VNets over the Azure Backbone network. To send encrypted network traffic between your Azure VNET and an on-premises datacenter, you must create a VPN Gateway for your VNet.

Each VNet can only have one VPN Gateway. However, you can create multiple connections to the same VPN Gateway. An example of this is a multi-site connection configuration. When you create multiple connections to the same VPN Gateway, all VPN tunnels, including Point-to-Site VPNs, share the bandwidth that is available for the gateway.

A Virtual Network gateway is composed of two or more VMs that are deployed to a specific subnet called the GatewaySubnet. The VMs that are in the GatewaySubnet are created upon creation of the VPN Gateway. VPN Gateway VMs are configured to contain routing tables and gateway services specific to that VPN Gateway. You can't directly configure the VMs that are part of the VPN Gateway, and you should never deploy additional resources to the GatewaySubnet.

When you create a VNet gateway using the gateway type **vpn**, it creates a specific type of VPN Gateway that encrypts traffic. The Gateway SKU that you select determines how powerful the provisioned VMs will be to handle the amount of aggregate throughput (network traffic for all devices sending and received through). It is important to note the Basic VPN Gateway is only recommended for very small networks with only a few users. The VPN options are captured in Table 4-8.

TABLE 4-8 VPN Gateway options

SKU	S2S/VNet-to-VNet Tunnels (Maximum)	P2S Connections (Maximum)	Aggregate Throughput Benchmark
VpnGw1	30	128	650 Mbps
VpnGw2	30	128	1 Gbps
VpnGw3	30	128	1.25 Gbps
Basic	10	128	100 Mbps

Identify support devices and software VPN solutions

To facilitate a Site-to-Site (S2S), cross-premises VPN connection using a VPN Gateway, a VPN device is required to configured on-premises. These S2S connections can be used to create a hybrid solution, or whenever you want to secure connections between your on-premises networks and your VNets.

In partnership with device vendors, Microsoft has a validated a set of standard VPN devices proven to be compatible. These devices range from routers and firewalls to software that can be run on Linux or Windows servers. All of the devices in the device guide should work with the Azure VPN Gateways. If you have a device that is not listed, there is guidance from Microsoft on the standards that are supported and by using this information, it is possible to connect to the Azure network.

MORE INFORMATION **APPROVED LIST OF DEVICES**

To locate the approved list of devices and configuration guides, read the following article: *https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpn-devices*.

Identify network prerequisites

Connecting to the Azure cloud does require proper planning, and there are some very important steps to look at prior to configuring VNets and hybrid connections to Azure.

Here are some considerations that you should start with when determining how to connect your network to Azure:

- **IP address spaces** It is critical that your IP address spaces don't overlap with existing networks or those that you are planning for in the future. This is complicated even further when considering that you are most likely building more than one VNet in Azure. It is best to use an offline IP Address Management (IPAM), tool to ensure that all your networks are accounted for before you build anything.
- **Public IP addresses & AS numbers** If you plan on using S2S or ExpressRoute connections to Azure, it is important to have public registered IP addresses. If you are using ExpressRoute or BGP with S2S connections having a registered AS number for your company is the proper way to configure the network.
- **VPN devices** Making sure that you have an approved VPN device is the best bet for creating VPN connections to Azure. You should test these devices by creating a VPN to a test network to ensure that it works properly as a part of your Azure pilot project.

- **Estimated network throughput** It is important to understand the amount of network traffic that goes from your site to Azure over the VPN Gateway. You need this information to select the VPN Gateway SKU to provision. This is also useful for deciding on the size of the ExpressRoute circuit if you chose to go with this type of configuration.
- **Subscriptions** It is also important to consider the number of subscriptions that you have and if they have the same Azure AD Tenet associated with them. This impacts your network design because there are limits to the various configurations and the number of possible connections, depending upon how you configured the subscriptions for your company.

Implement Virtual Network peering service chaining

VNet peering allows you to connect two Azure Resource Manager (ARM), VNets located in the same region together without complex integration. The peered VNets must have non-overlapping IP address spaces. The address spaces in either VNet cannot be added to or deleted from a VNet once a VNet is peered with another VNet.

Once they are peered, the VNets appear as one network and the VM in the peered VNets can communicate with each other directly. The traffic between VMs is routed through the Microsoft backbone network in the same way that traffic is routed between VM in the same VNet through private IP addresses.

You can peer VNets that exist in two different subscriptions, if a privileged user of both subscriptions authorizes the peering, and the subscriptions are associated to the same Azure Active Directory tenant. There is also a limit of 50 VNet peerings per subscription.

Figure 4-37 shows two VNets in the North Central region peered together. Notice their IP address spaces do not overlap as this would make the peering impossible.

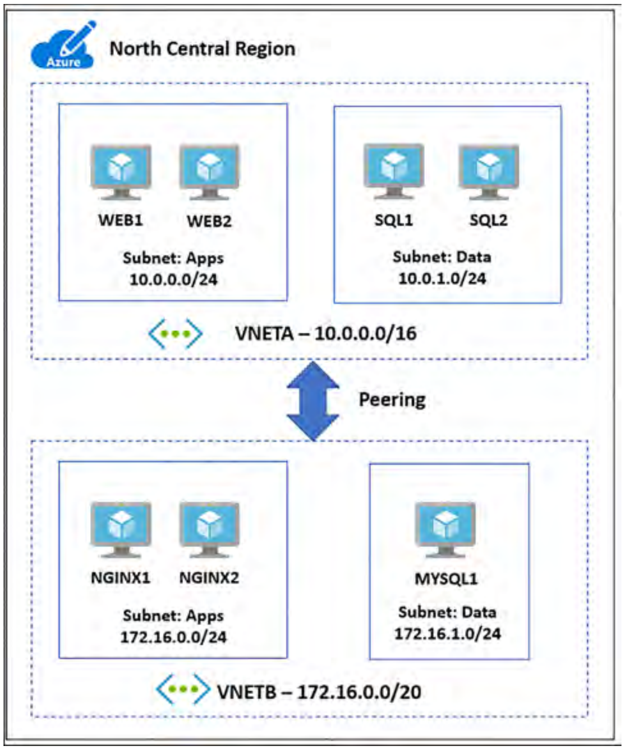


FIGURE 4-37 VNet peering between two networks in the North Central Region

Network Traffic flowing through VNet peerings is private, as it never touches any gateways or the internet. This connection is low-latency and high-bandwidth between resources in different VNets. The same speed considerations for VMs on the local VNet apply (based on the VM size). Once Vnet peering is enabled, VMs on each VNet will have the ability to use resources in one VNet from another.

EXAM TIP

It is possible to peer Virtual Networks created through the Azure Resource Manager to a Virtual Network created using the Azure classic deployment model. Two Azure classic VNets cannot be peered together.

IMPORTANT AZURE VNET PEERING

Azure VNet peering between VNets in different Azure regions is in Public Preview.

It is important to understand that VNet peering is between two Virtual Networks. There is no derived transitive relationship across the peerings. If you peer VNetA to VNetB and then peer VNetB with VNetC, VNetA is not peered to VNetC. Figure 4-38 shows you a functional architecture of this and describes how VNetA is not peered to VNetC.

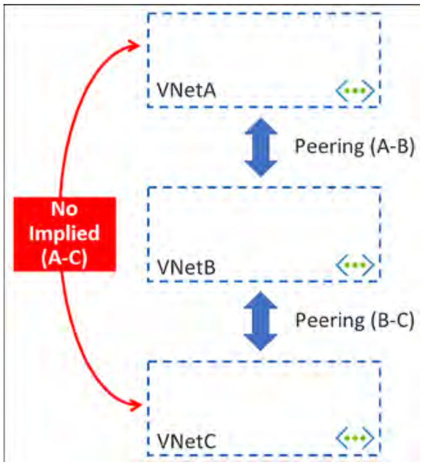


FIGURE 4-38 Vnet Peerings do not have a transitive relationship

Service Chaining

You can configure user-defined routes that point to VMs in peered VNets as the "next hop," IP address to enable service chaining. Service chaining enables you to direct traffic from one VNet to a virtual appliance in a peered Virtual Network through user-defined routes. Figure 4-39 provides a view of a network where service chaining is implemented.

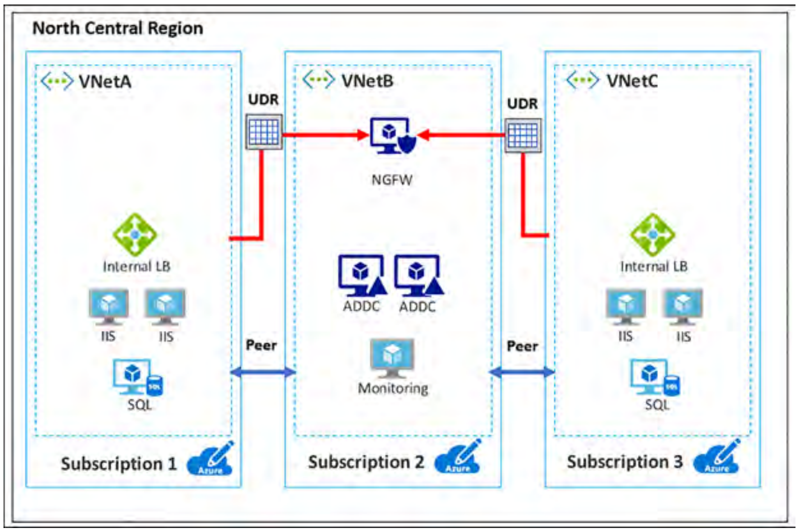


FIGURE 4-39 Service chaining allows for the use of common services across VNet Peerings

Configure Virtual Network and Multi-Site Virtual Networks

You can connect VNets to each other, enabling resources connected to either VNet to communicate with each other across VNets. You can use either or both of the following options to connect VNets to each other:

- **Peering** Enables resources connected to different Azure VNets within the same Azure region to communicate with each other. The bandwidth and latency across the VNets is the same as if the resources were connected to the same VNet.
- **VNet-to-VNet connection** Enables resources connected to different Azure VNets within the same, or different Azure regions. Unlike peering, bandwidth is limited between VNets because traffic must flow through an Azure VPN Gateway.

Using VNet-to-VNet connections allows for connecting Azure regions to each other via the Microsoft backbone. This configuration allows for always on connections between the Azure datacenters.

Peering and VNet-to-VNet connections can be used together to enable all networks to see each other. Peering would be setup between one or more VNets within the same Azure regions and then one of these networks would have a VNet-to-VNet VPN setup. The Allow Gateway Transit must be turned on for these packets to be leveraged.

Creating a VNet-to-VNet connection across Azure Regions using a VPN Gateway using the Azure Portal

Figure 4-40 shows a diagram of what you should create. VNETA and VNETB are peered in the North Central Azure regions. Another VNet, VNETC, should be deployed to the North Europe region and then connected to VNETB via a S2S VPN.

The following steps capture the basic process of connecting these networks together:

- Create gateway subnets on each VNETB and VNETC
- Provision virtual network gateways on VNETB and VNETC
- Create connections between the two networks
- Configure VNETA to VNET B peering to Use Remote Gateway
- Configure VNETB to VNETA peering to Allow Gateway Transit

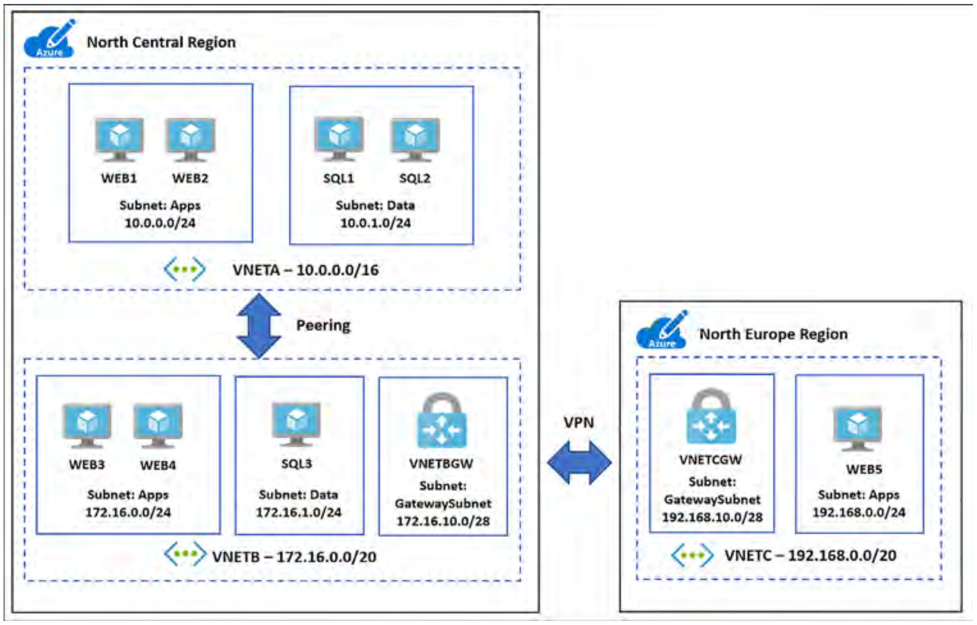
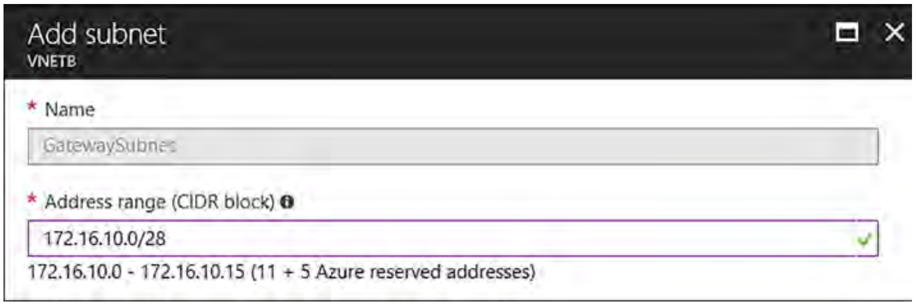


FIGURE 4-40 Multi-Region VNet deployment using Peering in and VPN Gateway over the Microsoft Backbone

Create GatewaySubnets on each VNETB and VNETC

Using the portal, navigate to each of the VNets, and click the subnets link under settings. From the subnets blade select the +Gateway subnet and assign an address space using a /28 CIDR, as seen in Figure 4-41 and Figure 4-42.


Add subnet
VNETB

* Name

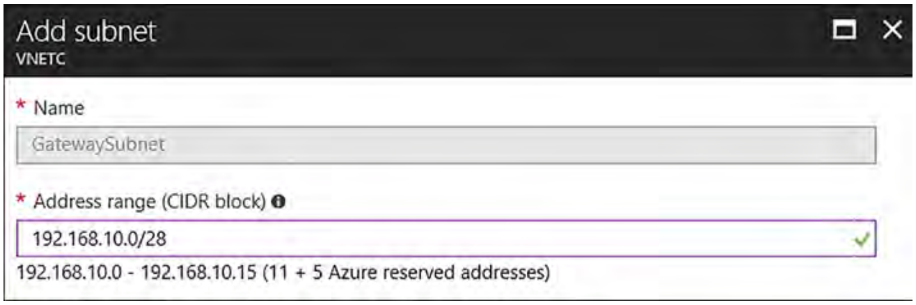
GatewaySubnet

* Address range (CIDR block)

172.16.10.0/28

172.16.10.0 - 172.16.10.15 (11 + 5 Azure reserved addresses)

FIGURE 4-41 Adding a GatewaySubnet to VNETB


Add subnet
VNETC

* Name

GatewaySubnet

* Address range (CIDR block)

192.168.10.0/28

192.168.10.0 - 192.168.10.15 (11 + 5 Azure reserved addresses)

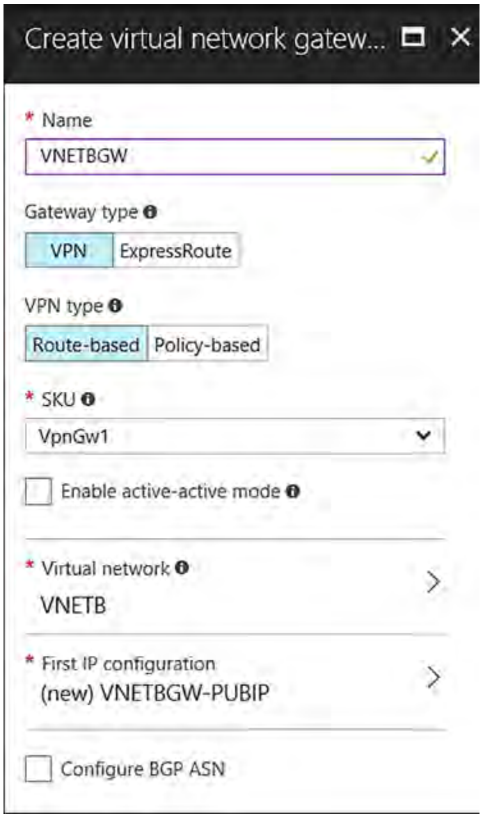
FIGURE 4-42 Adding a GatewaySubnet to VNETC

Provision Virtual Network Gateways on VNETB and VNETC

As shown in Figure 4-43 and Figure 4-44, provision the VPN Gateways by using the Azure portal. Click New, Networking, and then select Virtual Network Gateway.

Complete the following information for VNETB:

- **Name** VNETBGW
- **Gateway type** VPN
- **VPN Type** Route-based
- **SKU** VpnGw1
- **Virtual Network** VNETB
- **First IP Configuration** Create New, VNETBGW-PUBIP
- **Location** North Central US


Create virtual network gatew...

* Name

VNETBGW

Gateway type

[✓ VPN] ExpressRoute

VPN type

[✓ Route-based] Policy-based

* SKU

VpnGw1

☐ Enable active-active mode

* Virtual network

VNETB

* First IP configuration

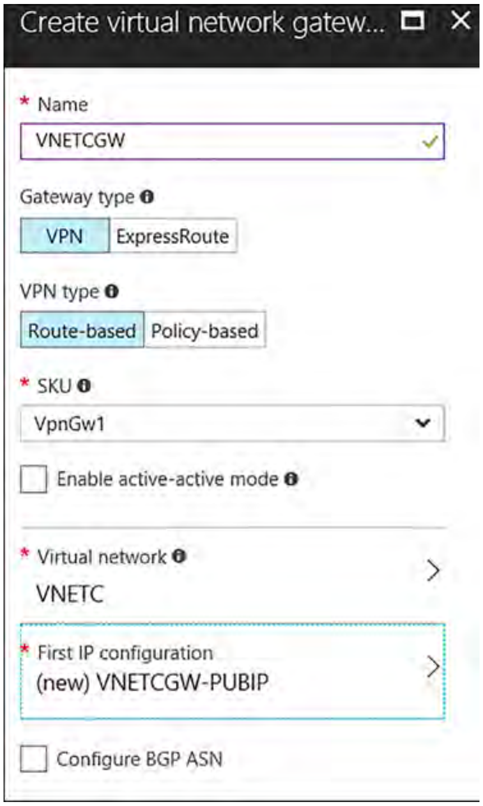
(new) VNETBGW-PUBIP

☐ Configure BGP ASN

FIGURE 4-43 Creating the Azure VPN Gateway for VNETB

Complete the following information for VNETC:

- **Name** VNETCGW
- **Gateway type** VPN
- **VPN Type** Route-based
- **SKU** VpnGw1
- **Virtual Network** VNETC
- **First IP Configuration** Create New, VNETCGW-PUBIP
- **Location** North Europe US



Create virtual network gatew...

