

Azure Networking Cookbook

Practical recipes to manage network traffic in Azure, optimize performance, and secure Azure resources



Packt

www.packt.com

Mustafa Toroman

Azure Networking Cookbook

Practical recipes to manage network traffic in Azure, optimize performance, and secure Azure resources

Mustafa Toroman

Packt

BIRMINGHAM - MUMBAI

Azure Networking Cookbook

Copyright © 2019 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Vijn Boricha

Acquisition Editor: Shrilekha Inani

Content Development Editor: Ronn Kurien

Technical Editor: Pratik Shet

Copy Editor: Safis Editing

Project Coordinator: Jagdish Prabhu

Proofreader: Safis Editing

Indexer: Tejal Daruwale Soni

Graphics: Tom Scaria

Production Coordinator: Saili Kale

First published: March 2019

Production reference: 1290319

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-78980-022-7

www.packtpub.com



mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

Packt.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Contributors

About the author

Mustafa Toroman is a program architect and senior system engineer with Authority Partners. With years of experience of designing and monitoring infrastructure solutions, lately, he focuses on designing new solutions in the cloud and migrating existing solutions to the cloud. He is very interested in DevOps processes, and he's also an Infrastructure-as-Code enthusiast. Mustafa has over 30 Microsoft certifications and has been an MCT for the last 6 years. He often speaks at international conferences about cloud technologies, and he has been awarded the MVP award for Microsoft Azure for the last three years in a row.

Mustafa also authored *Hands-On Cloud Administration in Azure* and co-authored *Learn Node.js with Azure*, both published by Packt.

About the reviewer

Kapil Bansal is a technical consultant at HCL Technologies in India. He has more than eleven years of experience in the IT industry. He has worked on Microsoft Azure (PaaS, IaaS, Kubernetes, and DevOps), ALM, ITIL, and Six Sigma. He provides technical supervision and guidance during clients' engagement execution. His expertise includes strategic design and architectural mentorship, assessments, POCs, sales life cycles, consulting on engagement processes, and so on. He has worked with companies such as IBM India Pvt Ltd., NIIT Technologies, Encore Capital Group, and Xavient Software Solutions, and he has served clients based in the United States, the United Kingdom, India, and Africa, including T-Mobile, WBMI, Encore Capital, and Airtel.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface	1
Chapter 1: Azure Virtual Network	7
Technical requirements	7
Creating a virtual network in the portal	8
Getting ready	8
How to do it...	8
How it works...	10
Creating a virtual network with PowerShell	10
Getting ready	10
How to do it...	11
How it works...	11
Adding a subnet in the portal	11
Getting ready	11
How to do it...	11
How it works...	14
Adding a subnet with PowerShell	14
Getting ready	15
How to do it...	15
How it works...	15
There's more...	16
Changing the address space size	16
Getting ready	16
How to do it...	16
How it works...	17
Changing the subnet size	17
Getting ready	18
How to do it...	18
How it works...	19
Chapter 2: Virtual Machine Networking	20
Technical requirements	20
Creating Azure VMs	20
Getting ready	20
How to do it...	21
How it works...	26
See also	26
Viewing VM network settings	27
Getting ready	27
How to do it...	27

How it works...	28
Creating a new network interface	29
Getting ready	29
How to do it...	29
How it works...	31
Attaching a network interface to a VM	31
Getting ready	31
How to do it...	31
How it works...	32
Detaching a network interface from a VM	32
Getting ready	32
How to do it...	33
How it works...	33
Chapter 3: Network Security Groups	34
Technical requirements	34
Creating a new NSG in a portal	35
Getting ready	35
How to do it...	35
How it works...	36
Creating a new NSG with PowerShell	36
Getting ready	36
How to do it...	36
How it works...	37
Creating a new allow rule in NSG	37
Getting ready	37
How to do it...	37
How it works...	39
Creating a new deny rule in NSG	39
Getting ready	39
How to do it...	39
How it works...	41
Creating a new NSG rule with PowerShell	41
Getting ready	41
How to do it...	41
How it works...	42
There's more...	42
Assigning an NSG to a subnet	42
Getting ready	42
How to do it...	43
How it works...	45
Assigning an NSG to a network interface	45
Getting ready	45
How to do it...	46
How it works...	48

Assigning an NSG with PowerShell	48
Getting ready	48
How to do it...	48
How it works...	49
Creating an Application Security Group (ASG)	49
Getting ready	49
How to do it...	49
How it works...	50
Associating an ASG with a VM	50
Getting ready	51
How to do it...	51
How it works...	52
Creating rules with an NSG and an ASG	52
Getting ready	52
How to do it...	53
How it works...	53
Chapter 4: Managing IP Addresses	54
Technical requirements	54
Creating a new public IP address in the portal	55
Getting ready	55
How to do it...	55
How it works...	57
Creating a new public IP address with PowerShell	57
Getting ready	57
How to do it...	57
How it works...	58
Assigning a public IP address	58
Getting ready	58
How to do it...	58
How it works...	60
Unassigning a public IP address	61
Getting ready	61
How to do it...	61
How it works...	62
Creating a reservation for a public IP address	62
Getting ready	62
How to do it...	63
How it works...	64
Removing a reservation for a public IP address	64
Getting ready	64
How to do it...	64
How it works...	65
Creating a reservation for a private IP address	66
Getting ready	66

How to do it...	66
How it works...	67
Changing a reservation for a private IP address	68
Getting ready	68
How to do it...	68
How it works...	69
Removing a reservation for a private IP address	70
Getting ready	70
How to do it...	70
How it works...	71
Chapter 5: Local and Virtual Network Gateways	72
 Technical requirements	72
 Creating a local network gateway in the portal	73
Getting ready	73
How to do it...	73
How it works...	74
 Creating a local network gateway with PowerShell	75
Getting ready	75
How to do it...	75
How it works...	75
 Creating a virtual network gateway in the portal	75
Getting ready	76
How to do it...	76
How it works...	78
 Creating a virtual network gateway with PowerShell	79
Getting ready	79
How to do it...	79
How it works...	80
 Modifying the local network gateway settings	80
Getting ready	80
How to do it...	80
How it works...	81
Chapter 6: Creating Hybrid Connections	82
 Technical requirements	83
 Creating a Site-2-Site connection	83
Getting ready	83
How to do it...	84
How it works...	88
 Downloading the VPN device configuration from Azure	88
Getting ready	88
How to do it...	89
How it works...	91
 Creating a Point-2-Site connection	91

Getting ready	92
How to do it...	96
How it works...	99
Creating a VNet-2-VNet connection	99
Getting ready	100
How to do it...	100
How it works...	104
Connecting VNets using network peering	104
Getting ready	105
How to do it...	105
How it works...	108
Chapter 7: DNS and Routing	109
Technical requirements	110
Creating an Azure DNS zone	110
Getting ready	110
How to do it...	110
How it works...	111
Creating a new record set and record in Azure DNS	112
Getting ready	112
How to do it...	112
How it works...	115
Creating a route table	115
Getting ready	115
How to do it...	115
How it works...	116
Changing the route table	116
Getting ready	117
How to do it...	117
How it works...	117
Associating a route table to a subnet	118
Getting ready	118
How to do it...	118
How it works...	121
Dissociating a route table from the subnet	121
Getting ready	122
How to do it...	122
How it works...	125
Creating a new route	125
Getting ready	125
How to do it...	125
How it works...	127
Changing a route	127
Getting ready	128
How to do it...	128

How it works...	129
Deleting a route	
Getting ready	129
How to do it...	130
How it works...	130
Chapter 8: Load Balancers	132
 Technical requirements	132
 Creating an internal load balancer	132
Getting ready	133
How to do it...	133
How it works...	134
 Creating a public load balancer	134
Getting ready	134
How to do it...	135
How it works...	136
 Creating a backend pool	136
Getting ready	137
How to do it...	137
How it works...	139
See also	139
 Creating health probes	140
Getting ready	140
How to do it...	140
How it works...	143
 Creating load balancer rules	143
Getting ready	143
How to do it...	143
How it works...	146
 Creating inbound Network Address Translation (NAT) rules	147
Getting ready	147
How to do it...	147
How it works...	150
Chapter 9: Traffic Manager	151
 Technical requirements	152
 Creating a new Traffic Manager profile	152
Getting ready	152
How to do it...	152
How it works...	154
 Adding an endpoint	155
Getting ready	155
How to do it...	155
How it works...	158
 Configuring distributed traffic	158

Getting ready	158
How to do it...	158
How it works...	161
Configuring traffic based on priority	161
Getting ready	161
How to do it...	161
How it works...	163
Configuring traffic based on geographical location	163
Getting ready	163
How to do it...	163
How it works...	164
Managing endpoint	165
Getting ready	165
How to do it...	165
How it works...	166
Managing profiles	167
Getting ready	167
How to do it...	167
How it works...	168
Configuring Traffic Manager with load balancers	168
Getting ready	169
How to do it...	169
How it works...	171
Chapter 10: Azure Application Gateway	172
 Technical requirements	173
 Creating a new application gateway	173
Getting ready	173
How to do it...	173
How it works...	178
 Configuring the backend pool	179
Getting ready	179
How to do it...	179
How it works...	180
 Creating HTTP settings	181
Getting ready	181
How to do it...	181
How it works...	184
 Creating a listener	184
Getting ready	184
How to do it...	185
How it works...	186
 Creating a rule	187
Getting ready	187
How to do it...	187

How it works...	188
Creating a probe	188
Getting ready	189
How to do it...	189
How it works...	190
Configuring a WAF	191
Getting ready	191
How to do it...	192
How it works...	194
Customizing WAF rules	194
Getting ready	195
How to do it...	195
How it works...	196
Chapter 11: Azure Firewall	197
 Technical requirements	197
 Creating a new Azure Firewall	197
Getting ready	198
How to do it...	199
How it works...	201
 Configuring a new allow rule	201
Getting ready	201
How to do it...	201
How it works...	201
 Configuring a new deny rule	202
Getting ready	202
How to do it...	202
How it works...	202
 Configuring a route table	202
Getting ready	203
How to do it...	203
How it works...	203
 Enabling diagnostic logs for Azure Firewall	203
Getting ready	203
How to do it...	204
How it works...	205
Other Books You May Enjoy	206
Index	209

Preface

Microsoft provides organizations with an effective way of managing their network with Azure's networking services. No matter the size of your organization, Azure provides highly reliable performance and secure connectivity with its networking services.

The book starts with an introduction to Azure networking, covering subjects such as creating Azure Virtual Networks (VNets), designing address spaces, and subnets. Then you will learn how to create and manage network security groups, application security groups, and IP addresses in Azure. Gradually, we move on to Site-to-Site, Point-to-Site, and VNet-to-VNet connections; DNS and routing; load balancers; and traffic manager. This book delivers practical recipes that cover every aspect and function required to help readers learn basic cloud networking practices, and plan, implement, and secure their infrastructure network with Azure. Readers will not only be able to upscale their current environment, but will also learn how to monitor, diagnose, and ensure secure connectivity. After learning how to deliver a robust environment, readers will also gain meaningful insights from recipes on best practices.

By the end of this book, readers will have hands-on experience of providing cost-effective solutions that benefit organizations.

Who this book is for

This book targets cloud architects, cloud solution providers, or any stakeholders dealing with networking on the Azure cloud. Some prior understanding of Microsoft Azure will be a plus point.

What this book covers

Chapter 1, *Azure Virtual Network*, teaches you about the basics of Azure networking, such as creating Azure VNets, designing address spaces, and subnets. This will lay the foundation for all future recipes in this book.

Chapter 2, *Virtual Machine Networking*, covers Azure VMs and the network interface that is used as an interconnection between Azure VMs and Azure VNet.

Chapter 3, *Network Security Groups*, contains sets of rules that allow or deny specific traffic to specific resources or subnets in Azure. An NSG can be associated with either a subnet (applying security rules to all resources associated with the subnet) or a NIC (applying security rules only to the VM associated with the NIC).

Chapter 4, *Managing IP Addresses*, covers types of IP addresses, private and public. Public addresses can be accessed over the internet. Private addresses are from the Azure VNet address space and are used for private communication on private networks. Addresses can be assigned to a resource or can exist as a separate resource.

Chapter 5, *Local and Virtual Network Gateways*, covers details of local and virtual network gateways. These gateways are virtual private network gateways that are used to connect to on-premises networks. They encrypt all traffic going between Azure VNet and a local network.

Chapter 6, *Creating Hybrid Connections*, allows us to create secure connections Azure VNets. These connections can either be from on the premises or from other Azure VNets. Establishing connections to an Azure VNet enables secure network traffic with other services that are located in different Azure VNets, different subscriptions, or outside Azure (in different clouds or on-premises).

Chapter 7, *DNS and Routing*, allows us to host DNS domains in Azure. When using Azure DNS, we use Microsoft infrastructure for the name resolution, which results in fast and reliable DNS queries. Microsoft Azure DNS infrastructure uses a vast number of servers to provide great reliability and availability of service.

Chapter 8, *Load Balancers*, supports scaling and high availability for applications and services. A load balancer is primarily made of two components—frontend and backend. Requests coming to the frontend of a load balancer are distributed to the backend, where we place multiple instances of a service.

Chapter 9, *Traffic Manager*, teaches you how to create a traffic manager. Also, you will look at the configurations of distributed traffic, traffic based on priority, traffic based on geographical location, and using traffic manager with load balancers.

Chapter 10, *Azure Application Gateway*, is essentially about load balancer for web traffic, but it also allows you better traffic control. Where classic load balancers operate on transport layer, they allow you to route traffic based protocol (TCP or UDP) and IP address, mapping IP address and protocol in the frontend to IP address(es) and protocol in the backend.

Chapter 11, *Azure Firewall*, will teach you how to increase Azure network security using Azure Firewall. It will help you to control inbound and outbound traffic and to be in charge of your network.

To get the most out of this book

This book assumes a basic level of knowledge of cloud computing and Azure. To use this book, all you need is a valid Azure subscription and internet connectivity. A Windows 10 OS with 4 GB of RAM is sufficient for using PowerShell.

Download the example code files

You can download the example code files for this book from your account at www.packt.com. If you purchased this book elsewhere, you can visit www.packt.com/support and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at www.packt.com.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at <https://github.com/PacktPublishing/Azure-Networking-Cookbook>. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at <https://github.com/PacktPublishing/>. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: https://www.packtpub.com/sites/default/files/downloads/9781789800227_ColorImages.pdf.

Conventions used

There are a number of text conventions used throughout this book.

CodeInText: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "An example of how to create a rule to allow traffic over the 443 port."

Any command-line input or output is written as follows:

```
$VirtualNetwork = Get-AzureRmVirtualNetwork -Name 'Packt-Script' -  
ResourceGroupName 'Packt-Networking-Script'
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "In the **Virtual network** blade, go to the **Subnets** section."

Warnings or important notes appear like this.



Tips and tricks appear like this.



Sections

In this book, you will find several headings that appear frequently (*Getting ready*, *How to do it...*, *How it works...*, *There's more...*, and *See also*).

To give clear instructions on how to complete a recipe, use these sections as follows:

Getting ready

This section tells you what to expect in the recipe and describes how to set up any software or any preliminary settings required for the recipe.

How to do it...

This section contains the steps required to follow the recipe.

How it works...

This section usually consists of a detailed explanation of what happened in the previous section.

There's more...

This section consists of additional information about the recipe in order to make you more knowledgeable about the recipe.

See also

This section provides helpful links to other useful information for the recipe.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at customercare@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packt.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packt.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packt.com.

1 Azure Virtual Network

In this very first chapter, you will learn about the basics of Azure networking, including creating Azure Virtual Networks, designing address spaces, and subnets. This will lay the foundation for all future recipes that we'll cover in this book.

We will cover the following recipes in this chapter:

- Creating a virtual network in the portal
- Creating a virtual network with PowerShell
- Adding a subnet in the portal
- Adding a subnet with PowerShell
- Changing the address space size
- Changing the subnet size

Technical requirements

For this chapter, the following is required:

- An Azure subscription
- Azure PowerShell

Code samples can be found in <https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter01>.

Creating a virtual network in the portal

Azure Virtual Network represents your local network in the cloud. It enables other Azure resources to communicate over a secure private network without exposing endpoints over the internet.

Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>.

How to do it...

In order to create a new virtual network using the Azure portal, use the following steps:

1. In the Azure portal, select **Create a resource** and choose **Virtual network** under **Networking** services (or, search for **virtual network** in the search bar).
2. A new blade will open where we need to provide information for the virtual network to include **Name**, define **Address space**, select the **Subscription** option we want to use, select the **Resource group** option for where the virtual network will be deployed, select **Location** (Azure data center) for where the virtual network will be deployed, and define **Name** and **Address range** for the first subnet. We also have the option to select what kind of **DDoS protection** we want to use and if we want to use the **Firewall** option; an example is shown in the following screenshot:

Create virtual network

* Name
Packt-Portal ✓

* Address space ⓘ
10.10.0.0/16 ✓
10.10.0.0 - 10.10.255.255 (65536 addresses)

* Subscription
Microsoft Azure Sponsorship

* Resource group
(New) Packt-Networking-Portal ✓
Create new

* Location
West Europe

Subnet

* Name
FrontEnd ✓

* Address range ⓘ
10.10.0.0/24 ✓
10.10.0.0 - 10.10.0.255 (256 addresses)

DDoS protection ⓘ
 Basic Standard

Service endpoints ⓘ
 Disabled Enabled

Firewall
 Disabled Enabled

Create [Automation options](#)

3. Creating a virtual network usually doesn't take much time and should be completed in under two minutes. Once deployment is finished, you can start using the virtual network.

How it works...

We deploy virtual networks to **Resource group** under **Subscription** in the Azure data center that we choose. **Location** and **Subscription** are important parameters; we will only be able to attach Azure resources to this virtual network if they are in the same subscription and region (as the Azure data center). The **Address space** option defines the number of IP addresses that will be available for our network. It uses the **Classless Inter-Domain Routing (CIDR)** format and the largest range we can choose is /8. In the portal, we need to create an initial subnet and define the subnet address range. The smallest subnet allowed is /29 and the largest is /8 (however, this can't be larger than the virtual network range).

Creating a virtual network with PowerShell

PowerShell is a command-line shell and scripting language based on the .NET Framework. It's often used by system administrators to automate tasks and manage operating systems. Azure PowerShell is a PowerShell module that allows us to automate and manage Azure resources. Azure PowerShell is also very often used to automate deployment tasks and can also be used to deploy a new Azure Virtual Network.

Getting ready

Before we start, we need to connect to the Azure subscription from a PowerShell console. Here's the command to do this:

```
Connect-AzureRmAccount
```

This will open a new window where we need to input the credentials for the Azure subscription.

Afterward, we need to create a resource group where our virtual network will be deployed:

```
New-AzureRmResourceGroup -name 'Packt-Networking-Script' -Location  
'westeurope'
```

The output should be similar to the following screenshot:

```
ResourceGroupName : Packt-Networking-Script
Location         : westeurope
ProvisioningState : Succeeded
Tags             :
ResourceId       : /subscriptions/cb638267-a366-463c-bfe5-7a49311c27a8/resourceGroups/Packt-Networking-Script
```

How to do it...

Deploying Azure Virtual Network is done in a single script. We need to define parameters for the resource group, location, name, and address range. Here is an example script:

```
New-AzureRmVirtualNetwork -ResourceGroupName 'Packt-Networking-Script' -  
Location 'westeurope' -Name 'Packt-Script' -AddressPrefix 10.11.0.0/16
```

You should receive the following output:

ResourceGroupName	Name	Location	ProvisioningState	EnableDdosProtection	EnableVmProtection
Packt-Networking-Script	Packt-Script	westeurope	Succeeded	False	False

How it works...

The difference between deploying a virtual network from the portal and using PowerShell is that no subnet needs to be defined in PowerShell. The subnet is deployed in a separate command that can be executed either when you are deploying a virtual network or later on. We are going to see this command in the *Adding subnets with PowerShell* recipe.

Adding a subnet in the portal

Beside adding subnets while creating a virtual network, we can add additional subnets to our network at any time.

Getting ready

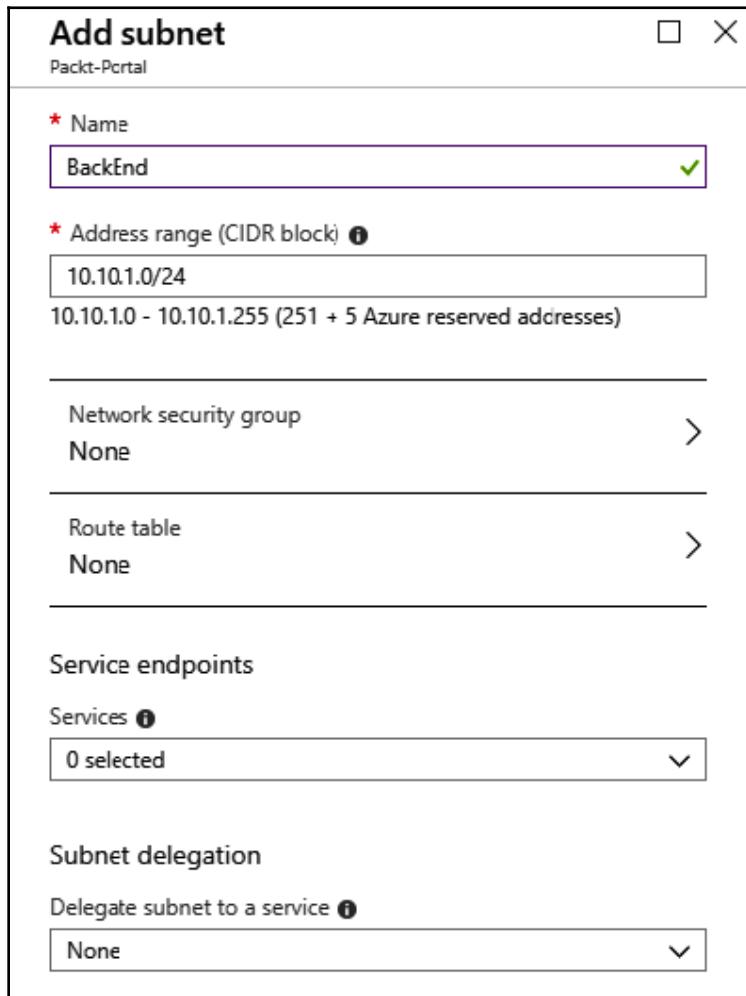
Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>. Here, locate the previously created virtual network.

How to do it...

In order to add a subnet to a virtual network using the Azure portal, we must use the following steps:

1. In the virtual network blade, go to the **Subnets** section.
2. Select the **Add subnet** option.

3. A new blade will open. We need to provide information for the subnet, including **Name** and **Address range** in the CIDR format. **Address range** must be in the range limit of the virtual network address range and cannot overlap with the address range of other subnets in the virtual network. Optionally, we can add information for **Network security group**, **Route tables**, **Service endpoints**, and **Subnet delegation**. These options will be covered in later recipes:



4. We can also add a gateway subnet in the same blade. To add a gateway subnet, select the **Gateway subnet** option.

For a gateway subnet, the only parameter we need to define is **Address range**. The same rules apply as for adding a regular subnet. This time, we don't have to provide a name as it's already defined. You can add only one gateway subnet per virtual network. Service endpoints are not allowed in the gateway subnet:

Add subnet

Packt-Portal

* Name
GatewaySubnet

* Address range (CIDR block) ⓘ
10.10.2.0/24
10.10.2.0 - 10.10.2.255 (251 + 5 Azure reserved addresses)

Route table >
None

Service endpoints

Services ⓘ
0 selected

Subnet delegation

Delegate subnet to a service ⓘ
None

OK

5. After the subnets are added, we can see the newly created subnets in the subnet blade under the virtual network:

NAME	ADDRESS RANGE	AVAILABLE ADDRESSES
FrontEnd	10.10.0.0/24	251
BackEnd	10.10.1.0/24	251
GatewaySubnet	10.10.2.0/24	251

How it works...

A single virtual network can have a multiple number of subnets defined. Subnets can't overlap and must be in the range of the virtual network address range. For each subnet, four IP addresses are used for management and can't be used. Depending on the network settings, we can define the communication rules between subnets in the virtual network. A gateway subnet is used for VPN connections, and this will be covered in later chapters.

Adding a subnet with PowerShell

When creating Azure Virtual Network with PowerShell, a subnet is not created in the same step and requires an additional command to be executed separately.

Getting ready

Before creating a subnet, we need to collect information about the virtual network that the new subnet will be associated with. The parameters that need to be provided are the name of the virtual network and the resource group that the virtual network is located in:

```
$VirtualNetwork = Get-AzureRmVirtualNetwork -Name 'Packt-Script' -  
ResourceGroupName 'Packt-Networking-Script'
```

How to do it...

1. To add a subnet to the virtual network, we need to execute a command and provide the name and address prefix. The address prefix is again in CIDR format:

```
Add-AzureRmVirtualNetworkSubnetConfig -Name FrontEnd -AddressPrefix  
10.11.0.0/24 -VirtualNetwork $VirtualNetwork
```

2. We need to confirm these changes by executing the following:

```
$VirtualNetwork | Set-AzureRmVirtualNetwork
```

3. We can add an additional subnet by running all commands in a single step, as follows:

```
$VirtualNetwork = Get-AzureRmVirtualNetwork -Name 'Packt-Script' -  
ResourceGroupName 'Packt-Networking-Script'  
Add-AzureRmVirtualNetworkSubnetConfig -Name BackEnd -AddressPrefix  
10.11.1.0/24 -VirtualNetwork $VirtualNetwork  
$VirtualNetwork | Set-AzureRmVirtualNetwork
```

How it works...

The subnet is created and added to the virtual network, but we need to confirm the changes before they can become effective. All the rules when creating or adding subnet size using the Azure portal apply here as well; the subnet must be within the virtual network's address space and cannot overlap with other subnets in the virtual network. The smallest subnet allowed is /29, and the largest is /8.

There's more...

We can create and add multiple subnets in a single script, as follows:

