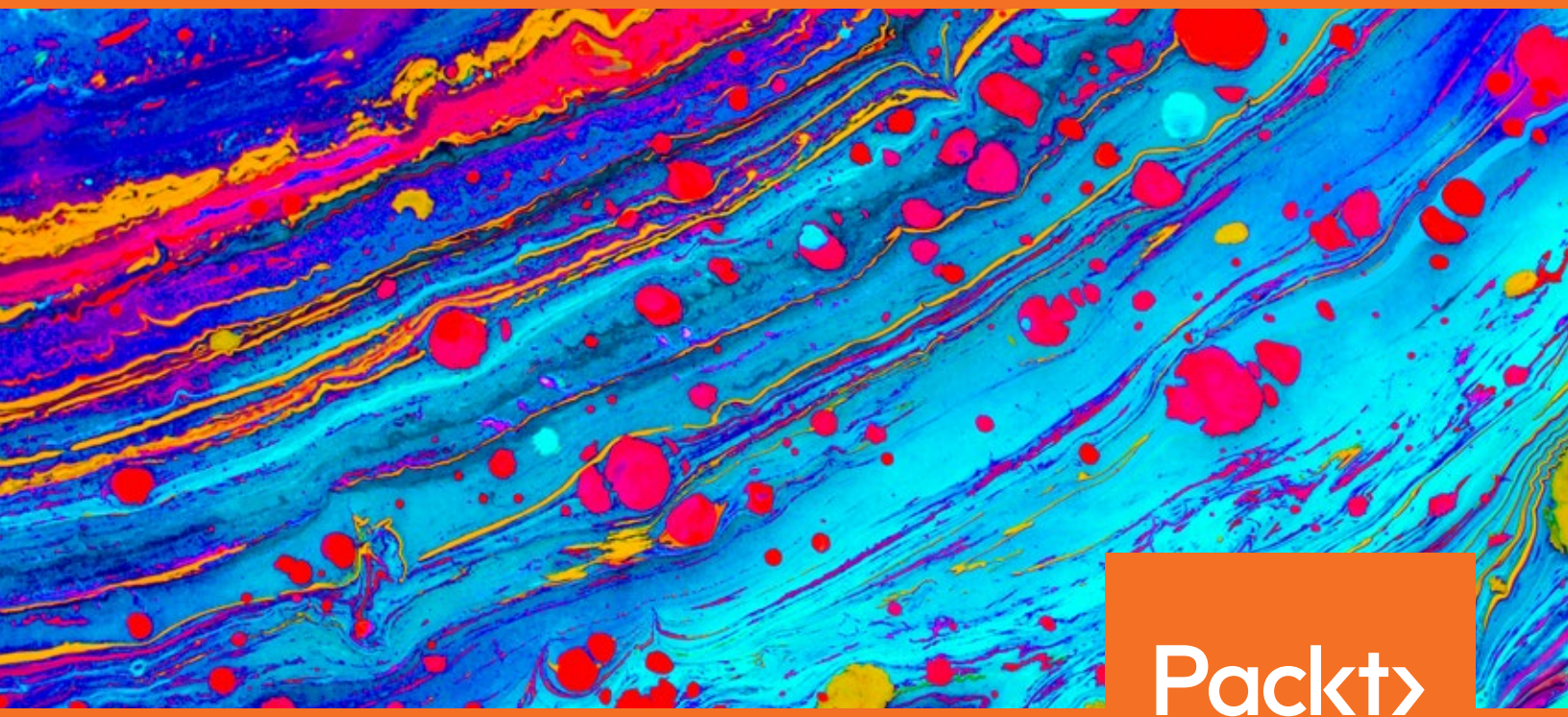
Azure Networking Cookbook

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



Mustafa Toroman

Packt
www.packt.com

Azure Networking Cookbook

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

Mustafa Toroman

Packt

BIRMINGHAM – MUMBAI

Azure Networking Cookbook

Commissioning Editor: Vijin Boricha
Acquisition Editor: Shrilekha Inan
Content Development Editor: Ronn Kurien
Technical Editor: Pratik Shet
Copy Editor: Safis Editing
Project Co-ordinator: Jagdish Prabhu
Proofreader: Safis Editing
Indexer: Tejal Daruwale Soni
Graphics: Tom Scaria
Production Co-ordinator: 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

https://mapt/io/

Subscribe to our online digital library for full access to over 7,000 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 six years. He often speaks at international conferences about cloud technologies, and 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 client engagement sessions. His expertise includes strategic design and architectural mentorship, assessments, POCs, sales life cycles and consulting on engagement processes. 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

Microsoft provides organisations with an effective way of managing their network through Azure's networking services. Whatever the size of your organisation, 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 Azure 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 organisations.

Who this book is for

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

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 **Virtual Machines** (**VMs**) and the network interface that is used as a connection between Azure VMs and Azure VNets.

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

Chapter 4, *Managing IP Addresses*, covers types of IP addresses, both 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 a given Azure VNet and a local network.

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

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

Chapter 8, *Load Balancers*, supports 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.

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

Chapter 10, *Azure Application Gateway*, is essentially about load balancers for web traffic, but this feature in Azure also allows you better traffic control. Where classic load balancers operate on the transport layer, application gateways allow you to route traffic based on protocol (TCP or UDP) and IP address, mapping IP addresses and protocols in the frontend to IP addresses and protocols 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 machine 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 `http://www.packt.com`. If you purchased this book elsewhere, you can visit `http://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 `http://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 on-screen 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 catalogue of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the colour images

We also provide a PDF file that has colour 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 on the screen, for example, in menus or dialog boxes. 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, `http://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 `http://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 Azure portal
- Creating a virtual network with PowerShell
- Adding a subnet in the Azure portal
- Adding a subnet with PowerShell
- Changing the address space size
- Changing a subnet's size

Technical requirements

For this chapter, the following is required:

- An Azure subscription
- Azure PowerShell

Code examples can be found at `https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter01`.

Creating a virtual network in the Azure 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.

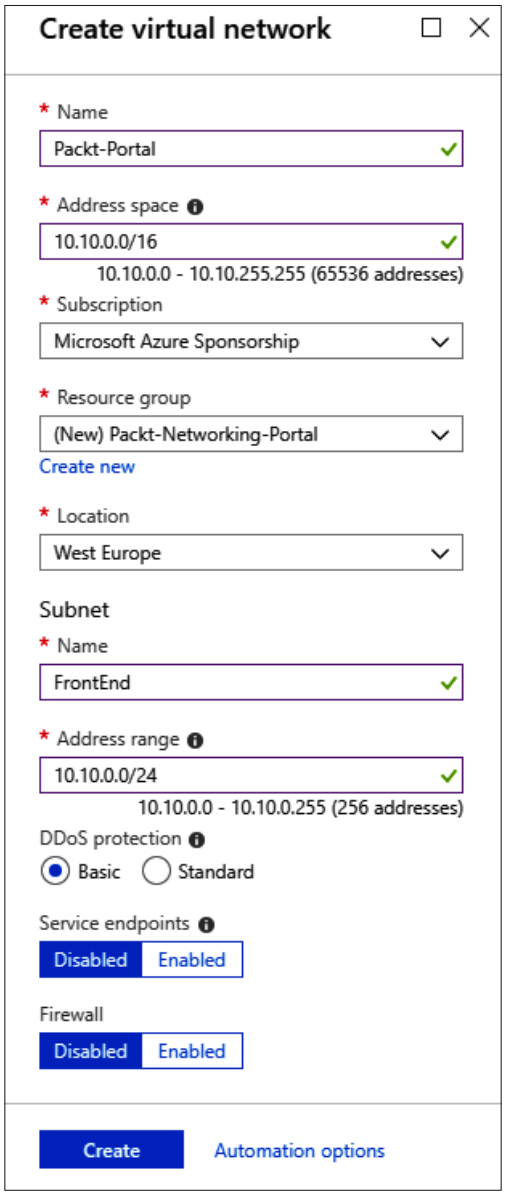
Getting ready

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, including a **Name**; define the **Address space**; select the **Subscription** option we want to use; select the **Resource group** option for where the virtual network will be deployed; select a **Location** (of the Azure datacentre) for where the virtual network will be deployed; and define the **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:



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 datacentre 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 datacentre. 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.

Afterwards, 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 an 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 a subnet with PowerShell* recipe.

Adding a subnet in the Azure portal

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

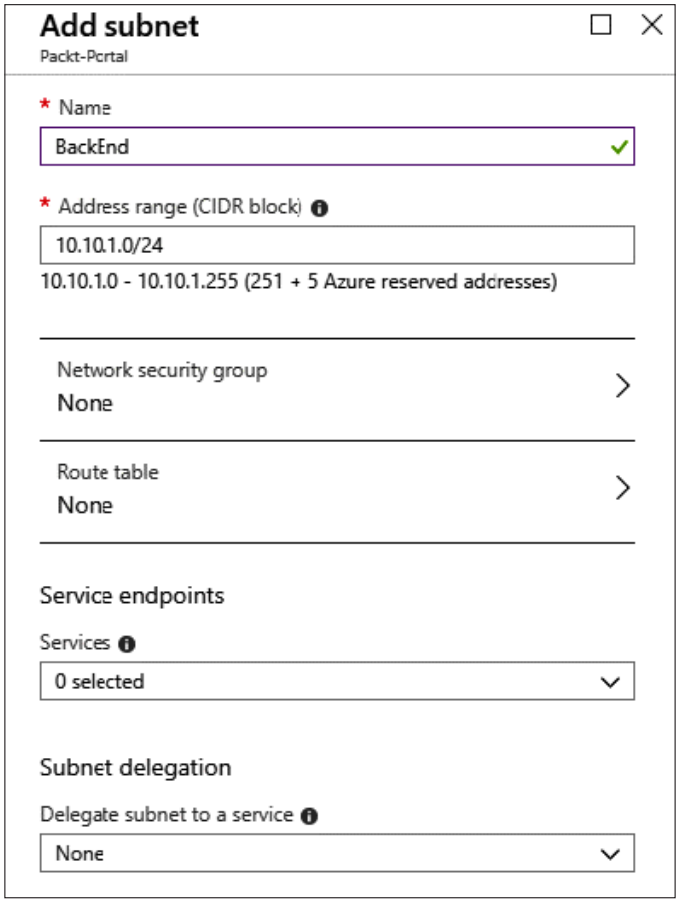
Getting ready

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...

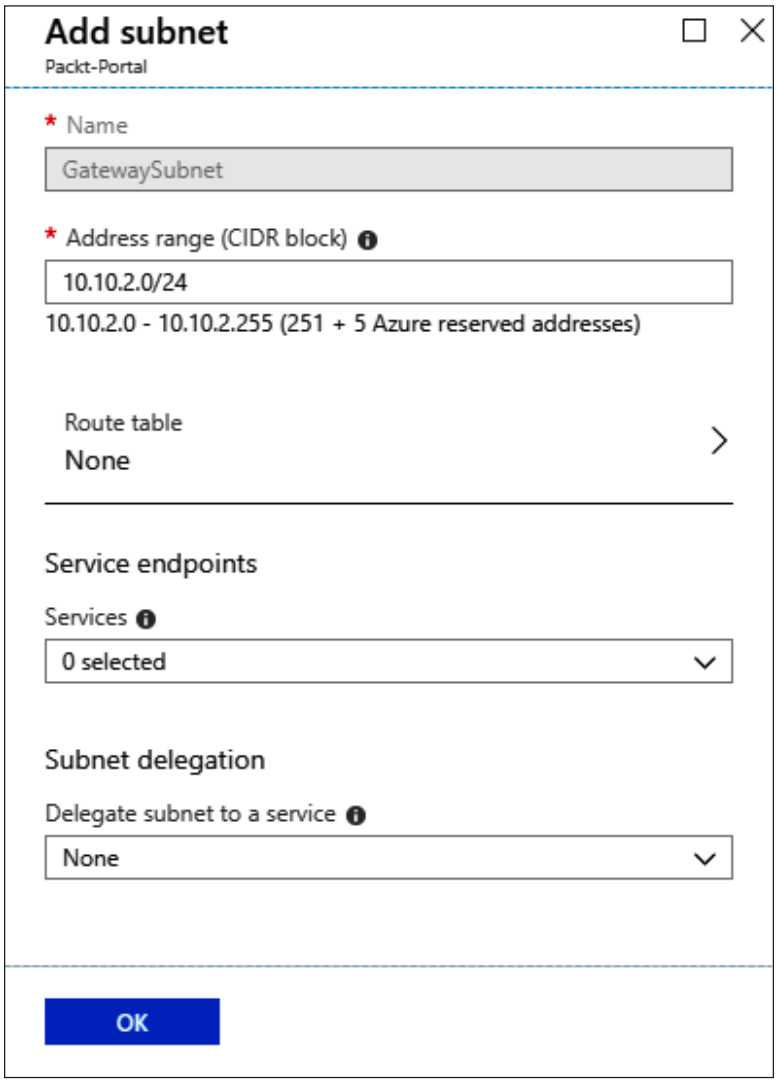
In order to add a subnet to a virtual network using the Azure portal, we must follow these steps:

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 the **Name** and **Address range** in CIDR format. The **Address range** must be in the range limit of the virtual network's 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:

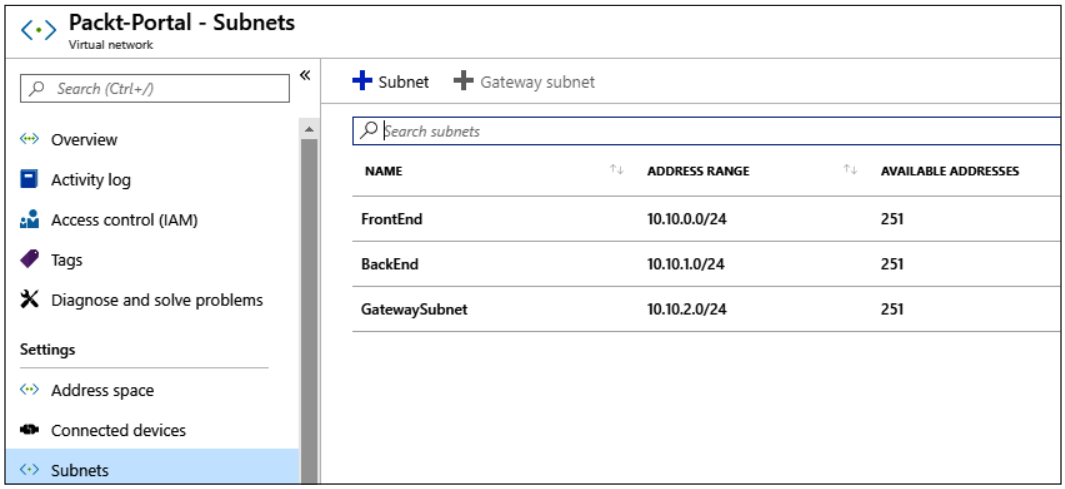


4. We can also add a gateway subnet in the same blade. To add a gateway subnet, select the **Gateway subnet** option.

   For a gateway subnet, the only parameter we need to define is the **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:



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



How it works...

A single virtual network can have a multiple number of subnets defined. Subnets can't overlap and must be within the virtual network's 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 **Virtual Private Network** (**VPN**) connections, and this will be covered in later chapters.

Adding a subnet with PowerShell

When creating an 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 subnets 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 with 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.

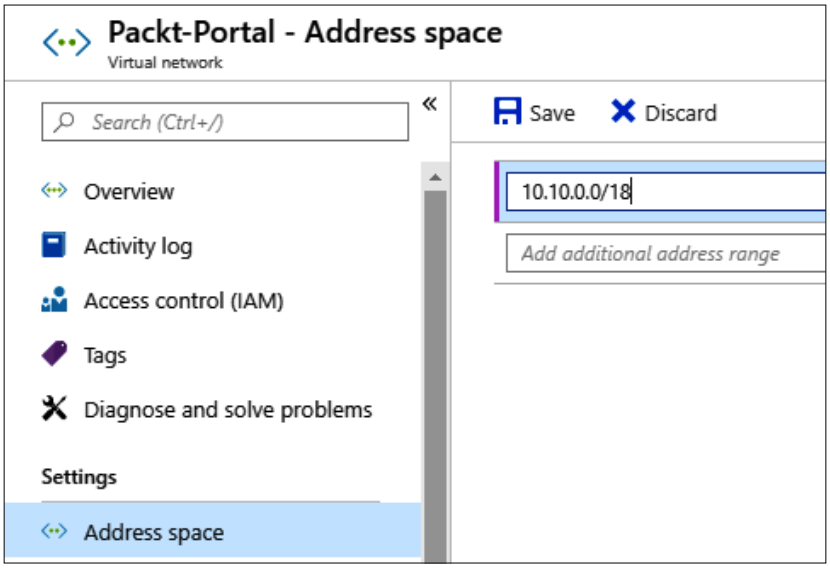
Getting ready

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 follow these steps:

1. In the virtual network blade, locate **Address space** under **Settings**.
2. 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 spaces can't be decreased if you have subnets defined in the address space that wouldn't be covered by the 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.

The address space can't be changed to a 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 a subnet's size

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

Getting ready

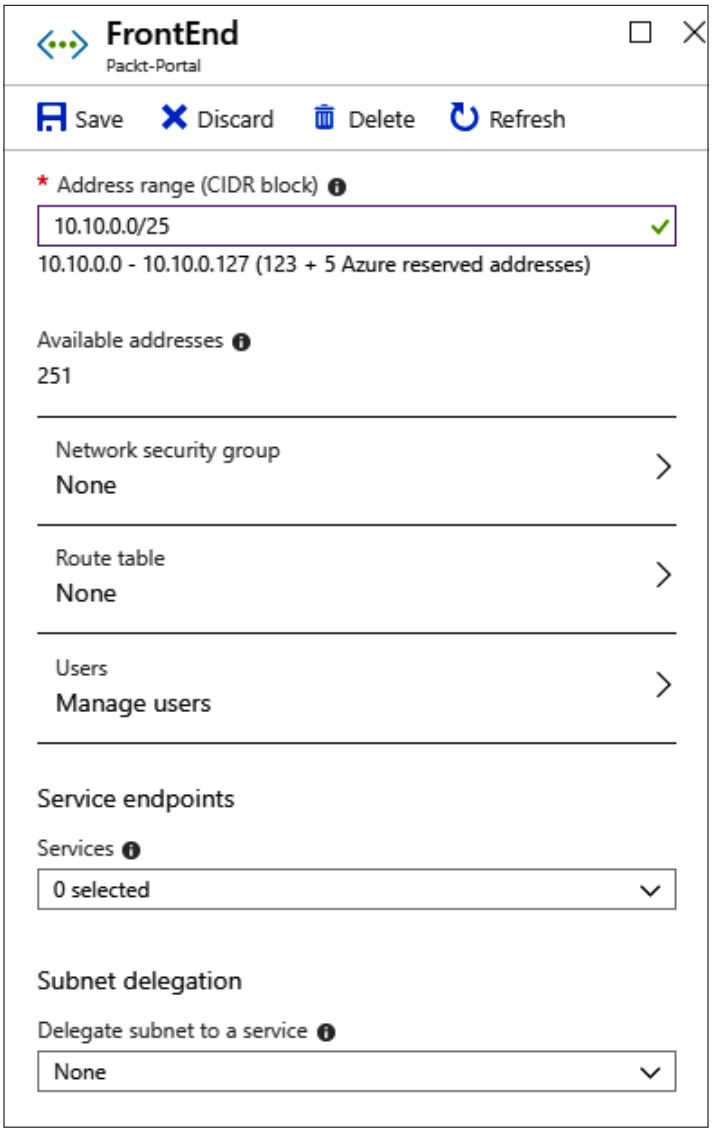
Before you start, open a web browser and go to the Azure portal, at `https://portal.azure.com`.

How to do it...

In order to change subnet size using the Azure portal, follow these steps:

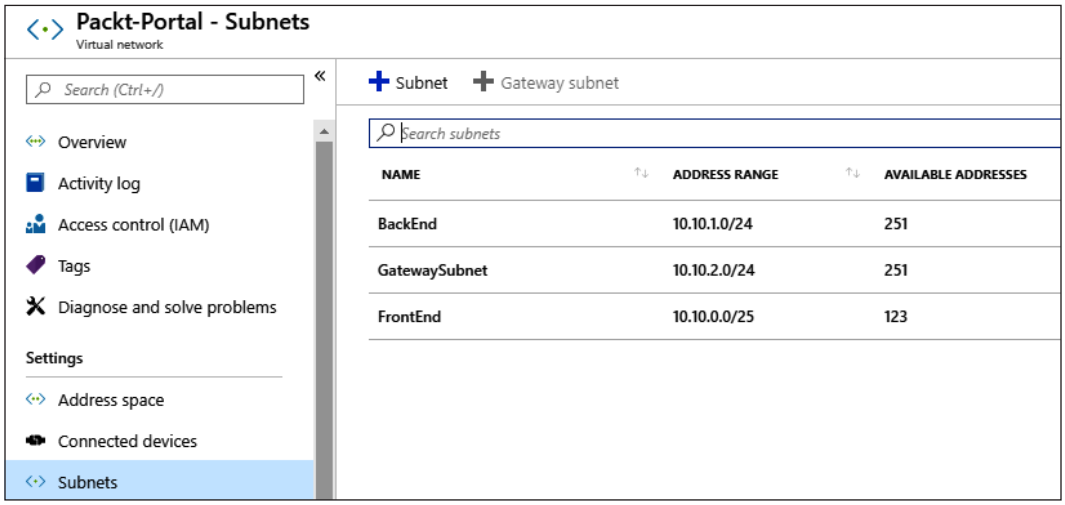
1. In the virtual network blade, select the **Subnets** option.
2. Select the subnet you want to change.

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 that the changes have been applied and the address space has changed, as shown in the following screenshot:



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's 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.

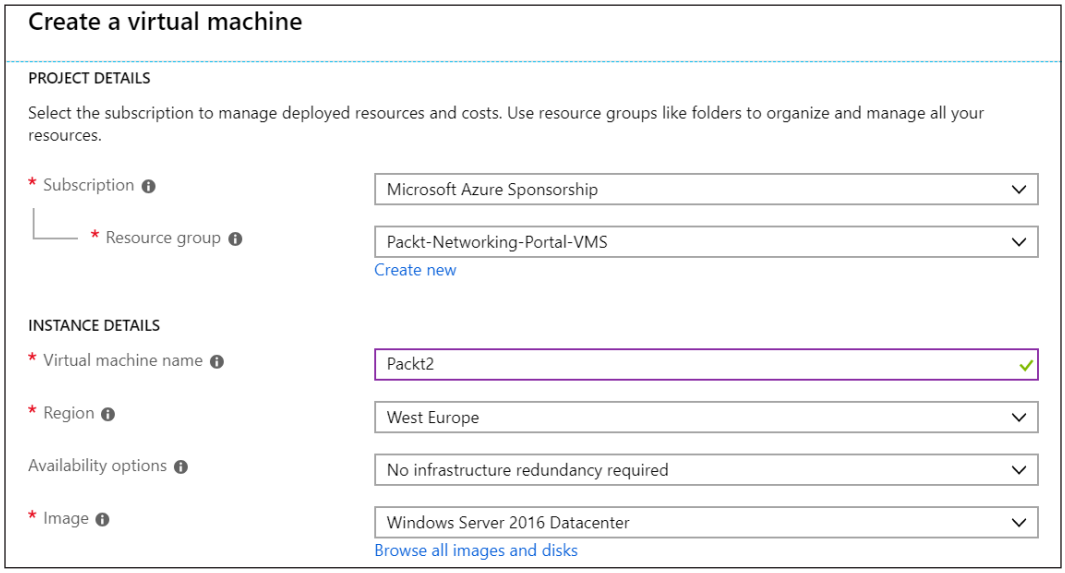
Getting ready

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

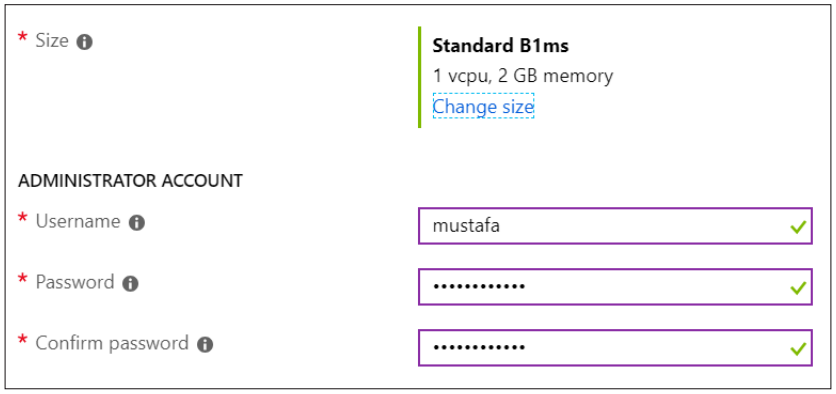
How to do it...

In order to create a new VM using the Azure portal, follow these steps:

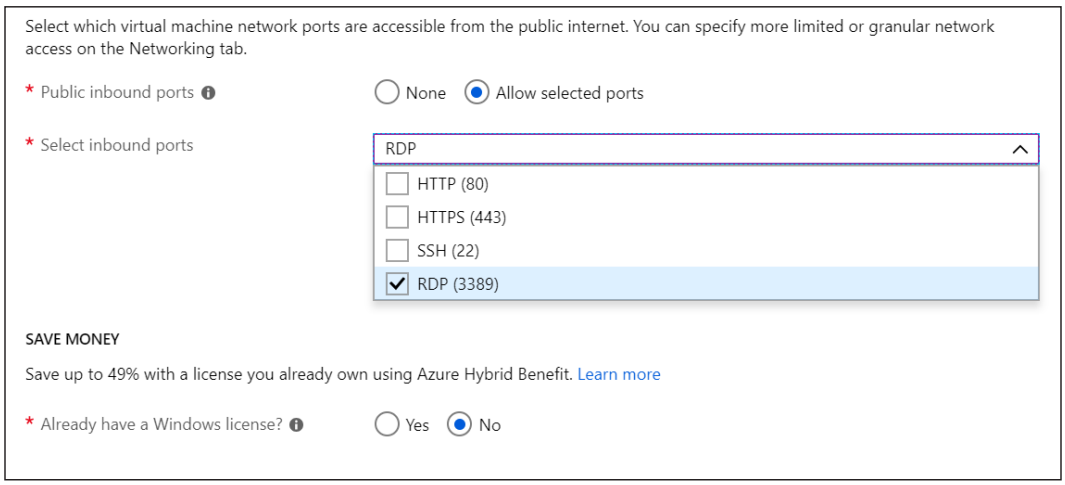
1. In the Azure portal, select **Create a resource** and choose **Windows Server 2016 VM** (or search for any VM image under the **Compute** section).
2. In the **Create a virtual machine** blade, we need to provide information for various options; not all of these are related to networking. First, we 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 options** and choose the appropriate option for **Image**. Example settings are shown in the following screenshot:



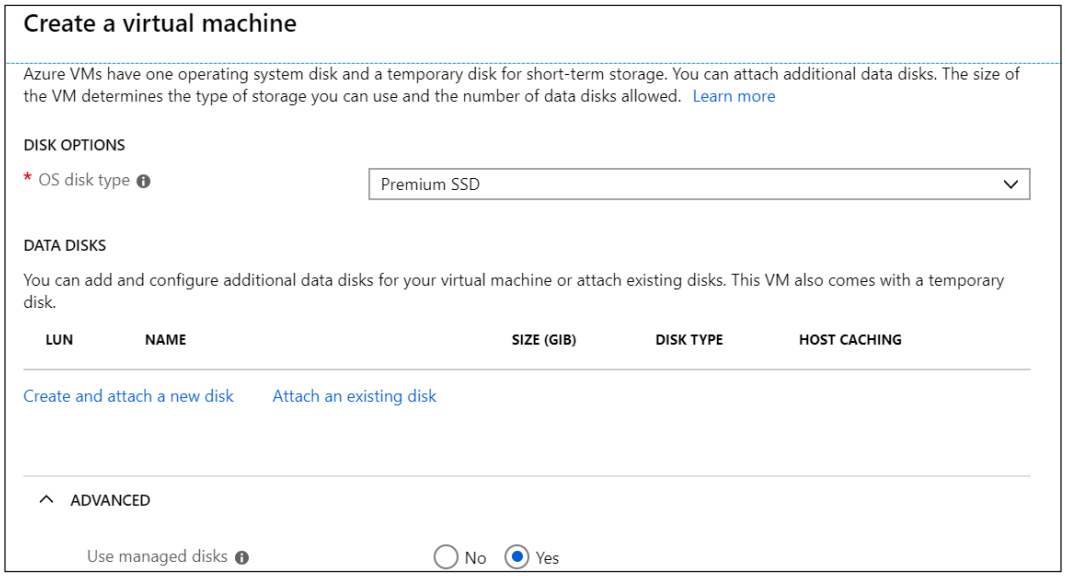
4. Next, we need to provide information on our VM's **Size**, **Username** and **Password**. Note that for the **Username**, you can't use names such as admin, administrator, sysadmin or root. The **Password** must be at least 12 characters long and satisfy three out of the four famous rules (that is,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



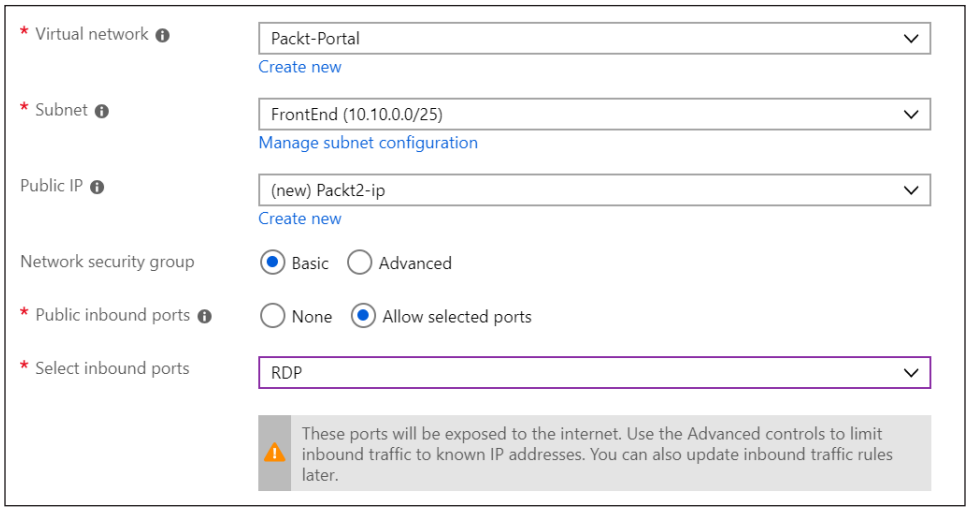
5. Next, we arrive at an option that concerns networking. We need to define whether we are going to allow any type of connection over a public IP address. We can select whether we want to deny all access, or allow a specific port. In the following example, I'm choosing **RDP (3389)**:



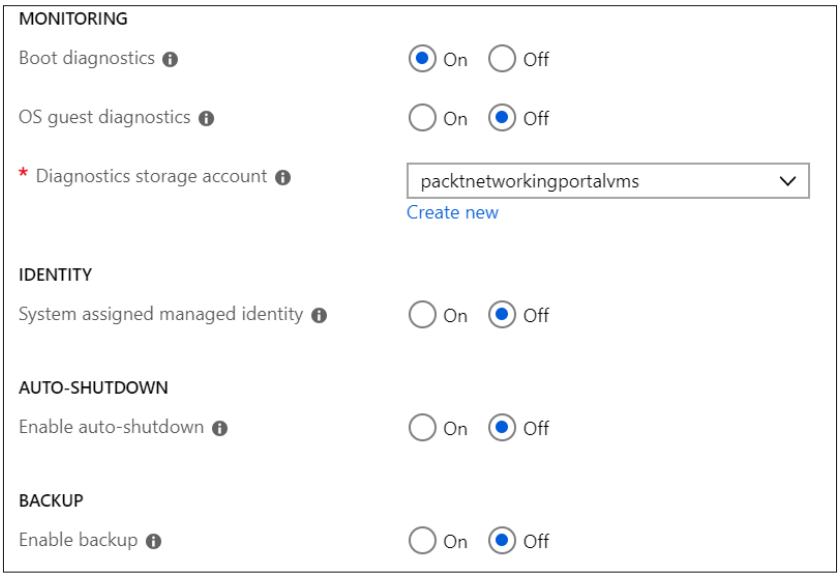
6. In the next section, we need to define disks. We can choose between **Premium SSD**, **Standard SSD** and **Standard HDD**. An OS disk is required and must be defined. We can attach additional data disks as needed. Disks can be added at a later time, as well. We can choose whether we are going to **Use managed disks** or not. My recommendation is 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:



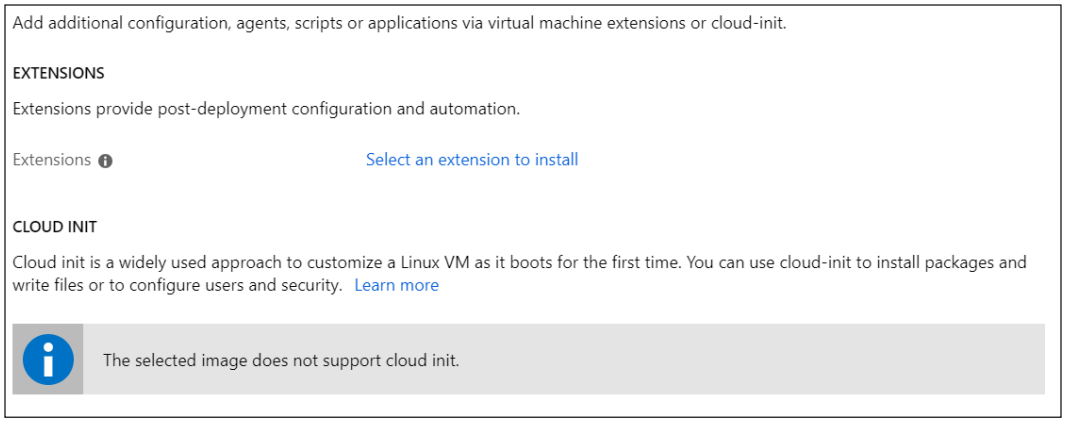
7. After defining disks, we get to the networking settings. Here, we need to define the **Virtual network** and **Subnet** options that the VM will use. These two options are mandatory. You can choose to assign the **Public IP** address to the VM (you can choose to disable the **Public IP** address,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



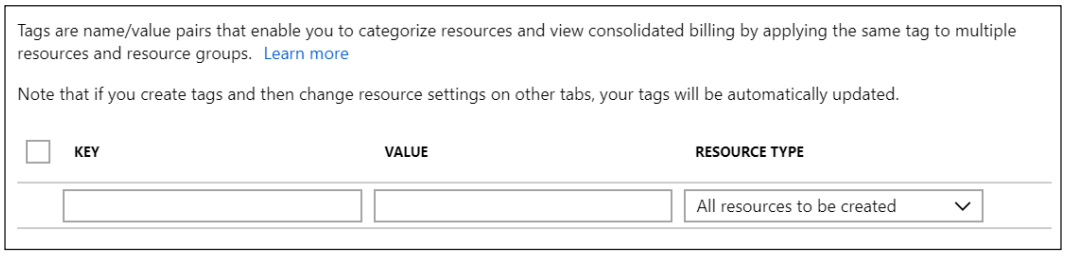
8. After the networking section, we need to set up **MONITORING**. **Boot diagnostics** are enabled by default, and you can enable additional features as needed. The default settings for **MONITORING** are shown in the following screenshot:



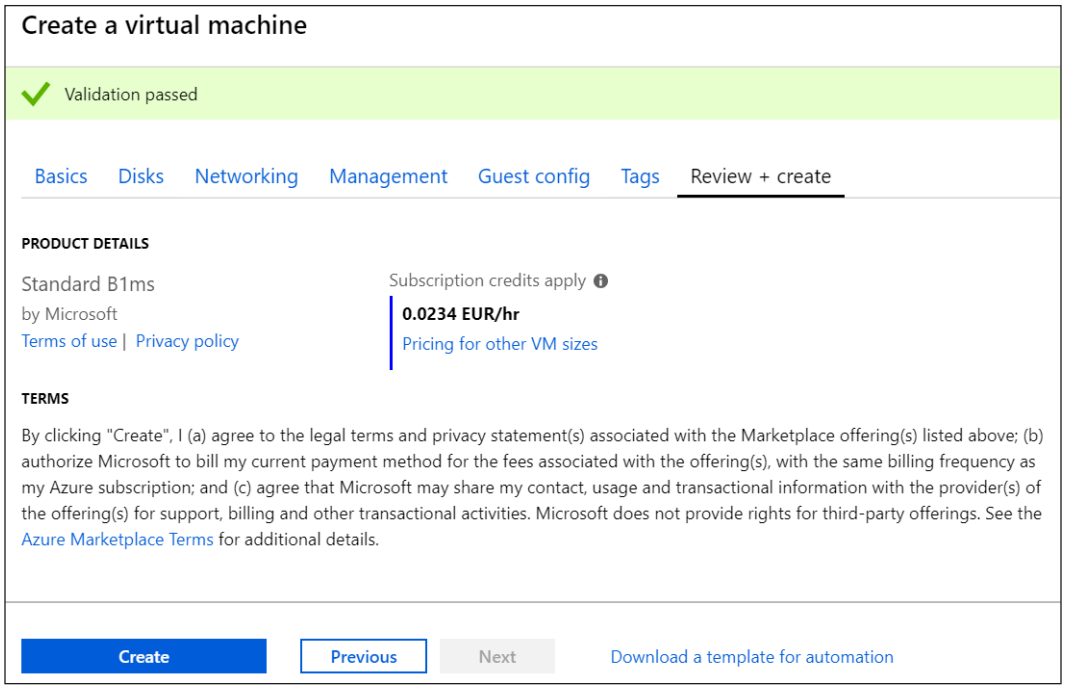
9. In **EXTENSIONS**, we can set up post-deployment configuration steps by adding software installations, configuration scripts, and more. The **EXTENSIONS** screen is shown in the following screenshot:



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



11. After all settings are defined, we get to the 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:



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 the same network. Additionally, NICs and VMs can be assigned public IP addresses as well. A 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*, from 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.

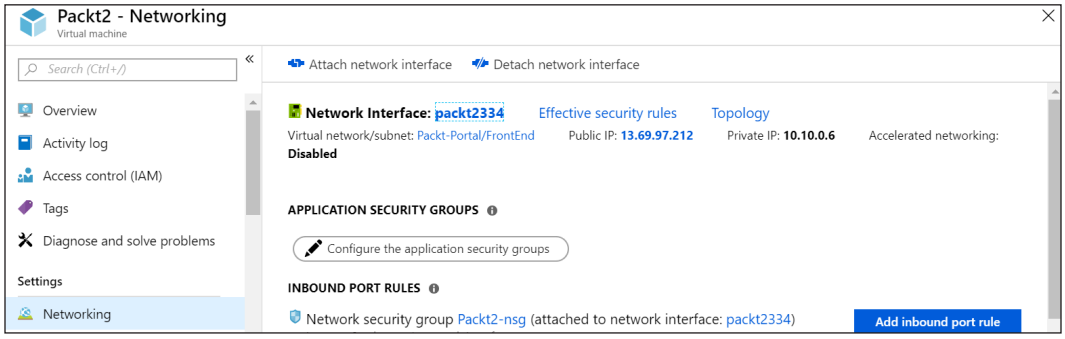
Getting ready

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

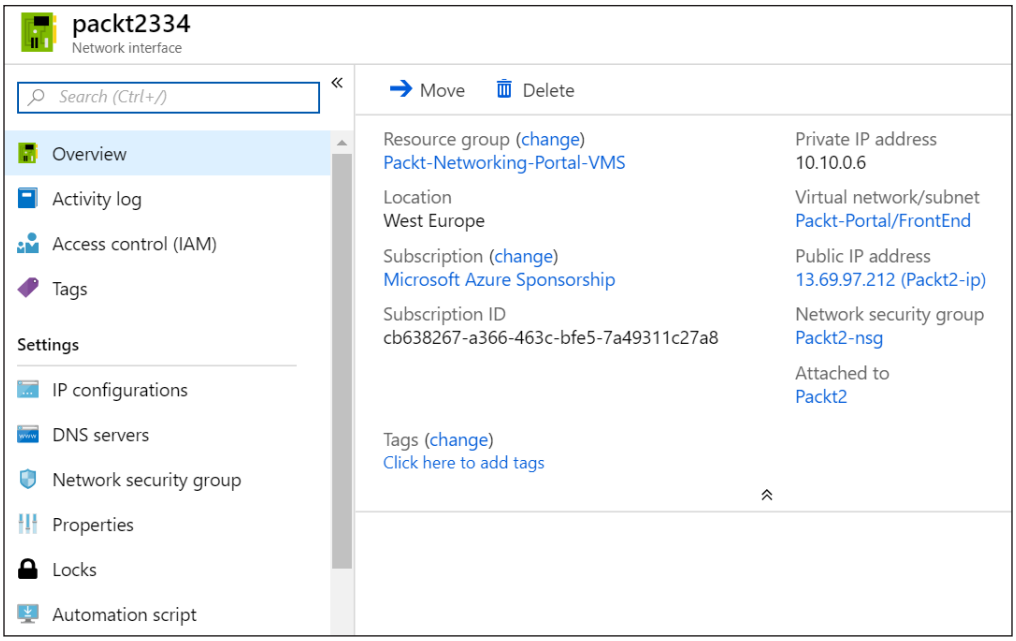
How to do it...

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** and **DNS servers**. The NIC view is shown in the following screenshot:



How it works...