* Name
VNETCGW

Gateway type
VPN [✓] ExpressRoute

VPN type
Route-based [✓] Policy-based

* SKU
VpnGw1

☐ Enable active-active mode

* Virtual network
VNETC

* First IP configuration
(new) VNETCGW-PUBIP

☐ Configure BGP ASN

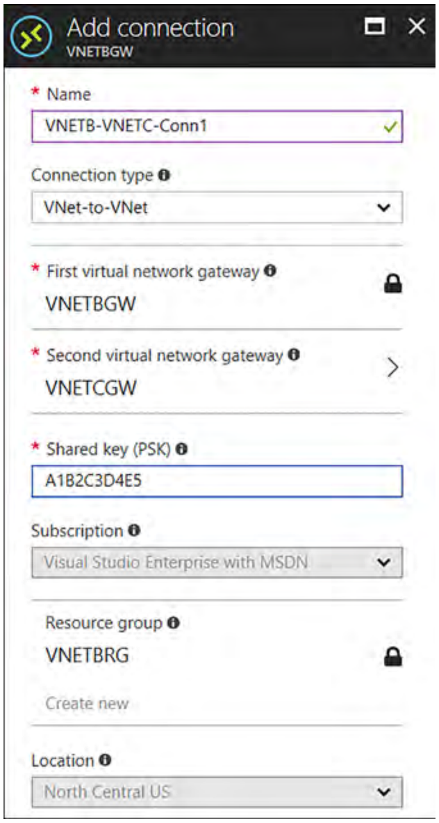
FIGURE 4-44 Creating the Azure VPN Gateway for VNETC

Create Connections between the two Networks

After you provision the VPN Gateways, create two connections in Azure to bring up the VPN tunnel. To create this object, open VNETB in the Azure portal, as shown in Figure 4-45. Under Settings, locate Connections and click it to open. When the Connections blade opens, click +Add.

Complete the VNETB Add connection blade by using the following inputs:

- **Name** VNETB-VNETC-Conn1
- **Connections** Vnet-to-Vnet
- **Second Virtual Network Gatewa** VNETCGW
- **Shared Key** A1B2C3D4E5 (any unique value matching on both sides)


Add connection
VNETBGW

* Name
VNETB-VNETC-Conn1

Connection type
VNet-to-VNet

* First virtual network gateway
VNETBGW

* Second virtual network gateway
VNETCGW

* Shared key (PSK)
A1B2C3D4E5

Subscription
Visual Studio Enterprise with MSDN

Resource group
VNETBRG

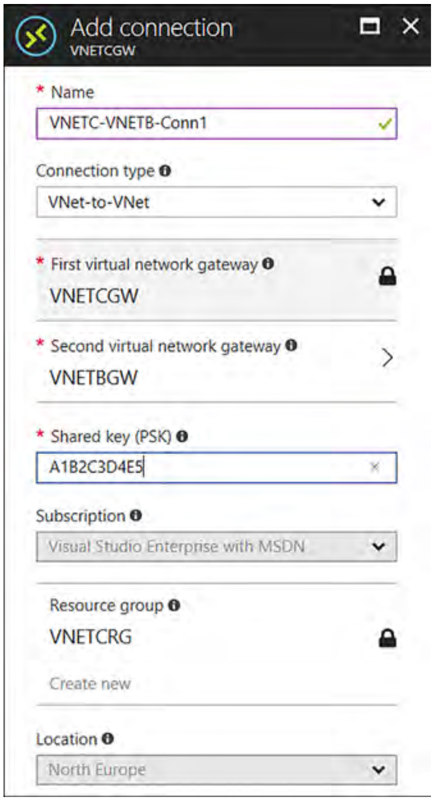
Location
North Central US

FIGURE 4-45 Creating the Connection between VNETB to VNETC

This process needs to be repeated for VNETC. Open VNETC in the Azure portal, as shown in Figure 4-46. Under Settings, locate Connections and click it to open. When the Connections blade opens, click +Add.

Complete the VNETC Add connection blade by using the following inputs:

- **Name** VNETC-VNETB-Conn1
- **Connections** Vnet-to-Vnet
- **Second Virtual Network Gateway** VNETBGW
- **Shared Key** A1B2C3D4E5 (any unique value matching on both sides)


Add connection
VNETCGW

* Name
VNETC-VNETB-Conn1

Connection type
VNet-to-VNet

* First virtual network gateway
VNETCGW

* Second virtual network gateway
VNETBGW

* Shared key (PSK)
A1B2C3D4E5

Subscription
Visual Studio Enterprise with MSDN

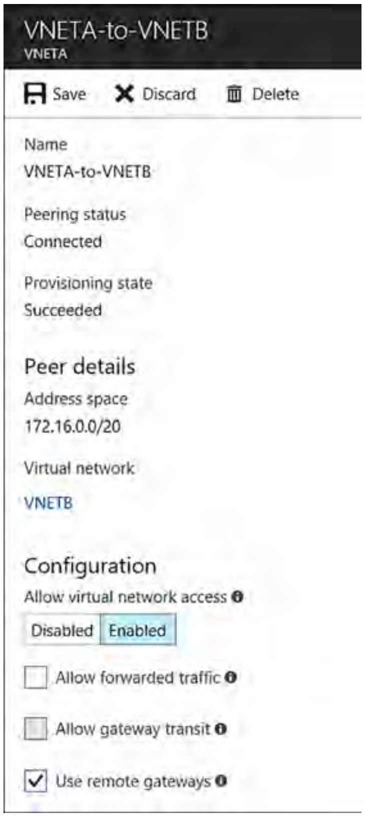
Resource group
VNETCRG
Create new

Location
North Europe

FIGURE 4-46 Creating the Connection between VNETC to VNETB

Configure VNETA to VNETB Peering to Use Remote Gateway

To allow packets from VNETA to cross the VPN Gateway configured on VNETB, reach VNETC, you must make a change to the VNETA-to-VNETB peering by opening VNETA in the portal to locate the peerings link. Then open the VNETA-to-VNETB, and select Use Remote Gateways in the Configuration section, as seen in Figure 4-47.



Name
VNETA-to-VNETB

Peering status
Connected

Provisioning state
Succeeded

Peer details
Address space
172.16.0.0/20

Virtual network
VNETB

Configuration
Allow virtual network access
Disabled [✓ Enabled]

☐ Allow forwarded traffic

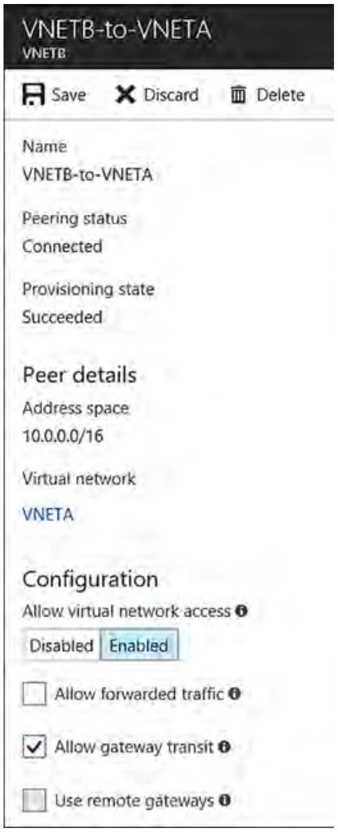
☐ Allow gateway transit

☑ Use remote gateways

FIGURE 4-47 Enabling the Use remote gateways option for the VNETA-to-VNETB Peering

Configure VNETB to VNETA Peering to Allow Gateway Transit

To allow packets from VNETA to VNETC you must make a configuration change to the VNETB-to-VNETA peering. To do so, open the VNETB in the portal and locate the peerings blade. Select the VNEB-to-VNETA connection followed by the Allow Gateway Transit option in the Configuration section, as seen in Figure 4-48.


VNETB-to-VNETA
VNETB

Save | Discard | Delete

Name
VNETB-to-VNETA

Peering status
Connected

Provisioning state
Succeeded

Peer details

Address space
10.0.0.0/16

Virtual network
VNETA

Configuration

Allow virtual network access
Disabled [✓ Enabled]

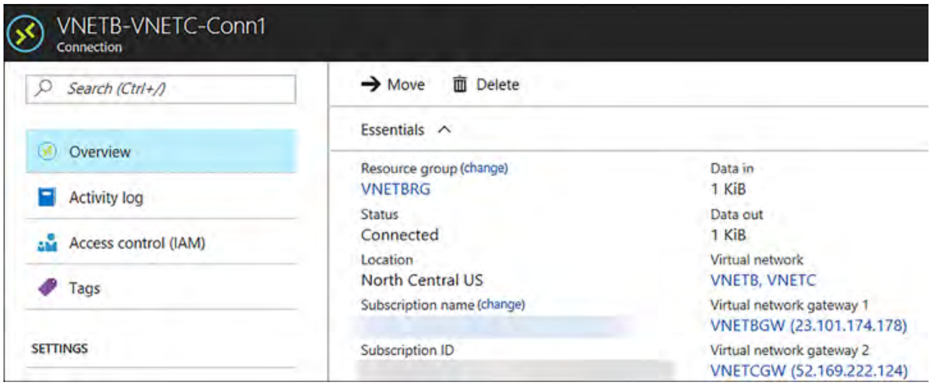
☐ Allow forwarded traffic

☑ Allow gateway transit

☐ Use remote gateways

FIGURE 4-48 Enabling the Allow gateway transit option for the VNETB-to-VNETA peering

After the connection objects are created, it might take a few minutes for the connection to come up. Figure 4-49 shows the status of the connection object by looking on the Overview blade of VNETB-VNETC-Conn1.


VNETB-VNETC-Conn1
Connection

Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
SETTINGS

Move | Delete

Essentials

Resource group (change)	Data in
VNETBRG	1 KiB
Status	Data out
Connected	1 KiB
Location	Virtual network
North Central US	VNETB, VNETC
Subscription name (change)	Virtual network gateway 1
[illegible]	VNETBGW (23.101.174.178)
Subscription ID	Virtual network gateway 2
[illegible]	VNETCGW (52.169.222.124)

FIGURE 4-49 Status of the Connection between VNETB and VNETC shown as Connected

Gateways and On-premises Connectivity

Any VNet can have its own VPN Gateway and use it to connect to an on-premises network. When VNets are peered, this doesn't change. You can also configure VNet- -VNet connections using VPN Gateways. Even the VNet(s) are peered to other VNet(s).

When both options for VNet interconnectivity are configured, the traffic between the VNets flows through the peering configuration (that is, through the Azure backbone).

When VNets are peered in the same region, you can configure the VPN Gateway in the peered VNet as a method to point traffic to an on-premises network. In this case, the VNet using a remote gateway unfortunately cannot have its own gateway. A VNet can have only one gateway. The gateway can either be a local or remote gateway. In Figure 4-50, you see VNetA and VNetB located in the West Europe region peered together with VNetA providing VPN services to both networks. In this configuration, all three networks can reach each other.

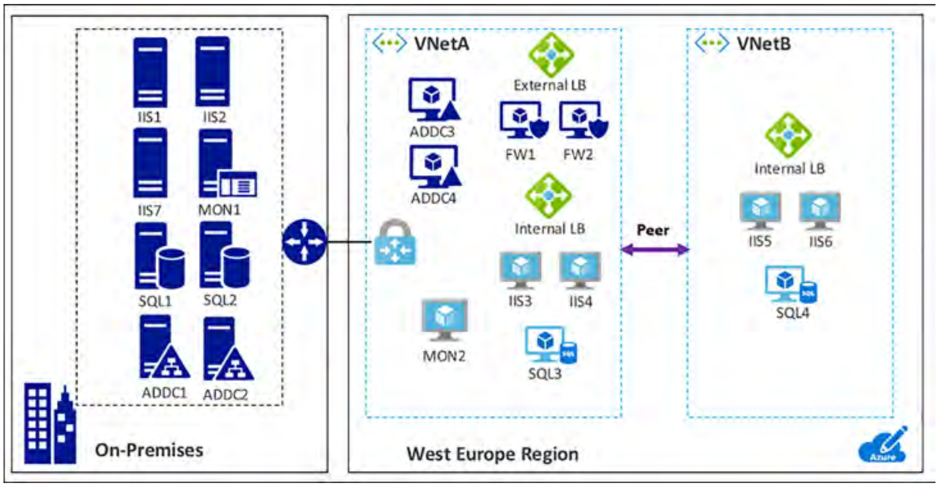


FIGURE 4-50 Gateway Transit allows peered VNets to connect with networks that are across a VPN Gateway

Skill 4.3: Configure ARM VM Networking

Arguably, VMs are the most widely deployed compute in the Public cloud. Azure provides a very deep and rich set of networking configuration possibilities for VMs. These are important to understand from the perspective of the VM, as many configurations that are possible at the VNet level are also possible at the VM level. In this skill, you will study these configurations. Many configurations are performed at the Network Interface level of VMs including public IP Addresses, private IP Addresses, network security groups, and DNS settings. Others focus more on the use of VMs as Web Servers integrate with VMs along with the Azure load balancer and the App Gateway.

This skill covers how to:

- Configure Private Static IP addresses
- Public IPs
- DNS at the network interface level
- Network Security Groups (NSGs)
- User Defined Routes (UDRs) with IP Forwarding
- External and Internal Load Balancing with HTTP and TCP Health probes
- Direct Server Return
- Design and Implement Application Gateway

Configure Private Static IP Addresses

VMs in Azure use TCP/IP to communicate with services in Azure, other VMs you have deployed in Azure, on-premises networks, and the internet.

There are two types of IP addresses you can use in Azure:

- **Public IP addresses** Used for communication with the internet
- **Private IP addresses** Used for communication within an Azure virtual network (Vnet), and your on-premises network when you use a VPN Gateway to build a hybrid network

Private IP addresses allow Azure resources to communicate with other resources in a virtual network or an on-premises network through a VPN Gateway or ExpressRoute circuit, without using an internet-reachable IP address.

Using the Azure Resource Manager, a private IP address is associated to the following types of Azure resources:

- VM network interfaces
- Internal load balancers (ILBs)
- Application gateways

Private IP addresses are created with an IPv4 or IPv6 address. VMs cannot communicate between private IPv6 addresses on a Vnet, but can communicate inbound to a private IPv6 address from the internet when using an internet-facing load balancer.

There are two methods used to assign private IP addresses: dynamic or static. The default allocation method is dynamic, where the IP address is automatically allocated from the resource's subnet (using an Azure DHCP server). This IP address can change when you stop and start the resource.

EXAM TIP

IPv4 address assignments can be either dynamic or static. Private IPv6 addresses can only be assigned dynamically.

A private IP address is allocated from the subnet's address range that the network interface is attached to. The address range of the subnet itself is a part of the Virtual Network's address space.

You can set the allocation method to static to ensure the IP address remains the same. When you specify static, you specify a valid IP address that is part of the resource's subnet.

Static private IP addresses are commonly used for:

- Virtual machines that act as domain controllers or DNS servers
- Resources that require firewall rules using IP addresses
- Resources accessed by other apps/resources through an IP address

All VMs on a Vnet are assigned a private IP address. If the VM has multiple NICs, a private IP address is assigned to each one. You can specify the allocation method as either dynamic or static for a NIC.

All Azure VMs are assigned Azure Managed DNS servers by default, unless custom DNS servers are assigned. These DNS servers provide internal name resolution for VMs within the same Vnet.

When you create a VM, a mapping for the hostname to its private IP address is added to the Azure DNS servers. If a VM has multiple network interfaces, the hostname is mapped to the private IP address of each NIC. VMs assigned Azure DNS servers can resolve the hostnames of all VMs within the same Vnet to their private IP addresses.

MORE INFORMATION **DNS SERVERS TO NICS**

To learn more about assigning custom DNS servers to NICs, see the DNS at the network interface (NIC) level in Skill 4.3.

Internal load balancers (ILB) & Application Gateways

Private IP addresses can be assigned to the front-end configuration of an internal load balancer (ILB), or an App Gateway. This IP becomes the internal endpoint, accessible only to the resources within the Vnet, and any remote networks that are connected with the proper network routing in place. You can assign either a dynamic or static private IP address to the front-end configuration.

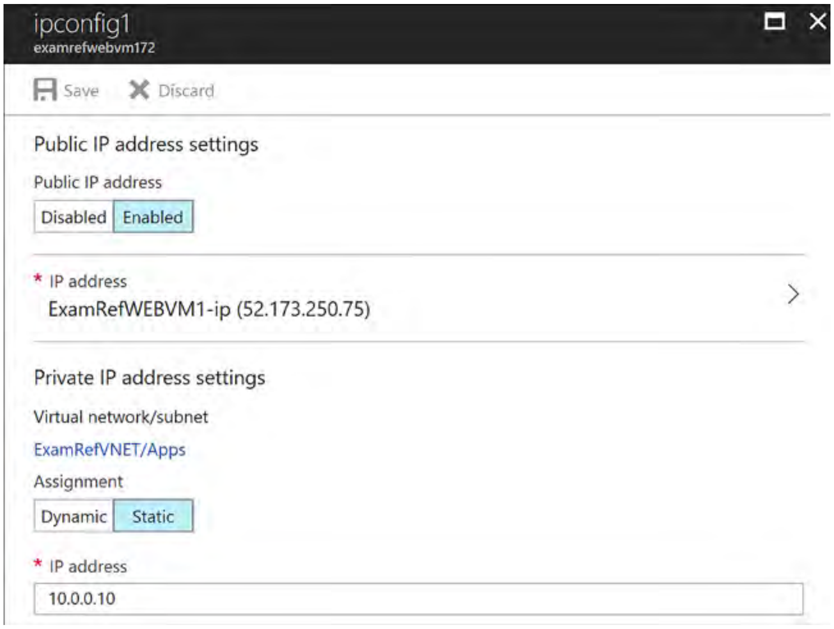
In Table 4-9, the various resources, their associations, and the type of IP allocation methods (dynamic or static) are captured.

TABLE 4-9 IP allocation methods

Resource	Association	Supports Dynamic	Support Static
Virtual Machine	Network Interface	Yes	Yes
Internal load balancer	Front-End Config	Yes	Yes
App Gateway	Front-End Config	Yes	Yes

Enabling static private IP addresses on VMs with the Azure portal

The Network Interface of a VM holds the configurations of the private IP address. This is known as IP Configuration. You can either add new IP Configurations to a NIC or update it from dynamic to static. Using the portal, locate the NIC for the VM that you wish to have a Static IP Address. Once the blade loads for the NIC, click on IP Configurations, then select the IP Configuration you wish to update, as seen in Figure 4-51. Here, you can update the private IP address settings Assignment to be Static, and assign the IP Address. This Address must be within the address range of the subnet where the NIC is located and not currently in use unless you are going to use the address the NIC is already assigned to.