```
$VirtualNetwork = Get-AzureRmVirtualNetwork -Name 'Packt-Script' -  
ResourceGroupName 'Packt-Networking-Script'  
$FrontEnd = Add-AzureRmVirtualNetworkSubnetConfig -Name FrontEnd -  
AddressPrefix 10.11.0.0/24 -VirtualNetwork $VirtualNetwork  
$BackEnd = Add-AzureRmVirtualNetworkSubnetConfig -Name BackEnd -  
AddressPrefix 10.11.1.0/24 -VirtualNetwork $VirtualNetwork  
$VirtualNetwork | Set-AzureRmVirtualNetwork
```

Changing the address space size

After the initial address space is defined during the creation of a virtual network, we can still change the address space size as needed. We can either increase or decrease the size of the address space, or change the address space completely by using a new address range.

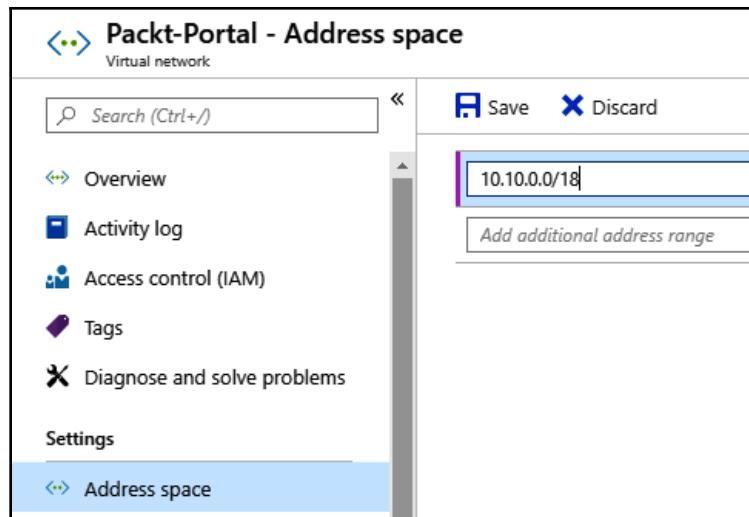
Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>.

How to do it...

In order to change the address space size for a virtual network using the Azure portal, we must observe the following steps:

1. In a virtual network blade, locate **Address space** under **Settings**.
2. In the available address space, click on **Address space** and change the value. An example is shown in the following screenshot:



3. After you have entered a new value for **Address space**, click **Save** to apply the changes.

How it works...

Although you can change the address space at any time, there are some rules that determine what you can or cannot do. Address space can't be decreased if you have subnets defined in the address space that wouldn't be covered by a new address space. For example, if the address space was in the range of 10.0.0.0/16, it would cover addresses from 10.0.0.1 to 10.0.255.254. If one of the subnets was defined as 10.0.255.0/24, we wouldn't be able to change the virtual network to 10.0.0.0/17, as this will leave the subnet outside the new space.

Address space can't be changed to the new address space if you have subnets defined. In order to completely change the address space, you need to remove all subnets first. For example, if we have the address space defined as 10.0.0.0/16, we wouldn't be able to change it to 10.1.0.0/16, since having any subnets in the old space would leave them in an undefined address range.

Changing the subnet size

Similar to the virtual network address space, we can change the size of a subnet at any time.

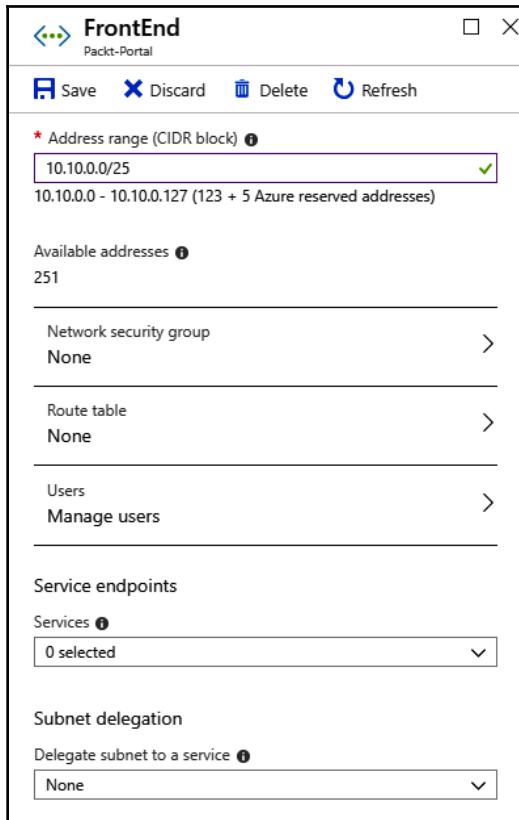
Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>.

How to do it...

In order to change subnet size using the Azure portal, we must use the following steps:

1. In a virtual network blade, select the **Subnets** option.
2. Select the subnet you want to change.
3. In the **Subnets** option, enter a new value for the subnet size under **Address range**. An example of how to do this is shown in the following screenshot:



4. After entering a new value, click on **Save**.
5. In the **Subnets** list, you can see the changes applied and the address space has changed, as shown in the following screenshot:

The screenshot shows the Azure Packt-Portal interface for managing subnets. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Address space, Connected devices, and Subnets. The Subnets link is currently selected and highlighted in blue. The main area is titled "Subnets" and contains a table with three rows. The columns are NAME, ADDRESS RANGE, and AVAILABLE ADDRESSES. The data is as follows:

NAME	ADDRESS RANGE	AVAILABLE ADDRESSES
BackEnd	10.10.1.0/24	251
GatewaySubnet	10.10.2.0/24	251
FrontEnd	10.10.0.0/25	123

How it works...

When changing subnet size, there are some rules that must be followed. You can't change the address space if it's not within the virtual network address space range and the subnet range can't overlap with other subnets in a virtual network. If devices are assigned to this subnet, you can't change the subnet to exclude the addresses that these devices are already assigned to.

2

Virtual Machine Networking

In this chapter, we'll cover Azure **Virtual Machines (VMs)** and the network interface that is used as an interconnection between Azure VMs and Azure Virtual Network.

We will cover the following recipes in this chapter:

- Creating Azure VMs
- Viewing VM network settings
- Creating a new network interface
- Attaching a network interface to a VM
- Detaching a network interface from a VM

Technical requirements

For this chapter, the following is required:

- An Azure subscription

Creating Azure VMs

Azure VMs depend on virtual networking, and during the creation process, we need to define the network settings.

Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>.

How to do it...

In order to create a new VM using the Azure portal, we must use the following steps:

1. In the Azure portal, select **Create a resource** and choose **Windows Server 2016 VM** (or search for any VM image under the **Compute** section).
2. In the **Create a virtual machine** blade, we need to provide information for various options; not all of these are related to networking. First, we need to provide information on our Azure **Subscription** and **Resource group** (create a new resource group or provide an existing one).
3. In **INSTANCE DETAILS**, we need to provide information on **Virtual machine name**, **Region**, **Availability option**, or change with the **Image** dropdown. Example settings are shown in the following screenshot:

The screenshot shows the 'Create a virtual machine' blade. It has two main sections: 'PROJECT DETAILS' and 'INSTANCE DETAILS'. In 'PROJECT DETAILS', there is a 'Subscription' dropdown set to 'Microsoft Azure Sponsorship' and a 'Resource group' dropdown set to 'Packt-Networking-Portal-VMS'. In 'INSTANCE DETAILS', there is a 'Virtual machine name' dropdown set to 'Packt2', a 'Region' dropdown set to 'West Europe', an 'Availability options' dropdown set to 'No infrastructure redundancy required', and an 'Image' dropdown set to 'Windows Server 2016 Datacenter'.

4. Next, we need to provide information on our VMs **Size**, **Username**, and **Password**. Note that for **Username**, you can't use names such as admin, administrator, sysadmin, or root. **Password** must be at least 12 characters long and satisfy three out of four of the famous rules (that is, to combine big letters, small letters, special characters, and numbers). An example of this is shown in the following screenshot:

* Size ⓘ

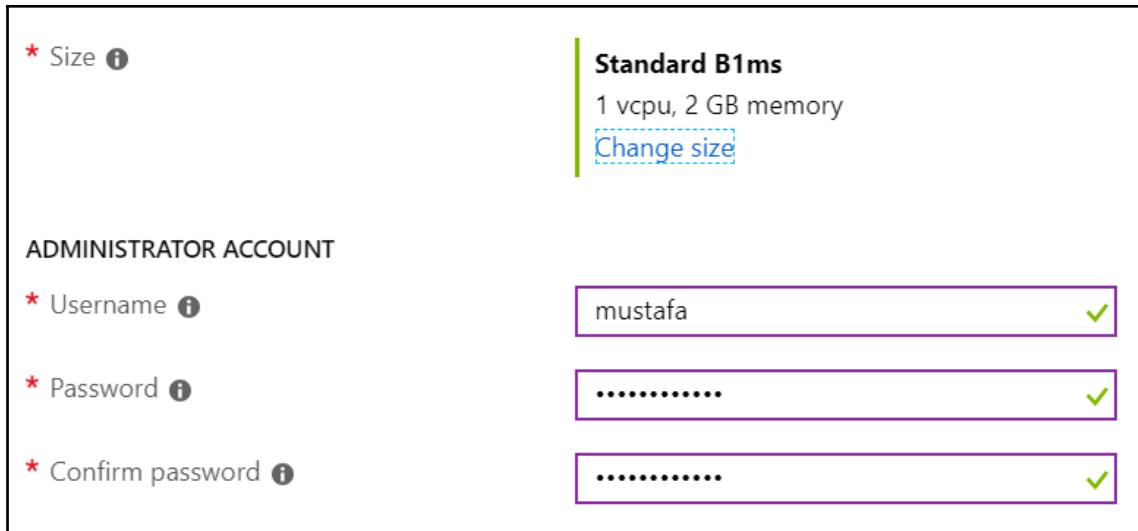
Standard B1ms
1 vcpu, 2 GB memory
[Change size](#)

ADMINISTRATOR ACCOUNT

* Username ⓘ mustafa ✓

* Password ⓘ ✓

* Confirm password ⓘ ✓



5. Next, we get to an option that concerns networking. We need to define whether we are going to allow any type of connection over a public IP address. We can select whether we want to deny all access, or allow a specific port. In the following example, I'm choosing **RDP (3389)**:

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

* Public inbound ports ⓘ None Allow selected ports

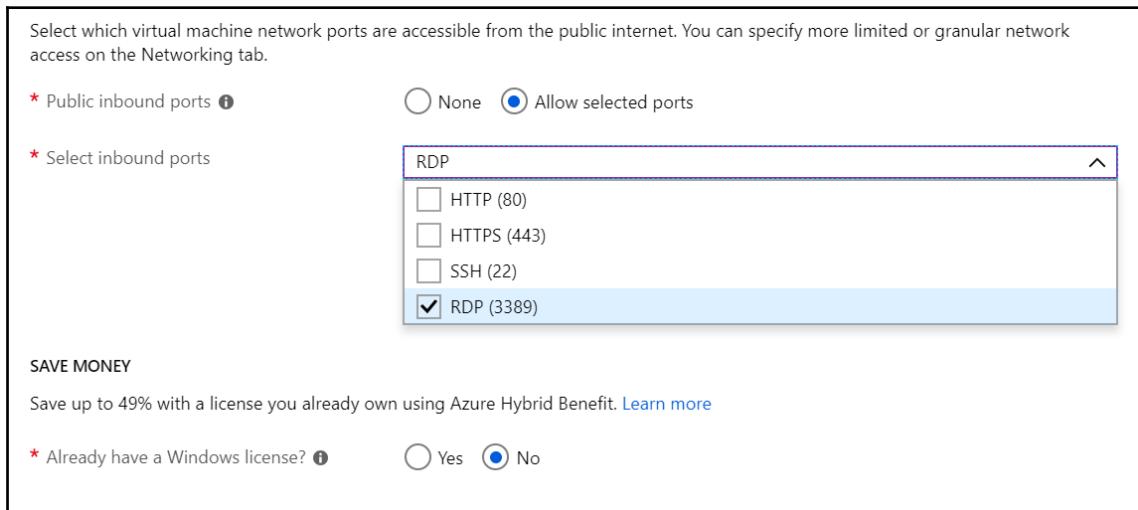
* Select inbound ports

RDP	^
<input type="checkbox"/> HTTP (80)	
<input type="checkbox"/> HTTPS (443)	
<input type="checkbox"/> SSH (22)	
<input checked="" type="checkbox"/> RDP (3389)	

SAVE MONEY

Save up to 49% with a license you already own using Azure Hybrid Benefit. [Learn more](#)

* Already have a Windows license? ⓘ Yes No



6. In next section, we need to define disks. We can choose between **Premium SSD**, **Standard SSD**, and **Standard HDD**. An OS disk is required and must be defined. We can attach additional data disks as needed. Disks can be added at a later time as well. We can choose whether we are going to **Use managed disks** or not. My recommendation would be to go with managed disks as they make maintenance much easier. An example of disk settings with only the OS disk is shown in the following screenshot:

The screenshot shows the 'Create a virtual machine' interface. Under 'DISK OPTIONS', the 'OS disk type' is set to 'Premium SSD'. In the 'DATA DISKS' section, there is a table with columns: LUN, NAME, SIZE (GiB), DISK TYPE, and HOST CACHING. Below the table are two buttons: 'Create and attach a new disk' and 'Attach an existing disk'. A 'Use managed disks' toggle switch is set to 'Yes'. An 'ADVANCED' section is collapsed.

7. After defining disks, we get to the networking settings. Here, we need to define the **Virtual network** and **Subnet** options that the VM will use. These two options are mandatory. You can choose to assign the **Public IP** address to the VM (you can choose to disable the **Public IP** address, create a new one, or assign to an existing IP address). The last part of the network settings relate to **Network security group**, where we need to choose if we are going to use a **Basic** or **Advanced** NSG, and another option to define whether we will allow public ports. A VM network settings example is shown in the following screenshot:

* Virtual network

* Subnet

Public IP

Network security group Basic Advanced

* Public inbound ports None Allow selected ports

* Select inbound ports

⚠ These ports will be exposed to the internet. Use the Advanced controls to limit inbound traffic to known IP addresses. You can also update inbound traffic rules later.

8. After the networking section, we need to set up **MONITORING**. **Boot diagnostics** are enabled by default and you can enable additional features as needed. The default settings for **MONITORING** are shown in the following screenshot:

MONITORING

Boot diagnostics On Off

OS guest diagnostics On Off

* Diagnostics storage account

IDENTITY

System assigned managed identity On Off

AUTO-SHUTDOWN

Enable auto-shutdown On Off

BACKUP

Enable backup On Off

9. In **EXTENSIONS**, we can set up post-deployment configuration steps by adding software installations, configuration scripts, and more. The **EXTENSIONS** screen is shown in the following screenshot:

Add additional configuration, agents, scripts or applications via virtual machine extensions or cloud-init.

EXTENSIONS

Extensions provide post-deployment configuration and automation.

Extensions i Select an extension to install

CLOUD INIT

Cloud init is a widely used approach to customize a Linux VM as it boots for the first time. You can use cloud-init to install packages and write files or to configure users and security. [Learn more](#)

i The selected image does not support cloud init.

10. The last setting that we can edit are tags. Tags apply additional metadata to Azure resources to logically organize them into a taxonomy. The **Tags** screen is shown in the following screenshot:

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. [Learn more](#)

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

<input type="checkbox"/>	KEY	VALUE	RESOURCE TYPE
<input type="checkbox"/>			All resources to be created ▼

11. After all settings are defined, we get to validation screen where all our settings are checked for the last time. After validation is passed, we confirm the creation of a VM by pressing the **Create** button, as shown in the following screenshot:

The screenshot shows the 'Create a virtual machine' interface in the Azure portal. At the top, a green bar indicates 'Validation passed'. Below it, a navigation bar includes 'Basics', 'Disks', 'Networking', 'Management', 'Guest config', 'Tags', and 'Review + create' (which is underlined). Under 'PRODUCT DETAILS', it shows 'Standard B1ms by Microsoft' and a price of '0.0234 EUR/hr'. A link 'Pricing for other VM sizes' is also present. In the 'TERMS' section, there's a detailed legal notice about agreeing to terms and privacy statements. At the bottom, there are buttons for 'Create' (in blue), 'Previous', 'Next', and 'Download a template for automation'.

How it works...

When a VM is created, a **network interface (NIC)** is created in the process. An NIC is used as a sort of interconnection between the VM and virtual network. An NIC is assigned a private IP address by the network. As an NIC is associated both with the VM and virtual network, the IP address is used by the VM. Using this IP address, the VM can communicate over a private network with other VMs (or other Azure resources) on same network. Additionally, NICs and VMs can be assigned public IP address as well. Public address can be used to communicate with the VM over the internet, either to access services or to manage the VM.

See also

If you are interested to find out more about Azure VMs, you can read my book, *Hands-On Cloud Administration in Azure*, by Packt Publishing, where VMs are covered in more detail.

Viewing VM network settings

After an Azure VM is created, we can review the network settings in the VM blade.

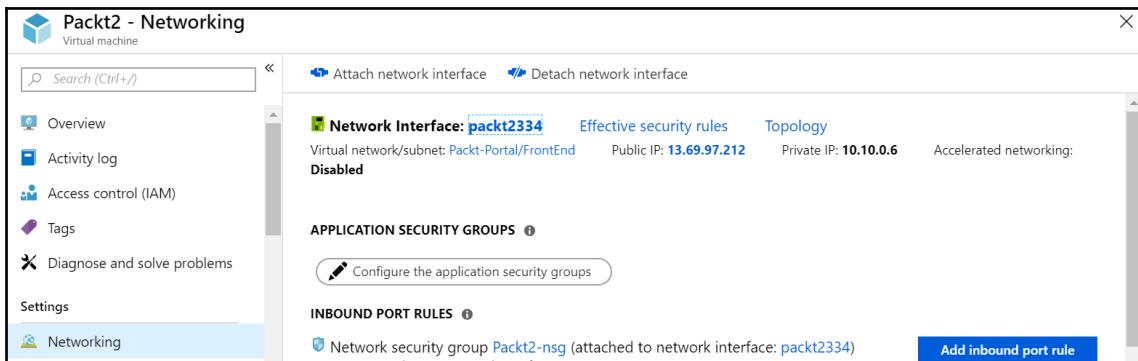
Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>. Here, locate the previously created VM.

How to do it...

In order to review the VM network settings, we must do the following steps:

1. In the VM blade, locate the **Networking** settings. Here, you can see **Network interface**, **APPLICATION SECURITY GROUPS**, and **Network security group** associated with the VM. An example of this is shown in the following screenshot:



2. If we select any of the associated network elements, we can discover more details. For example, if we select the **Network interface** option associated with the VM, we can see other networking information such as **Private IP address**, **Virtual network/subnet**, **Public IP address**, **Network security group**, **IP configurations**, **DNS servers**, and more. The network interface view is shown in the following screenshot:

Resource group (change)	Private IP address 10.10.0.6
Packt-Networking-Portal-VMS	Virtual network/subnet Packt-Portal/FrontEnd
Location West Europe	Public IP address 13.69.97.212 (Packt2-ip)
Subscription (change) Microsoft Azure Sponsorship	Network security group Packt2-nsg
Subscription ID cb638267-a366-463c-bfe5-7a49311c27a8	Attached to Packt2
Tags (change) Click here to add tags	

How it works...

Networking information is displayed in several places, including the VM's network settings. Additionally, each Azure resource has a separate blade and exists as an individual resource, so we can view these settings in multiple places. However, the most complete picture about VM network settings we can find is in the VM blade and NIC blade.

Creating a new network interface

A network interface is usually created during the VM creation process, but each VM can have multiple network interfaces. Based on this, we can create an NIC as an individual resource and attach it or detach it as needed.

Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>.

How to do it...

In order to create a new NIC using the Azure portal, we must use the following steps:

1. In the Azure portal, select **Create a resource** and choose **Network interface** under **Networking** services (or search for `network interface` in the search bar).
2. In the creation blade, we need to provide information relating to the **Name**, **Virtual network**, and **Subnet** that the NIC will be associated with. Other information to be provided includes the IP address assignment type (**Dynamic** or **Static**), whether we want the NIC to be associated with a **Network security group** type, and whether we want to use **IPv6**. All Azure resources require information on **Subscription**, **Resource group**, and **Location**, and NIC is no exception. The information needed to create a new NIC is shown in the following screenshot:

Create network interface □ X

* Name ✓

* Virtual network i

* Subnet i

Private IP address assignment Dynamic Static

Network security group i >

Private IP address (IPv6)

* Subscription

* Resource group i Create new

* Location

Create Automation options

How it works...

A network interface can't exist without network association, and this must be assigned to a virtual network and subnet. This is defined during the creation process and cannot be changed later. On the other hand, association with a VM can be changed and the NIC can be attached or detached from a VM at any time.

Attaching a network interface to a VM

Each VM can have multiple network interfaces. Because of this, we can add a new network interface at any time.

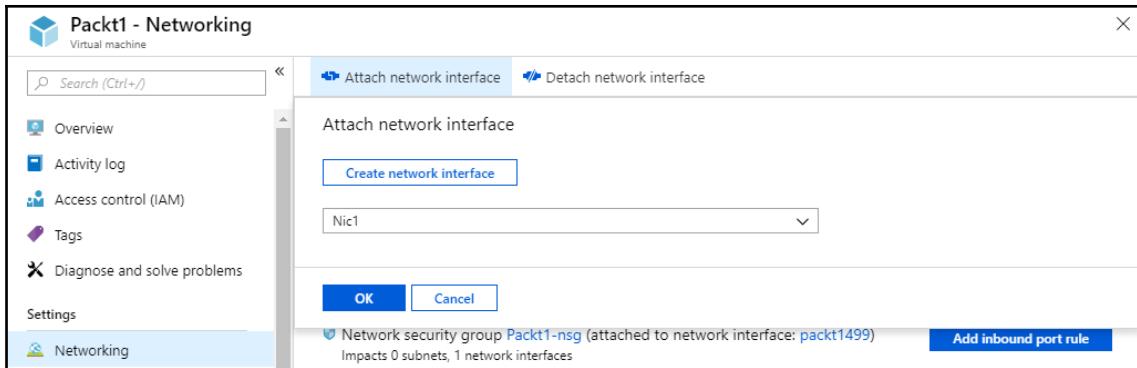
Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>. Here, locate the previously created VM.

How to do it...

To attach a network interface to a VM, we must do the following:

1. In the VM blade, make sure the VM is stopped (that is, deallocated).
2. Locate the **Networking** settings in the VM blade.
3. At the top of the **Networking** settings screen in the VM blade, select the option to **Attach network interface**.
4. A new option will appear, allowing you to create a new NIC or select an already-existing NIC that is not associated with the VM.
5. Click **OK** and, in a few moments, the process will finish and the NIC will be associated with the VM. An example of this is shown in the following screenshot:



How it works...

Each VM can have multiple network interfaces. The number of NICs associated with a VM depends on the type and size of the VM. To attach an NIC to a VM, the VM needs to be stopped (that is, deallocated); you can't add an additional NIC to a running VM.

Detaching a network interface from a VM

Just as with attaching a network interface, we can detach network interface at any time and attach it to another VM.

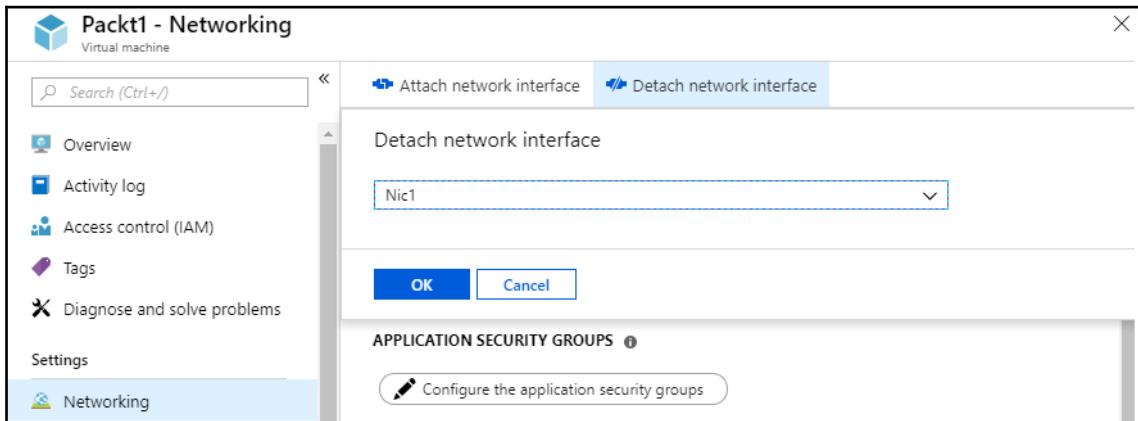
Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>. Here, locate the previously created VM.

How to do it...

To detach a network interface from a VM, we must do the following:

1. In the VM blade, make sure the VM is stopped (that is, deallocated).
2. Locate the **Networking** settings in the VM blade.
3. In the top of **Networking** settings screen in the VM blade, select the option to **Detach network interface**.
4. Select the NIC you want to detach from the VM.
5. Click **OK** and, in a few moments, the process will finish and the NIC will be removed from the VM. An example of this is shown in the following screenshot:



How it works...

To detach a network interface, the VM associated with the NIC must be stopped (that is, deallocated). At least one NIC must be associated with the VM; so, you can't remove the last NIC from a VM. All network associations stay with the NIC; they are assigned to the NIC, not to the VM.

3

Network Security Groups

Network Security Groups (NSGs) contain sets of rules that allow or deny specific traffic to specific resources or subnets in Azure. An NSG can be associated with either a subnet (applying security rules to all resources associated with the subnet) or a **network interface (NIC)** (applying security rules only to the VM associated with the NIC).

We will cover the following recipes in this chapter:

- Creating a new NSG in a portal
- Creating a new NSG with PowerShell
- Creating a new allow rule in NSG
- Creating a new deny rule in NSG
- Creating a new NSG rule with PowerShell
- Assigning an NSG to a subnet
- Assigning an NSG to network interface
- Assigning an NSG with PowerShell
- Creating an Application Security Group (ASG)
- Associating an ASG with a virtual machine (VM)
- Creating rules with an NSG and an ASG

Technical requirements

For this chapter, the following is required:

- Azure subscription
- Azure PowerShell

Code samples can be found at <https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter03>.

Creating a new NSG in a portal

As a first step to controlling network traffic better, we are going to create a new NSG. NSGs are built-in tools for network control and allow us to control incoming and outgoing traffic on a network interface or at the subnet level.

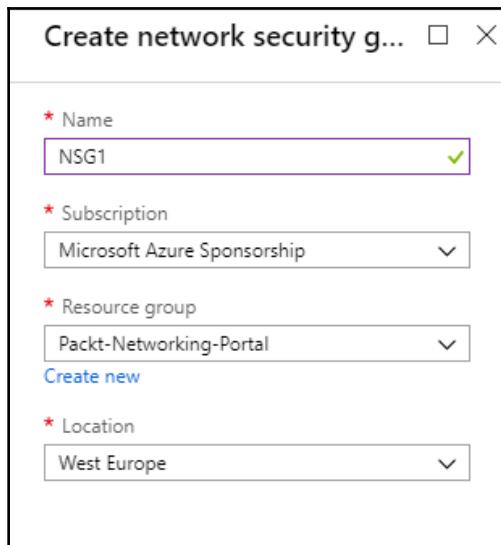
Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

How to do it...

To create a new NSG using the Azure portal, we must follow these steps:

1. In the Azure portal, select **Create a resource** and choose **Network security group** under the **Networking** services (or search for network security group in the search bar).
2. The parameters we need to define for the deployment are **Name**, **Subscription**, **Resource group**, and **Location**. An example of the required parameters is shown in the following screenshot:



3. After the deployment has been validated and started (it takes a few moments to complete), the NSG is ready for use.

How it works...

The NSG deployment can be initiated during a VM deployment. This will associate the NSG to the NIC associated with the VM. In this case, the NSG is already associated with the resource, and rules defined in the NSG will apply only to the associated VM.

If the NSG is deployed separately, as in this recipe, it is not associated and the rules that are created are not applied until the association has been created with NIC or the subnet. When it is associated with a subnet, the NSG rules will apply to all resources on the subnet.

Creating a new NSG with PowerShell

Alternatively, we can create an NSG using Azure PowerShell. The advantage of this approach is that we can add NSG rules in a single script, creating custom rules right after the NSG is created. This allows us to automate the deployment process and have our own "default" rules right after the NSG has been created.

Getting ready

Open the PowerShell console and make sure you are connected to your Azure subscription.

How to do it...

To deploy a new NSG, execute the following command:

```
New-AzureRmNetworkSecurityGroup -Name "nsg1" -ResourceGroupName "Packt-Networking-Script" -Location "westeurope"
```

How it works...

The final outcome will be the same as creating a new NSG using the Azure portal: a new NSG will have been created with default rules. An advantage of using PowerShell is that we can add additional rules and automate the process. We will see an example of this in the *Creating NSG rule with PowerShell* recipe later in this chapter.

Creating a new allow rule in NSG

When a new NSG is created, only the default rules are present. Default rules allow all outbound traffic and block all inbound traffic. To change these, additional rules need to be created. First, we are going to show you how to create a new rule to allow inbound traffic.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>. Locate the previously created NSG.

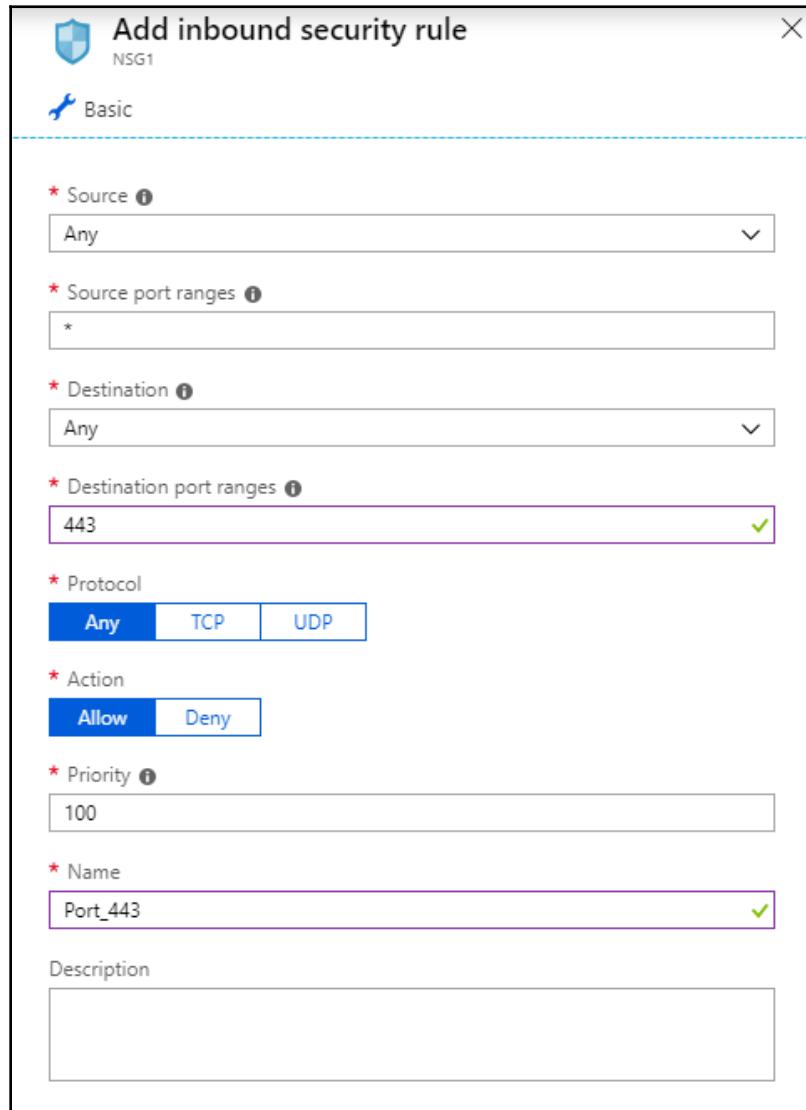
How to do it...

To create a new NSG allow rule using the Azure portal, we must follow these steps:

1. In the NSG blade, locate the **Inbound security rules** option under **Settings**.
2. Click on the **Add** button at the top of the page and wait for the new blade to open:

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION	...
65000	AllowVnetInbound	Any	Any	VirtualNetwork	VirtualNetwork	Allow	...
65001	AllowAzureLoadBalancerInbound	Any	Any	AzureLoadBalancer	Any	Allow	...
65500	DenyAllInbound	Any	Any	Any	Any	Deny	...

3. In the new blade, we need to provide information for **Source** (location and port), **Destination** (location and port), **Protocol**, **Action**, **Priority**, **Name**, and **Description**. If you want to allow traffic, make sure you select **Allow** for **Action**. An example of how to create a rule to allow traffic over the 443 port (allowing traffic to the web server) is shown in the following screenshot:



How it works...

By default, all traffic coming from Azure Load Balancer or Azure Virtual Network is allowed. All traffic coming over the internet is denied. To change this, we need to create additional rules. Make sure you set the right priority when creating rules. Rules with highest priority (lower number) are processed first, so if you have two rules where one is denying traffic and one is allowing it, one with higher priority will take over while one with lower priority will be ignored.

Creating a new deny rule in NSG

When a new NSG is created, only default rules are present. Default rules allow all outbound traffic and block all inbound traffic. To change this, additional rules need to be created. Now, we are going to show you how to create a new outbound rule to deny traffic.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>. Locate the previously created NSG.

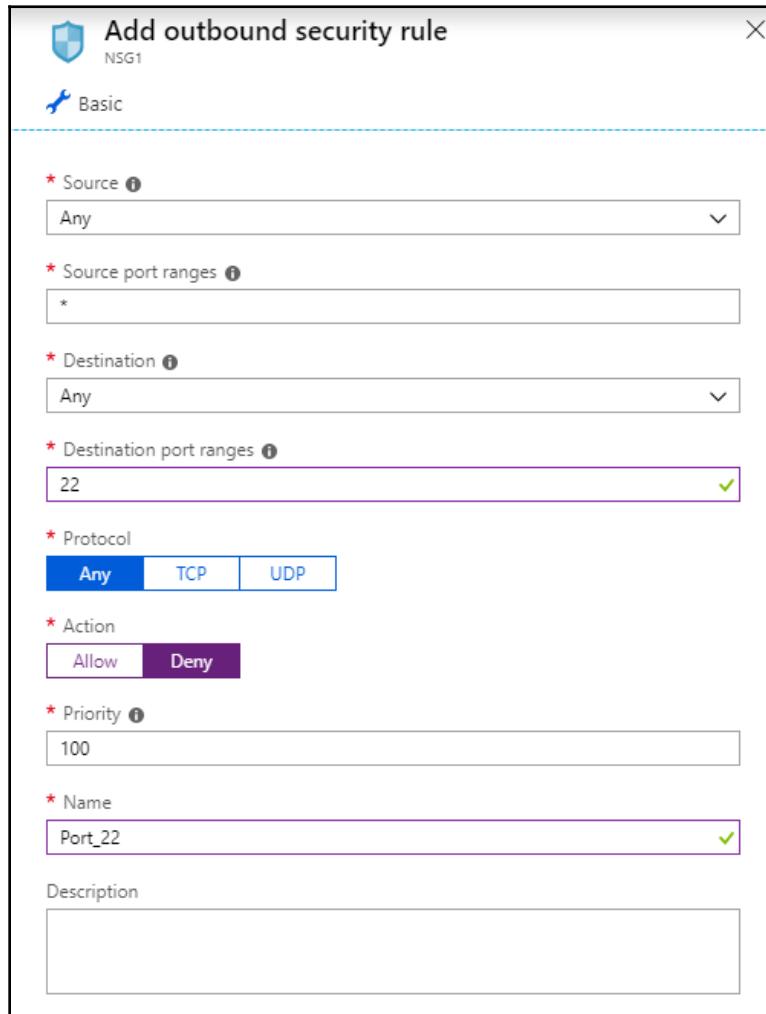
How to do it...

To create a new NSG deny rule using the Azure portal, we must follow these steps:

1. In the NSG blade, locate the **Outbound security rules** option under **Settings**.
2. Click on the **Add** button at the top of the page and wait for the new blade to open:

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
65000	AllowVnetOutBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowInternetOutBound	Any	Any	Any	Internet	Allow
65500	DenyAllOutBound	Any	Any	Any	Any	Deny

3. In the new blade, we need to provide information for **Source** (location and port), **Destination** (location and port), **Protocol**, **Action**, **Priority**, **Name**, and **Description**. If you want to deny traffic, make sure you select **Deny** for **Action**. An example of how to create a rule to deny traffic over the 22 port is shown in the following screenshot:



How it works...

All outbound traffic is allowed by default, regardless of where it is going. If we want to explicitly deny traffic on a specific port, we need to create a rule to do so. Make sure you set the priority right when creating rules. Rules with the highest priority (lower number) are processed first, so if you have two rules where one is denying traffic and one is allowing it, the rule with higher priority will apply.

Creating a new NSG rule with PowerShell

Alternatively, we can create an NSG rule using Azure PowerShell. This command can be executed right after the NSG has been created, allowing us to create and configure NSG in a single script. This way, we can standardize deployment and have rules applied each time an NSG is created.

Getting ready

Open the PowerShell console and make sure you are connected to your Azure subscription.

How to do it...

To create a new NSG rule, execute the following command:

```
$nsg = Get-AzureRmNetworkSecurityGroup -Name 'nsg1' -ResourceGroupName  
'Packt-Networking-Script'  
$nsg | Add-AzureRmNetworkSecurityRuleConfig -Name 'Allow_HTTPS' -  
Description 'Allow_HTTPS' -Access Allow -Protocol Tcp -Direction Inbound -  
Priority 100 -SourceAddressPrefix Internet -SourcePortRange * -  
DestinationAddressPrefix * -DestinationPortRange 443 | Set-  
AzureRmNetworkSecurityGroup
```

How it works...

Using a script, creating an NSG rule is just a matter of parameters. The access parameter, which can be either allow or deny, will determine if we want to allow traffic or deny it. The direction parameter, which can be inbound or outbound, determines if the rule is for inbound or outbound traffic. All other parameters are the same, no matter what kind of rule we want to create. Again, priority plays a very important role and so we must make sure it's chosen correctly.

There's more...

As mentioned in the *Creating a new NSG with PowerShell* recipe, we can create the NSG and the rules that are needed in a single script. The following script is an example of this:

```
$nsg = New-AzureRmNetworkSecurityGroup -Name 'nsg1' -ResourceGroupName  
'Packt-Networking-Script' -Location "westeurope"  
$nsg | Add-AzureRmNetworkSecurityRuleConfig -Name 'Allow_HTTPS' -  
Description 'Allow HTTPS' -Access Allow -Protocol Tcp -Direction Inbound -  
Priority 100 -SourceAddressPrefix Internet -SourcePortRange * -  
DestinationAddressPrefix * -DestinationPortRange 443 | Set-  
AzureRmNetworkSecurityGroup
```

Assigning an NSG to a subnet

The NSG and rules must be assigned to a resource to make any impact. First, we are going to show you how to associate an NSG with a subnet.

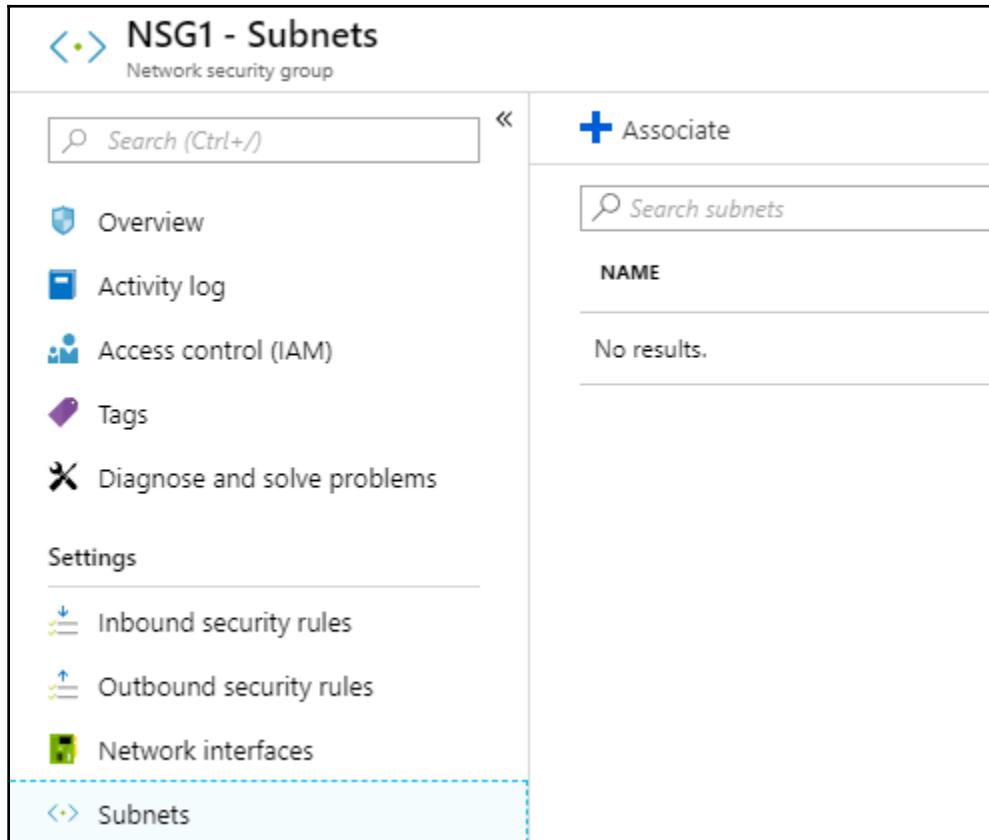
Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>. Locate the previously created NSG.

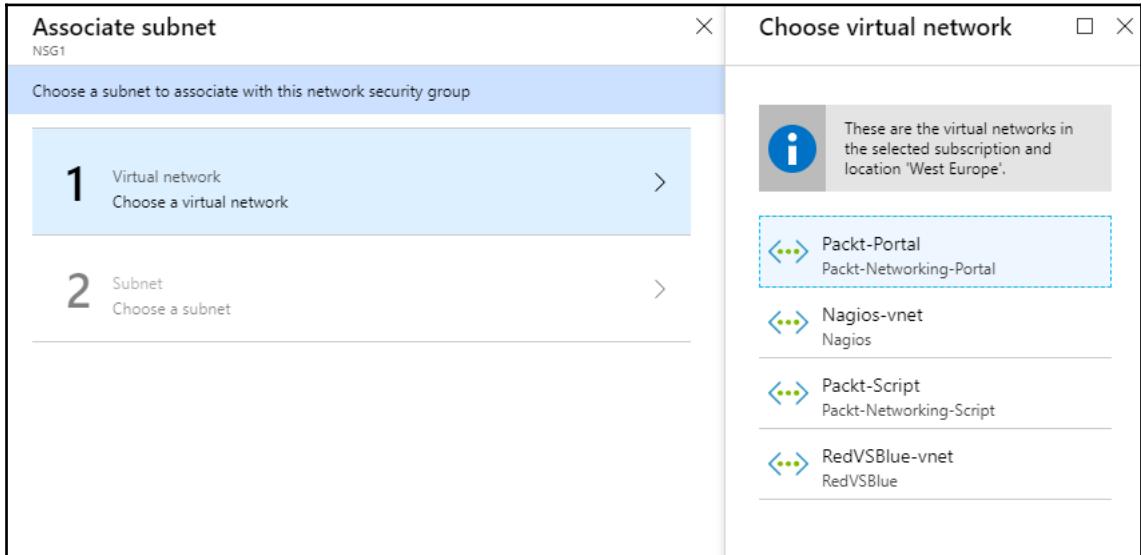
How to do it...

To assign an NSG to a subnet, we must follow these steps:

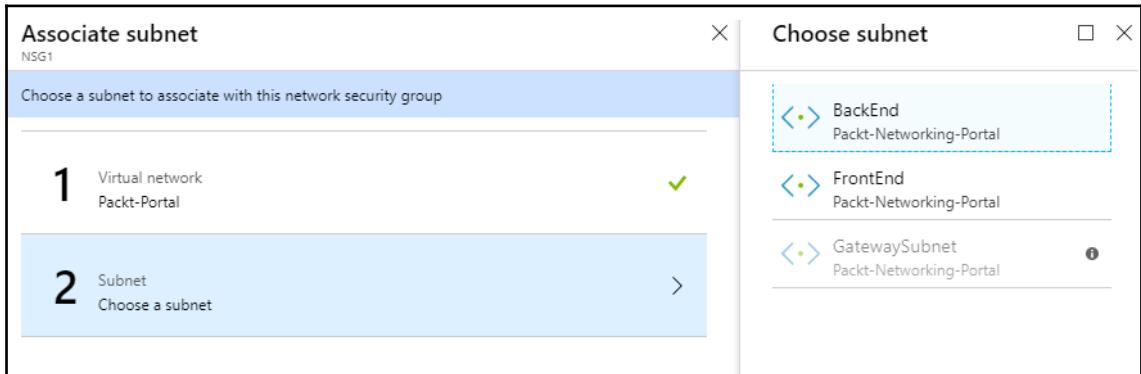
1. In the NSG blade, locate the **Subnets** option under **Settings**
2. Click on the **Associate** button at the top of the page and wait for the new blade to open:



3. In the new blade, first select **Virtual network**, which contains the subnet you want to associate the NSG with:



4. After selecting the virtual network, select the **Subnet** you want to associate it with:



- After submitting the change, the subnet will appear in a list of associated subnets:

NAME	ADDRESS RANGE	VIRTUAL NETWORK
BackEnd	10.10.1.0/24	Packt-Portal

How it works...

When an NSG is associated with a subnet, the rules in the NSG will apply to all of the resources in the subnet. Note that the subnet can be associated with more than one NSG and the rules from all the NSGs will apply in that case. Priority is the most important factor when looking at single NSGs, but when the rules from more NSGs are observed, the deny rule will prevail. So, if we have two subnets, one with allow on the 443 port and another one with the deny rule on the same port, traffic on this port will be denied.

Assigning an NSG to a network interface

An NSG and its rules must be assigned to a resource to make any impact. Now, we are going to show you how to associate an NSG with a network interface.

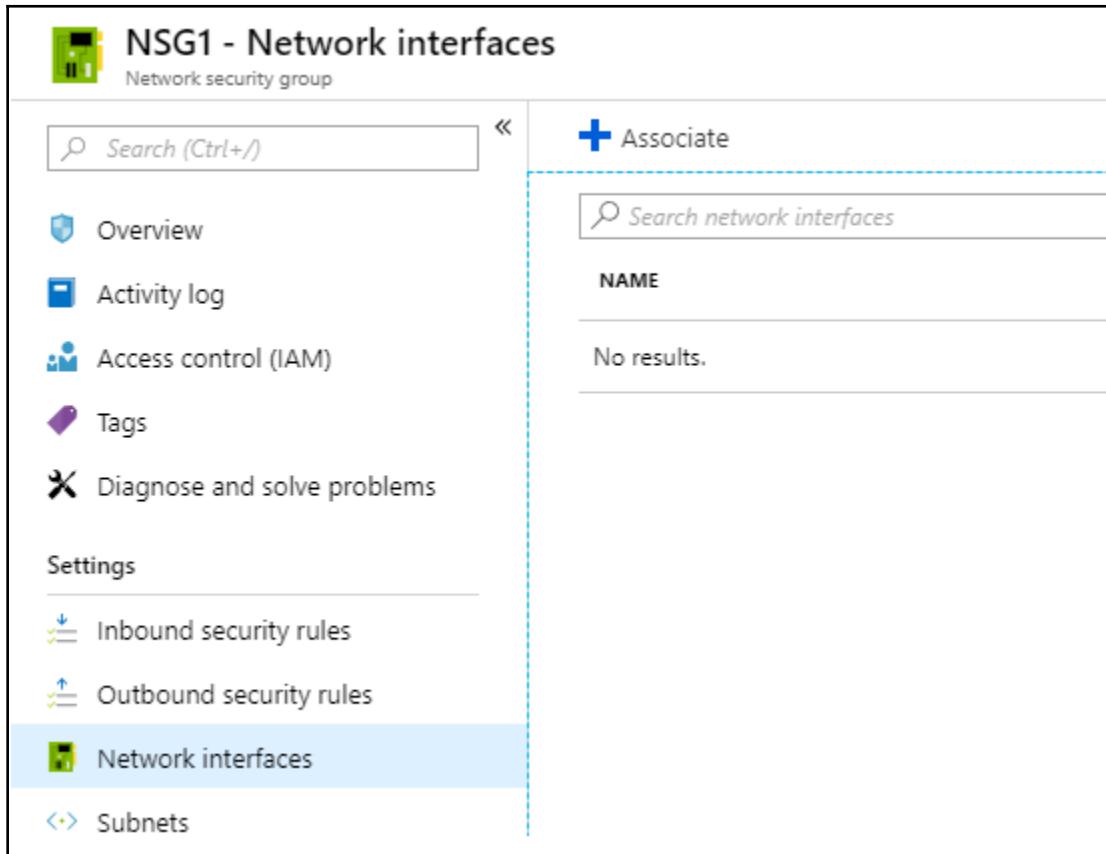
Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>. Locate the previously created NSG.

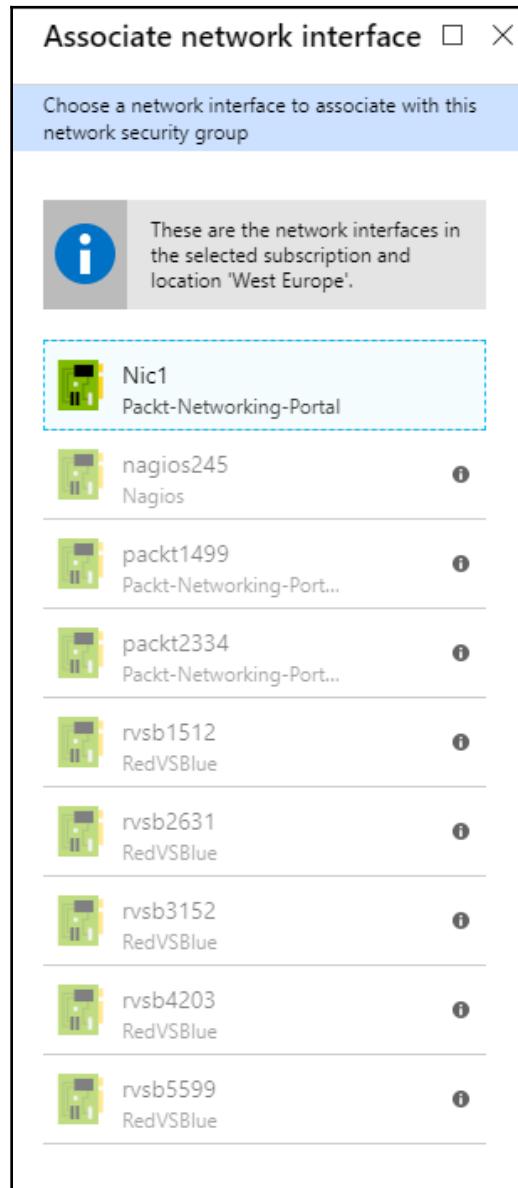
How to do it...

To assign an NSG to a network interface, we must follow these steps:

1. In the NSG blade, locate the **Network interfaces** option under **Settings**
2. Click on the **Associate** button at the top of the page and wait for the new blade to open:



3. Select the NIC you want to associate it with from the list of those available:



How it works...

When an NSG is associated with a NIC, the NSG rules will apply only to a single NIC (or a VM associated with the NIC). The NIC can be associated with only one NSG directly, but a subnet associated with a NIC can have an association with another NSG (or more of them). This is similar to when we have more NSGs in a single subnet, and the deny rule will take higher priority. If either of the NSGs allows traffic on a port, but another NSG is blocking it, traffic will be denied.

Assigning an NSG with PowerShell

Alternatively, we can associate an NSG using Azure PowerShell. In this recipe, we are going to show you how to associate an NSG with a subnet.

Getting ready

Open the PowerShell console and make sure you are connected to your Azure subscription.

How to do it...

To associate an NSG with a subnet, execute the following command:

```
$vnet = Get-AzureRmVirtualNetwork -Name 'Packt-Script' -ResourceGroupName  
'Packt-Networking-Script'  
$subnet = Get-AzureRmVirtualNetworkSubnetConfig -VirtualNetwork $vnet -Name  
BackEnd  
$nsg = Get-AzureRmNetworkSecurityGroup -ResourceGroupName 'Packt-  
Networking-Script' -Name 'nsg1'  
$subnet.NetworkSecurityGroup = $nsg  
Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

How it works...