Networking information is displayed in several places, including in 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

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

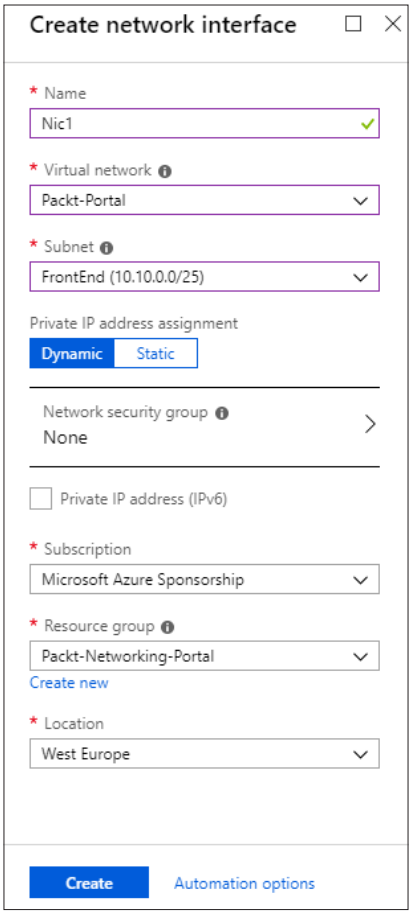
Getting ready

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, follow these steps:

1. In the Azure portal, select **Create a resource** and choose **Network interface** under **Networking** (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 the **Subscription**, **Resource group** and **Location** and NICs are no exception. The information needed to create a new NIC is shown in the following screenshot:



How it works...

An NIC can't exist without network association, and this association 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 NICs. Because of this, we can add a new NIC at any time.

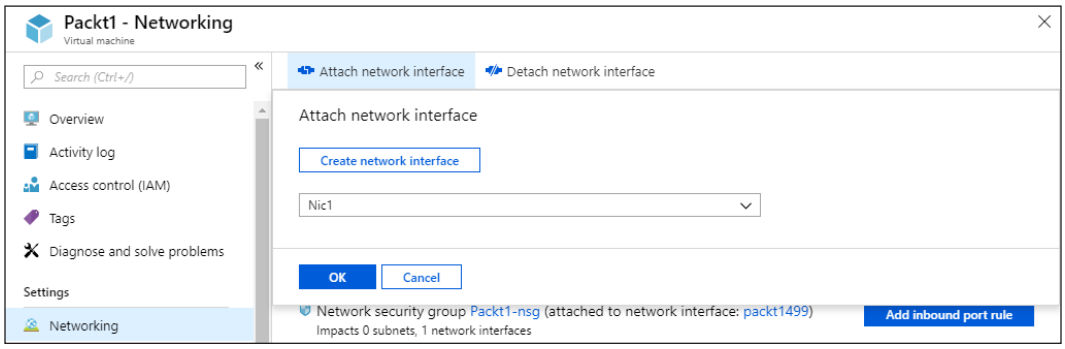
Getting ready

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 NIC 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 **Attach network interface** option.
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 NICs. 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 NIC, we can detach NIC at any time and attach it to another VM.

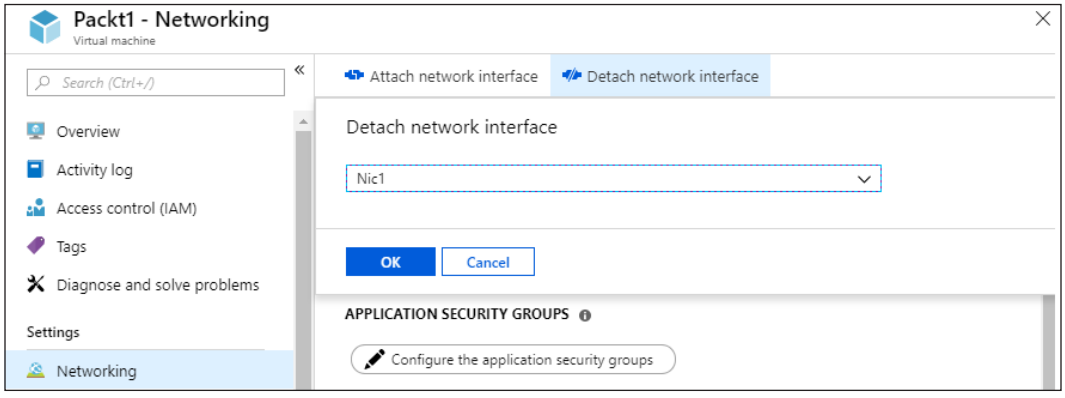
Getting ready

Before you start, open a web browser and got 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. At the top of the **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 NIC, 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 access to specific resources or subnets in Azure. An NSG can be associated with either a subnet (by applying security rules to all resources associated with the subnet) or a **network interface** (**NIC**) (by applying security rules only to the **virtual machine** (**VM**) associated with the NIC).

We will cover the following recipes in this chapter:

- Creating a new NSG in the Azure 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 a network interface
- Assigning an NSG with PowerShell
- Creating an Application Security Group (ASG)
- Associating an ASG with a VM
- Creating rules with an NSG and an ASG

Technical requirements

For this chapter, the following is required:

- An Azure subscription
- Azure PowerShell

Code examples can be found at `https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter03`.

Creating a new NSG in the Azure 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.

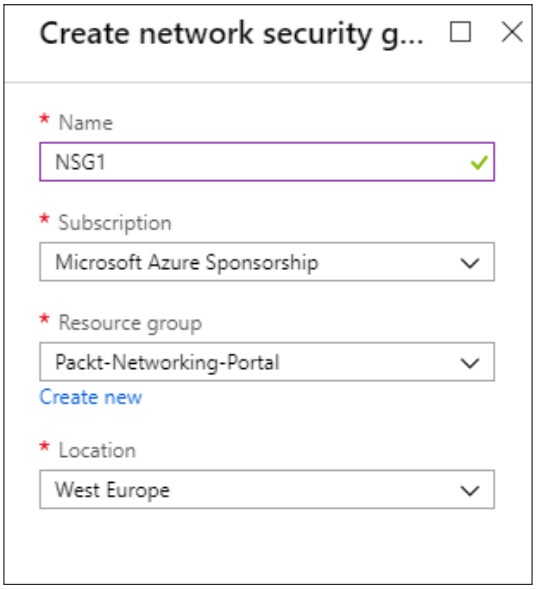
Getting ready

Before you start, open your browser and go to the Azure portal at `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 VM deployment. This will associate the NSG with 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 the 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 a new 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.

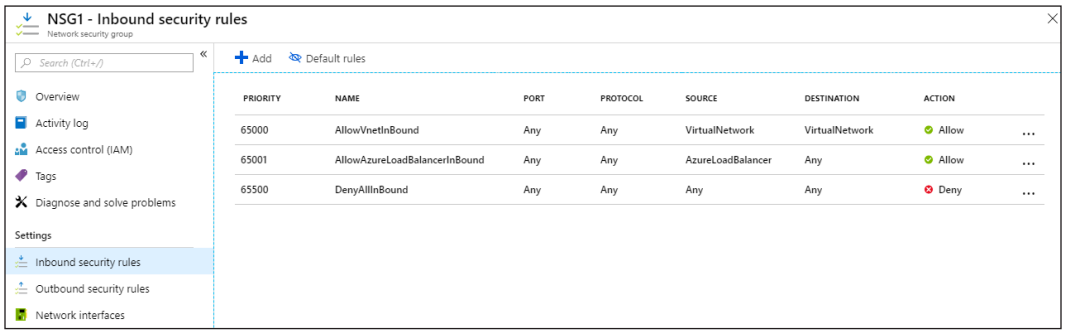
Getting ready

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

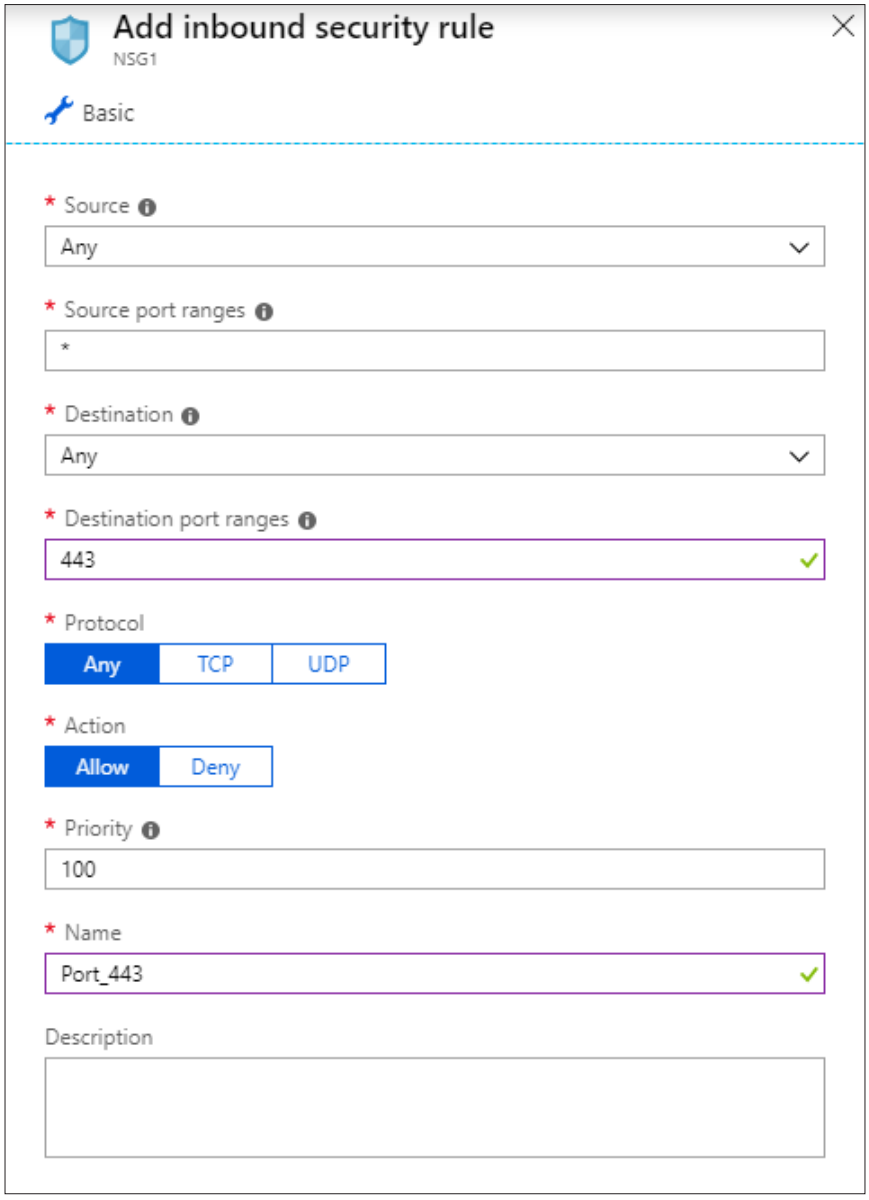
How to do it...

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:



3. In the new blade, we need to provide information for the **Source** (location and port), **Destination** (location and port), **Protocol**, **Action**, **Priority**, **Name** and **Description** fields. 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 port `443` (thus 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 (that is, the ones with the lower number) are processed first, so if you have two rules, with one denying traffic and one allowing it, the one with higher priority will take precedence, 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.

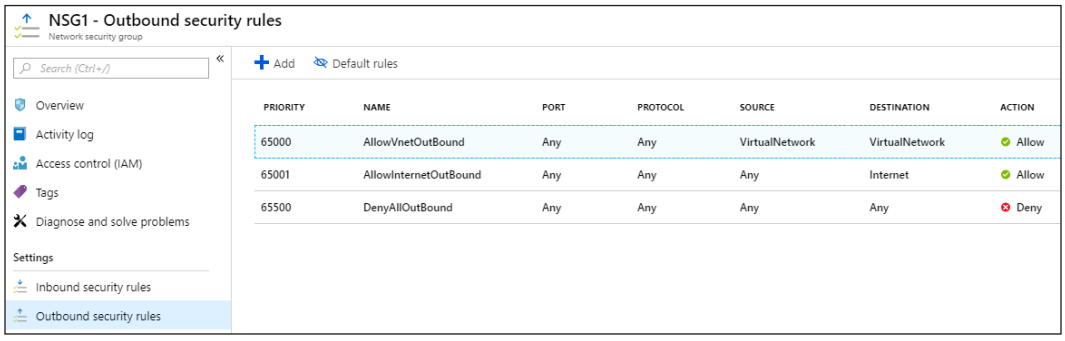
Getting ready

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

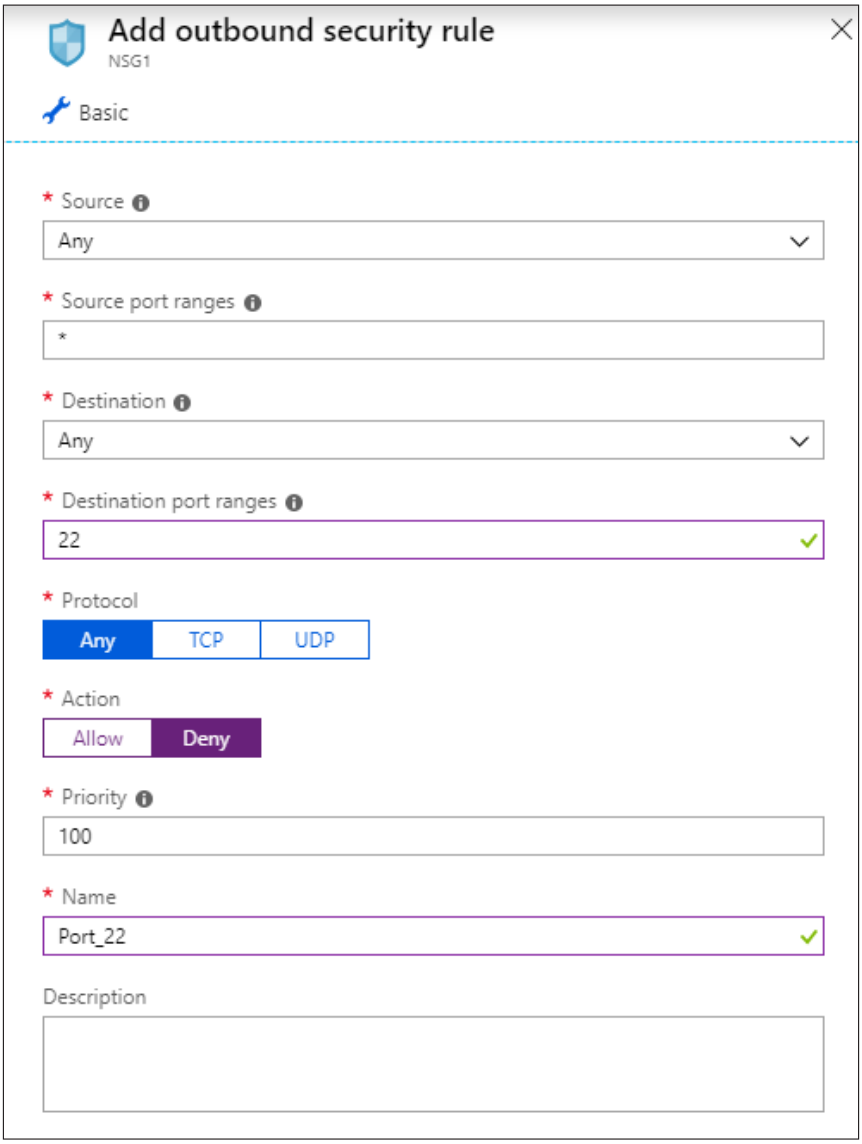
How to do it...

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:



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 port `22` 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 over 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 (the ones with the lower number) are processed first, so if you have two rules, with one denying traffic and one 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 an NSG in a single script. This way, we can standardise 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 whether we want to allow traffic or deny it. The direction parameter, which can be inbound or outbound, determines whether 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, 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.

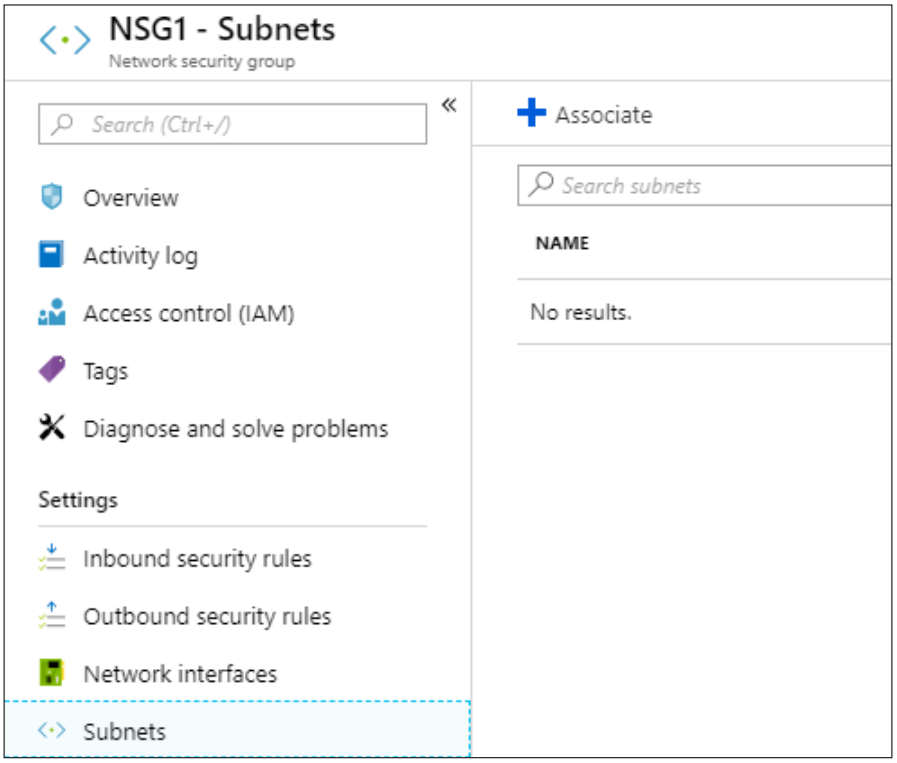
Getting ready

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

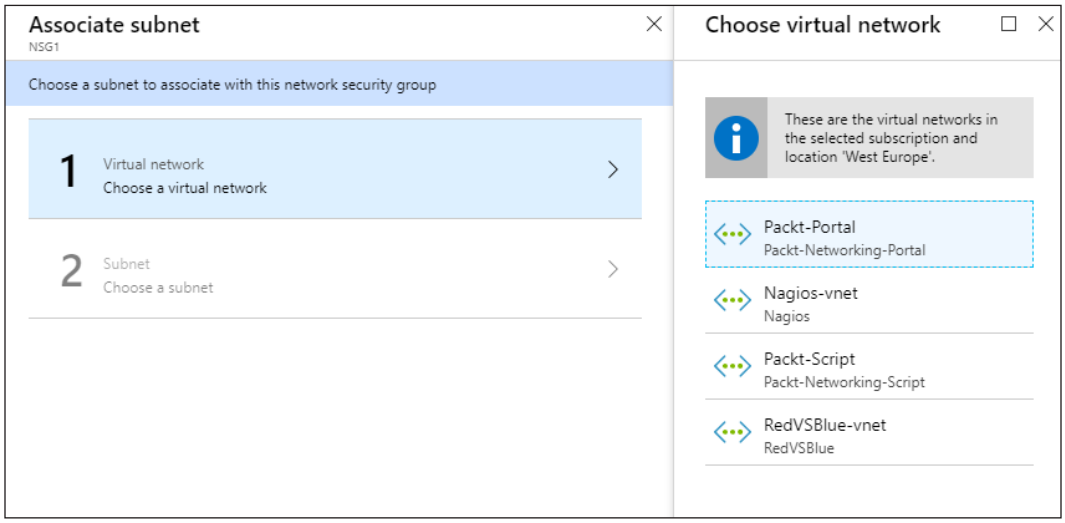
How to do it...

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

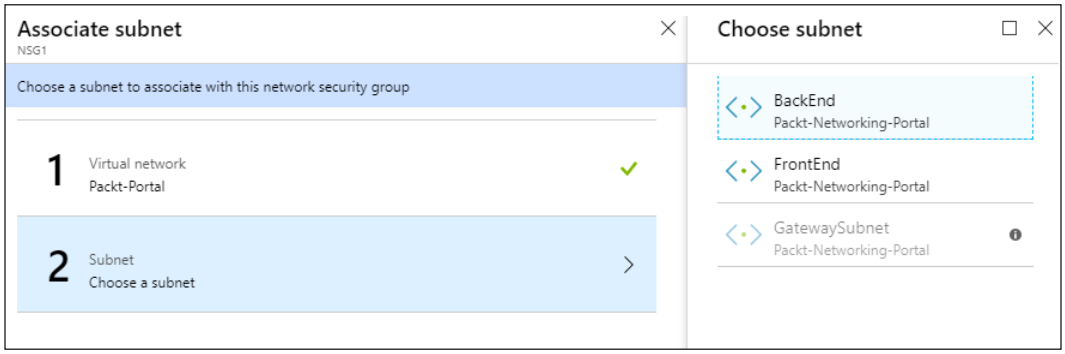
1. In the NSG blade, locate the **Subnets** option under **Settings**.
2. Click on the **Associate** button at the top of the page and wait for the new blade to open:



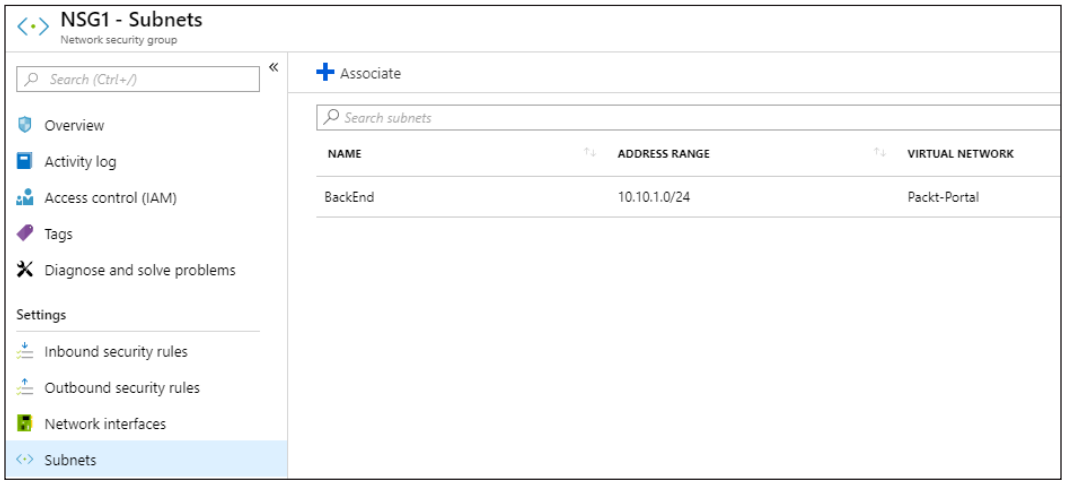
3. In the new blade, first select **Virtual network**, which contains the subnet you want to associate the NSG with:



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



5. After submitting the change, the subnet will appear in the list of associated subnets:



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 a single NSG, but when the rules from more NSGs are observed, the `Deny` rule will prevail. So, if we have two subnets, one with `Allow` on port `443` 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.

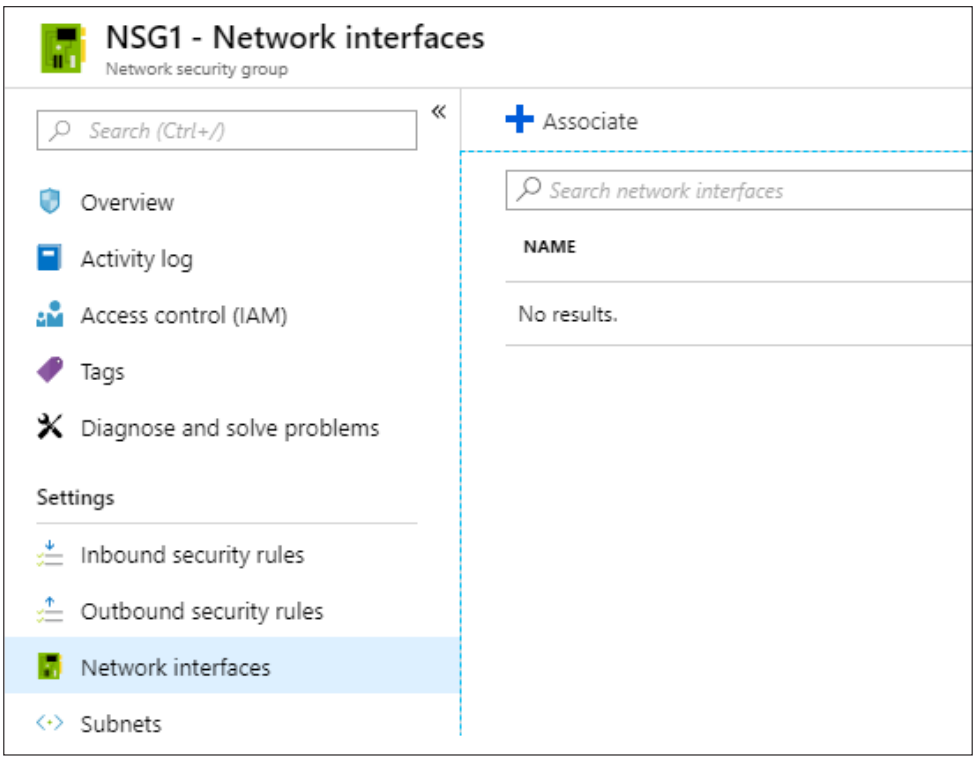
Getting ready

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

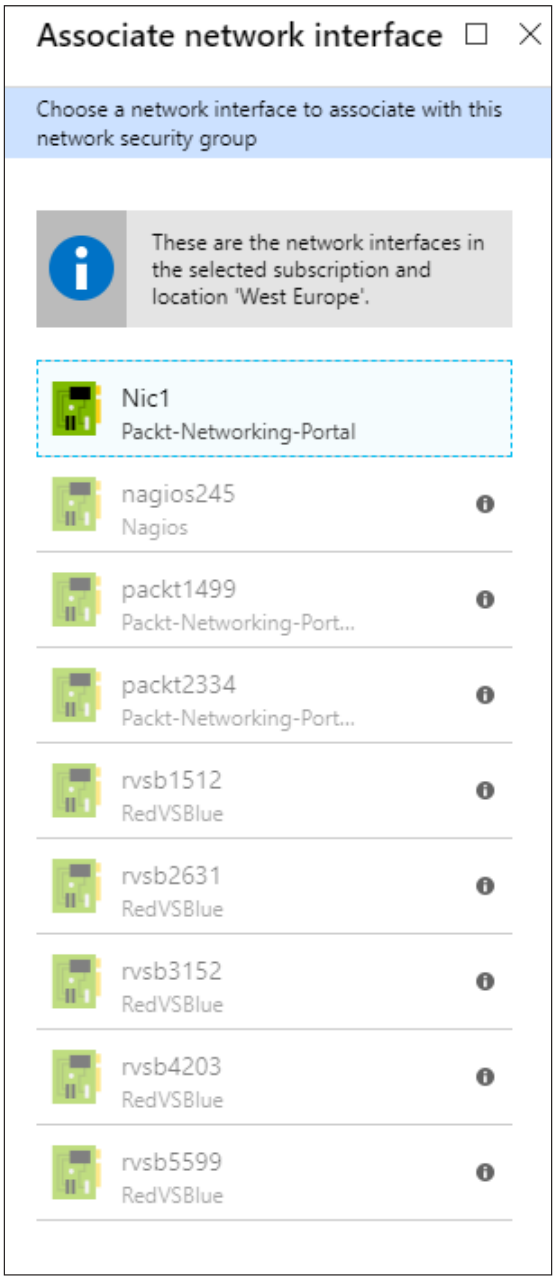
How to do it...

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 an 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 an NIC can be associated with another NSG (or even multiple NSGs). 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 take 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.

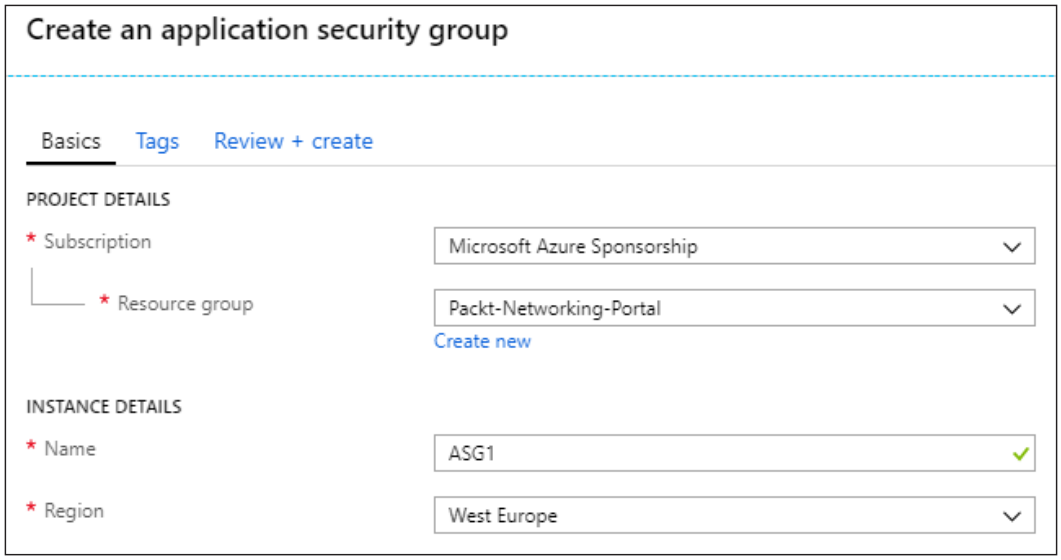
Getting ready

Before you start, open your browser and go to the Azure portal at `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:



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, applying 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.

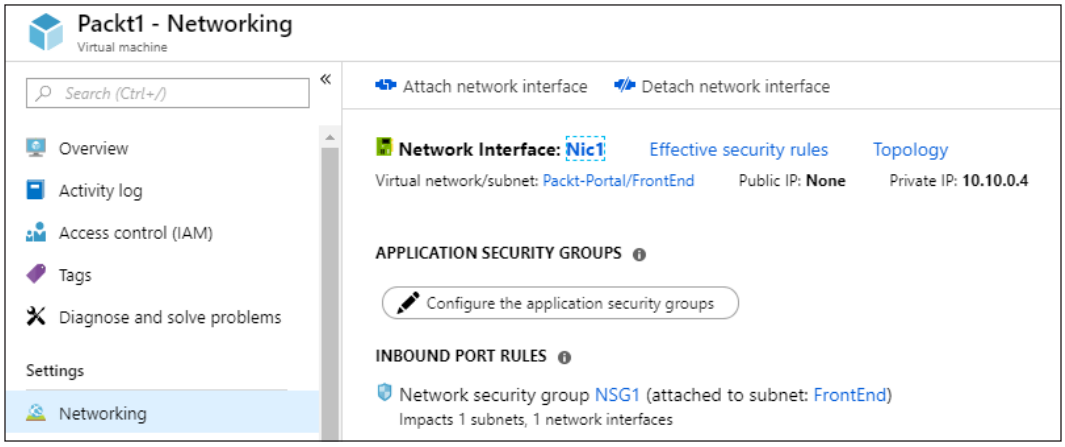
Getting ready

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

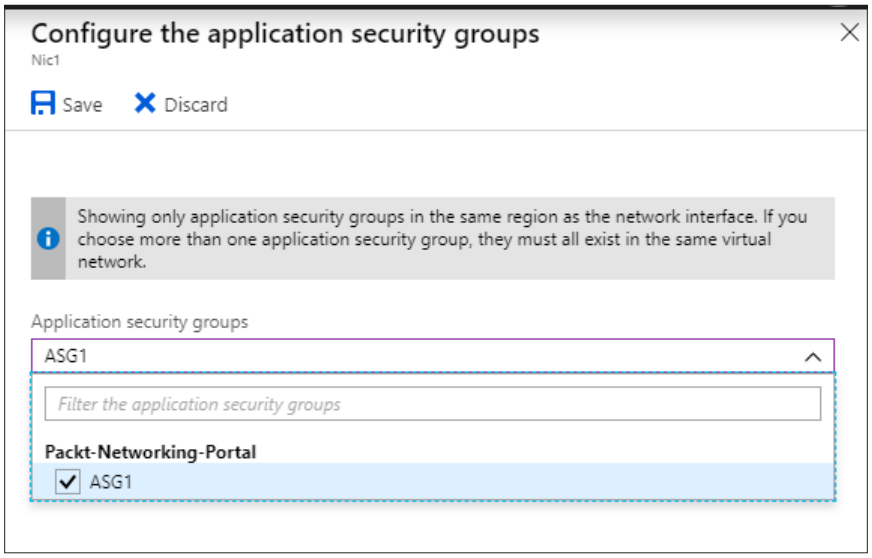
How to do it...

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 applying the 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 also only based on whether or not the resource is part of the ASG.

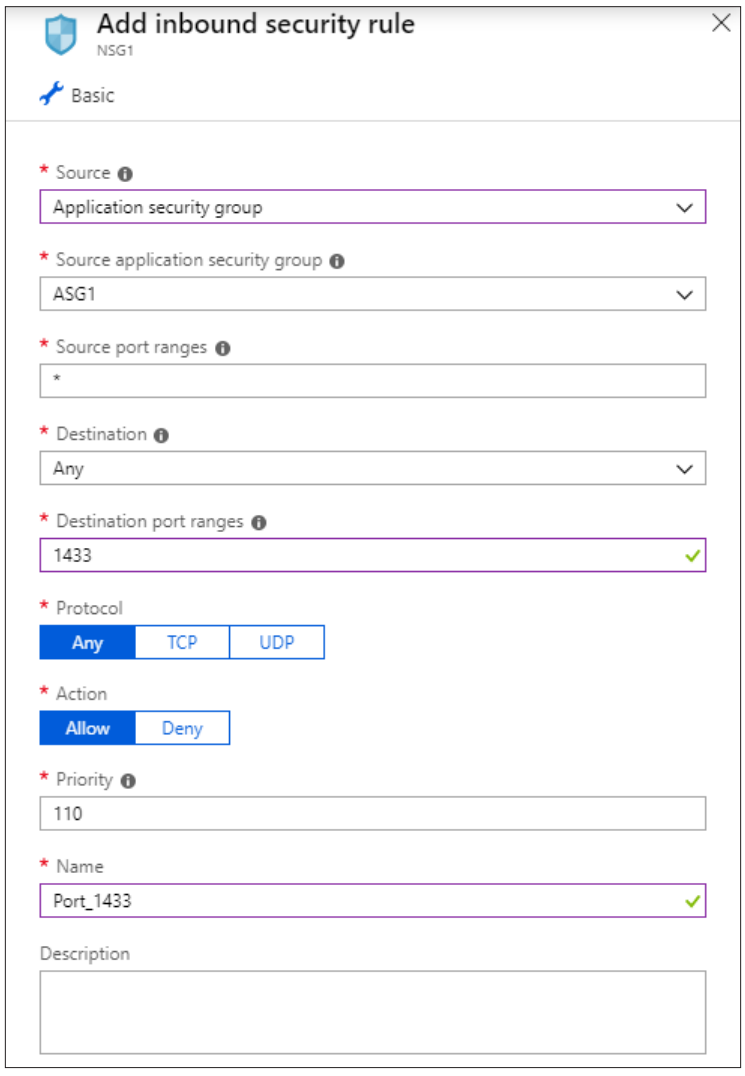
Getting ready

Before you start, open your browser and go to the Azure portal at `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 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** (**VM**), 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.

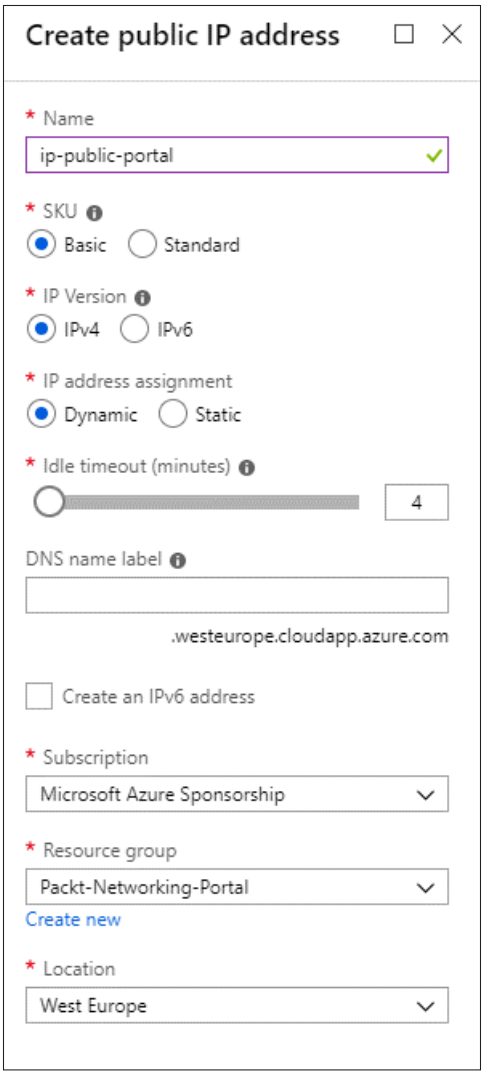
Getting ready

Before you start, open your browser and go to the Azure portal at `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:



How it works...

The **stock keeping unit** (**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 **Network Security Groups** (**NSGs**)) and **Standard** is zone-redundant. Another difference is that a standard SKU public IP address has a static assignment, while a basic SKU can be either static or dynamic.

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

The DNS name label is optional; 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 to 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. The settings in this case will be a basic SKU dynamic assignment, IPv4 version and without a DNS label. Furthermore, we can use additional switches such as `-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 already-existing resource. If the resource is no longer in use or has been 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.

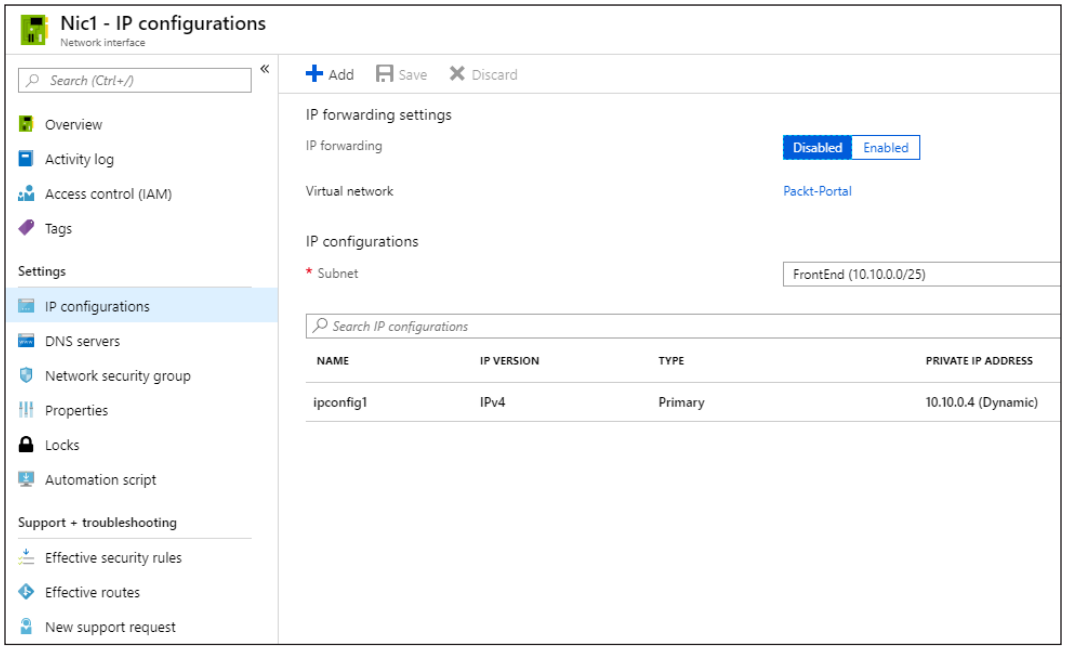
Getting ready

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

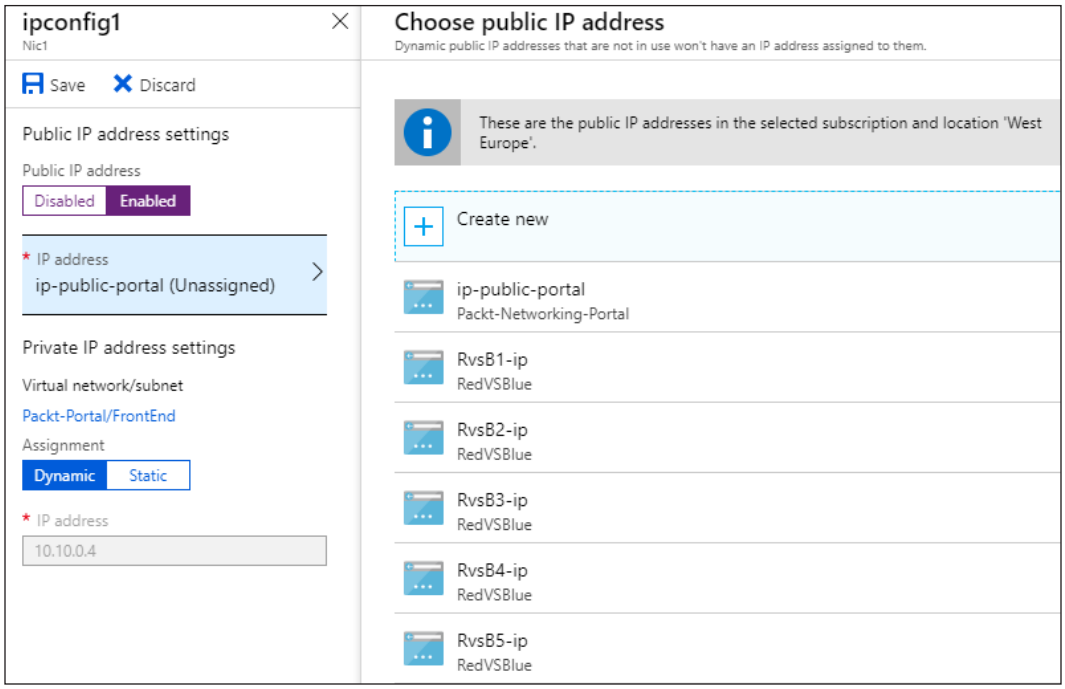
How to do it...