Public IP address settings

Public IP address: Disabled [✓ Enabled]

* IP address: ExamRefWEBVM1-ip (52.173.250.75)

Private IP address settings

Virtual network/subnet: ExamRefVNET/Apps

Assignment: Dynamic [✓ Static]

* IP address: 10.0.0.10

FIGURE 4-51 Assigning a Static Private IP Address to a NIC

Enabling static private IP addresses on VMs with PowerShell

When updating an existing NIC to use a static IP address, use two PowerShell cmdlets: Get-AzureRmNetworkInterface and Set-AzureRmNetworkInterface. First, use the Get-AzureRm-NetworkInterface to locate the NIC that should be assigned the static IP followed by updating the configuration to be static with its IP address. To save this to Azure, use the Set-AzureRm-NetworkInterface cmdlet.

```
# Update existing NIC to use a Static IP Address and set the IP
$nic=Get-AzureRmNetworkInterface -Name examrefwebvm1892 -ResourceGroupName ExamRefRGPS
```

```
$nic.IpConfigurations[0].PrivateIpAllocationMethod = "Static"
$nic.IpConfigurations[0].PrivateIpAddress = "10.0.0.5"
Set-AzureRmNetworkInterface -NetworkInterface $nic
```

Enabling static private IP addresses on VMs with the Azure CLI

By using the Azure CLI to update a NIC to a static private IP address, use one simple command, az network NIC ip-config with the update argument. Again, the name of the NIC and the resource group are required along with the IP configuration. Remember you are updating the IP configuration properties of the NIC resource.

```
# Update existing nic to use a Static IP Address and set the IP
az network nic ip-config update --name ipconfig1 --nic-name examrefwebvm1400 --resource-
group ExamRefRGCLI --private-ip-address 10.0.1.5
```

Public IP Address

Public IP addresses allow Azure resources to communicate with internet and public-facing Azure services. Public IP addresses can be created with an IPv4 or IPv6 address. Only internet-facing load balancers can be assigned a Public IPv6 address.

A Public IP address is an Azure Resource that has its own properties, in the same way a VM or VNet is a Resource. Some of the resources you can associate a Public IP address resource with include:

- Virtual machine via network interfaces
- internet-facing load balancers
- VPN Gateways
- Application gateways

Like private IP addresses, there are two methods an IP address is allocated to an Azure Public IP address resource: dynamic or static. The default allocation method is dynamic. In fact, an IP address is not allocated at the time the Public IP address resource is created by the Azure fabric.

EXAM TIP

Dynamic Public IP addresses are released when you stop (or delete) the resource. After being released from the resource, the IP address will be assigned to a different resource by Azure. If the IP address is assigned to a different resource while your resource is stopped, once you restart the resource, a different IP address will be assigned. If you wish to retain the IP address, the Public IP address should be changed to static, and that is assigned immediately and never changed.

If you change the allocation method to static, you as the administrator cannot specify the actual IP address assigned to the Public IP address resource. Azure assigns the IP address from a pool of IP addresses in the Azure region where the resource is located.

Static Public IP addresses are commonly used in the following scenarios:

- When you must update firewall rules to communicate with your Azure resources.
- DNS name resolution, where a change in IP address would require updating host or A records.
- Your Azure resources communicate with other apps or services that use an IP address-based security model.
- You use SSL certificates linked to an IP address.

One unique property of the Public IP address is the DNS domain name label. This allows you to create a mapping for a configured, fully qualified domain name (FQDN) to your Public IP address. You must provide an Azure globally unique host name that consists of 3-24 alpha-numeric characters, and then Azure adds the domains, creating a FQDN.

EXAM TIP

When you add a DNS Name to your Public IP address, the name will follow this pattern: hostname.region.cloudapp.azure.com. For example, if you create a public IP resource with contosowebvm1 as a DNS Name, and the VM was deployed to a VNet in the Central US region, the fully-qualified domain name (FQDN) of the VM would be: contosowebvm1.centralus.cloudapp.azure.com. This DNS name would resolve on the public internet as well as your VNet to the Public IP address of the resource. You could then use the FQDN to create a custom domain CNAME record pointing to the Public IP address in Azure. If you owned the contoso.com domain, you could create a CNAME record of www.contoso.com to resolve to contosowebvm1.centralus.cloudapp.azure.com. That DNS name would resolve to your Public IP Address assigned by Azure. The client traffic would be directed to the public IP Address associated with that name.

Table 4-10, shows the specific property through which a Public IP address can be associated to a top-level resource and the possible allocation methods (dynamic or static) that can be used.

TABLE 4-10 Public IP Address associations

Top-level resource	IP Address association	Dynamic	Static
Virtual machine	Network interface	Yes	Yes
internet-facing load balancer	Front-end configuration	Yes	Yes
VPN Gateway	Gateway IP configuration	Yes	No
Application gateway	Front-end configuration	Yes	No

Virtual machines

You can associate a Public IP address with any VM by associating it to its NIC. Public IP addresses, by default, are set to dynamic allocation, but this can be changed to static.

IMPORTANT **VM AND PUBLIC IP ADDRESSES**

When a VM is assigned a public IP Address, it is exposed to the public internet. The only protection from attackers would include network security groups (NSGs) associated with the NIC or the subnet where the VM resides. The administrator of the VM could implement OS based Firewalling techniques providing additional security.

internet-facing Load Balancers

Public IP addresses (either a dynamic or static) can be associated with an Azure load balancer by assigning it to the front-end configuration. The Public IP address then becomes the load balancer's virtual IP address (VIP). It is possible to assign multiple Public IP addresses to a load balancer front end which enables multi-VIP scenarios like a multi-site environment with SSL-based websites.

VPN Gateways

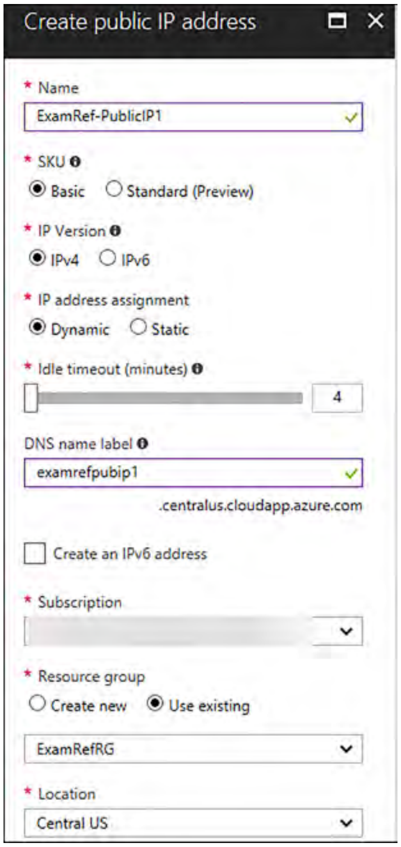
An Azure VPN Gateway connects an Azure VNet to other Azure VNets or to an on-premises network. A Public IP address is required to be assigned to the VPN Gateway to enable it to communicate with the remote network. You can only assign a dynamic Public IP address to a VPN Gateway.

Application gateways

You can associate a Public IP address with an Azure App Gateway by assigning it to the gateway's frontend configuration. This Public IP address serves as a load-balanced VIP. You can only assign a dynamic Public IP address to an application gateway frontend configuration.

Creating a Public IP Address using the Azure Portal

Creating a new Public IP Address is a simple process when using the portal. Click New, and then search for Public IP Address in the Marketplace. Like all resources in Azure, some details will be required including the Name of the resource, IP Version, assignment or allocation method, DNS name label, subscription, resource group and location/region. The location is critical, as an IP Address must be in the same location/region as the resource where you want to assign it. Figure 4-52 shows the Azure Create Public IP Address Blade.


Create public IP address

* Name
ExamRef-PublicIP1

* SKU
[● Basic] ○ Standard (Preview)

* IP Version
[● IPv4] ○ IPv6

* IP address assignment
[● Dynamic] ○ Static

* Idle timeout (minutes)
4

DNS name label
examrefpubip1
.centralus.cloudapp.azure.com

☐ Create an IPv6 address

* Subscription

* Resource group
○ Create new [● Use existing]

ExamRefRG

* Location
Central US

FIGURE 4-52 Creating a Public IP Address in the Azure Portal

Creating a Public IP Address using the PowerShell

When creating a new Public IP address by using PowerShell, the New-AzureRmPublicIpAddress cmdlet is employed. In this script, variables are created for reuse of the code. This command assumes that your resource group is already created.

```
# Creating a Public IP Address
# Set Variables
$publicIpName = "ExamRef-PublicIP1-PS"
$rgName = "ExamRefRGPS"
$dnsPrefix = "examrefpubip1ps"
$location = "centralus"

# Create the Public IP
New-AzureRmPublicIpAddress -Name $publicIpName `
                           -ResourceGroupName $rgName `
                           -AllocationMethod Static `
                           -DomainNameLabel $dnsPrefix `
                           -Location $location
```

Creating a Public IP Address using the Azure CLI

When creating a new Public IP address by using the Azure CLI, only one command is required. The command, `az network public-ip` with the `create` argument. You need to provide details with respect to the name, resource group, DNS lab, and where it is dynamic or static.

```
# Creating a Public IP Address
az network public-ip create -g ExamRefRGCLI -n ExamRef-PublicIP1-CLI --dns-name
examrefpubip1cli --allocation-method Static
```

DNS at the Network Interface (NIC) Level

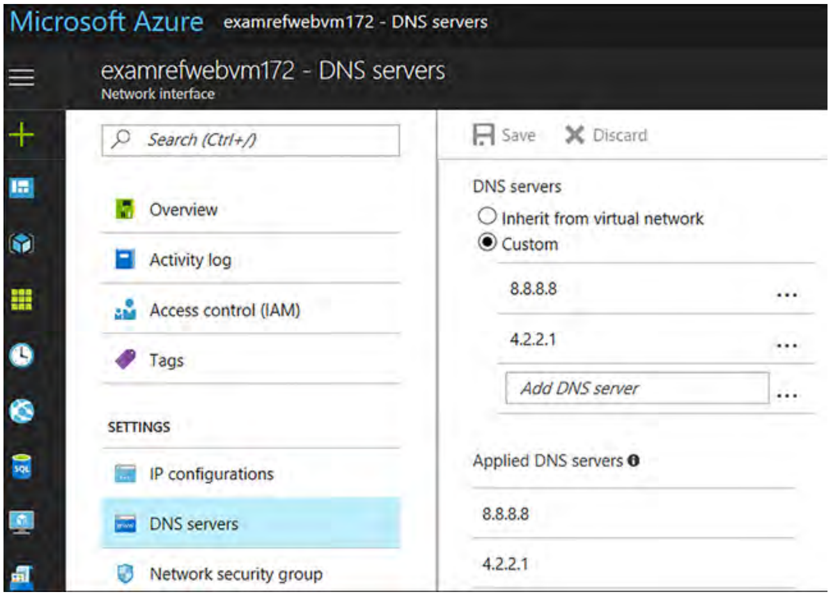
By default, Azure DNS is configured when creating a VM. VMs that are connected to the VNet use this service for name resolution inside the VNet and on the public internet. Along with the resolution of public DNS names, Azure provides internal name resolution for VMs that reside within the same VNet. Within a VNet, the DNS suffix is consistent, so the FQDN is not needed. This means that VMs on the same VNet using the Azure DNS Server can connect directly via their host names.

Just as it is possible to configure your own Customer Managed DNS Servers for a VNet, this configuration is also possible at the NIC level. When using your own DNS servers, Azure provides the ability to specify multiple DNS servers per NIC.

The DNS Servers configuration for a Network Interface on a VM can be completed using the Azure portal, PowerShell or Azure CLI.

Configure DNS Settings on Network Interfaces using the Azure Portal

To configure the DNS Servers on a Network Interface using the Azure portal, open the NIC associated with the VM requiring the custom settings, as seen in Figure 4-53. Next, click on the DNS Servers link in the Settings menu. You can then enter the DNS Servers you wish for this VM to use. In the example, the two DNS Servers are well known, and they are publicly available on the internet but different than the VNet.



DNS servers
Inherit from virtual network [✓ Custom]

8.8.8.8
4.2.2.1
Add DNS server

Applied DNS servers
8.8.8.8
4.2.2.1

FIGURE 4-53 Custom DNS Servers for Network Interface configured using the Portal

Configure DNS Settings on Network Interfaces using the PowerShell

When creating a new VNet, you can specify the DNS Configure the DNS Servers using PowerShell. The Set-AzureRmNetworkInterface along with the Get-AzureRmNetworkInterface cmdlets will be used together to make the change to an existing Network Interface. The first line of PowerShell will get the existing interface and its configurations followed by using the Clear and Add Methods that will update the PowerShell pipeline with new configuration that will be passed to the Set-AzureRmNetworkInterface to update and save to the Network Interface. The DnsSettings.DnsServers.Clear() method is first used to clear out the current configuration no matter what is there. Then the DnsSettings.DnsServers.Add (“dns-server-ip-address”) will be used to input the IPs that should be used by the Network Interface.

```
$nic = Get-AzureRmNetworkInterface `
            -ResourceGroupName "ExamRefRG" `
            -Name "examrefwebvm172"
$nic.DnsSettings.DnsServers.Clear()
$nic.DnsSettings.DnsServers.Add("8.8.8.8")
$nic.DnsSettings.DnsServers.Add("4.2.2.1")
$nic | Set-AzureRmNetworkInterface
```

EXAM TIP

When you change your VNet settings to point to your customer provided DNS servers on its network interfaces, you must restart VMs for the new settings to be assigned to the VM’s operating system. When the VM reboots it re-acquires its IP address and the new DNS settings are in place.

Configure DNS Settings on Network Interfaces using the Azure CLI

To update an existing a Network Interface using the Azure CLI as well as specify custom DNS Servers add the --dns-servers argument when using the az network nic update command. The first command will revert the settings to the default servers. Just as in the PowerShell example, we want to remove the current settings back to default first, so you know exactly what the settings are on the Network Interface. If you only run the second command, your DNS Servers will be added to the bottom of the existing list if there are already custom settings on the Network Interface.

```
az network nic update -g ExamRefRG -n examrefwebvm172 --dns-servers ""
az network nic update -g ExamRefRG -n examrefwebvm172 --dns-servers 8.8.8.8 4.2.2.1
```

EXAM TIP

DNS Servers specified for a Network Interface take precedence over those specified for the VNet. This means if you want specific machines on your VNet, use a different DNS Server, you can assign this at the NIC level, and the VNet DNS Server setting will be ignored by that VM.

Network Security Groups (NSGs)

A network security group (NSG) is a networking filter containing a list of security rules that when applied will allow or deny network traffic to resources connected to Azure VNets. These rules can manage both inbound and outbound Traffic. NSGs can be associated to subnets, individual Network Interfaces attached to ARM VMs, and Classic VMs.

In this section, you will learn how to configure NSGs on Network Interfaces and are associated with VMs in Azure Resource Manager. The process of filtering network traffic via a NSG on a NIC is multi-step. First, the network security group must be created. Next, the desired rules would need to be created in that NSG. Once these two steps are completed it will then be associated with the NIC.

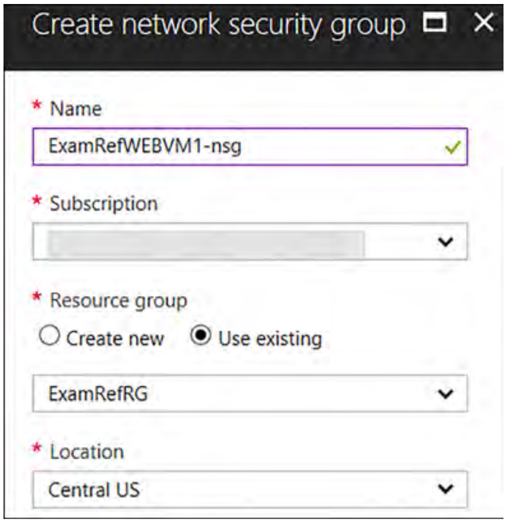
EXAM TIP

NSGs that are associated to network interfaces are said to be filtering “East/West” traffic.

Creating an NSG and Associating with a NIC using the Azure Portal

To create a NSG to be associated with a NIC in the portal click New, followed by Networking and then, select network security group. Once the Create network security group blade loads, provide a Name, the Subscription where your resources are located, the resource group for the NSG and the Location (this must be the same as the resources you wish to apply to the NSG).

In Figure 4-54, the NSG will be created to allow HTTP traffic on Port 80 into a NIC of a VM named ExamRefWEBVM1. The name of the NSG is ExamRefWEBVM1-nsg located in the same resource group ExamRefRG.


Create network security group

* Name

ExamRefWEBVM1-nsg

* Subscription

[redacted]

* Resource group

Create new [✓ Use existing]

ExamRefRG

* Location

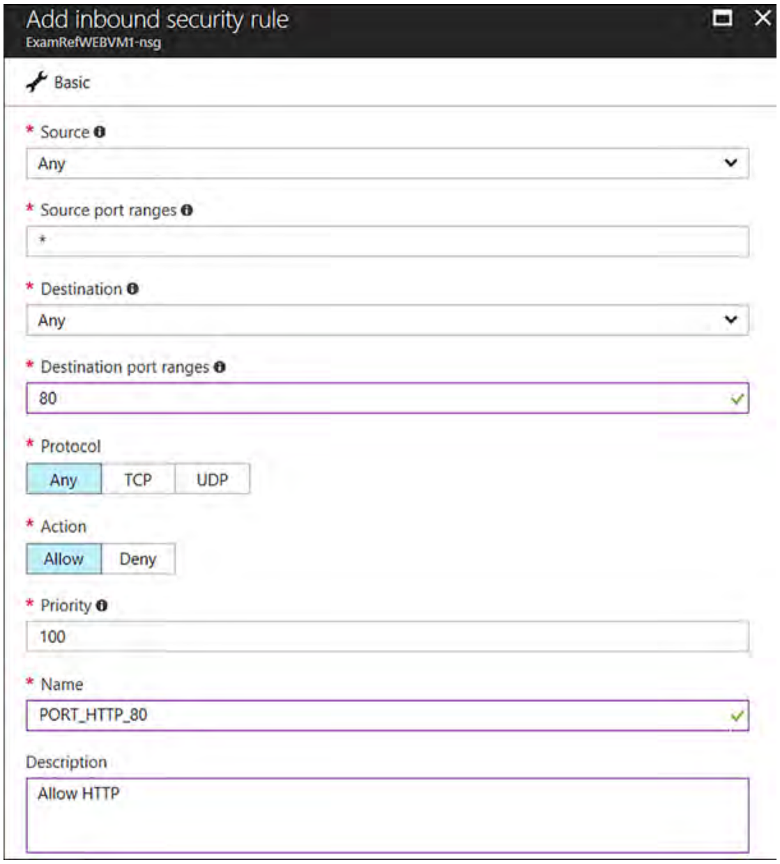
Central US

FIGURE 4-54 Creating an NSG that is associated with a VM NIC

Once the ExamRefWEBVM1-nsg has been created, the portal will open the Overview blade. Here, you will see the NSG has been created, but there are no inbound or outbound security rules beyond the default rules.