Using PowerShell, we need to collect information on the virtual network, subnet, and NSG. When all of the information is gathered, we can perform the association using `Set-AzureRmVirtualNetwork` and apply changes.

Creating an Application Security Group (ASG)

ASGs are an extension of NSGs, allowing us to create additional rules and better control of traffic. Using only NSGs allows us to create rules that will allow traffic only for a specific source, IP address, or subnet. ASGs allow us to create better filtering and create additional checks on which traffic is allowed.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

How to do it...

To create an ASG using the Azure portal, we must follow these steps:

1. In the Azure portal, select **Create a resource** and choose **Application security group** under the **Networking** services (or search for application security group in the search bar).
2. The parameters we need to define for deployment are **Subscription**, **Resource group**, **Name**, and **Region**. An example of the required parameters is shown in the following screenshot:

Create an application security group

Basics Tags Review + create

PROJECT DETAILS

* Subscription Microsoft Azure Sponsorship

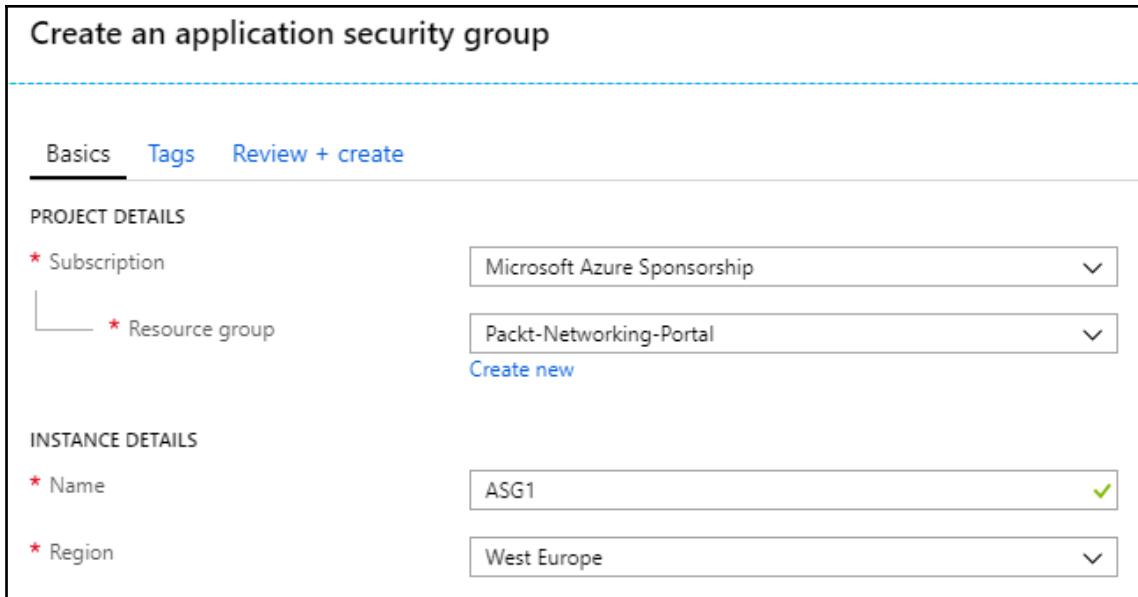
* Resource group Packt-Networking-Portal

Create new

INSTANCE DETAILS

* Name ASG1

* Region West Europe



How it works...

ASGs don't make much difference on their own and must be combined with NSGs to create NSG rules that will allow for better control of traffic and apply additional checks before traffic flow is allowed.

Associating an ASG with a VM

After creating an ASG, we must associate it with a VM. After this is completed, we can create rules with the NSG and ASG for traffic control.

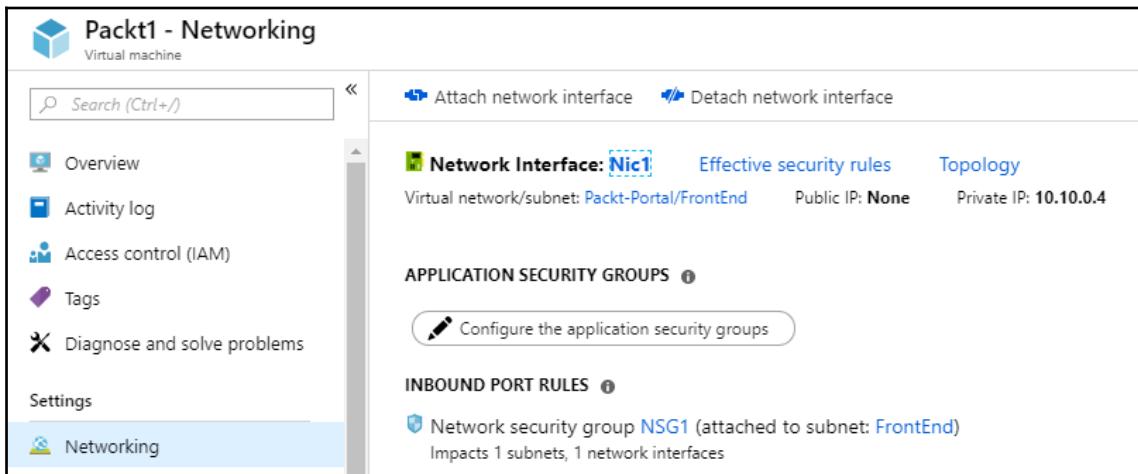
Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>. Locate the previously created virtual machine.

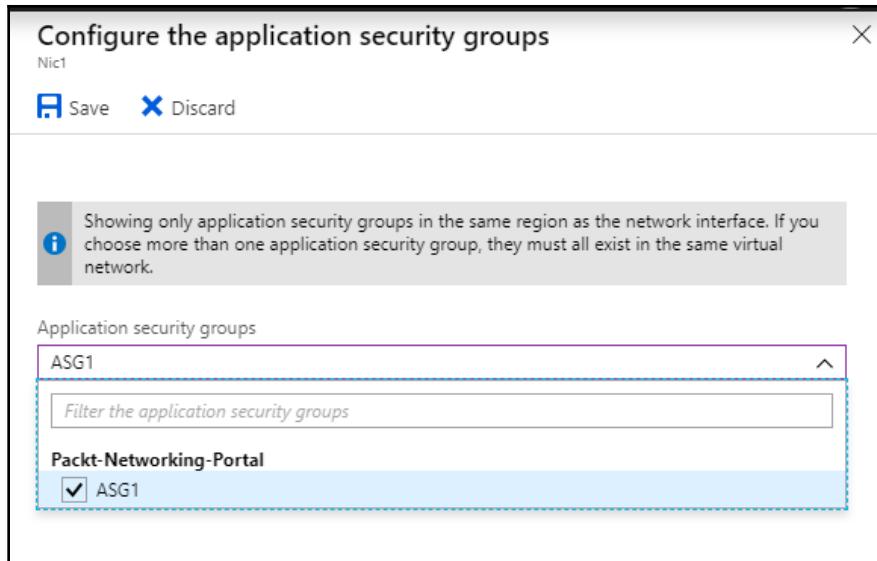
How to do it...

To associate an ASG with a virtual machine, we must follow these steps:

1. In the virtual machine blade, locate the **Networking** settings
2. In the **Networking** settings, select the **Configure the application security groups** option, as shown in the following screenshot:



3. In the new blade from the list of available ASGs, select the ASG that you want to associate the VM with:



4. After clicking **Save**, it takes a few seconds to apply changes until the VM is associated with the ASG

How it works...

The VM must be associated with the ASG. We can associate more than one VM with each ASG. The ASG is then used in combination with the NSG to create new NSG rules.

Creating rules with an NSG and an ASG

As a final step, we can use NSGs and ASGs to create new rules with better control. This approach allows us to have better control of traffic, limiting incoming traffic not only to a specific subnet, but only if the resource is also part of the ASG.

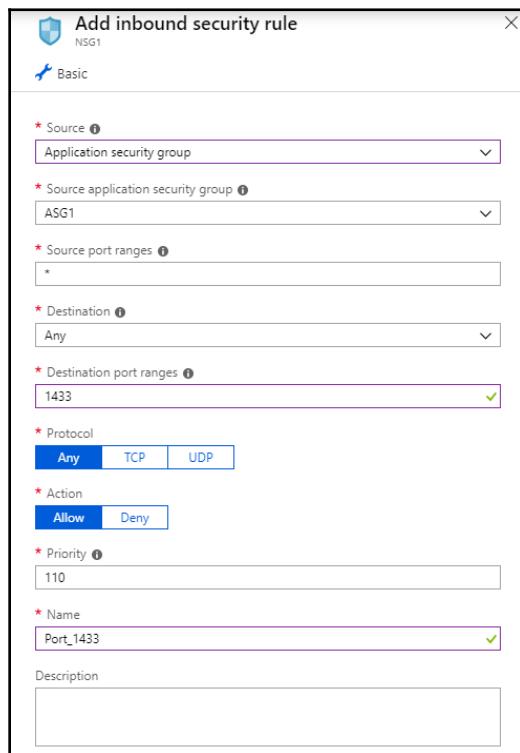
Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>. Locate the previously created NSG.

How to do it...

To create a rule using both an ASG and an NSG, we must follow these steps:

1. In the NSG blade, find **Inbound security rules**. Select **Add** to add a new rule.
2. For the source, select **Application Security Group**, and then select the ASG you want to use as the source. We also need to provide parameters for **Source**, **Source port ranges**, **Destination**, **Destination port ranges**, **Protocol**, **Action**, **Priority**, and **Name**. An example is shown in the following screenshot:



How it works...

Using only NSGs to create rules, we can create allow or deny traffic only for a specific IP address or range. With an ASG, we can widen or narrow this as needed. For example, we can create a rule to allow VMs from a frontend subnet, but only if these VMs are in a specific ASG. Alternatively, we can allow access to a number of VMs from different virtual networks and subnets, but only if they belong to a specific ASG.

4

Managing IP Addresses

In Azure, we can have two types of IP addresses, private and public. Public addresses can be accessed over the internet. Private addresses are from the Azure Virtual Network address space and are used for private communication on private networks. Addresses can be assigned to a resource or exist as a separate resource.

We will cover the following recipes in this chapter:

- Creating a new public IP address in the portal
- Creating a new public IP address with PowerShell
- Assigning a public IP address
- Unassigning a public IP address
- Creating a reservation for a public IP address
- Removing a reservation for a public IP address
- Creating a reservation for a private IP address
- Changing a reservation for a private IP address
- Removing a reservation for a private IP address

Technical requirements

For this chapter, the following is required:

- An Azure subscription
- Azure PowerShell

Code samples can be found at <https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter04>.

Creating a new public IP address in the portal

Public IP addresses can be created as a separate resource or during the creation of some other resources (a virtual machine, for example). Therefore, the public IP can exist as part of a resource or as a standalone resource of its own. First, we are going to show you how to create a new public IP address.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

How to do it...

To create a new public IP address, we must follow these steps:

1. In the Azure portal, select **Create a resource** and choose **Public IP address** under the **Networking** services (or search for `public IP address` in the search bar).
2. The parameters we need to define for deployment are **Name**, **SKU**, **IP Version**, **IP address assignment**, **DNS name label**, **Subscription**, **Resource group**, and **Location**. An example of the required parameters is shown in the following screenshot:

Create public IP address X

* Name ✓

* SKU !
 Basic Standard

* IP Version !
 IPv4 IPv6

* IP address assignment
 Dynamic Static

* Idle timeout (minutes) !
 4

DNS name label !

.westeurope.cloudapp.azure.com

Create an IPv6 address

* Subscription

* Resource group
 ▼
[Create new](#)

* Location
 ▼

How it works...

The SKU can be either basic or standard. The main difference is that standard is closed to inbound traffic by default (inbound traffic must be whitelisted in NSG), and standard is zone redundant. Another difference is that a standard SKU public IP address has a static assignment and a basic SKU can be either static or dynamic.

You can choose either the IPv4 or IPv6 version of an IP, but choosing IPv6 will limit you to a dynamic assignment.

The DNS name label is optional, and it can be used to resolve the endpoint in case a dynamic assignment is selected. Otherwise, there is no point in creating a DNS label, as an IP address can always be used to resolve the endpoint in case a static assignment is selected.

Creating a new public IP address with PowerShell

Alternatively, we can create a public IP address using Azure PowerShell. Again, this approach is better when we want to automate the process. Even a public IP address can exist on its own; it's usually created to be joined with other resources and to be used as an endpoint. When using PowerShell to create a resource, we can continue with the next step and join it with a resource in a single script.

Getting ready

Open the PowerShell console and make sure you are connected to your Azure subscription.

How to do it...

To deploy a new public IP address, execute the following command:

```
New-AzureRmPublicIpAddress -Name 'ip-public-script' -ResourceGroupName  
'Packt-Networking-Script' -AllocationMethod Dynamic -Location 'westeurope'
```

How it works...

As an outcome, a new public IP address will be created. Settings in this case will be a basic SKU dynamic assignment, IPv4 version, and without a DNS label. Furthermore, we can use additional switches like `-SKU`, `-IPAddressVersion`, or `-DomainNameLabel` to define these options if needed.

Assigning a public IP address

A public IP address can be created as a separate resource or can be unassigned from another resource and exist on its own. Such an IP address can then be assigned to a new or another resource. If the resource is no longer in use or migrated, we can still use the same public IP address. In this case, the public endpoint that's used to access a service may stay unchanged. This can be useful when a publicly available application or service is migrated or upgraded as we can keep using the same endpoint and end users don't need to be aware of any change.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

How to do it...

To assign a public IP address, we must do the following:

1. Locate the network interface that you want the IP address to be assigned to. This can be done directly by finding the NIC or through the VM blade that the NIC is assigned to.

2. In the NIC blade, go to **IP configurations** under **Settings** and select the configuration, as shown in the following screenshot:

The screenshot shows the 'Nic1 - IP configurations' blade for a Network interface. The left sidebar has sections for Overview, Activity log, Access control (IAM), Tags, Settings (selected), IP configurations (selected), DNS servers, Network security group, Properties, Locks, Automation script, Support + troubleshooting, Effective security rules, Effective routes, and New support request. The main area has tabs for IP forwarding settings and IP configurations. Under IP configurations, there is a Subnet section with a dropdown set to 'FrontEnd (10.10.0.0/25)'. Below it is a table with columns NAME, IP VERSION, TYPE, and PRIVATE IP ADDRESS. One row is listed: ipconfig1, IPv4, Primary, 10.10.0.4 (Dynamic).

NAME	IP VERSION	TYPE	PRIVATE IP ADDRESS
ipconfig1	IPv4	Primary	10.10.0.4 (Dynamic)

3. In the new blade, select to enable **Public IP address** and select the public IP address that you want to assign. Only unassigned IP addresses in the same region will show in the list. An example of this is shown in the following screenshot:

The screenshot shows the Azure portal interface for managing IP addresses. On the left, a sidebar titled 'ipconfig1' shows 'Public IP address settings' with 'Enabled' selected. Below it, 'Private IP address settings' and 'Dynamic' are shown. On the right, a modal window titled 'Choose public IP address' displays a list of available public IP addresses in the 'West Europe' location. The list includes:

IP Address	Subnet	Assignment Type
ip-public-portal	Packt-Networking-Portal	
RvsB1-ip	RedVSBlue	
RvsB2-ip	RedVSBlue	
RvsB3-ip	RedVSBlue	
RvsB4-ip	RedVSBlue	
RvsB5-ip	RedVSBlue	

4. After the public IP address has been selected, click **Save** to apply the settings.

How it works...

A public IP address exists as a separate resource and can be assigned to a resource at any time. When a public IP address is assigned, you can use this IP address to access services running on a resource that the IP address is assigned to (an appropriate NSG must be applied). We can also remove an IP address from a resource and assign it to a new resource. For example, if we want to migrate services to a new virtual machine, the IP address can be removed from the old VM and assigned to the new one. This way, service endpoints running on the VM will not change. This is especially useful when static IP addresses are used.

Unassigning a public IP address

A public IP address can be unassigned from a resource in order to be saved for later use or assigned to another resource. When a resource is deleted or decommissioned, we can still put the public IP address to use and assign it to the next resource.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>. Make sure that the virtual machine using a public IP address is not running.

How to do it...

1. Locate the NIC that the public IP address is associated with
2. In the NIC blade, go to **IP configurations** under **Settings** and select the IP configuration:

Nic1 - IP configurations

Network interface

Search (Ctrl+ /)

Add Save Discard

IP forwarding settings

IP forwarding: Disabled

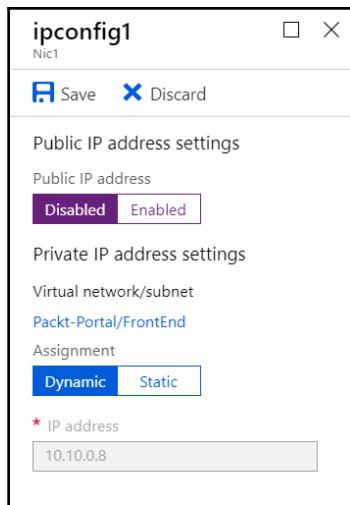
Virtual network: Packt-Portal

IP configurations

* Subnet: FrontEnd (10.10.0.0/25)

NAME	IP VERSION	TYPE	PRIVATE IP ADDRESS
ipconfig1	IPv4	Primary	10.10.0.4 (Dynamic)

3. In the new blade, set **Public IP address** to **Disabled**:



4. After the changes are made, click **Save** to apply the new configuration

How it works...

A public IP address can be assigned or unassigned from a resource in order to save it for future use or to transfer it to a new resource. To remove it, we simply disable the public IP address in the IP configuration under the NIC that the IP address is assigned to. This will remove the association but keep the IP address as a separate resource.

Creating a reservation for a public IP address

The default option for a public IP address is a dynamic IP assignment. This can be changed during the public IP address creation or later. Then, the public IP address becomes reserved (or static).

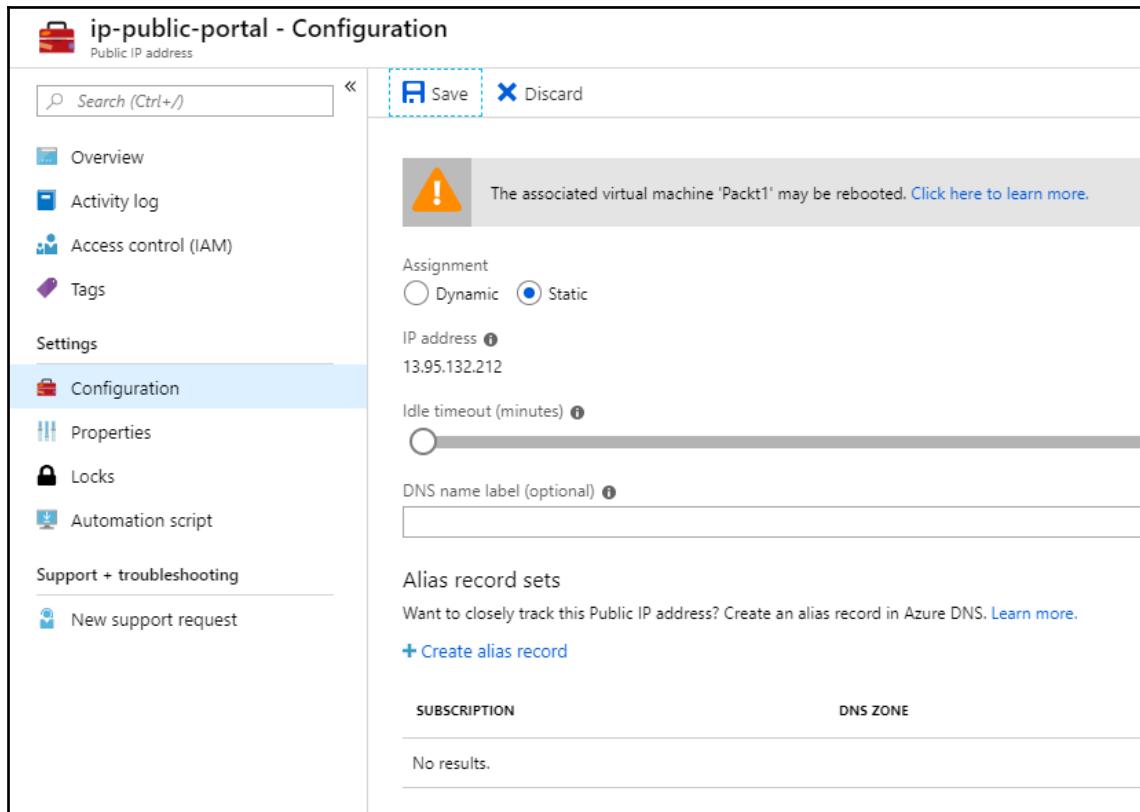
Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

How to do it...

To create a reservation for a public IP address, follow these steps:

1. Locate the public IP address in the Azure portal. This can be done by finding the IP address directly or through the resource it's assigned to (either the NIC or VM).
2. In the IP address blade, go to **Configuration** under **Settings**. Change **Assignment** from **Dynamic** to **Static**, as shown in the following screenshot:



3. After this change has been made, click **Save** to apply the new settings.

How it works...

A public IP address is set to dynamic by default. This means that an IP address might change in time. For example, if a VM that an IP address is assigned to is turned off or rebooted, there is a possibility that the IP address will change after the VM is up and running again. This can cause issues if services that are running on the VM are accessed over the public IP address, or there is a DNS record associated with the public IP address.

We create an IP reservation and set the assignment to static to avoid such a scenario and keep the IP address reserved for our services.

Removing a reservation for a public IP address

If the public IP address is set to static, we can remove a reservation and set the IP address assignment to dynamic. This isn't done often as there is usually a reason why the reservation is set in the first place. But as the reservation for the public IP address has an additional cost, there is sometimes a need to remove the reservation if it is not necessary.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>. Make sure that the IP address is not associated with any resource.

How to do it...

To remove a reservation for a public IP address, follow these steps:

1. Locate the public IP address in the Azure portal

2. In the public IP address blade, go to **Configuration** under **Settings** and set **Assignment** to **Dynamic**:

The screenshot shows the Azure portal's configuration blade for a public IP address. On the left, a sidebar lists various options like Overview, Activity log, Access control (IAM), Tags, Settings, Configuration (which is selected and highlighted in blue), Properties, Locks, and Automation script. Under Support + troubleshooting, there's a link to New support request. The main content area has a header with a search bar, Save, and Discard buttons. A warning message states: "The associated virtual machine 'Packt1' may be rebooted. Click here to learn more." Below this, the "Assignment" section shows "Dynamic" is selected (indicated by a blue radio button). The IP address is listed as 13.95.132.212. There's a slider for Idle timeout (minutes) which is set to 0. A section for DNS name label (optional) has an empty input field. At the bottom, there are two tables: one for SUBSCRIPTION (No results.) and one for DNS ZONE (No results.).

3. After these changes have been made, click **Save** to apply the new configuration

How it works...

To remove an IP reservation from a public IP address, the public IP address must not be associated with a resource. Then, we can remove the reservation by setting the IP address assignment to dynamic.

The main reason for this is pricing. In Azure, the first five public IP reservations are free. After the initial five, each new reservation is billed. To avoid paying anything unnecessary, we can remove a reservation when it is not needed or when the public IP address is not used.

Creating a reservation for a private IP address

Similar to public IP addresses, we can make a reservation for private IP addresses. This is usually done to ensure communication between servers on the same virtual network and to allow for the usage of IP addresses in connection strings.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

How to do it...

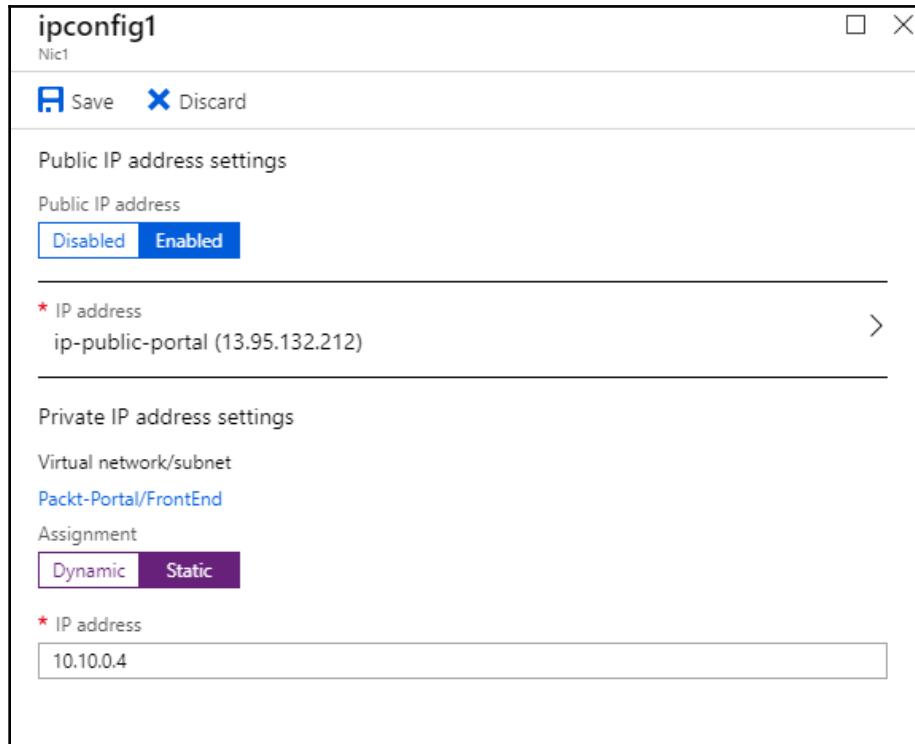
To create a reservation for a private IP address, follow these steps:

1. In the Azure portal, locate the NIC you want to make the reservation for.
2. In the NIC blade, go to **IP configurations** under **Settings** and select the IP configuration:

NAME	IP VERSION	TYPE	PRIVATE IP ADDRESS
ipconfig1	IPv4	Primary	10.10.0.4 (Dynamic)

3. In the new blade, under **Private IP address settings**, set **Assignment** to **Static**.

4. The current IP address value will be set automatically. If needed, you can change that value to another value, but it must be in the address space of the subnet associated with the NIC:



5. After these changes have been made, click **Save** to apply the new configuration.

How it works...

A reservation for an IP address can be made for private IP addresses. The difference is that a private IP address does not exist as a separate resource but is assigned to an NIC.

Another difference is that you can select a value for a private IP address. A public IP address is assigned randomly and can be reserved, but you cannot choose which value this will be. For private IP addresses, you can select the value for the IP, but it must be an unused IP from the subnet associated with the NIC.

Changing a reservation for a private IP address

For private IP addresses, you can change an IP address at any time to another value. With public IP addresses, this isn't the case as you get the IP address randomly from a pool, and you aren't able to change the value for public IP addresses. With a private IP address, you can change the value to another IP address from the address space.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

How to do it...

To change a reservation for a private IP address, follow these steps:

1. In the Azure portal, locate the NIC you want to make changes for
2. In the NIC blade, go to **IP configurations** under **Settings** and select the IP configuration:

Nic1 - IP configurations

Network interface

Search (Ctrl+ /)

Add Save Discard

IP forwarding settings

IP forwarding: Enabled

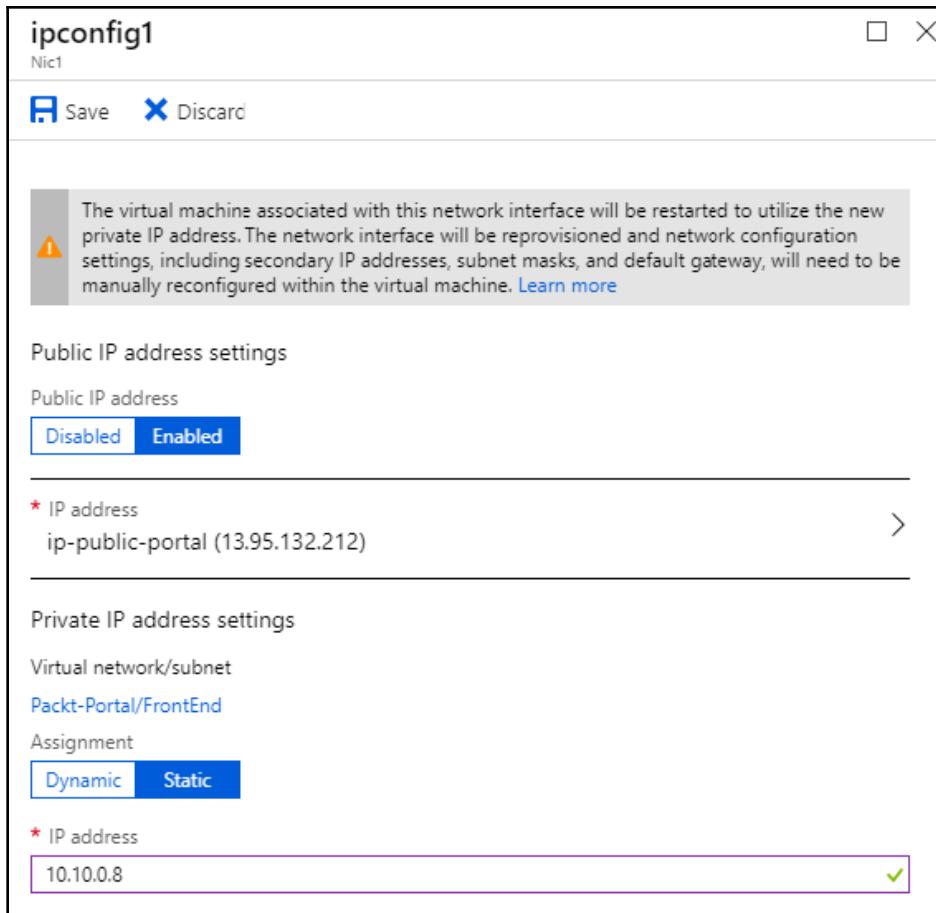
Virtual network: Packt-Portal

IP configurations

* Subnet: FrontEnd (10.10.0.0/25)

NAME	IP VERSION	TYPE	PRIVATE IP ADDRESS
ipconfig1	IPv4	Primary	10.10.0.4 (Dynamic)

3. In **Private IP address settings**, enter a new value for the **IP address**:



4. After these changes have been made, click **Save** to apply the new configuration

How it works...

A reservation for a private IP address can be changed. Again, the value must be an unused IP address from a subnet associated with the NIC. If the virtual machine associated with the NIC is turned off, the new IP address will be assigned on its next startup. If the virtual machine is running, it will be restarted to apply new changes.

Removing a reservation for a private IP address

Similar to public IP addresses, we can remove a reservation at any time. A private IP address is free, so additional costs aren't a factor in this case. But there are scenarios where a dynamic assignment is required, and we can set it at any time.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

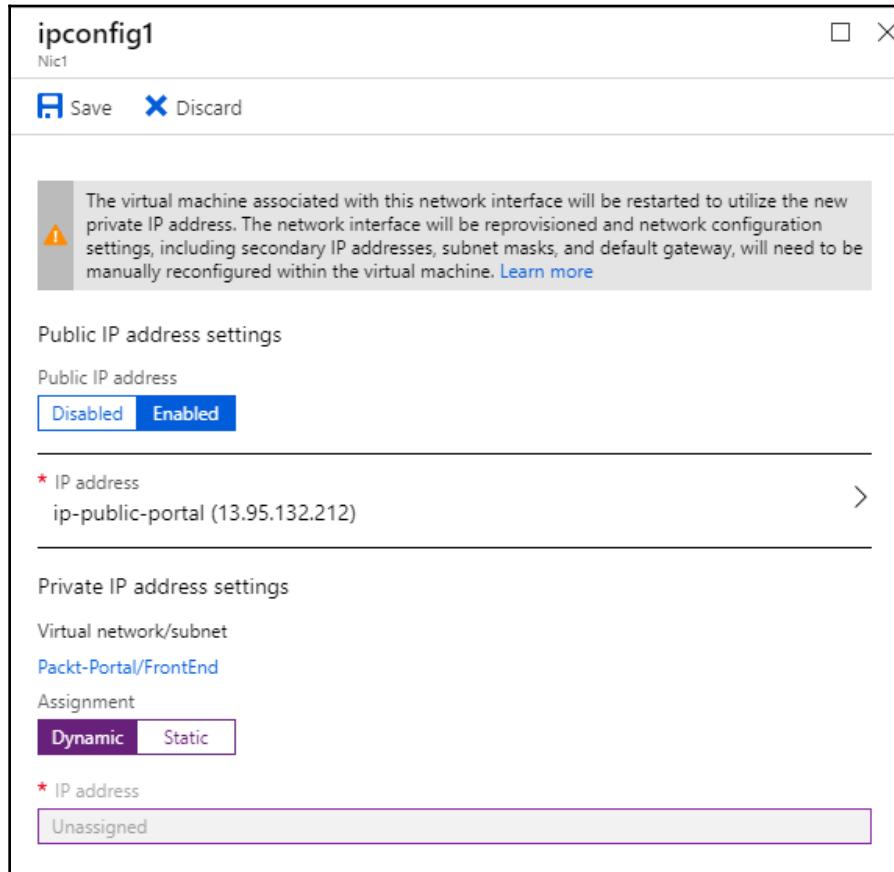
How to do it...

To remove a reservation for a private IP address, follow these steps:

1. In the Azure portal, locate the NIC you want to make changes for
2. In the NIC blade, go to **IP configurations** under **Settings** and select the IP configuration:

NAME	IP VERSION	TYPE	PRIVATE IP ADDRESS
ipconfig1	IPv4	Primary	10.10.0.8 (Static)

3. In the new blade, under **Private IP address settings**, change **Assignment** to **Dynamic**:



4. After these changes have been made, click **Save** to apply the new configuration

How it works...

We can remove a private IP address reservation at any time by switching the assignment to dynamic. When this change is made, the virtual machine associated with the NIC will be restarted apply the new changes. After a change is made, a private IP address may change after the VM is restarted or turned off.

5

Local and Virtual Network Gateways

Local and virtual network gateways are **virtual private network (VPN)** gateways that are used to connect to on-premises networks and encrypt all traffic going between Azure **Virtual Network (VNet)** and a local network. Each virtual network can have only one virtual network gateway, but one virtual network gateway can be used to configure multiple VPN connections.

We will cover the following recipes in this chapter:

- Creating a local network gateway in the portal
- Creating a local network gateway with PowerShell
- Creating a virtual network gateway in the portal
- Creating a virtual network gateway with PowerShell
- Modifying the local network gateway settings

Technical requirements

For this chapter, the following is required:

- An Azure subscription
- Azure PowerShell

Code samples can be found in <https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter05>.

Creating a local network gateway in the portal

Although a local network gateway is created in Azure, it represents your local (on-premises) network and holds configuration information on your local network settings. It's an essential component for creating the VPN connection that is needed to create a Site-to-Site connection between the Azure VNet and the local network.

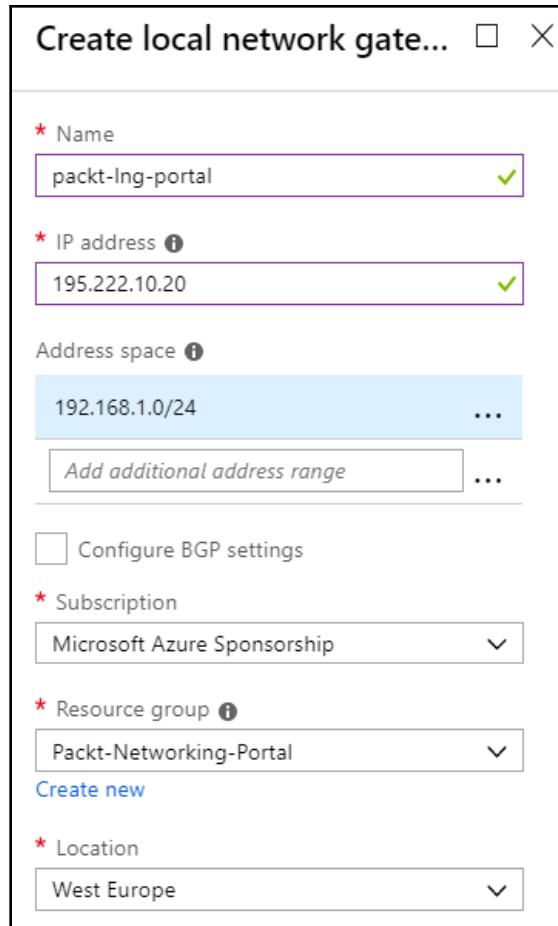
Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>.

How to do it...

In order to create a new local network gateway, the following steps are required:

1. In the Azure portal, select **Create a resource** and choose **Local network gateway** under the **Networking** services (or search for **local network gateway** in the search bar).
2. The parameters that we need to provide are **Name**, **IP address** (that is, the public IP address of the local firewall), **Address space** (the local address space that you want to connect to), **Subscription**, **Resource group**, and **Location**. Optionally, we can configure **Border Gateway Protocol (BGP)** settings:



How it works...

The local network gateway is used to connect a virtual network gateway to an on-premises network. The virtual network gateway is directly connected to the virtual network and has all the relevant VNet information needed to create a VPN connection. On the other hand, a local network gateway holds all the local network information needed to create a VPN connection.

Creating a local network gateway with PowerShell

As mentioned in the previous recipe, the local network gateway holds information on the local network that we want to connect to Azure VNet. In addition to creating a local network gateway through the Azure portal, we can create it with Azure PowerShell.

Getting ready

Open the PowerShell console and make sure you are connected to your Azure subscription.

How to do it...

To create a new local network gateway, execute the following command:

```
New-AzureRmLocalNetworkGateway -Name packt-lng-script -ResourceGroupName  
'Packt-Networking-Script' -Location 'westeurope' -GatewayIpAddress  
'195.222.10.20' -AddressPrefix '192.168.1.0/24'
```

How it works...

In order to deploy a new local network gateway, we need to provide parameters for name, resource group, location, gateway IP address, and address prefix. The gateway IP address is the public IP address of the local firewall that you are trying to connect to. The address prefix is the subnet prefix of the local network that you are trying to connect to. This address must be associated with a firewall address that is provided as a gateway IP address.

Creating a virtual network gateway in the portal

After a local network gateway is created, we need to create a virtual network gateway in order to create a VPN connection between the local and Azure networks. As a local network gateway holds information on the local network, the virtual network gateway holds information for the Azure VNet that we are trying to connect.

Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>.

How to do it...

In order to create a new virtual network gateway, the following steps are required:

1. In the Azure portal, select **Create a resource** and choose **Virtual network gateway** under the **Networking** services (or search for **virtual network gateway** in the search bar).
2. Everything is done in a single blade, but for the purpose of better visibility, I'm going to break it down into three sections. In the first section, we need to provide **Name**, **Gateway type**, **VPN type**, and **SKU**. Optionally, we can select **Enable active-active mode**:

The screenshot shows the 'Create virtual network gateway' blade. It includes a planning guide link, fields for Name (filled with 'packt-vng-portal'), Gateway type (set to VPN), VPN type (set to Route-based), SKU (set to 'VpnGw1'), and an optional checkbox for 'Enable active-active mode'.