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

1. Locate the **network interface** (**NIC**) 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 shown in the following screenshot:



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:



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 (remember that 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 VM, 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

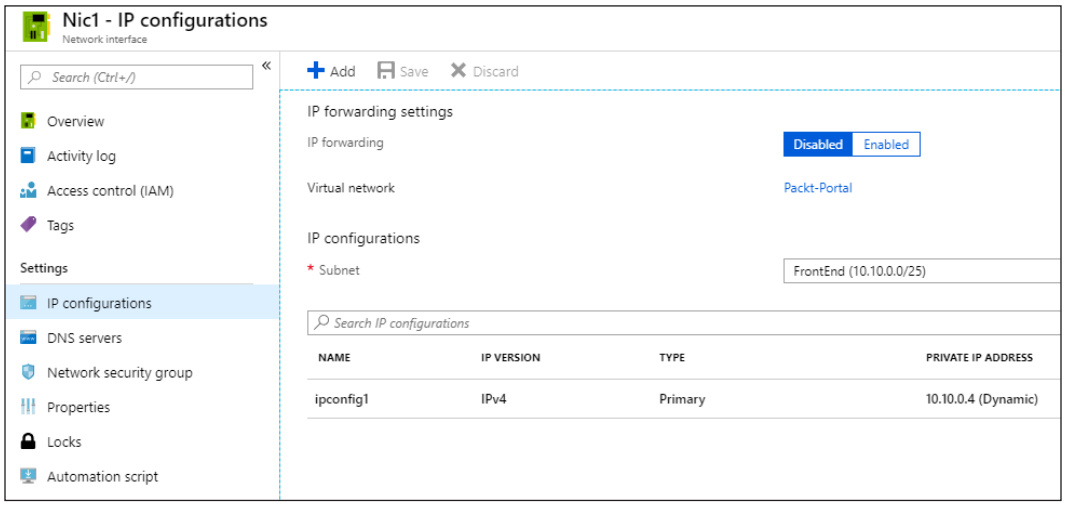
A public IP address can be unassigned from a resource in order to be saved for later use, or assigned to another resource. When a resource is deleted or decommissioned, we can still put the public IP address to use and assign it to the next resource.

Getting ready

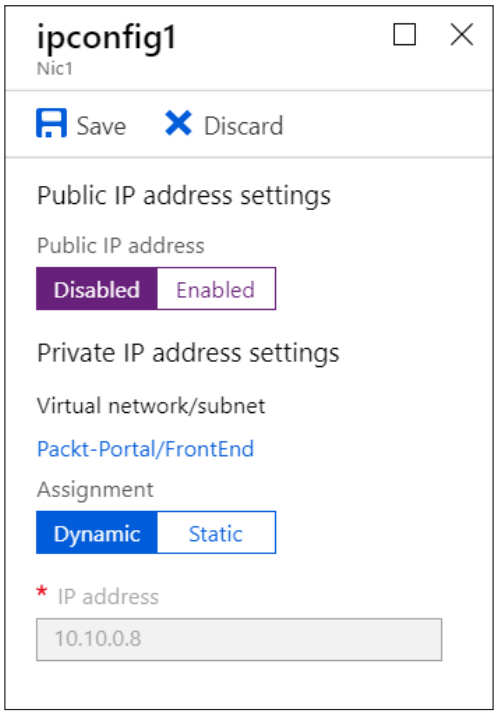
Before you start, open your browser and go to the Azure portal at `https://portal.azure.com`. Make sure that the VM using a public IP address is not running.

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:



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 dynamic IP assignment. This can be changed during the public IP address creation, or later. If this is changed from dynamic, then the public IP address becomes reserved (or static).

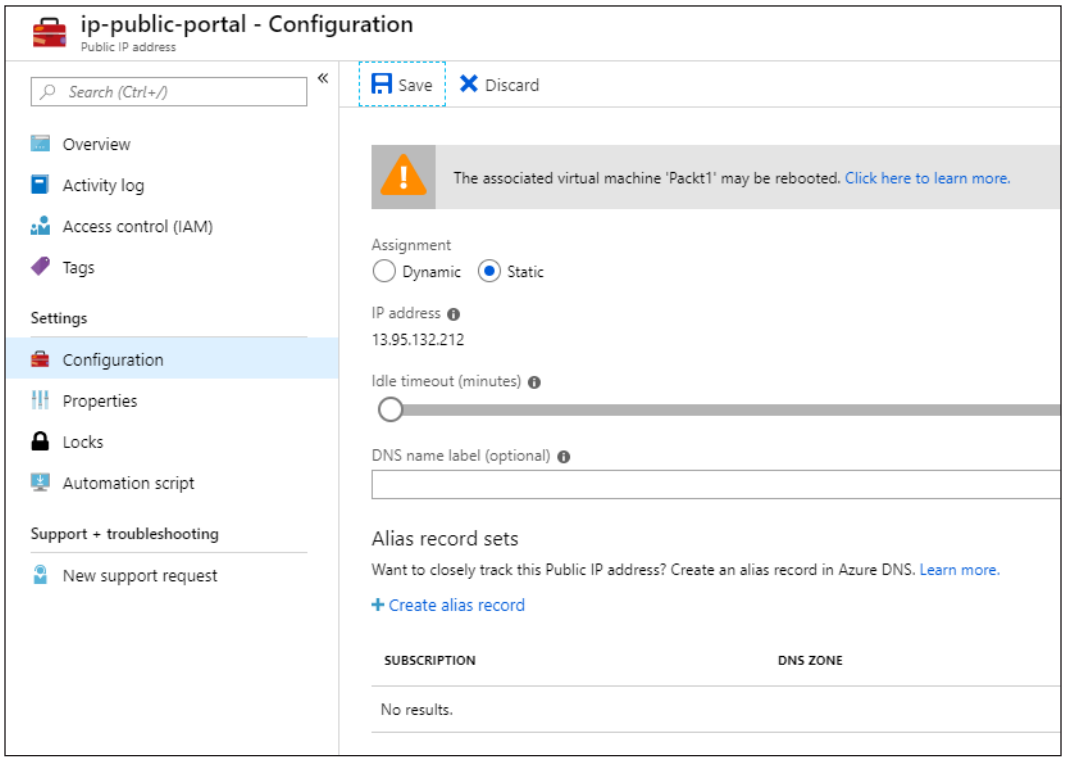
Getting ready

Before you start, open your browser and go to the Azure portal at `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 if 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.

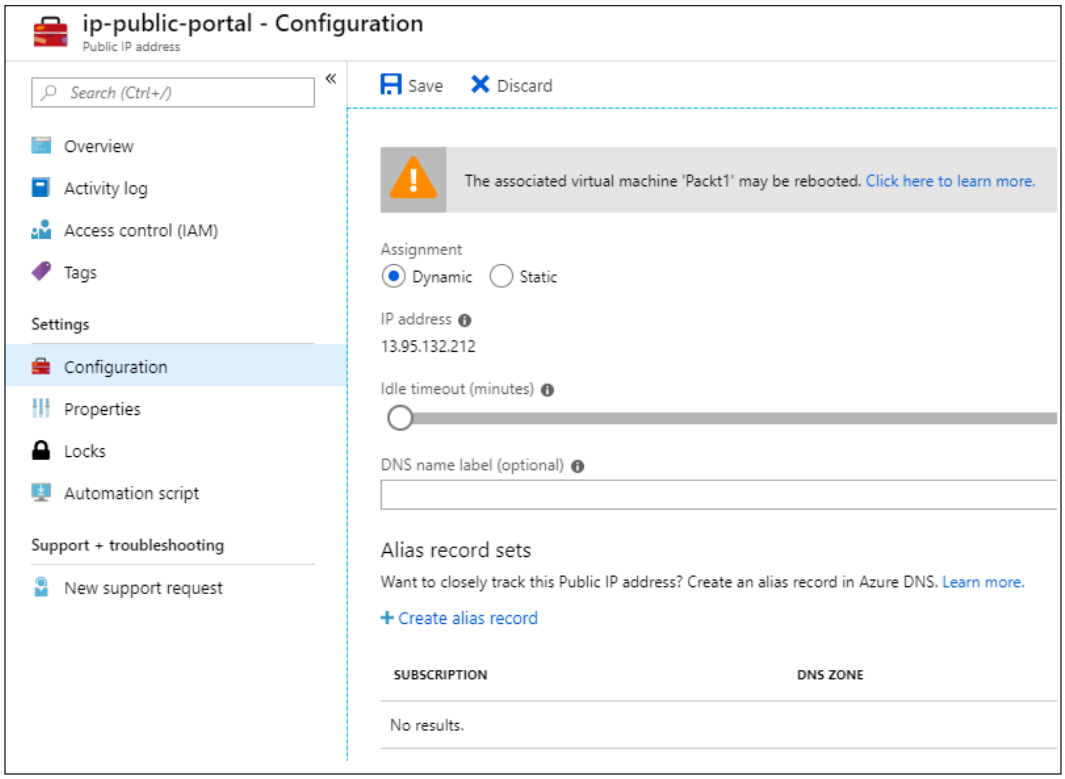
Getting ready

Before you start, open your browser and go to the Azure portal at `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**:



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. 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 being 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.

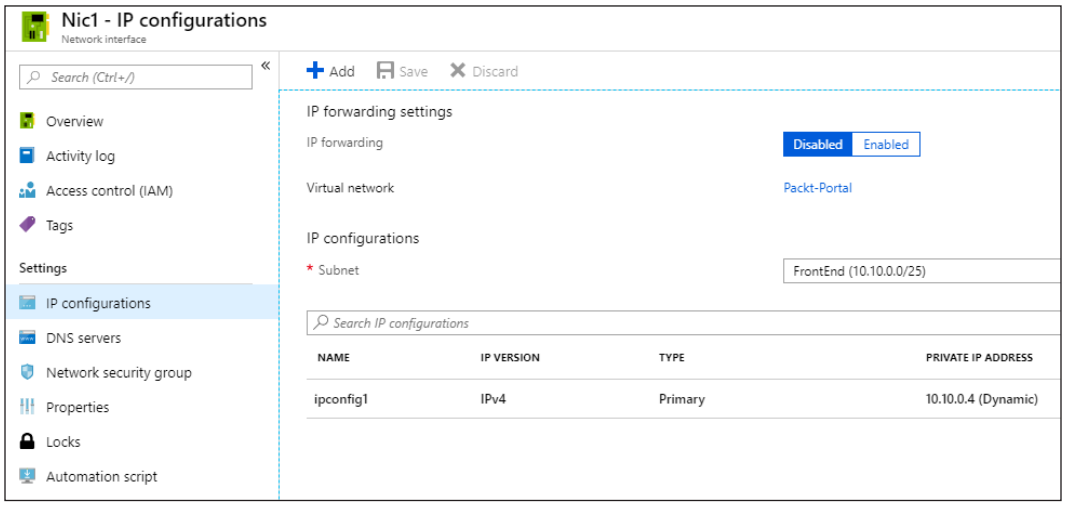
Getting ready

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

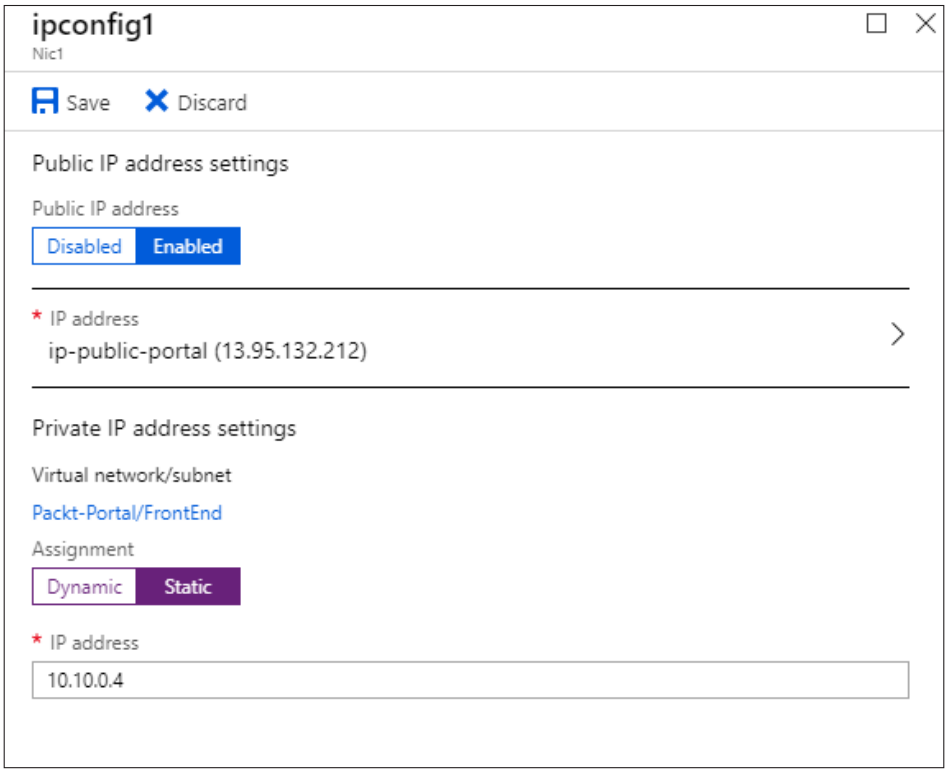
How to do it...

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:



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 aren't able to change the value of public IP addresses. With a private IP address, you can change the value to another IP address from the address space.

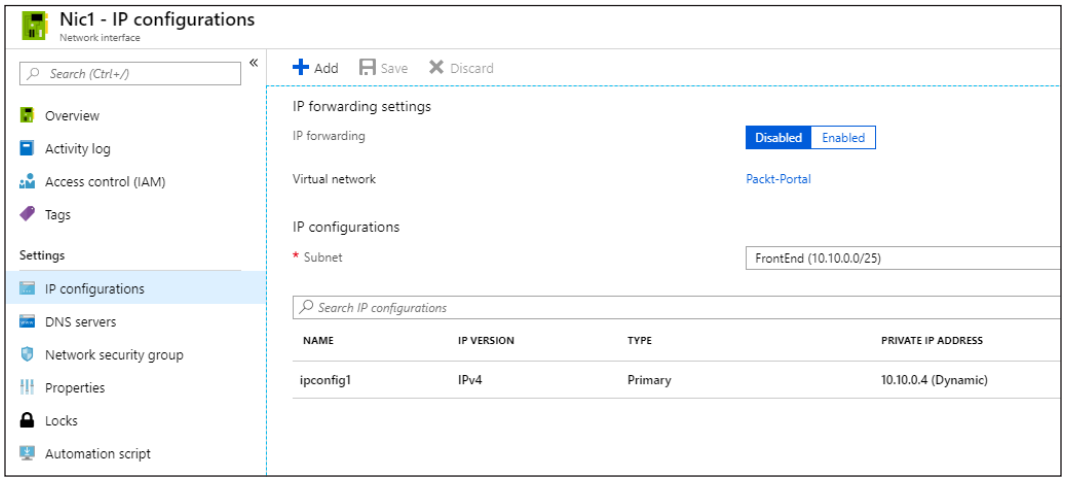
Getting ready

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

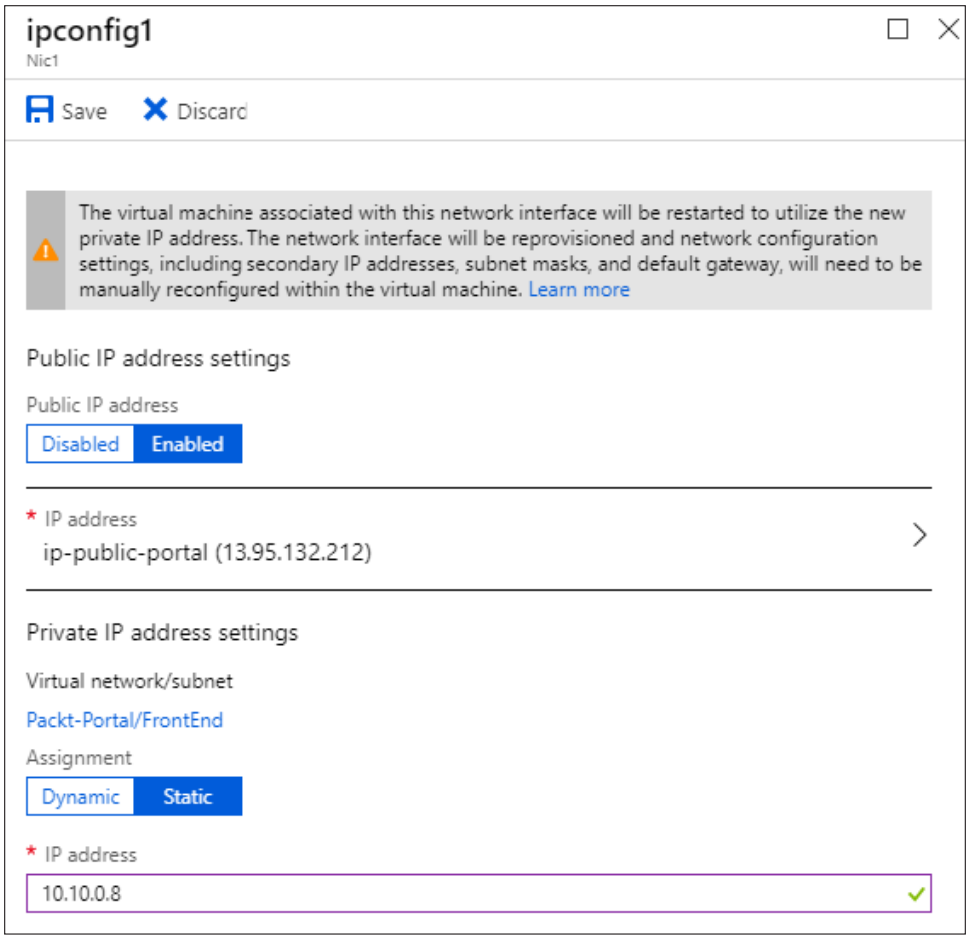
How to do it...

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:



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 VM associated with the NIC is turned off, the new IP address will be assigned on its next start-up. If the VM 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.

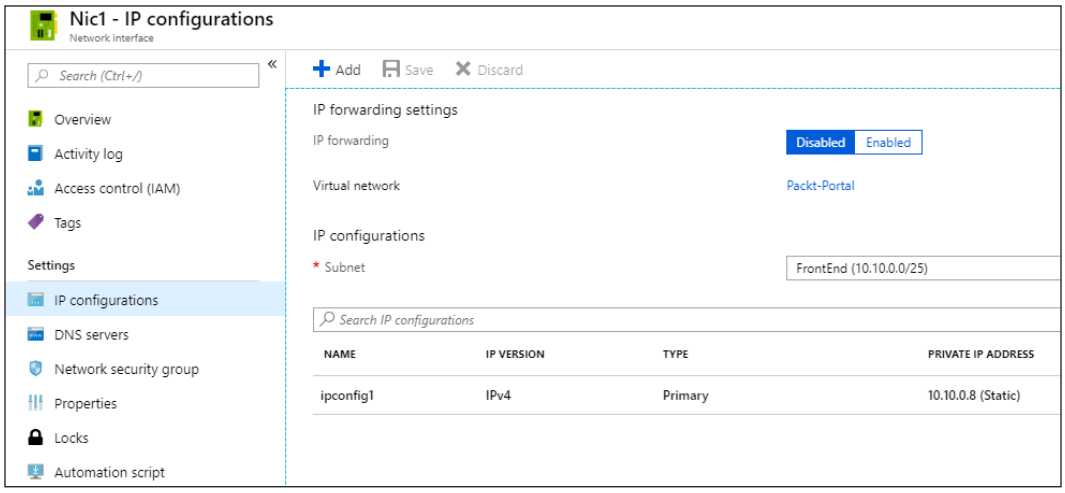
Getting ready

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

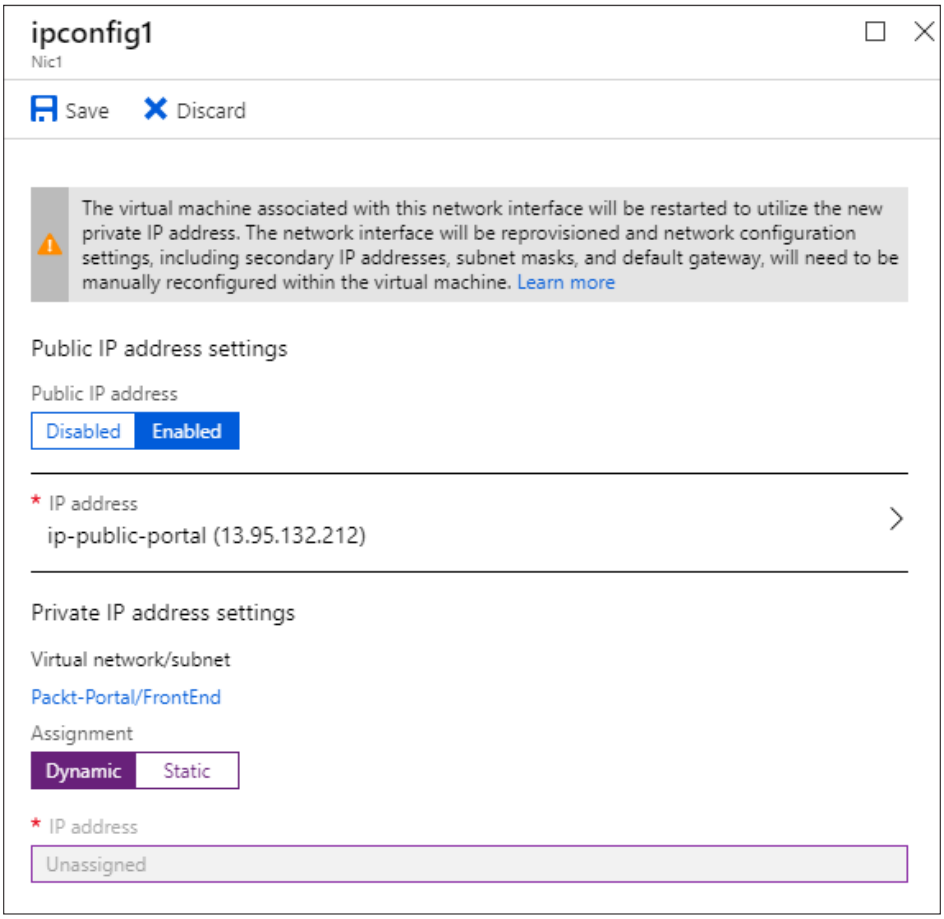
How to do it...

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 to.
2. In the NIC blade, go to **IP configurations** under **Settings** and select the IP configuration:



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 VM associated with the NIC will be restarted to 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 Azure portal
- Creating a local network gateway with PowerShell
- Creating a virtual network gateway in the Azure 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 examples can be found in `https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter05`.

Creating a local network gateway in the Azure 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.

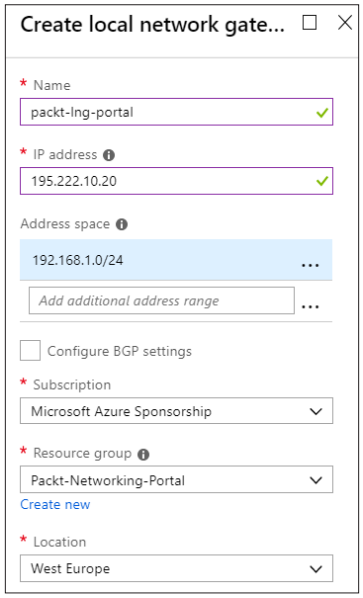
Getting ready

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 Azure 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 the name, resource group, location, gateway IP address and address prefix that we want. 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 to.

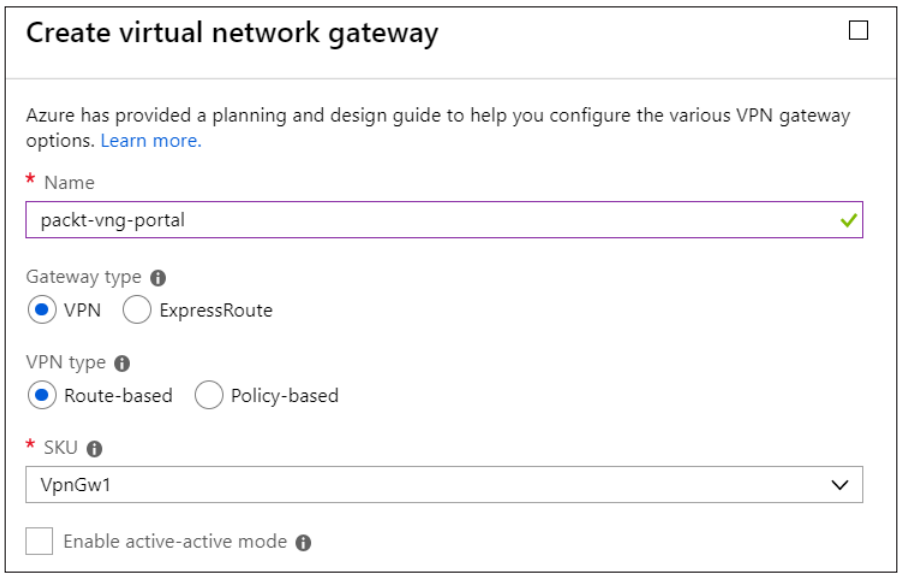
Getting ready

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

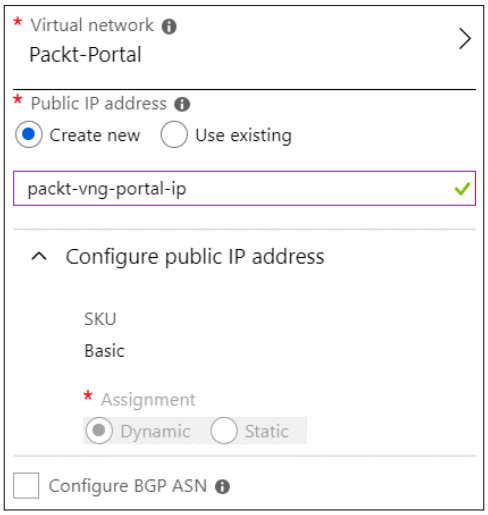
How to do it...

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

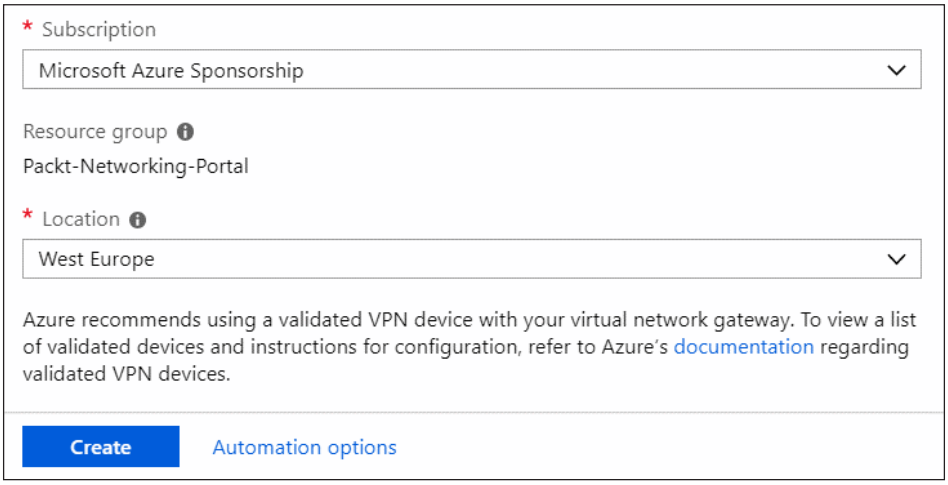
1. In the Azure portal, select **Create a resource** and choose **Virtual network gateway** under the **Networking** services (or search for `virtual network gateway` in the search bar).
2. Everything is done in a single blade, but for the purpose of better visibility, I'm going to break it down into 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**:



3. In the second section, we need to select the **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 where the virtual network gateway will be created:



