7

Managing Networking

In this chapter, we cover the following recipes:

* Configure IP Addressing
* Testing Network Connectivity
* Installing DHCP
* Configure DHCP Scopes and Options
* Using DHCP
* Implementing DHCP Fail-Over/Load Balancing
* Deploying DNS in the Enterprise
* Configuring DNS Forwarding
* Managing DNS Zones and DNS Resource Records

# Introduction

The heart of every organization is the network—the infrastructure that enables your client and server systems to inter-operate. Windows has included networking features since the early days of Windows for Workgroups 3.1 (and earlier with Microsoft LAN Manager).

Every server or workstation in your environment needs to have a correct IP configuration. While IPv6 is gaining in popularity, most organizations rely on IPV4. In the Configuring IP addressing recipe, we look at how to set a network interface's IPv4 configuration, including DNS settings.

Many organizations assign a static IPv4 address to most server systems. The servers used throughout this book, for example, make use of static IP addresses. For client hosts, and for some servers, an alternative to assigning a static IP address is to use Dynamic Host Control Protocol (DHCP). DHCP is a network management protocol that enables a workstation to lease an IP address (and release it when the lease expires). You set up a DHCP server to issue IP address configuration to clients using the “Installing and authorizing a DHCP”server recipe.

Once you have installed your DHCP server, you can use the “Configuring DHCP Scopes*”* recipe to set up the details that your DHCP server is to hand out to clients. In the “Configuring DHCP failover and load balancing” recipe, we deploy a second DHCP server and configure it to act as a failover/load balancing DHCP service.

In this chapter's final recipe, “Configuring DNS zones and resource records*”*, you configure the DNS server on DC1 with zones and additional resource records. Before you can begin to administer your Windows Server 2019 infrastructure, you need to create an environment where you can use PowerShell to carry out the administration.

# Configure IP Addressing

By default, Windows uses DHCP to configure all NICs that the Windows installation process discovers when installing Windows. Once you complete the installation of Windows, you can use the control panel applet (ncpa.cpl). the network shell console application (netsh.exe), or, of course, PowerShell to set IP configuration manually. In this recipe, you set the IP address details for SRV2 and ensure the host registers DNS names in the Reskit.Org DNS domain (on the DNS service running on DC1).

Setting up any host requires setting an IP address, a subnet mask and a default gateway which you do in the first part of this recipe. Then you configure SRV2 (a workgroup host), to register with the DNS Server on DC1.Reskit.Org. This approach raises some challenges. By default, when you created DC1.Reskit.Org as a DC, the domain's DNS zone is set to require secure updates. So, again, by default, a workgroup host can not register. You can overcome this by setting the zone to allow all updates. But this could be dangerous as it allows ANY host to, potentially, register their address. A second challenge is that since SRV2 is not a domain member, remoting to DC1 fails. A solution to that issue is to set the WinRM service to trust all hosts. Configuring WinRM to disregard server authentication has security implications you should consider before using this approach in production.

## Getting Ready

This recipe uses SRV2, a recently added workgroup host. By default, this host is a DHCP client.

## How to do it...

1. Discovering the network adapter, adapter interface and adapter interface index

$IPType    = 'IPv4'

$Adapter   = Get-NetAdapter |  Where-Object Status -eq 'Up'

$Interface = $Adapter | Get-NetIPInterface -AddressFamily $IPType

$Index     = $Interface.IfIndex

Get-NetIPAddress -InterfaceIndex $Index -AddressFamily $IPType |

  Format-Table -Property Interface\*, IPAddress, PrefixLength

1. Setting a new IP address for the NIC

$IPHT = @{

  InterfaceIndex = $Index

  PrefixLength   = 24

  IPAddress      = '10.10.10.51'

  DefaultGateway = '10.10.10.254'

  AddressFamily  = $IPType

}

New-NetIPAddress @IPHT

1. Verifying the new IP address

Get-NetIPAddress -InterfaceIndex $Index -AddressFamily $IPType |

  Format-Table IPAddress, InterfaceIndex, PrefixLength

1. Setting DNS Server IP address

$CAHT = @{

  InterfaceIndex  = $Index

  ServerAddresses = '10.10.10.10'

}

Set-DnsClientServerAddress @CAHT

1. Verifying the new IP configuration

Get-NetIPAddress -InterfaceIndex $Index -AddressFamily $IPType |

  Format-Table

1. Testing that SRV2 can see the domain controller

Test-NetConnection -ComputerName DC1.Reskit.Org |

  Format-Table

1. Creating a credential for DC1

$U    = 'Reskit\Administrator'

$PPT  = 'Pa$$w0rd'

$PSC  = ConvertTo-SecureString -String $ppt -AsPlainText -Force

$Cred = [pscredential]::new($U,$PSC)