Create virtual network gateway

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

* Name
packt-vng-portal

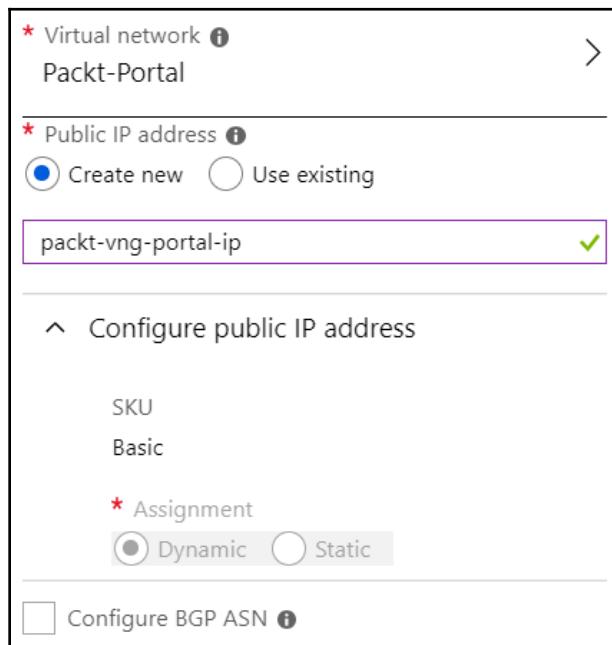
Gateway type i
 VPN ExpressRoute

VPN type i
 Route-based Policy-based

* SKU i
VpnGw1

Enable active-active mode i

3. In the second section, we need to select **Virtual network** (that will be used in the connection), and set the **Public IP address** options. Note that the gateway subnet must be created prior to this, and only virtual networks with a gateway subnet will be available for selection:



4. In the final section, we need to select the **Subscription** and **Location** options of where the virtual network gateway will be created:

* Subscription

Microsoft Azure Sponsorship

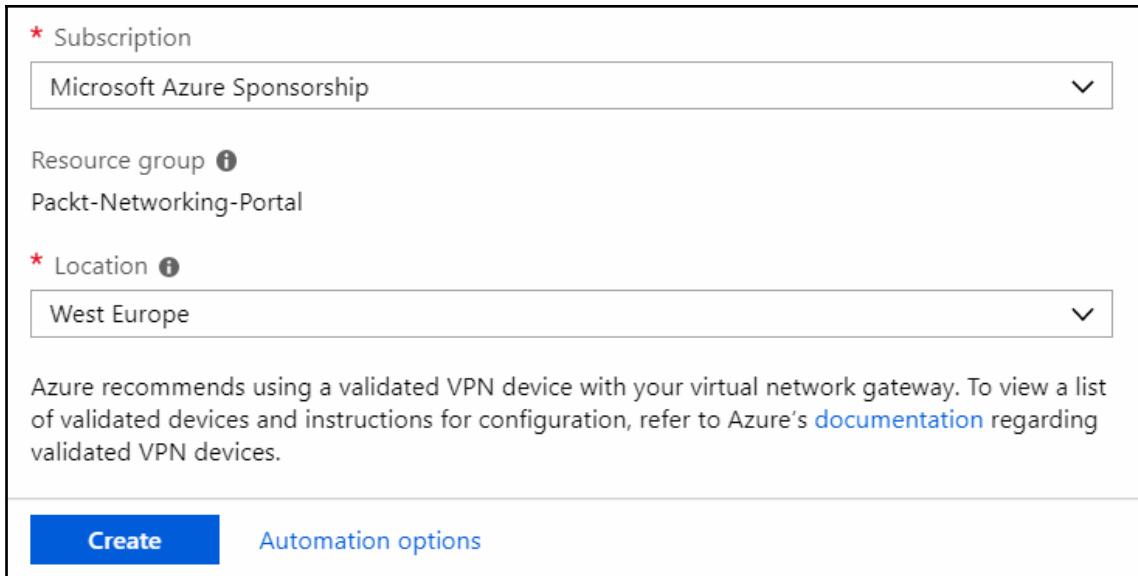
Resource group ⓘ
Packt-Networking-Portal

* Location ⓘ

West Europe

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

Create Automation options



5. After validation, we can click on **Create** and start deployment. Note that creating the virtual network gateway takes longer than most other Azure resources; deployment can take from 45 to 90 minutes.

How it works...

The virtual network gateway is the second part needed to establish the connection to the Azure VNet. It's directly connected to the virtual network and is needed to create both Site-to-Site and Point-to-Site connections. We need to set the VPN type that needs to match to the type of the local VPN device when a Site-to-Site connection is created.

Creating a virtual network gateway with PowerShell

Creating a virtual network gateway is possible with PowerShell. Again, this helps automate processes. For example, if we start creating a virtual network gateway using a portal and notice that our virtual network isn't listed, it's probably because it's missing a gateway subnet. So, we must abandon the process, go back and create the gateway subnet, and start creating the virtual network gateway. Using PowerShell, we can ensure that all the requisite resources are present before starting and then continue with creating the virtual network gateway.

Getting ready

Open the PowerShell console and make sure you are connected to your Azure subscription.

How to do it...

To create a new virtual network gateway, execute the following command:

```
$vnet = Get-AzureRmVirtualNetwork -ResourceGroupName 'Packt-Networking-Script' -Name 'Packt-Script'  
Add-AzureRmVirtualNetworkSubnetConfig -Name 'GatewaySubnet' -AddressPrefix 10.11.2.0/27 -VirtualNetwork $vnet  
$vnet | Set-AzureRmVirtualNetwork  
$gwpip = New-AzureRmPublicIpAddress -Name VNet1GWIP -ResourceGroupName 'Packt-Networking-Script' -Location 'westeurope' -AllocationMethod Dynamic  
  
$vnet = Get-AzureRmVirtualNetwork -ResourceGroupName 'Packt-Networking-Script' -Name 'Packt-Script'  
$subnet = Get-AzureRmVirtualNetworkSubnetConfig -Name 'GatewaySubnet' -VirtualNetwork $vnet  
$gwipconfig = New-AzureRmVirtualNetworkGatewayIpConfig -Name gwipconfig1 -SubnetId $subnet.Id -PublicIpAddressId $gwpip.Id  
New-AzureRmVirtualNetworkGateway -Name VNet1GW -ResourceGroupName 'Packt-Networking-Script' -Location 'westeurope' -IpConfigurations $gwipconfig -GatewayType Vpn -VpnType RouteBased -GatewaySku VpnGw1
```

How it works...

The script performs a few different operations to make sure all requirements are met so that we can create a virtual network gateway. The first step is to collect information on the virtual network that we are going to use. Next, we add the gateway subnet to VNet, and create a public IP address that will be used by the virtual network gateway. We collect all the information and ensure that all the required resources are present, and then finally create a new virtual network gateway.

Modifying the local network gateway settings

Network configurations may change over time and we may need to address these changes in Azure as well. For example, the public IP address of a local firewall may change and we need to reconfigure the local network gateway. Or, a local network might be reconfigured and the address space or subnet has changed, so we need to reconfigure the local network gateway once again.

Getting ready

Before you start, open a web browser and go to the Azure portal at <https://portal.azure.com>.

How to do it...

In order to modify local network gateway settings, we must do the following:

1. Locate the local network gateway in the Azure portal and go to **Configuration**.

2. In configuration, we can edit **IP address** or **Address space**. We can also add additional address spaces in case we want to connect multiple local subnets to the Azure VNet:

The screenshot shows the Azure portal interface for managing a Local network gateway. The title bar says "packt-lng-portal - Configuration". The left sidebar has several options: Overview, Activity log, Access control (IAM), Tags, Settings (which is selected), Configuration (which is highlighted in blue), Connections, and Properties. The main content area shows configuration details. At the top right are "Save" and "Discard" buttons. Below them, there's a section for "IP address" with a value of "195.222.10.20". Another section below it is for "Address space" with a value of "192.168.1.0/24". There is also a button labeled "Add additional address range" and a checkbox for "Configure BGP settings" which is currently unchecked.

How it works...

The local network gateway holds the local network information needed to create a Site-to-Site connection between the local and Azure networks. If this information changes, we can edit it in the **Configuration** settings. Changes that can be made are the IP address (that is, the public IP address of the local firewall) and the address space we are connecting to. Additionally, we can add or remove address spaces if we want to add or remove subnets that are able to connect to Azure VNet. If configuration in the local network gateway is no longer valid, we can still use it to create a completely new connection to a new local network if needed.

6

Creating Hybrid Connections

Hybrid connections allow us to create secure connections with Azure VNets. These connections can either be from on-premises or from other Azure VNets. Establishing connections to Azure VNet enables secure network traffic with other services that are located in different Azure VNets, different subscriptions, or outside Azure (in different clouds or on-premises). Using secure connections removes the need for publicly exposed endpoints that present a potential security risk. This is especially important when we consider management, where opening public endpoints creates a security risk and presents a major issue. For example, if we consider managing virtual machines, it's a common practice to use either **Remote Desktop Protocol (RDP)** or PowerShell for management. Exposing these ports to public access presents a great risk. A best practice is to disable any kind of public access to such ports and use only access from an internal network for management. In this case, we use either a Site-2-Site or a Point-2-Site connection to enable secure management.

In another scenario, we might need the ability to access a service or a database on another network, either on-premises or via another Azure VNet. Again, exposing these services might present a risk, and we use either Site-2-Site, VNet-2-VNet, or VNet peering to enable such a connection in a secure way.

We will cover the following recipes in this chapter:

- Creating a Site-2-Site connection
- Downloading the VPN device configuration from Azure
- Creating Point-2-Site connection
- Creating a VNet-2-VNet connection
- Connecting VNets using network peering

Technical requirements

For this chapter, the following are required:

- Azure subscription
- Windows PowerShell

Code samples can be found at <https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter06>.

Creating a Site-2-Site connection

A Site-2-Site connection is used to create a secure connection between an on-premises network and Azure VNet. This connection is used to perform a number of different tasks, such as enabling hybrid connections or secure management. In a hybrid connection, we allow a service in one environment to connect to a service in another environment. For example, we could have an application in Azure that uses a database located in an on-premises environment. Secure management allows us to limit management operations to be allowed only when coming from a secure and controlled environment, from our local network.

Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

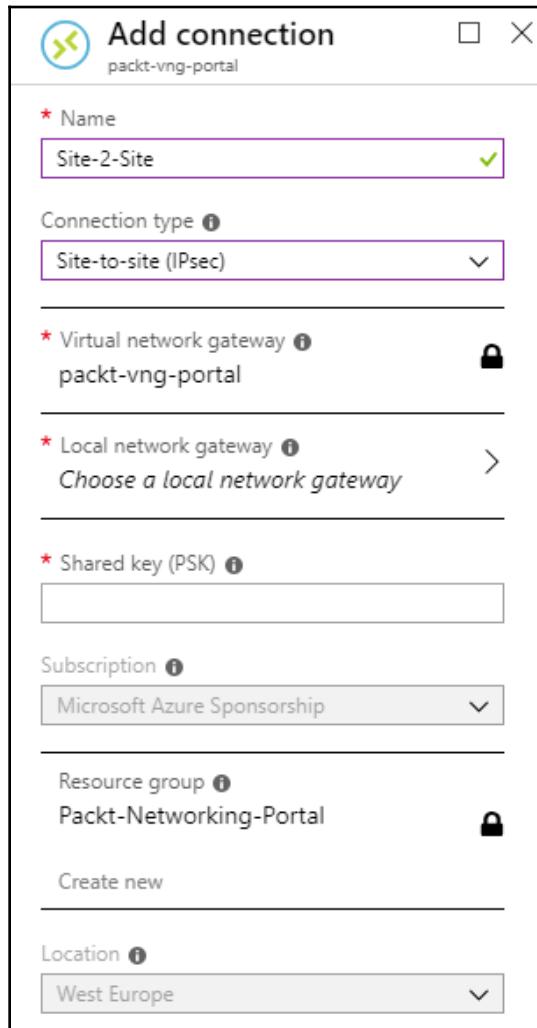
How to do it...

To create a new Site-2-Site connection, we must follow these steps:

1. Locate the virtual network gateway (the one we created in [Chapter 5, Local and Virtual Network Gateways](#)) and select **Connections**.
2. In **Connections**, select the **Add** option to add a new connection:

The screenshot shows the Azure portal interface for managing connections on a virtual network gateway. The left sidebar lists various navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Configuration, Connections, Point-to-site configuration, Properties, Locks, Automation script), Monitoring (Alerts, Metrics), and Support + troubleshooting (Resource health, Reset, New support request). The 'Connections' option under 'Settings' is currently selected and highlighted with a blue background. On the right, a large panel titled 'packt-vng-portal - Connections' displays a search bar ('Search connections') and a table with a single row labeled 'NAME' and 'No results'. A prominent blue 'Add' button with a plus sign is located at the top right of this panel.

3. In a new blade, we need to enter some information for the connection **Name** and select **Site-to-site (IPsec)** for **Connection type**:

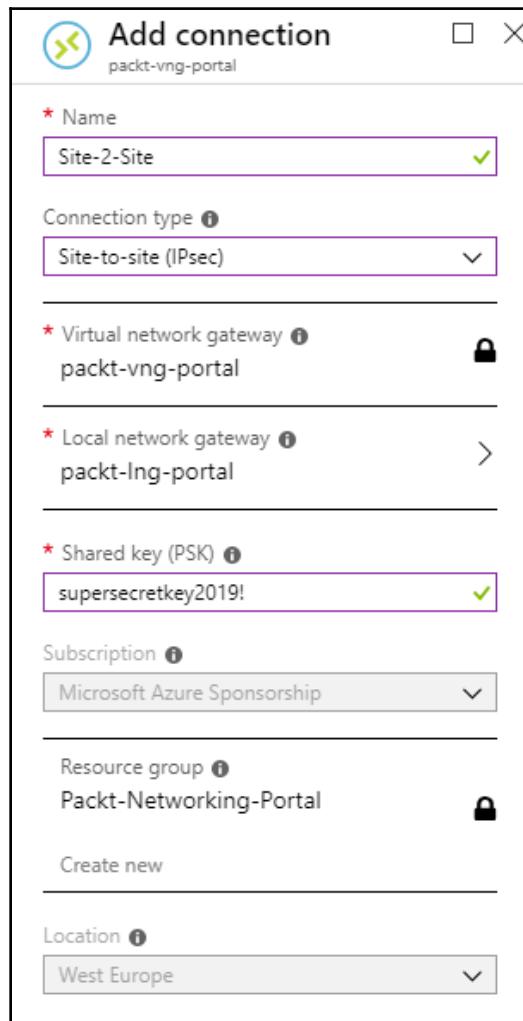


4. Under **Local network gateway**, we need to select a local network gateway from the list (a local network gateway was created in Chapter 5, *Local and Virtual Network Gateways*):

The screenshot shows two overlapping windows in the Azure portal:

- Add connection (Left Window):** This window is titled "Add connection" and has a resource group "packt-vng-portal" selected. It contains fields for:
 - Name:** Site-2-Site (Valid)
 - Connection type:** Site-to-site (IPsec) (Selected)
 - Virtual network gateway:** packt-vng-portal (Locked)
 - Local network gateway:** A dropdown menu with the option "Choose a local network gateway" highlighted.
 - Shared key (PSK):** An empty input field.
 - Subscription:** Microsoft Azure Sponsorship (Selected).
 - Resource group:** Packt-Networking-Portal (Locked).
 - Create new:** A link to create a new resource group.
 - Location:** West Europe (Selected).
- Choose local network gateway (Right Window):** This window lists existing local network gateways:
 - Create new:** A button with a plus sign.
 - packt-lng-portal:** Packt-Networking-Portal (Icon)
 - packt-lng-script:** Packt-Networking-Script (Icon)

5. We need to provide a **Pre-Shared Key (PSK)** that will be used for IPSec connection. Note that options for **Subscription**, **Resource group**, and **Location** are locked and will be the same as they are for the virtual network gateway:



6. Finally, we select **Create** and the deployment will start.

How it works...

Using the virtual network gateway, we set up the Azure side of the IPsec tunnel. The local network gateway provides information on the local network, defining the local side of the tunnel with the public IP address, and local subnet information. This way, Azure's side of the tunnel has all the relevant information that's needed to form a successful connection with an on-premises network. However, this completes only half of the work, as the opposite side of the connection must be configured as well. This part of the work really depends on the VPN device that's used locally, and each device has unique configuration steps. After both sides of the tunnel are configured, the result is a secure, encrypted VPN connection between networks.

Downloading the VPN device configuration from Azure

After creating the Azure side of the Site-2-Site connection, we still need to configure the local VPN device. Configuration depends upon the vendor and the device type. You can see all the supported devices here: <https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpn-devices>. In some cases, there is an option to download configuration for a VPN device directly from the Azure portal.

Getting ready

Before you start, open the browser and go to the Azure portal: <https://portal.azure.com>.

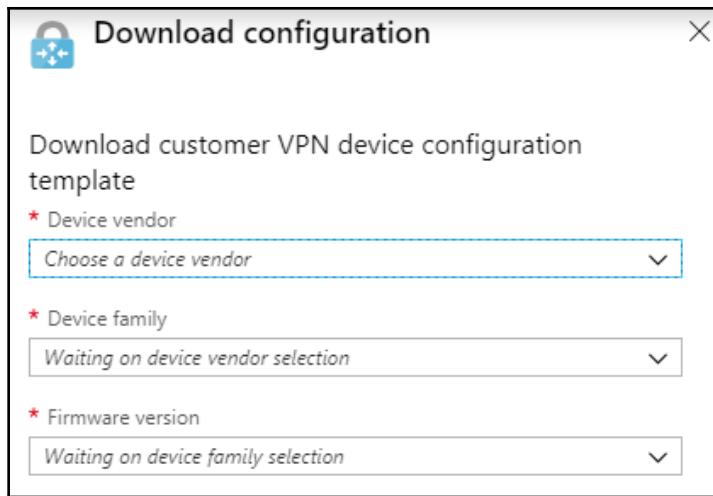
How to do it...

To download the VPN device configuration, we must follow these steps:

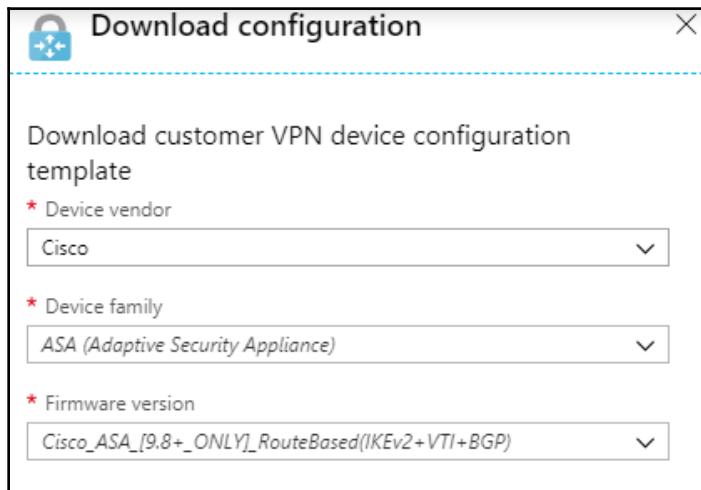
1. Locate the Site-2-Site connection in the Azure portal. **Overview** will be opened by default.
2. Select the **Download configuration** option from the top of the blade:

The screenshot shows the 'Site-2-Site Connection' blade in the Azure portal. On the left, there's a sidebar with icons for Overview, Activity log, Access control (IAM), Tags, Settings, Shared key, Configuration, and Properties. The 'Overview' item is selected and highlighted in blue. On the right, the main content area displays the connection details. At the top of this area are three buttons: 'Move', 'Download configuration' (which is highlighted in blue), and 'Delete'. Below these buttons, the connection information is listed in pairs: Resource group (change) : Packt-Networking-Portal, Status : Unknown, Location : West Europe, Subscription (change) : Microsoft Azure Sponsorship, Subscription ID : cb638267-a366-463c-bfe5-7a49311c27a8, and Tags (change) : Click here to add tags.

3. A new blade will open, and you will see that all the options in the blade are predefined:



4. Select the option for **Device vendor**, **Device family**, and **Firmware version**. Note that only some options are available, and not all the supported devices have this option. After all of these options have been selected, download the configuration file. The sample file can be found in the GitHub repository associated with this book:



5. After using the configuration file for the local VPN device, both sides of the IPsec tunnel are configured. **Status** under Site-2-Site connection will change to **Connected**:

The screenshot shows the Azure portal interface for a Site-2-Site connection. On the left, there's a sidebar with icons for Overview, Activity log, Access control (IAM), Tags, Settings, Shared key, Configuration, and Properties. The main area displays connection details:

Resource group (change) : Packt-Networking-Portal	
Status	: Connected
Location	: West Europe
Subscription (change)	: Microsoft Azure Sponsorship
Subscription ID	: cb638267-a366-463c-bfe5-7a49311c27a8
Tags (change)	: Click here to add tags

Now, let's have a look at how it works.

How it works...

After we set up the Azure side of the IPsec tunnel, we need to configure the other side as well as the local VPN device. The steps and configuration are different for each device. In some cases, we can download the configuration file directly from the Azure portal. After the VPN device has been configured, everything is set up, and we can use the tunnel for secure communication between the local network and Azure VNet.

Creating a Point-2-Site connection

Accessing resources in a secure way is important, and this must be performed securely. It's not always possible to perform this using a Site-2-Site connection, especially when we have to perform something out of work hours. In this case, we can use Point-2-Site to create a secure connection that can be established from anywhere.

Getting ready

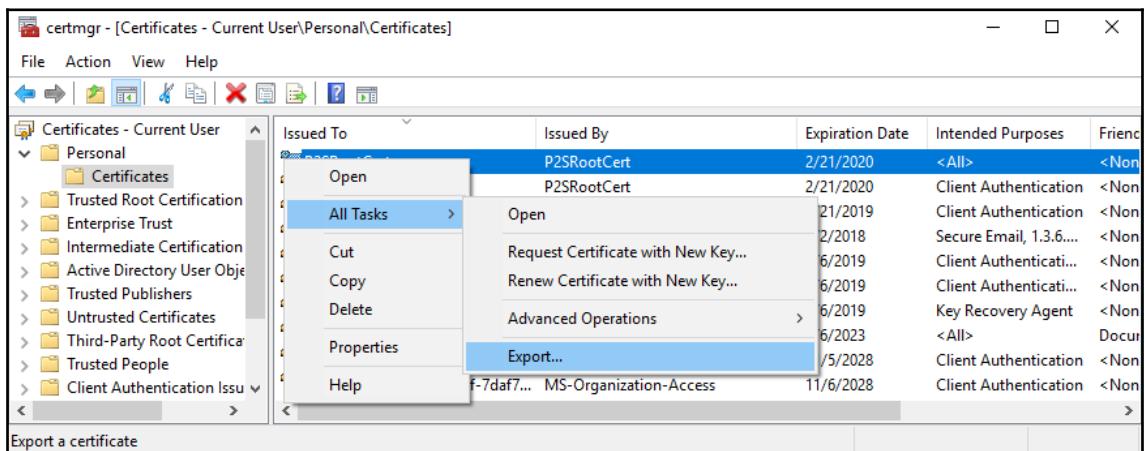
To create a Point-to-Site connection, we'll need to generate a certificate that will be used for connection. To create a certificate, we must follow these steps:

1. Execute the following PowerShell script to generate a certificate:

```
$cert = New-SelfSignedCertificate -Type Custom -KeySpec Signature ` 
-Subject "CN=P2SRootCert" -KeyExportPolicy Exportable ` 
-HashAlgorithm sha256 -KeyLength 2048 ` 
-CertStoreLocation "Cert:\CurrentUser\My" -KeyUsageProperty Sign - 
KeyUsage CertSign

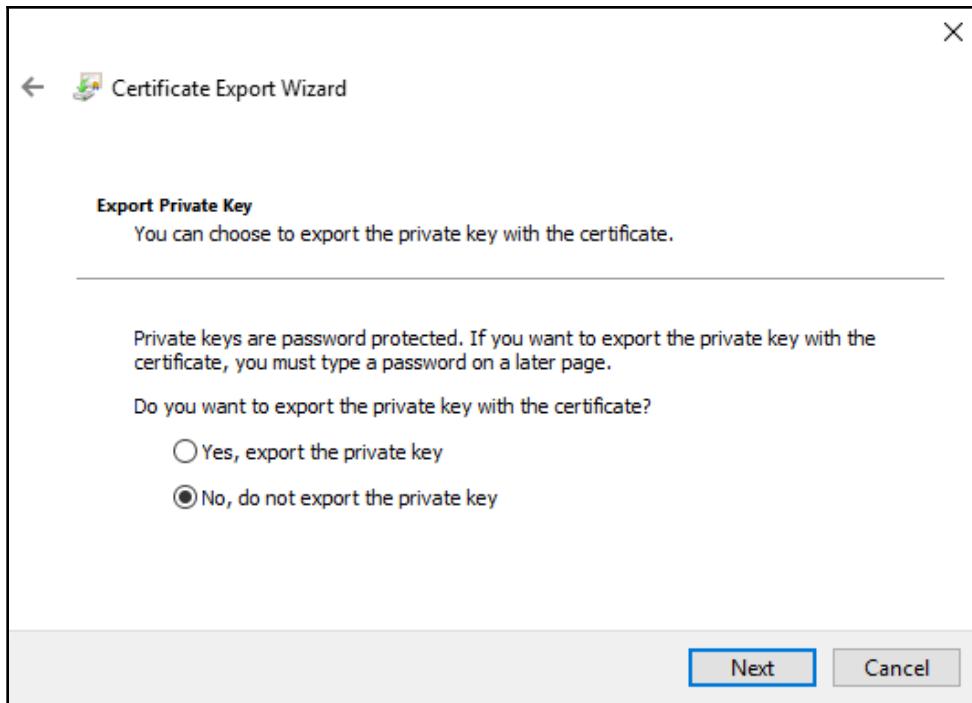
New-SelfSignedCertificate -Type Custom -DnsName P2SChildCert - 
KeySpec Signature ` 
-Subject "CN=P2SChildCert" -KeyExportPolicy Exportable ` 
-HashAlgorithm sha256 -KeyLength 2048 ` 
-CertStoreLocation "Cert:\CurrentUser\My" ` 
-Signer $cert -TextExtension @("2.5.29.37={text}1.3.6.1.5.5.7.3.2")
```

2. Next, we need to export the certificate. Open **certmgr**, locate the personal certificate, select **P2SRootCert**, and then choose the **Export...** option:

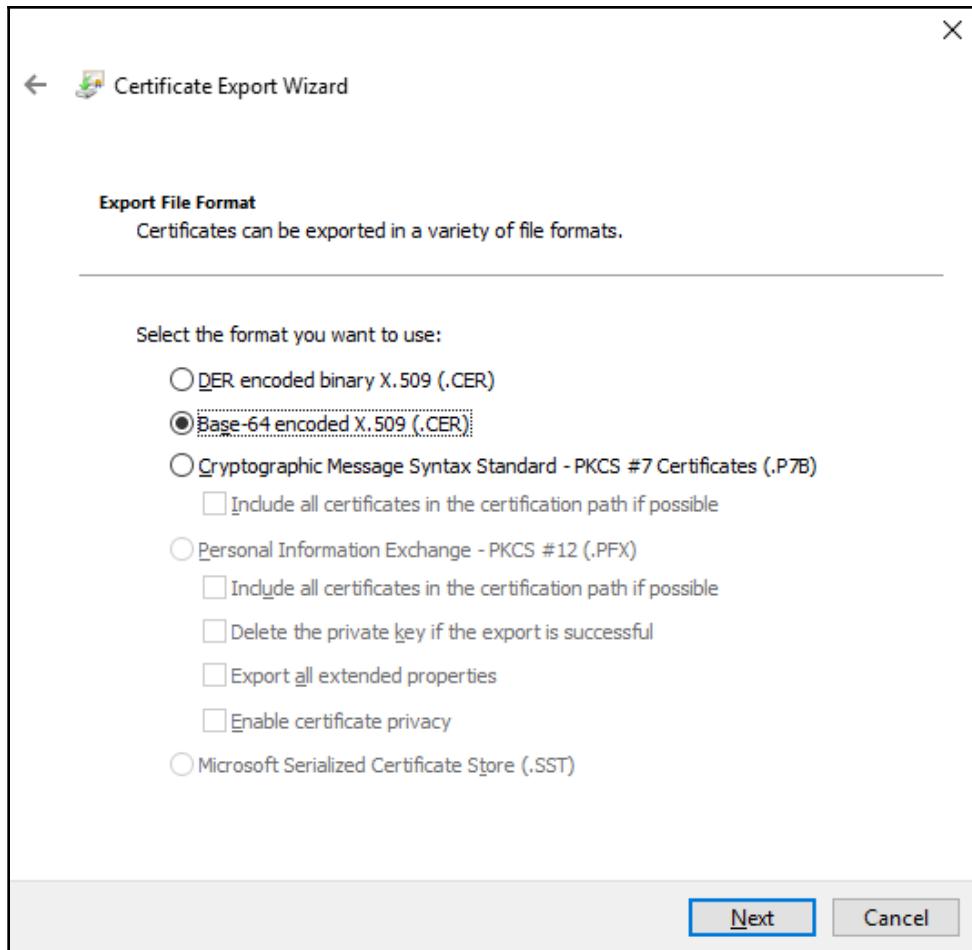


3. This will start the **Certificate Export Wizard**. Click **Next**.

4. Select the option **No, do not export the private key** and click **Next**:

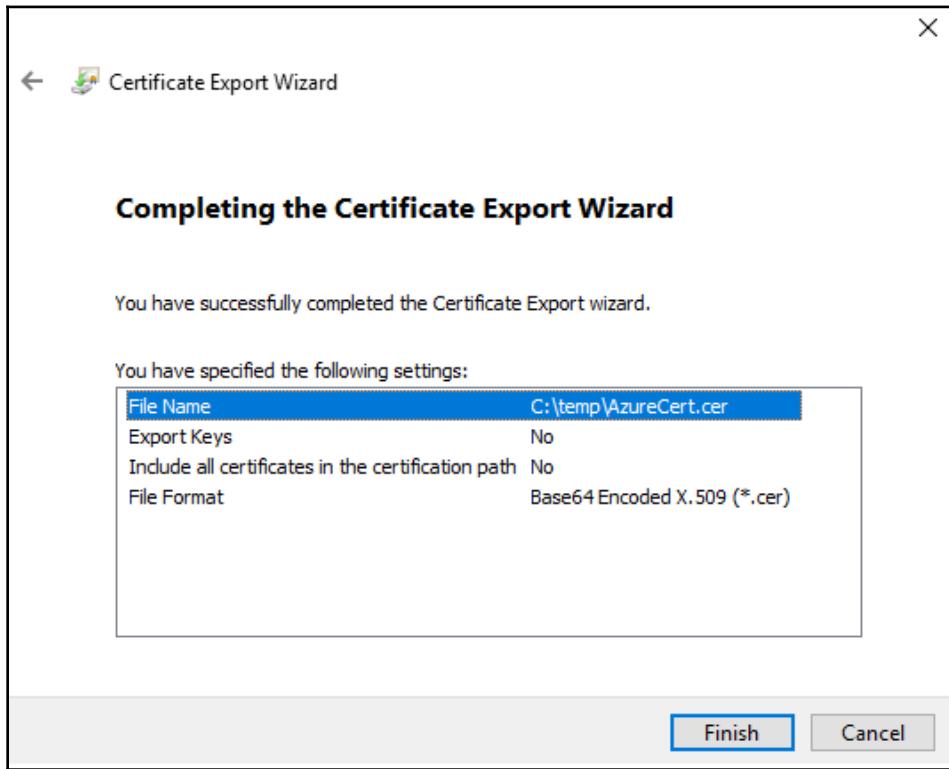


5. Select the format **Base-64 encoded X.509** and click **Next**:



6. Select the location where you want to save the certificate and click **Next**.

- Finally, we have the option to review all the information. After clicking **Finish**, the export will be complete:



Now, let's look at the steps to create a Point-2-Site connection.

How to do it...

To create a Point-2-Site connection, we need to do the following:

1. In Azure Portal, locate the virtual network gateway and **Point-to-site configuration**.
2. We need to define the **Address pool**. The address pool cannot overlap with the address pool of the VNet associated with the virtual network gateway:

The screenshot shows the Azure Portal interface for configuring a Point-to-site connection. On the left, there's a sidebar with navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Configuration, Connections, and Point-to-site configuration (which is highlighted). The main content area is titled "packt-vng-portal - Point-to-site configuration" and "Virtual network gateway". It contains several configuration fields:

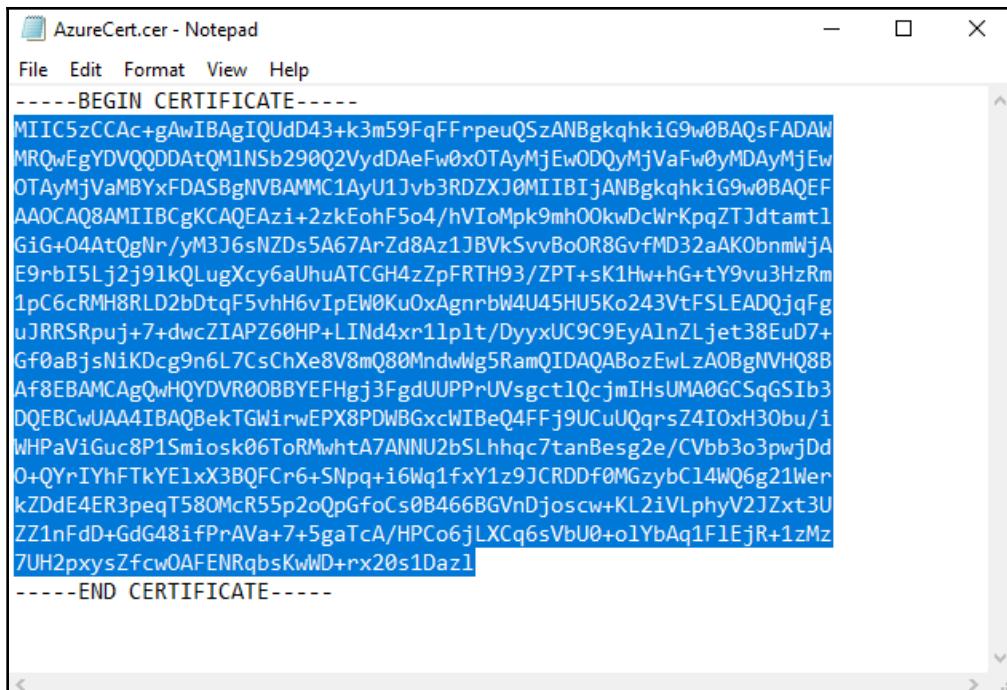
- Address pool:** A text input field containing "10.20.3.0/24" with a green checkmark icon.
- Tunnel type:** A dropdown menu set to "OpenVPN (SSL)".
- Authentication type:** A radio button group where "Azure certificate" is selected, while "RADIUS authentication" is unselected.
- Root certificates:** A section currently empty.

At the top right, there are "Save", "Discard", and "Download VPN client" buttons.

3. Next, we need to select **Tunnel type** from the list of predefined options. In this recipe, we'll select **OpenVPN (SSL)**, but any option is valid:

The screenshot shows the Azure portal interface for configuring a Point-to-site connection. On the left, there's a sidebar with icons for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Configuration, Connections, and Point-to-site configuration. The 'Point-to-site configuration' item is highlighted. The main area has a title 'packt-vng-portal - Point-to-site configuration' and a subtitle 'Virtual network gateway'. It includes a search bar, save, discard, and download VPN client buttons. A section for 'Address pool' shows '10.20.3.0/24' with a green checkmark. Below that, a 'Tunnel type' dropdown menu is open, listing several options: OpenVPN (SSL) (selected), OpenVPN (SSL), SSTP (SSL), IKEv2, IKEv2 and OpenVPN (SSL), and IKEv2 and SSTP (SSL). A 'Root certificates' section is also visible.

4. Locate the exported certificate (from the *Getting ready* section) and open it in Notepad (or any other text editor). Select the value of the certificate and copy this value:

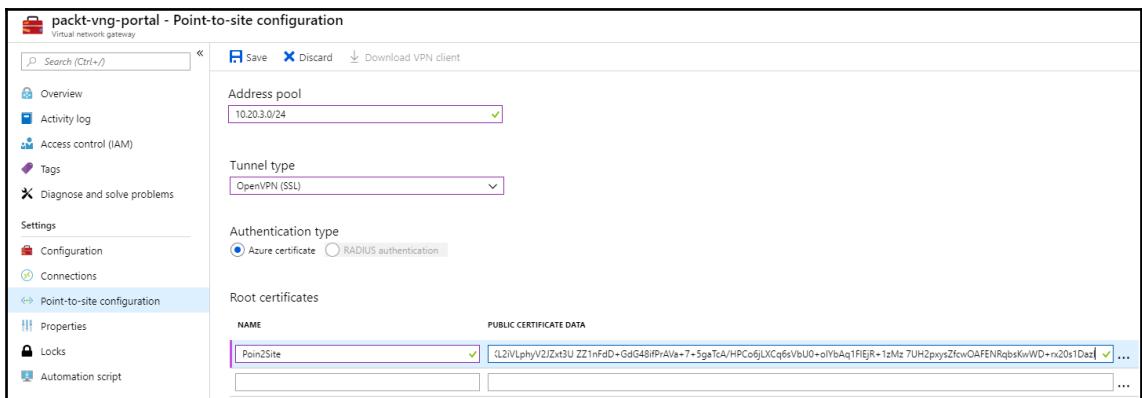


```

AzureCert.cer - Notepad
File Edit Format View Help
-----BEGIN CERTIFICATE-----
MIIC5zCCAc+gAwIBAgIQUD43+k3m59FqFFrpeuQSzANBgkqhkiG9w0BAQsFADAW
MRQwEgYDVQQDDAtQM1NSb290Q2VydDAeFw0xOTAyMjEwODQyMjVaFw0yMDAyMjEw
OTAyMjVaMBYxFDASBqNVBAMMC1AyI1Jvb3RDZXJ0MIIBIjANBgkqhkiG9w0BAQE
AAOCAQ8AMIIBCgKCAQEazi+2zkEohF5o4/hViOmpk9mh00kwDcWrKpqZTJdtamt1
GiG+04AtQgNr/yM3J6sNZDs5A67ArZd8Az1JBvkSvvBoOR8GvfMD32aAKObnmWjA
E9rb15Lj2j91kQlugXcy6aUhuATCGH4zZpFRTH93/ZPT+sK1Hw+hG+tY9vu3HzRm
1pC6cRMH8RLD2bDtqF5vhH6vIpEW0KuOxAgnrbW4U45HU5Ko243VtFSLEADQjqFg
uJRRSRpuj+7+dwcZIAPZ60HP+LINd4xr1lplt/DyyxUC9C9EyAlnZLjet38EuD7+
Gf0aBjsNiKDcg9n6L7CsChXe8V8mQ80Mndlwg5RamQIDAQABozEwLzAOBgNVHQ8B
Af8EBAMCAgQwHQYDVRO0BBYEFhgj3FgdUUPPrUVsgct1QcjmiHsUMA0GCSqGSIb3
DQEBCwUA4IBAQBekTGWirwEPX8PDWBGxclWIBeQ4FFj9UCuUQqrsZ4IOxH30bu/i
WHPaViGuc8P1Smiosk06ToRMwhtA7ANNU2bSLhhqc7tanBesg2e/CVbb3o3pwjDd
O+QYrIYhFTkYE1x3BQFCr6+SNpq+i6Wq1fxY1z9JCRDDf0MGzybC14WQ6g21Wer
kZDdE4ER3peqT580McR55p2oQpGfoCs0B466BGVnDjoscw+KL2iVLphyV2JZxt3U
ZZ1nFdD+GdG48ifPrAVa+7+5gaTcA/HPCo6jLXcq6sVbU0+o1YbAq1F1EjR+1zMz
7UH2pxysZfcwOAFENRqbsKwWD+rx20s1Daz1
-----END CERTIFICATE-----

```

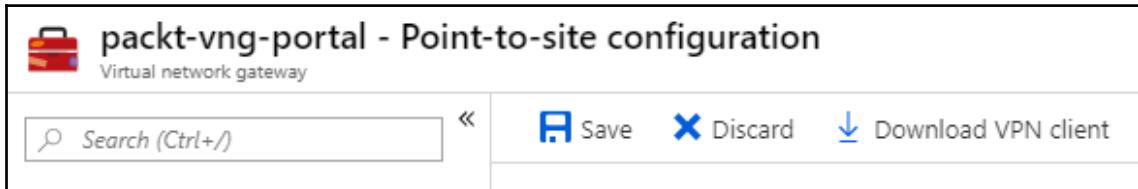
- In Azure Portal, we need to define the root certificate. Enter the name of the certificate and then paste value of the certificate (from the previous step) into the PUBLIC CERTIFICATE DATA field:



The screenshot shows the 'Point-to-site configuration' blade in the Azure Portal. The left sidebar lists 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Configuration', 'Connections', 'Point-to-site configuration' (which is selected), 'Properties', 'Locks', and 'Automation script'. The main area has tabs for 'Save', 'Discard', and 'Download VPN client'. Under 'Address pool', the value '10.20.3.0/24' is selected. Under 'Tunnel type', 'OpenVPN (SSL)' is chosen. Under 'Authentication type', 'Azure certificate' is selected. In the 'Root certificates' section, there is a table with one row:

NAME	PUBLIC CERTIFICATE DATA
Poin2Site	[REDACTED]

- After clicking **Save** for the Point-2-Site configuration, a new option will become available: **Download VPN client**. We can download the configuration and start using this connection:



Now, let's have a look at how it works.