5. After validation, we can click on **Create** and start the 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 script:

```
$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 the Azure 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 we 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'd need to reconfigure the local network gateway. Or, a local network might be reconfigured and the address space or subnet has changed, so we'd need to reconfigure the local network gateway once again.

Getting ready

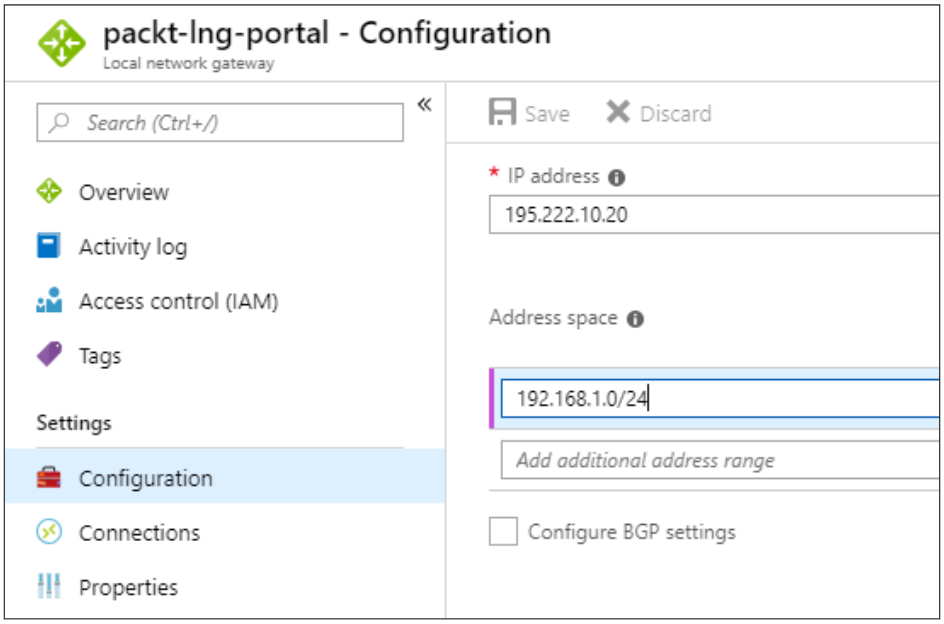
Before you start, open a web browser and go to the Azure portal at `https://portal.azure.com`.

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:



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 the 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 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). 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-to-Site or a Point-to-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-to-Site, VNet-to-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-to-Site connection
- Downloading the VPN device configuration from Azure
- Creating a Point-to-Site connection
- Creating a VNet-to-VNet connection
- Connecting VNets using network peering

Technical requirements

For this chapter, the following are required:

- An Azure subscription
- Windows PowerShell

Code examples can be found at `https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter06`.

Creating a Site-to-Site connection

A Site-to-Site connection is used to create a secure connection between an on-premises network and an 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 lets us limit management operations to be allowed only when coming from a secure and controlled environment, such as our local network.

Getting ready

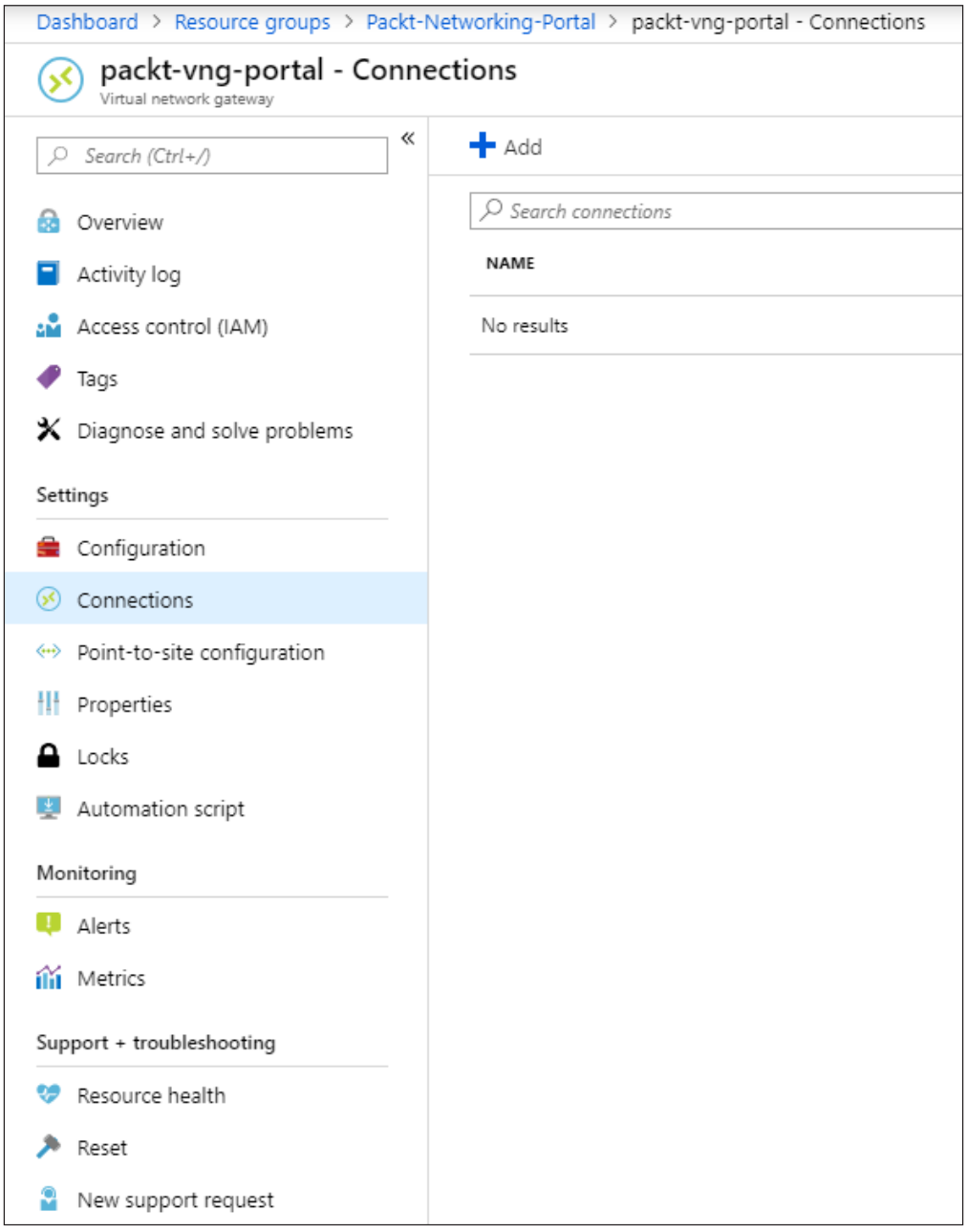
Before you start, open your browser and go to the Azure portal at `https://portal.azure.com`.

How to do it...

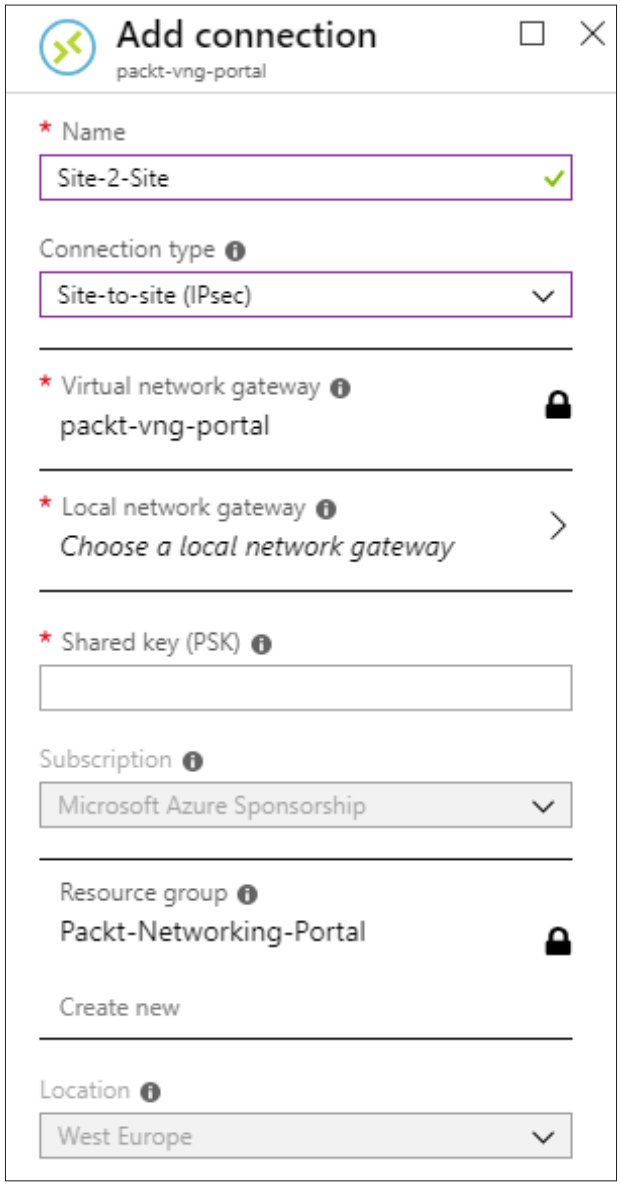
To create a new Site-to-Site connection, we must follow these steps:

1. Locate the virtual network gateway (the one we created in *Chapter 5*, *Local and Virtual Network Gateways*) and select **Connections**.

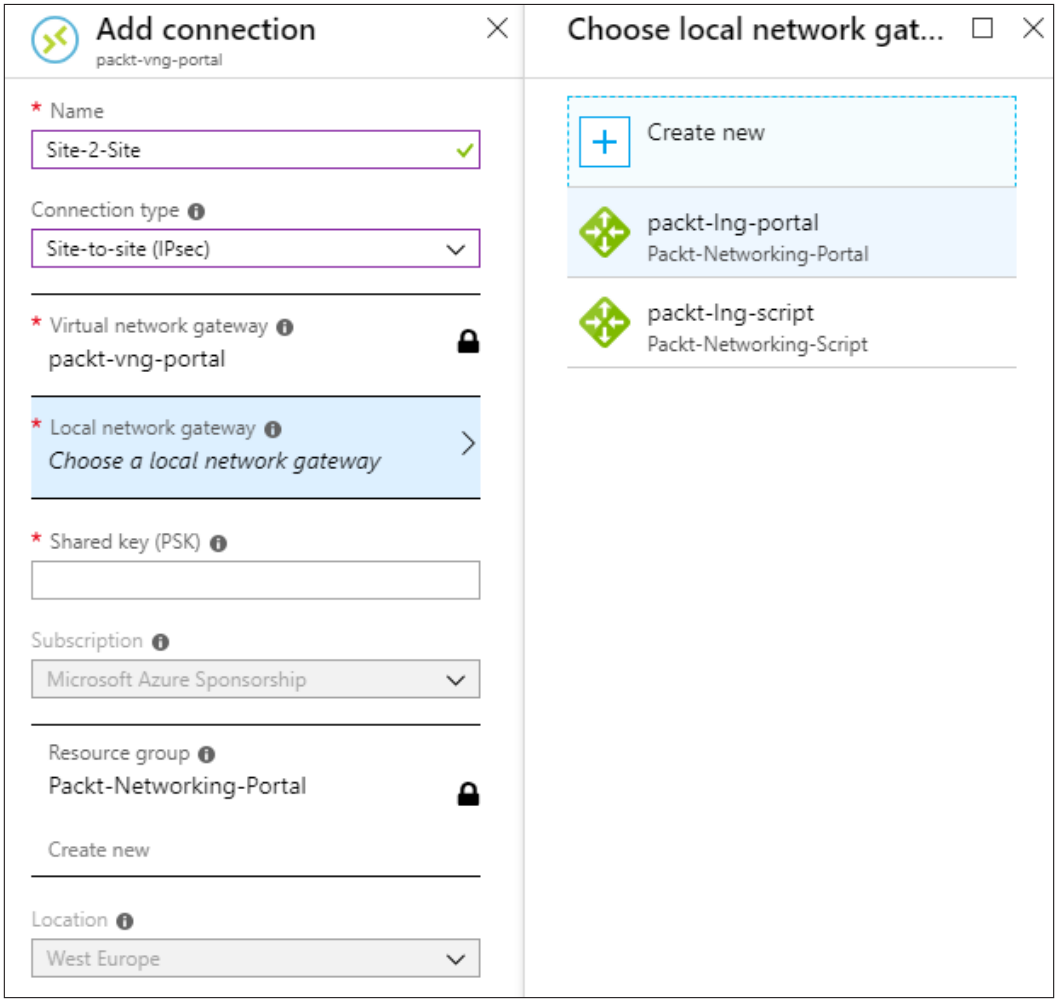
2. In **Connections**, select the **Add** option to add a new connection:



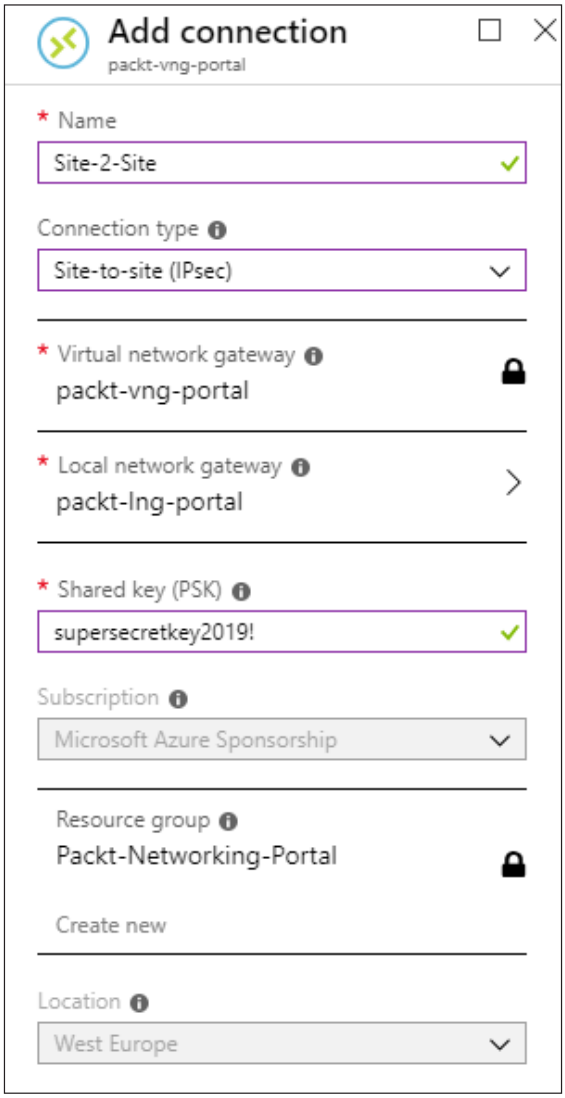
3. In the new blade, we need to enter some information for the connection's **Name** and select **Site-to-Site (IPsec)** for **Connection type**:



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*):



5. We need to provide a **Pre-Shared Key** (**PSK**) that will be used for the 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 and encrypted VPN connection between networks.

Downloading the VPN device configuration from Azure

After creating the Azure side of the Site-to-Site connection, we still need to configure the local VPN device. The configuration depends upon the vendor and the device type. You can see all the supported devices at `https://docs.microsoft.com/en-in/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

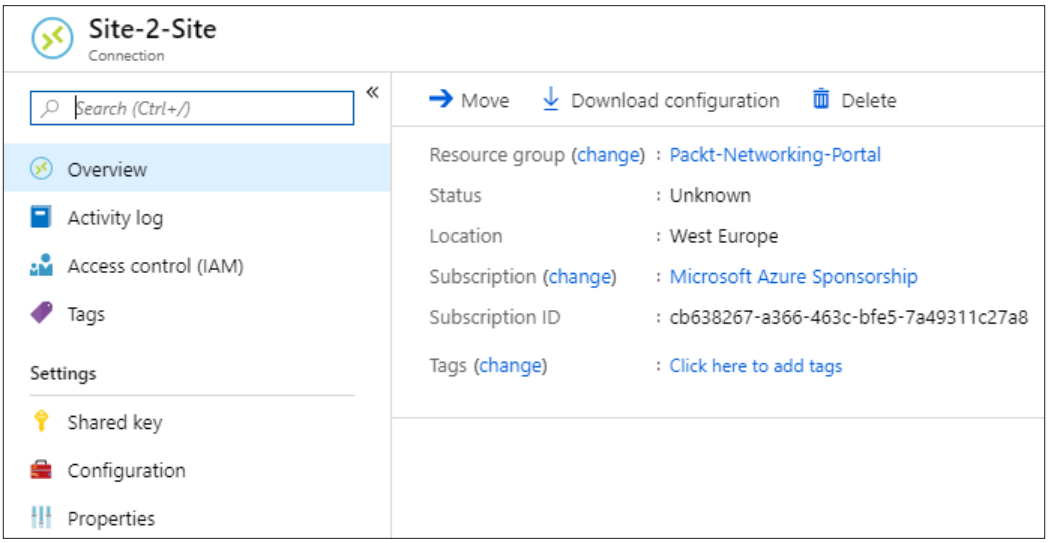
Before you start, open the browser and go to the Azure portal at `https://portal.azure.com`.

How to do it...

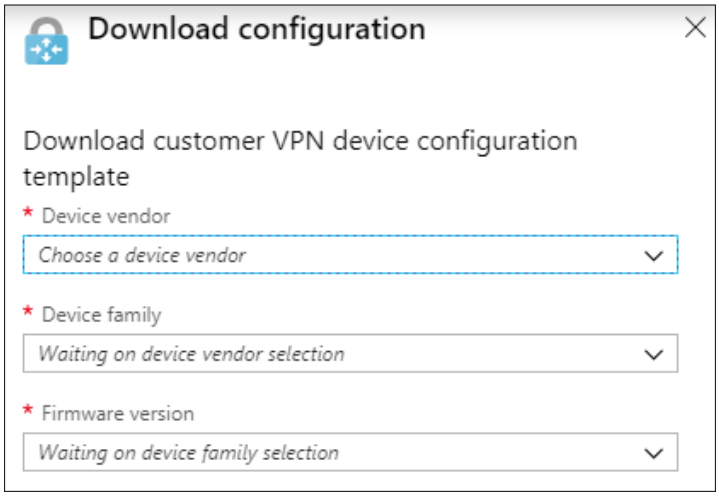
To download the VPN device configuration, we must follow these steps:

1. Locate the Site-to-Site connection in the Azure portal. The **Overview** blade will be opened by default.

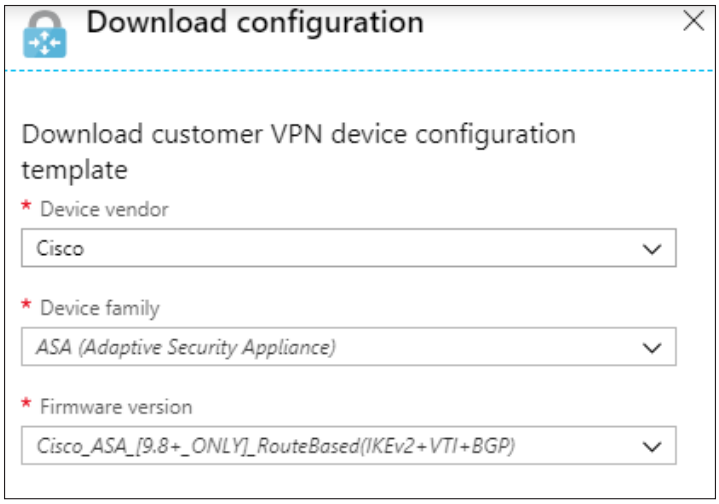
2. Select the **Download configuration** option from the top of the blade:



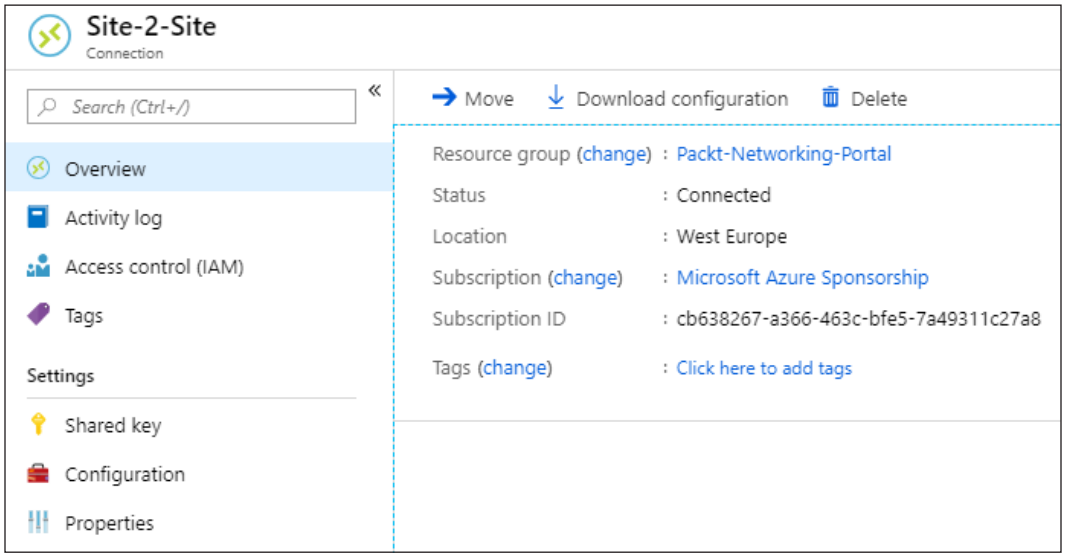
3. A new blade will open, and you will see that all the options in the blade are predefined:



4. Select the relevant options for your given **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 example 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. The **Status** under **Site-2-Site Connection** will change to **Connected**:



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 an Azure VNet.

Creating a Point-to-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-to-Site connection, especially when we have to perform something out of work hours. In this case, we can use Point-to-Site to create a secure connection that can be established from anywhere.

Getting ready

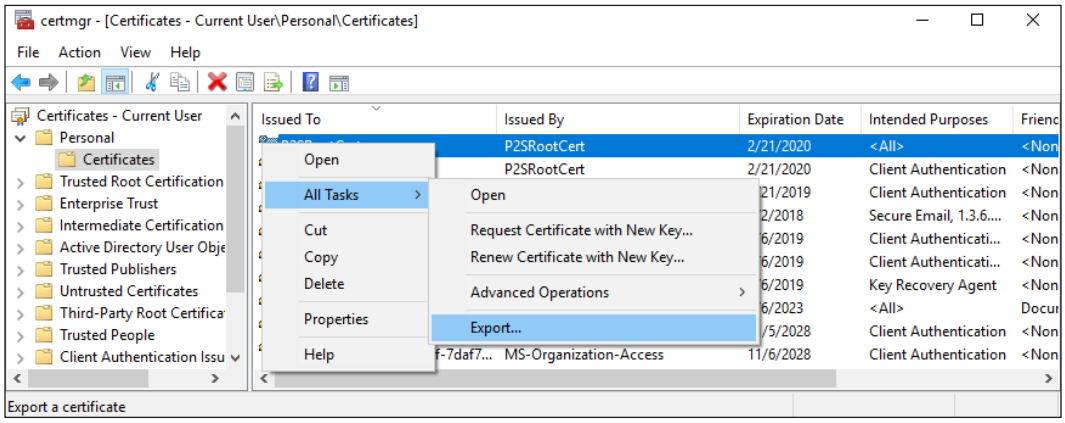
To create a Point-to-Site connection, we'll need to generate a certificate that will be used for the connection. To create a certificate, we must follow these steps:

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

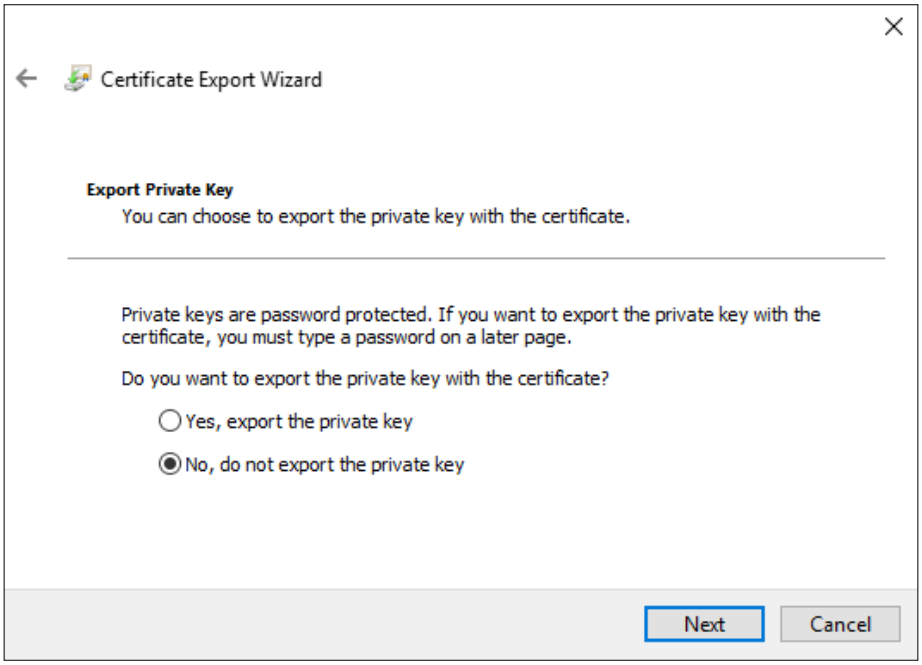
```
$cert = New-SelfSignedCertificate -Type Custom -KeySpec Signature
`
-Subject "CN=P2SRootCert" -KeyExportPolicy Exportable `
-HashAlgorithm sha256 -KeyLength 2048 `
-CertStoreLocation "Cert:\CurrentUser\My" -KeyUsageProperty Sign
-KeyUsage CertSign