To create the inbound rule to allow port 80 select Inbound Security Rules followed by +Add. For the Add inbound security rule, update using the following details, as seen in Figure 4-55:

- **Source** Any
- **Source port ranges** *
- **Destination** Any
- **Destinations port ranges** 80
- **Protocol** Any
- **Action** Allow
- **Priority** 100
- **Name** PORT_HTTP_80
- **Description** All HTTP


Add inbound security rule
ExamRefWEBVM1-nsg

Basic

* Source: Any

* Source port ranges: *

* Destination: Any

* Destination port ranges: 80

* Protocol: [✓ Any] TCP UDP

* Action: [✓ Allow] Deny

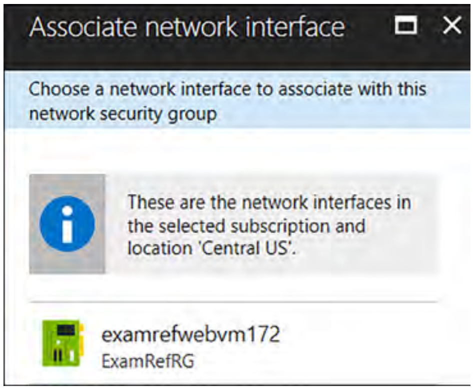
* Priority: 100

* Name: PORT_HTTP_80

Description: Allow HTTP

FIGURE 4-55 An Inbound Rule to Allow traffic on Port 80 is created

Once the portal has configured the inbound rule, it will appear in the portal. Review your rule to ensure it has been created correctly. Now, this NSG with its default rules and newly created inbound rule named Port_80_HTTP is currently not filtering any traffic since it has yet to be associated with NIC. In Figure 4-56, you see the NIC of ExamRefWEBVM1 being selected after selecting Network Interfaces under Settings, followed by selecting +Associate. The portal will ask you to select the NIC name associated with ExamRefWEBVM1. The NSG will immediately start filtering traffic.


Associate network interface

Choose a network interface to associate with this network security group

These are the network interfaces in the selected subscription and location 'Central US'.

examrefwebvm172
ExamRefRG

FIGURE 4-56 Associating the NSG with the NIC of the VM

Upon association of the NSG with the NIC, TCP traffic on Port 80 will be allowed to this VM. Of course, you would need to have a webserver VM configured and listening on Port 80 to respond, but with this NSG, the ability is now opened for that traffic to flow to the VMs.

Creating an NSG and Associating with a Subnet using PowerShell

To create a NSG and configure the rules using PowerShell, use the New-AzureRmNetworkSecurityRuleConfig and New-AzureRmNetworkSecurityGroup PowerShell cmdlets together. In this example, it's assumed you have run the Login-AzureRmAccount command as well as created a resource group and the VNet from the earlier example using PowerShell. The NSG will be created to allow HTTP traffic on Port 80 into a NIC of a VM named ExamRefWEBVM1. The name of the NSG is ExamRefWEBVM1-nsg located in the same resource group ExamRefRGPS.

```
#Build a new Inbound Rule to Allow TCP Traffic on Port 80 to the Subnet
$rule1 = New-AzureRmNetworkSecurityRuleConfig -Name PORT_HTTP_80 `
                                              -Description "Allow HTTP" `
                                              -Access Allow `
                                              -Protocol Tcp `
                                              -Direction Inbound `
                                              -Priority 100 `
                                              -SourceAddressPrefix * `
                                              -SourcePortRange * `
                                              -DestinationAddressPrefix 10.0.0.0/24 `
                                              -DestinationPortRange 80

#Create a new Network Security Group and add the HTTP Rule
$nsg = New-AzureRmNetworkSecurityGroup -ResourceGroupName ExamRefRGPS `
                                       -Location centralus `
                                       -Name "ExamRefWEBVM1-nsg" `
                                       -SecurityRules $rule1
```

After the NSG is created, along with the inbound rule, next you need to associate this with the NIC to control the flow of network traffic by using this filter. To achieve this goal, use Get-AzureRmNetworkInterface. After the configuration on the NIC has been set, use Set-AzureRm-NetworkInterface to save the configuration to the NIC.

```
#Associate the Rule with the NIC from ExamRefWEBVM1
$nic = Get-AzureRmNetworkInterface -ResourceGroupName ExamRefRGPS -Name examrefwebvm1892
$nic.NetworkSecurityGroup = $nsg
Set-AzureRmNetworkInterface -NetworkInterface $nic
```

Creating an NSG and Associating with a NIC using the Azure CLI

Creating an NSG by using the CLI is a multi-step process, just as it is with the portal and PowerShell. The `az network nsg create` command is first used to create the NSG. After the NSG is created, next create the rule to allow Port 80 to the subnet. This is created by using the `az network nsg rule create` command. After the rule has been created, this is associated with the NIC of a VM named ExamRefWEBVM1 by using the `az network nic update` command. The name of the NSG is ExamRefWEBVM1-nsg, located in the same resource group ExamRefRGCLI.

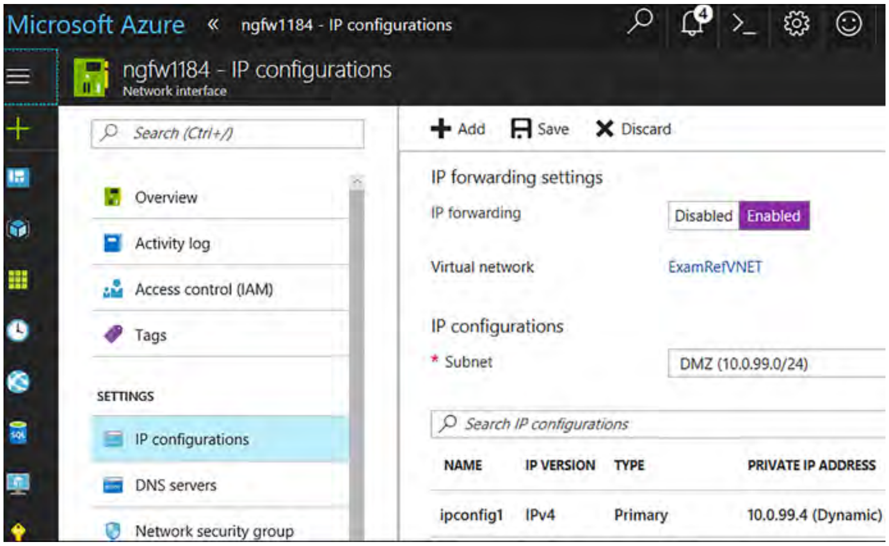
```
# Create the NSG
az network nsg create --resource-group ExamRefRGCLI --name ExamRefWEBVM1-nsg

# Create the NSG Inbound Rule allowing TCP traffic on Port 80
az network nsg rule create --resource-group ExamRefRGCLI --name PORT_HTTP_80
--nsg-name ExamRefWEBVM1-nsg --direction Inbound --priority 100 --access Allow
--source-address-prefix "*" --source-port-range "*" --destination-address-prefix
"*" --destination-port-range "80" --description "Allow HTTP" --protocol TCP

# Associate the NSG with the NIC from ExamRefWEBVM1
az network nic update --resource-group ExamRefRGCLI --name examrefwebvm1400
 --network-security-group ExamRefWEBVM1-nsg
```

User Defined Routes (UDR) with IP Forwarding

User Defined Routes (UDR) allow for changing the default system routes that Azure creates for you in an Azure VNet. The UDRs forward traffic to a virtual appliance such as a firewall. For that traffic to be allowed to pass to that virtual appliance, you must enable IP forwarding on the network interface of the VM.

A virtual appliance is nothing more than a VM that runs an application used to handle network traffic in some way. By default, VMs in Azure do not have the ability to forward packets that are intended for a different destination, so this configuration allows those packets to flow through the device. Of course, the firewall device would have to be configured as well by using its internal tools to pass this traffic. This configuration doesn't typically involve any changes to the Azure UDR or VNet.

IP forwarding can be enabled on a network interface by using the Azure portal, PowerShell, or the Azure CLI. In Figure 4-57, you see that the network interface of the NGFW1 VM has the IP forwarding set as Enabled. This VM is now able to accept and send packets that were not originally intended for this VM.



IP forwarding settings

IP forwarding: Disabled [✓ Enabled]

Virtual network: ExamRefVNET

IP configurations

* Subnet: DMZ (10.0.99.0/24)

Search IP configurations

NAME	IP VERSION	TYPE	PRIVATE IP ADDRESS
ipconfig1	IPv4	Primary	10.0.99.4 (Dynamic)

FIGURE 4-57 IP Forwarding Enabled on a Virtual Appliance

External and Internal load balancing with HTTP and TCP health probes

The Azure load balancer provides the means for you to deliver highly available and high performing web based applications using your VMs running in a VNet. This is a Layer 4 (TCP, UDP) load balancer that distributes inbound traffic among healthy instances of services defined in a load-balanced set.

The load balancer can run in three different configurations:

- Load balance incoming internet traffic to VMs. This configuration is known as internet-facing load balancing.
- Traffic between virtual machines in a VNet is also supported. These can be between virtual machines in cloud services, or between on-premises computers and virtual machines in a cross-premises Virtual Network. This is known as internal load balancing.
- Forward external traffic to a specific virtual machine, which is a means of using Network Address Translation (NAT).

The load balancer uses a hash-based distribution algorithm. Like most load balancers, it uses a 5-tuple hash composed of source IP, source port, destination IP, destination port, and protocol type to map traffic to available servers, by default. This provides stickiness within a transport session, meaning that packets in the same TCP or UDP session are directed to the same instance behind the load-balanced endpoint. If, or when, the client closes and reopens the connection or starts a new session from the same source IP, the source port changes.

You have control over how inbound communication comes into your endpoints, known as the input endpoint. An input endpoint listens on a public port and forwards traffic to an internal port. You can map the same ports for an internal or external endpoint or use a different port for them.

For example, you can have a web server configured to listen to port 81 while the public endpoint mapping is port 80. The creation of a public endpoint triggers the creation of a load balancer instance.

Health Probes

At its core, the purpose of a load balancer is twofold: to spread traffic across a farm of VMs that are providing a service so you don't overload them and to ensure that the VMs are healthy and ready to accept traffic.

The Azure load balancer can probe the health of your VMs deployed into a VNet. When a VM probe experiences a failure, this means that the VM is no longer able to provide the service, therefore the load balancer marks it as an unhealthy instance and stops sending new connections to the VM. Existing connections are not impacted by being removed from the pool of healthy instances, but users could experience failures if they have current open connections to that VM.

The Azure load balancer supports two types of probes for virtual machines:

- **TCP Probe** This probe relies on a successful TCP session establishment to a defined probe port. If the VM responds to the request to establish a simple TCP connection on the port defined when creating the probe, the VM is marked as healthy. For example, a TCP probe could be created connecting to portal 80. If the machine is active and allows connections on port 80, the load balancer would be able to connect and the machine would pass the probe. If for some reason the machine was stopped or the load balancer could no longer connect to port 80, it would be marked as unhealthy.
- **HTTP Probe** This probe is used to determine if a VM is serving webpages without issues by using the HTTP protocol. When a webpage loads successfully, there is an HTTP error code that is given: 200. This error code means that the page loaded successfully. One of the configurations on the HTTP probe is the path to the file used for the probe which by default, is marked a "/". This tells the Azure load balancer to load the default webpage from the VM. Often this would be default.aspx or index.html, but you can configure this if you want to create your own page to check the health of a site. Just returning the default page with 200 doesn't provide deeper insight as to the true functionally of your site, so using some custom logic to determine the health of the site could make sense. A developer would create a custom page and then you would configure the load balancer to load that page. If it loads correctly, the 200 is provided and the VM is put into the pool of available VMs to service client requests.

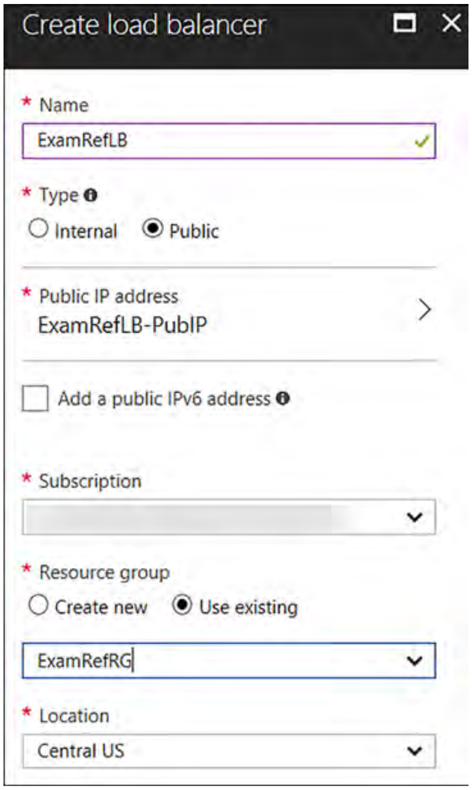
EXAM TIP

For both TCP and HTTP probes you can configure the interval and the unhealthy threshold. The interval is the amount of time between the probe attempts, or in other words how often Azure uses this probe. Unhealthy threshold is the number of consecutive failures that must occur before the VM is considered unhealthy.

Creating the Azure load balancer using the Azure portal

To use the Azure load balancer, the administrator must first provision the resource including the Frontend IP configuration. After this step has been completed then you will need to create the backend pool, heath probes, and load balancing rules.

To create the load balancer in the portal, select New, followed by Networking and locate the load balancer. As seen in Figure 4-58, supply a name for the load balancer, assign a type of Public or Internal, select a public or private IP Address, along with the subscription, resource group, and location. In the case of the example, an internet facing load balancer will be created with a public IP Address, and it will point to two Web Servers named ExamRefWEBVM1 and ExamRefWEBVM2 that are part of an Availability Set called ExamRefWebAVSet. Both VMs have one NIC that is connected to the Apps subnet of the ExamRefVNET VNet.


Create load balancer

* Name: ExamRefLB

* Type: ( ) Internal [✓ Public]

* Public IP address: ExamRefLB-PubIP

☐ Add a public IPv6 address

* Subscription

* Resource group: ( ) Create new [✓ Use existing]

ExamRefRG

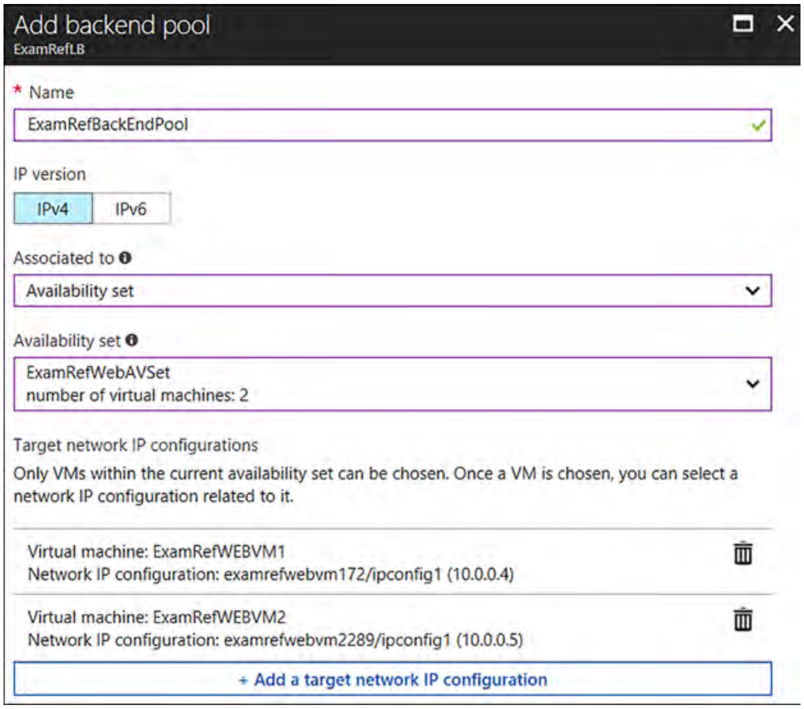
* Location: Central US

FIGURE 4-58 Creating a Load Balancer with the Azure portal

After the load balancer has been created, execute the next steps to integrate your VMs with the load balancer:

- Backend Pool
- Health Probe HTTP
- Load Balancing Rule

Figure 4-59 shows the portal to add a backend pool. To create the backend pool, open the load balancer in the portal and then in the Settings section, click Backend pools. Next click +Add Provide a Name for the Backend Pool, leave the IPv4 section for the IP Version and move to the Associated to drop-down. Select Availability Set and another drop-down appears. Select ExamRefWebAVSet from the drop-down list. Next click the +Add a target network IP configuration. Next a Target virtual machine drop-down appears and you can select ExamRefWEBVM1 along with its network IP configuration. Click +Add a target network IP configuration again and follow this same procedure to select ExamRefWEBVM2. After you complete adding both VMs, click OK.


Add backend pool
ExamRefLB

* Name
ExamRefBackEndPool

IP version
IPv4 | IPv6

Associated to
Availability set

Availability set
ExamRefWebAVSet
number of virtual machines: 2

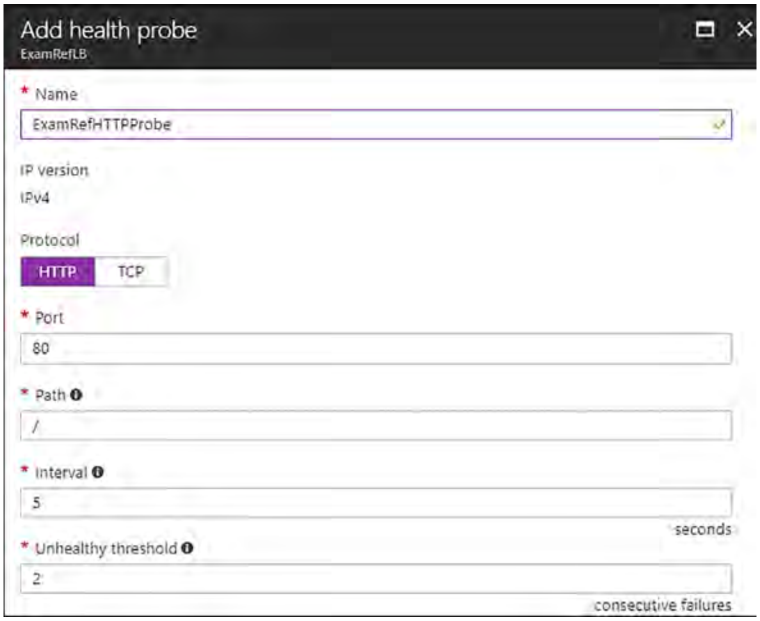
Target network IP configurations
Only VMs within the current availability set can be chosen. Once a VM is chosen, you can select a network IP configuration related to it.