How it works...

Point-2-Site allows us to access Azure VNet in a secure way. A Site-2-Site connection is restricted to access from our local network, but Point-2-Site allows us to connect from anywhere. Certificate-based certification is used, which uses the same certificate on both the server (Azure) and the client (VPN client) side to verify the connection and permit access. This allows us to access Azure VNet from anywhere and at any time. This type of connection is usually used for management and maintenance tasks, as it's an on-demand connection. If a constant connection is needed, you need to consider a Site-2-Site connection.

Creating a VNet-2-VNet connection

Similar to the need to connect Azure VNet to resources on a local network, we may have the need to connect to resources in another Azure VNet. In such cases, we can create a VNet-2-VNet connection that will allow us to use services and endpoints in another VNet. This process is very similar to creating a Site-2-Site connection; the difference is that we don't require a local network gateway; we use two virtual network gateways, one for each VNet.

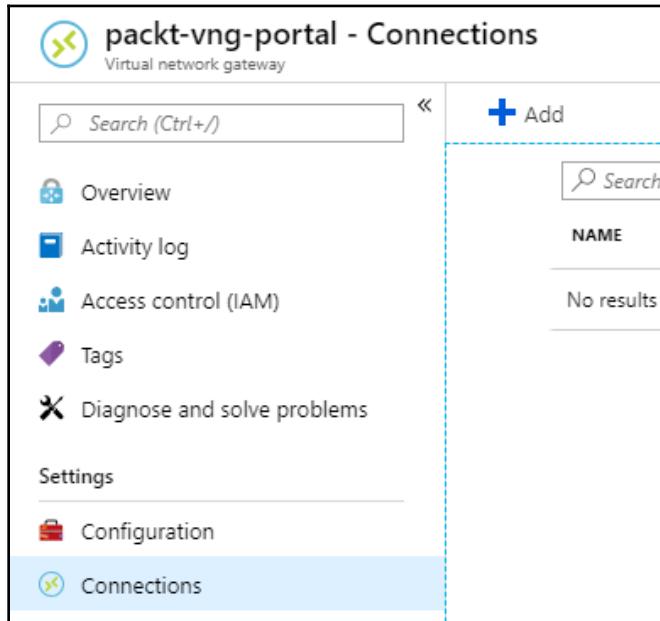
Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

How to do it...

To create a VNet-2-VNet connection, we must follow these steps:

1. In Azure portal, locate one of the virtual network gateways (associated with one of the VNets you are trying to connect to).
2. In the virtual network gateway blade, select **Connections** and select **Add** to add a new connection:



3. In a new blade, enter the name for a new connection and select **VNet-to-VNet** under **Connection type**:

 **Add connection** □ X
packt-vng-portal

* Name ✓

Connection type i ▾

* First virtual network gateway i lock
packt-vng-portal

* Second virtual network gateway i >
Choose another virtual network g...

* Shared key (PSK) i

Subscription i ▾

Resource group i lock
Packt-Networking-Portal

Create new

Location i ▾

4. The first virtual network gateway will be automatically highlighted. We need to select the second virtual network gateway:

The screenshot shows two overlapping windows in the Azure portal.

Left Window: Add connection

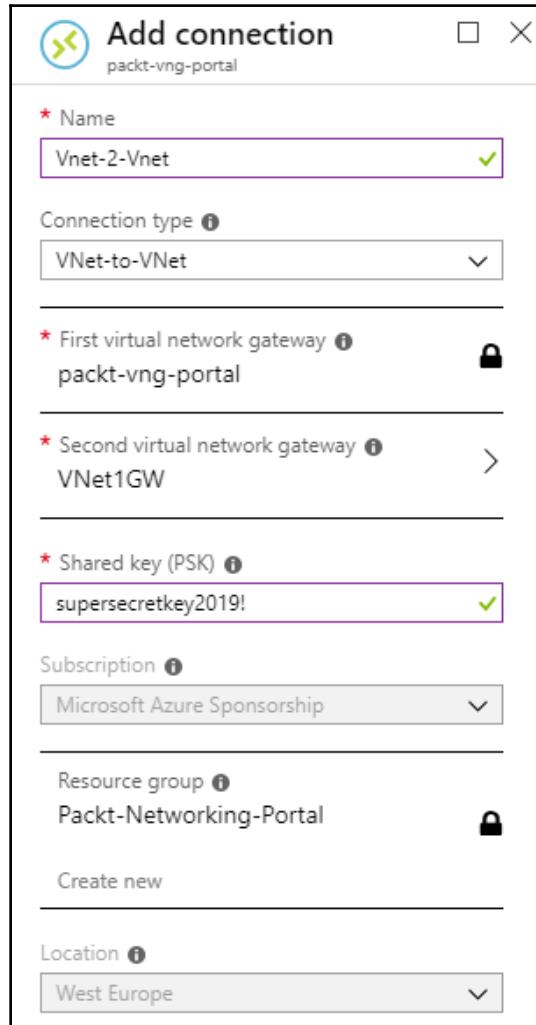
- Name:** Vnet-2-Vnet
- Connection type:** VNet-to-VNet
- First virtual network gateway:** packt-vng-portal (locked)
- Second virtual network gateway:** Choose another virtual network gateway... (link)
- Shared key (PSK):** (empty input field)
- Subscription:** Microsoft Azure Sponsorship
- Resource group:** Packt-Networking-Portal (locked)
- Create new:** (link)
- Location:** West Europe

Right Window: Choose virtual network g...

To use a virtual network with a connection, it must be associated to a virtual network gateway.
[Learn more](#)

- VNet1GW
Packt-Networking-Script (selected, highlighted with a dashed blue border)
- packt-vng-portal
Packt-Networking-Portal

5. We need to provide a shared key for our connection before we select **Create** and start the deployment. Note that **Subscription**, **Resource group**, and **Location** are locked and that the values for the first virtual network gateway are used here:



6. The deployment of VNet-2-VNet doesn't take long and should be done in a few minutes. However, it takes some time to establish connections, so you may see the status **Unknown** for up to 15 minutes before the status changes to **Connected**:

NAME	STATUS	CONNECTION TYPE	PEER
Vnet-2-Vnet	Unknown	VNet-to-VNet	packt-vng-portal

Now, let's have a look at how it works.

How it works...

A VNet-2-VNet connection works very similar to a Site-2-Site connection. The difference is that Azure uses a local network gateway for information on the local network. In this case, we don't need this information; we use two virtual network gateways to connect. Each virtual network gateway provides network information for the VNet that it's associated with. The result is secure, encrypted VPN connections between two Azure VNets that can be used to establish connections between Azure resources on both VNets.

Connecting VNets using network peering

Another option to connect two Azure VNets is to use **network peering**. This approach doesn't require the use of a virtual network gateway, so it's more economical to use if the only requirement is to establish a connection between Azure VNets. Network peering uses the Microsoft backbone infrastructure to establish a connection between two VNets, and traffic is routed through private IP addresses only. However, this traffic is not encrypted; it's private traffic that stays on the Microsoft network, similar to what happens to traffic on the same Azure VNet.

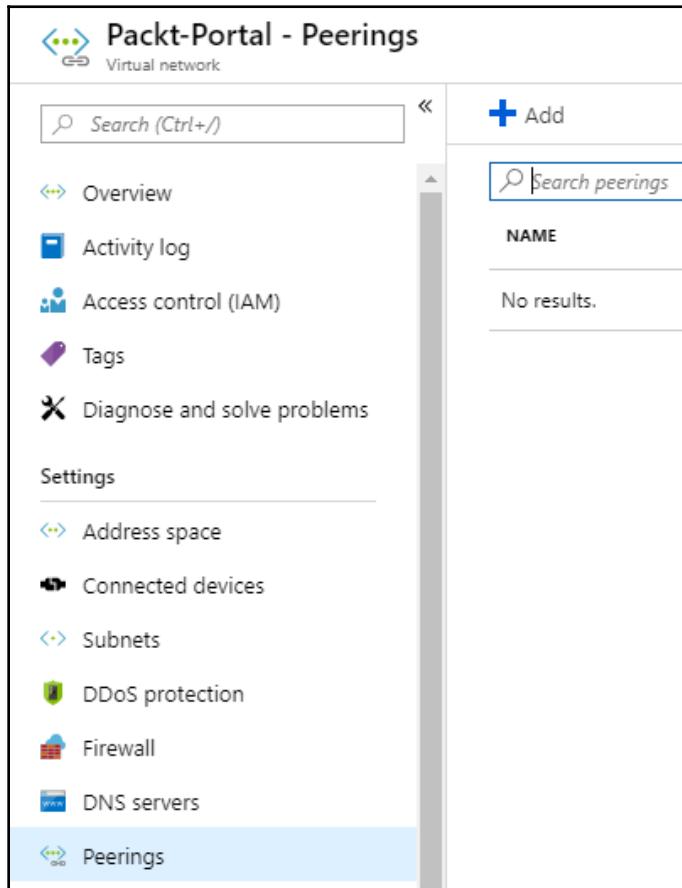
Getting ready

Before you start, open the browser and go to the Azure portal: <https://portal.azure.com>.

How to do it...

To create network peering, we must do the follow these steps:

1. In the Azure portal, locate one of the VNets that you want to connect to.
2. In the VNet blade, select the **Peerings** option, and select **Add** to add a new connection:



3. In the new blade, we must enter the name of the connection, select **Virtual network deployment model (Resource manager or Classic)**, and select the VNet we are connecting to. This information can be provided by either providing a resource ID or by selecting a subscription and a VNet from the drop-down menu. There is some additional configuration that is optional and that allows us better traffic control:

Add peering

Packt-Portal

*** Name**
Peering ✓

Peer details

Virtual network deployment model i
 Resource manager Classic

I know my resource ID i

*** Subscription i**
Microsoft Azure Sponsorship

*** Virtual network**
Packt-Script (Packt-Networking-Script)

Configuration

Allow virtual network access i

Allow forwarded traffic i

Allow gateway transit i

Use remote gateways i

i Virtual network 'Packt-Portal' has a gateway; peerings created from this virtual network can't enable 'use remote gateways'.

- After a connection is created, we can see the information and the status for peering. We can also change the **Configuration** options at any time:

The screenshot shows the 'Peering' configuration page in the Packt-Portal. At the top, there are 'Save', 'Discard', and 'Delete' buttons. The main area contains the following details:

- Name:** Peering
- Peering status:** Initiated
- Provisioning state:** Succeeded
- Peer details:**
 - Address space:** 10.11.0.0/16
 - Remote Vnet Id:** /subscriptions/cb638267-a366-463c-bfe5-7a49311c27a8/resourceGroups/Packt-Networki... (with a copy icon)
 - Virtual network:** Packt-Script
- Configuration:**
 - Allow virtual network access:** Enabled (radio button selected)
 - Allow forwarded traffic
 - Allow gateway transit
 - Use remote gateways

Now, let's have a look at how it works.

How it works...

Network peering allows us to establish a connection between two Azure VNets in the same Azure tenant. Peering uses a Microsoft backbone network to route private traffic between resources on the same network, using private IP addresses only. There is no need for virtual network gateways (that create additional cost), as a virtual "remote gateway" is created to establish a connection. The downside of this approach is that the same VNet can't use peering and a virtual network gateway at the same time. If there is a need to connect VNet to both the local network and another VNet, we must use a different approach and use a virtual network gateway that will allow us to create a Site-2-Site connection with a local network and a VNet-2-VNet connection with another VNet.

7

DNS and Routing

Azure DNS allows us to host **Domain Name System (DNS)** domains in Azure. When using Azure DNS, we use Microsoft infrastructure for the name resolution, which results in fast and reliable DNS queries. Microsoft Azure DNS infrastructure uses a vast number of servers to provide great reliability and availability of service. Using Anycast networking, each DNS query is answered by the closest DNS server available to provide a quick reply.

We will cover the following recipes in this chapter:

- Creating an Azure DNS zone
- Creating a new record set and a record in Azure DNS
- Creating a route table
- Changing the route table
- Associating the route table to a subnet
- Dissociating the route table from the subnet
- Creating a new route
- Changing a route
- Deleting a route

Technical requirements

For this chapter, the following is required:

- Azure subscription

Creating an Azure DNS zone

To start using Azure DNS, we must first create a DNS zone. A DNS zone holds a DNS record for a specific domain, and it can hold records for a single domain at the time. A DNS zone will hold DNS records for this domain and possible subdomains. DNS name servers are set up to reply to any query on a registered domain, and point to a destination.

Getting ready

Before you start, open your browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to create a new Azure DNS zone with the Azure portal, we must follow these steps:

1. In the Azure portal, select **Create a resource** and choose **DNS Zone** under **Networking** services (or search for **DNS Zone** in the search bar).
2. In a new blade, we must enter information for **Name**, **Subscription**, and **Resource group**. If we select the existing resource group, the location will be same as the one for the resource group selected. **Name** must be a **Fully Qualified Domain Name (FQDN)**:

Create DNS zone

[Basics](#) [Tags](#) [Review + create](#)

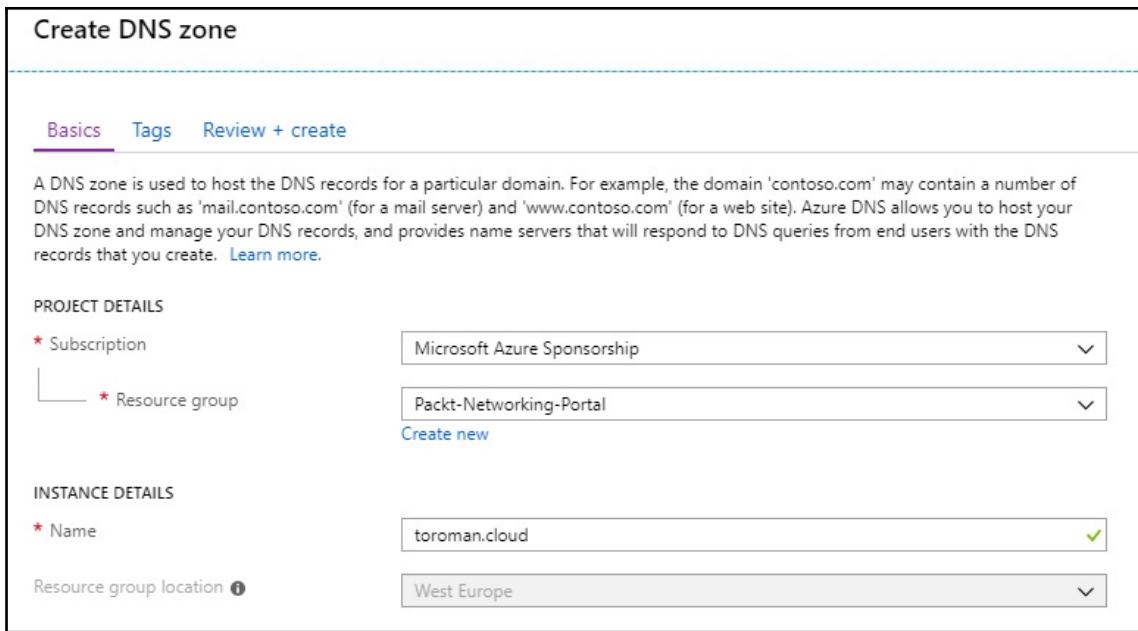
A DNS zone is used to host the DNS records for a particular domain. For example, the domain 'contoso.com' may contain a number of DNS records such as 'mail.contoso.com' (for a mail server) and 'www.contoso.com' (for a web site). Azure DNS allows you to host your DNS zone and manage your DNS records, and provides name servers that will respond to DNS queries from end users with the DNS records that you create. [Learn more.](#)

PROJECT DETAILS

* Subscription Microsoft Azure Sponsorship ▾
└─ * Resource group Packt-Networking-Portal ▾
Create new

INSTANCE DETAILS

* Name toroman.cloud ▾
Resource group location ⓘ West Europe ▾



How it works...

A DNS zone is required to start using Azure DNS. A new DNS zone is required for each domain we want to host with Azure DNS, as a single DNS zone can hold information for a single domain. After we create a DNS zone, we can add records, records sets, and route tables to a domain hosted with Azure DNS. Using these, we can route traffic and define destinations using an FQDN for Azure resources (and other resources as well). We'll show how to create and manage these in the coming recipes in this chapter.

Creating a new record set and record in Azure DNS

After creating a DNS zone, we define what domain we're going to hold records for. A DNS zone is created for a "root" domain defined with an FQDN. We can add additional subdomains and add records and record sets to hold information on other resources on the same domain.

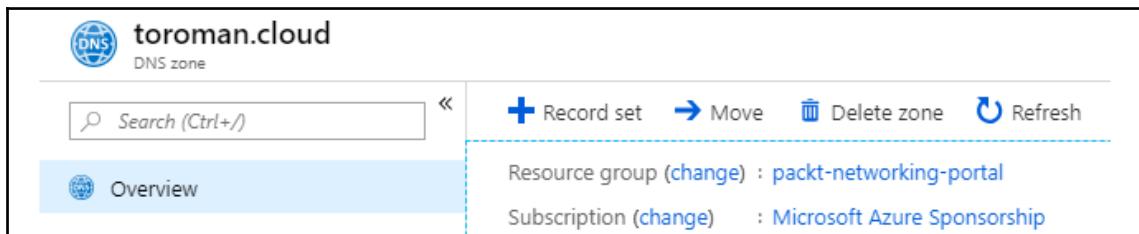
Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

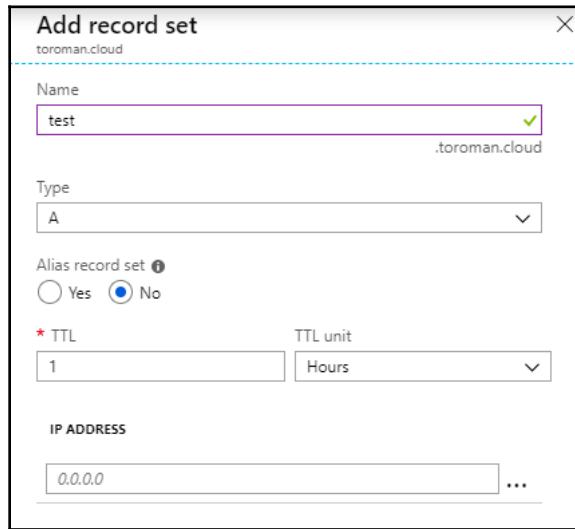
How to do it...

In order to add a new record to the DNS zone, we must use the following steps:

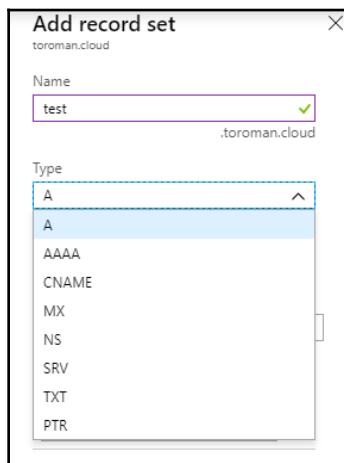
1. In the Azure portal, locate the DNS zone.
2. In the overview, select the option for adding a record set:



3. A new blade will open. Enter the name of the subdomain for which you want to add a record to:



4. We need to select the type of record we want to add. The options are **A**, **AAAA**, **CNAME**, **MX**, **NS**, **SRV**, **TXT**, and **PTR**. The most common record type is **A**, so let's select that one:



- After we select the record type, we need to select whether this is an alias, and the **TTL (Time To Live)** option. Finally, we add a record destination. This depends on the record type, and in the case of record A, it's going to be an IP address:

Add record set

toroman.cloud

Name

test .toroman.cloud

Type

A

Alias record set ⓘ

Yes No

* TTL

1 Hours

IP ADDRESS

10.10.0.8 ...
0.0.0.0 ...

- Adding a single entry to our record creates a new record set and a new record. We can add more records to the record set by adding additional IP addresses (in this case).

How it works...

A DNS record set holds information on the subdomain in the domain hosted with the DNS zone. In this case, the domain would be `toroman.cloud`, and the subdomain would be `test`. This forms an FQDN, `test.toroman.cloud`, and the record points this domain to the IP address we defined. The record set can hold multiple records for a single subdomain, usually used for redundancy and availability.

Creating a route table

Azure routes network traffic in subnets by default. But in some cases, we want to use custom traffic routes to define where and how traffic flows. In this case, we use **route tables**. A route table defines the next hop for our traffic and determines where the network traffic needs to go.

Getting ready

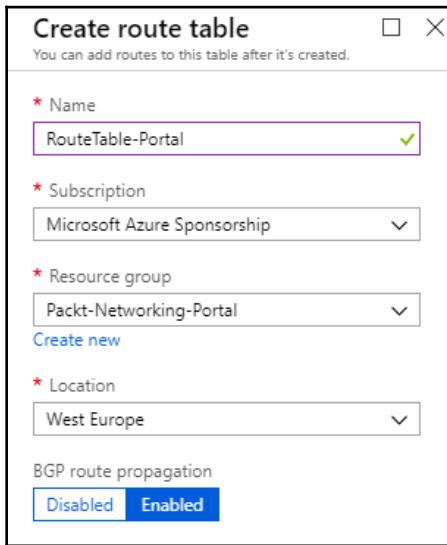
Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to add a new record to the DNS zone, we must use the following steps:

1. In the Azure portal, select **Create a resource** and choose **Route Table** under the **Networking** services (or search for `route table` in the search bar).

2. In the new blade, we need to provide the name of the route table and select the subscription, resource group, and location. Optionally, we can define whether we want to enable or disable **BGP (Border Gateway Protocol)** route propagation (enabled by default):



How it works...

Network routing in Azure VNet is done automatically, but we can use custom routing with route tables. Route tables use rules and subnet associations to define traffic flow in Azure VNet. When a new route table is created, no configuration is created, only an empty resource. After the resource is created, we need to define rules and subnets in order to use a route table for the traffic flow. We will show in coming recipes in this chapter how we create and apply rules in route tables.

Changing the route table

As mentioned in the previous recipe, creating a new route table will result in an empty resource. Once a resource is created, we can change the settings as needed. Before we configure the routes and subnets associated with the route table, the only setting we can change is the BGP route propagation. We may change other settings after creation as well.

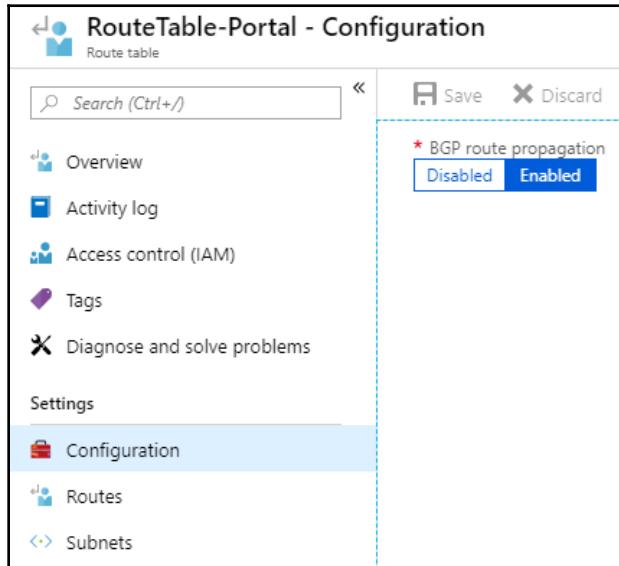
Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to change the route table, we must do the following:

1. In the Azure portal, locate **Route table**.
2. Under **Settings**, we may change the **BGP route propagation** settings in the **Configuration** blade. Under **Settings**, we may change **Routes** and **Subnets**, if they were previously configured:



How it works...

Under the settings of the route table, we can disable or enable BGP route propagation at any time. This option, if disabled, prevents on-premises routes from being propagated via BGP to the network interfaces in a virtual network subnet. Under the settings, we can create, delete, or change routes and subnets. These options will be addressed in the coming recipes in this chapter.

Associating a route table to a subnet

Once a route table is created, it doesn't do anything until it's properly configured. There are two things we need to address: which resources are affected and how. To define which resources are affected, we must make an association between a subnet and a route table.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

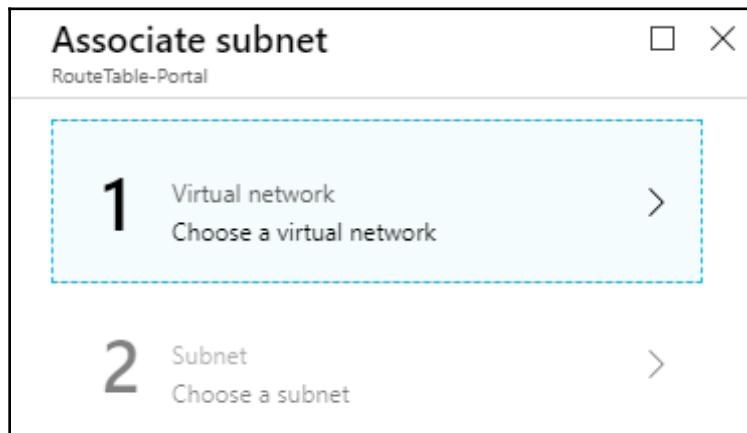
How to do it...

In order to associate a subnet with a route table, we must do the following:

1. In the Azure portal, locate **Route table**.
2. Under **Settings** in the route table, select the **Subnets** option. In the **Subnets** blade, select the **Associate** option to create a new association:

The screenshot shows the Azure portal interface for managing a route table named 'RouteTable-Portal'. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, and others. Below that is a 'Settings' section with Configuration, Routes, and Subnets. The 'Subnets' link is currently selected and highlighted in blue. The main content area is titled 'RouteTable-Portal - Subnets' and shows a 'Route table' section. On the right, there's a 'Associate' blade with a search bar 'Search subnets'. A table header is shown with columns: NAME, ADDRESS RANGE, VIRTUAL NETWORK, and SECURITY GROUP. Below the header, it says 'No results.'

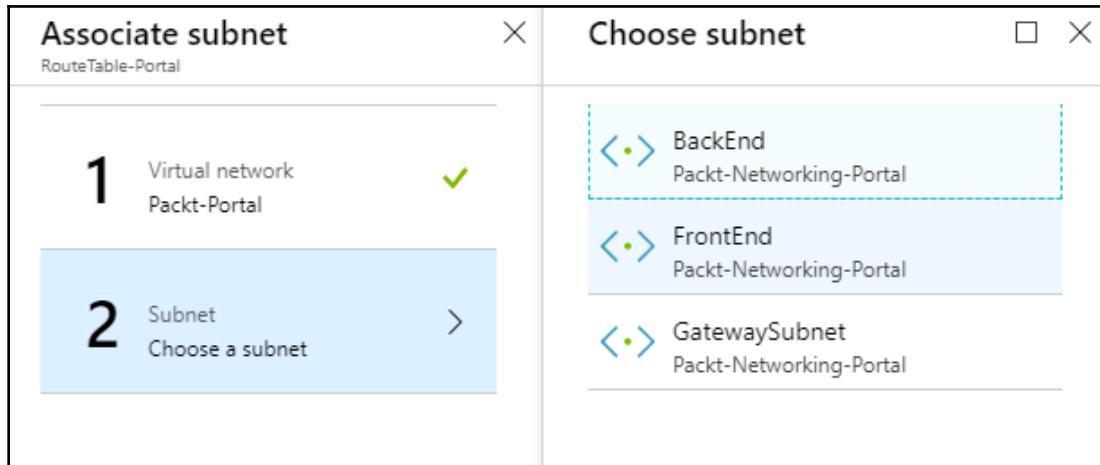
3. A new blade will open. There are two options available to select a virtual network and the subnet we want to associate:



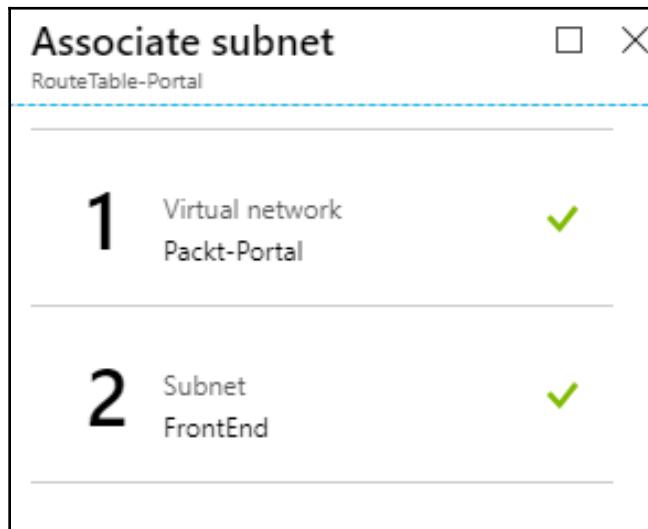
4. First, we must select **Virtual network**. Selecting this option will list all the available virtual networks. Select the one you want to associate from this list:

A screenshot of the 'Associate subnet' blade. The left side shows the steps 'Virtual network' (selected) and 'Subnet'. The right side is titled 'Resource' and lists five virtual networks: Nagios-vnet (westeurope), Packt-Portal (westeurope), Packt-Script (westeurope), RedVSBlue-vnet (westeurope), and SQL-WFG-vnet (westeurope). The first item, 'Nagios-vnet westeurope', is highlighted with a blue dashed border.

5. After a virtual network is selected, we can proceed to select a subnet. The subnet option will list all the subnets from the virtual network we selected in the previous step. Choose the subnet you want to associate from this list:



6. After both options are selected, we may proceed and create an association:



- After a subnet has been associated, it will appear in a list of subnets under the route table:

The screenshot shows the Azure portal interface for managing subnets under a route table. On the left, a sidebar lists navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Configuration, Routes, and Subnets. The 'Subnets' option is currently selected and highlighted in blue. The main content area is titled 'Associate' and contains a table with one row. The table has columns for NAME, ADDRESS RANGE, and VIRTUAL NETWORK. The single row shows 'FrontEnd' as the name, '10.10.0.0/25' as the address range, and 'Packt-Portal' as the virtual network.

NAME	ADDRESS RANGE	VIRTUAL NETWORK
FrontEnd	10.10.0.0/25	Packt-Portal

How it works...

The route table, to be effective, must have two parts defined: what and how. What is affected by the route table we define with a subnet association. This is only one part of the configuration, as just associating a subnet to a route table will do nothing. We must create rules that will apply to this association. We'll explain the rules in the following recipes in this chapter.

Dissociating a route table from the subnet

After we create an association and rules, rules will apply to all resources on an associated subnet. If we want rules to no longer apply to a specific subnet, we can remove the association.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

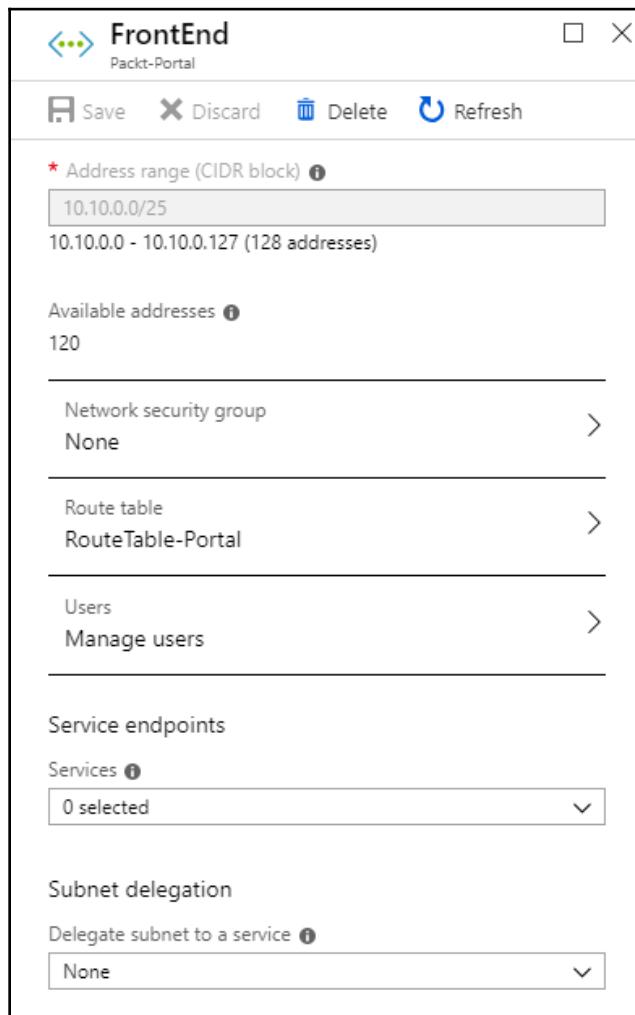
In order to remove the association between the subnet and the route table, we must do the following:

1. In the Azure portal, locate **Route table**.
2. Under **Settings**, select the **Subnets** option, and select the subnet you want to remove:

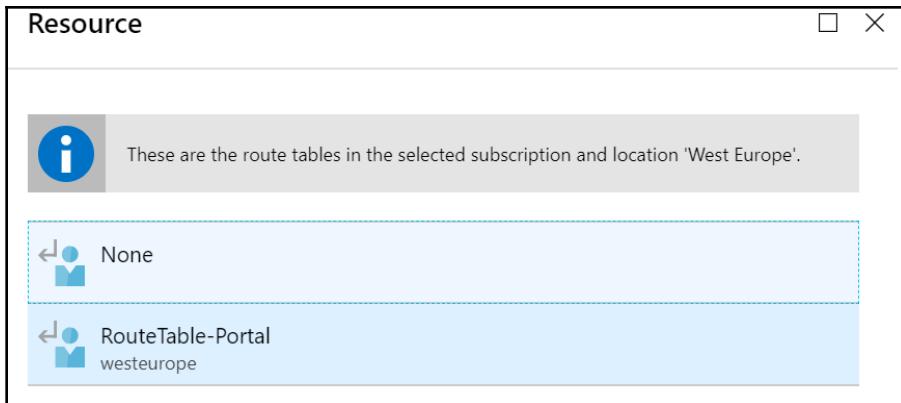
The screenshot shows the Azure portal interface for managing a route table. The left sidebar has a 'RouteTable-Portal - Subnets' title and a 'Route table' section. Below are navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Configuration and Routes), and Subnets (which is highlighted). The main content area is titled 'Associate' and shows a table with one row. The table has columns: NAME, ADDRESS RANGE, and VIRTUAL NETWORK. The row contains: FrontEnd, 10.10.0.0/25, and Packt-Portal.

NAME	ADDRESS RANGE	VIRTUAL NETWORK
FrontEnd	10.10.0.0/25	Packt-Portal

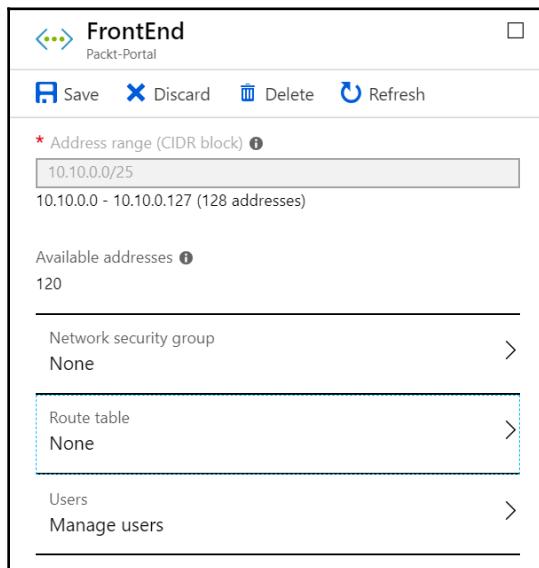
3. The subnet configuration blade will open. Select the route table option. Note that this actually opens a subnet configuration. It's a common mistake to confuse this blade with the association and to choose the **Delete** option. This will not only remove the association but remove the subnet altogether:



4. It will show a list of the available route tables for a specific subnet. Choose **None**:



5. After selecting **None**, click the **Save** button to apply the new settings. The route table association is removed from the subnet:



How it works...

At some point, we may have created rules in a route table that apply to multiple subnets. If we no longer want to apply one or more rules to a specific subnet, we can remove the association. Once the association is removed, the rules will no longer apply to the subnet removed. All rules will apply to all the associated subnets. If we need a single rule not to apply to a specific subnet, we must remove the association.

Creating a new route

After we create a route table and the associated subnets, there is still a piece missing. We defined the route table that will be affected with subnet association, and we're missing the part that defines how. We define how associated subnets are affected with rules called **routes**. Routes define traffic routes, telling us where specific traffic needs to go. If the default route for specific traffic is the internet, we can change this and reroute the traffic to a specific IP or subnet.