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"
)
```

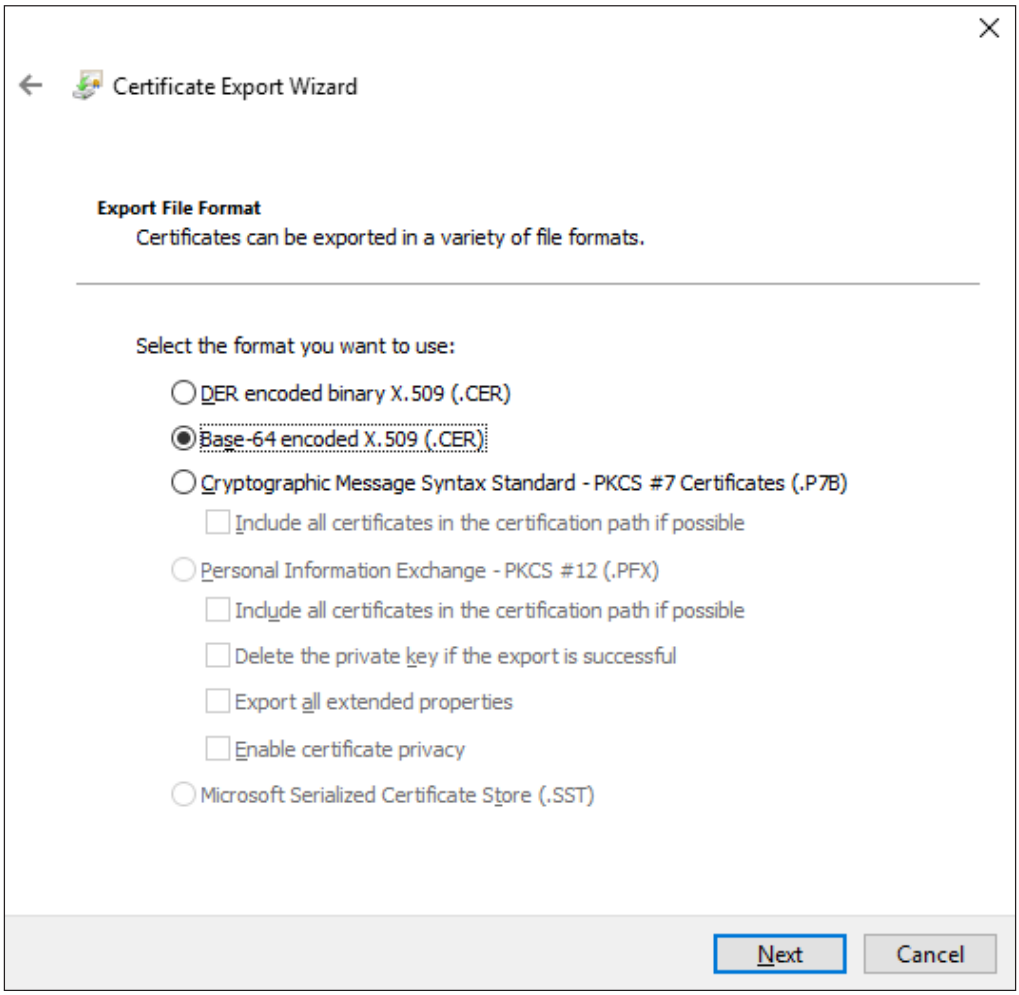
2. Next, we need to export the certificate. Open **certmgr**, locate the personal certificate, select **P2SRootCert** and then choose the **Export...** option:



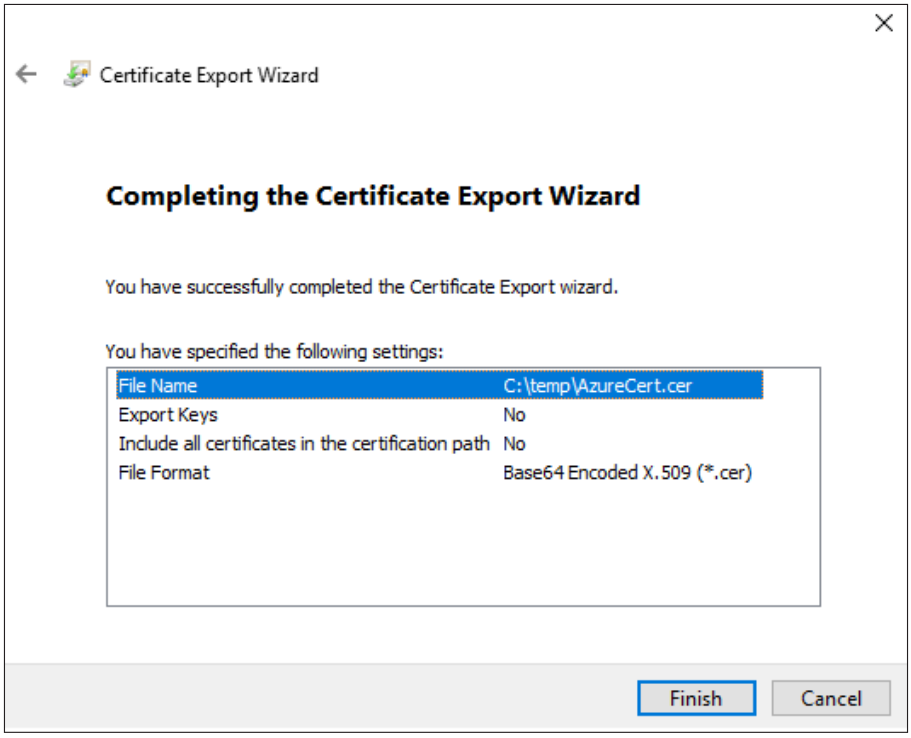
3. This will start the **Certificate Export Wizard**. Click **Next**.
4. Select the **No, do not export the private key** option and click **Next**:



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



6. Select the location where you want to save the certificate and click **Next**.
7. Finally, we have the option to review all the information. After clicking **Finish**, the export will be complete:

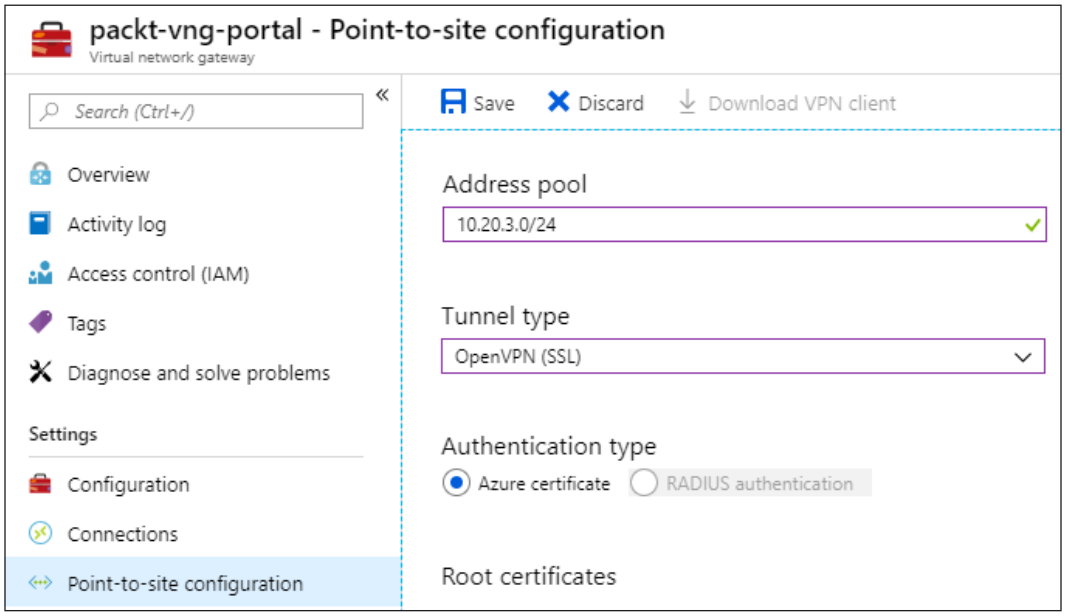


Now, let's look at the process of creating a Point-to-Site connection.

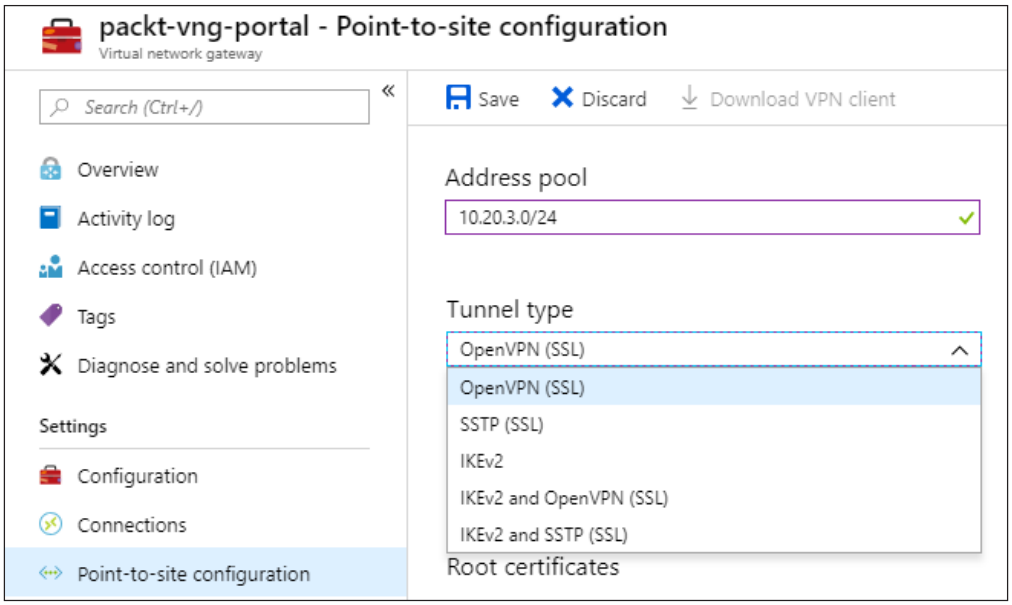
How to do it...

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

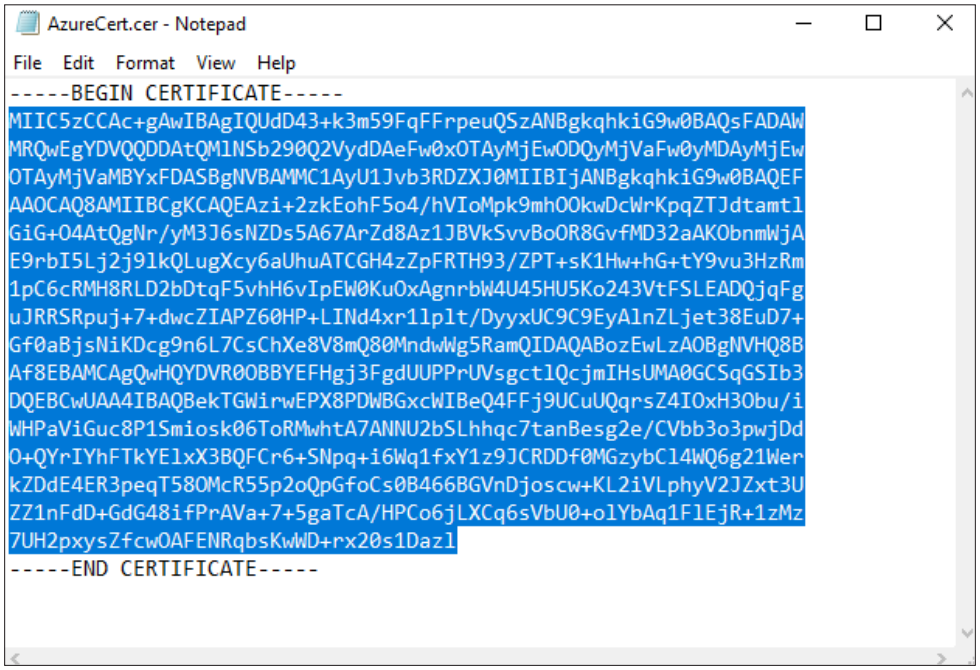
1. In the Azure portal, locate the virtual network gateway and **Point-to-Site configuration**.
2. We need to define the **Address pool**. The **Address pool** here cannot overlap with the address pool of the VNet associated with the virtual network gateway:



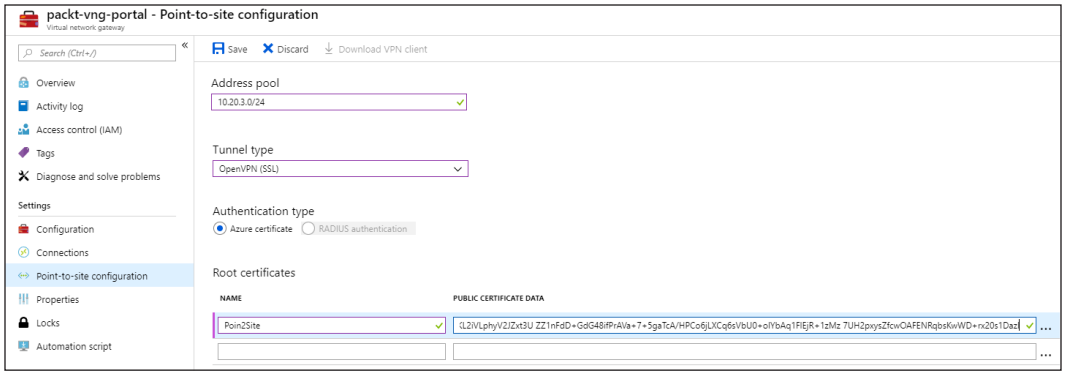
3. Next, we need to select **Tunnel type** from the list of predefined options. In this recipe, we'll select **OpenVPN (SSL)**, but any option is valid:



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:



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



6. After clicking **Save** for the Point-to-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-to-Site allows us to access Azure VNet in a secure way. A Site-to-Site connection is restricted to access from our local network, but Point-to-Site allows us to connect from anywhere. Certificate-based authentication is used, which uses the same certificate on both the server (Azure) and the client (the VPN client) 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-to-Site connection.

Creating a VNet-to-VNet connection

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

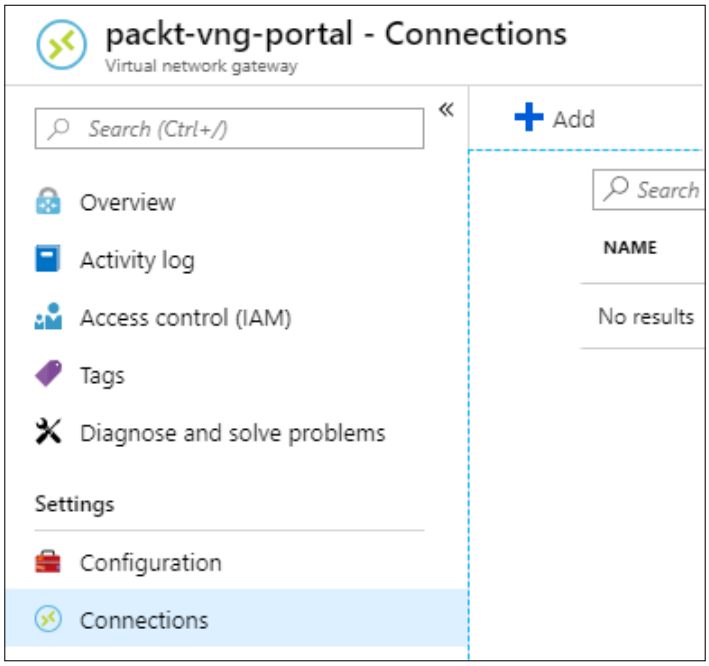
Getting ready

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

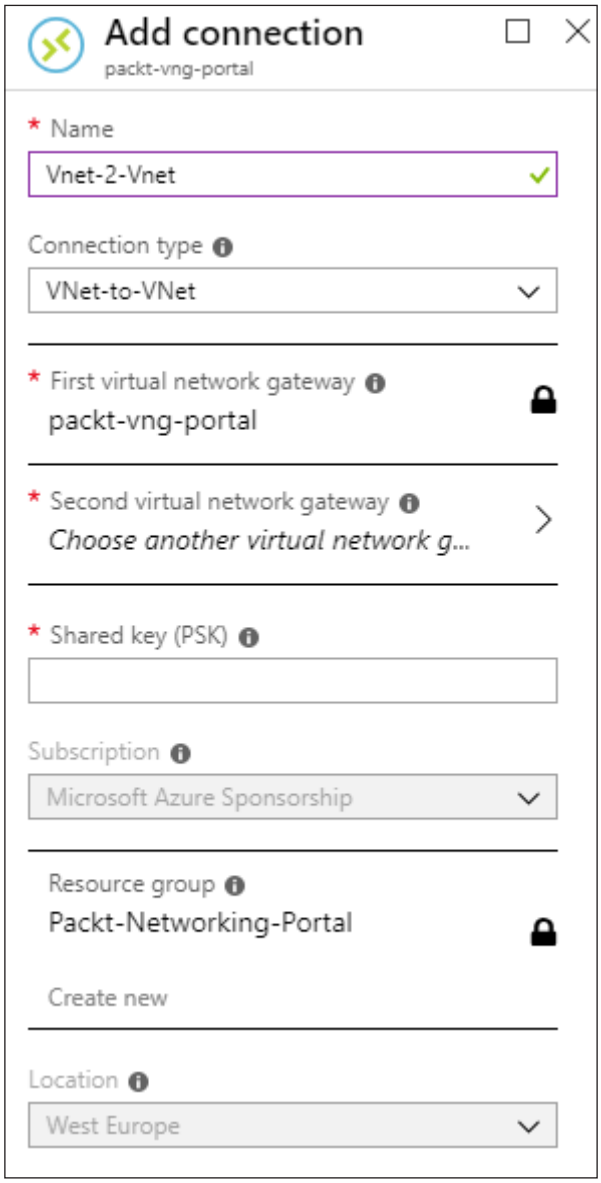
How to do it...

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

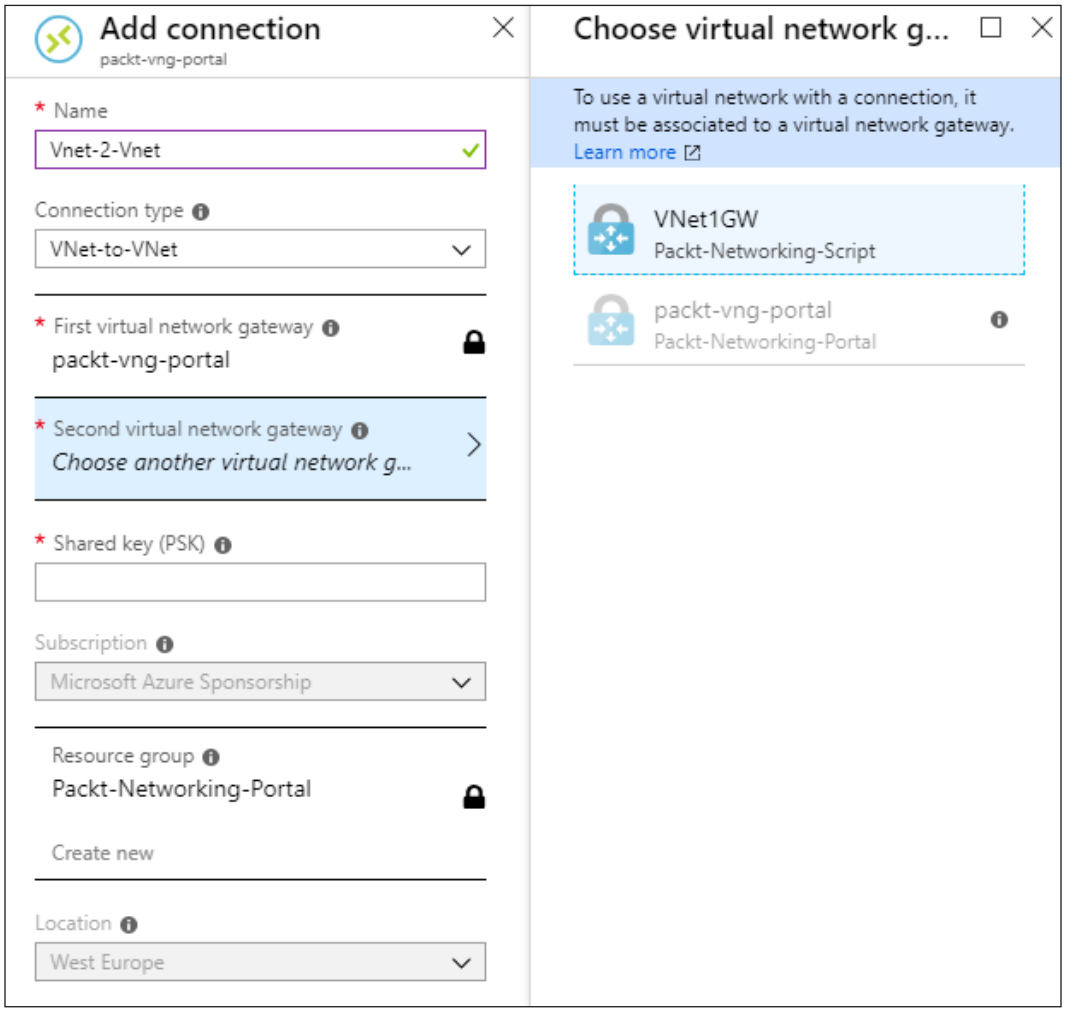
1. In the Azure portal, locate one of the virtual network gateways (associated with one of the VNets you are trying to connect to).
2. In the virtual network gateway blade, select **Connections** and select **Add** to add a new connection:



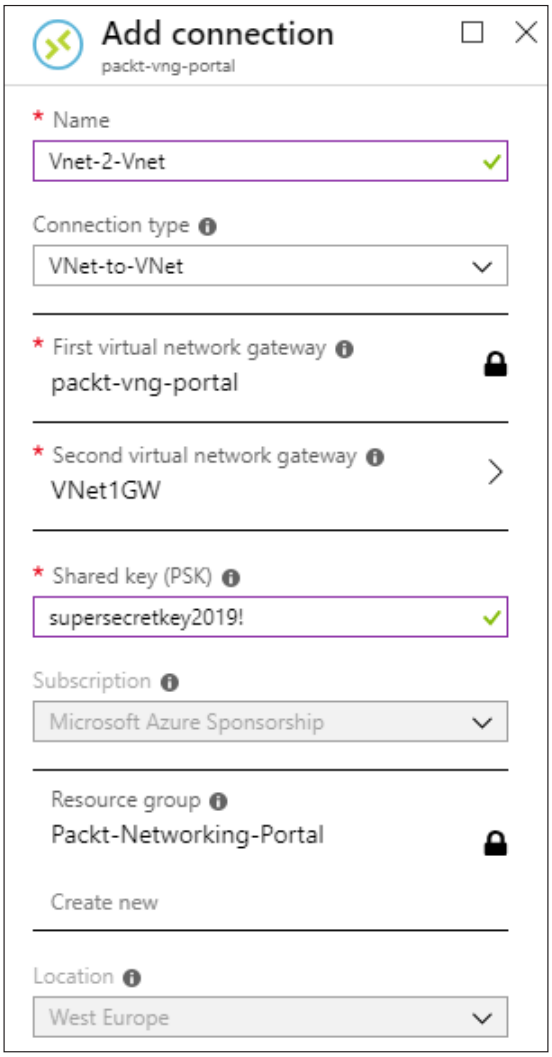
3. In the new blade, enter the **Name** for a new connection and select **VNet-to-VNet** under **Connection type**:



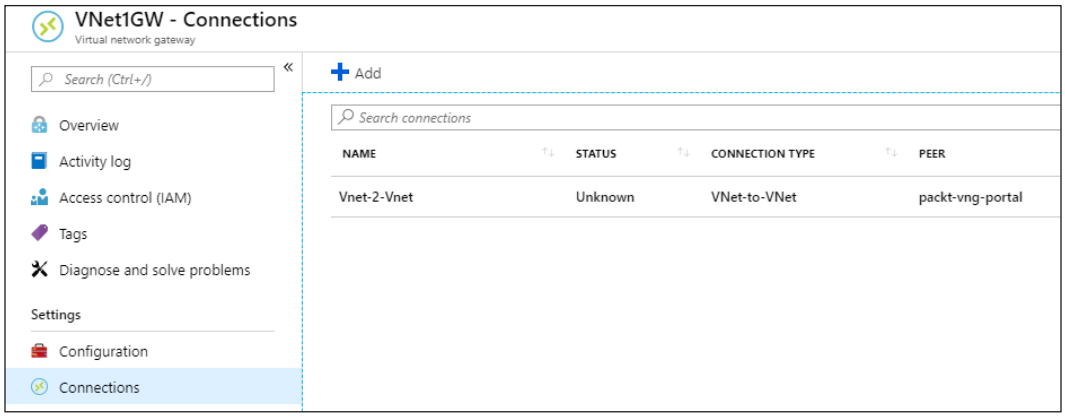
4. The first virtual network gateway will be automatically highlighted. We need to select the second virtual network gateway:



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



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

How it works...

A VNet-to-VNet connection works very similarly to a Site-to-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. This results in 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.

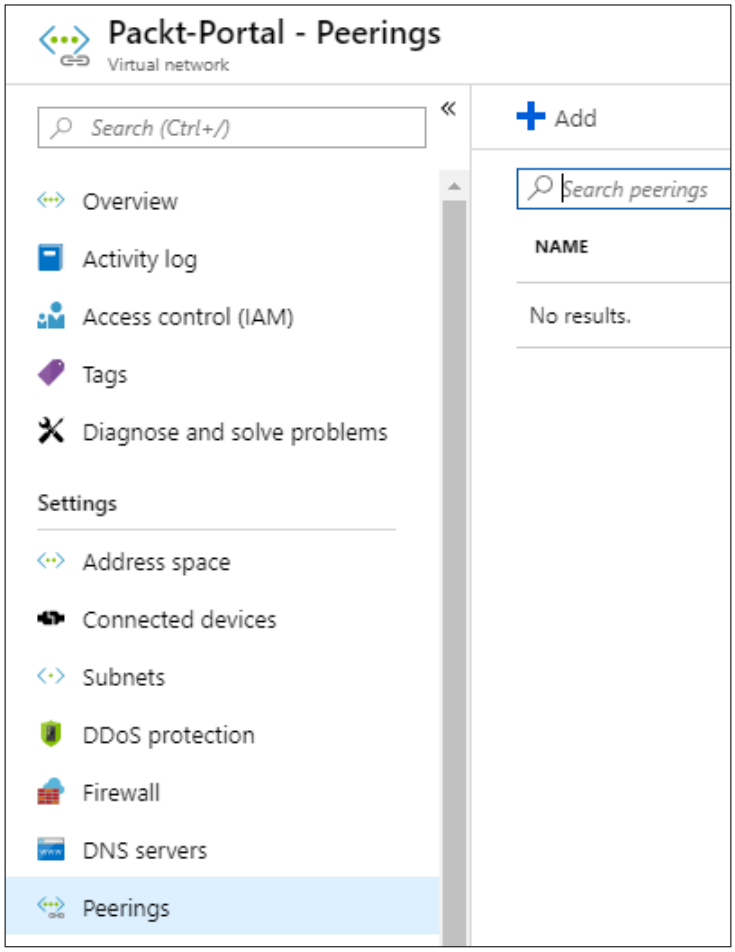
Getting ready

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

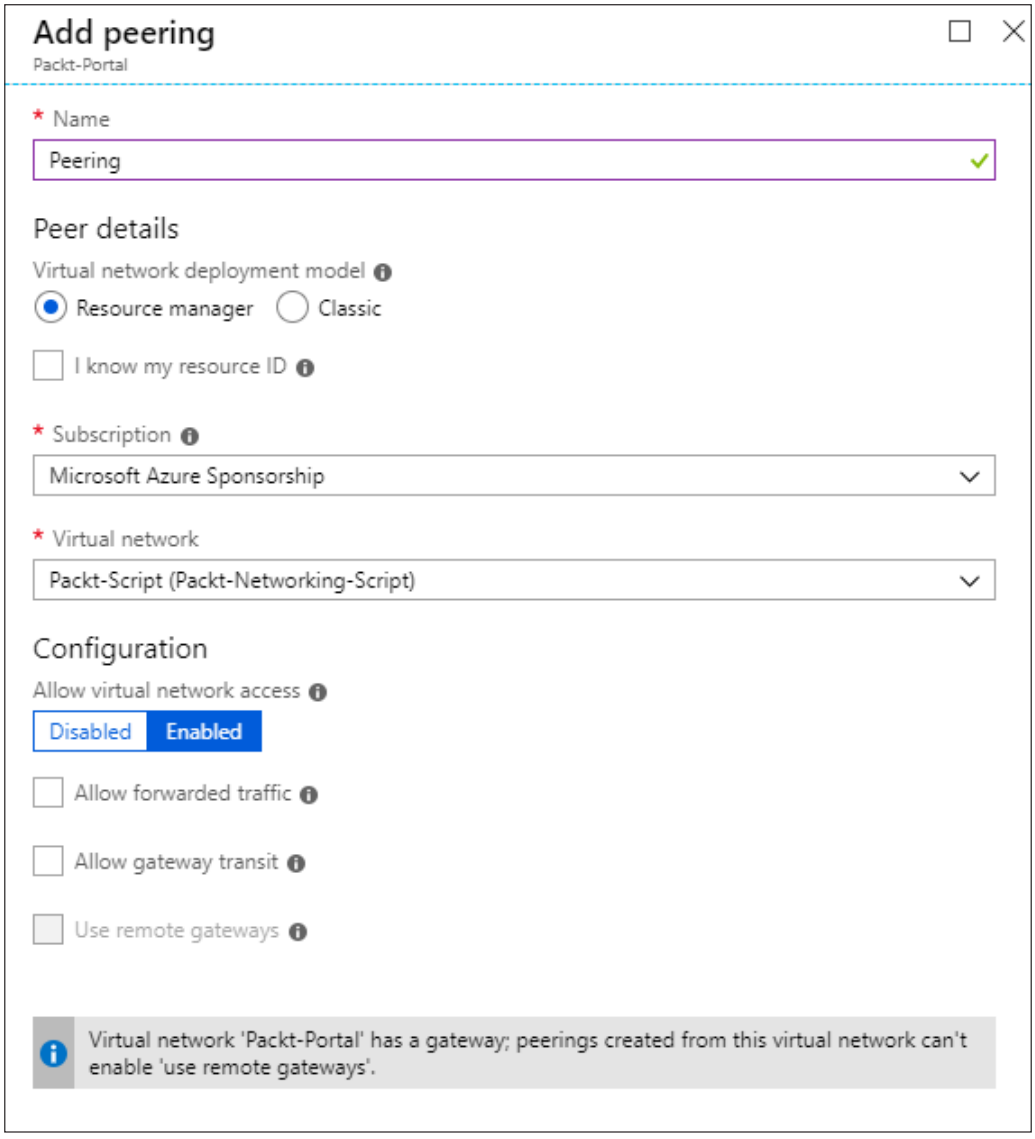
How to do it...

To create network peering, we must do the following steps:

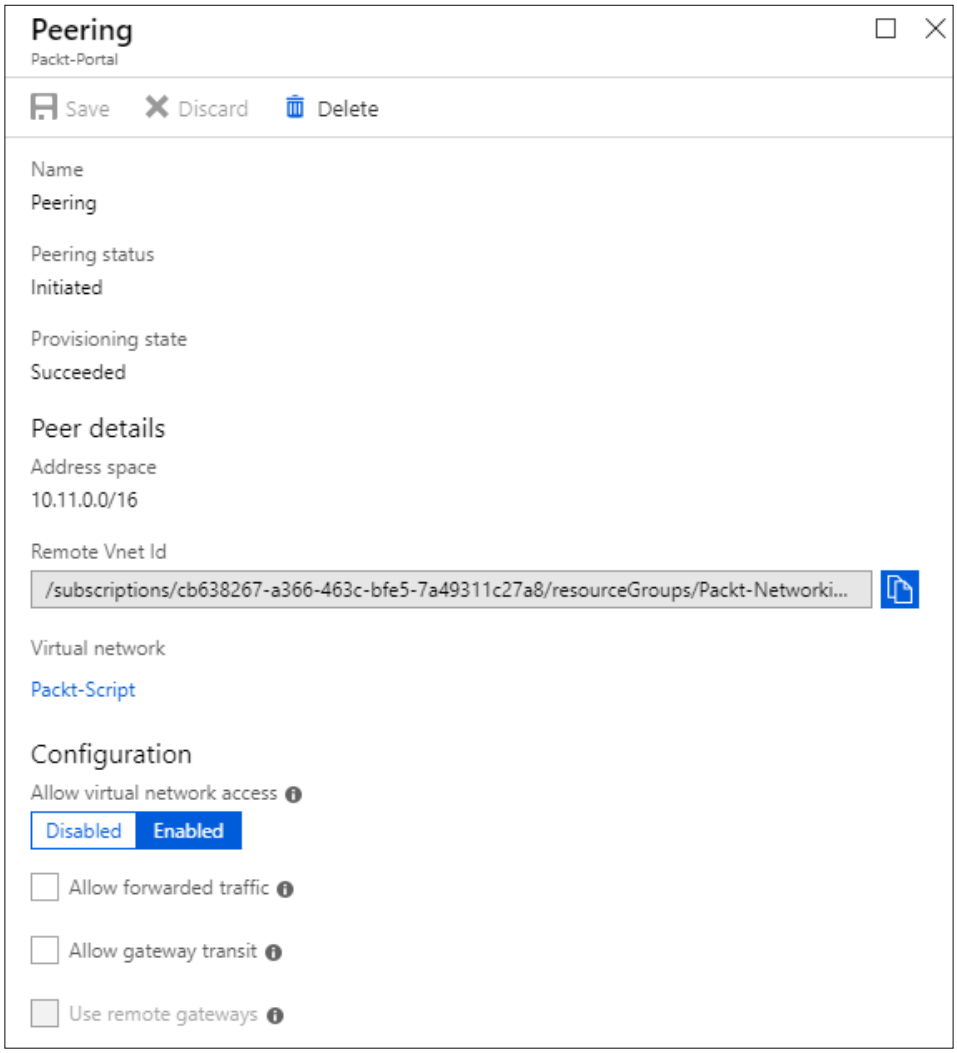
1. In the Azure portal, locate one of the VNets that you want to connect to.
2. In the VNet blade, select the **Peerings** option, and select **Add** to add a new connection:



3. In the new blade, we must enter the name of the connection, choose a **Virtual network deployment model** (**Resource manager** or **Classic**) and select the VNet we are connecting to. This information can be provided either by 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, but provides us with better traffic control:



4. 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:



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 (which 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 a VNet to both the local network and another VNet, we must take a different approach and use a virtual network gateway that will allow us to create a Site-to-Site connection with a local network and a VNet-to-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 with a subnet
- Dissociating the route table from a subnet
- Creating a new route
- Changing a route
- Deleting a route

Technical requirements

For this chapter, the following is required:

- An 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.

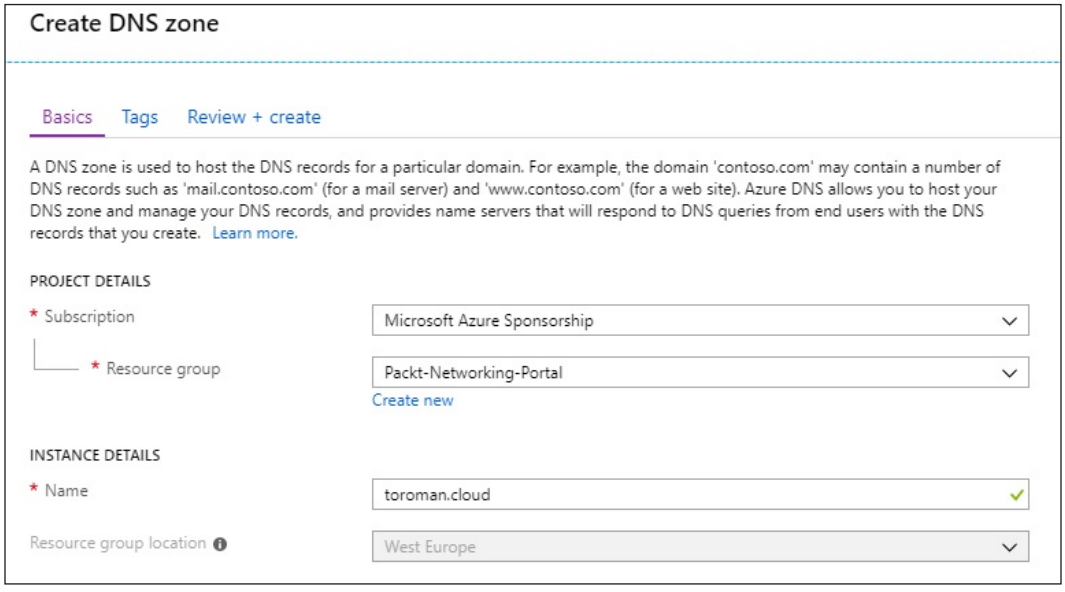
Getting ready

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 the new blade, we must enter information for the **Name**, **Subscription** and **Resource group** fields. If we select the existing resource group, the location will be the same as the one for the resource group selected. **Name** must be a **Fully Qualified Domain Name** (**FQDN**):



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, record 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.

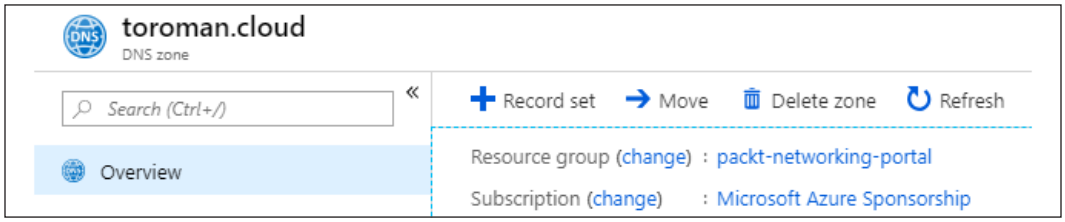
Getting ready

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

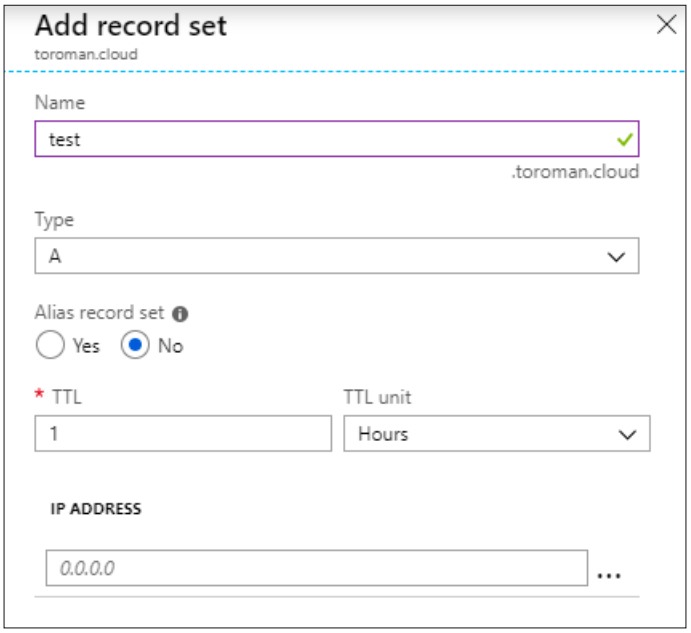
How to do it...

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

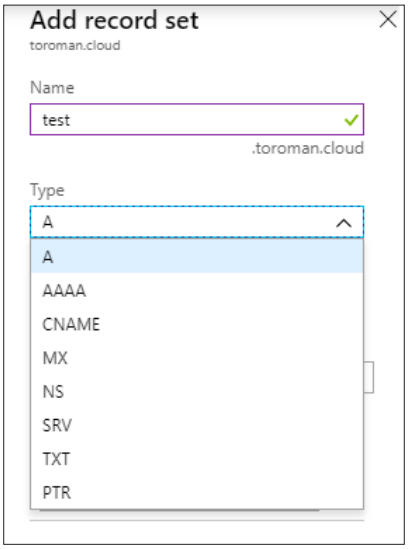
1. In the Azure portal, locate the DNS zone.
2. In the overview, select the option for adding a record set:



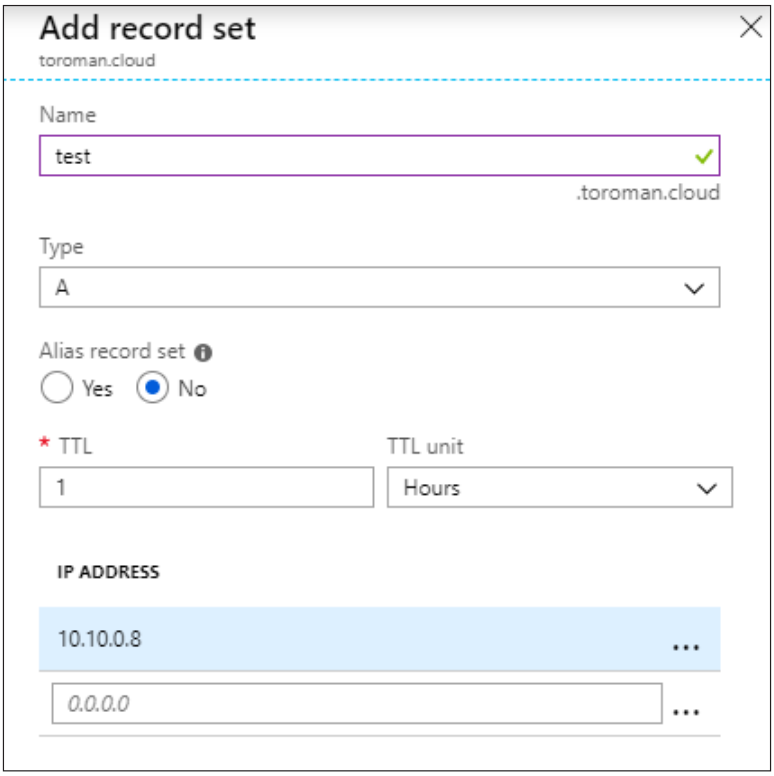
3. A new blade will open. Enter the name of the subdomain to which you want to add a record:



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:



5. 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:



6. 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 is `toroman.cloud`, and the subdomain is `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.

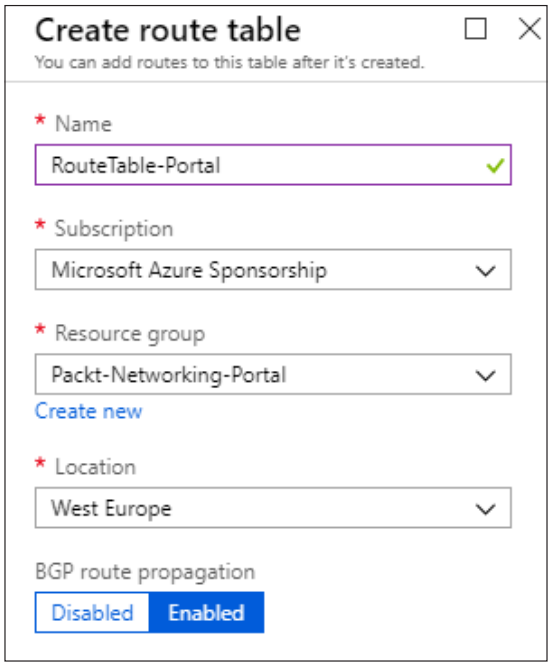
Getting ready

Before you start, open the browser and go to the Azure portal at `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 (which is 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 the 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.

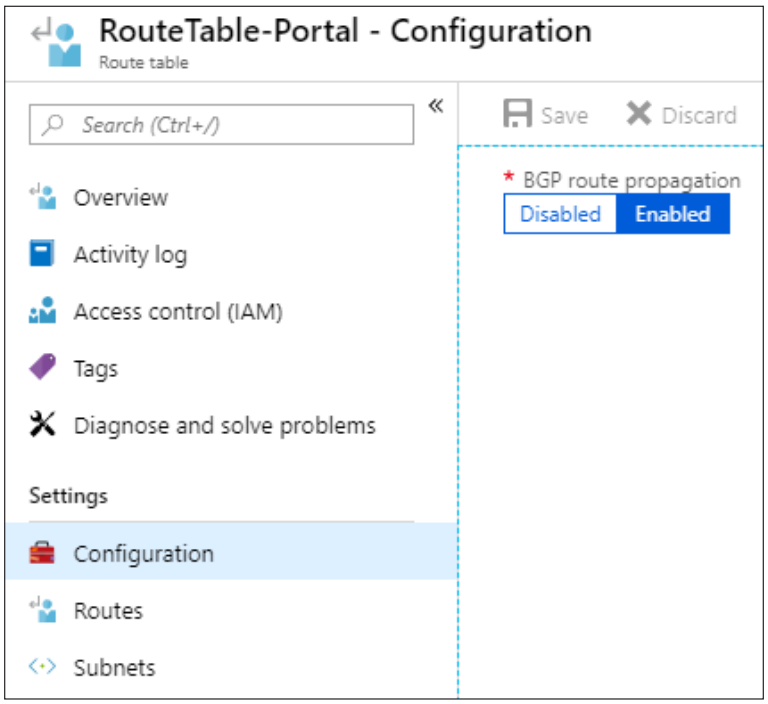
Getting ready

Before you start, open the browser and go to the Azure portal at `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 can change the **BGP route propagation** settings in the **Configuration** blade. Under **Settings**, we can 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 with 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.

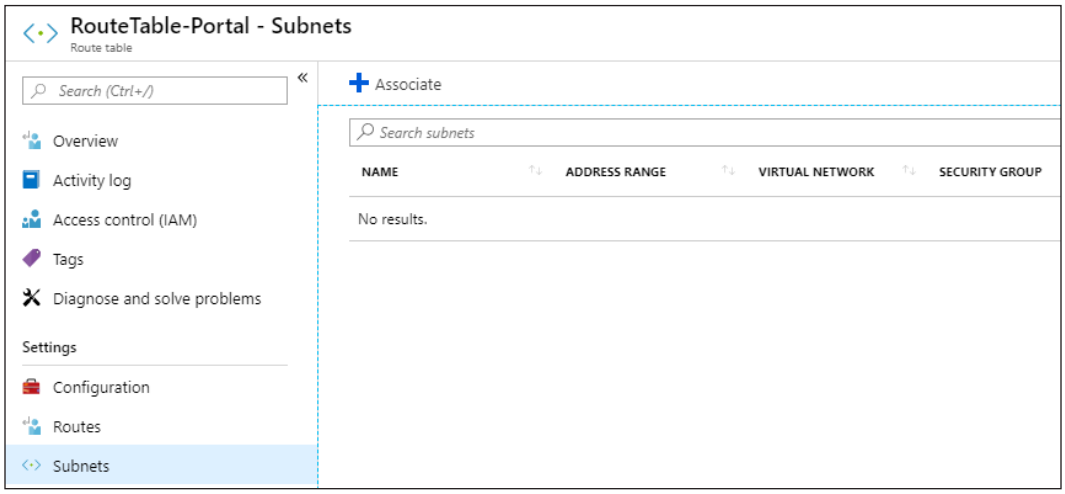
Getting ready

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

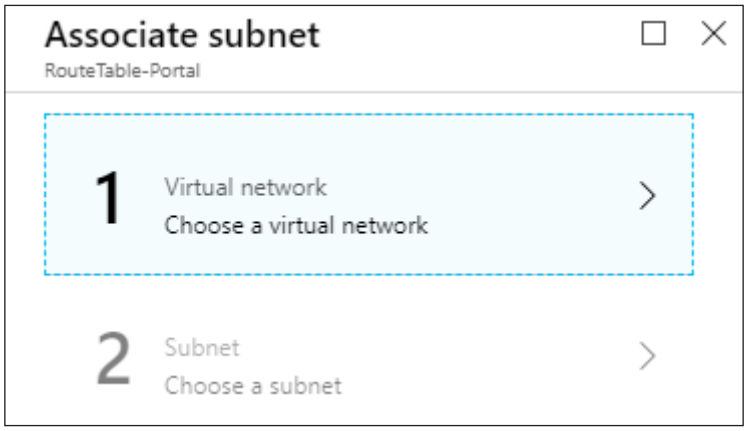
How to do it...

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

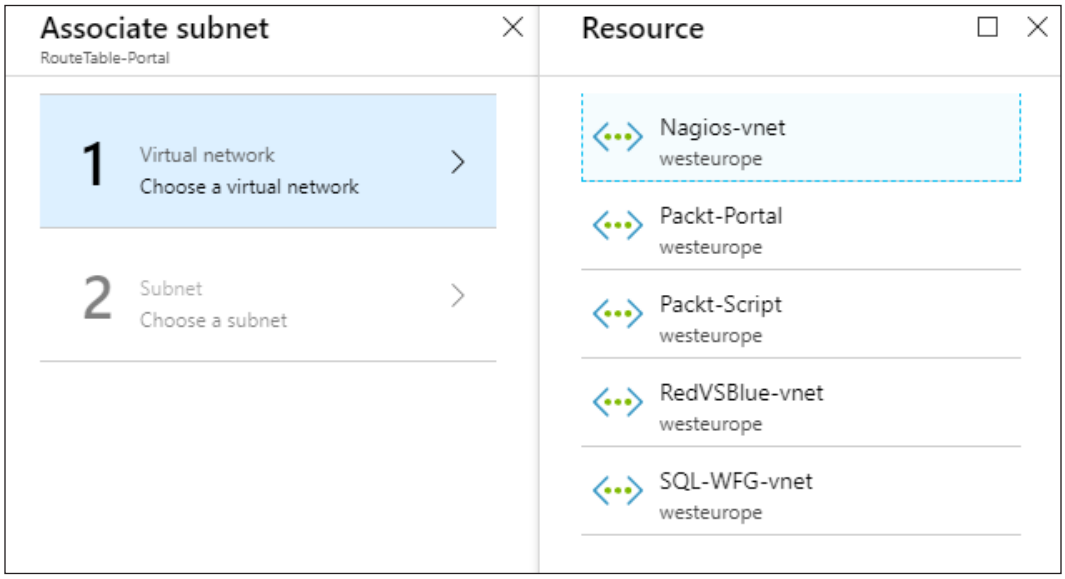
1. In the Azure portal, locate **Route table**.
2. Under **Settings** in the route table, select the **Subnets** option. In the **Subnets** blade, select the **Associate** option to create a new association:



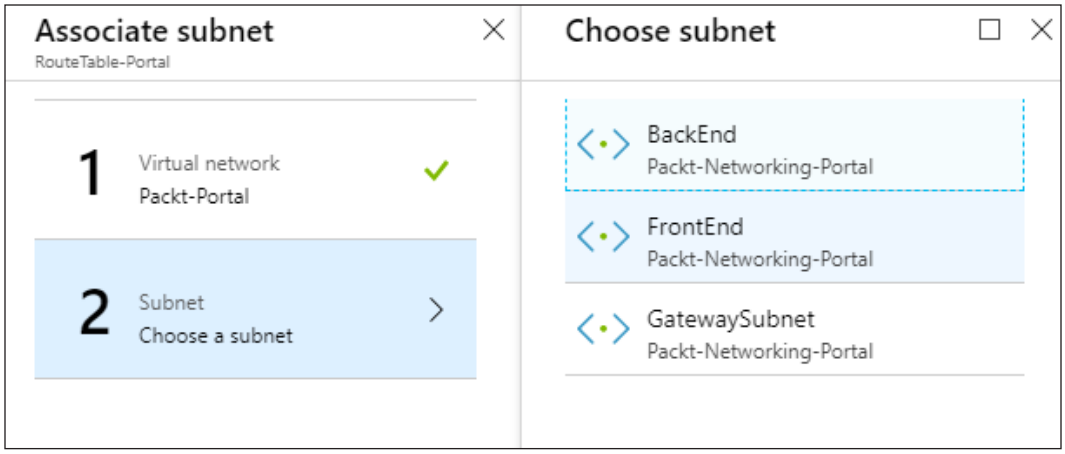
3. A new blade will open. There are two options available to select a **Virtual network** and the **Subnet** we want to associate from the list:



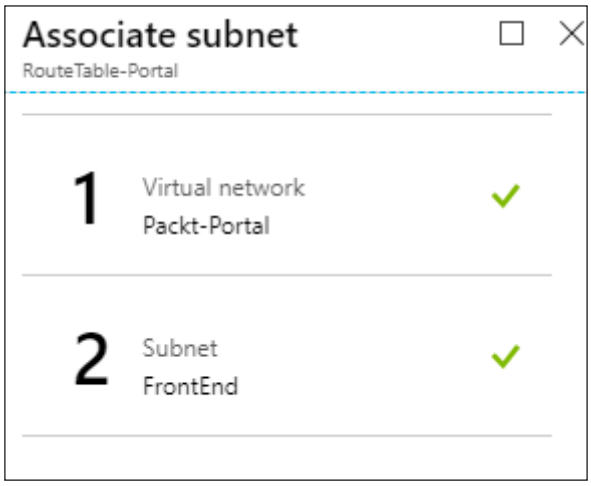
4. First, we must select **Virtual network**. Selecting this option will list all the available virtual networks. Select the one you want to associate from this list:



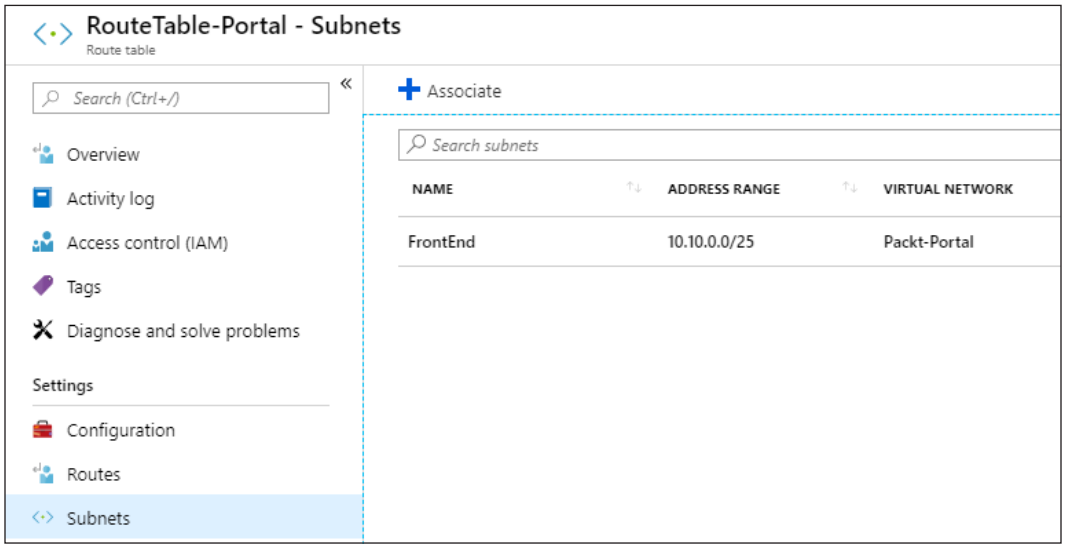
5. After a virtual network is selected, we can proceed to select a subnet. The subnet option will list all the subnets from the virtual network we selected in the previous step. Choose the subnet you want to associate from this list:



6. After both options are selected, we can proceed to create an association:



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