Virtual machine: ExamRefWEBVM1
Network IP configuration: examrefwebvm172/ipconfig1 (10.0.0.4)

Virtual machine: ExamRefWEBVM2
Network IP configuration: examrefwebvm2289/ipconfig1 (10.0.0.5)

FIGURE 4-59 Creating the Backend pool which exposes the VMs

To ensure your Web Servers are ready, you will need to add the HTTP Probe. To begin configuring the HTTP probe, select the Health probe link in Settings and then +Add. As seen in Figure 4-60, provide a Name, select the HTTP protocol, and accept the defaults of Port 80, Interval of 5 and Unhealth threshold of 2. Then click OK. Notice that there is an additional item named path which is the location of a file or folder on the web server for the load balancer to connect.


Add health probe
ExamRefLB

* Name
ExamRefHTTPProbe

IP version
IPv4

Protocol

* Port
80

* Path
/

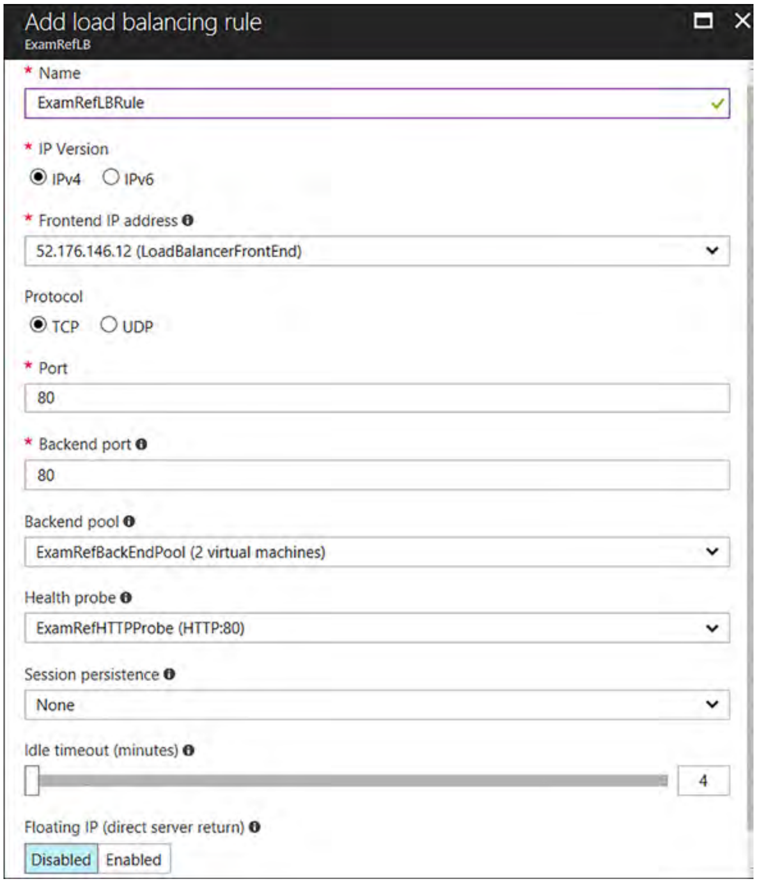
* Interval
5 seconds

* Unhealthy threshold
2 consecutive failures

FIGURE 4-60 Creating a HTTP Health Probe

Now that you have created the backend pool telling the load balancer which machines are to be used, and you have configured the probes to help determine which ones are healthy, you will now put it all together with the load balancing rules. These rules help to bring these configurations together connecting the Frontend to the Backend. To create the rule, click the load balancing rules link under settings, and then select +Add. Complete the following configurations, as seen in Figure 4-61:

- **Name** ExamRefLBRule
- **IP Version** IPv4
- **Frontend IP Address** Select the Public IP Address
- **Protocol** TCP
- **Port** 80
- **Backend port** 80
- **Backend pool** Select the Pool you created
- **Heath Probe** Select the TCP Rule
- **Session** Persistence None
- **Idle Timeout** 4 minutes
- **Floating IP** Disabled


Add load balancing rule
ExamRefLB

* Name
ExamRefLBRule

* IP Version
(● IPv4) ○ IPv6

* Frontend IP address
52.176.146.12 (LoadBalancerFrontEnd)

Protocol
(● TCP) ○ UDP

* Port
80

* Backend port
80

Backend pool
ExamRefBackEndPool (2 virtual machines)

Health probe
ExamRefHTTPProbe (HTTP:80)

Session persistence
None

Idle timeout (minutes)
4

Floating IP (direct server return)
[✓ Disabled] Enabled

FIGURE 4-61 Creating the Load Balancing Rule using the Backend Pool and Health Probe

After this is put in the place, if the VMs added to the backend pool are configured with a web server and there are no network security groups or other firewalls blocking port 80, you should be able to connect to the Public IP address of the load balancer and see the webpage.

Creating the Azure Load Balancer using PowerShell

When creating a new load balancer by using PowerShell, there are quite a few steps involved. The script must first create a Public IP address, the front IP configuration, backend pool, and the HTTP probe. Then, the load balancer rule is followed by the load balancer resource where these other configurations will be put in place. After you create the load balancer, update the backend pool referencing the NICs of the VMs that are serving the website. The PowerShell cmdlets used for this script include:

```
New-AzureRmPublicIpAddress
New-AzureRmLoadBalancerFrontendIpConfig
New-AzureRmLoadBalancerBackendAddressPoolConfig
New-AzureRmLoadBalancerProbeConfig
New-AzureRmLoadBalancer
Get-AzureRmVirtualNetwork
Get-AzureRmVirtualNetworkSubnetConfig
Get-AzureRmNetworkInterface
Set-AzureRmNetworkInterfaceIpConfig
Set-AzureRmNetworkInterface
```

In this example, an internet facing load balancer will be created with a public IP Address, and it will point to two Web Servers named ExamRefWEBVM1 and ExamRefWEBVM2, and they are part of an Availability Set called ExamRefWebAVSet. Both VMs have one NIC connected to the Apps subnet of the ExamRefVNET-PS VNet created in earlier steps.

```
# Set Variables
$publicIpName = "ExamRefLB-PublicIP-PS"
$rgName = "ExamRefRGPS"
$dnsPrefix = "examreflbps"
$location = "centralus"
$lbname = "ExamRefLBPS"
$vnetName = "ExamRefVNET-PS"

# Create the Public IP
$publicIP = New-AzureRmPublicIpAddress -Name $publicIpName `
                           -ResourceGroupName $rgName `
                           -AllocationMethod Static `
                           -DomainNameLabel $dnsPrefix `
                           -Location $location

#Create Frontend IP Configuration
$frontendIP = New-AzureRmLoadBalancerFrontendIpConfig -Name ExamRefFrontEndPS `
                                                      -PublicIpAddress $publicIP

# Create Backend Pool
$beAddressPool = New-AzureRmLoadBalancerBackendAddressPoolConfig -Name
ExamRefBackEndPoolPS

#Create HTTP Probe
$healthProbe = New-AzureRmLoadBalancerProbeConfig -Name HealthProbe `
                                                  -RequestPath '/' `
                                                  -Protocol http `
                                                  -Port 80 `
                                                  -IntervalInSeconds 5 `
                                                  -ProbeCount 2

#Create Load Balancer Rule
$lbrule = New-AzureRmLoadBalancerRuleConfig -Name ExamRefRuleHTTPPS `
                                            -FrontendIpConfiguration $frontendIP `
                                            -BackendAddressPool  $beAddressPool `
                                            -Probe $healthProbe `
                                            -Protocol Tcp `
                                            -FrontendPort 80 `
                                            -BackendPort 80

#Create Load Balancer
New-AzureRmLoadBalancer -ResourceGroupName $rgName `
                        -Name $lbName `
                        -Location $location `
                        -FrontendIpConfiguration $frontendIP `
                        -LoadBalancingRule $lbrule `
                        -BackendAddressPool $beAddressPool `
                        -Probe $healthProbe

# Add the Web Servers to the Backend Pool
$vnet = Get-AzureRmVirtualNetwork -Name $vnetName `
                                  -ResourceGroupName $rgName
$subnet = Get-AzureRmVirtualNetworkSubnetConfig -Name Apps `
                                                -VirtualNetwork $vnet
$nic1 = Get-AzureRmNetworkInterface -Name examrefwebvm1480 `
                                    -ResourceGroupName $rgName
$nic1 | Set-AzureRmNetworkInterfaceIpConfig -Name ipconfig1 `
                                            -LoadBalancerBackendAddressPool
                                            $beAddressPool `
                                            -Subnet $subnet
$nic1 | Set-AzureRmNetworkInterface

$nic2 = Get-AzureRmNetworkInterface -Name examrefwebvm2217 `
                                    -ResourceGroupName $rgName
$nic2 | Set-AzureRmNetworkInterfaceIpConfig -Name ipconfig1 `
                                            -LoadBalancerBackendAddressPool
                                            $beAddressPool `
                                            -Subnet $subnet
$nic2 | Set-AzureRmNetworkInterface
```

Creating the Azure load balancer using the Azure CLI

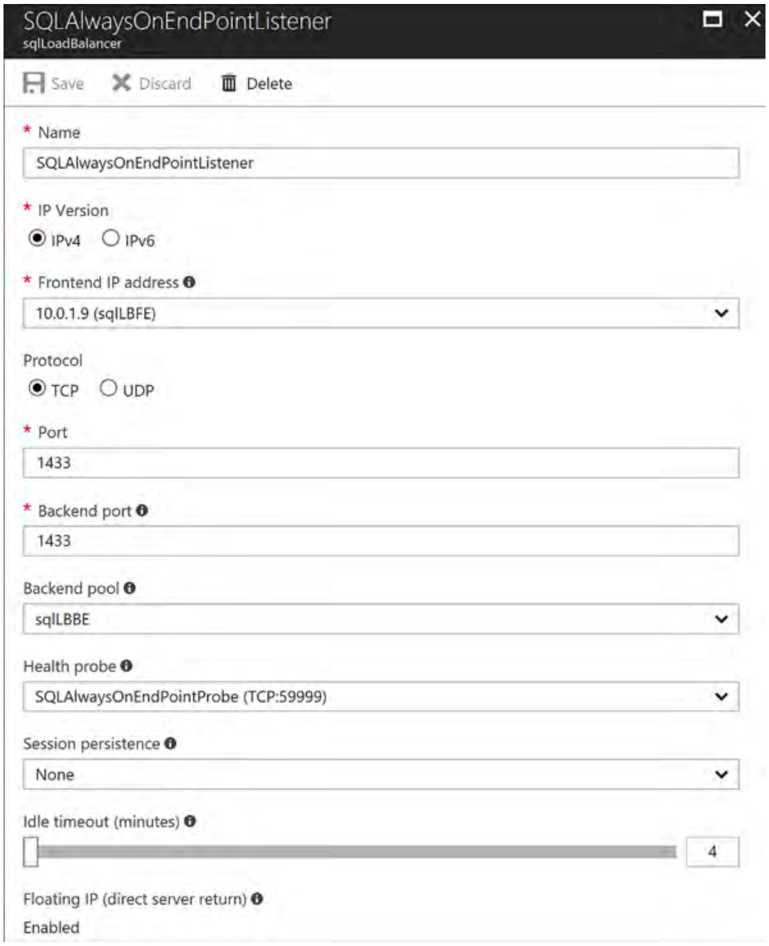
The same configurations are required when creating a load balancer by using the Azure CLI as they are when creating load balancers in the portal or PowerShell. The process using CLI is not quite as intricate as the process using PowerShell though. You can leverage the az network lb command with a few different arguments along with the az network public-ip to create the Public IP address.

```
# Creating a Public IP Address
az network public-ip create -g ExamRefRGCLI -n ExamRefLB-PublicIP-CLI --dns-name
 examreflbcli --allocation-method Static

#Create Load Balancer
az network lb create -n ExamRefLBCLI -g ExamRefRGCLI -l centralus --backend-pool-name
 ExamRefBackEndPoolCLI --frontend-ip-name ExamRefFrontEndCLI --public-ip-address
 ExamRefLB-PublicIP-CLI

#Create HTTP Probe
az network lb probe create -n HealthProbe -g ExamRefRGCLI --lb-name ExamRefLBCLI
--protocol http --port 80 --path / --interval 5 --threshold 2

#Create Load Balancer Rule
az network lb rule create -n ExamRefRuleHTTPCLI -g ExamRefRGCLI --lb-name
ExamRefLBCLI --protocol Tcp --frontend-port 80 --backend-port 80
--frontend-ip-name ExamRefFrontEndCLI --backend-pool-name ExamRefBackEndPoolCLI
 --probe-name HealthProbe

# Add the Web Servers to the Backend Pool
az network nic ip-config address-pool add --address-pool ExamRefBackEndPoolCLI
 --lb-name ExamRefLBCLI -g ExamRefRGCLI --nic-name examrefwebvm160 --ip-config-name
 ipconfig1
az network nic ip-config address-pool add --address-pool ExamRefBackEndPoolCLI
--lb-name ExamRefLBCLI -g ExamRefRGCLI --nic-name examrefwebvm2139 --ip-config-name
 ipconfig1
```

Direct Server Return

Some application scenarios require the same port to be used on the frontend configuration and the backend on the VMs. Common examples of port reuse include clustering for high availability, network virtual appliances, and exposing multiple TLS endpoints without re-encryption.

If you want to reuse the backend port across multiple rules, you must enable Floating IP in the load balancer rule definition. Floating IP is a portion of what is known as Direct Server Return (DSR). In Azure, this ability is typically only used for deploying SQL Server Always On Availability Groups. This is a clustering technology for providing highly available databases using Azure VMs. Microsoft recommends only using this feature when configuring SQL Always On Availability Groups Listener. This can only be configured with creating a load balancing rule and the frontend port must match the backend port. Figure 4-62 shows a SQL Always on Listener with the direct server return set to Enabled.



SQLAlwaysOnEndPointListener
sqlLoadBalancer

* Name
SQLAlwaysOnEndPointListener

* IP Version
[✓ IPv4] IPv6

* Frontend IP address
10.0.1.9 (sqlLBFE)

Protocol
[✓ TCP] UDP

* Port
1433

* Backend port
1433

Backend pool
sqlLBBE

Health probe
SQLAlwaysOnEndPointProbe (TCP:59999)

Session persistence
None

Idle timeout (minutes)
4

Floating IP (direct server return)
Enabled

FIGURE 4-62 SQL Always on Listener with the Direct Server Return Enabled

This functionality should only be used if the VM that is responding as a part of the backend pool is connecting directly to a VM within the VNet or networks that are connected to the VNet. It is not supported to use direct server return with clients that are on the internet. This is due to the server talking back directly to the client rather than its traffic going through the load balancer back to the client.

MORE INFORMATION **SQL SERVER ALWAYS ON AVAILABILIT GROUPS IN AZURE**

To learn more about running SQL Server Always On Availability Groups in Azure review the following article: *https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sql/virtual-machines-windows-portal-sql-alwayson-int-listener.*

Design and Implement Application Gateway (App Gateway)

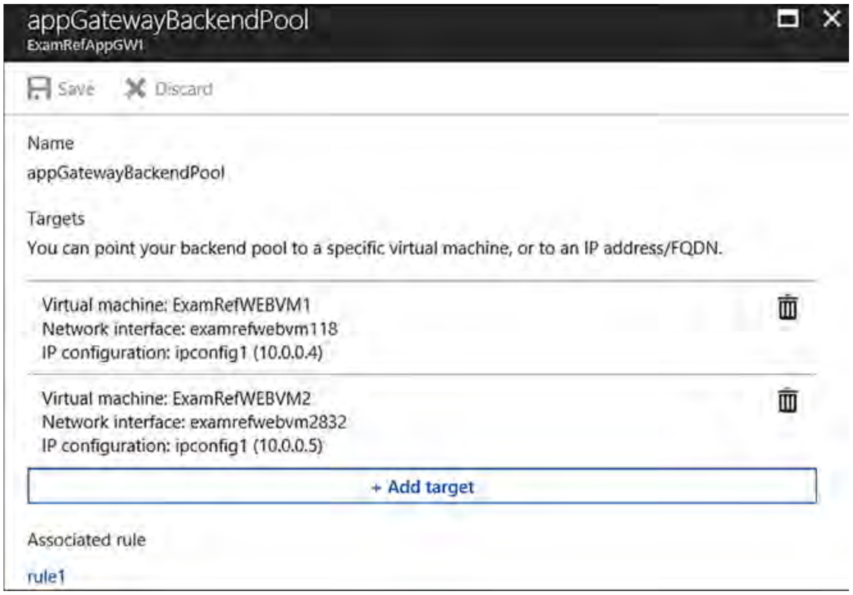
The Application Gateway can be deployed as a Layer 7 load balancer into a VNet. VMs that are a part of the VNet can then be added as the backend pool of an App Gateway. These VMs host the application, much like the Azure load balancer being added to the backend pool is a part of the NIC IP configuration. This can be accomplished by using the Azure portal, PowerShell, and the Azure CLI.

MORE INFORMATION APPLICATION GATEWAY

To learn about how to Implement the Application Gateway refer to Skill 4.1 Configure Virtual Networks.

Configure VMs as Backend Pool for App Gateway using the Azure Portal

To configure the VMs as part of the Backend pool using the Azure portal open the App Gateway and then select backend pools under the Settings section. Next, click on the name of the Backend pool created when the App Gateway was provisioned. Then, select +Add target followed by VM. From there, you can select the first VM and its IP Configuration. In the case of this example, the ExamRefWEBVM1 and ExamRefWEBVM2 have been selected with ipconfig1 from each VM. Notice their private IP address is listed, as this is the IP Address the App Gateway will use when directing traffic to the VMs. Once they have been added click Save and the App Gateway will update. Figure 4-63 shows the appGatewayBackendPool with the VMs configurations added.


appGatewayBackendPool
ExamRefAppGW1

Name
appGatewayBackendPool

Targets
You can point your backend pool to a specific virtual machine, or to an IP address/FQDN.

Virtual machine: ExamRefWEBVM1
Network interface: examrefwebvm118
IP configuration: ipconfig1 (10.0.0.4)

Virtual machine: ExamRefWEBVM2
Network interface: examrefwebvm2832
IP configuration: ipconfig1 (10.0.0.5)

Associated rule
rule1

FIGURE 4-63 App Gateway Backend Pool with two VMs added

After the portal reports that the App Gateway update is complete, you can connect to the Public IP address of the App Gateway by using your web browser. It does take a few minutes for the sites to come online, so be patient.