Getting ready

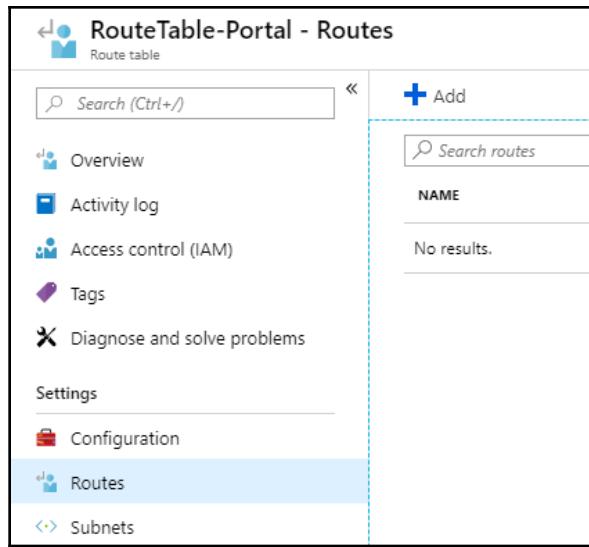
Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to create a new route, we must do the following:

1. In the Azure portal, locate **Route table**.

2. In the route table, under **Settings**, select **Routes**. Select **Add** to add a new route:



3. In the new blade, we need to define the **Route name**, **Address prefix** (in CIDR format) for the destination IP address range, and select **Next hop type**. Next hop types are **Virtual network gateway**, **Virtual network**, **Internet**, **Virtual appliance**, and **None**:

The screenshot shows the 'Add route' dialog box. It has fields for 'Route name' (Route1) and 'Address prefix' (10.10.0.0/25). Under 'Next hop type', 'Internet' is selected from a dropdown menu that also includes 'Virtual network gateway', 'Virtual network', 'Virtual appliance', and 'None'.

4. The last option, **Next hop address** is active only when a virtual appliance is used. In that case, we need to provide the virtual appliance IP address in this field, and all traffic will go through the virtual appliance:

The screenshot shows a 'RouteTable-Portal' interface titled 'Add route'. It contains the following fields:

- Route name**: A text input field containing 'Route1' with a green checkmark.
- Address prefix**: A text input field containing '10.10.0.0/25' with a green checkmark.
- Next hop type**: A dropdown menu set to 'Internet'.
- Next hop address**: An empty text input field.

How it works...

The route defines the traffic flow. All traffic from the associated subnet will follow the route defined by these rules. If we define that traffic will go to the internet, all traffic will go outside the network to an IP address range defined with an IP address prefix. If we choose that traffic goes to a virtual network, it will go to a subnet defined by the IP address prefix. If that virtual network gateway is used, all traffic will go through the virtual network gateway and reach its connection on the other side, either another virtual network or our local network. The virtual appliance option will send all traffic to the virtual appliance, which then, with its own set of rules, defines where the traffic goes next.

Changing a route

Route requirements may change over time. In such cases, we can either remove the route or edit it, depending on our needs. If the route needs to be adjusted, we can select the option to change the route and apply the new traffic flow at any time.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

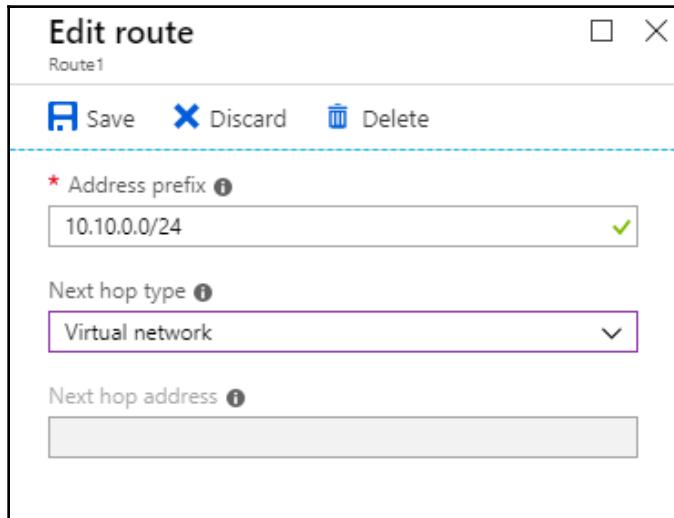
In order to change the existing route, we need to do the following:

1. In the Azure portal, locate **Route table**.
2. Under **Settings**, select **Routes** and select the route you want to change from the list of available routes:

The screenshot shows the Azure portal interface for a 'RouteTable-Portal' resource. On the left, there's a sidebar with navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Configuration, and Routes. The 'Routes' link is highlighted with a blue background. The main content area has a header 'RouteTable-Portal - Routes' with a 'Route table' icon. Below the header is a search bar labeled 'Search (Ctrl+ /)'. To the right of the search bar is a blue 'Add' button with a plus sign. A dashed blue line separates the sidebar from the main content area. The main content area contains a table titled 'Search routes' with columns 'NAME' and 'ADDRESS PREFIX'. One row is visible: 'Route1' and '10.10.0.0/24'.

NAME	ADDRESS PREFIX
Route1	10.10.0.0/24

3. A new blade will open. We can change **Address prefix** (for destination IP range) and **Next hop type**. If the next hop type is a virtual appliance, an option for the next hop address will be available:



How it works...

The requirements for the route may change over time. We can change the route and adjust it to suit the new requirements as needed. The most common scenarios are that the traffic needs to reach a specific service and the service IP changes. For example, we may need to route all traffic through a virtual appliance, and the IP address of virtual appliance changes. We may change the route in the route table to reflect this change and force the traffic flow through the virtual appliance. Another example is when traffic needs to reach our local network through a virtual network gateway. The destination IP address range may change over time, and we need to reflect these changes in the route once again.

Deleting a route

As we already mentioned, route requirements may change over time. In some cases, the rules are no longer applicable and we must remove them. In such cases, changing the route will not complete the task, and we need to remove the route completely. This task may be completed by deleting the route.

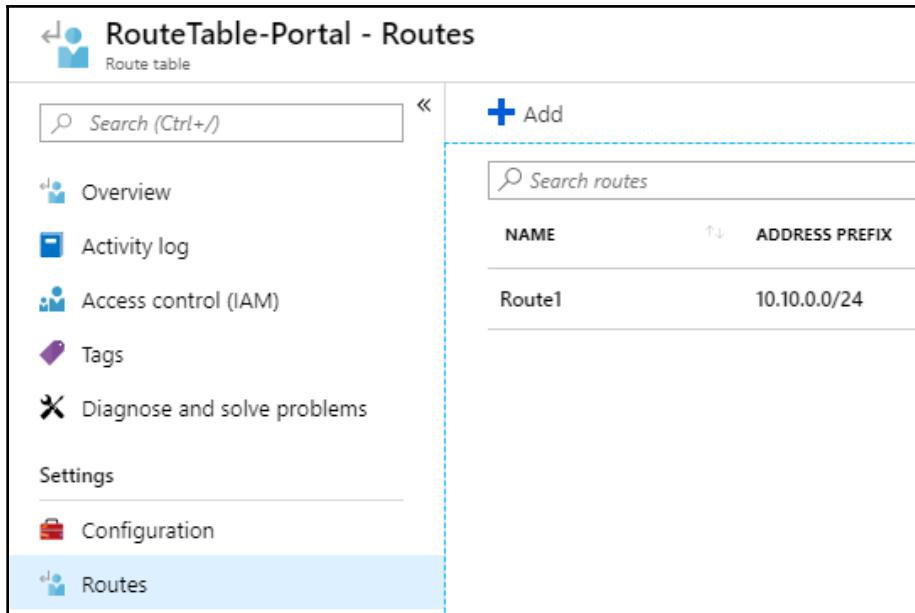
Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to delete the route, we must do the following:

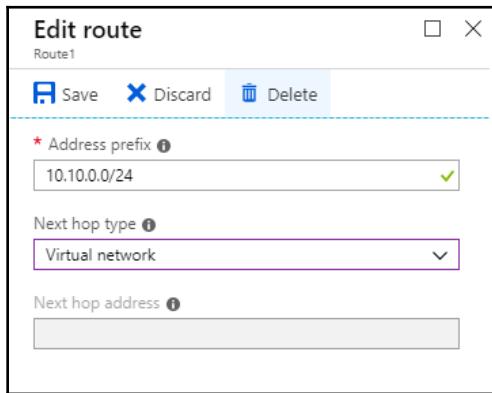
1. In the Azure portal, locate **Route table**.
2. Under **Settings**, select **Routes** and select the route you want to delete:



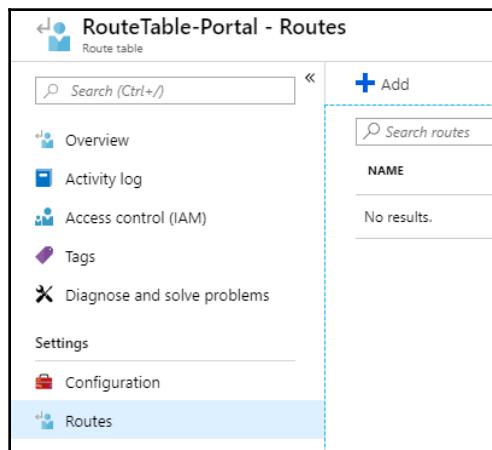
The screenshot shows the Azure portal interface for managing routes in a route table. The left sidebar lists navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Configuration, and Routes. The 'Routes' option is currently selected, highlighted with a blue background. The main content area is titled 'RouteTable-Portal - Routes' and shows a table of routes. The table has columns for NAME and ADDRESS PREFIX. One route is listed: Route1 with an address prefix of 10.10.0.0/24.

NAME	ADDRESS PREFIX
Route1	10.10.0.0/24

3. A new blade will open. Select the delete option and confirm your action:



4. After this action has been confirmed, you will return to the previous blade and the deleted route will no longer be listed:



How it works...

As requirements change, we need to address the new requirements in route tables. We can either edit routes or remove them to meet the new requirements. When multiple routes are used in a single route table, one of the routes may become obsolete or even block new requirements. In such cases, we may want to delete a route to resolve any issues.

8 Load Balancers

Load balancers are used to support scaling and high availability for applications and services. A load balancer is primarily composed of two components—a frontend and a backend. Requests coming to the frontend of a load balancer are distributed to the backend, where we place multiple instances of a service. This can be used for performance-related reasons, where we would like to distribute traffic equally between endpoints in the backend, or for high availability, where multiple instances of services are used to increase the chance that at least one endpoint will be available at all times.

We will cover the following recipes in this chapter:

- Creating an internal load balancer
- Creating a public load balancer
- Creating a backend pool
- Creating health probes
- Creating load balancer rules
- Creating inbound NAT rules

Technical requirements

For this chapter, an Azure subscription is required.

Creating an internal load balancer

Microsoft Azure supports two types of load balancers—**internal** and **external**. An internal load balancer is assigned a private IP address (from the address range of subnets in VNet) for a frontend IP address, and it targets private IP addresses of our services (usually, Azure VM) in the backend. An internal load balancer is usually used by services that are not internet-facing and are accessed only from within our VNet.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to create a new internal load balancer with the Azure portal, we must use the following steps:

1. In the Azure portal, select **Create a resource** and choose **Load Balancer** under the **Networking** services (or search for **Load Balancer** in the search bar).
2. In the new blade, we must select a subscription and resource group where a load balancer is created. Then, we must provide information for the **Name**, **Region**, **Type**, and **SKU**. In this case, we select the **Internal** type to deploy an internal load balancer. Finally, we must select the **Virtual network** and the **Subnet** that the load balancer will be associated with, along with information of the IP address assignment, which can be **Dynamic** or **Static**:

Create load balancer

* Subscription: Microsoft Azure Sponsorship

* Resource group: Packt-Networking-Portal
Create new

INSTANCE DETAILS

* Name: Packt-LoadBalancer-Internal

* Region: West Europe

* Type: Internal (selected)

* SKU: Basic (selected)

CONFIGURE VIRTUAL NETWORK.

* Virtual network: Packt-Portal

* Subnet: FrontEnd (10.10.0.0/25)
Manage subnet configuration

* IP address assignment: Dynamic (selected)

Review + create **Previous** **Next : Tags >** Download a template for automation

3. After all the information is entered, we select the **Review + create** option to validate the information and start the deployment of the load balancer.

How it works...

An internal load balancer is assigned a private IP address and all requests coming to the frontend of an internal load balancer must come to a private address, limiting the traffic coming to the load balancer to be from within the VNet associated with the load balancer. Traffic can come from other networks (other VNets or local networks) if there is some kind of **virtual private network (VPN)** in place. The traffic coming to the frontend of the internal load balancer will be distributed across the endpoints in the backend of the load balancer. Internal load balancers are usually used for services that are not placed in a **demilitarized zone (DMZ)** (and therefore not accessible over the internet) but rather in middle- or back-tier services in a multi-tier application architecture.

Creating a public load balancer

The second type of load balancer in Microsoft Azure is a **public load balancer**. The main difference is that a public load balancer is assigned a public IP address in the frontend and all requests are coming over the internet. The requests are then distributed to the endpoints in the backend.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to create a new public load balancer with the Azure portal, we must follow these steps:

1. In the Azure portal, select **Create a resource** and choose **Load Balancer** under the **Networking** services (or search for **Load Balancer** in the search bar).
2. In the new blade, we must select a subscription and a resource group where a load balancer is created. Then, we must provide information for **Name**, **Region**, **Type**, and **SKU**. In this case, we select **Public** to deploy the public load balancer:

The screenshot shows the 'Create load balancer' blade. It has two main sections: 'PROJECT DETAILS' and 'INSTANCE DETAILS'. In 'PROJECT DETAILS', 'Subscription' is set to 'Microsoft Azure Sponsorship' and 'Resource group' is set to 'Packt-Networking-Portal'. In 'INSTANCE DETAILS', 'Name' is 'Packt-LoadBalancer-Public', 'Region' is 'West Europe', 'Type' is 'Public' (selected), and 'SKU' is 'Basic' (selected). The 'Public' type selection is highlighted with a green checkmark.

3. Selecting **Public** as the load balancer type will slightly change the blade. We no longer have the option to select VNet and subnet, like for internal load balancer. Instead, we can choose options for public IP address (new or existing), public IP address SKU, IP address assignment, and whether we want to use IPv6. Note that the public IP address SKU depends directly on the load balancer SKU, so the SKU selected for the load balancer will transfer automatically to the IP address:

PUBLIC IP ADDRESS

* Public IP address Create new Use existing

* Public IP address name ✓

Public IP address SKU Basic

* Assignment Dynamic Static

Add a public IPv6 address No Yes

- After all the information is entered, we select the **Review + create** option to validate the information and start the deployment of the load balancer.

How it works...

The public load balancer is assigned a public IP address at the frontend. Therefore, all requests coming to the public load balancer will be coming over the internet, targeting the load balancer's public IP address. Requests are then distributed to endpoints in the backend of the load balancer. What's interesting is that the public load balancer does not target the public IP addresses in the backend, but private IP addresses. For example, let's say that we have one public load balancer with two Azure VMs in the backend. Traffic coming to the public IP address of the load balancer will then be distributed to VMs, but will target the VMs' private IP addresses.

Public load balancers are used for public-facing services, most commonly for web servers.

Creating a backend pool

After the load balancer is created, either internally or publicly, we must apply further configuration in order to start using it. During the creation process, we define the frontend of the load balancer and know where traffic needs to go to reach the load balancer. But in order to define where that traffic needs to go after reaching the load balancer, we must define a backend pool.

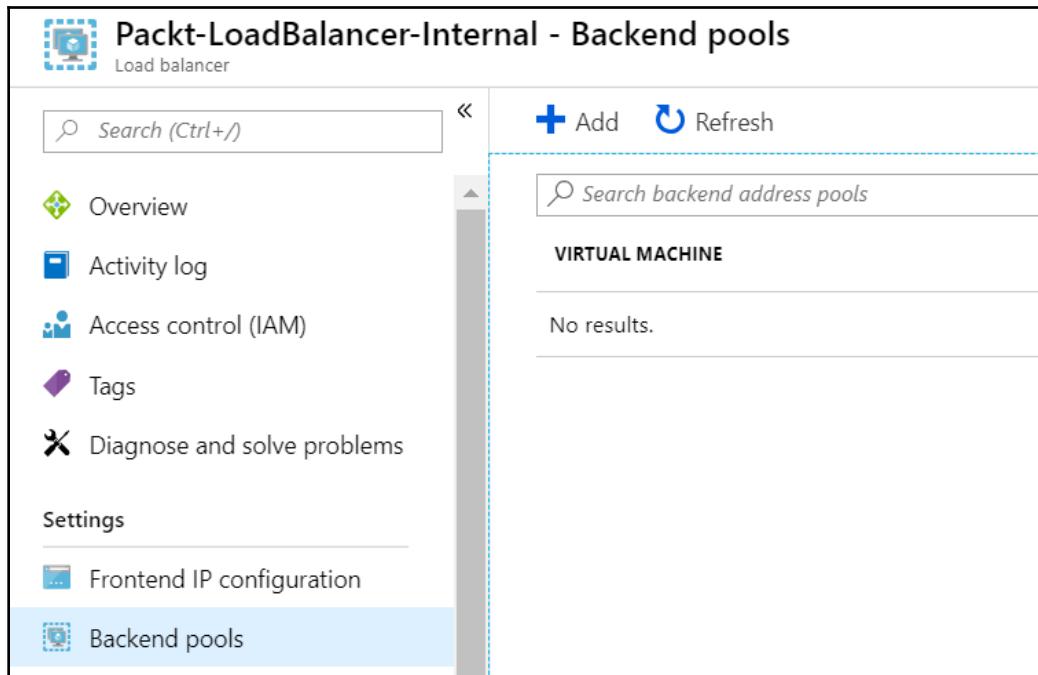
Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to create the backend pool, we must do the following:

1. In the Azure portal, locate the previously created load balancer (either internal or public).
2. In the **Load balancer** blade, under **Settings**, select **Backend pools**. Select **Add** to add the new backend pool:



3. In the new blade, we must provide a name and what the load balancer is associated to. Association can be created for a single VM, availability set, or virtual machine scale set. I recommend using **Availability set**. Based on this selection, you will be offered a list of available resources in the selected category to choose from:

Add backend pool

Packt-LoadBalancer-Internal

* Name
BackendPool ✓

IP version i
IPv4

Associated to i
Availability set ✓

Availability set i
Web
number of virtual machines: 2, resource group: Packt-Networking-Portal-VMs ✓

4. After an association is created, you will be offered further options to select a network IP configuration. In the case of selecting a virtual machine, you will be offered to select the **network interface (NIC)** (as any VM can have more than one), or in the case of the availability set, you can select between VMs in the selected set. In this recipe, I selected an availability set and two VMs in that set:

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: PacktWeb1 ✓

Network IP configuration: packtweb1107/ipconfig1 (10.10.0.7) ✓

* Target virtual machine i
PacktWeb2
size: Standard_D2s_v3, network interfaces: 1, resource group: PACKT-NETWORKING-P... ✓

* Network IP configuration i
ipconfig1 (10.10.0.9) ✓

+ Add a target network IP configuration

5. After configuration is entered, it takes a few minutes to create an association. After that, the associated resources will show up in the backend pool list:

How it works...

The two main components of any load balancer are the frontend and the backend. The frontend defines the endpoint of the load balancer, and the backend defines where the traffic needs to go after reaching the load balancer. As the frontend information is created along with the load balancer, we must define the backend. Then traffic will be evenly distributed across endpoints in the backend. The available options for the backend pool are **virtual machines**, **availability sets**, and **virtual machines scale sets**.

See also

More information on virtual machines, availability sets, and virtual machines scale sets is available in my book, *Hands-On Cloud Administration in Azure*, published by Packt at <https://www.packtpub.com/virtualization-and-cloud/hands-cloud-administration-azure>.

Creating health probes

After the frontend and the backend of the load balancer are defined, traffic is evenly distributed among endpoints in the backend. But *what if one of the endpoints is unavailable?* In that case, some of the requests will fail until we detect the issue or even fail indefinitely, in case issue is not detected. The load balancer would send a request to all the defined endpoints in the backend pool and the request would fail when directed to an unavailable server.

This is why we introduce the next two components in the load balancer—**health probes** and **rules**. These components are used to detect issues and describe what to do when issues are detected.

Health probes constantly monitor all endpoints defined in the backend pool and detect whether any of them become unavailable.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

To create a new health probe in the load balancer, we must do the following:

1. In the Azure portal, locate the previously created load balancer (either internal or public).

2. In the **Load balancer** blade, under **Settings**, select **Health probes**. Select **+ Add** to add a new health probe:

The screenshot shows the 'Health probes' blade for a load balancer named 'Packt-LoadBalancer-Internal'. The left sidebar lists various management options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Frontend IP configuration, Backend pools, and Health probes. The 'Health probes' option is currently selected and highlighted in blue. The main content area is titled 'Health probes' and contains a search bar labeled 'Search probes', a 'NAME' input field, and a message stating 'No results.'

3. In the new blade, we need to provide information about the health probe's name, the protocol we want to use, and the port, interval, and unhealthy threshold, as shown in the following screenshot:

Add health probe

Packt-LoadBalancer-Internal

* Name
HTTPS ✓

IP version
IPv4

Protocol i
TCP ▼

* Port i
443 ✓

* Interval i
5 seconds

* Unhealthy threshold i
2 consecutive failures

OK

How it works...

After we define the health probe, it will be used to monitor the endpoints in the backend pool. We define the protocol and the port as useful information that will provide information regarding whether the service we are using is available or not. Monitoring the state of the server would not be enough, as it could be misleading. For example, the server could be running and available, but the IIS or SQL server that we use might be down. So, the protocol and the port are going to detect change in a service that we are interested in, and not only whether the server is running. The interval defines how often a check is performed, and the unhealthy threshold defines after how many consecutive fails the endpoint is declared unavailable.

Creating load balancer rules

The last piece of puzzle, when speaking of Azure load balancers, is the **rule**. Rules finally tie all things together and define which health probe (there can be more than one) will monitor which backend pool (more than one can be available). Furthermore, rules enable port mapping from the frontend of a load balancer to the backend pool, defining how ports relate and how incoming traffic is forwarded to backend.

Getting ready

Before you start, open your browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to create a load balancer rule, we must do the following:

1. In the Azure portal, locate the previously created load balancer (either internal or public).
2. In the **Load balancer** blade, under **Settings**, select **Load balancing rules**. Select **Add** to add the load balancing rule:

The screenshot shows the Azure portal interface for managing load balancing rules. The main title is "Packt-LoadBalancer-Internal - Load balancing rules". A sidebar on the left lists navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Frontend IP configuration, Backend pools, Health probes, and Load balancing rules. The "Load balancing rules" option is highlighted with a blue background. The main content area has a search bar at the top right labeled "Search (Ctrl+/)". Below it is a "Add" button with a plus sign. A search bar for "Search load balancing rules" is present. A table header row includes "NAME" and a sorting icon. The message "No results." is displayed below the table.

3. In the new blade, we must provide information for the name and the IP version we are going to use, which frontend IP address we are going to use (as a load balancer can have more than one), the protocol, and the port mapping (traffic from the incoming port will be forwarded to the backend port):

Add load balancing rule

Packt-LoadBalancer-Internal

* Name
Rule1 ✓

* IP Version
 IPv4 IPv6

* Frontend IP address ⓘ
10.10.0.4 (LoadBalancerFrontEnd) ▾

Protocol
 TCP UDP

* Port
443 ✓

* Backend port ⓘ
443 ✓

4. In the second part of the blade, we need to provide information for **Backend pool**, **Health probe**, **Sessions persistence**, **Idle timeout (minutes)** settings, and whether we want to use a floating IP:

Backend pool ⓘ

BackendPool (2 virtual machines)

Health probe ⓘ

HTTPS (TCP:443)

Session persistence ⓘ

Client IP

Idle timeout (minutes) ⓘ

4

Floating IP (direct server return) ⓘ

Disabled Enabled

How it works...

The load balancer rule is the final piece that ties all the components together. We define which frontend IP address is used and to which backend the pool traffic will be forwarded to. The health probe is assigned to monitor the endpoints in the backend pool and to keep track of whether there are any unresponsive endpoints. We also create port mapping that will determine which protocol and port the load balancer will listen on, and, when the traffic arrives, where this traffic will be forwarded.

Creating inbound Network Address Translation (NAT) rules

Inbound NAT rules are an optional setting in the Azure load balancer. These rules essentially create another port mapping from frontend to backend, forwarding traffic over a specific port on the frontend to a specific port in the backend. The difference between inbound NAT rules and port mapping in load balancer rules is that inbound NAT rules apply to direct forwarding to a VM, whereas load balancer rules forward traffic to a backend pool.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to create new inbound NAT rule, we must do the following:

1. In the Azure portal, locate the previously created load balancer (either internal or public).
2. In the **Load balancer** blade, under **Settings**, select **Inbound NAT rules**.
Select **Add** to add a new inbound NAT rule:

The screenshot shows the AWS Lambda console interface. At the top, it displays the name of the load balancer: "Packt-LoadBalancer-Internal - Inbound NAT rules". Below the title, there is a "Load balancer" icon. On the left side, there is a sidebar with several navigation items:

- Search bar: "Search (Ctrl+ /)"
- Access control (IAM)
- Tags
- Diagnose and solve problems

Below the sidebar, under "Settings", are the following options:

- Frontend IP configuration
- Backend pools
- Health probes
- Load balancing rules
- Inbound NAT rules (this item is highlighted with a blue background)

The main content area on the right is titled "Add" and contains a search bar: "Search inbound NAT rules". There is also a "NAME" field which is currently empty. A message at the bottom of the list states: "No results."

3. In the new blade, we must provide **Name**, select **Frontend IP address**, **Service**, **Protocol**, and **Port**:

Add inbound NAT rule

Packt-LoadBalancer-Internal

NATRule1

Frontend IP address

LoadBalancerFrontEnd (10.10.0.4)

IP Version

IPv4

Service

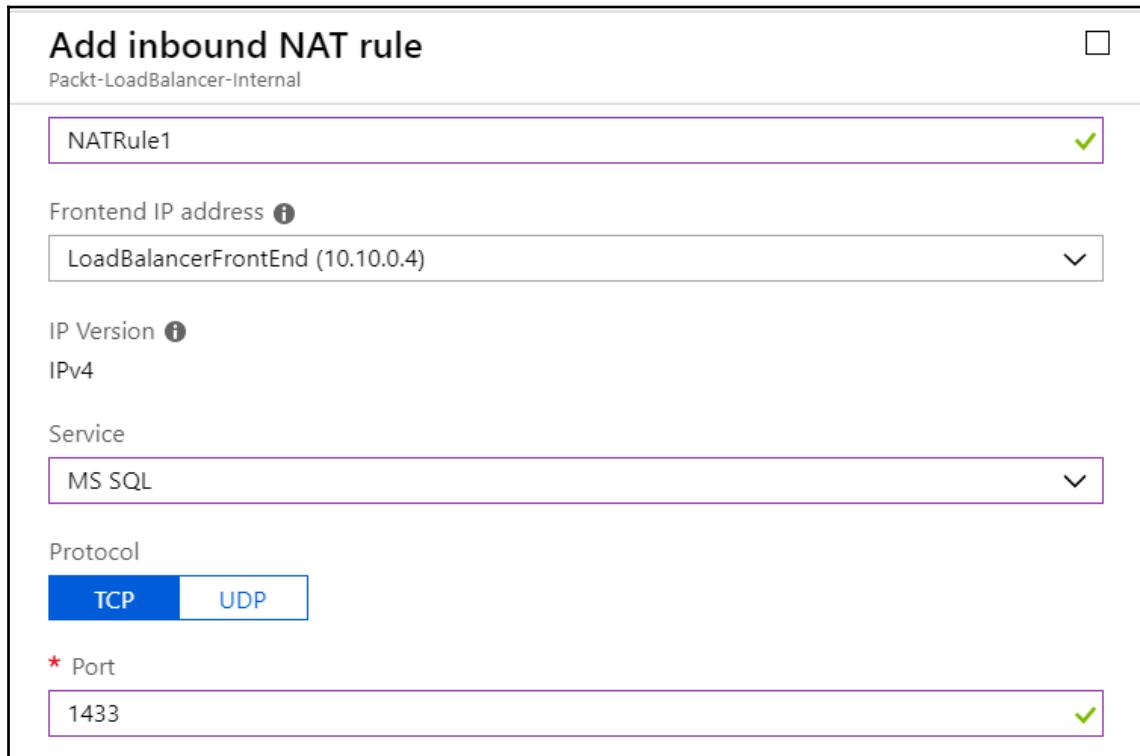
MS SQL

Protocol

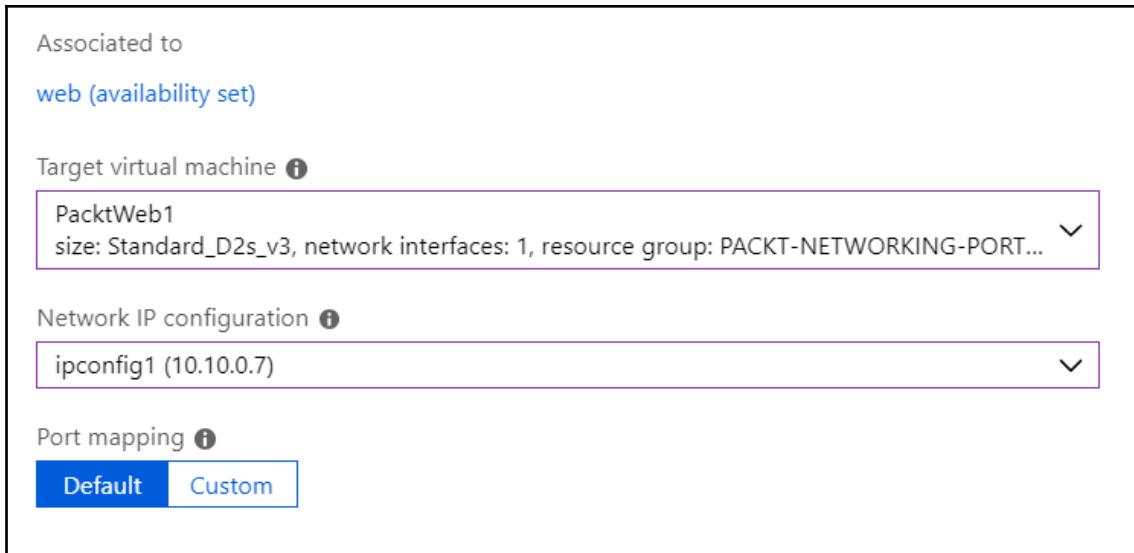
TCP UDP

* Port

1433



4. In the next set of options, you need to set an association and select one VM from that association. Note that from an associated availability set, you can select only one VM. Finally, you can select default or custom port mapping:



How it works...

The inbound NAT rules create port mapping similar to the port mapping created with load balancer rules. The load balancer rule creates additional settings such as the health probe or session persistence. Inbound NAT rules exclude these settings and create unconditional mapping from the frontend to the backend. Another difference is that the inbound NAT rule forwards traffic directly to a single VM, whereas a load balancer rule forwards traffic to the backend. With the inbound NAT rule, forwarded traffic will always reach the single server in the backend, whereas the load balancer will forward the traffic to the backend pool and it will randomly end in any of the servers in the backend pool.

9

Traffic Manager

An Azure load balancer is limited to providing high availability and scalability only to Azure virtual machines. Also, single load balancer is limited to VMs in a single Azure region. If we want to provide the same thing to other Azure services that are globally distributed , we must introduce a new component—**Azure Traffic Manager**. Azure Traffic Manager is DNS-based and provides the ability to distribute traffic over services and spread traffic across Azure regions. But Traffic Manager is not limited to Azure services only; we can add external endpoints as well.

We will cover the following recipes in this chapter:

- Creating a new Traffic Manager profile
- Adding an endpoint
- Configuring distributed traffic
- Configuring traffic based on priority
- Configuring traffic based on geographical location
- Managing an endpoint
- Managing profiles
- Configuring Traffic Manager with load balancers

Technical requirements

For this chapter, an Azure subscription is required.

Creating a new Traffic Manager profile

Traffic Manager provides load balancing to services but traffic is routed and directed using DNS entries. The frontend is a **Fully Qualified Domain Name (FQDN)** assigned during creation, and all traffic coming to Traffic Manager is distributed to endpoints in the backend. In this recipe, we'll create new Traffic Manager.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to create a new Traffic Manager profile, we must do the following:

1. In the Azure portal, select **Create a resource** and choose **Traffic Manager** under the **Networking** services (or search for **Traffic Manager** in the search bar).

2. In the new blade, we must provide information for the **Name**, **Routing method**, **Subscription** and **Resource group** fields:

Create Traffic Manager pr... □ ×

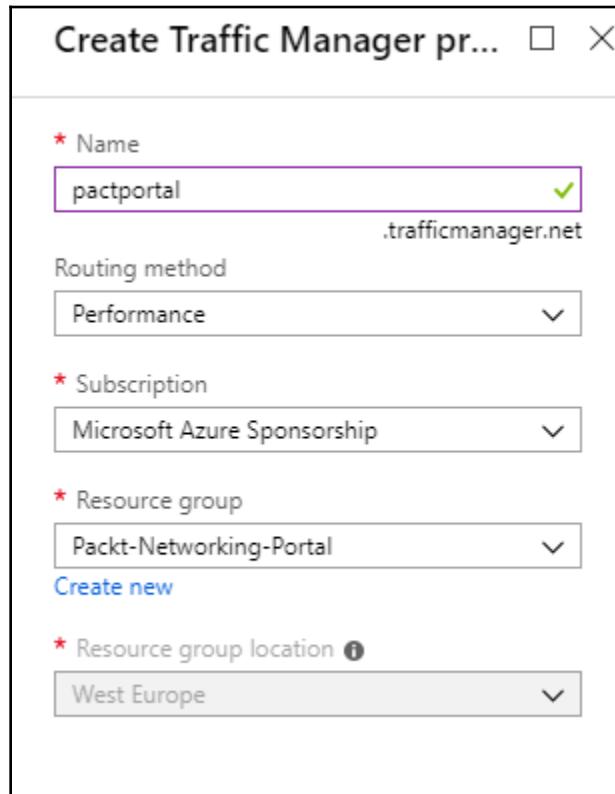
* Name
pactportal .trafficmanager.net

Routing method
Performance

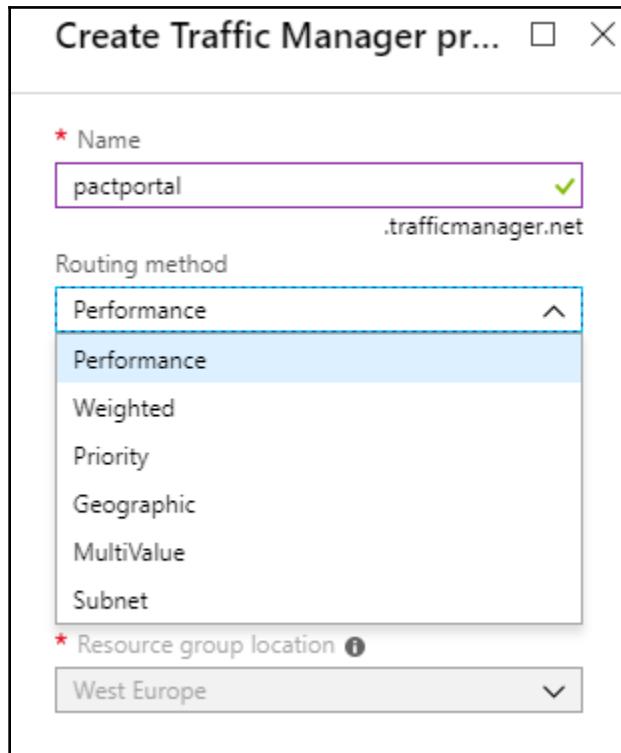
* Subscription
Microsoft Azure Sponsorship

* Resource group
Packt-Networking-Portal [Create new](#)

* Resource group location ⓘ
West Europe

The screenshot shows the 'Create Traffic Manager' blade. It has four main sections with red asterisks indicating they are required: 'Name' (containing 'pactportal' with a green checkmark and '.trafficmanager.net' suffix), 'Subscription' (containing 'Microsoft Azure Sponsorship'), 'Resource group' (containing 'Packt-Networking-Portal' with a link to 'Create new'), and 'Resource group location' (containing 'West Europe'). Each section has a dropdown arrow icon to its right.

3. Note that under the routing methods, we have multiple options—**Performance**, **Weighted**, **Priority**, **Geographic**, **MultiValue**, and **Subnet**. In this recipe, let's leave it at the default option (**Performance**), but we will cover the rest in other recipes in this chapter:



How it works...

Traffic Manager is assigned a public endpoint that must be an FQDN. All traffic arriving to that endpoint will be distributed to endpoints in the backend, using the routing method defined. The default routing method is performance. The performance method will distribute traffic based on the best possible performance available. For example, if we have more than one backend endpoint in the same region, traffic will be spread evenly. If the endpoints are located across different regions, Traffic Manager will direct traffic to the endpoint closest to the incoming traffic in terms of geographical location and minimum network latency.

Adding an endpoint

After a Traffic Manager profile is created, we have the frontend endpoint and routing method defined. But we still need to define where the traffic needs to go after it's reached Traffic Manager. We need to add endpoints to the backend and define where the traffic is directed. In this recipe, we'll add new endpoint to Traffic Manager.

Getting ready

Before you start, open the browser and go to the Azure portal on <https://portal.azure.com>.

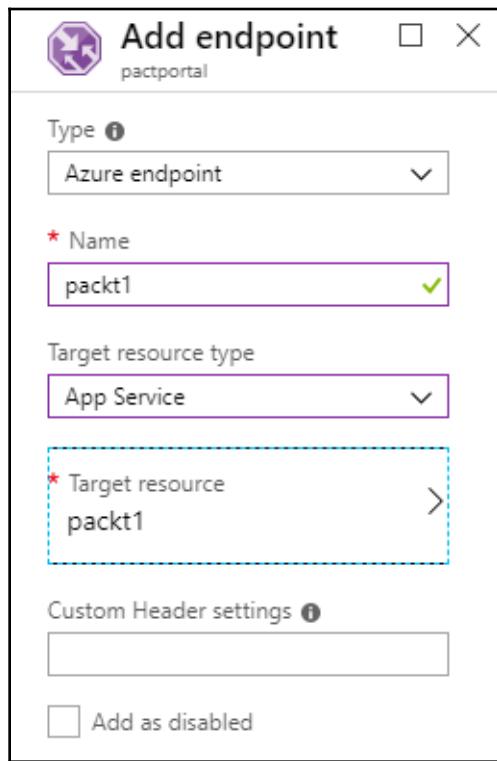
How to do it...

In order to add endpoints to Traffic Manager, we must do the following:

1. In the Azure portal, locate the previously created Traffic Manager profile.
2. In the **Traffic Manager** blade, under the settings, select **Endpoints**. Select **Add** to add a new endpoint:

The screenshot shows the 'pactportal - Endpoints' blade for a Traffic Manager profile. On the left, there's a sidebar with icons for Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. Below that is a 'Settings' section with Configuration, Real user measurements, Traffic view, and Endpoints. The 'Endpoints' item is highlighted with a blue background. The main area has a search bar at the top labeled 'Search endpoints'. Below it is a table with columns 'NAME' and 'STATUS'. A message 'No results.' is displayed. At the top right of the main area are 'Add' and 'Refresh' buttons.

3. In the new blade, we need to provide information for the type (of endpoint we are adding) and the name. Based on the type, we can select certain target resource types, and based on the target resource type selection, we can select resources that fit the target resource type selected:



4. Adding a single endpoint will only work as a redirection from one FQDN to another. We need to repeat the process at least one more time and add at least one more endpoint:

The screenshot shows the 'Add endpoint' dialog box. At the top, there's a logo for 'pactportal' and a close button (X). The main area has a title 'Add endpoint' and a sub-section 'pactportal'.
The 'Type' dropdown is set to 'Azure endpoint'.
The 'Name' field is filled with 'Packt2', which has a green checkmark next to it.
The 'Target resource type' dropdown is set to 'App Service'.
Below that, the 'Target resource' section shows 'packt2' with a right-pointing arrow.
There's a section for 'Custom Header settings' with an empty input field.
At the bottom, there's a checkbox labeled 'Add as disabled'.