How it works...

The route table, to be effective, must have two parts defined: the *what* and the *how*. We define what will be affected by the route table 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 a subnet

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

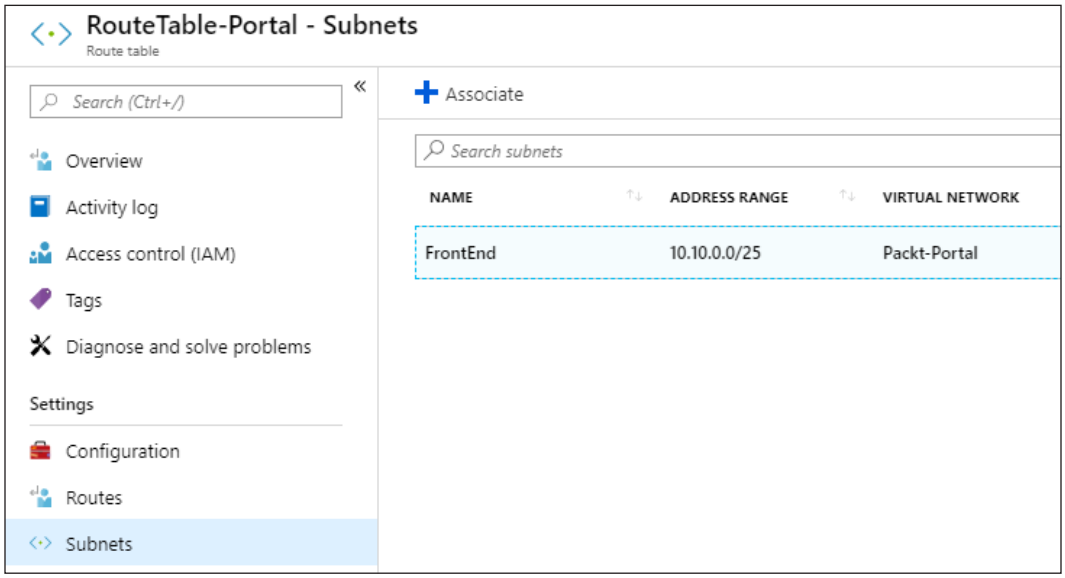
Getting ready

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

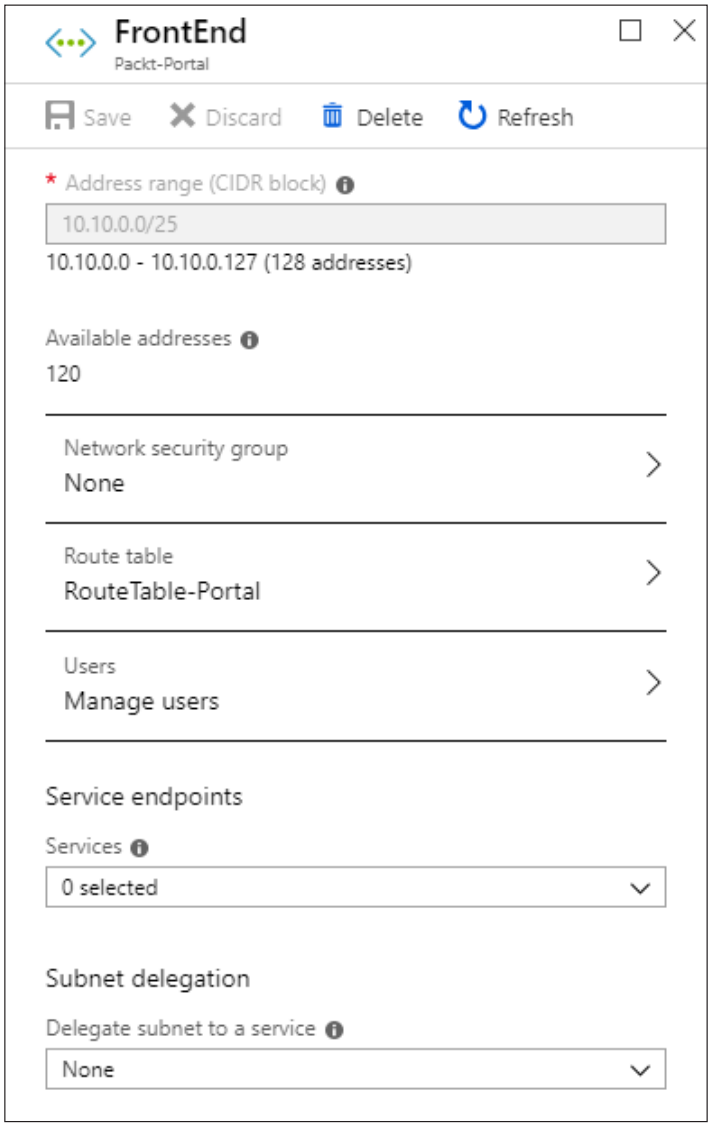
How to do it...

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

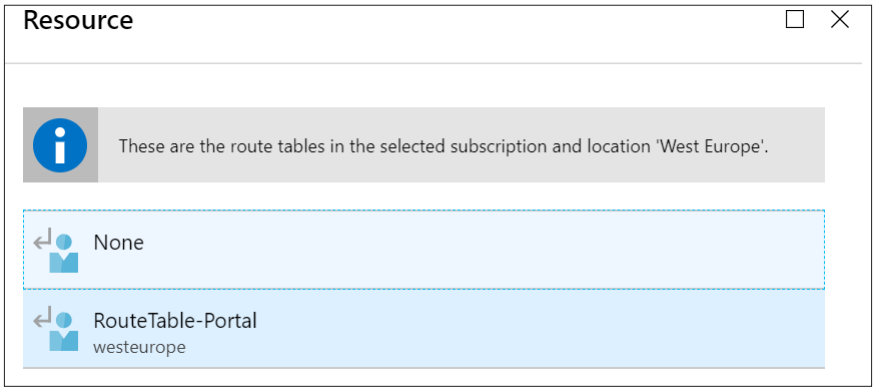
1. In the Azure portal, locate **Route table**.
2. Under **Settings**, select the **Subnets** option and select the subnet you want to remove:



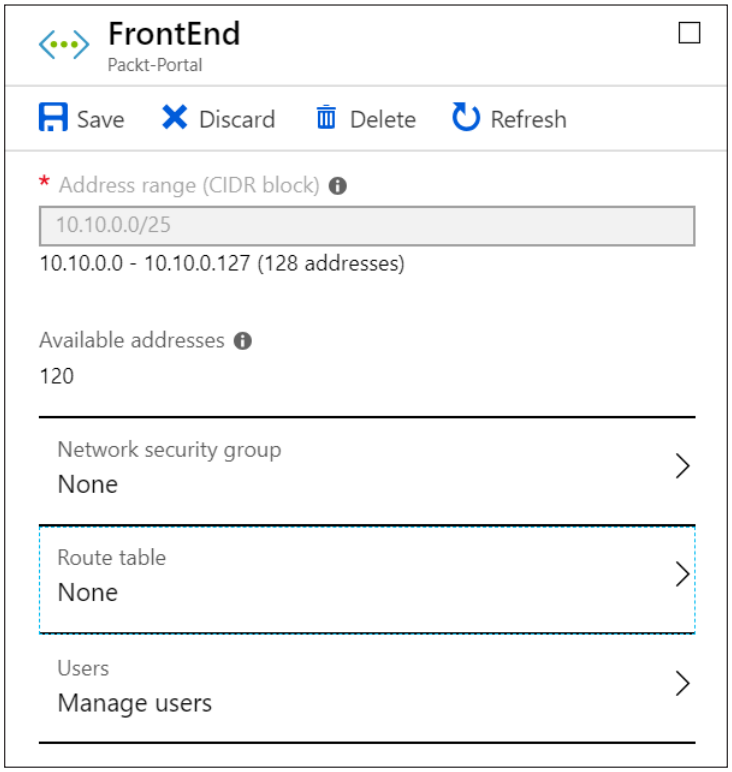
3. The subnet configuration blade will open. Select the route table option. Note that this actually opens a subnet configuration. It's a common mistake to confuse this blade with the association and to choose the **Delete** option. This will not only remove the association, but also remove the subnet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



4. The Azure portal 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. All rules will apply to all the associated subnets. If we need to make a single rule no longer 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, but we're missing the part that defines *how* it will be affected. 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.

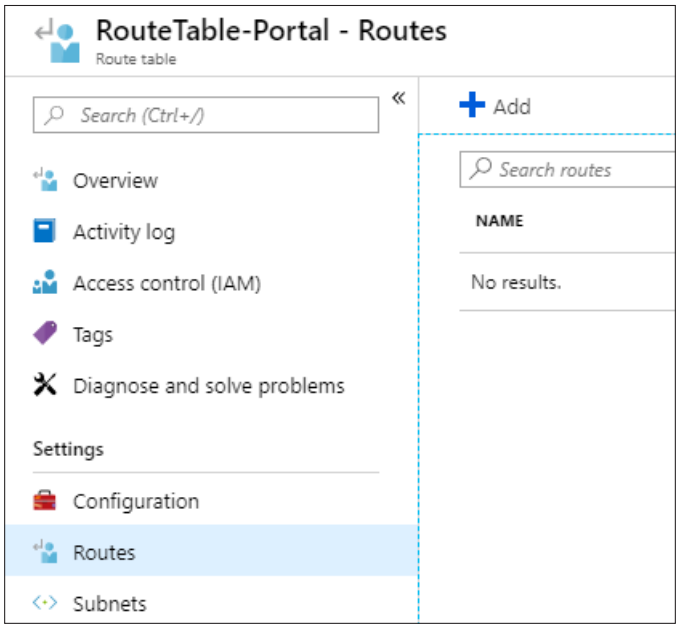
Getting ready

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

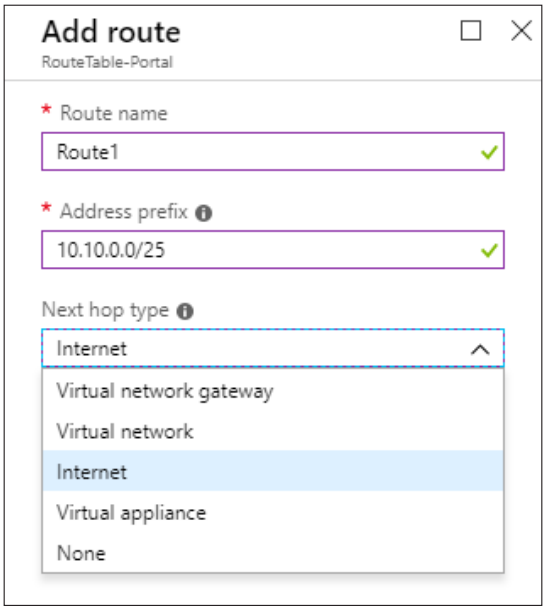
How to do it...

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

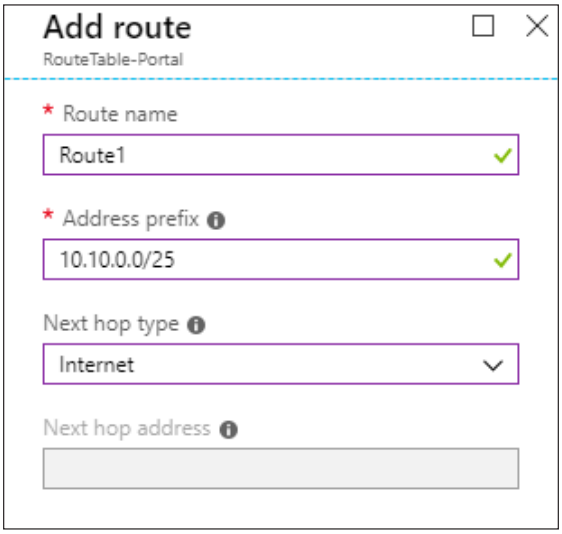
1. In the Azure portal, locate **Route table**.
2. In the **Route table** blade, under **Settings**, select **Routes**. Select **Add** to add a new route:



3. In the new blade, we need to define the **Route name** and the **Address prefix** (in CIDR format) for the destination IP address range and select **Next hop type**. Options for this include **Virtual network gateway**, **Virtual network**, **Internet**, **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:



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.

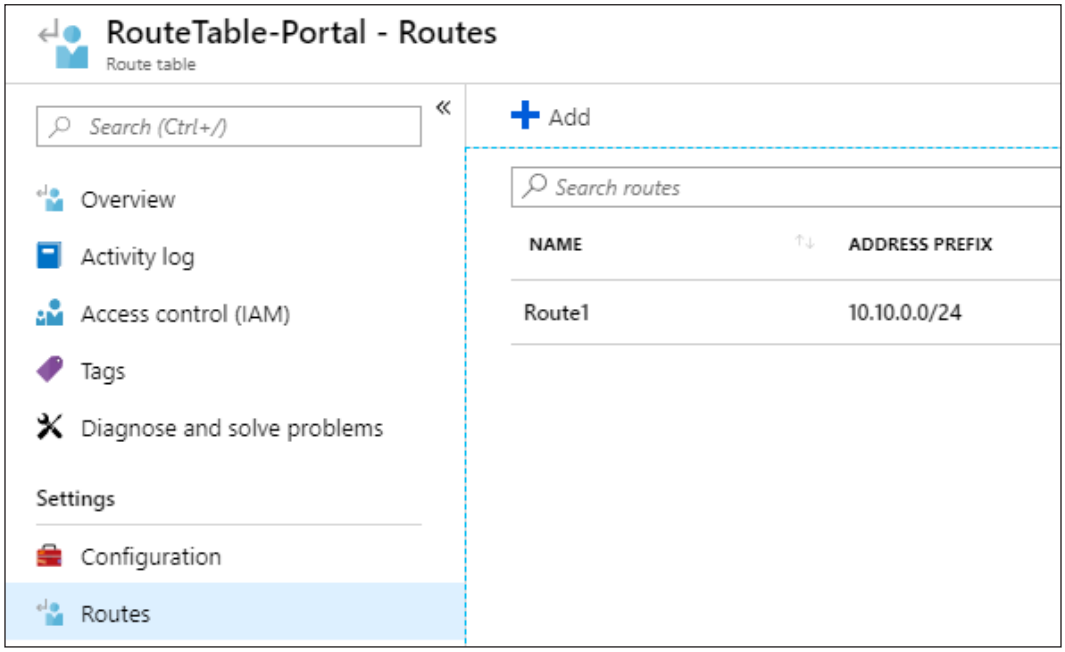
Getting ready

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

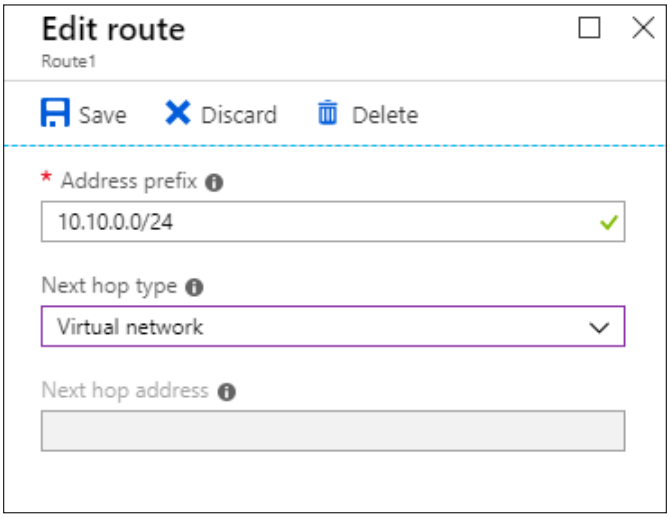
How to do it...

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:



3. A new blade will open. We can change the **Address prefix** (for destination IP range) and **Next hop type** settings. 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 when the service IP changes over time. For example, we may need to route all traffic through a virtual appliance, but the IP address of the virtual appliance changes over time. 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.

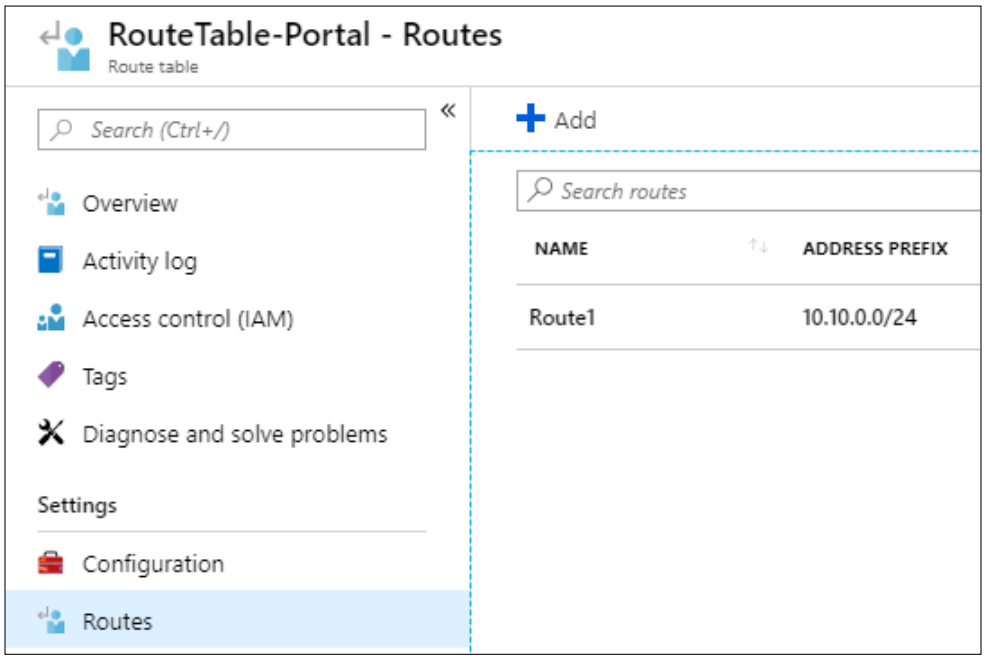
Getting ready

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

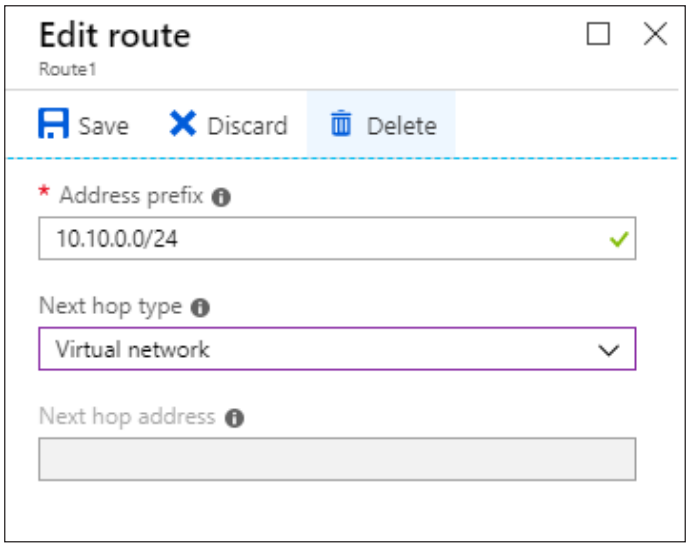
How to do it...

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

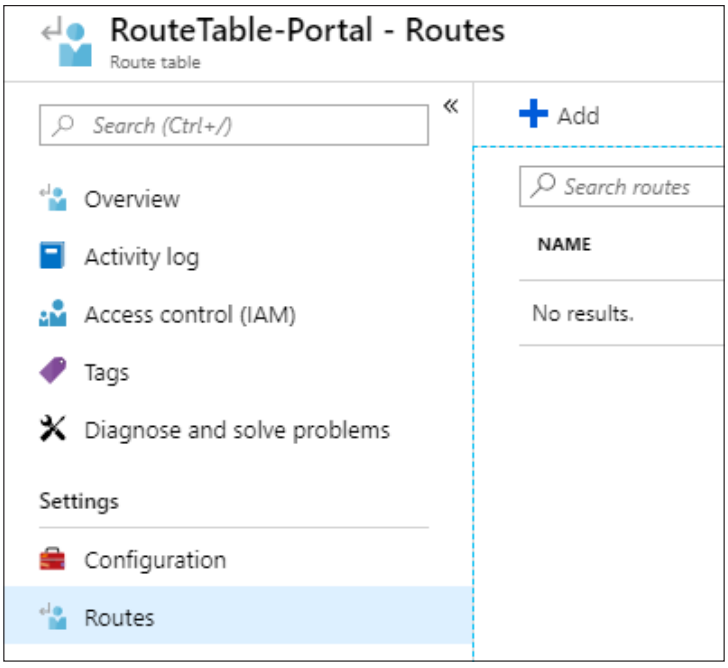
1. In the Azure portal, locate the **Route table** link.
2. Under **Settings**, select **Routes**, and then select the route you want to delete:



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 our requirements change, we need to address the new requirements in our route tables. We can either edit routes or remove them to meet these 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 our VNet) for a frontend IP address, and it targets the private IP addresses of our services (usually, an Azure **virtual machine** (**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.

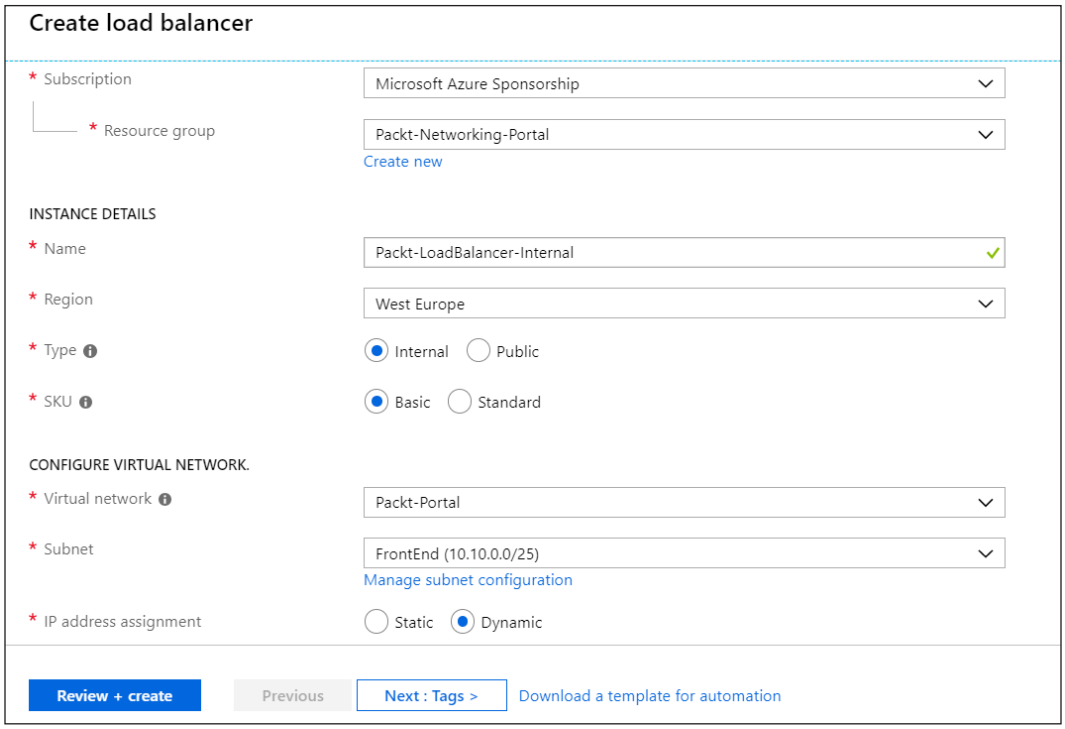
Getting ready

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

How to do it...

In order to create a new internal load balancer with the Azure portal, 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 resource group for where the load balancer is to be created. Then, we must provide information for the **Name**, **Region**, **Type** and **SKU** options. In this case, we select **Internal** for **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**:



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 **demilitarised 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 come over the internet. The requests are then distributed to the endpoints in the backend.

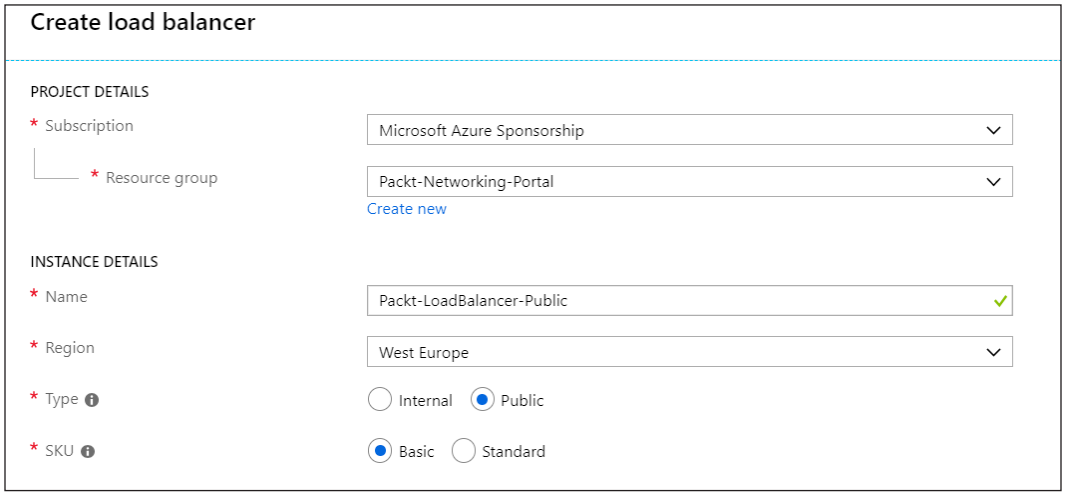
Getting ready

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

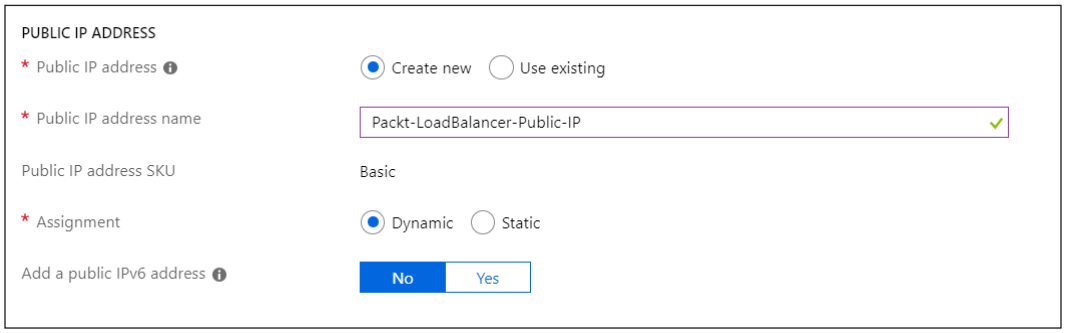
How to do it...

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 for where the load balancer is to be created. Then, we must provide information for **Name**, **Region**, **Type** and **SKU**. In this case, we select **Public** for **Type** to deploy a public load balancer:



3. Selecting **Public** as the load balancer type will slightly change the blade. We no longer have the option to select VNet and subnet, as we did for the internal load balancer. Instead, we can choose options for a **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:



4. 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 come 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 first define a backend pool.

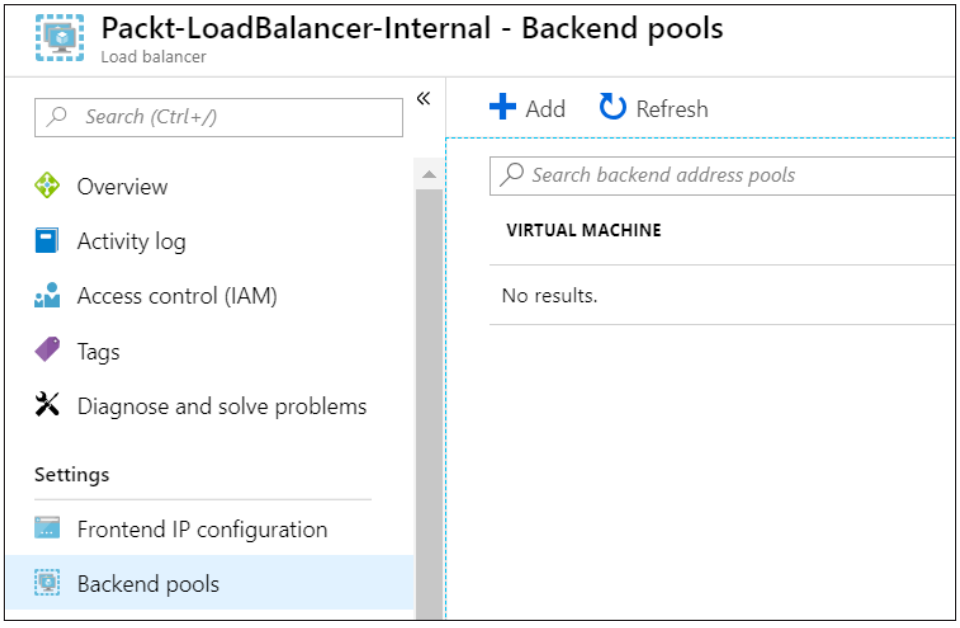
Getting ready

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

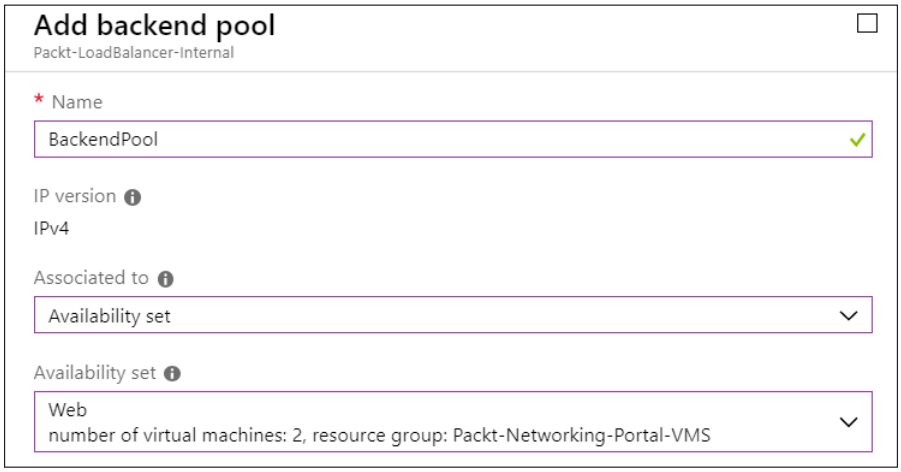
How to do it...

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

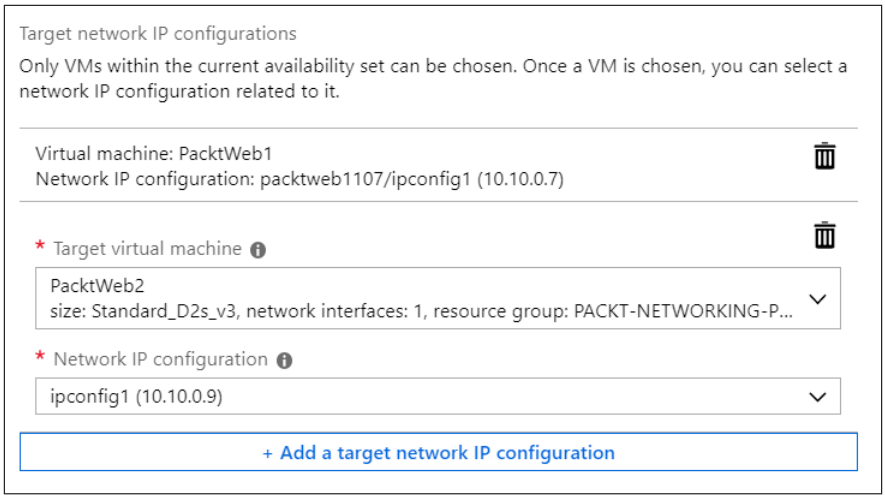
1. In the Azure portal, locate the previously created load balancer (either internal or public).
2. In the **Load balancer** blade, under **Settings**, select **Backend pools**. Select **Add** to add the new backend pool:



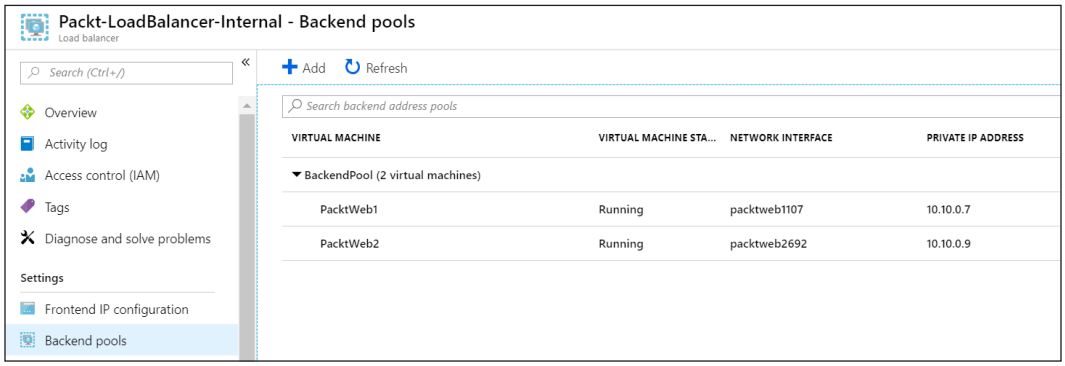
3. In the new blade, we must provide a name and specify what the load balancer is associated with. Associations can be created for a single VM, availability set or virtual machine scale set. I recommend using **Availability set**. Based on this selection, you will be offered a list of available resources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



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:



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 should the issue remain undetected. 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.

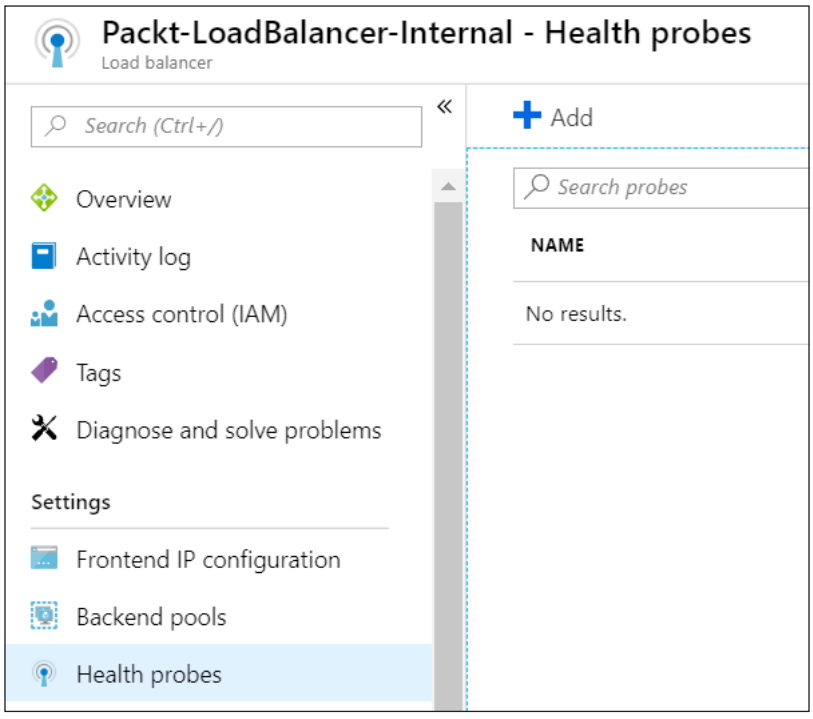
Getting ready

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

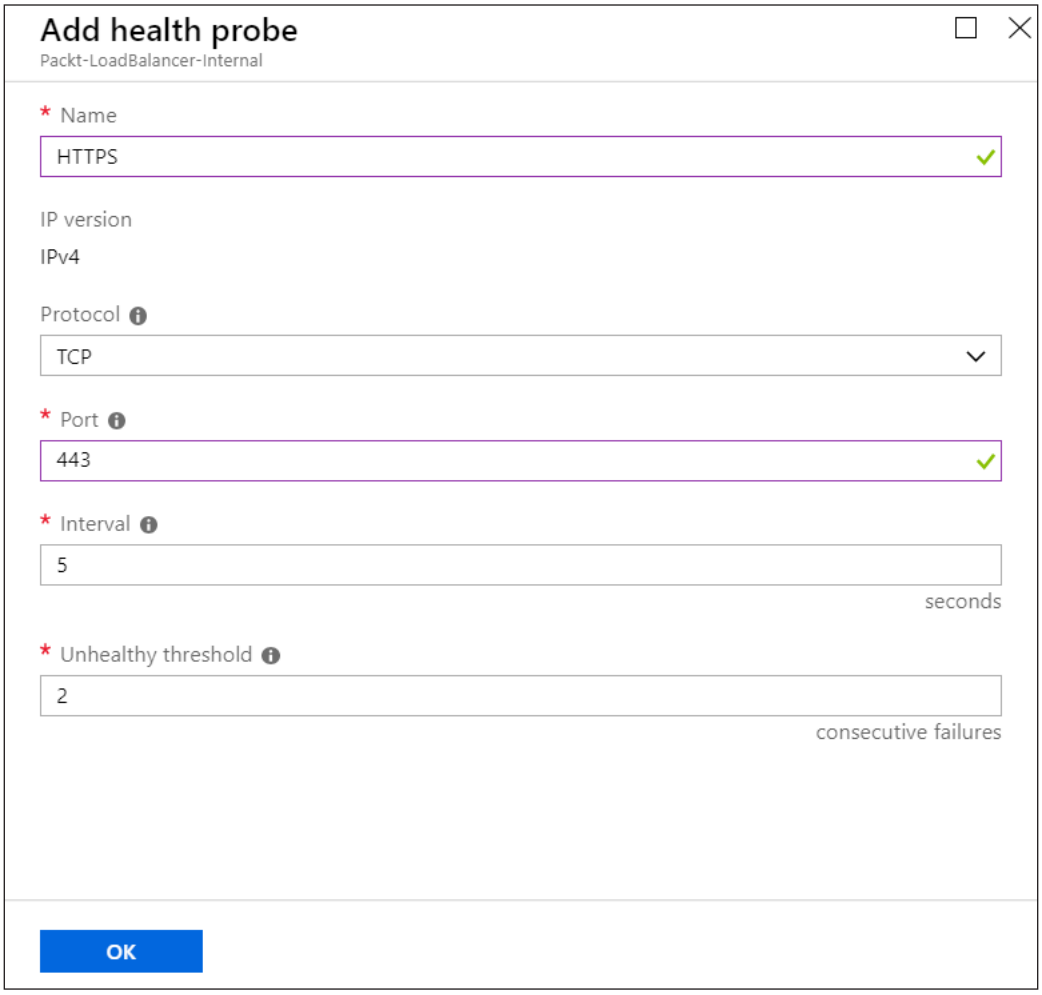
How to do it...

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:



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:



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 will 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 unavailable. So, the protocol and the port will detect changes in the 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 the puzzle when speaking of Azure load balancers is the **rules**. 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 the backend.

Getting ready

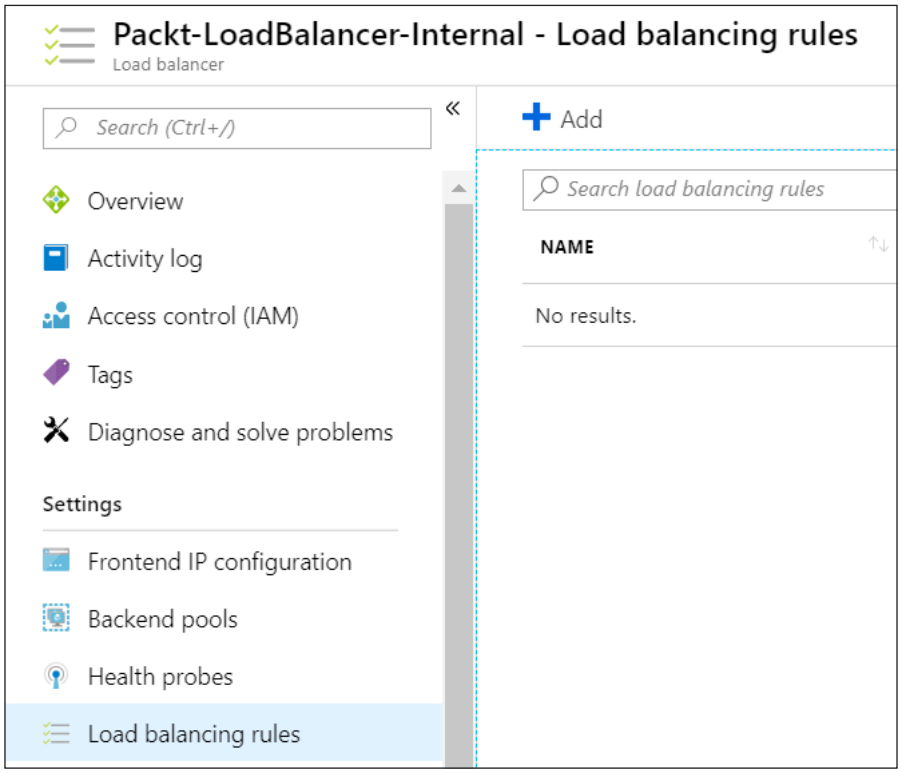
Before you start, open your browser and go to the Azure portal at `https://portal.azure.com`.

How to do it...

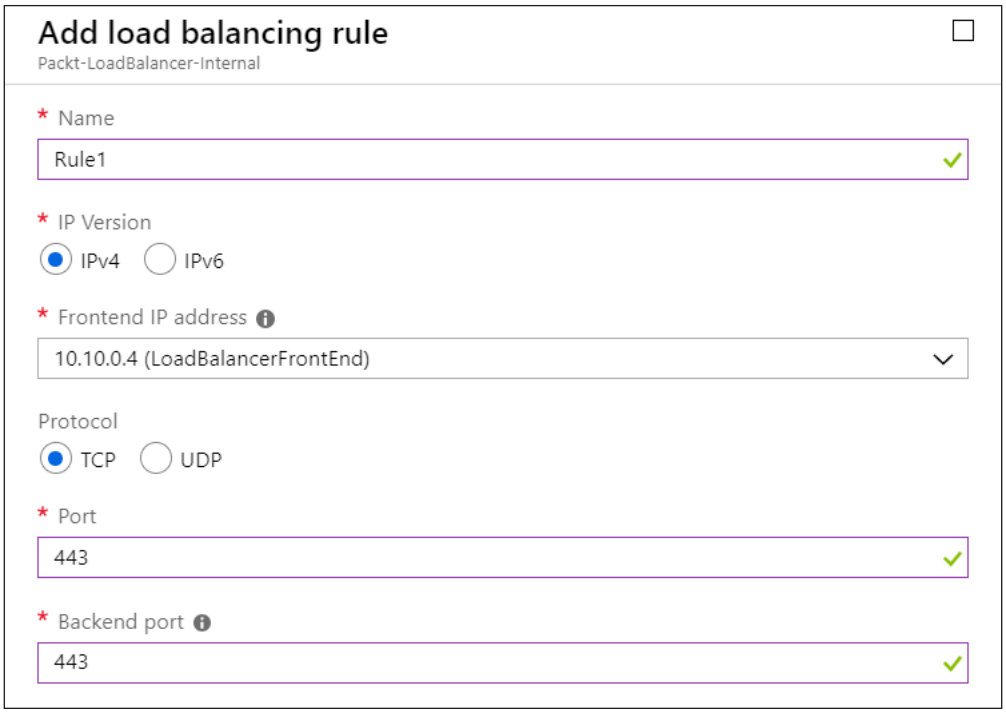
In order to create a load balancer rule, we must do the following:

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

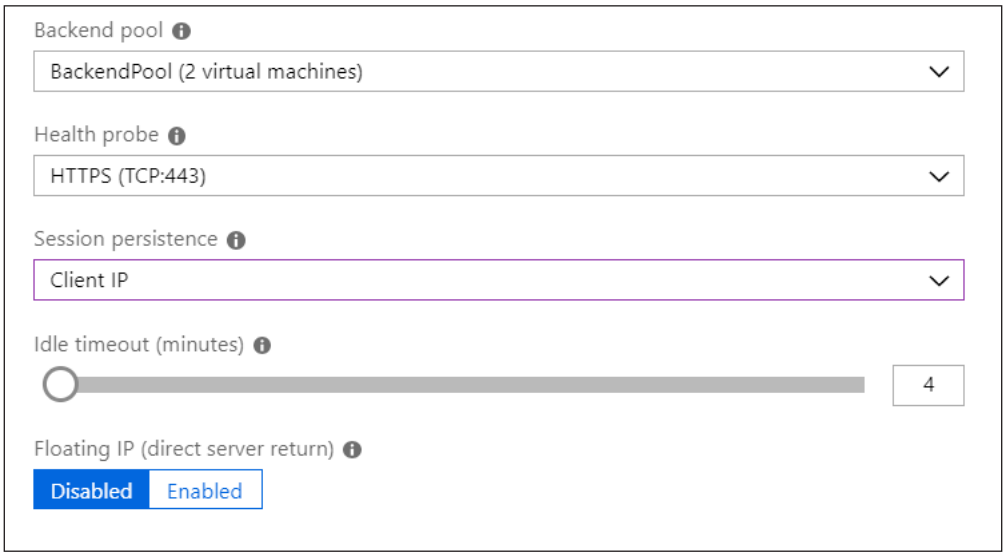
2. In the **Load balancer** blade, under **Settings**, select **Load balancing rules**. Select **Add** to add the load balancing rule:



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**):



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



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 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 a 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 NAT rules

Inbound **Network Address Translation** (**NAT**) rules are optional settings in the Azure load balancer. These rules essentially create another port mapping from the frontend to the 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.