Configure VMs as Backend Pool for App Gateway using the PowerShell

To update the backend pool of the App Gateway with the VMs you will use a combination of the Get-AzureRmApplicationGateway cmdlet along with the with the Get-AzureRmApplicationGatewayBackendAddressPool and Get-AzureRmNetworkInterface. These cmdlets will be used to load information about the various resources into variables that can then be used with the Set-AzureRmApplicationGatewayBackendAddressPool cmdlet to configure the backend pool with the addresses of your web servers. These backend pool members are all validated to be healthy by probes, whether they are basic probes or custom probes. Traffic is then routed to them when requests come into the application gateway. Backend pools can be used by multiple rules within the application gateway. This means one backend pool could be used for multiple web applications that reside on the same host.

```
# Add VM IP Addresses to the Backend Pool of App Gateway
$appGw = Get-AzureRmApplicationGateway -Name "ExamRefAppGWPS" -ResourceGroupName
 "ExamRefRGPS"
$backendPool = Get-AzureRmApplicationGatewayBackendAddressPool -Name
 "appGatewayBackendPool" -ApplicationGateway $AppGw
$nic01 = Get-AzureRmNetworkInterface -Name "examrefwebvm1480" -ResourceGroupName
"ExamRefRGPS"
$nic02 = Get-AzureRmNetworkInterface -Name "examrefwebvm2217" -ResourceGroupName
"ExamRefRGPS"
Set-AzureRmApplicationGatewayBackendAddressPool -ApplicationGateway $appGw `
                          -Name $backendPool `
                          -BackendIPAddresses
$nic01.IpConfigurations[0].PrivateIpAddress,$nic02.IpConfigurations[0].PrivateIpAddress
```

Configure VMs as Backend Pool for App Gateway using the Azure CLI

Only one command is required to update the backend pool of the App Gateway using the Azure CLI. The `az network application-gateway address-pool` is used, but you must know the IP Addresses of the VM's IP Configurations to make this work properly.

```
# Add VM IP Addresses to the Backend Pool of App Gateway
az network application-gateway address-pool update -n appGatewayBackendPool
--gateway-name ExamRefAppGWCLI -g ExamRefRGCLI --servers 10.0.0.6 10.0.0.7
```

Skill 4.4: Design and implement a communication strategy

This section focuses on the various methods and connectivity choices to securely extend your on-premises network into the Microsoft cloud. There are two types of connections for connecting to the MS cloud covered in this section: the S2S VPN and an Azure resource known as Azure App Hybrid Connections for exposing services inside your network without a VPN.

This skill covers how to:

- Leverage Site-to-Site (S2S) VPN to connect to an on-premises infrastructure
- Implement Hybrid Connections to access data sources on-premises

Leverage Site-to-Site (S2S) VPN to connect to an on-premises infrastructure

S2S connections are used for building hybrid configurations. Once established between your VPN on-premises device and an Azure VPN Gateway, this connection type allows your on-premises users and VMs to VMs and services that are running in an Azure VNet.

Running workloads in the cloud and on-premises is very common and as such, these connections make this possible. In the case of adding a VPN connection to Azure you are essentially just adding another datacenter to your network; it just happens to be the public Azure cloud.

Figure 4-64 shows a VNET that has been connected to an on-premises datacenter by using ExpressRoute. There is an intranet SharePoint farm deployed along with its data tier and domain controllers. Notice the use of the multiple load balancers to facilitate the deployment requirement of the application. This entire Virtual Network, application, data, and identity is available and functional for the clients that are on the other side of the ExpressRoute circuit.

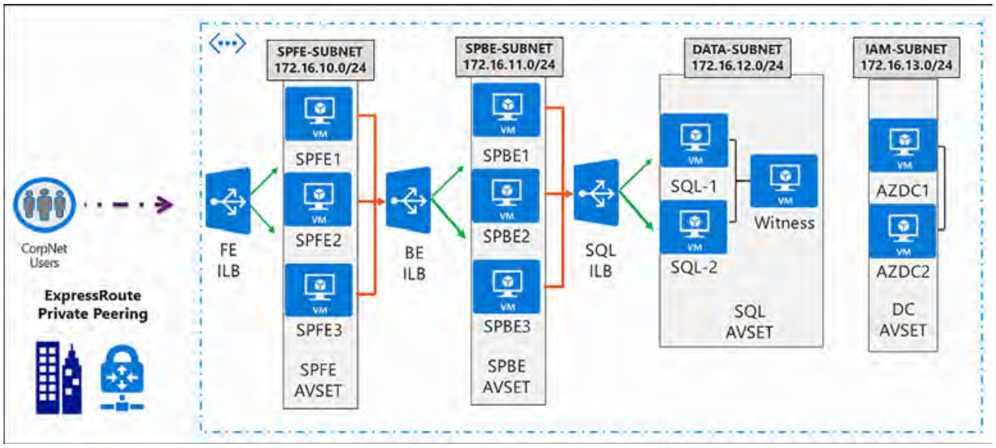

SPFE-SUBNET 172.16.10.0/24
SPBE-SUBNET 172.16.11.0/24
DATA-SUBNET 172.16.12.0/24
IAM-SUBNET 172.16.13.0/24
CorpNet Users
ExpressRoute Private Peering
FE ILB
SPFE1
SPFE2
SPFE3
SPFE AVSET
BE ILB
SPBE1
SPBE2
SPBE3
SPBE AVSET
SQL ILB
SQL-1
SQL-2
Witness
SQL AVSET
AZDC1
AZDC2
DC AVSET
VM

FIGURE 4-64 Virtual Network and Services connected to On-premises using ExpressRoute

Implement Hybrid Connections to access data sources on-premises

Within the Azure App Service, hybrid connections can be used to access application resources in other networks. This connection provides access FROM your application running in Azure TO an application endpoint hosted in your datacenter. It does this is such a way that there is not a requirement to build out a complex full S2S or hybrid cloud infrastructure.

In many scenarios, this type of connection solves the problem customers have if that want to build an application in the cloud, but the data is "locked," in their datacenter. By using the hybrid connection in Azure, the App Service can connect to that data. Figure 4-65 shows a functional diagram of this type of hybrid connection.

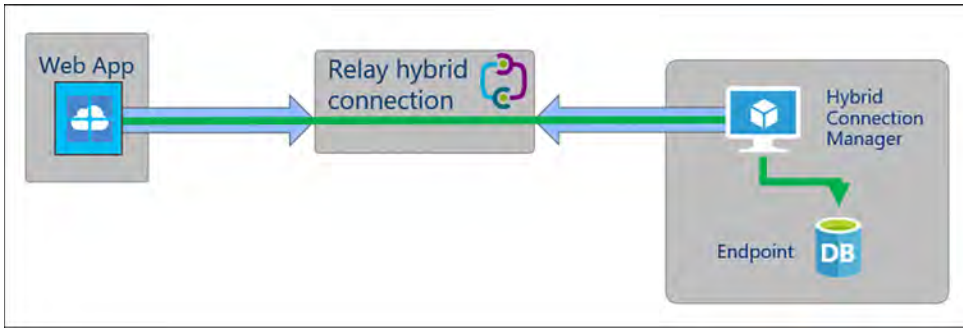


FIGURE 4-65 Hybrid Connection from App Services to an On-premises Database

When these connections are created it only allows the Web App to talk to the datacenter. It does not enable full communication between the local server and the Web App running in Azure. Each hybrid connection correlates to a single TCP host and port combination. This means that the hybrid connection endpoint can be on any operating system and any application, provided you are hitting a TCP listening port. Hybrid connections do not know or care what the application protocol is or what you are accessing. It is simply providing network access.

The hybrid connections feature consists of two outbound calls to Service Bus Relay. There is a connection from a library on the host where your app is running in the app service and then there is a connection from the Hybrid Connection Manager (HCM) to Service Bus Relay. The HCM is a relay service that you deploy within the network hosting.

Through the two joined connections your app has a TCP tunnel to a fixed host:port combination on the other side of the HCM. The connection uses TLS 1.2 for security and SAS keys for authentication/authorization.

There are many benefits to the hybrid connections capability including:

- Apps can securely access on premises systems and services securely
- The feature does not require an internet accessible endpoint

- Each hybrid connection matches to a single host:port combination, which is an excellent security aspect
- It normally does not require firewall holes as the connections are all outbound over standard web ports
- The feature is running at the network level, so it is agnostic to the language used by your app and the technology used by the endpoint

Thought experiment

In this thought experiment, apply what you have learned about virtual networks in this chapter. You can find answers to these questions in the Answers section at the end of this chapter.

Your management team has named you as the lead architect to implement the first cloud deployment in Contoso's history. There is a new web based application that runs on IIS Server using a SQL database that they want implemented in Azure.

During a meeting with the application vendor and your manager, you have gained a better understanding of the implementation needs and Contoso's requirements. The application must run on Azure VMs and the SQL server needs to be implemented as an Always on Availability Group cluster. The vendor has told you that the application supports multiple web front ends for high-availability. Your manager has mentioned multiple times how important security is given this is the first cloud installation. During the conversation, she made it clear that the Azure implementation should be secured using a multi-layered approach using firewall rules, and that it must be deployed using a web application firewall (WAF).

At the end of the meeting your manager also mentioned that as a part of this project you should implement a permanent low latency connection between your primary datacenter and Azure, as there are many follow-on projects after this one. It is also important to have all servers be able to communicate using their host names and not IP addresses as well because it must support authentications to your Active Directory domain controllers. The onsite network is a class A 10.0.0.0/16 Network, but you do have access to 8 class C public addresses provided by your network service provider and registered in your company's name with the ARIN.

1. Given that the solution required VMs, the configuration will require a VNet. What should you consider with respect to the address space of the VNet? What address space will you use? Also, what subnets should you create to support the requirements? What are the CIDR ranges for these subnets?
2. Where would each tier of the application be deployed using the subnets that you have defined? How will you secure these subnets and VMs?
3. What type of connection will be created between your on-premises datacenter and Azure? How will DNS Services be implemented?
4. What is the basic architecture for the application?

Thought experiment answers

This section contains the solution to the thought experiment for the chapter.

1. The VNet should have an address space that has an ample number of IP addresses (for future growth), but it cannot overlap the current network space, so a class B private network at 172.16.0.0/20 will be implemented. The following subnets will be created:
   - **Apps** 172.16.0.0/24
   - **Data** 172.16.1.0/24
   - **Identity** 172.16.2.0/24
   - **AppGateway** 172.16.99.0/24
   - **GatewaySubnet** 172.16.100.0/28
2. The different subnets will contain various VMs to meet the requirements. There will be network security groups on each subnet and each VMs NIC. Inbound rules will be created to allow the least privileged access for traffic into a subnet and then into the NIC.
   - **Apps** IIS VMs hosting the application
   - **Data** SQL Server VMs providing the databases
   - **Identity** AD Domain Controllers/DNS
   - **AppGateway** App Gateway VMs
   - **GatewaySubnet** ExpressRoute VPN Gateways
3. An ExpressRoute connection between Azure and the on-premises datacenter will be created. The Public IP addresses will be configured as the endpoint for the ExpressRoute circuit. The AD domain controllers will be setup as DNS servers by configuring the VNet to point to those servers.
4. The Azure Application Gateway will be used to publish the IIS VMs as a Backend Pool. These VMs which will be installed into the Apps subnet. The App Gateway will also be configured as a web application firewall (WAF). The SQL Server Always On cluster will be installed into the Data subnet behind an internal load balancer that will be configured using Direct Server Return. The domain controllers will be installed into the Identity subnet and configured to provide DNS services, including the configuration on the VNet to point to these servers for DNS.

Chapter summary

This chapter covered the many topics that make up Virtual Networks in Azure. These topics range from designing and implementing Virtual Networks, to connecting Virtual Networks to other Virtual Networks. Configuring Azure VMs for use with Virtual Networks was also covered including how to secure them using network security groups which are essentially firewalls. You also reviewed deploying web applications, both internet and Intranet facing, by using the

Azure load balancer and the Azure Application Gateway. This chapter also discussed the different options for connecting on-premises networks to Azure, including Site-to-Site VPNs and ExpressRoute.

Below are some of the key takeaways from this chapter:

- Azure Virtual Networks are isolated cloud networks using the IP address space and are required for deploying virtual machines in Azure.
- Subnets allow you to isolate workloads and can be used with network security groups to create firewall rules.
- The GatewaySubnet is a special subnet that is only used for VPN Gateways.
- Azure provides DNS services, but a customer can implement their own DNS servers. The DNS servers can be configured either at the VNet or the network interface level.
- The Azure Application Gateway is a Layer 7 load balancer that can offload SSL traffic, provide web application firewall services, and URL based routing.
- Azure VNets can be connected to each other either by using peering or VPN tunnels.
- VNet peering allows VMs to see each other as one network, but their relationships are non-transitive. If VNETA and VNETB are peered and VNETB and VNETC are peered VNETA and VNETC are not peered.
- There are three types of hybrid connections with Azure Point to Site, Site-to-Site and ExpressRoute.
- VPN Gateways make hybrid connections possible and choosing the correct one should be based on the throughput that is required and the type of connection, but most connections are route-based.
- BGP Routing is used for ExpressRoute and Multi-Site VPN connections.
- ExpressRoute is only available in certain cities around the world and has a premium add-on to support large global networks.
- Public and private IP addresses have two allocation methods: dynamic or static.
- Public IPs can be assigned to VMs, VPN Gateways, internet-facing load balancers or Application Gateways.
- User Defined Routes change the default behavior of subnets allowing you to direct the traffic to other locations. Typically, traffic is sent through a virtual appliance such as a firewall. If traffic is sent to a virtual appliance, IP forwarding must be enabled on the NIC of the VM.
- The Azure load balancer can be used for internet or intranet workloads providing web based applications in a highly available configuration. Health probes are used to ensure the VMs are ready to accept traffic.
- Direct Server Return is an Azure load balancer configuration that is used with SQL Server Always On Availability group clusters deployed on VMs in an Azure VNet.

- Hybrid connections in Azure are a specific type of connection that allows for Azure Applications Apps to connect to on-premises resources such as databases without the need for a VPN. These are different than the hybrid cloud connections that are created by using S2S VPNs.

Index

A

access control 178–184, 310–321. *See also* security
  access policies 338
  ARM authentication 311–315
  lock resources 319–321
  management policies 315–318
  role-based 192–195, 322–330
  SaaS applications 370–371
  Shared Access Signatures 180–182
  stored access policy 182–183
  Virtual Network Service Endpoints 183–184
access control lists (ACLs) 193
access panel extension 368
ACE. *See* access control entries
ACLs. *See* access control lists
ACR. *See* Azure Container Registry
ACS. *See* Azure Container Services
Active Directory (AD) 311, 469
  registering application in 311–313
  service principals in 313–314
Active Directory Federation Services (ADFS) 469
  proxy monitoring 477–478
activity data 457–459
activity log alerts 119, 122–123
activity logs 456–459
activity reports 479
AD. *See* Active Directory
Adaptive application controls 357–358
Add-AzureRmAccount cmdlet 66
Add-AzureRmVirtualNetworkPeering cmdlet 221
Add-AzureRmVirtualNetworksubnetConfig cmdlet 237
Add-AzureRmVmssExtension cmdlet 134
ADFS. *See* Active Directory Federation Services
Alert Rules 189–190
alerts 39
  activity log 119, 122–123
  Azure Storage 189–190
  based on log search queries 461
  configuration 119–123
  critical, email notifications for 476–477
  metric 119–121
  security 359–361
Allow Gateway Transit option 258–259
Antimalware Assessment management solution 462–463
append blobs 158
application delivery controller (ADC) 232
Application Gateway (App Gateway)
  cookie-based session affinity 233
  creating 234–239
  deployment into virtual networks 234
  design and implementation 285–286
  end to end SSL 233
  implementing 232–239
  internal load balancers and 262
  load balancing 233
  secure sockets layer (SSL) offload 233
  sizes 234
  URL-based content routing 234
  web application firewall 233
application gateways 266
Application Insights 6, 35–39, 111, 116–117
application logs 115
applications. *See also* Web Apps
  Adaptive application controls 357–358
  adding users and groups to 369–370
  availability tests 37–39
  deploying to web apps 14
  desktop 495–496
  diagnostic logs 28–29

directory-aware 485
Enterprise 369
health check pages 49
integration with Azure AD 495–496
integration with Azure AD B2B 510–511
LOB 130
migrating on-premise to Azure 485
registering, in Azure AD 311–313
registering with Azure AD B2C 502–504
SaaS 363–368, 485
revoking access to 370–371
scaling, in ACS 148–149
service principals and 315
virtual 27
web 477–478, 495–496
firewalls 355–357
registering 374–375
application settings
connection strings and 18–20
App Service Certificate 344–346
App Service Certificates 23–24
App Service Environments (ASE) 4
app service plans
creating 2–6
Azure portal 4–5
CLI 6
PowerShell 5
instances, scaling 43–45
migrating web app to 15–16
pricing tiers 2–4
resource monitoring 34–35
scaling up or down 42–43
apt-get package manager 170
A records 21, 26
ARM template files 294
ARM template parameter files 294
artifact files 295
ASC. *See* Azure Security Center
ASE. *See* App Service Environments
ASN. *See* Autonomous System Numbers
AS numbers 247
ASR. *See* Azure Site Recovery
ASR Capacity Planner 395, 403
asverify records 176–177
async blob copy service 164–170
authentication 311–315, 371
multi-factor 488–493, 506
social identity provider 503–505
storage account 178
author ARM templates 294–308
automation 446
accounts, creating 416–417
certificates 426–427
configuration 432–441
connections 425–426
credentials 424–425
for cloud management 415–441
integrating, with Web Apps 428–431
modules 422–424
process 416–428
runbooks 418–431
schedules 421–422
variables 427–428
Autonomous System Numbers (ASN) 243
Autoscale 134
Autoscale feature 42, 44–46
availability
managing, with Load Balancer 128–129
sets 125–129
virtual machines 124–131
zones 124–125
availability sets 231
availability tests 37–39
Average Response Time metric 34
az account list-locations command 207
az ad app create command 312
az ad sp create command 313
AzCopy
async blob copy 167
blob and container management 163
az group create command 70, 309
az group deployment create command 309
az image command 97
az image list command 72
az lock create command 321
az network application-gateway address-pool command 286
az network application gateway command 239
az network command 239
az network lb command 282
az network nic create command 72
az network nic update command 274
az network nsg create command 72
az network public-ip create command 72
az network rule create command 72
az network vnet create command 71, 207
az network vnet peering list command 221

az network vnet subnet create command 71, 207
az policy assignment create command 319
az provider list CLI command 298
az role definition command 324
az role definition create command 329
az storage account create command 71
az storage blob generate-sas command 182
az storage container create command 161
Azure Active Directory 469
  adding application to 364
  adding custom domains 485–486
  Azure AD Connect and 487–488
  Editions 369
  integrating on-premises AD with 485–496
  integration in web and desktop applications 495–496
  Microsoft Graph API and 496
  Multi-Factor Authentication 488–493
  registering application in 311–313
  service principals in 313–314
  single-sign on 363–368
Azure Activity Log 456–459
Azure AD. *See* Azure Active Directory
Azure AD B2C (business to consumer) 371–377
Azure AD Business to Business (B2B) 469, 497
  application integration 510–511
  collaboration implementation 508–510
  partner users configuration 508–510
Azure AD Business to Consumer (B2C) 469, 497–508
  directory creation 497–502
  enabling multi-factor authentication 506
  registering application 502–504
  Self-Service Password Reset 507
  social identity provider authentication 503–505
  tenant creation 497–502
Azure AD Connect 469, 485, 487–488
Azure AD Connect Health 470–479
  activity reports 479
  ADFS and web application proxy server monitoring 477–478
  domain controller monitoring 473–476
  email notifications for critical alerts 476–477
  security reports 479
  sync engine and replication monitoring 470–473
  utilization reports 479
Azure AD Domain Join 493–495
Azure AD Domain Services 469, 479–485
  implementation 480–482
  joining Azure virtual machines to 482–483
  on-premise app migration 485
  VM management using Group Policy 484
Azure AD tenant 495
Azure AppService Web Apps. *See* Web Apps
Azure Automation. *See* automation
Azure Automation DSC service 88
Azure Backup
  agents 380–382
  backup and restore data 383–387
  encryption passphrase 382
Azure Cloud Shell 61, 73, 82, 206
Azure Container Registry (ACR) 142–144, 149
Azure Container Services (ACS) 57, 138–150
  Kubernetes cluster in 145
  managing containers with 146–148
  open-source tooling configuration 138–139
  scaling applications in 148–149
Azure Data Lake 192–195
Azure Diagnostics agent 119
Azure Diagnostics Extension 112
Azure Disk Encryption 105
Azure Domain Services
  adding and managing devices 493–495
Azure Fabric Controller 59
Azure files
  adding new share with 169–170
  hierarchy 168
  use cases 169
Azure File Service 106–110
  connecting to, outside of Azure 108–110
Azure Key Vault 19, 24, 179–180, 302, 336–342
  certificate management 342–344
Azure Load Balancer 128–129
Azure Log Analytics. *See* Log Analytics
Azure Managed Disks 126
Azure Monitor 111
Azure portal
  App Gateway creation in 234–236
  Application Insights in 36–37
  app service plan creation in 4–5
  ARM template deployment in 309
  automation account creation in 416–417
  availability set creation in 127
  Azure AD B2C tenant creation in 497–502
  Azure File Share creation in 106
  blob and container management in 159–160
  configuration of VMs as backend pool in 285–286
  deployment slot creation in 8–9