5. All the added endpoints will appear in the list of endpoints under the **Endpoint** section in the **Settings** option of Traffic Manager:

The screenshot shows the 'Endpoints' page in the Azure portal for the 'pactportal' Traffic Manager profile. The left sidebar has navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Configuration, Real user measurements, Traffic view, and Endpoints (which is selected and highlighted in blue).
The main area has a search bar, an 'Add' button, and a 'Refresh' button. A table lists the endpoints:

NAME	STATUS	MONITOR STATUS	TYPE	LOCATION
packt1	Enabled	Online	Azure endpoint	West Europe
Packt2	Enabled	Online	Azure endpoint	West US

How it works...

Incoming requests reach Traffic Manager by hitting the frontend endpoint of Traffic Manager. Based on rules (mainly the routing method), traffic is then forwarded to the backend endpoints. The load balancer works by forwarding traffic to private IP addresses. On the other hand, Traffic Manager uses public endpoints in the backend. The supported endpoint types are Azure, external, and nested. Based on an endpoint type, we can add Azure or external endpoints. Endpoints can be either (public) FQDN or public IP address. Nested endpoints allow us to add other Traffic Manager profiles to the backend of Traffic Manager.

Configuring distributed traffic

The default routing method for Traffic Manager is performance. The performance method will distribute traffic based on the best possible performance available. This method only takes full effect if we have multiple instances of a service in multiple regions. As this often isn't the case, other methods are available, such as distributed traffic or the weighted routing method. In this recipe, we'll configure Traffic Manager to work in distributed mode.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to set the distributed traffic, we must do the following:

1. In the Azure portal, locate the previously created Traffic Manager profile.
2. Under **Settings**, select the **Configuration** option. Here, we have multiple options that we can change, such as **DNS time to live (TTL)**, protocols, or failover settings:

pactportal - Configuration
Traffic Manager profile

«

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Configuration (selected)

Real user measurements

Traffic view

Endpoints

Properties

Locks

Automation script

Monitoring

Alerts

Metrics

Support + troubleshooting

Resource health

New support request

Routing method i

Performance

*** DNS time to live (TTL)** i

Endpoint monitor settings i

Protocol

*** Port**

*** Path**

Custom Header settings i

Expected Status Code Ranges (default: 200) i

Fast endpoint failover settings

Probing interval i

*** Tolerated number of failures** i

*** Probe timeout** i

3. Change **Routing method** to **Weighted** as shown in the following screenshot.
Furthermore, we can set up weight settings if needed:

The screenshot shows the Azure Traffic Manager configuration interface for a profile named "pactportal". The left sidebar lists various management options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Configuration (which is selected and highlighted in blue), Real user measurements, Traffic view, and Endpoints. The main content area displays the configuration details. At the top right are "Save" and "Discard" buttons. Below them, the "Routing method" is set to "Weighted". Other configuration parameters shown include "DNS time to live (TTL)" set to 60, "Protocol" set to HTTP, "Port" set to 80, and "Path" set to /. A vertical dashed line separates the sidebar from the main content area.

How it works...

The weighted routing method will distribute traffic evenly across all endpoints in the backend. We can further set weight settings to give an advantage to a certain endpoint and say that some endpoints will receive a bigger or smaller percentage of the traffic. This method is usually used when we have multiple instances of an application in the same region, or for scaling out to increase performance.

Configuring traffic based on priority

Another routing method available is **Priority**. Priority, as its name suggests, gives priority to some endpoints, while some endpoints are kept as backup. Backup endpoints are only used if endpoints with priority become unavailable. In this recipe, we'll configure Traffic Manager to route traffic based on priority.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to set the routing method to Priority, we must do the following:

1. In the Azure portal, locate the previously created Traffic Manager profile.
2. Under **Settings**, select the **Configuration** option.

3. Change **Routing method** to **Priority**, as shown in the following screenshot:

The screenshot shows the 'pactportal - Configuration' page for a Traffic Manager profile. On the left, there's a sidebar with icons for Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. Below that is a 'Settings' section with 'Configuration' selected, highlighted by a blue bar. Other options include Real user measurements, Traffic view, and Endpoints. On the right, the main configuration area has a 'Save' button with a dashed border and a 'Discard' button. Under 'Routing method', 'Priority' is selected. Other settings shown include 'DNS time to live (TTL)' set to 60, 'Protocol' set to HTTP, 'Port' set to 80, and 'Path' set to '/'. The entire screenshot is enclosed in a light gray border.

How it works...

Priority sets a priority order for endpoints. All traffic will first go to the endpoints with the highest priority. Other endpoints (with lower priority) are backed up, and traffic is routed to these endpoints only when higher-priority endpoints become unavailable. The default priority order is the order of adding endpoints to Traffic Manager, where the first-added endpoint becomes the one with the highest priority and the last-added endpoint becomes the endpoint with the least priority. Priority can be changed under the endpoint settings.

Configuring traffic based on geographical location

Geographical location is another routing method in Traffic Manager. This method is based on network latency and directs a request based on the geographical location of the origin and the endpoint. When a request comes to Traffic Manager, based on the origin of the request, it's routed to the nearest endpoint in terms of region. This way, it provides the least network latency possible. In this recipe, we'll configure Traffic Manager to route traffic based on geographical location.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to set the routing method to the geographical location, we must do the following:

1. In the Azure portal, locate the previously created Traffic Manager profile
2. Under **Settings**, select the **Configuration** option

3. Change **Routing method** to **Geographic**, as shown in the following screenshot:

The screenshot shows the Azure Traffic Manager configuration interface. On the left, there's a sidebar with icons for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Configuration (which is selected and highlighted in blue), Real user measurements, Traffic view, and Endpoints. The main panel has a search bar at the top. Below it, there are two tabs: 'Routing method' (set to 'Geographic') and 'Endpoint monitor settings'. Under 'Endpoint monitor settings', there are fields for Protocol (HTTP), Port (80), and Path ('/'). At the top right, there are 'Save' and 'Discard' buttons, with 'Save' being highlighted by a dashed blue border.

How it works...

The geographic routing method matches the request origin with the closest endpoint in terms of geographical location.

For example, let's say we have multiple endpoints, each on different continents. If a request comes from Europe, it would make no sense to route it to Asia or North America. The geographic routing method will make sure that the request coming from Europe will be pointed to the endpoint located in Europe.

Managing endpoint

After we add endpoints to Traffic Manager, we may have to make changes over time. This can be either to make adjustments or to completely remove endpoints. In this recipe, we'll edit existing Traffic Manager endpoints.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

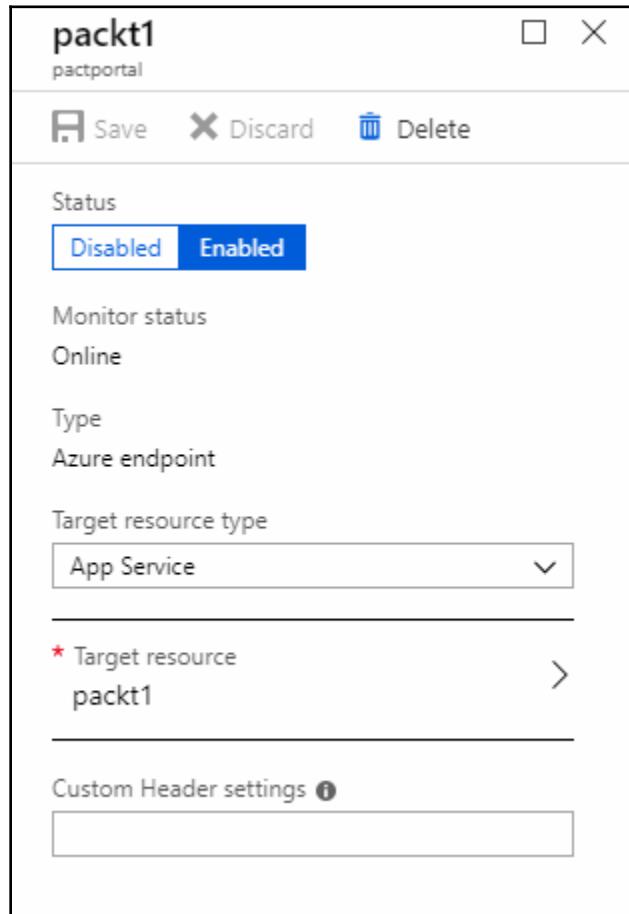
In order to make changes to endpoints in Traffic Manager, we must do the following:

1. In the Azure portal, locate the previously created Traffic Manager.
2. Under **Settings**, select **Endpoints**. From the list that appears, select the endpoint you want to change:

The screenshot shows the Azure portal interface for managing Traffic Manager endpoints. On the left, there's a sidebar with navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Configuration, Real user measurements, Traffic view, Endpoints), and Help & feedback. The 'Endpoints' link under 'Settings' is highlighted with a blue background. The main content area is titled 'pactportal - Endpoints' and shows a table of endpoints. The table has columns: NAME, STATUS, MONITOR STATUS, TYPE, and LOCATION. Two entries are listed: 'packt1' (Enabled, Online, Azure endpoint, West Europe) and 'Packt2' (Enabled, Online, Azure endpoint, West US). There are 'Add' and 'Refresh' buttons at the top of the table, and a search bar labeled 'Search endpoints'.

NAME	STATUS	MONITOR STATUS	TYPE	LOCATION
packt1	Enabled	Online	Azure endpoint	West Europe
Packt2	Enabled	Online	Azure endpoint	West US

3. In the new blade, we can either delete, disable, or make adjustments to the endpoint:



How it works...

The existing endpoint in the Traffic Manager backend can be changed. We can delete the endpoint to completely remove it from Traffic Manager, or we can disable it to temporarily remove it from the backend. We can also change the endpoint completely, to point to another service or a completely different type.

Managing profiles

The Traffic Manager profile is another setting that we can manage and adjust. Although it has very limited options, where we can only disable and enable Traffic Manager, managing the profile setting can be very useful for maintenance purposes. In this recipe, we'll manage Traffic Manager profile.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to make changes to the Traffic Manager profile, we must do the following:

1. In the Azure portal, locate the previously created Traffic Manager profile
2. In **Overview**, select the **Disable profile** option and confirm:

The screenshot shows the Azure portal interface for a Traffic Manager profile named 'pactportal'. On the left, there's a sidebar with navigation links: Overview (which is selected), Activity log, Access control (IAM), Tags, Diagnose and solve problems, Configuration, Real user measurements, Traffic view, and Endpoints. The main content area has a search bar at the top. Below it, there's a button bar with 'Enable profile', 'Disable profile' (which is highlighted in blue), 'Refresh', 'Move', and 'Delete profile'. A confirmation dialog box is centered, asking 'Do you want to disable the Traffic Manager profile 'pactportal'?'. At the bottom of the dialog are 'Yes' and 'No' buttons. Below the dialog, there's a table titled 'Search endpoints' with columns: NAME, STATUS, MONITOR STATUS, TYPE, and LOCATION. The table contains two rows: 'packt1' (Enabled, Online, Azure endpoint, West Europe) and 'Packt2' (Enabled, Online, Azure endpoint, West US).

NAME	STATUS	MONITOR STATUS	TYPE	LOCATION
packt1	Enabled	Online	Azure endpoint	West Europe
Packt2	Enabled	Online	Azure endpoint	West US

- Once the profile has been disabled, it can be again enabled by the **Enable profile** option:

The screenshot shows the Azure portal interface for managing a Traffic Manager profile named 'pactportal'. The top navigation bar includes a search bar, a back arrow, and several action buttons: 'Enable profile', 'Disable profile', 'Refresh', 'Move', and 'Delete profile'. Below the navigation bar, the 'Overview' tab is selected, indicated by a blue background. To the right of the tabs, the resource group is shown as 'Packt-Networking-Portal'. Under the 'Status' section, it is explicitly stated that the profile is 'Disabled'. There are also links for 'Activity log'.

How it works...

Managing the Traffic Manager profile with the disable and enable options will make the Traffic Manager frontend unavailable or available (based on the option selected). This can be very useful for maintenance purposes. If we must apply changes across all endpoints and changes need to be applied to all endpoints at the same time, we can disable the Traffic Manager profile temporarily. Once the changes are applied to all the endpoints, we can make Traffic Manager available again by enabling profile.

Configuring Traffic Manager with load balancers

Combining Traffic Manager with load balancers is often done to provide maximum availability. Load balancers are limited to providing high availability to a set of resources located in the same region. This gives us an advantage if a single resource fails, as we have multiple instances of a resource. But *what if a complete region fails?* Load balancers can't handle resources in multiple regions, but we can combine load balancers with Traffic Manager to provide even better availability with resources across Azure regions. In this recipe, we'll configure Traffic Manager to work with Load Balancers.

Getting ready

Before you start, open the browser and go to the Azure portal via <https://portal.azure.com>.

How to do it...

In order to set up Traffic Manager with load balancer, we must do the following:

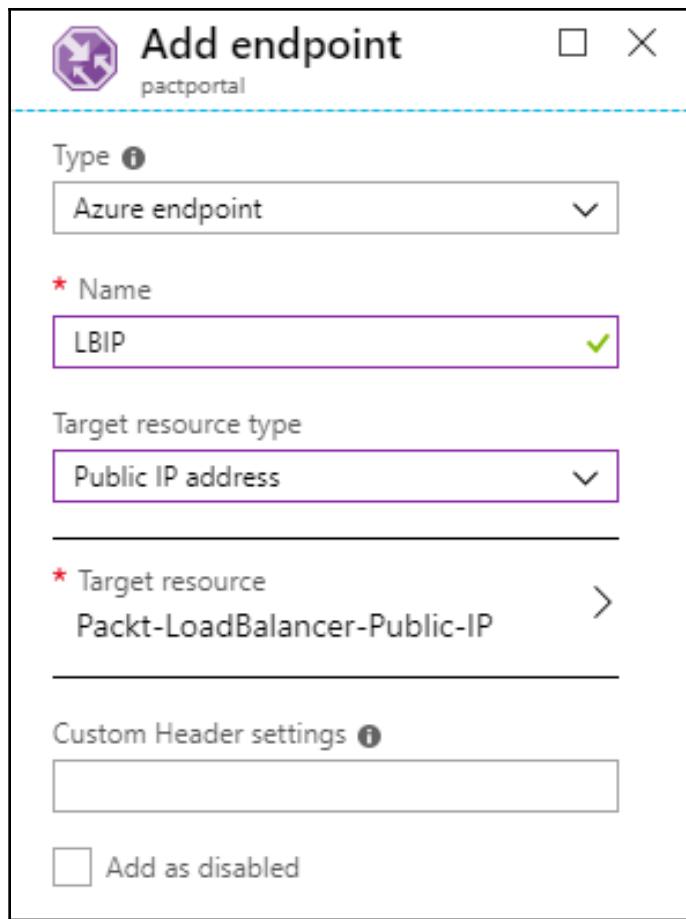
1. In the Azure portal, locate the load balancer and verify that it has the assigned IP address. Only public IP addresses can be used:

The screenshot shows the Azure portal interface for configuring a Public IP address. The title bar reads "Packt-LoadBalancer-Public-IP - Configuration". On the left, there's a sidebar with links: Overview, Activity log, Access control (IAM), Tags, Settings, Configuration (which is selected and highlighted in blue), and Properties. The main pane shows the following configuration details:

- Assignment:** Static (radio button selected)
- IP address:** 51.136.48.107
- Idle timeout (minutes):** 4 (set by a slider)
- DNS name label (optional):** packtportal1 (with a green checkmark)
- Domain name:** .westeurope.cloudapp.azure.com

At the top right, there are "Save" and "Discard" buttons.

2. Go to Traffic Manager and select **Add** to add new endpoint. Select **Azure endpoint** as **Type**, provide a name for the endpoint, and select **Public IP address** as the target resource type. Under **Target resource**, select the public IP address associated with the load balancer:



3. Repeat the process and add another load balancer (from another region) as the Traffic Manager endpoint.

How it works...

Load balancers provide better high availability, keeping a service active even if one of the services in the backend pool fails. If a region fails, load balancers can't provide help because they are limited to a single region. We must provide another set of resources in another region to truly increase availability. But these sets will be completely independent and will not provide failover unless we include Traffic Manager. Traffic Manager will become the frontend, and we will add load balancers as the backend endpoints of Traffic Manager. All requests will come to Traffic Manager first, and then will be routed to the appropriate load balancer in the backend. Traffic Manager will monitor the health of the load balancers, and if one of them becomes unavailable, the traffic will be rerouted to the active one.

10

Azure Application Gateway

Azure Application Gateway is essentially a load balancer for web traffic, but it also allows you better traffic control. Where classic load balancers operate on transport layer, they allow you to route traffic based protocol (TCP or UDP) and IP address, mapping IP address and protocol in the frontend to IP address(es) and protocol in the backend. Application gateway expands on that and allows us to use URLs and paths to determine where traffic should go. For example, we can have multiple servers that are optimized for different things. If one of our servers is optimized for video, then all video requests will be routed to that specific server based on the incoming URL request.

We will cover the following recipes in this chapter:

- Creating a new application gateway
- Configuring the backend pool
- Creating HTTP settings
- Creating a listener
- Creating a rule
- Creating a probe
- Configuring a **Web Application Firewall (WAF)**
- Customizing WAF rules

Technical requirements

For this chapter, an Azure subscription is required.

Creating a new application gateway

Azure Application Gateway can be used as a simple load balancer to perform traffic distribution from frontend to backend based on protocols and ports. But it can also expand on that and perform additional routing based on URLs and paths. This allows us to have resource pools based on roles and also allows us to optimize for specific performance. Using these options and performing routing based on context will increase application performance, along with high availability. Of course, in this case, we need to have multiple resources for each performance type in each backend pool (each performance type requests a separate backend pool).

Getting ready

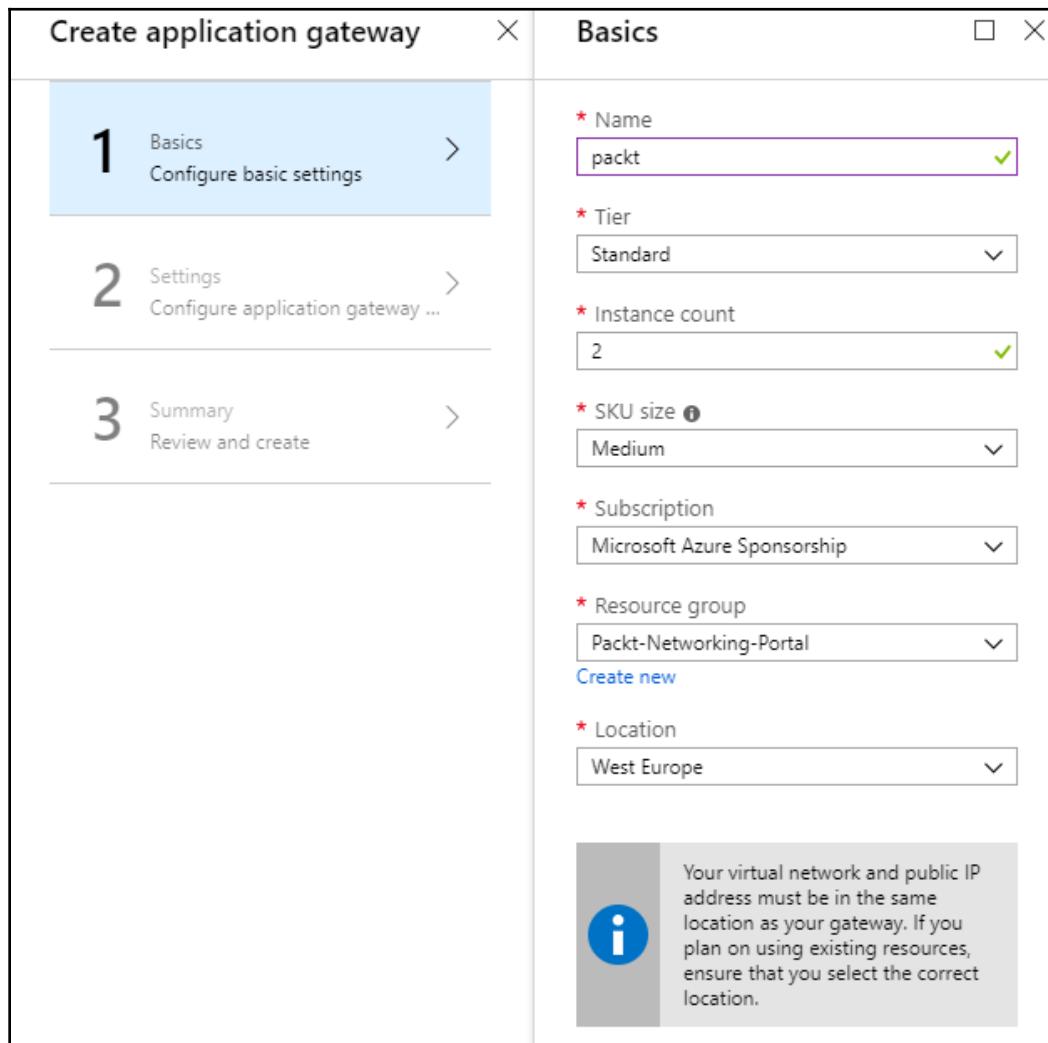
Before you start, open the browser and go to the Azure portal through <https://portal.azure.com>.

How to do it...

In order to create a new application gateway, we must do the following:

1. In the Azure portal, select **Create a resource** and choose **Application Gateway** under the **Networking** services (or search for **application gateway** in the search bar).

2. In the new blade, we must provide information for **Name**, **Tier**, **Instance count**, **SKU size**, **Subscription**, **Resource group**, and **Location**. Note that the virtual network and public IP address (associated with the application gateway) must be in the same region as the application gateway:



3. Under **Settings**, we must select **Virtual network** that will be associated with our application gateway. You will be limited to virtual networks that are located in the region that is selected for the application gateway:

The screenshot displays three overlapping Azure dialog boxes:

- Create application gateway**: Shows the three steps: 1. Basics (Configure basic settings), 2. Settings (Configure application gateway ...), and 3. Summary (Review and create). Step 2 is currently active.
- Settings**: Shows the configuration for a new application gateway. In the "Frontend IP configuration" section, under "IP address type", the "Public" option is selected. Below it, there's a field for "Public IP address" with the value "packt-ip".
- Choose virtual network**: A list of existing virtual networks in the selected subscription and location ("West Europe"). The list includes:
 - Packt-Portal (Packt-Networking-Po...)
 - Packt-Script (Packt-Networking-Sc...)
 - Nagios-vnet (Nagios)
 - RedVSBlue-vnet (RedVSBlue)
 - SQL-WFG-vnet (SQL-WFG)

4. Based on your virtual network selection, we must choose **Subnet** from the virtual network associated with the application gateway. Further to this, we must provide information for the frontend IP address (select either the public or private address type). If **IP address type** selected is **Public**, a new set of options will appear, where we need to define whether we want a new or existing public IP address and also define an **SKU** and **DNS name label** for the public IP address, as shown in the following screenshot:

Create application gateway

Settings

1 Basics ✓
Configure basic settings

2 Settings >
Configure application gateway ...

3 Summary >
Review and create

Subnet configuration

* Virtual network [?](#) Packt-Portal

* Subnet [?](#) BackEnd (10.10.1.0/24)

Frontend IP configuration

* IP address type Public Private

* Public IP address [?](#) Create new Use existing

packt-ip

^ Configure public IP address

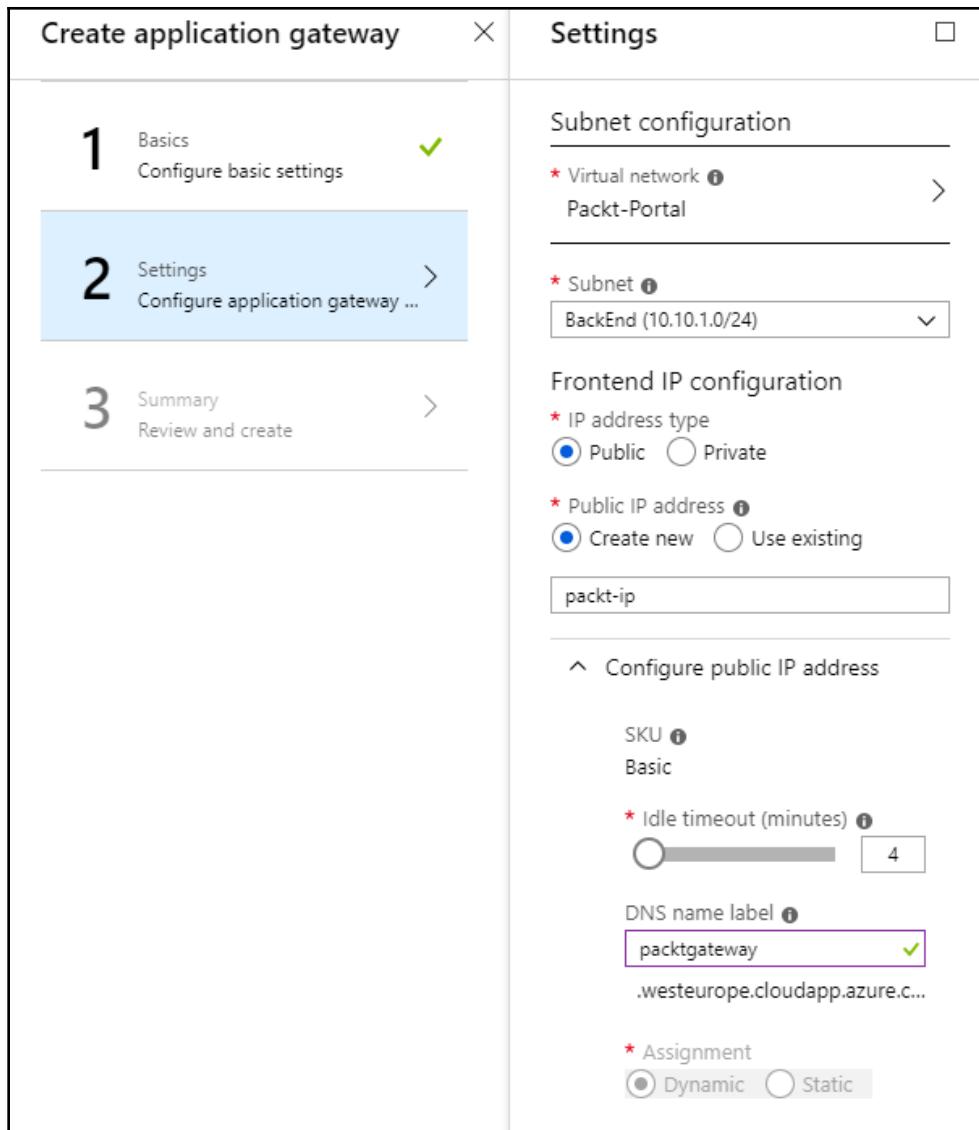
SKU [?](#)
Basic

* Idle timeout (minutes) [?](#) 4

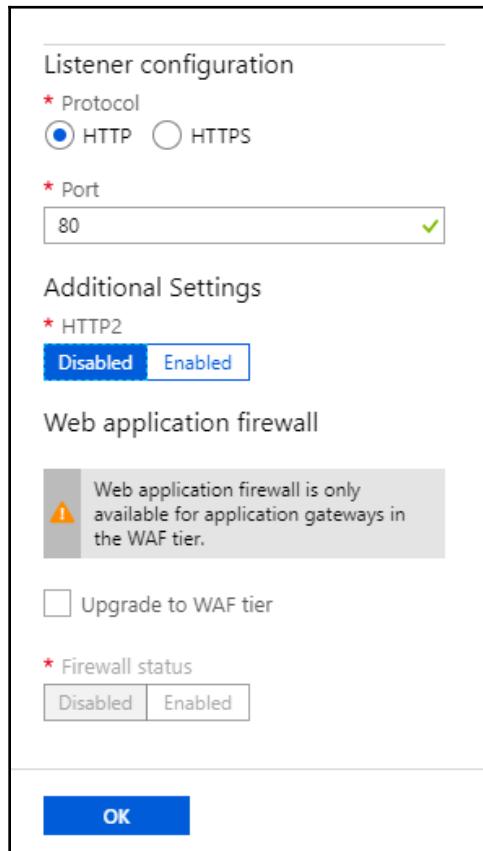
DNS name label [?](#)
packtgateway

.westeurope.cloudapp.azure.c...

* Assignment [?](#)
 Dynamic Static



- Finally, we must define the default listener. For listener, we must select whether we are going to use **HTTP** or **HTTPS**, assign the port, and enable or disable **HTTP2 (Disabled is the default option)**. Note that the firewall settings are available only when the application gateway SKU is set to WAF:



- Finally, the summary is shown, where we can check the settings one more time before deployment is started:

The screenshot shows the 'Create application gateway' wizard in the Azure portal. It is on the 'Summary' step, indicated by a blue header bar. The left sidebar lists three steps: 1. Basics (Configure basic settings), 2. Settings (Configure application gateway ...), and 3. Summary (Review and create). Step 3 is highlighted with a blue background. The main area displays summary details:

Basics	
Name	packt
Tier	Standard
SKU size	Medium
Instance count	2
Maximum scale units	10
Subscription	Microsoft Azure Sponsorship
Resource Group	Packt-Networking-Portal
Location	West Europe
Settings	
Virtual network	Packt-Portal
Subnet	BackEnd
IP address type	Public
Public IP address	(new) packt-ip
Enable SSL for listener	No
Firewall status	Disabled

At the bottom right of the summary table are 'OK' and 'Automation options' buttons.

How it works...

Azure Application Gateway is very similar to load balancers, with some additional options. It will route traffic coming to the frontend of the application gateway to a defined backend based on rules that we will define. In addition to routing based on protocol and port, the application gateway also allows defined routing based on URL and request type. Using these additional rules, we can route incoming requests to endpoints that are optimized for some roles. For example, we can have multiple backend pools with different settings that are optimized to perform only specific tasks. Based on incoming requests, the application gateway will route the request to appropriate backend pool. This approach, along with high availability, will provide better performance by routing each request to a backend pool that will process the request in a more optimized way.

Configuring the backend pool

After the application gateway is created, we must define backend pools. Traffic coming to the frontend of the application gateway will be forwarded to the backend pools. Backend pools in application gateways are the same as backend pools in load balancers, and are defined as possible destinations where traffic will be routed based on other settings that will be added in future recipes in this chapter.

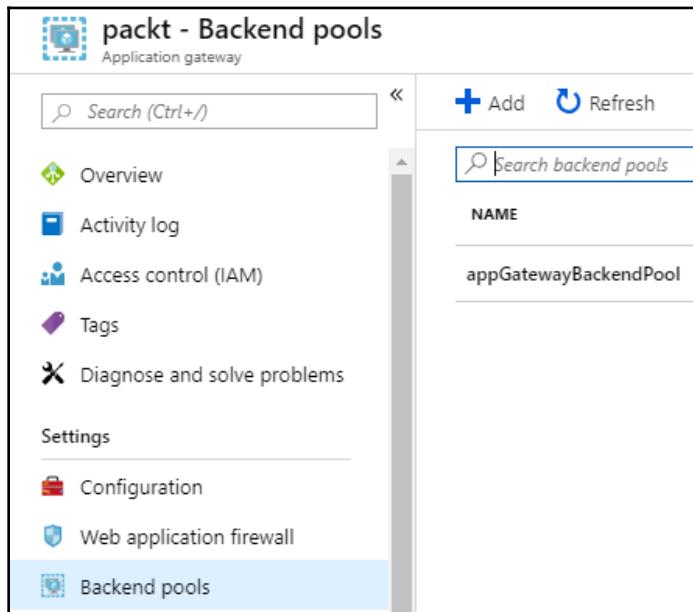
Getting ready

Before you start, open the browser and go to the Azure portal through <https://portal.azure.com>.

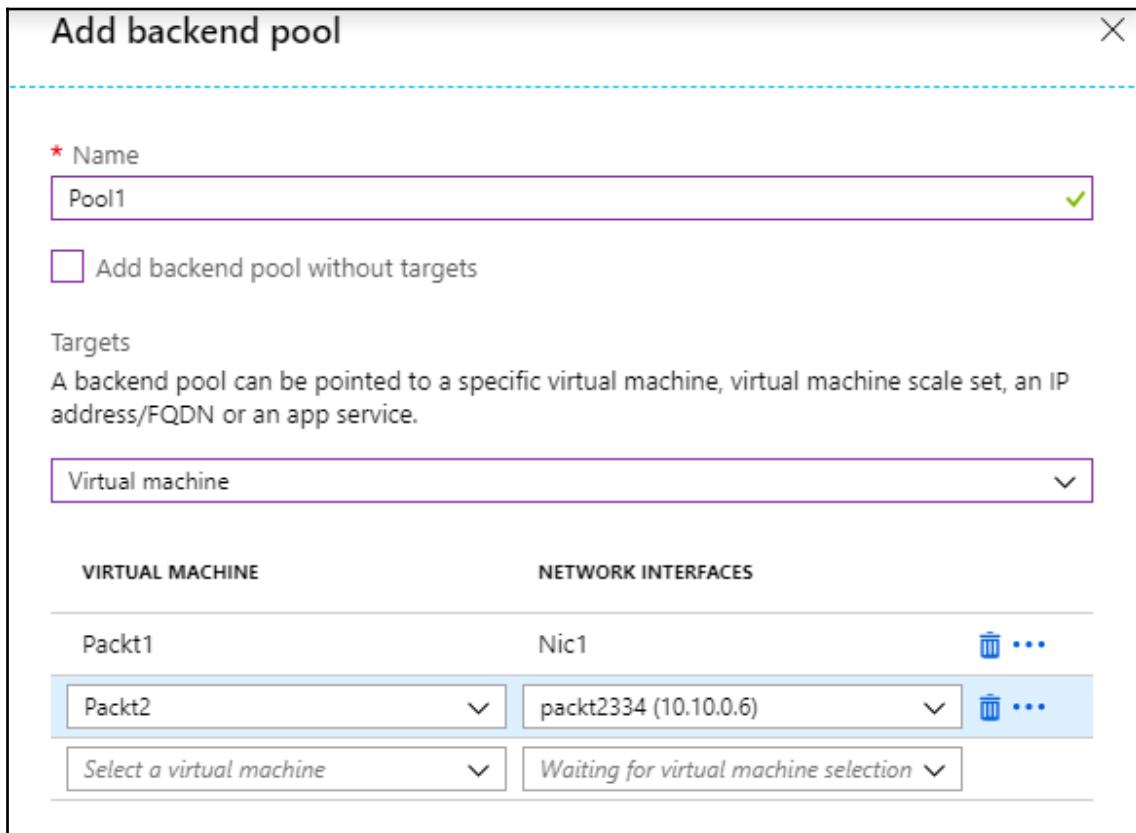
How to do it...

In order to add backend pools to application gateway, we must do the following:

1. In the Azure portal, locate the previously created application gateway.
2. In the **Application gateway** blade, under **Settings**, select **Backend pools**. Select **Add** to add a new backend pool:



3. In the new blade, we must provide the name and type of backend pool target. Available types are virtual machines, virtual machine scale sets, app services, and IP addresses/FQDN. Based on the type selection, you can add appropriate targets:



How it works...

With backend pools, we define targets to which traffic will be forwarded. As the application gateway allows us to define routing by request type, it's best to have targets based on performance and types grouped in same way. For example, if we have multiple web servers, these should be placed in the same backend pool. Servers used for data processing should be placed in a separate pool, and servers used for video in another separate pool. This way, we can separate pools based on performance types and route traffic based on operations that need to be completed.

This will increase the performance of our application, as each request will be processed by the resource best suited for a specific task. To achieve high availability, we should add more servers to each backend pool.

Creating HTTP settings

HTTP settings in application gateways are used for validation and various traffic settings. Their main purpose is to ensure that the request is directed to the appropriate backend pool. Some other HTTP settings are also included, such as affinity or connection draining. Override settings are also part of HTTP settings that will allow you to force-redirect if an incomplete or wrong request is sent.

Getting ready

Before you start, open the browser and go to the Azure portal through <https://portal.azure.com>.

How to do it...

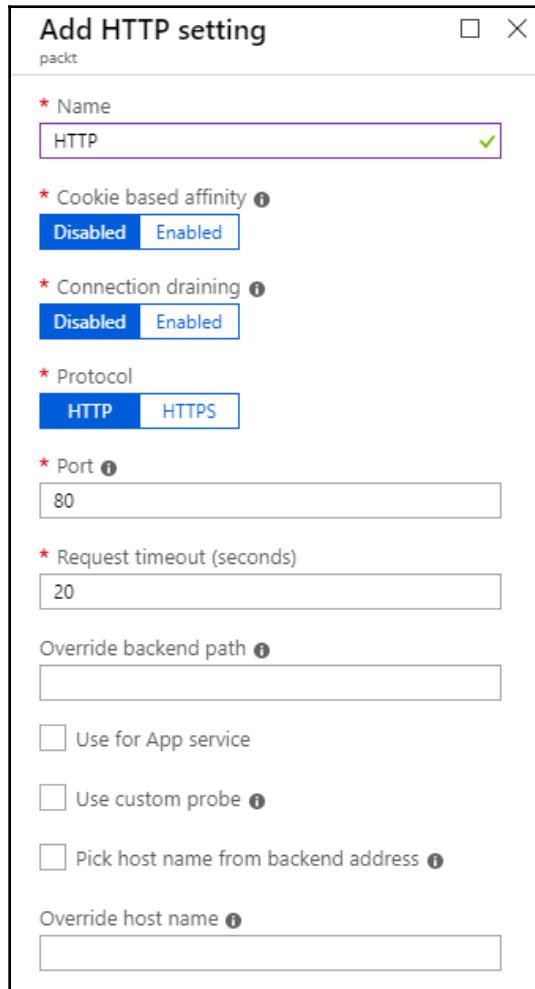
In order to add HTTP settings to application gateway, we must do the following:

1. In the Azure portal, locate the previously created application gateway.

2. In the **Application gateway** blade, under **Settings**, select **HTTP settings**. Select **Add** to add a new HTTP setting:

The screenshot shows the 'packt - HTTP settings' blade in the Azure portal. On the left, there's a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Configuration, Web application firewall, Backend pools, **HTTP settings** (which is selected and highlighted in blue), and Frontend IP configurations. On the right, there's a form titled 'Add' for creating a new HTTP setting. It has a search bar labeled 'Search HTTP settings' and a 'NAME' field containing 'appGatewayBackendHttpSettings'. A blue dashed line indicates the boundary between the navigation menu and the main content area.

3. In the new blade, first, we need to provide **Name**. The next options allow us to disable or enable options such as **Cookie based affinity** and **Connection draining**. Further to this, we select **Protocol**, **Port**, and the **Request timeout (seconds)** period. Optional settings are to use it for the app service or custom probes, or to pick a host name from the backend address pool. Override settings that can be set are **Override backend path** and **Override host name**:



How it works...