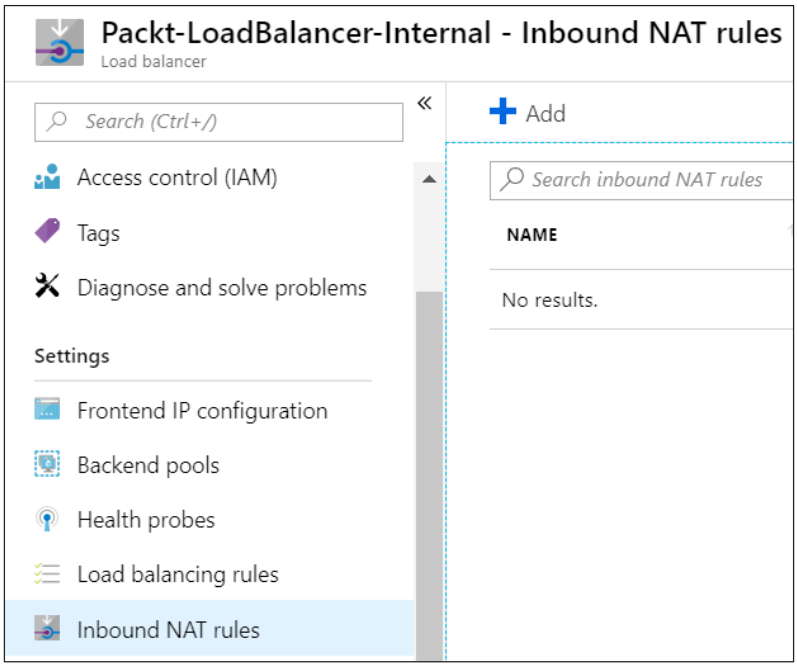
Getting ready

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

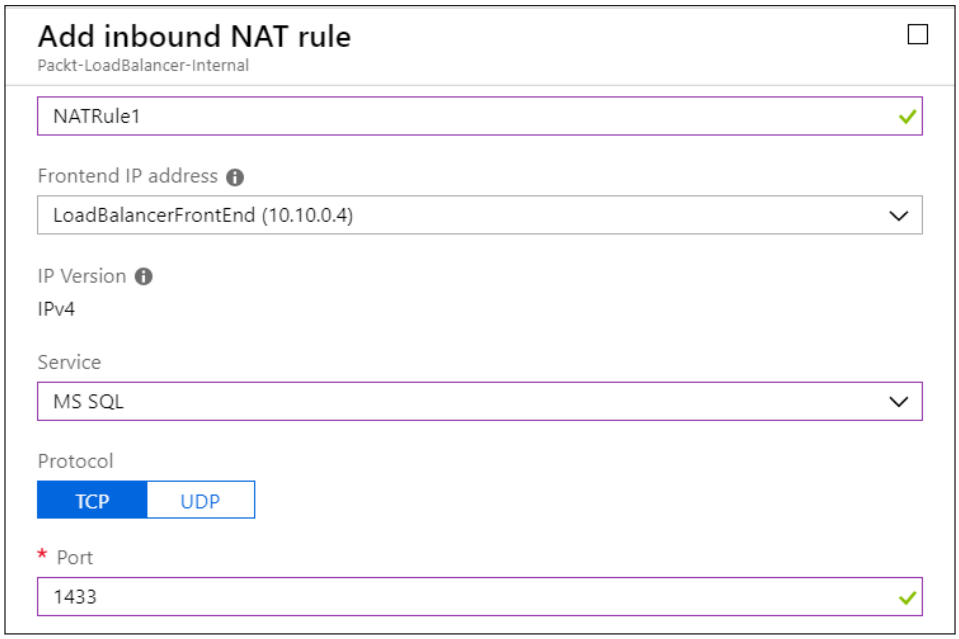
How to do it...

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

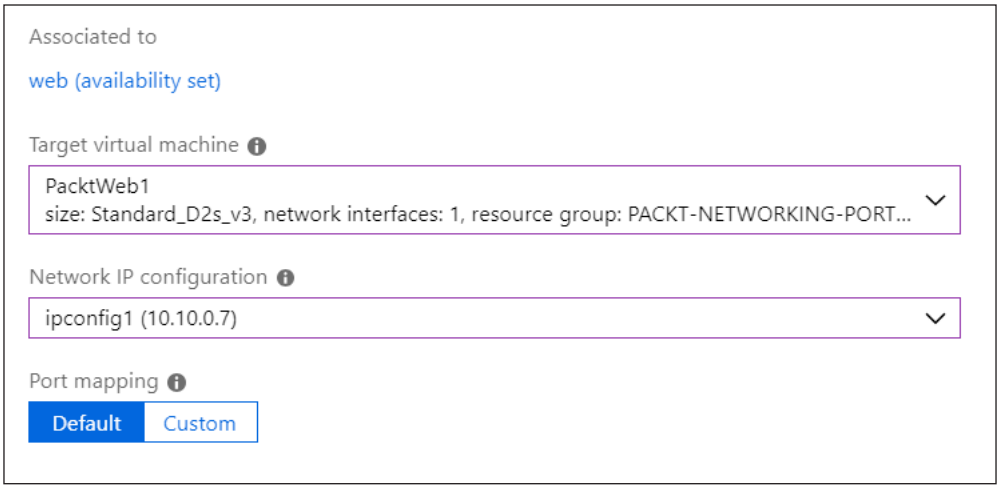
1. In the Azure portal, locate the previously created load balancer (either internal or public).
2. In the **Load balancer** blade, under **Settings**, select **Inbound NAT rules**. Select **Add** to add a new inbound NAT rule:



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



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 a port mapping similar to the port mapping created with the load balancer rule. The load balancer rule creates additional settings, such as the health probe or session persistence. Inbound NAT rules exclude these settings and create an 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 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, a 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; 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 endpoints
- 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 a new Traffic Manager profile.

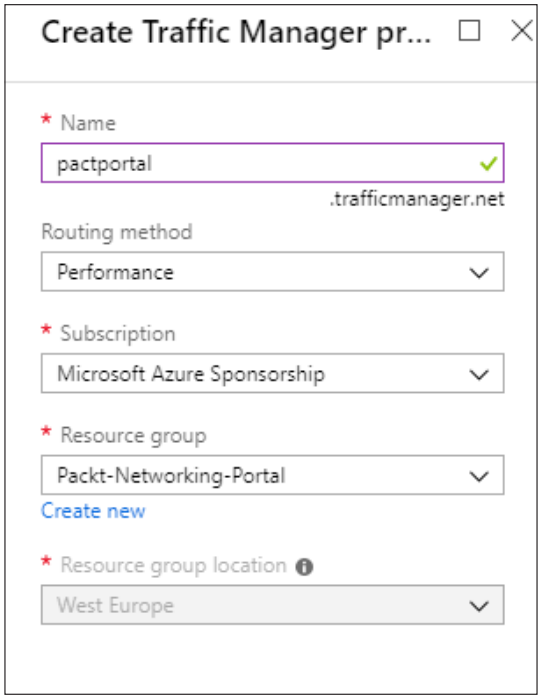
Getting ready

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

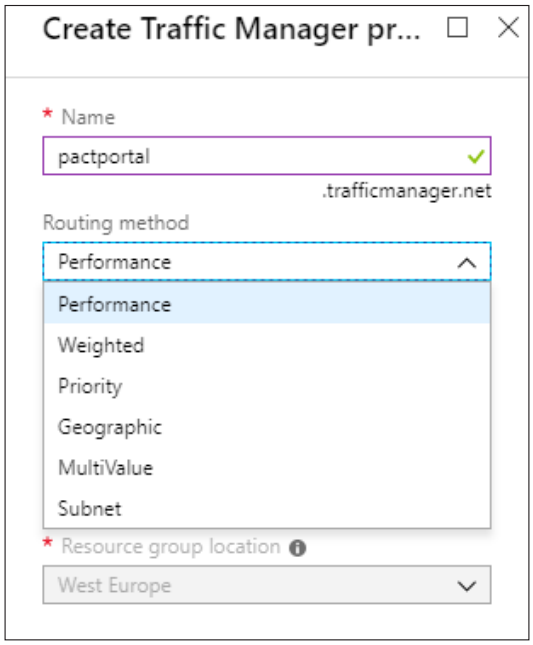
How to do it...

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:



3. Note that under **Routing method**, we have multiple options to choose from – **Performance**, **Weighted**, **Priority**, **Geographic**, **MultiValue** and **Subnet**. For 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 at 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 a new endpoint to Traffic Manager.

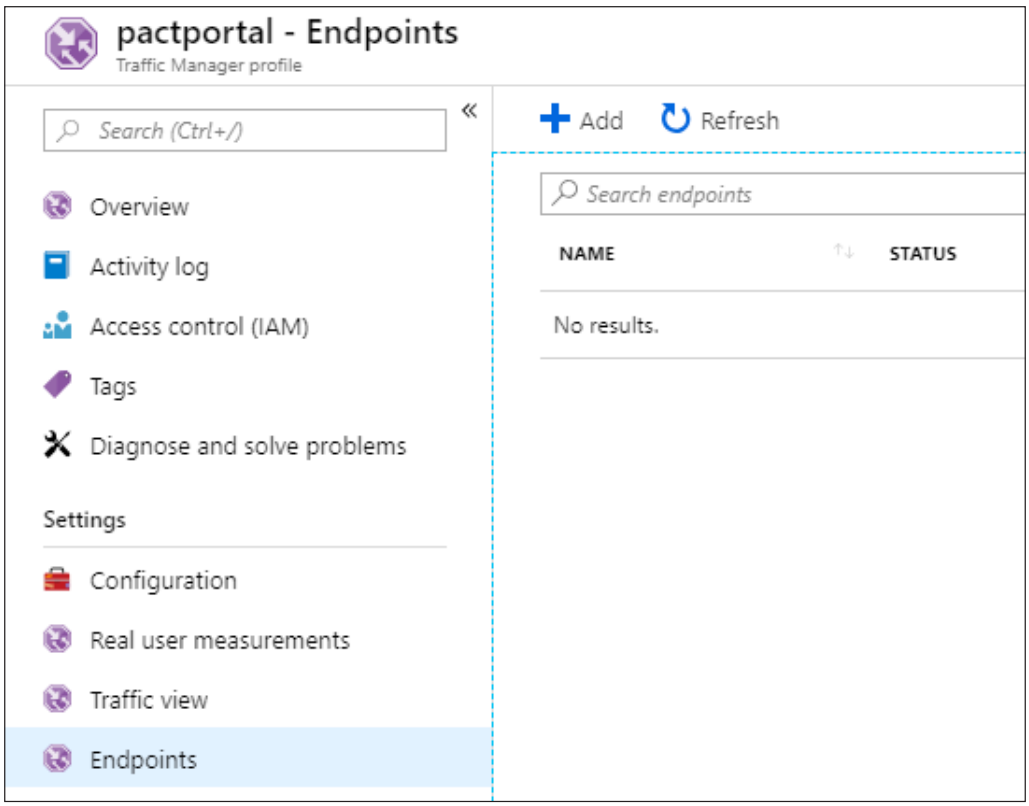
Getting ready

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

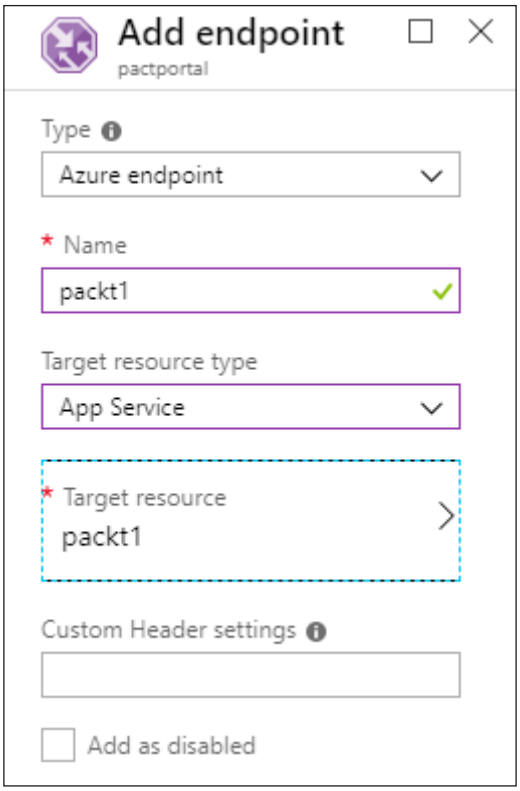
How to do it...

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

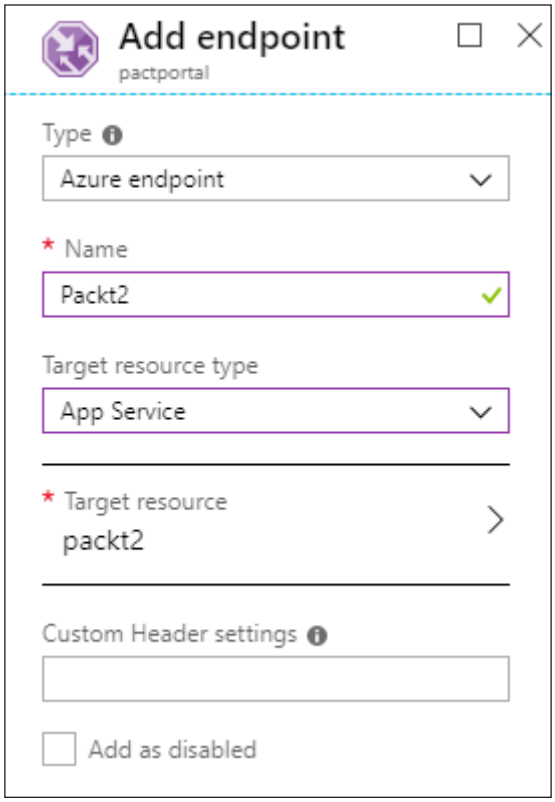
1. In the Azure portal, locate the previously created Traffic Manager profile.
2. In the **Traffic Manager** blade, under **Settings**, select **Endpoints**. Select **Add** to add a new endpoint:



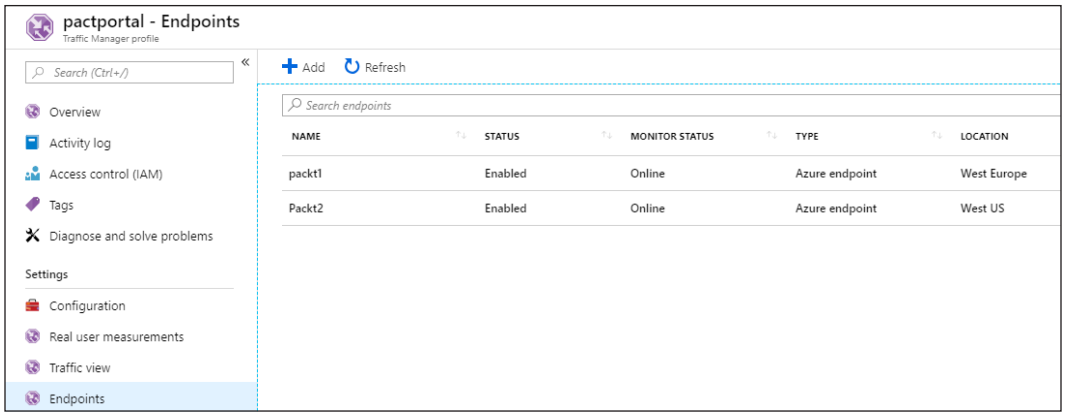
3. In the new blade, we need to provide information for the **Type** of endpoint we are adding and the **Name**. Based on the type, we can select certain target resource types and based on the target resource type selection, we can select resources that fit the target resource type selected:



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:



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



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) FQDNs or public IP addresses. 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.

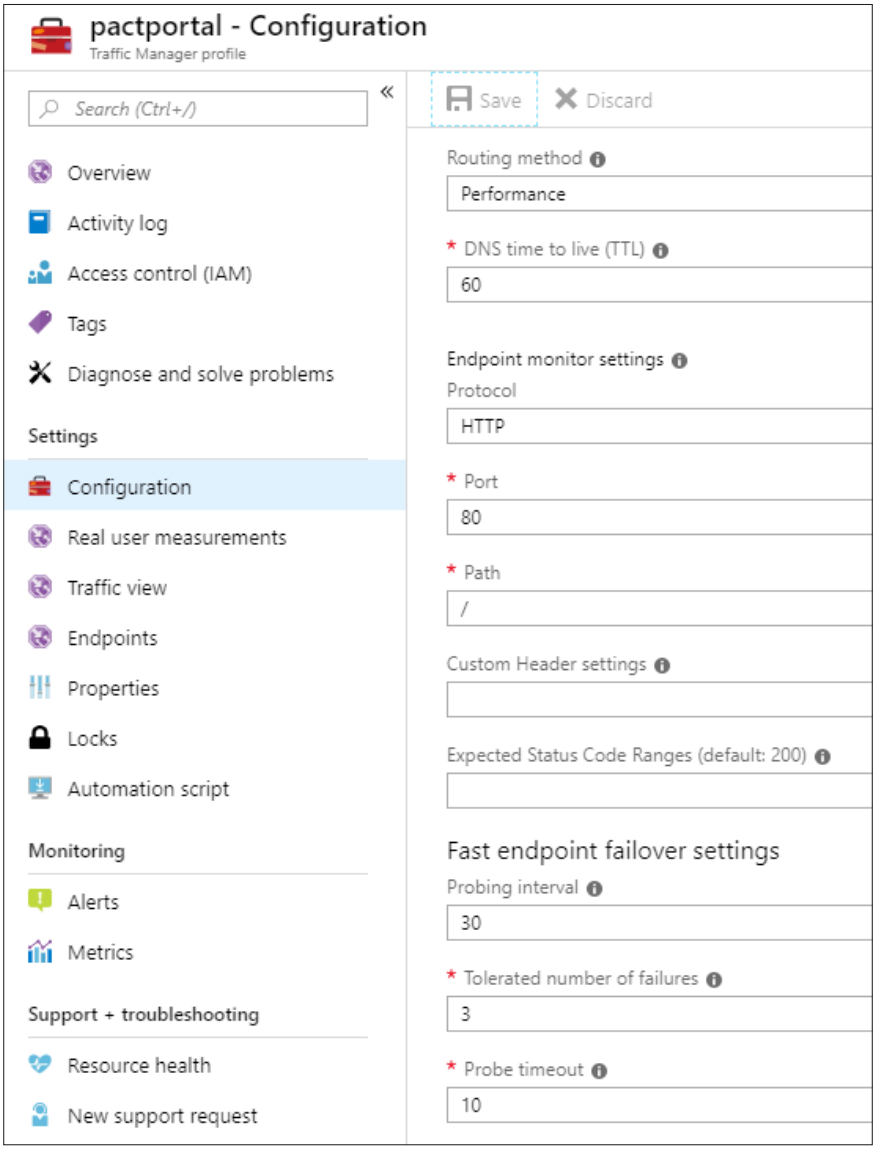
Getting ready

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

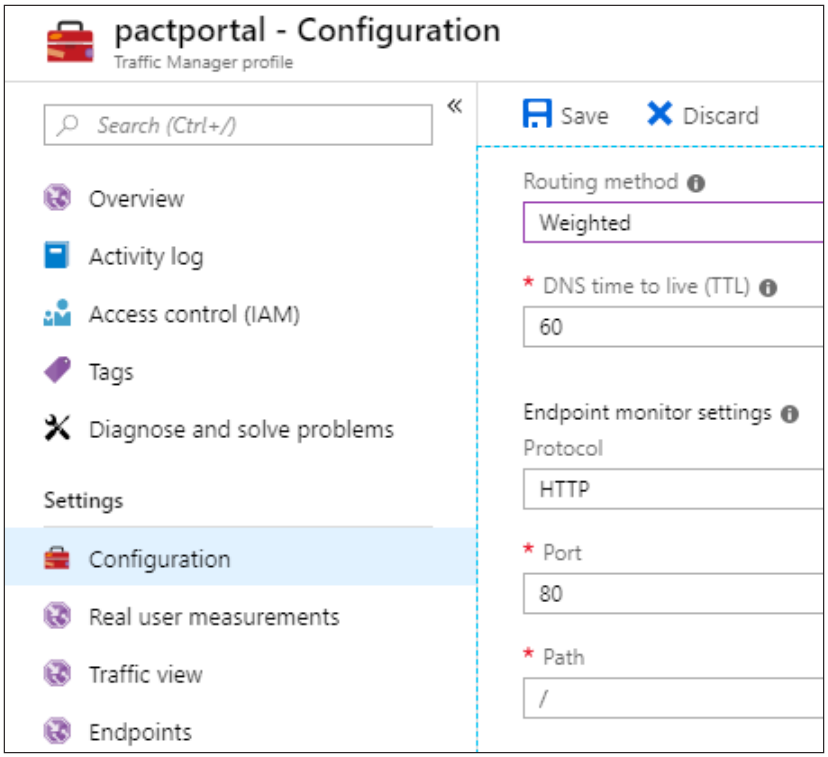
How to do it...

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)**, **Protocol** and failover settings:



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



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 backups. 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.

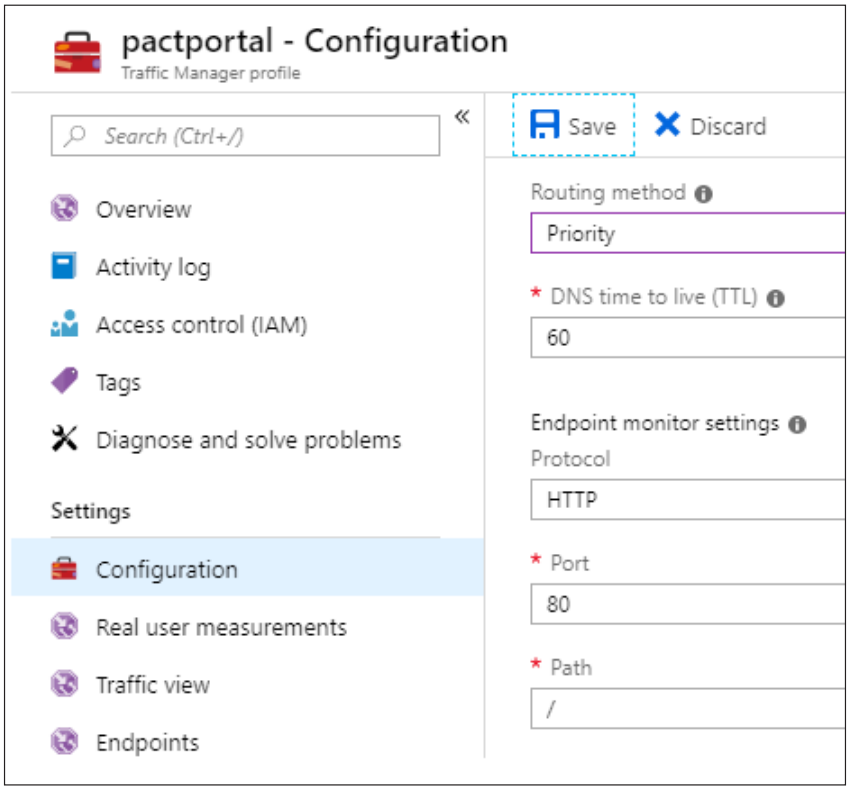
Getting ready

Before you start, open the browser and go to the Azure portal at `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:



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 endpoint added first becomes the one with the highest priority and the endpoint added last 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

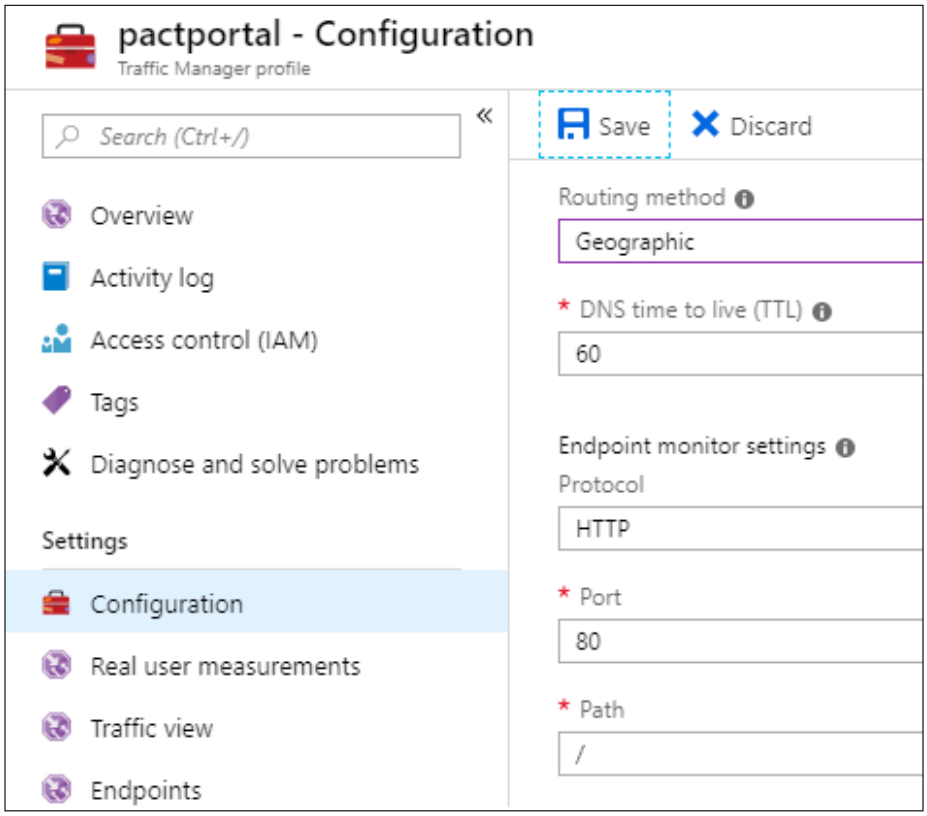
Before you start, open the browser and go to the Azure portal at `https://portal.azure.com`.

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:



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 requests coming from Europe will be pointed to the endpoint located in Europe.

Managing endpoints

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.

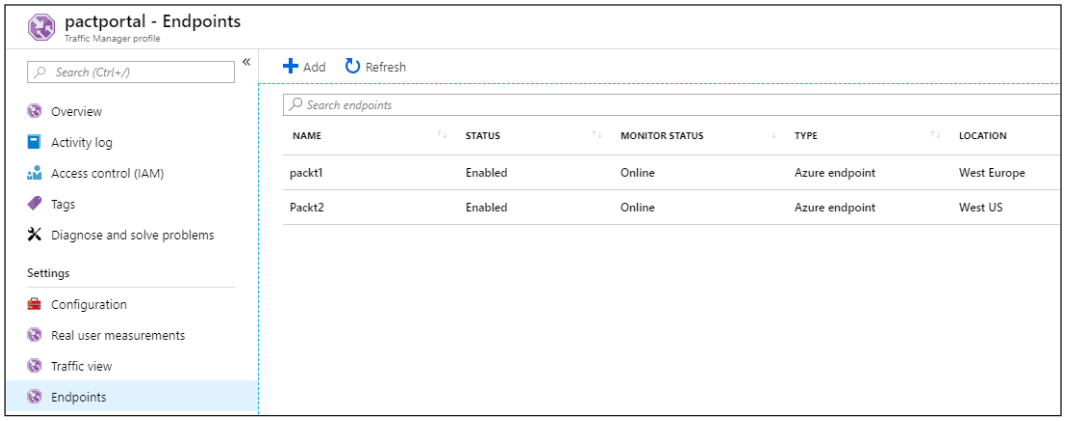
Getting ready

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

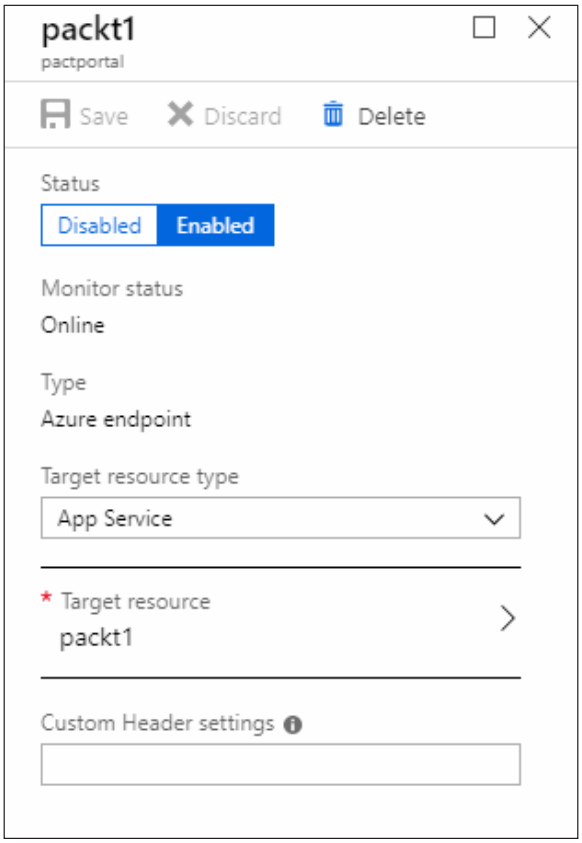
How to do it...

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:



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 our Traffic Manager profile.

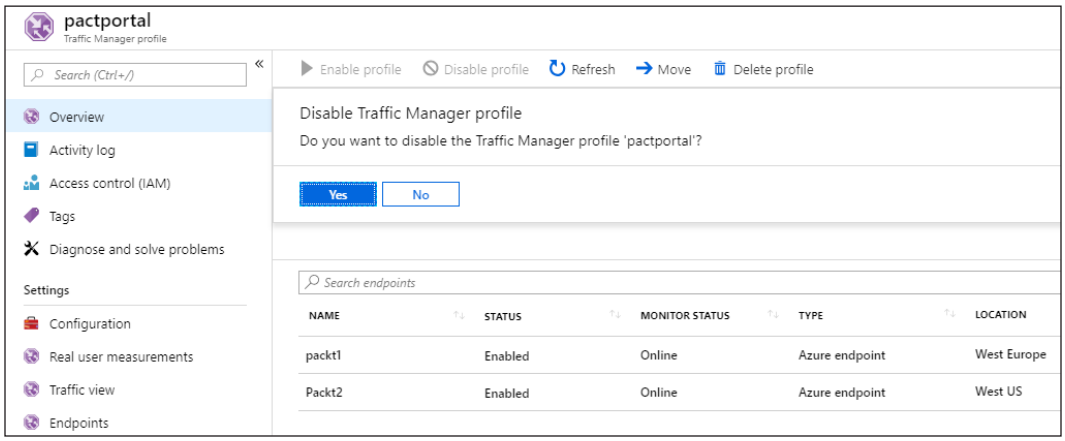
Getting ready

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

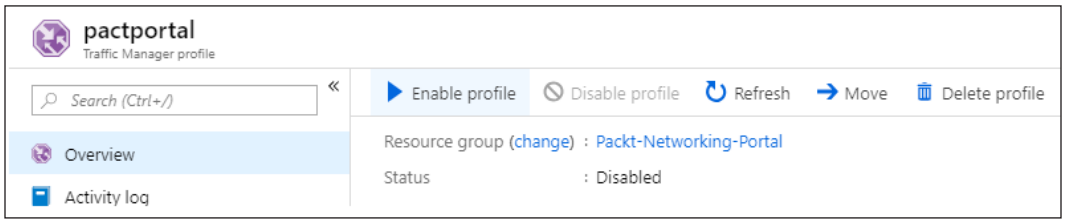
How to do it...

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:



3. Once the profile has been disabled, it can be enabled again with the **Enable profile** option:



How it works...

Managing the Traffic Manager profile with the disable and enable options will make the Traffic Manager frontend unavailable or available (depending 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 the 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.

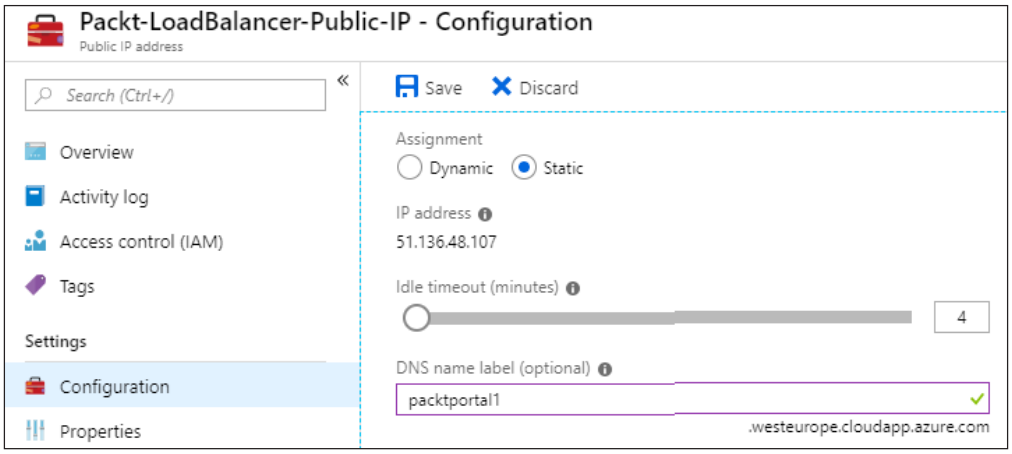
Getting ready

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

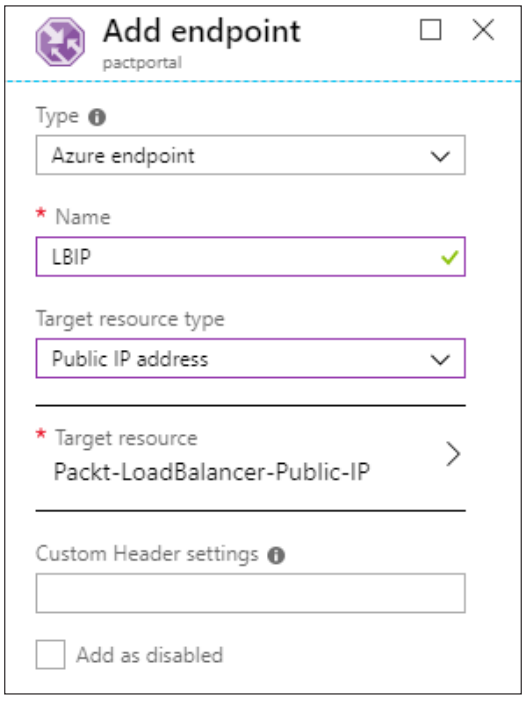
How to do it...

In order to set up Traffic Manager with a 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:



2. Go to Traffic Manager and select **Add** to add a new endpoint. Select **Azure endpoint** as the **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 will then 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 an active load balancer.

10
Azure Application Gateway

Azure Application Gateway is essentially a load balancer for web traffic, but it also provides you with better traffic control. Classic load balancers operate on the transport layer, and they allow you to route traffic based on protocol (TCP or UDP) and IP address, mapping IP addresses and protocols in the frontend to IP addresses and protocols in the backend. Azure 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 optimised for different things. If one of our servers is optimised for video, then all video requests should 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**)
- Customising 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 the frontend to the 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 optimise specific performance. Using these options and performing routing based on context will increase application performance, along with providing 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

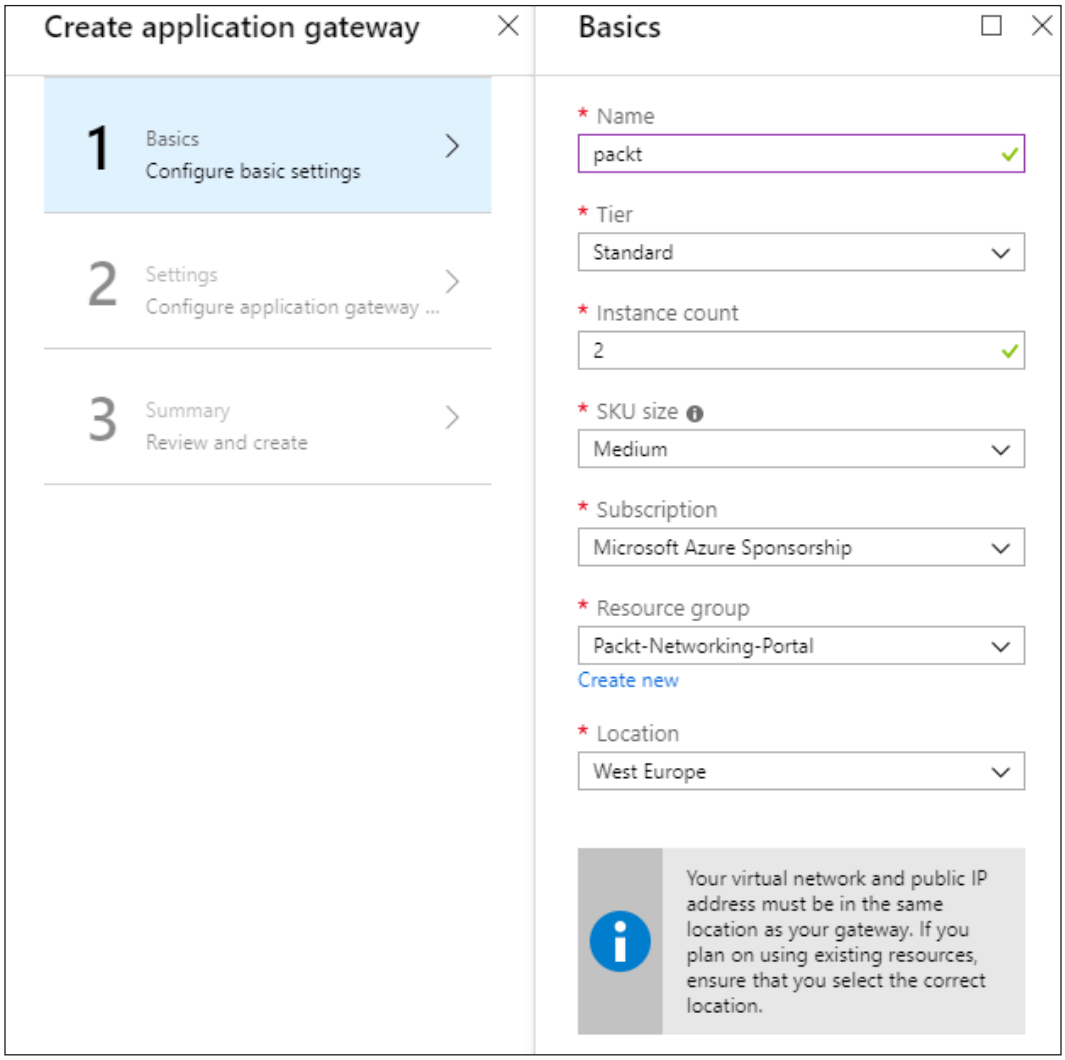
Before you start, open the browser and go to the Azure portal at `https://portal.azure.com`.

How to do it...

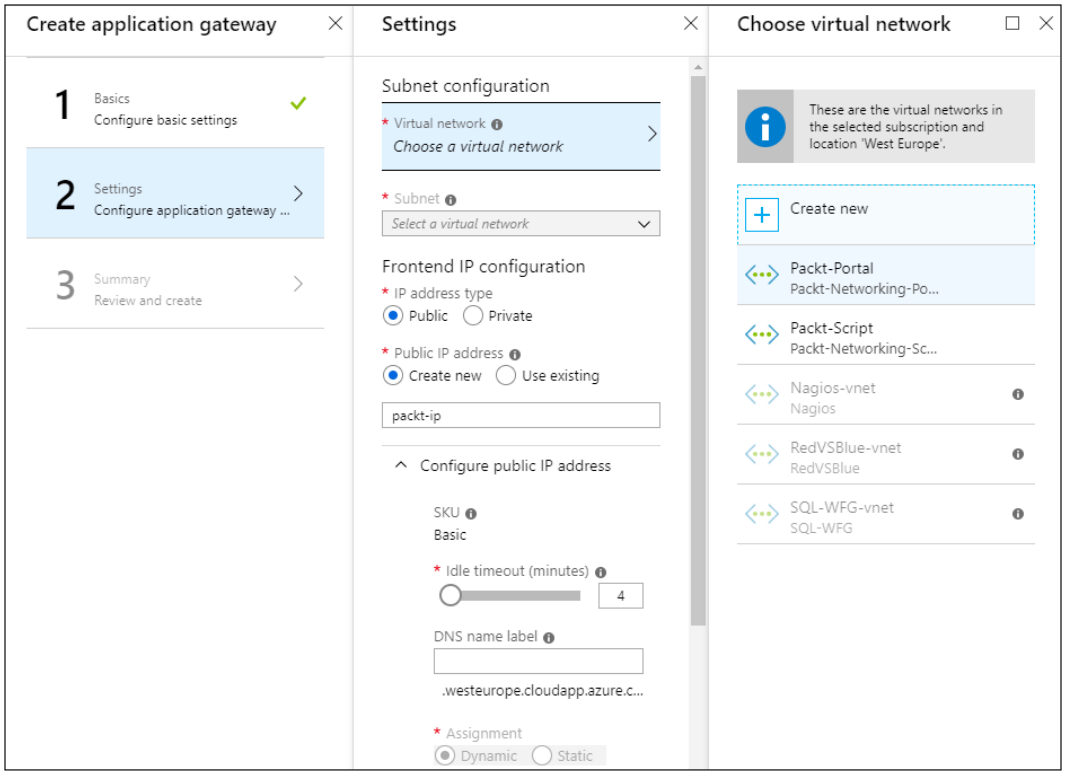
In order to create a new application gateway, we must do the following:

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).

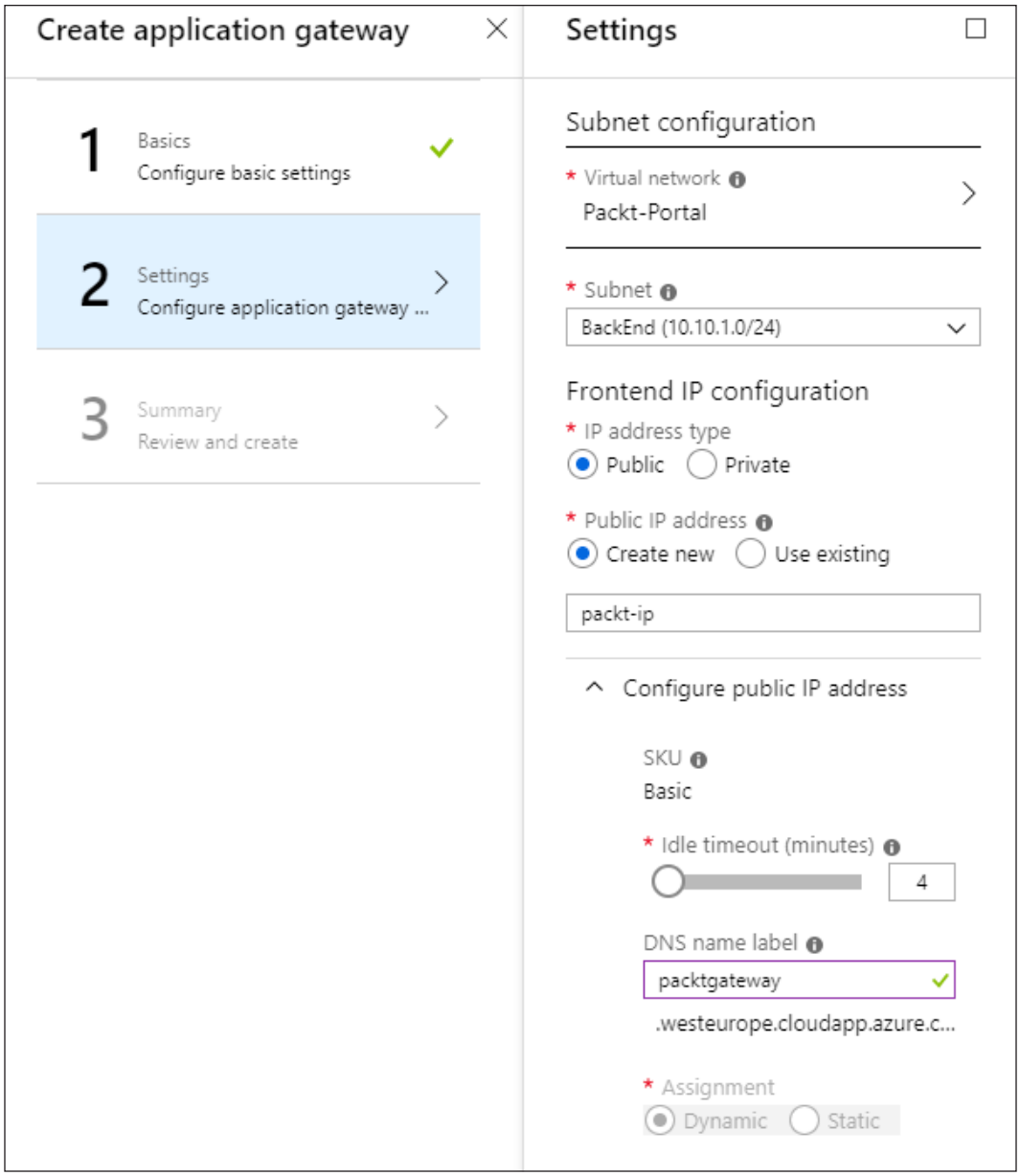
2. In the new blade, we must provide information for **Name**, **Tier**, **Instance count**, **SKU size**, **Subscription**, **Resource group** and **Location**. Note that the virtual network and public IP address (associated with the application gateway) must be in the same region as the application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



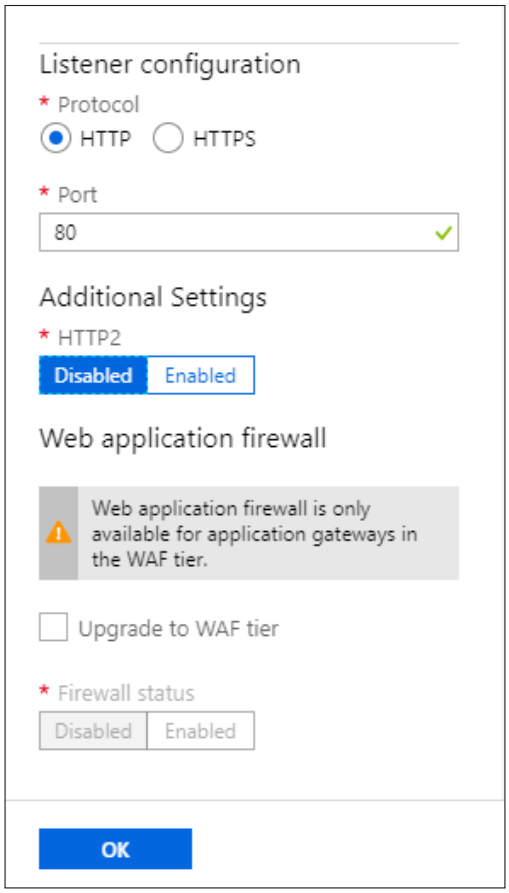
3. Under **Settings**, we must select the **Virtual network** that will be associated with our application gateway. You will be limited to virtual networks that are located in the region that is selected for the application gateway:



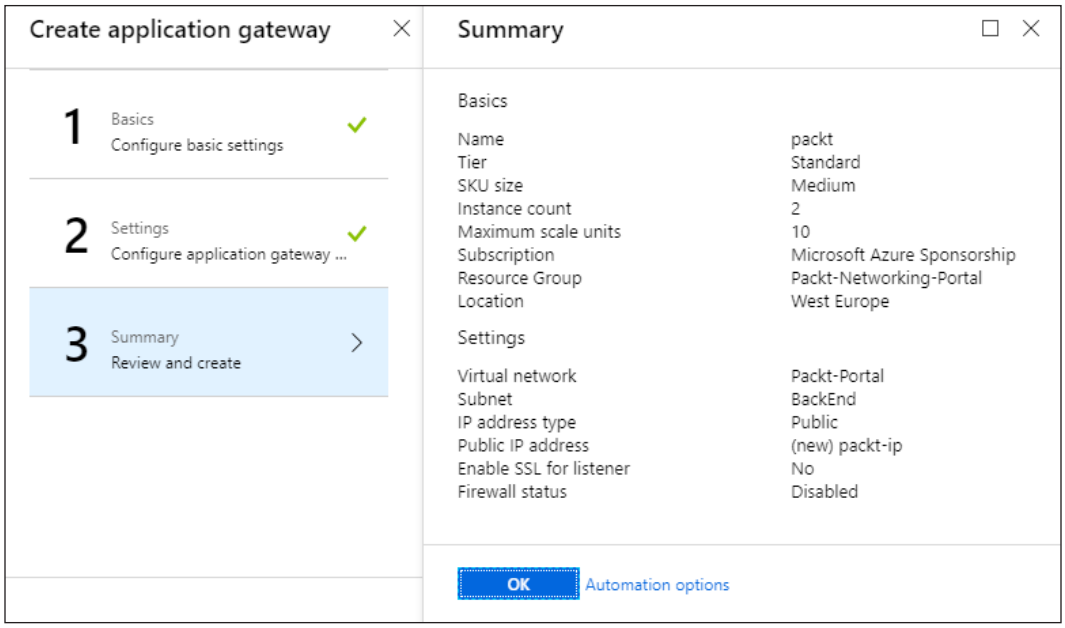
4. Based on your virtual network selection, we must also provide a **Subnet** from the virtual network associated with the application gateway. Further to this, we must provide information for the frontend IP address (select either the public or private address type). If **IP address type** is set to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


Create application gateway
1
Basics
Configure basic settings
2
Settings
Configure application gateway ...
3
Summary
Review and create
Settings
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