- diagnostics configuration 114–117
- DNS settings configuration in 212, 268–269
- enabling diagnostic logs in 28
- gateway subnet creation in 210–211
- handler mappings configuration in 26
- load balancer creation in 277–280
- Log Analytics workspace creation in 443
- migrating web app to separate app service plan in 15
- NSG creation in 270–273
- NSGs using 226–229
- public IP address creation with 266–267
- Recovery Services vault in 378–379
- static private IP addresses with 263
- swapping deployment swaps in 11–13
- using custom script extension 88
- viewing streaming logs in 32–33
- virtual machine creation in 60–66
- virtual machine scale set creation in 132–134
- virtual network creation in 201–204
- VNet peering in 218–221
- VNet-to-VNet connections in 251–257
- web app creation in 6–7

Azure private peering 244
Azure public peering 244
Azure resource connectivity 200
Azure Resource Manager (ARM)
- authentication 311–315
- private IP addresses 261
- templates 293–334
  - author 294–308
  - creating VM from 74–79
  - deployment 308–309
  - file types 294–295
  - functions 299
  - implementing 294–309
  - NIC resources 299–302
  - parameter file 305–307
  - schemas 295–296
  - virtual machine resource 302–308
  - virtual network resources 296–299
- virtual machines
  - alerts configuration 119–123
  - availability 124–129
  - diagnostics 112–119
  - monitoring 110–123
  - networking 260–286
  - resizing 130–132
  - scaling 129–137
- VNet peering 248–250
- VNets
  - connecting 217–222

Azure resource policy 316–317
Azure Security Center (ASC) 346–363
- applications node 355–358
- data collection 347
- email notifications 349
- enabling 346–350
- enabling protection for non-Azure computers 350
- Identity and Access solution 358–359
- networking node 353–354
- preventing security threats with 351–359
- pricing tiers 349–350
- recommendations 358
- responding to security threats with 359–363
- security policy 348
- Storage and Data node 354–356
- Threat intelligence detection capability 362–363

Azure Site Recovery (ASR) 393–412
- failover testing 401–402
- Hyper-V virtual machine protection 402–408
- Recovery Plan 401
- Unified Setup and vault credential 398
- virtual machine protection 408–412
- VMWare and physical machine protection 394–402

Azure Storage 155–177
- access control 178–184
- async blob copy service 164–170
- blob 155–164
- encryption 190–195
- monitoring and alerts 189–190
- replication options 163–164
- SMB file storage 168–170

Azure Storage Diagnostics 185–188
Azure Storage Explorer 162, 167–168, 182–183
Azure Storage Service Encryption 104–105
Azure Storage Service Encryption (SSE) 191
Azure-to-Azure Site Recovery 408–412
Azure Traffic Manager
- adding endpoints 51–52
- configuration 47–52
- profile creation 48–50

Azure Virtual Networks (VNets) 199–292
- address ranges 202–203
- App Gateway 232–239
- ARM VM networking 260–286
- communication strategy 287–289
- configuration 199–239, 251–260
- connecting, using VNet peering 217–222

connectivity 200–201
creating 200–207
design subnets 208–209
DNS setup 211–214
gateway subnets 210
introduction to 199
multi-site 251–260
network connectivity 239–259
network security groups 222–230
service chaining 250
system routes 214–215
user defined routes 214–217
virtual machine deployment into 230–232
az vm availability-set create command 72, 128
az vm create command 73, 98
az vm disk attach command 99
az vm extension set command 90
az vm generalize command 97
az vm list-vm-resize-options command 132
az vmss create command 137
az vmss update-instances command 137
az vm unmanaged-disk command 99

B

backend pools 129, 130, 278–279, 285–286
backup agents 380–382
Backup policy 385–386
backups 383–387
configuration 40–41
BGP routing protocol 243
BlobCache 98–100
blob files 93–94
blob snapshots 387–392
blob storage 155–164
account types 157
async blob copy service 164–170
CacheControl property of 174
CDN endpoints and 172
encryption 191
managing 159–164
metadata 158–159
time-to-live (TTL) period 173
types 158
Blob Storage account 94
block blobs 158
BMC. *See* baseboard management controller (BMC)
boot diagnostics 112, 117–118, 119
Bring Your Own Device (BYOD) policy 493
brute-force attacks 359
built-in roles 323–327

C

CacheControl HTTP header 173
Capacity Planner for Hyper-V Workloads 403–404
CDN. *See* Content Delivery Network
certificate authorities 23
certificate authority (CA) 343
certificate management 342–344
certificates 342
App Service Certificate 344–346
automation 426–427
creating 343–344
importing 342–343
SAML signing 365
Change Tracking management solution 463–466
CIFS. *See* Command Internet File System
cifs-utils package 170
Classless Inter-Domain Routing (CIDR) 200
CLI. *See* Command Line Interface
client-side telemetry data 39
cloud computing 1
cloud identity 469
cloud management
automation for 415–441
cloud services 351
Cloud Shell 73, 82, 206
CLR. *See* Common Language Runtime (CLR)
Cmdkey.exe 170
CNAME records 21, 22–23, 52, 176, 177
Command Line Interface (CLI) 1
App Gateway creation 239–240
application settings configuration 20
app service plan creation 6
ARM template deployment 309–310
async blob copy 166–167
availability set creation 128
Azure File Share creation 107
blob and container management 161–162
configuring VMs as backend pools 286
custom script extension 89–91
deployment slot creation 10
DNS settings configuration 213–214, 270

enabling diagnostic logs 29
load balancer creation 282–283
NSG creation 274
NSG creation and association 230
public IP addresses 268
resizing VM 132
retrieving diagnostic logs 32
static private IP addresses 264
swapping deployment slots 14
viewing streaming logs 33
virtual machine scale set creation 137
virtual network creation 206–208
VM creation 70–73
VM image creation 97
VNet peering 221–222
web app creation 8
Common Internet File System (CIFS) 106
communication strategy 287–289
community sourced runbooks 419
Conditional Access Policy 490
configuration
ACS 138–139
alerts 119–123
application settings 17–21
availability sets 125–129
availability zones 124–125
Azure diagnostics 112–119
Azure Storage Diagnostics 185–188
Azure Traffic Manager 47–52
backups 40–41
Content Delivery Network 170–176
custom domains 20–22, 176–177
DNS settings 212–214, 268–270
handler mappings 26
private static IP addresses 261–264
social identity providers 376–377
SSL certificates 22–26
virtual applications and directories 27
virtual machines 64–65, 82–92
virtual machine scale sets 132–137
virtual networks 199–239, 251–260
Virtual Network Service Endpoints 183–184
Web Apps 16–27
for scalability and resilience 42–51
configuration automation 432–441
Configuration Server 396, 397–398, 399–400
connections
automation 425–426
connection strings
application settings and 18–20
connectivity. *See also* network connectivity
VNets 200–201
containers 156
Azure Container Registry 142–144
Azure Container Services 138–150
images 139–142
managing 146–148, 159–164
metadata 158, 159
migrating workloads 149
root 156, 157
troubleshooting 150
Containers Monitoring Solution 149–150
Content Delivery Network (CDN) 2–16
configuration 170–176
custom domains for 176–177
endpoints 170–172
pricing tiers 171
profile creation 171
versioning assets with 174–176
content routing
URL-based 234
continuous integration/continuous delivery (CI/CD) workflow 433
Contributor built-in role 324–326
ConvertTo-SecureString cmdlet 110, 342
cookie-based session affinity 233
copyIndex() function 300
CPU Percentage 35
creation, renaming, updating, and deletion (CRUD) operations 457
credentials
authentication of 488–493
automation 424–425
on-premises 486
custom domains 485–486
associating with web apps 22
configuration, for web apps 20–22
customer-managed DNS settings 213
custom resource policy 317–318
Custom Script Extension 82, 88–91
custom security alerts 360–362
custom visualizations
in Log Analytics 452–455

D

Dashboard tile 453–455
data
- activity 457–459
- backup and restore 383–387
- diagnostic, analyzing 187–188
- logging 187, 188
- machine 441–443, 448–449
- metrics 187, 188
- querying. *See* queries
- resource 457–459

data analysis. *See* Log Analytics
database as a service (DBaaS) 354
data churn 394
data collection 347, 441–466
data disks 95
data encryption. *See* encryption
data protection 335–336. *See also* security
- encryption 342–346

DCs. *See* domain controllers
debugging
- remote, of VMs 91–92

default tags 225
deployment
- applications 14
- ARM template 308–309
- ARM templates 74–77
- backup agents 380–382
- Web Apps 2–16

deployment script files 294
deployment slots
- cloning existing 9
- creating
  - in Azure portal 8–9
  - in CLI 10
  - in PowerShell 9–10
- defining 8–10
- multi-phase 11, 12–13
- production 8
- swapping 11–14
  - Azure portal 11–13
  - CLI 14
  - with PowerShell 13

design subnets 208–209
Desired State Configuration (DSC) 432–441
- Azure Automation 434–436
  - monitor and update machine configurations with 436–441
- configurations
  - creating 432
  - managing 433
- metaconfigurations 438–441
- node configurations
  - generating 434–436
- nodes
  - adding 436–441
  - registration options 436
- resources
  - built-in 434
  - custom 434
  - importing 433–434

Desired State Configuration (DSC) extension 82, 83–88
desktop applications 495–496
DevOps principles 433
diagnostic logs
- application 28–29
- enabling 27–29
- locations 30
- retrieving 29–32
  - in PowerShell 32
  - using FTP 30
  - using Site Control Manager 30–31
- streaming, viewing 32–33
- web server 28–29

diagnostics
- Azure Storage Diagnostics 185–188
- boot 112, 117–118, 119
- configuring Azure 112–119
- data analysis 187–188
- guest operating system 112
- Linux, enabling and configuring 118–119

DigiCert 343
directories
- virtual 27

Direct Server Return (DSR) 283–284
Disable-AzureRmTrafficManagerEndpoint cmdlet 52
disaster recovery. *See* recovery services
disk caching 98–100
disk encryption 104–106
disk redundancy 103–104
disks
- managed 126
- mounting 104
- VM 95

DNS names 48
DNS records
- adding 21–23
- updating 48

docker-compose command 140
Docker containers 6
Domain Controller (DC) 213
domain controllers (DCs)
  monitoring 473–476
Domain Naming Service (DNS) 211–214
  at NIC level 268–270
domains
  custom 485–486
    for storage and CDN 176–177
  custom, configuration of 20–22
DPM protection agent 380
DSR. *See* Direct Server Return
dynamic IP addresses 261

E

elastic scale 42
email notifications 349
encryption
  Azure Data Lake 192–195
  data 342–346
  disk 104–106
  keys, create and import 336–342
  passphrase, Azure Backup 382
  storage 190–195
Enterprise applications 369
enterprise Azure scaffold 316
error messages 28
event categories 122
event log data 115
Event Tracing for Windows (ETW) 115–116
ExpressRoute 242–245
External ASEs 4

F

Facebook 371–377, 503–505
failed requests
  logs style sheet file 30
  tracing 28
failover testing 401–402
federation
  with public consumer identity providers 371–377
federation-based single sign-on 363–366
files
  change tracking 464
  hierarchy 168
  purging 175
file shares
  adding new 169–170
file share service 106–110
file system permissions 193–195
firewall rules 193–194
firewalls 201
  network 353
  web application 353, 355–357
  web application firewall 233
FTP client
  for retrieving log files 30
Fully Qualified Domain Name (FQDN) 211, 265

G

gateway subnets 210–211, 252–253
generate-ssh-keys parameter 71
geo-redundant storage (GRS) 94
Geo-Redundant Storage (GRS) 392–393
geo-replication 392–393
Get-AzureKeyVaultCertificateOperation cmdlet 343, 344
Get-AzureRmApplicationGatewayBackendAddressPool cmdlet 286
Get-AzureRmApplicationGateway cmdlet 286
Get-AzureRmLocation cmdlet 204
Get-AzureRmNetworkInterface cmdlet 263, 269, 273, 286
Get-AzureRmRemoteDesktopFile cmdlet 80
Get-AzureRmResourceGroup cmdlet 67, 204
Get-AzureRmResourceProvider cmdlet 298
Get-AzureRmStorageAccount cmdlet 67
Get-AzureRmStorageAccountKey cmdlet 110
Get-AzureRmStorageKey cmdlet 165
Get-AzureRmStoragerAccountKey cmdlet 107
Get-AzureRmTrafficManagerProfile cmdlet 51
Get-AzureRmVirtualNetwork cmdlet 230
Get-AzureRMVirtualNetwork cmdlet 237
Get-AzureRmVirtualNetworkPeering cmdlet 221
Get-AzureRmVM cmdlet 99
Get-AzureRmVMImageOffer cmdlet 69
Get-AzureRmVMImagePublisher cmdlet 69
Get-AzureRmVMImageSku cmdlet 69
Get-AzureRmVMSize cmdlet 131
Get-AzureRmWebAppSlot cmdlet 10
Get-AzureStorageBlobCopyState cmdlet 166
Get-AzureWebsiteLog cmdlet 33
Get-PhysicalDisk cmdlet 103
GlobalSign 343

Google 371–377
Group Policy Objects (GPOs) 484
groups
  adding to applications 369–370
GRS. *See* geo-redundant storage
guest operating system diagnostics 112

H

handler mappings 26
hard disk drives (HDDs) 63
hardware security modules (HSMs) 180, 336
HCM. *See* Hybrid Connection Manager
health check pages 49
health probes 276–277, 279
host caching 99
hot access tier 157
HSMs. *See* hardware security modules
HTTP GET requests 37
HTTP probe 276
HTTP probes 129
HTTPS traffic 233
hub and spoke network topology 320
hybrid cloud 239
Hybrid Connection Manager (HCM) 288
hybrid connections 288–289
hybrid network connectivity 239–259
Hybrid Runbook Worker 419–421
Hyper-V 58
Hyper-V-based workloads 402–408
Hyper-V hosts 407
Hyper-V sites 405–407

I

identity infrastructure 470–479
identity management 469–514
  Azure Active Directory 485–496
  Azure AD B2C 497–508
  Azure AD Connect Health 470–479
  Azure AD Domain Services 479–485
  social identity provider authentication 503–505
identity providers 376–377
ILB ASEs 4
Import-AzureKeyVaultCertificate cmdlet 343
Independent Software Vendors (ISV) 495
Infrastructure as a Service (IaaS) 351
infrastructure-as-code (IaC) assets 433
internal load balancer (ILB) 4
internal load balancers (ILBs) 262
Internet connectivity 200
Internet-facing load balancers 266
IP addresses
  allocation of 262
  default tags 225
  dynamic 261
  private 261
  private static 261–264
  public 247, 261, 264–268
  static private 263–264
  subnets 208, 210–211
  VNets 202–203
IP address spaces 247
IP forwarding 274–275
IPSec VPN 240

J

JavaScript Object Notation (.json) files 77
Just in time (JIT) VM access 352–353

K

Key Vault 302, 336–342
  certificate management 342–344
Kubernetes 147–149
  clusters 145
  monitoring 149–150
Kubernetes API endpoints 138

L

large scale sets 132
LCM. *See* Local Configuration Manager
line of business (LOB) applications 130
Linux agent 60
Linux-based virtual machines
  connect and mount Azure File from 110
  connecting to 81–82
  custom script extension with 89–91
  diagnostics, enabling and configuring 118–119
Linux distributions
  for VMs 58–59
load balancers 128–129, 233, 262, 266, 275–283
  health probes 276–277
Local Configuration Manager (LCM) 433, 436

locally redundant storage (LRS) 94
lock resources 319–321
Log Analytics 112, 441–466
  connecting Activity Log to 457–458
  custom visualizations 452–455
  data sources
    connecting 444–448
    searching 441–452, 449–452
  default workspace 442–443
  malware status monitoring with 462–463
  management solutions 448–449
  monitoring system updates with 459–461
  queries 458–459
  server configuration change tracking in 463–466
  visualizing Azure resources across multiple subscriptions 456–457
  workspace creation 443–444
  workspace ID and keys 446–447
  writing activity data to 457–459
Log Analytics query language 361
log files. *See also* diagnostic logs
  retrieving
    using FTP 30
    using Site Control Manager 30–31
logging data 187, 188
Logic Apps 121
Login-AzureRmAccount cmdlet 66, 204
Log Search feature 449–452
LRS. *See* locally redundant storage

M

machine configurations 436–441
machine data 441–443, 448–449
makecert.exe 80
malware 462–463
managed disks 95, 126
managed images 96, 97
management policies 315–318
man-in-the-middle attacks 80
MARS. *See* Microsoft Azure Recovery Services
Master Target Server 396
Memory Percentage 35
metadata
  setting, with storage 158–159
metric alerts 39, 119–121
metric-based scale conditions 44–45
metrics data 187, 188
Microsoft Azure AppService Web Apps. *See* Web Apps
Microsoft Azure Datacenter
  IP ranges 225
Microsoft Azure Linux Agent (waagent) 59–60, 95
Microsoft Azure Recovery Services (MARS) agent 380–382
Microsoft Graph API 496
Microsoft Monitoring Agent (MMA) 446–448
Microsoft peering 244
migration
  lift and shift 106
  on-premises apps to Azure 485
  workloads 149
MMA. *See* Microsoft Monitoring Agent
mobile devices 493–495
Mobility service extension 396, 409
modules 422–424
monitoring
  ARM VMs 110–123
  Azure Storage 189–190
  clusters 150
  Kubernetes 149–150
  options 111–112
Most attached resources 359
multi-factor authentication (MFA) 488–493, 506
multi-phase deployment swaps 11, 12–13
Multiprotocol Label Switching (MPLS) 242
multi-site network connectivity 239–259
multi-step web tests 37