As previously mentioned, the main purpose of HTTP settings is to ensure that requests are directed to the correct backend pool. But various other options are available. Cookie-based affinity allows us to route requests from the same source to the same target server in the backend pool. Connection draining will control how the server can be removed from the backend pool. If this is enabled, the server can be removed only when all active connections have stopped. Override settings allow us to override the path of the URL to a different path or a completely new domain, before forwarding the request to the backend pool.

Creating a listener

Listeners in the application gateway listen for any incoming requests. After a new request is detected, it's forwarded to the backend pool based on the rules and settings we have defined.

Getting ready

Before you start, open the browser and go to the Azure portal through <https://portal.azure.com>.

How to do it...

In order to add a listener to application gateway, we must do the following:

1. In the Azure portal, locate the previously created application gateway.
2. In the **Application gateway** blade, under **Settings**, select **Listeners**, then select **Basic** to add a new listener:

The screenshot shows the 'Listeners' blade for an application gateway named 'packt'. The 'Basic' tab is selected. A table lists one listener:

NAME	PROTOCOL
appGatewayHttpListener	HTTP

SSL Policy

Application Gateway provides native support for WebSockets. If a WebSocket traffic is received on using the appropriate backend pool as specified in application configuration.

Configure a centralized SSL policy to match your organization's security requirements. If you don't specify an SSL policy, the default settings will be used.

Default Predefined Custom

Min protocol version
TLSv1_0

3. In the new blade, we need to provide a name for the listener, select the frontend IP, and provide a **Port** and **Protocol** that will be monitored:

Add basic listener

packt

* Name
listener1

* Frontend IP configuration
appGatewayFrontendIP

* Frontend port
+ New

* Name
HTTP

* Port
8080

* Protocol
 HTTP HTTPS

OK

How it works...

Listener monitors for new requests coming to the application gateway. Each listener monitors only one frontend IP address and only one port. If we have multiple frontend IPs and traffic coming on multiple protocols and ports, we must create a listener for each IP address and each port that traffic may be coming to.

Creating a rule

Rules in application gateways are used to determine how traffic flows. Based on different settings, we can determine where a specific request is forwarded to and how this is done.

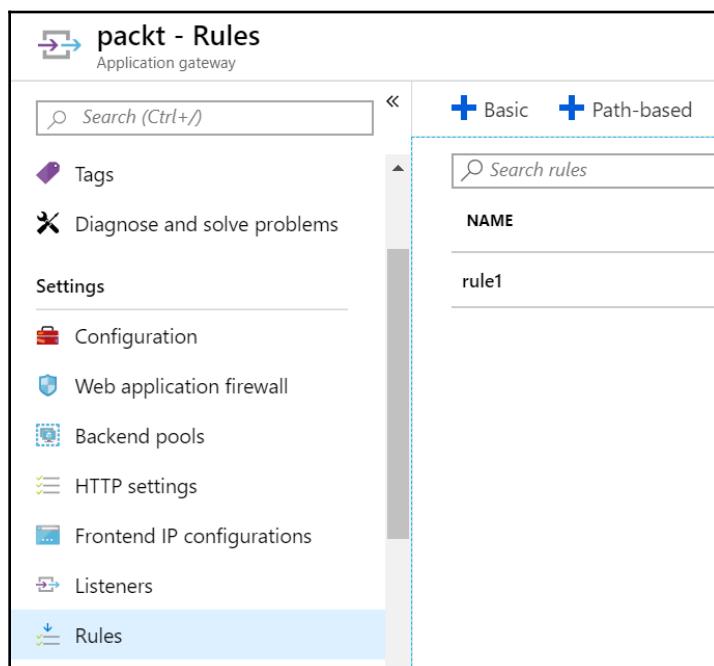
Getting ready

Before you start, open the browser and go to the Azure portal through <https://portal.azure.com>.

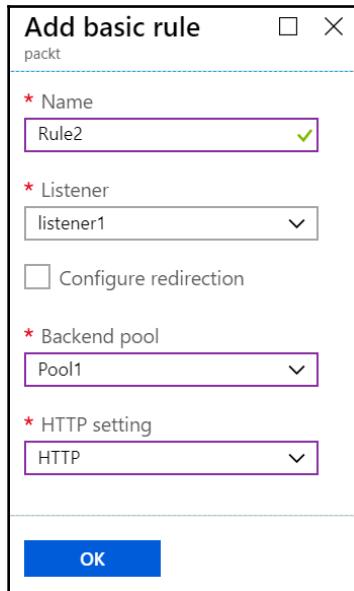
How to do it...

In order to add a rule to the application gateway, we must do the following:

1. In the Azure portal, locate the previously created application gateway.
2. In the **Application gateway** blade, under **Settings**, select **Rules**. Select **Basic** to add a new rule:



3. In the new blade, we must provide a name for the new rule and select the **Listener**, **Backend pool**, and **HTTP setting**, as shown in the following screenshot:



How it works...

Rules tie some previously created settings together. We define a listener that defines what request on what IP address over which port are we expecting. Then these requests are forwarded to the backend pool; forwarding is performed based on the HTTP settings. Optionally, we can also add redirection into the rules.

Creating a probe

Probes in Azure Application Gateway are used to monitor the health of the backend endpoints. Each endpoint is monitored, and if found to be unhealthy, is temporarily taken out of the pool. Once the status changes, it's added back to the pool. This prevents requests from being sent to unhealthy endpoints that couldn't resolve the request.

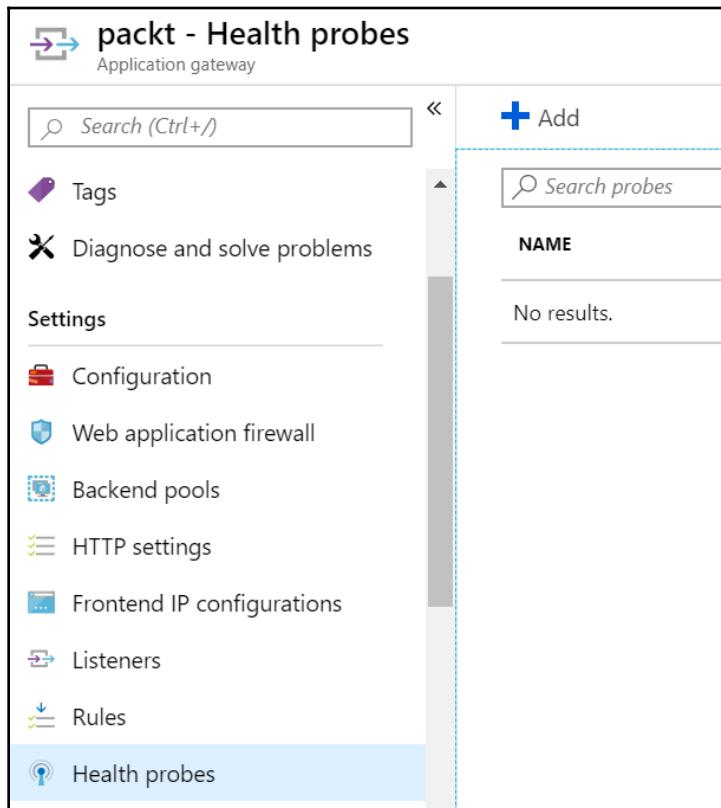
Getting ready

Before you start, open the browser and go to the Azure portal through <https://portal.azure.com>.

How to do it...

In order to add a probe to our application gateway, we must do the following:

1. In the Azure portal, locate the previously created application gateway.
2. In the **Application gateway** blade, under **Settings**, select **Health probes**. Select **Add** to add the new probe:



3. In the new blade, we must provide the **Name** of the probe, along with the **Protocol**, **Host** and **Path**. We also need to set the **Interval**, **Timeout**, and **Unhealthy threshold**:

The screenshot shows the 'Add health probe' dialog box. It has fields for Name (probe1), Protocol (HTTP selected), Host (toroman.cloud), Path (/video/*), Interval (30 seconds), Timeout (30 seconds), and Unhealthy threshold (3). There are also checkboxes for picking host from backend http settings and using probe matching conditions, both of which are unchecked. A blue 'OK' button is at the bottom.

Setting	Value
Name	probe1
Protocol	HTTP
Host	toroman.cloud
Path	/video/*
Interval (seconds)	30
Timeout (seconds)	30
Unhealthy threshold	3

How it works...

Protocol, **Host**, and **Path** define what probe is being monitored. **Interval** defines how often checks are performed. **Timeout** defines how much time must pass before the check is declared to be failed. Finally, **Unhealthy threshold** is used to set how many failed checks must occur before the endpoint is declared unavailable.

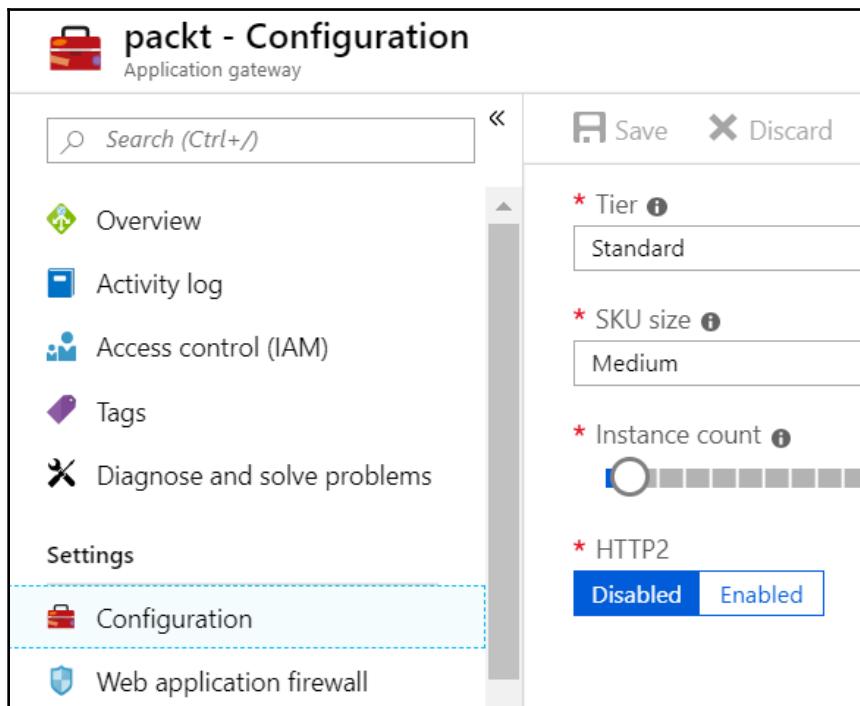
Configuring a WAF

WAF is an additional setting for the application gateway. It's used to increase the security of applications behind the application gateway and also provides centralized protection.

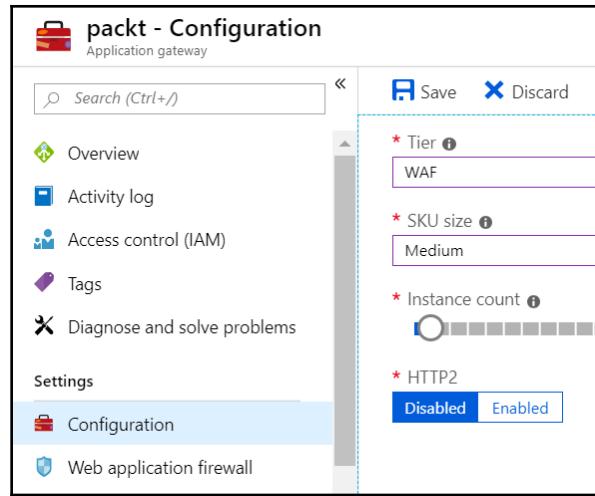
Getting ready

To enable WAF, we must set the application gateway to the WAF tier. To do so, we must do the following:

1. In the **Application gateway** blade, go to **Configuration**, under **Settings**, as shown in the following screenshot:



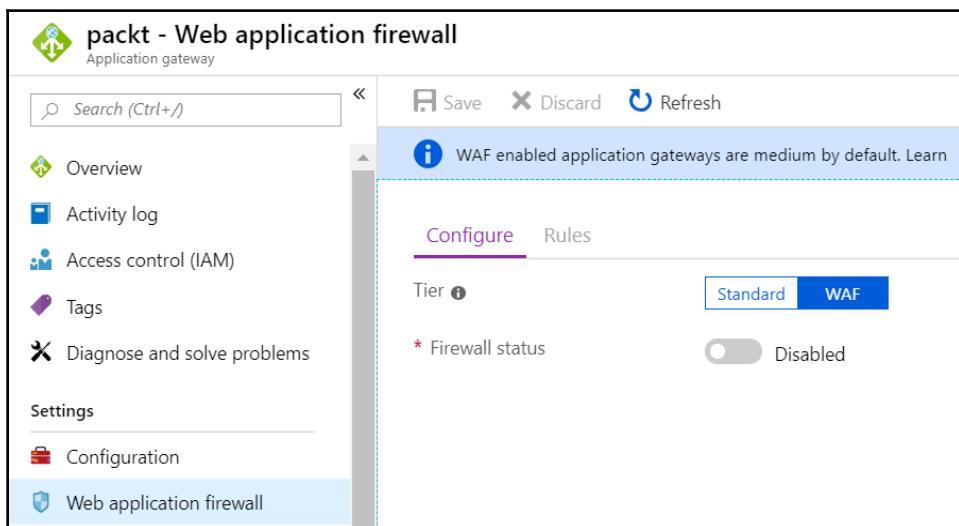
2. Change the **Tier** selection from **Standard** to **WAF**. Click the **Save** button:



How to do it...

After the application gateway is set to WAF, we can enable and set the firewall rules. To do so, we must do the following:

1. In the **Application gateway** blade, go to **Web application firewall**, under **Settings**:



2. After we set **Firewall status** to **Enabled**, a new set of options will appear:

The screenshot shows the 'Firewall' configuration section of the Azure Application Gateway settings. At the top, the 'Firewall status' is set to 'Enabled'. Below it, the 'Firewall mode' is set to 'Prevention'. A section titled 'Exclusions' contains a note that 'packt will evaluate everything in the request except for the items included in this list.' There is a table with columns 'FIELD', 'OPERATOR', and 'SELECTOR' for defining exclusions. Under 'Global parameters', the 'Inspect request body' option is turned 'On'. The 'Max request body size (KB)' field is highlighted with a red border and has validation errors: 'The value should not be empty.' and 'The value must be between 8 and 128.'. The 'File upload limit (MB)' field is shown with a green checkmark indicating it is valid.

FIELD	OPERATOR	SELECTOR
<input type="text"/>	<input type="text"/>	<input type="text"/>

Global parameters

Inspect request body On

Max request body size (KB) ⓘ The value should not be empty. The value must be between 8 and 128.

File upload limit (MB) ✓

3. We must select **Firewall mode**, the exclusion list, and **Global parameters** as follows:

The screenshot shows the WAF configuration page. At the top, there are two sections: 'Firewall status' (Enabled) and 'Firewall mode' (Prevention). Below these are 'Exclusions' and a table for defining them. The table has columns: FIELD, OPERATOR, and SELECTOR. One row is shown: Request header name Equals barerToken. There are also three empty rows below it. Under 'Global parameters', there are three settings: 'Inspect request body' (On), 'Max request body size (KB)' (8), and 'File upload limit (MB)' (10).

How it works...

The WAF feature helps increase security by checking all incoming traffic. As this can slow down performance, we can exclude some items. Excluded items will not be checked. WAF can work in two modes: detection and prevention. Detection will only detect if a malicious request is sent, while prevention will stop any such request.

Customizing WAF rules

WAF comes with a predetermined set of rules. These rules are enforced to increase application security and prevent malicious requests. We can change these rules to address specific issues or requirements.

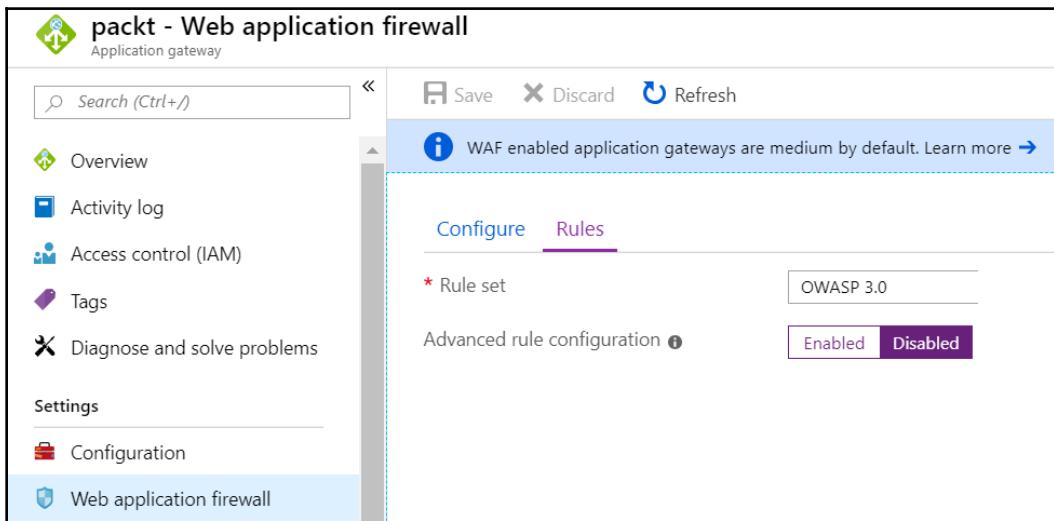
Getting ready

Before you start, open the browser and go to the Azure portal through <https://portal.azure.com>.

How to do it...

In order to change the WAF rules, we must do the following:

1. Select **Web application firewall** under **Settings** in the **Application gateway** blade.
2. Select **Rules** in the WAF settings. Select **Enabled** under **Advanced rule configuration**, as shown in the following screenshot:



3. Rules will appear in the form of a list. We can check or uncheck boxes to enable or disable rules:

The screenshot shows the 'Rules' tab selected in the Azure Application Gateway WAF configuration interface. At the top, there is a dropdown menu for the 'Rule set' containing 'OWASP 3.0'. Below it, there are two buttons: 'Enabled' (highlighted in purple) and 'Disabled'. A search bar labeled 'Search rules' is present. The main area displays a table with columns: 'ENABLED', 'NAME', and 'DESCRIPTION'. All seven rules listed are currently enabled, indicated by checked checkboxes in the 'ENABLED' column. The rules are: General, REQUEST-911-METHOD-ENFORCEMENT, REQUEST-913-SCANNER-DETECTION, REQUEST-920-PROTOCOL-ENFORCEMENT, REQUEST-921-PROTOCOL-ATTACK, REQUEST-930-APPLICATION-ATTACK-LFI, and REQUEST-931-APPLICATION-ATTACK-RFI.

ENABLED	NAME	DESCRIPTION
<input checked="" type="checkbox"/>	▶ General	
<input checked="" type="checkbox"/>	▶ REQUEST-911-METHOD-ENFORCEMENT	
<input checked="" type="checkbox"/>	▶ REQUEST-913-SCANNER-DETECTION	
<input checked="" type="checkbox"/>	▶ REQUEST-920-PROTOCOL-ENFORCEMENT	
<input checked="" type="checkbox"/>	▶ REQUEST-921-PROTOCOL-ATTACK	
<input checked="" type="checkbox"/>	▶ REQUEST-930-APPLICATION-ATTACK-LFI	
<input checked="" type="checkbox"/>	▶ REQUEST-931-APPLICATION-ATTACK-RFI	

How it works...

WAF comes with all rules activated by default. This can slow down performance, so we can disable some of the rules if needed. Also, there are two rule sets available—**OWASP 2.2.9** and **OWASP 3.0**. The default rule set is OWASP 3.0, but we can switch between rule sets as per requirements.

11

Azure Firewall

Most Azure networking components used for security are there to stop incoming unwanted traffic. Whether we use network security groups, application security groups, or a web application firewall, they all have one single purpose—to stop unwanted traffic reaching our services. Azure Firewall has similar functionality, including one extension, which we can use to stop outbound traffic from leaving the virtual network.

We will cover the following recipes in this chapter:

- Creating a new Azure Firewall
- Configuring a new allow rule
- Configuring a new deny rule
- Configuring a route table
- Enabling diagnostic logs for Azure Firewall

Technical requirements

For this chapter, the following is required:

- An Azure subscription
- Azure PowerShell

The code samples can be found at <https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter%2011>.

Creating a new Azure Firewall

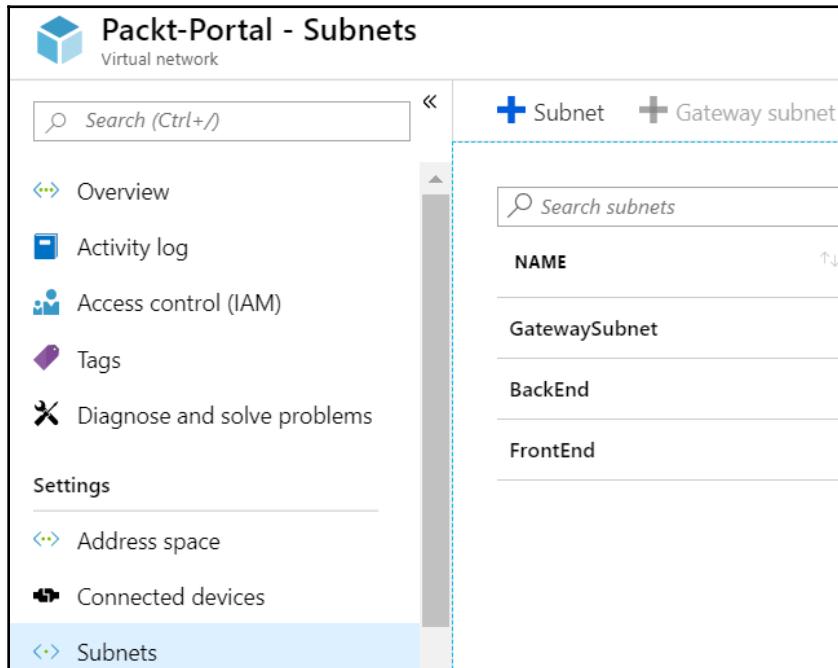
Azure Firewall allows us total control over our traffic. Besides controlling inbound traffic, with Azure Firewall, we can control outbound traffic as well.

Getting ready

Before we can create an Azure Firewall instance, we must prepare the subnet.

In order to create a new subnet for Azure Firewall, we must do the following:

1. Locate a virtual network that will be associated with Azure Firewall.
2. Select the **Subnets** option under settings and select the option to add new subnet, as shown in the following screenshot:



3. In the new blade, we must provide the name and address range. It's very important that the subnet is named `AzureFirewallSubnet`:

Add subnet

Packt-Portal

* Name
AzureFirewallSubnet 

* Address range (CIDR block) 
10.10.3.0/24
10.10.3.0 - 10.10.3.255 (251 + 5 Azure reserved addresses)

Network security group >
None

Route table >
None

Service endpoints

Services 
0 selected

Subnet delegation

Delegate subnet to a service 

OK

How to do it...

In order to create a new Azure Firewall instance using the Azure portal, take the following steps:

1. In the Azure portal, select **Create a resource** and choose **Azure Firewall** under **Networking** services (or, search for **Azure Firewall** in the search bar).

2. In the new blade, first, we must provide values for **Subscription** and **Resource group**. We need to fill in the **Name** and **Region** fields for Azure Firewall, as shown in the following screenshot:

The screenshot shows the 'Basics' tab of the Azure Firewall configuration blade. It includes sections for 'PROJECT DETAILS' and 'INSTANCE DETAILS'. Under 'PROJECT DETAILS', 'Subscription' is set to 'Microsoft Azure Sponsorship' and 'Resource group' is set to 'Packt-Networking-Portal'. Under 'INSTANCE DETAILS', 'Name' is set to 'packt-firewall' and 'Region' is set to 'West Europe'.

3. Next, we proceed to virtual network selection. Only virtual networks in the region where Azure Firewall will be created are available. Also, the selected virtual network must contain `AzureFirewallSubnet`. Finally, we define **Public IP address**, as shown in the following screenshot:

The screenshot shows the 'INSTANCE DETAILS' section of the Azure Firewall configuration blade. It includes fields for 'Name' (set to 'packt-firewall'), 'Region' (set to 'West Europe'), 'Virtual network' (set to 'Packt-Portal (Packt-Networking-Portal)'), and 'Public IP ADDRESS' (with 'Create new' selected and name 'azureFirewalls-ip').

How it works...

Azure Firewall uses a set of rules to control outbound traffic. We can either block everything by default and allow only white-listed traffic, or we can allow everything and block only black-listed traffic.

Configuring a new allow rule

If we want to allow specific traffic, we must create an allow rule. Rules are applied on priority level, so a rule will be applied only when there is not another rule with higher priority.

Getting ready

Open the PowerShell console and make sure you are connected to your Azure subscription.

How to do it...

In order to create a new allow rule in Azure Firewall, execute the following command:

```
$RG="Packt-Networking-Portal"
$Location="West Europe"
$Azfw = Get-AzureRmFirewall -ResourceGroupName $RG
$Rule = New-AzureRmFirewallApplicationRule -Name Rule1 -Protocol
"http:80", "https:443" -TargetFqdn "*packt.com"
$RuleCollection = New-AzureRmFirewallApplicationRuleCollection -Name
RuleCollection1 -Priority 100 -Rule $Rule -ActionType "Allow"
$Azfw.ApplicationRuleCollections = $RuleCollection
Set-AzureRmFirewall -AzureFirewall $Azfw
```

How it works...

An allow rule in Azure Firewall will whitelist specific traffic. If there is a rule that would also block this traffic, the higher priority rule will be applied.

Configuring a new deny rule

If we want to deny specific traffic, we must create a deny rule. Rules are applied by priority, so this rule will be applied only if there is not a higher priority rule.

Getting ready

Open the PowerShell console and make sure you are connected to your Azure subscription.

How to do it...

In order to create a new deny rule in Azure Firewall, execute the following command:

```
$RG="Packt-Networking-Portal"
$Location="West Europe"
$Azfw = Get-AzureRmFirewall -ResourceGroupName $RG
$Rule = New-AzureRmFirewallApplicationRule -Name Rule1 -Protocol
"http:80","https:443" -TargetFqdn "*google.com"
$RuleCollection = New-AzureRmFirewallApplicationRuleCollection -Name
RuleCollection1 -Priority 100 -Rule $Rule -ActionType "Deny"
$Azfw.ApplicationRuleCollections = $RuleCollection
Set-AzureRmFirewall -AzureFirewall $Azfw
```

How it works...

The deny rule is the most commonly used option with Azure Firewall. An approach where you block everything and allow only white-listed traffic isn't very practical, as we may end up adding too many allow rules. Therefore, the most common approach is to use deny rules to block certain traffic that we want to prevent.

Configuring a route table

Route tables are commonly used with Azure Firewall when there is cross-connectivity. Cross-connectivity can be either be with other Azure virtual networks or with on-premises networks. In such cases, Azure Firewall uses route tables to forward traffic based on rules in route tables.

Getting ready

Open the PowerShell console and make sure you are connected to your Azure subscription.

How to do it...

In order to create a new route table in Azure Firewall, execute the following command:

```
$RG="Packt-Networking-Portal"
$Location="West Europe"
$Azfw = Get-AzureRmFirewall -ResourceGroupName $RG
$config = $Azfw.IpConfigurations[0].PrivateIPAddress
$route = New-AzureRmRouteConfig -Name 'Route1' -AddressPrefix 0.0.0.0/0 -
NextHopType VirtualAppliance -NextHopIpAddress $config
$routeTable = New-AzureRmRouteTable -Name 'RouteTable1' -ResourceGroupName
$RG -location $Location -Route $route
```

How it works...

Using route tables associated with Azure Firewall, we can define how traffic between networks is handled and how we route traffic from one network to another. In a multi-network environment, especially in a hybrid network where we connect an Azure virtual network with a local on-premises network, this option is very important. This allows us to determine what kind of traffic can flow where and how.

Enabling diagnostic logs for Azure Firewall

Diagnostics are a very important of any IT system, and networking is no exception. Diagnostics settings in Azure Firewall allow us to collect various information that can be used for troubleshooting or for auditing.

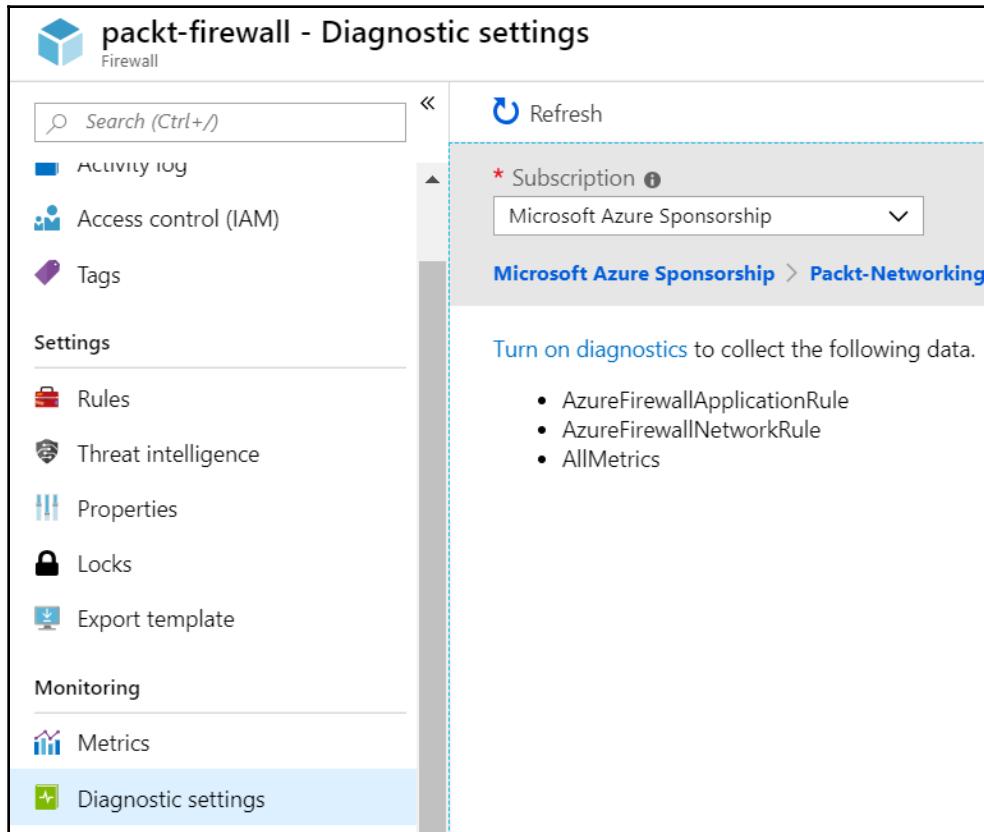
Getting ready

Before you start, open your browser and go to the Azure portal: <https://portal.azure.com>.

How to do it...

To enable diagnostics in Azure Firewall, we must follow these steps:

1. In the Azure Firewall blade, locate **Diagnostics settings** under **Monitoring**.
2. Select the **Turn on diagnostics** option, as shown in the following screenshot:



3. In the new blade, fill in the **Name** field and specify where the logs will be stored. Choose the **Storage account** where the logs will be stored and specify the retention period and which logs will be stored, as shown in the following screenshot:

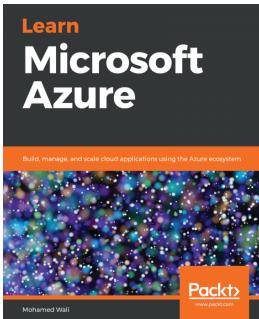
The screenshot shows the 'Diagnostics settings' configuration page. At the top, there are three buttons: 'Save' (blue), 'Discard' (red), and 'Delete' (grey). Below these, a red asterisk indicates a required field: 'Name'. The name 'AZFW_Logs' is entered in the text input field, which has a green checkmark icon to its right. A checked checkbox labeled 'Archive to a storage account' is present. Under 'Storage account', 'nagiosdiag316' is listed with a blue arrow icon to its right. There are two unchecked checkboxes: 'Stream to an event hub' and 'Send to Log Analytics'. The 'LOG' section contains two entries: 'AzureFirewallApplicationRule' and 'AzureFirewallNetworkRule', both with checked checkboxes. Each entry includes a 'Retention (days)' slider set to 30, with a value box showing '30'. The 'METRIC' section contains one entry: 'AllMetrics', with a checked checkbox and a 'Retention (days)' slider set to 30, with a value box showing '30'.

How it works...

Diagnostics has two purposes—auditing and troubleshooting. Based on traffic and settings, these logs can grow over time, so it's good to know about the main purpose for enabling diagnostics in the first place. If diagnostics are enabled for auditing, you will probably want to choose a maximum of 365 days of retention. If the main purpose is troubleshooting, the retention period can be kept at 7 days or an even shorter time.

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:

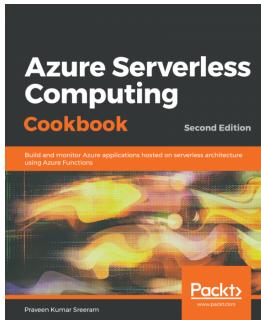


Learn Microsoft Azure

Mohamed Wali

ISBN: 978-1-78961-758-0

- Understand the cloud services offered by Azure
- Design storage and networks in Azure for your Azure VM
- Work with web apps and Azure SQL databases
- Build your identity management solutions on Azure using Azure AD
- Monitor, protect, and automate your Azure services using Operation Management Suite (OMS)
- Implement OMS for Azure services



Azure Serverless Computing Cookbook - Second Edition

Praveen Kumar Sreeram

ISBN: 978-1-78961-526-5

- Integrate Azure Functions with other Azure services
- Understand cloud application development using Azure Functions
- Employ durable functions for developing reliable and durable serverless applications
- Use SendGrid and Twilio services
- Explore code reusability and refactoring in Azure Functions
- Configure serverless applications in a production environment

Leave a review - let other readers know what you think

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

Index

A

address space size
 modifying 16, 17
allow rule
 configuring 201
 creating, in NSG 37, 38, 39
application gateway
 creating 173, 175, 177, 178
Application Security Group (ASG)
 associating, with VM 50, 51, 52
 creating 49, 50
 rules, creating with 52, 53
availability sets 139
Azure Application Gateway 172, 173
Azure DNS zone
 creating 110, 111
Azure DNS
 record set, creating 112, 113, 114
 record, creating 112, 113, 114
Azure Firewall
 creating 197, 198, 199, 200, 201
 diagnostic logs, enabling for 203, 204, 205
Azure portal
 local network gateway, creating 73, 74
 NSG, creating in 35, 36
 public IP address, creating in 55, 57
 reference 20
 subnet, adding in 11, 12, 14
 virtual network gateway, creating 75, 76, 77, 78
 virtual network, creating in 8, 9
Azure Traffic Manager 151
Azure VMs
 creating 20, 21, 22, 23, 24, 25, 26
Azure
 VPN device configuration, downloading from 88,
 90, 91

B

backend pool
 configuring 179, 180
 creating 136, 138, 139
Border Gateway Protocol (BGP) 73

C

Classless Inter-Domain Routing (CIDR) 10

D

demilitarized zone (DMZ) 134
deny rule
 configuring 202
 creating, in NSG 39, 40, 41
diagnostic logs
 enabling, for Azure Firewall 203, 204, 205
distributed traffic
 configuring 158, 160, 161

E

endpoints
 managing, in Traffic Manager 165, 166
external load balancers 132

F

Fully Qualified Domain Name (FQDN) 152

H

health probes
 creating 140, 141, 143
HTTP settings
 creating 181, 183, 184

I

inbound Network Address Translation (NAT) rules
 creating 147, 149, 150
internal load balancer
 creating 132, 134

L

listener
 creating 184, 185, 186
load balancer rules
 creating 143, 145, 146
load balancers
 Traffic Manager, configuring with 168, 171
local network gateway settings
 modifying 80, 81
local network gateway
 creating, in portal 73, 74
 creating, with PowerShell 75

N

network interface (NIC) 26, 34
network interface
 attaching, to VM 31, 32
 creating 29, 31
 detaching, from VM 32, 33
 NSG, assigning to 45, 46, 47, 48
network peering
 used, for connecting VNets 104, 106, 107
Network Security Groups (NSGs) 34
NSG rule
 creating, with PowerShell 41
NSG
 allow rule, creating 37, 38, 39
 assigning, to network interface 45, 46, 47, 48
 assigning, to subnet 42, 43, 44, 45
 assigning, with PowerShell 48, 49
 creating, in portal 35, 36
 creating, with PowerShell 36, 37, 42
 deny rule, creating 39, 40, 41
 rules, creating with 52, 53

O

OWASP 2.2.9 196
OWASP 3.0 196

P

Point-to-Site connection
 creating 91, 92, 93, 94, 95, 96, 97, 98, 99
PowerShell
 local network gateway, creating 75
 NSG rule, creating 41, 42
 NSG, assigning with 48, 49
 NSG, creating 36, 37
 used, for adding subnet 14, 15
 used, for creating public IP address 57
 used, for creating virtual network 10, 11
 virtual network gateway, creating 79, 80
Pre-Shared Key (PSK) 87
Priority routing method
 traffic, configuring 161, 162, 163
private IP address
 modifying, for private IP address 69
 reservation, creating for 66, 67
 reservation, modifying for 68
 reservation, removing for 70, 71
probe
 creating 188, 189, 190
public IP address
 assigning 58, 59, 60
 creating, in portal 55, 57
 creating, with PowerShell 57
 reservation, creating for 62, 63
 reservation, removing for 64, 65
 unassigning 61, 62
public load balancer
 creating 134, 135, 136

R

record set
 creating, in Azure DNS 112, 113, 114
record
 creating, in Azure DNS 112, 113, 114
Remote Desktop Protocol (RDP) 82
reservation
 creating, for private IP address 66, 67
 creating, for public IP address 62, 63
 modifying, for private IP address 68, 69
 removing, for private IP address 70, 71
 removing, for public IP address 64, 65

route table
associating, to subnet 118, 119, 120, 121
configuring 202, 203
creating 115, 116
dissociating, from subnet 121, 122, 123, 124, 125
modifying 116, 117
route
creating 125, 126, 127
deleting 129, 130, 131
modifying 127, 128, 129
rules
creating 187, 188
creating, with ASG 52, 53
creating, with NSG 52, 53

S

Site-2-Site connection
creating 83, 84, 85, 86, 87, 88
subnet size
modifying 17, 18, 19
subnet
adding, in portal 11, 12, 14
adding, with PowerShell 14, 15
NSG, assigning to 42, 43, 44, 45
route table, associating to 118, 119, 120, 121
route table, dissociating from 121, 122, 123, 124, 125

T

Traffic Manager profile
creating 152, 153, 154
endpoint, adding 155, 156, 158
managing 167, 168
Traffic Manager

configuring, with load balancers 168, 171
traffic
configuring, based on geographical location 163, 164
configuring, based on priority 161, 162, 163

V

virtual machines 139
virtual machines scale sets 139
virtual network gateway
creating, in portal 75, 76, 77, 78
creating, with PowerShell 79, 80
virtual network
creating, in portal 8, 9
creating, with PowerShell 10, 11
virtual private network (VPN) 134
VM network settings
viewing 27, 28
VM
ASG, associating with 50, 51, 52
network interface, attaching to 31, 32
network interface, detaching from 32, 33
VNet-2-VNet connection
creating 99, 100, 101, 102, 103, 104
VNets
connecting, network peering used 104, 106, 107
VPN device configuration
downloading, from Azure 88, 90, 91
reference 88

W

WAF rules
customizing 194, 196
WAF
configuring 191, 192, 194