5. Finally, we must define the default listener. For the 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:



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



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 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 optimised for certain roles. For example, we can have multiple backend pools with different settings that are optimised to perform only specific tasks. Based on incoming requests, the application gateway will route the request to the 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 optimised 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.

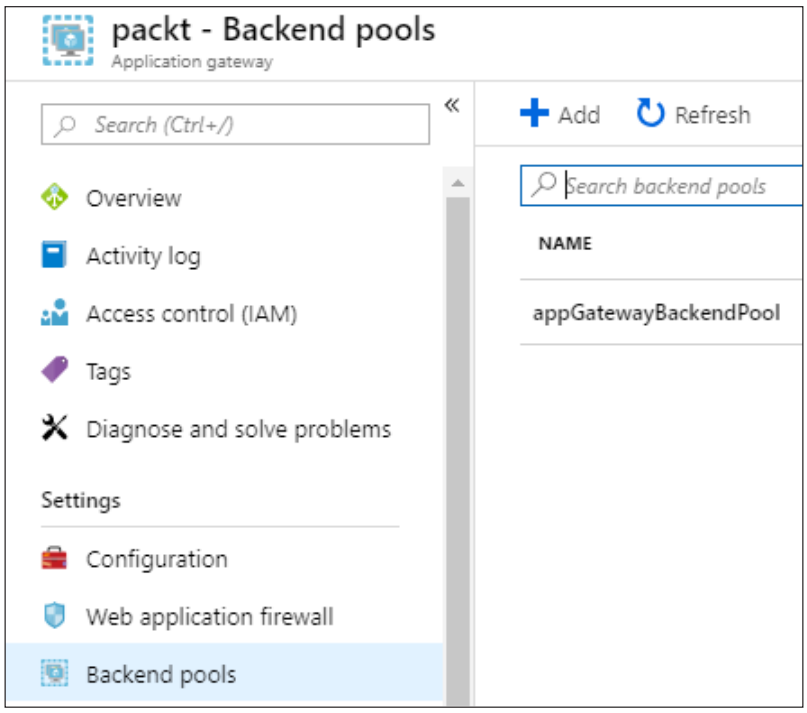
Getting ready

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

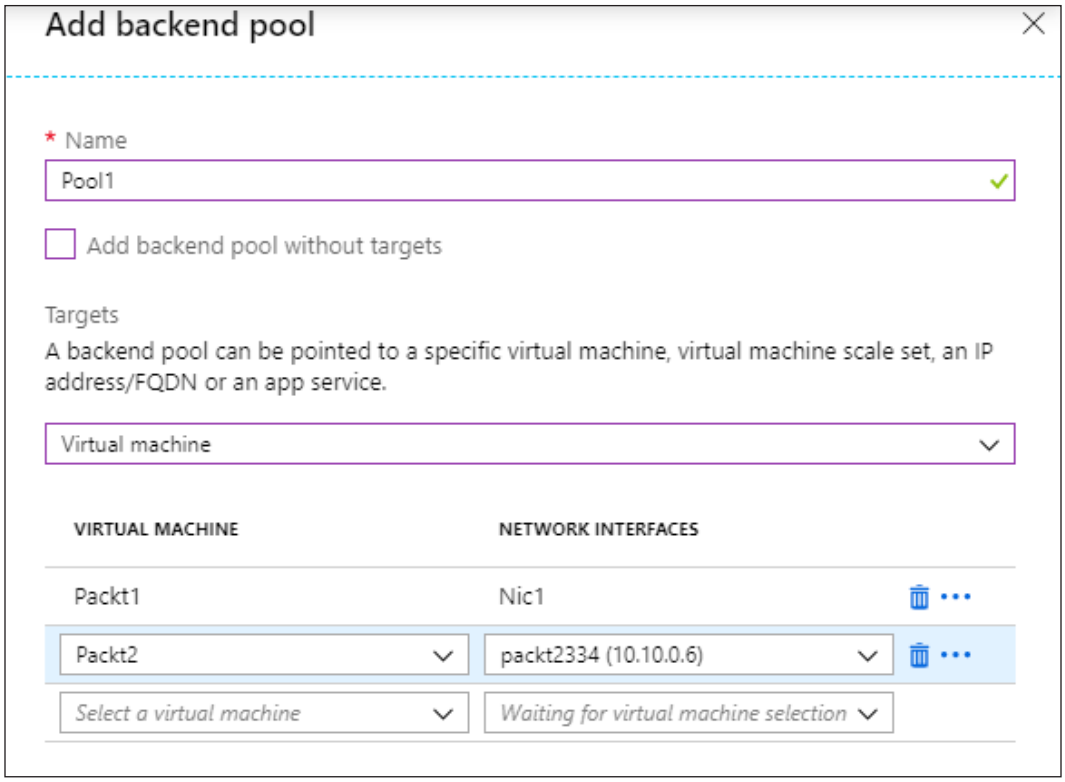
How to do it...

In order to add backend pools 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 **Backend pools**. Select **Add** to add a new backend pool:



3. In the new blade, we must provide the name of the backend pool and the type of target. Available types are virtual machines, virtual machine scale sets, app services and IP addresses/FQDNs. Based on the type you choose, 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 the 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 that's 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 and 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.

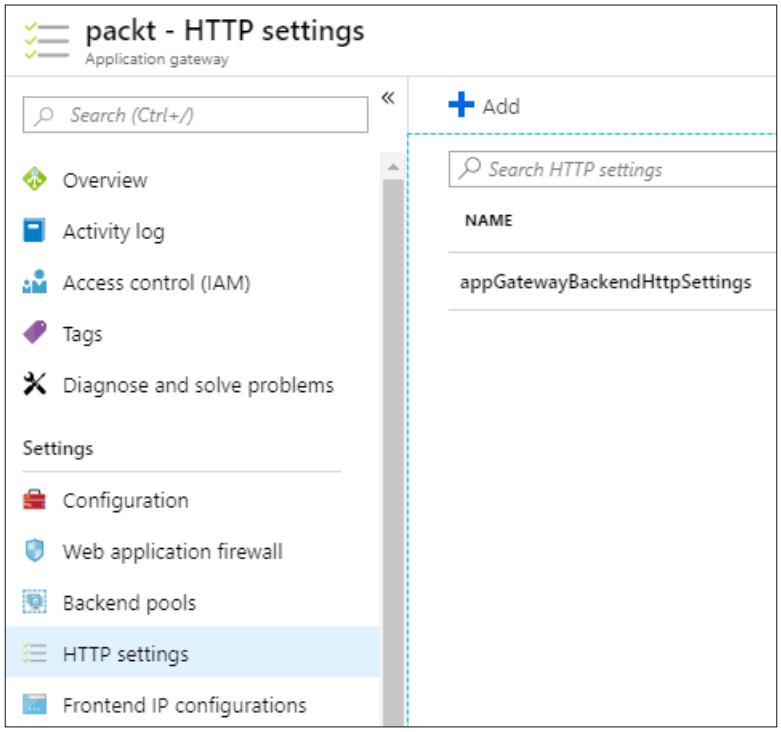
Getting ready

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

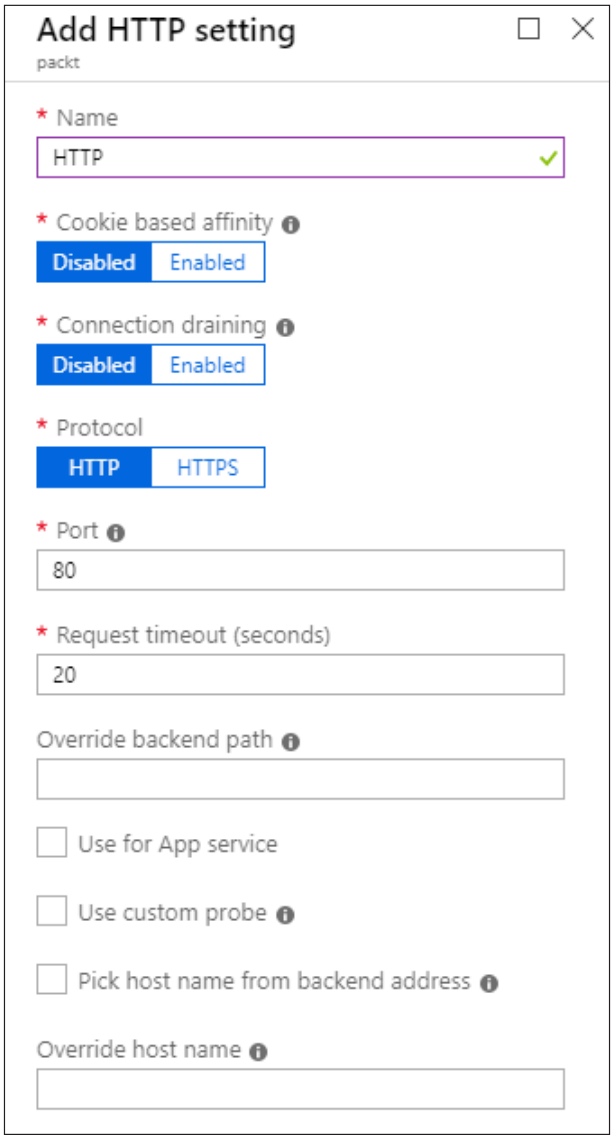
How to do it...

In order to add HTTP settings 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 **HTTP settings**. Select **Add** to add a new HTTP setting:



3. In the new blade, first, we need to provide a **Name**. The next options allow us to disable or enable options such as **Cookie based affinity** and **Connection draining**. Further to this, we select our **Protocol**, **Port** and the **Request time out (seconds)** period. Optional settings include using it for the app service or custom probes, or to pick a host name from the backend address pool. The override settings on offer 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.

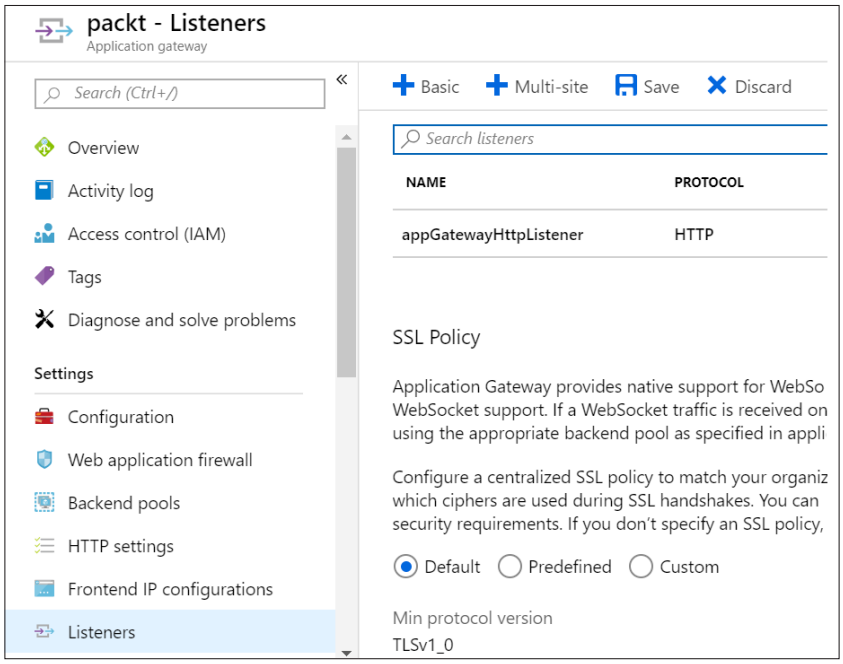
Getting ready

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

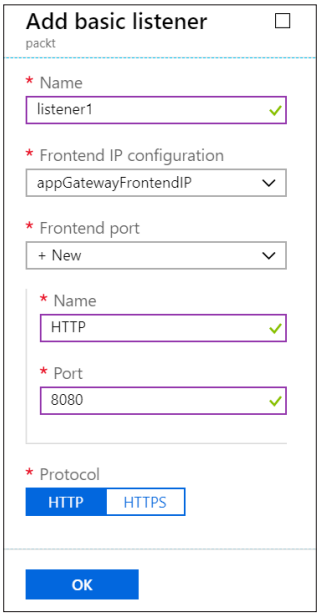
How to do it...

In order to add a listener 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 **Listeners**, then select **Basic** to add a new listener:



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



How it works...

A 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 in over 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.

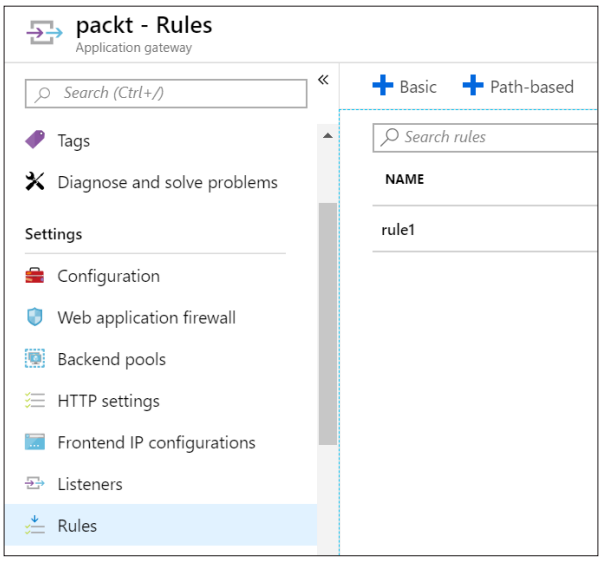
Getting ready

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

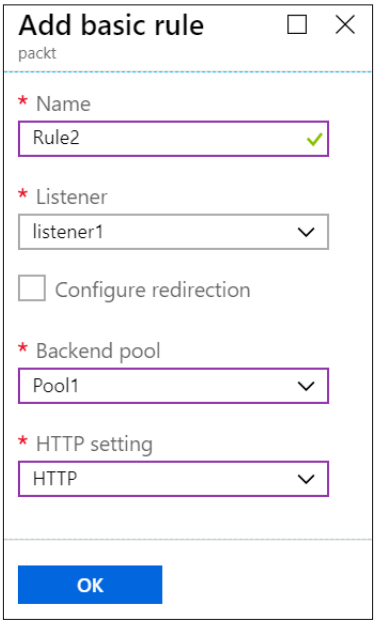
How to do it...

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...

Using rules, we can tie some previously created settings together. We define a listener that specifies which request on what IP address are we expecting over which port. 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, it 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.

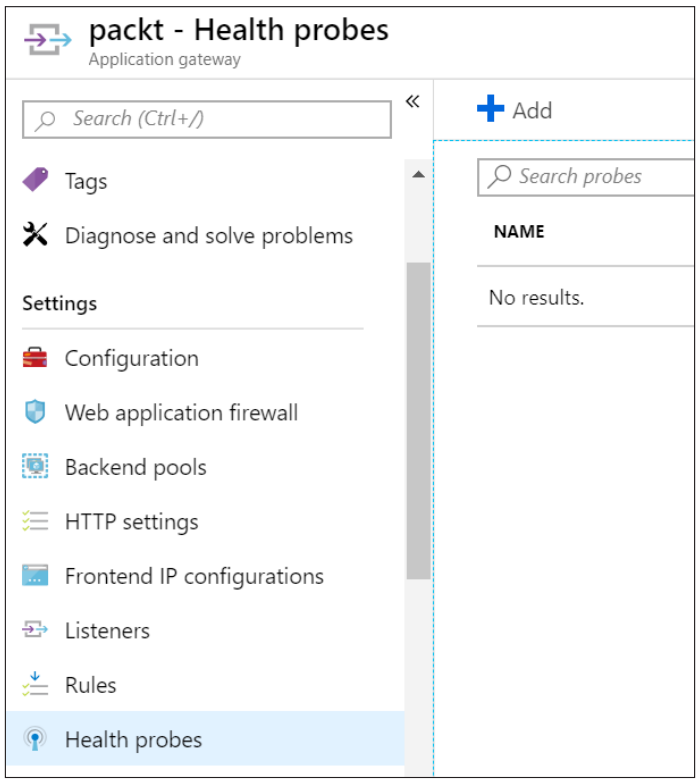
Getting ready

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

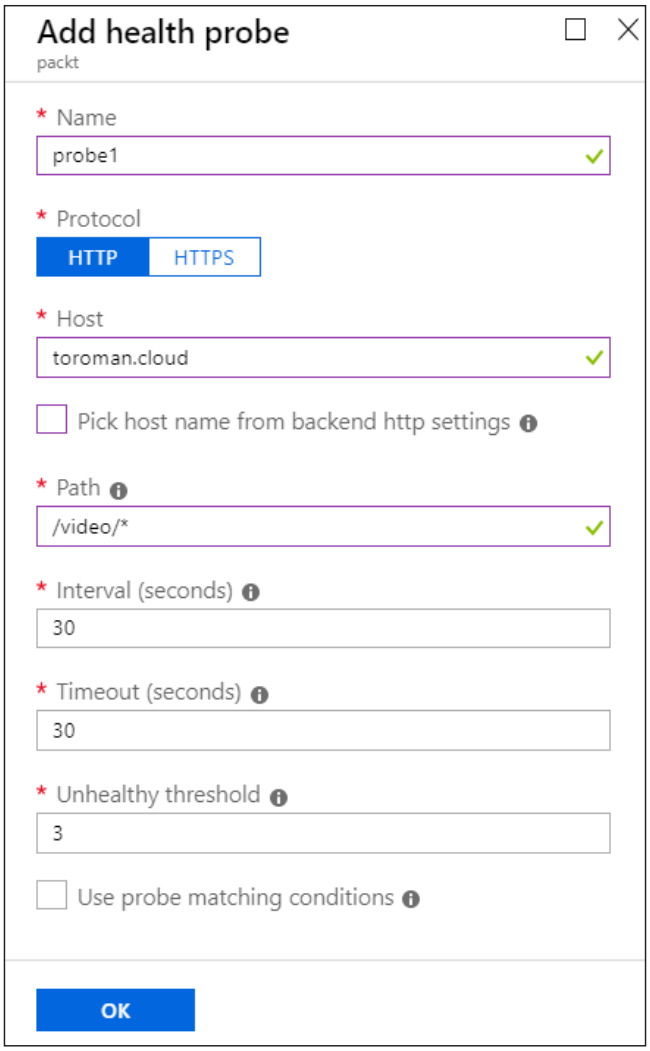
How to do it...

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**, **Time out** and **Unhealthy threshold**:



How it works...

Protocol, **Host** and **Path** define what probe is being monitored. **Interval** defines how often checks are performed. **Time out** 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.

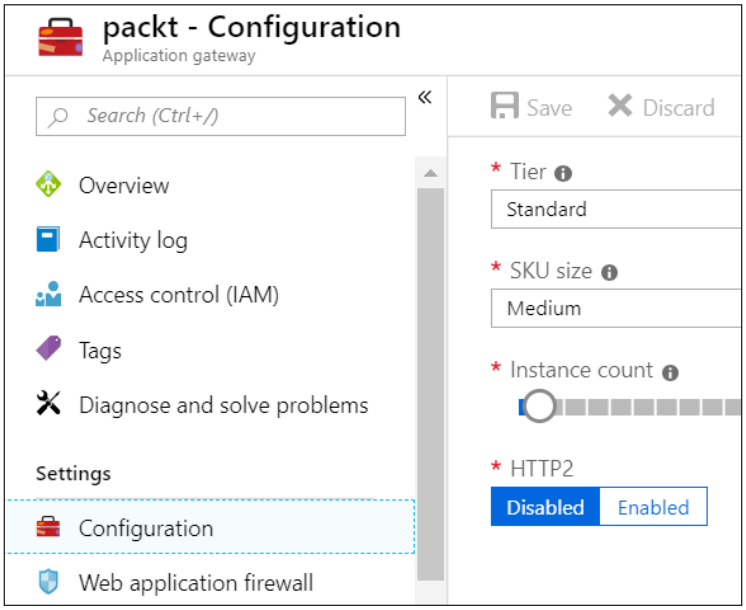
Configuring a Web Application Firewall (WAF)

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 centralised protection.

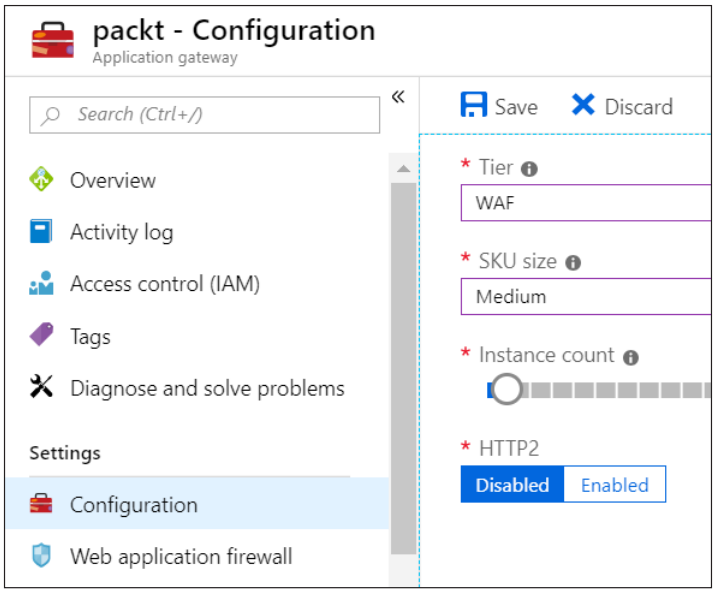
Getting ready

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:



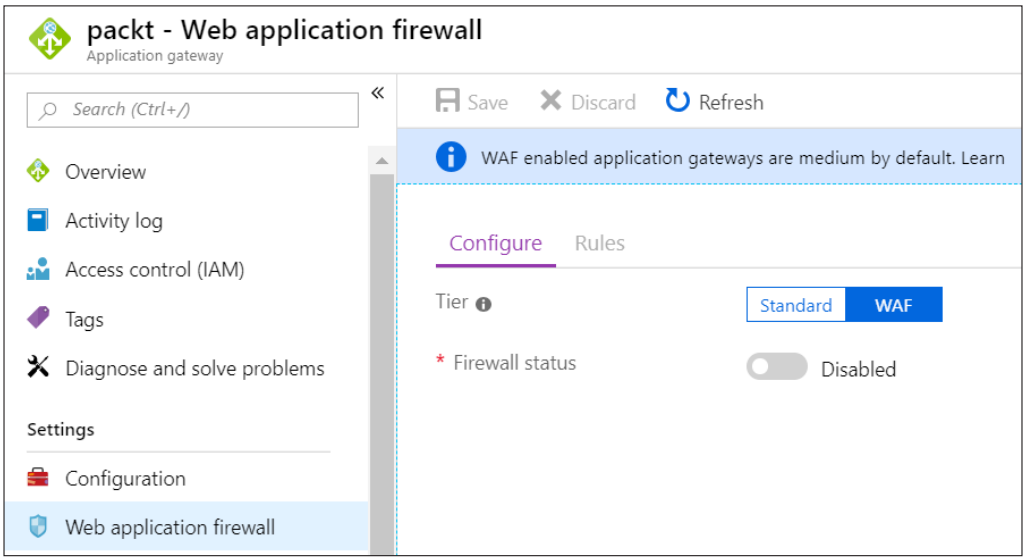
2. Change the **Tier** selection from **Standard** to **WAF**. Click the **Save** button:



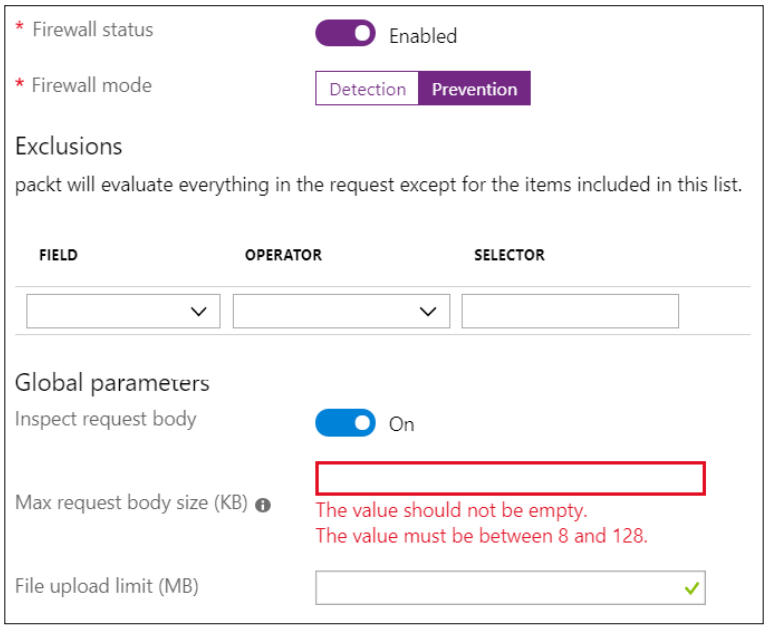
How to do it...

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

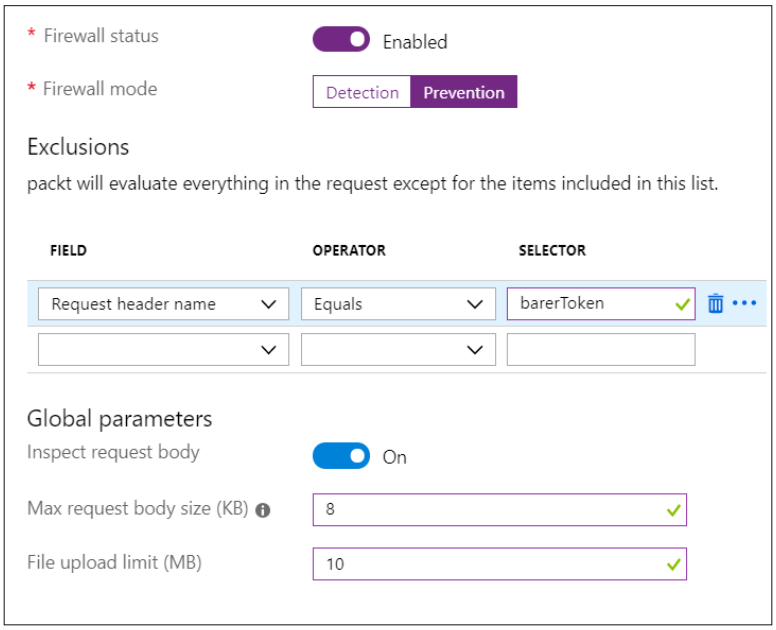
1. In the **Application gateway** blade, go to **Web application firewall**, under **Settings**:



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



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



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.

Customising 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 as needed.

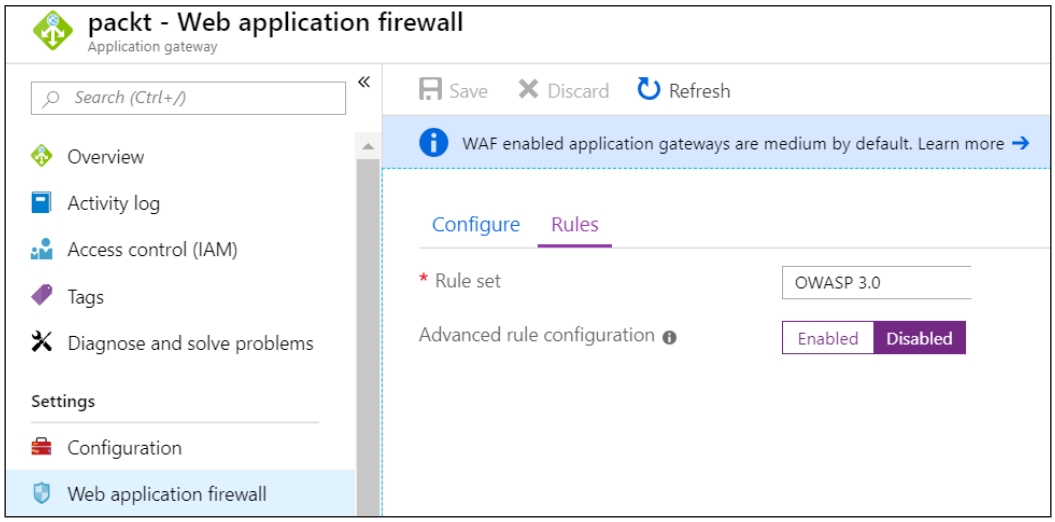
Getting ready

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

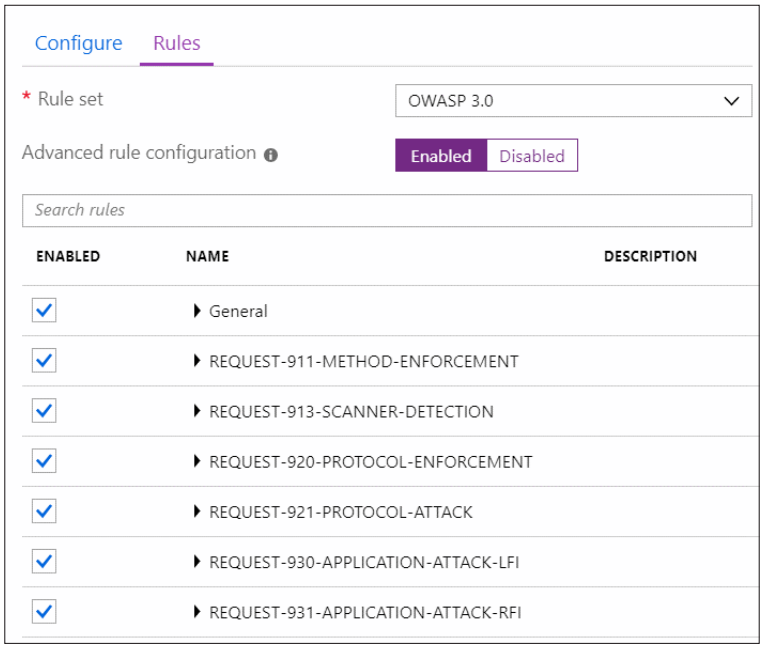
How to do it...

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. Set **Advanced rule configuration** to **Enabled**, as shown in the following screenshot:



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



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 our requirements.

11
Azure Firewall

Most Azure networking components used for security are there to stop unwanted incoming 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 that 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

Code examples can be found at `https://github.com/PacktPublishing/Azure-Networking-Cookbook/tree/master/Chapter11`.

Creating a new Azure Firewall

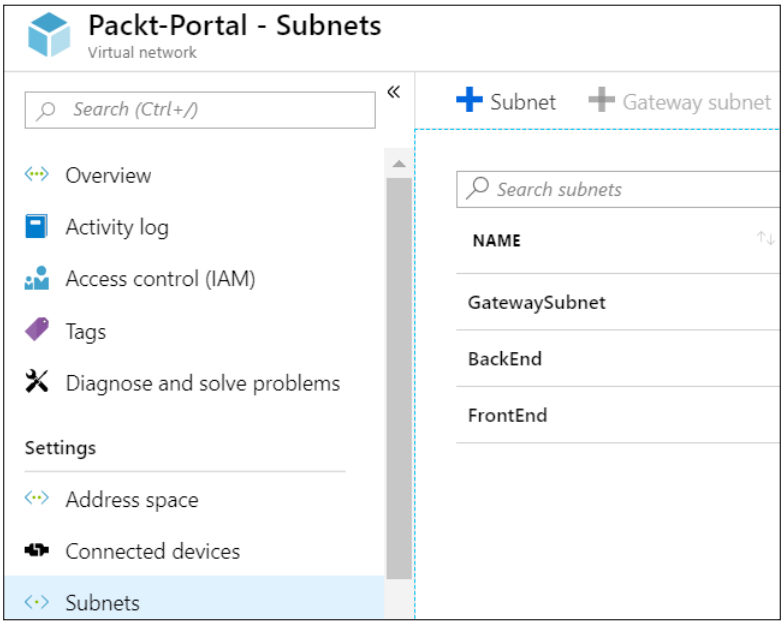
Azure Firewall allows us total control over our traffic. Besides controlling inbound traffic, with Azure Firewall, we can control outbound traffic as well.

Getting ready

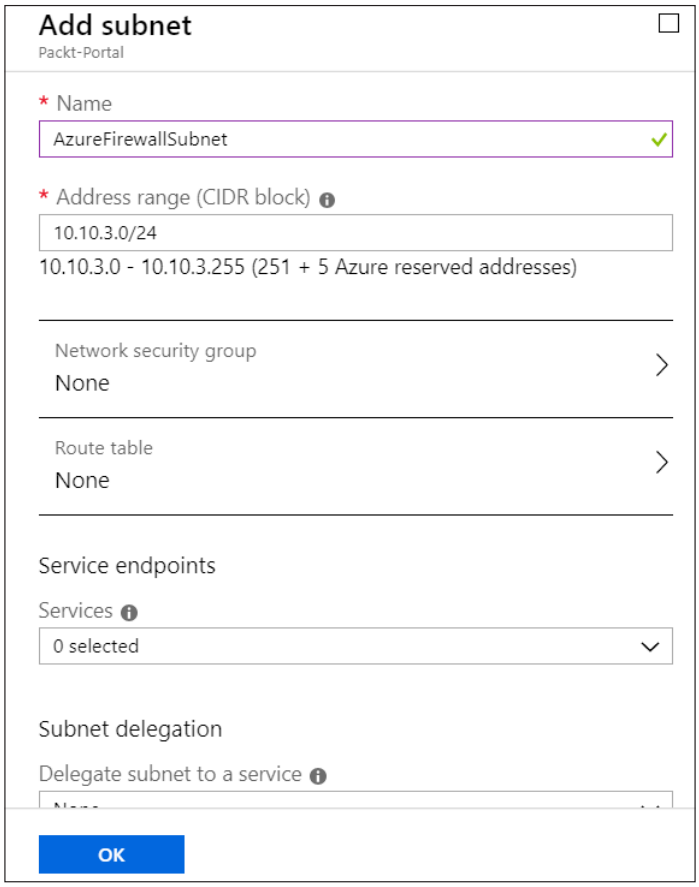
Before we can create an Azure Firewall instance, we must prepare the subnet.

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

1. Locate the virtual network that will be associated with our Azure Firewall.
2. Select the **Subnets** option under **Settings** and select the option to add a 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`:

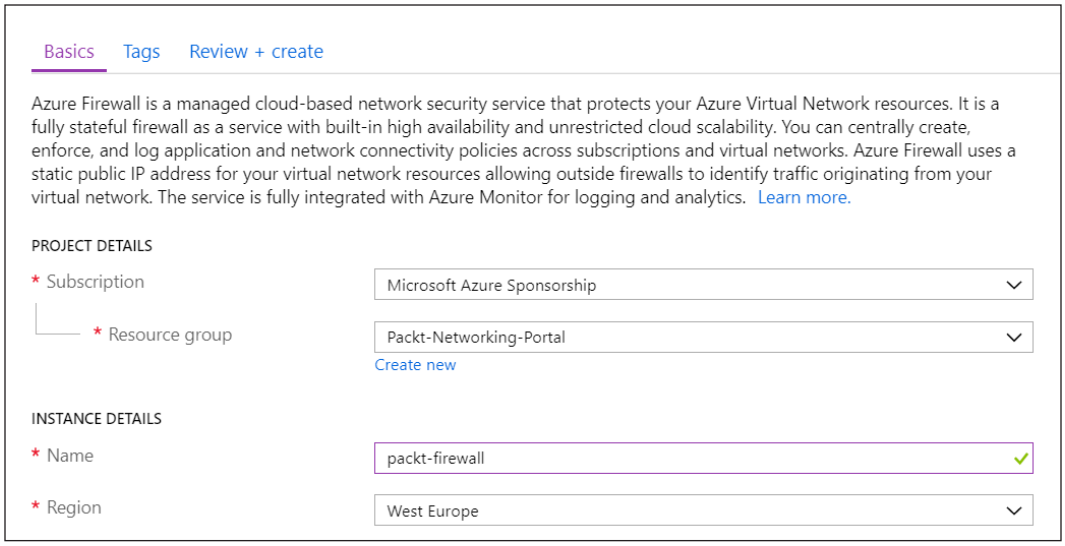


How to do it...

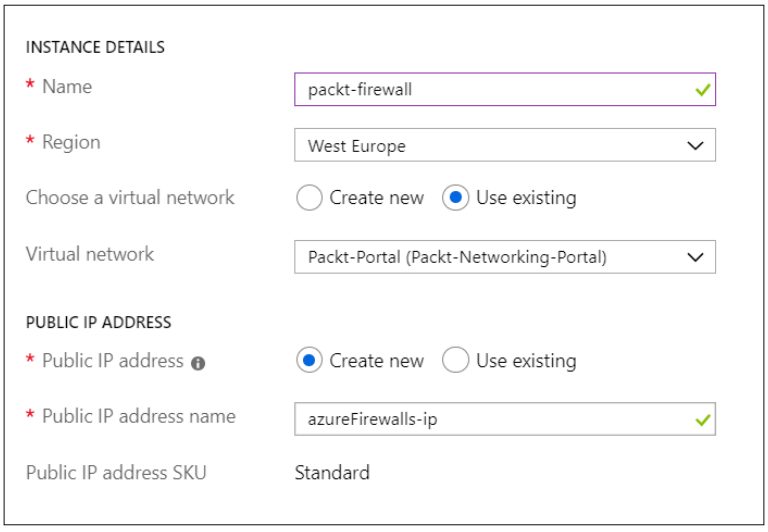
In order to create a new Azure Firewall instance using the Azure portal, take the following steps:

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 the **Subscription** and **Resource group** drop-down menus. We need to fill in the **Name** and **Region** fields for Azure Firewall, as shown in the following screenshot:



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



How it works...

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

Configuring a new allow rule

If we want to allow specific traffic, we must create an allow rule. Rules are applied based on priority level, so a rule will be applied only when there is no other 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 in effect.

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 whitelisted 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 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 the rules specified in the 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 part of any IT system, and networking is no exception. The diagnostics settings in Azure Firewall allow us to collect various information that can be used for troubleshooting or auditing.

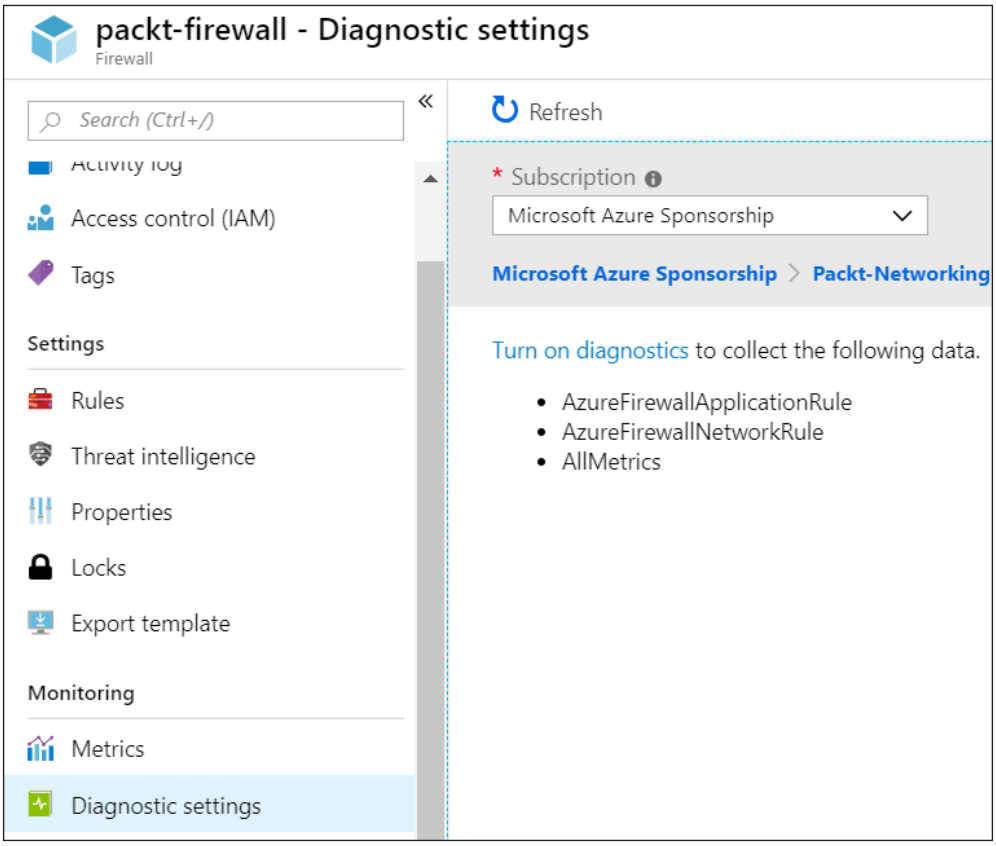
Getting ready

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

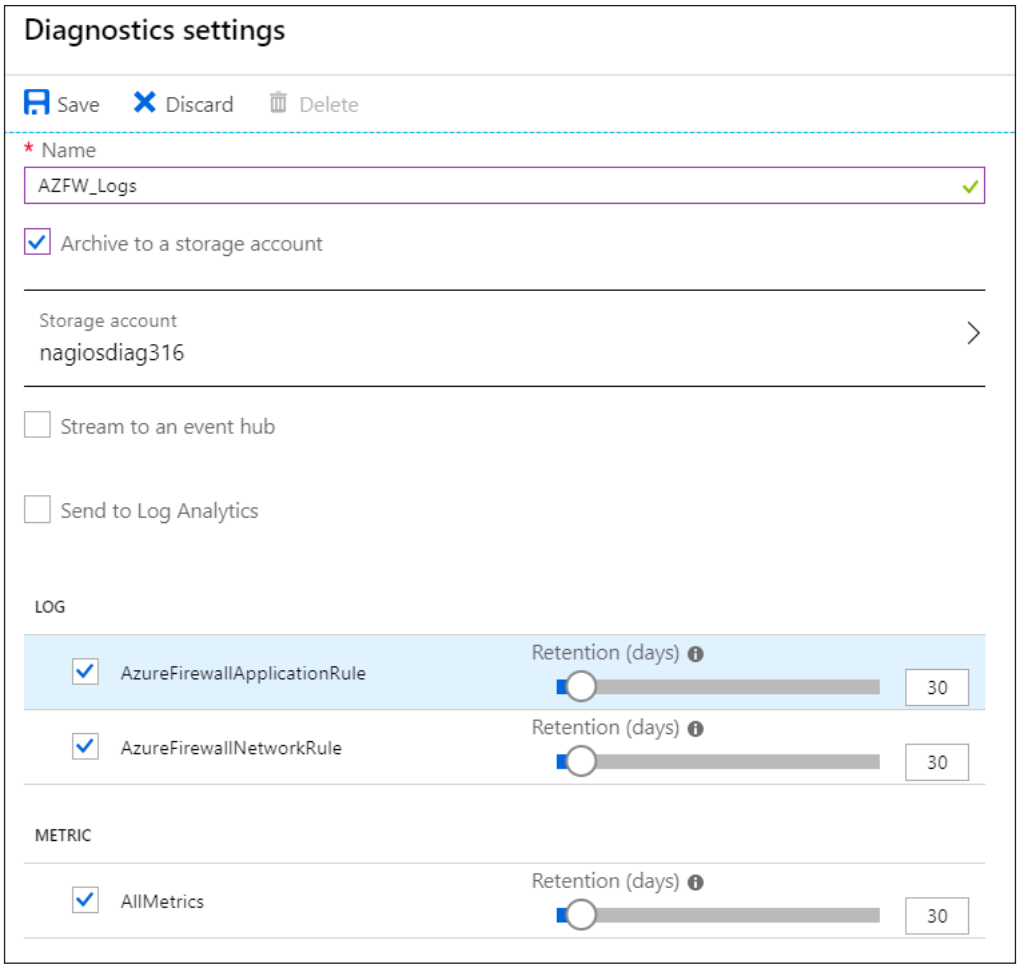
How to do it...

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

1. In the **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:

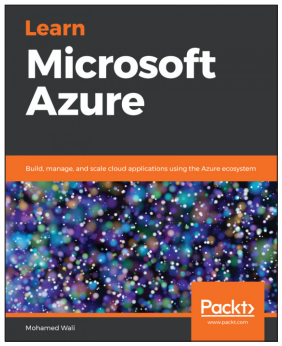


How it works...

Diagnostics has two purposes – auditing and troubleshooting. Based on traffic and settings, these logs can grow over time, so it's important to consider 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 seven 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:

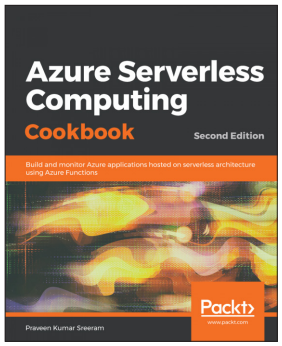


Learn Microsoft Azure

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

S

T

V

W