N

net use command 110
network connectivity 239–259
  ExpressRoute 242–245
  network prerequisites 247–248
  on-premises 260
  VNet peering 248–250, 257–259
  VNet-to-VNet connections 251–256
  VPN Gateway 246–247
network interface (NIC) 268–270
  associating NSG with 270–274
  resources 299–302
network security groups (NSGs) 208, 209, 222–230, 270–278, 353, 391
  associating 225–229
  default rules 224–225
  default tags 225

properties 222, 222–223
rules 222–223
Network Security Groups (NSGs) 4
network traffic 248, 249
Network Watcher 111
New-AzureKeyVaultCertificateOrganizationDetails cmdlet 343
New-AzureKeyVaultCertificatePolicy cmdlet 343
New-AzureRmADApplication cmdlet 312
New-AzureRmAppServicePlan cmdlet 5
New-AzureRmAutomationCertificate cmdlet 427
New-AzureRmAutomationVariable cmdlet 428
New-AzureRmAvailabilitySet cmdlet 68, 128
New-AzureRmImage cmdlet 97
New-AzureRmImageConfig cmdlet 97
New-AzureRmKeyVault cmdlet 339
New-AzureRmNetworkInterface cmdlet 69
New-AzureRmNetworkSecurityGroup cmdlet 68, 273
New-AzureRmNetworkSecurityGroup PowerShell cmdlet 229
New-AzureRmNetworkSecurityRuleConfig cmdlet 68, 229, 273
New-AzureRmOperationalInsightsWorkspace cmdlet 444
New-AzureRmPolicyDefinition cmdlet 318
New-AzureRmPublicIpAddress cmdlet 267
New-AzureRMPublicIpAddress cmdlet 237
New-AzureRmResourceGroup cmdlet 67, 204, 308, 338
New-AzureRmResourceGroupDeployment cmdlet 79, 308
New-AzureRmResourceLock cmdlet 321
New-AzureRmRoleDefinition cmdlet 329
New-AzureRmStorageAccount cmdlet 67
New-AzureRmTrafficManagerEndpoint cmdlet 51
New-AzureRmTrafficManagerProfile cmdlet 50
New-AzureRmVirtualNetwork cmdlet 205, 213
New-AzureRmVirtualNetworkSubnetConfig cmdlet 67, 205
New-AzureRmVMConfig cmdlet 69, 70
New-AzureRmVmssConfig cmdlet 134
New-AzureRmWebApp cmdlet 7
New-AzureRmWebAppSlot cmdlet 9–10
New-AzureStorageAccount cmdlet 164
New-AzureStorageBlobSASToken cmdlet 181
New-AzureStorageContainer cmdlet 160–161, 165
New-AzureStorageContext cmdlet 107, 165
New-AzureStorageShare cmdlet 107
New-PSDrive cmdlet 110
New-SelfSignedCertificate cmdlet 80
New-VirtualDisk cmdlet 103
NIC. *See* network interface
NSGs. *See* network security groups

O

OAuth2 protocol 312
OMS. *See* Operating Management Suite; *See* Operations Management Suite
OMS Gateway 442
OMS Portal 461
on-premises connectivity 201, 260
on-premises credentials 486
on-premises environment
data collection in 442
on-premises infrastructures 287
OpenID Connect 363, 374
open-source tooling 138–139
operating system disks 95
operating system images 95–98
creating VMs from 97–98
managed 96, 97
unmanaged 96–97
Operations Management Suite (OMS) 149–150, 378
Organization Units (OUs) 484
Overview tile 452–453
Owner built-in role 324
owning groups 195
owning users 195–196

P

page blobs 158
password-based single sign-on 366–368
passwords
Self-Service Password Reset 507
synchronization 482
performance counters 113, 115
permissions 311
file system 193–195
VMWare 398–399
Personal Information Exchange (.pfx) files 23
ping tests 38
platform as a service (PaaS) 354
Platform-as-a-Service (PaaS) 1
point-to-site virtual private network (VPN) 240–241
port reuse 283

PowerShell
  App Gateway creation in 236–238
  application settings configuration in 20
  app service plan creation in 5
  ARM template deployment using 308
  async blob copy with 164–166
  automation account creation using 417
  availability set creation with 128
  Azure File Share creation with 107–108
  blob and container management with 160–161
  configuring VMs as backend pools with 286
  connect and mount Azure File using 110
  Custom Script Extension 82, 88–91
  deployment slot creation with 9–10
  Desired State Configuration 432–441
  DNS settings configuration with 213–214, 269
  DSC extension 82, 83–88
  enabling diagnostic logs with 29
  load balancer creation with 280–282
  Log Analytics workspace creation with 444
  modules 422–424
  NSG creation with 273–274
  NSGs using 229–230
  public IP address creation using 267–268
  Recovery Services vault with 379
  remoting 80–81
  resizing VM with 131
  retrieving diagnostic logs with 32
  runbooks 416–428
  static private IP addresses with 263–264
  swapping deployment slots with 13
  Traffic Manager profile creation with 50
  unmanaged VM image creation with 96–97
  viewing streaming logs in 33
  virtual machine scale set creation with 134–136
  virtual network creation in 204–205
  VM creation with 66–70
  VNet peering with 221
  web app creation in 7
PowerShell cmdlets. *See also* specific cmdlets
  Azure 1
PowerShell Gallery 423
Premium storage 101–102
pricing tiers 336, 349–350
primary keys 179
private IP addresses 263–264
private static IP addresses
  configuration 261–264
proactive diagnostic alerts 39
process automation 416–428
Process Server 396
production deployment slot 8
ProvisionVMAgent parameter 69, 83
public IP addresses 261, 264–268
Publish-AzureRmVMDscConfiguration cmdlet 85

Q

query strips 175–176

R

RAID 0 disk striping 103
RBAC. *See* role based access control
read-access geo-redundant storage (RA-GRS) 94
Read Access-Geo Redundant Storage (RA-GRS) 393–394
Reader built-in role 326–327
Recover Data Wizard 384
recovery services
  Azure Site Recovery 393–412
  backup agents 380–382
  backup and restore data 383–387
  geo-replication 392–393
  planning 394–396
  snapshots 387–392
  vault 378–379, 393, 398, 405
Recovery Services Agent 403–404, 407
redundancy 125
Register-AzureRmAutomationDscNode cmdlet 437
registry change tracking 464
remote debugging
  of VMs 91–92
remote desktop protocol (RDP) 79–80, 359
Remote Gateways 257–259
Remove-AzureRmPolicyDefinition cmdlet 318
Remove-AzureRmTrafficManagerEndpoint
    cmdlet 51–52
replication 400–401, 409–412
  monitoring 470–473
resizing
  virtual machines 130–132
resource data 457–459
resource locks 319–321
resource policies
  assignment 318–319
  Azure 316–317

built-in 316
custom 317–318
resource schemas 296
role assignment 327–328
role-based access control 192–195
role based access control (RBAC) 95
role-based access control (RBAC) 322–330
custom roles 328–329
standard roles, implementing 322–328
root container 156, 157
route tables 208
Run As accounts 417
Runbook Gallery 418
runbooks 121, 416–431

S

SAML 2.0 363
SAML signing certificates 365
SAN. *See* subject alternative name
Save-AzureRmVMImage cmdlet 96
Save-AzureWebsiteLog cmdlet 32
scalability features 42–47
scaling
Azure VMs 129–137
basic tier Web App 428–431
in ACS 148–149
virtual machine scale sets 132–137
schedule-based scale conditions 46–47
schedules
automation 421–422
$schema property 305
SCOM. *See* System Center Operations Manager; *See* System Center Configuration Manager
secondary keys 179
secure shell (SSH) protocol 81–82
secure sockets layer (SSL)
end to end 233
secure sockets layer (SSL) offload 233
security 335–377. *See also* access control
authentication 371
Azure Security Center 346–363
encryption 336–342
multi-factor authentication 488–493, 506
network security groups 270–274, 391
single sign-on 363–368
social accounts 371–377
SSL/TLS certificates 342–346
threat prevention 351–359
threat response 359–363
Security alerts 359–363
security policy 348
security reports 479
security rules 208, 271–272
Self-Service Password Reset 507
self-signed SSL certificates 80
server configuration changes 463–466
Server Messaging Block (SMB) protocol 106
Server Name Indication (SNI) 25
Service Bus Relay 288
service chaining 250
Service Health 39
service level agreements (SLAs) 125
service principals 313–314
Set-AzureKeyVaultCertificateIssuer cmdlet 344
Set-AzureKeyVaultSecret cmdlet 342
Set-AzureRmApplicationGatewayBackendAddressPool cmdlet 286
Set-AzureRmNetworkInterface cmdlet 263, 269, 273
Set-AzureRmOsDisk cmdlet 98
Set-AzureRmVirtualNetwork cmdlet 237
Set-AzureRmVirtualNetworkSubnetConfig cmdlet 230
Set-AzureRmVMCustomScriptExtension cmdlet 89
Set-AzureRmVMDataDisk cmdlet 99
Set-AzureRmVMOperatingSystem cmdlet 69, 80, 81, 83
Set-AzureRmVMOSDisk cmdle 99
Set-AzureRmVMOSDisk cmdlet 98
Set-AzureRmVMSourceImage cmdlet 69
Set-AzureRmWebApp cmdlet 20, 29
Set-AzureStorageAccount cmdlet 164
Set-AzureStorageBlobContent cmdlet 161
Set-AzureStorageServiceLoggingProperty cmdlet 186
Set-AzureStorageServiceMetricsProperty cmdlet 186
shadow IT system 315
Shared Access Signature (SAS) 180–182
shared access signature (SAS URL) 88
single sign-on 363–368
federated 363–366
password-based 366–368
Site Control Manager (Kudu) 30–31
Site Recovery Deployment Planner (SRDP) 394
Site Recovery Provider 403–404, 405–406
site-to-site (S2S) virtual private networks 240–242, 247, 287
SLAs. *See* service level agreements
SMB. *See* Server Messaging Block

SMB file storage 168–170
snapshots 387–392, 408
social identity provider authentication 503–505
social identity providers 376–377
social media accounts 469
software as a service (SaaS) applications 485
  revoking access to 370–371
  single sign-on with 363–368
solid state disks (SSDs) 63
SQL Server Always On Availability Groups 284
SSDs. *See* solid state disks
SSH certificates 62–63
SSL certificates
  App Service Certificates and 23–24
  configuration of 22–26
  self-signed 80
  third-party 23, 25
SSL/TLS certificates 342–346
Standard Azure Storage account 100–101
standard certificates 23
Standard storage 100–101
Start-AzureStorageBlobCopy cmdlet 164–165, 166
static IP addresses 261–264
  enabling on VMs 263–264
storage 155–198
  Azure Storage 155–177
  Azure Storage Diagnostics 185–188
  blob 155–164
    async blob copy service 164–170
  capacity planning 100–104
  custom domains for 176–177
  disk caching 98–100
  encryption 190–195
  Geo-Redundant Storage 392–393
  monitoring and alerts 189–190
  redundancy type 379
  setting metadata with 158–159
  SMB file storage 168–170
  virtual machines 93–110
    blob types 93–94
    storage accounts 93–94
storage accounts
  access control 178–184
  accessing content from CDN instead of 172–173
  blob 157
  custom domains 176–177
  diagnostics 185–188
  entities and hierarchy relationships 156
  key management 178–180
  replication options 163–164
  root container 157
  types 157
  zone replicated 164
Storage Explorer
  async blob copy 167–168
  blob and container management with 162
  stored access policies 182–183
storage pools 103
Storage Spaces 103
stored access policy 182–183
streaming log files 32–33
subject alternative name (SAN) 23
subnets
  associating NSG with 273–274
  deleting 209
  design 208–209
  gateway 210–211, 252–253
  NSGs with 222, 226–230
  properties 209
subscription policies 311
subscriptions 248
  visualizing resources across multiple 456–457
super users 194
support devices 247
Swap-AzureRmWebAppSlot cmdlet 13
Sync Error 476
synchronization services 470–479
sysprep.exe tool 96
System Center Configuration Manager (SCOM) 442
System Center Operations Manager (SCOM) 38
System Center Virtual Machine Manager (SCVMM) 404–405
system routes 214–215
System Update Assessment management solution 459–461
system updates 459–461

T

TCP probe 276
telemetry data 39
TemplateParameterFile parameter 79
TemplateParameterObject parameter 79
TemplateParameterUri parameter 79
temporary disks 95
Thales nShield family 180

Threat intelligence detection capability 362–363
time-to-live (TTL) period 173
top-AzureRmVM cmdlet 73
trace data 115
traffic encryption/decryption 233
traffic filtering 201
Transport Layer Security (TLS) 342

U

unmanaged disks 95
unmanaged images 96–97
update-AzureRmVM cmdlet 99
Update-AzureRmVmssInstance cmdlet 137
URL-based content routing 234
URLs
  Azure Automation 420
Usage location 489
user defined routes (UDRs) 214–217, 274–275
users
  adding to applications 369–370
utilization reports 479

V

variables
  automation 427–428
vault credentials 380
vCenter/vSphere server 400–401
View Designer 452–453
virtual appliances 274
virtual applications
  configuration 27
virtual CPUs (vCPUs) 63
virtual directories
  configuration 27
virtual hard disks (VHDs) 95
virtual hard disk (VHD) files 67
virtual machine scale sets (VMSS) 57, 95, 129, 132–137
  upgrading 137
virtual machines (VMs) 57
  adding DSC node 436–438
  agents 83
  ARM
    alerts configuration 119–123
    availability 124–129
    diagnostics 112–119
    monitoring 110–123
    networking 260–286
    resizing 130–132
    scaling 129–137
  ASR Azure to Azure protection 408–412
  Azure Container Services 138–150
  configuration 64–65
    management 82–92
  configuring as backend pools for App Gateway 285–286
  connecting to 79–82
  connecting to Log Analytics workspace 444–446
  creating 60–79
    from ARM template 74–79
    from images 97–98
    in Azure portal 60–66
    in PowerShell 66–70
    with CLI 70–73
  deployment into virtual network 230–232
  disk caching 98–100
  disk encryption 104–106
  disk redundancy 103–104
  enabling static private IP addresses on 263–264
  GPOs with 484
  Hyper-V
    protection of 402–408
  IaaS, backing up 384–386
  joining to a Domain 482–483
  Just-in-time access 352–353
  Linux distributions for 58–59
  name resolution 268
  operating system images 95–98
  public IP addresses 266
  redundancy for 125
  remote debugging 91–92
  replication 400–401, 409–412
  resources 302–308
  setting size of 64, 72
  snapshots 391
  stopping 73
  storage 93–110
    account replication 94
    accounts 93–94
    Azure File Service 106–110
    blob types 93–94
    capacity planning 100–104
    disks 95
    overview 93–95
  workloads
    deployment 58–82
    identify and run 58–60
virtual network resources 296–299

virtual networks. *See* Azure Virtual Networks
Virtual Network Service Endpoints (VSPE) 178, 183–184
virtual private networks (VPNs)
  devices 248
  ExpressRoute 242–245
  point-to-site 240–241
  site-to-site 240–242, 287
  support devices and software solutions 247
Visual Studio Cloud Explorer 91
Visual Studio Code 295
Visual Studio Community 2017 295
VMs. *See* virtual machines
VMSnapshot extension 380, 386
VMSnapshotLinux extension 380, 386
VMSS. *See* virtual machine scale sets
VNet peering 217–222, 248–250, 251, 257–259
VNets. *See* Azure Virtual Networks
VNet-to-VNet connections 251–257
VPN Gateways 246–247, 253–255, 260, 266
VPNs. *See* virtual private networks

W

WAF. *See* web application firewall
web application firewall (WAF) 233, 353, 355–357
web applications
  integration with Azure AD 495–496
  proxy monitoring 477–478
  registering 374–375
Web Apps 1–56
  application settings 17–21
  app service plans 2–6
  availability tests 37–39
  Azure Traffic Manager for 47–51
  basic tier 428–431
  configuration 16–27
    application settings 17–21
    backups 40–41
    custom domain 20–22
    for scale and resilience 42–52
    handler mappings 26
    SSL certificates 22–26
    virtual applications and directories 27
  connection strings 18–20
  creating 6–8
    Azure portal 6–7
    in CLI 8
    in PowerShell 7
  deploying application to 14
  deployment 2–16
  deployment slots
    defining 8–10
    swapping 11–14
  diagnostic logs
    enabling 27–29
    retrieving 29–32
  integrating Azure Automation with 428–431
  introduction to 1
  migration to separate App Service Plan 15–16
  monitoring 27
    app service plan resources 34–35
    Azure services 39–40
    resources 33–34
    with Application Insights 35–39
  multiple deployments of 47
  restoring from backup 41–42
Web-Asp-Net45 feature 83
webhooks 121
Web-Server role 83
web servers
  diagnostic logs 28–29
web tests alerts 39
wildcard certificates 23
Windows 10 493–495
Windows Explorer 108–109
Windows Management Framework version 5 (WMF 5) 438
Windows PowerShell. *See* PowerShell
Windows Remote Management (WinRM) 80–81
Windows Server 2003 58
Windows virtual machines
  connecting to 79–81
  diagnostics, enabling and configuring 112–118
WinRMHttp 80
WinRMHttps 80
workloads
  Hyper-V-based 402–408
  migrating 149
  on virtual machines
    identify and run 58–60
WS-Federation 363

X

xPSDesiredStateConfiguration module 84

Y

yum package manager 170

Z

zone redundant storage (ZRS) 94
zone-replicated storage accounts 164

This page intentionally left blank

About the authors

RICK RAINEY is Principal Program Manager in Microsoft's Azure Customer Advisory Team (CAT).

MICHAEL WASHAM, Microsoft Azure MVP, Insider and Advisor, is CEO of the cloud readiness company Opsgility. Michael has extensive history in the IT Industry where he has worked as an IT Professional, developer, Evangelist, and Program Manager before turning to his passion of enabling companies of all sizes make the digital transformation to the cloud. Michael is an avid blogger, author speaker, and trainer on cloud computing, debugging, and DevOps

DAN PATRICK is the Chief Cloud Strategist for Opsgility http://www.opsgility.com. He has an extensive background in the IT industry with a focus on Virtualization, Networking and Identity management. Dan is a 15 year veteran of Microsoft, where was a Principal Consultant and Practice Manager. Today he still works with Microsoft as an Azure MVP and advisor. He regularly speaks about the cloud all over the world, and you can follow him on twitter @deltadan.

STEVE ROSS is Partner Technology Strategist with Microsoft's One Commercial Partner Technology Team and is heavily involved in helping partners build services using cloud technologies. Steve has also been a Principal Cloud Solution Architect with Opsgility, building Microsoft Azure training content and teaching enterprise customers and Microsoft partners and employees all around the world.