1. Setting WinRM on SRV2 to trust s

$TPPATH = 'WSMan:\localhost\Client\TrustedHosts'

Set-Item -Path $TPPATH -Value 'DC1' -Force

Restart-Service -Name WinRM -Force

1. Enabling non-secure updates to Reskit.Org DNS domain

$DNSSSB = {

  $SBHT = @{

    Name          = 'Reskit.Org'

    DynamicUpdate = 'NonsecureAndSecure'

}

  Set-DnsServerPrimaryZone @SBHT

}

Invoke-Command -ComputerName DC1 -ScriptBlock $DNSSSB -Credential $Cred

1. Ensuring SRV2 registers within the Reskit.Org DNS zone

$DNSCHT = @{

  InterfaceIndex                 = $Index

  ConnectionSpecificSuffix       = 'Reskit.Org'

  RegisterThisConnectionsAddress = $true

  UseSuffixWhenRegistering       = $true

}

Set-DnsClient  @DNSCHT

1. Registering host IP address at DC1

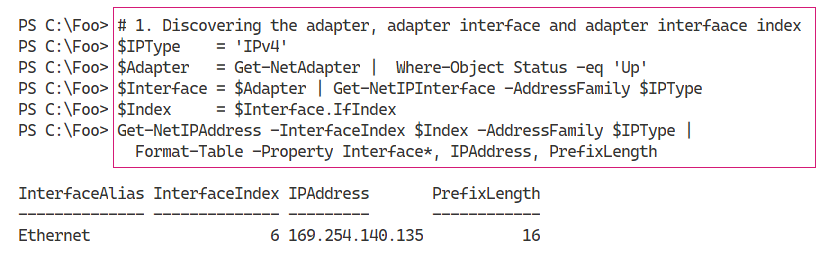
Register-DnsClient

1. Testing the DNS server on DC1.Reskit.Org resolves SRV2

Resolve-DnsName -Name SRV2.Reskit.Org -Type 'A' -Server DC1.Reskit.Org

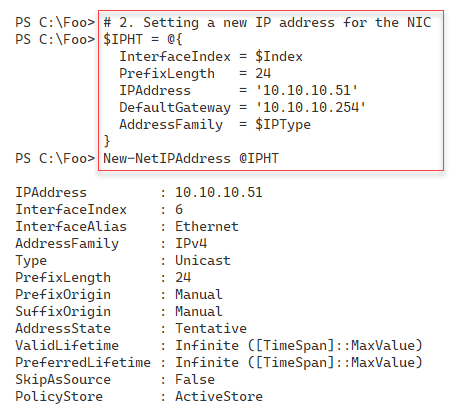
## How it works...

In step 1, you use Get-NetAdapter and Get-NetIPAddress to determine the IP address of this server. Then you display the resultant address, which looks like this:



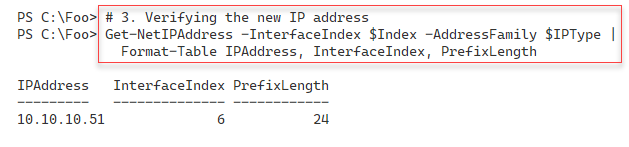
1. Insert image B1672\_01\_01.png

In step 2, you use the New-NetIPAddress cmdlet to set a static IP address on SRV2. The output looks like this:



1. Insert image B1672\_01\_02.png

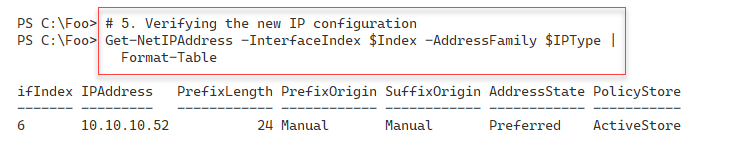
To double-check that you have configured SRV2 with the correct IP address configuration, you can use the Get-NetIPaddress cmdlet. The output looks like this:



1. Insert image B1672\_01\_03.png

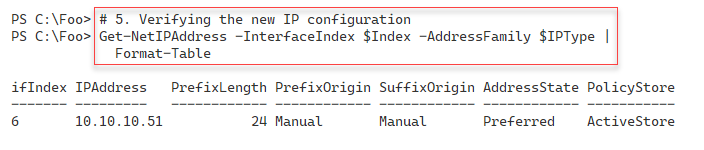
In addition to setting an IP address, subnet mask and default gateway, you also need to configure SRV2 with a DNS server address. In step 4, you use the Set‑DnsClientServerAddress cmdlet, which creates no output.

To check the updated IP configuration on SRV2, in step 5, you verify the configuration by (re)-using the Get-Get-NetIPAddress cmdlet, with output like this:



1. Insert image B1672\_01\_04.png

In step 6, you use Test-NetConnection to ensure SRV2 can connect to DC1, the domain controller in the Reskit.Org domain with this as the output:



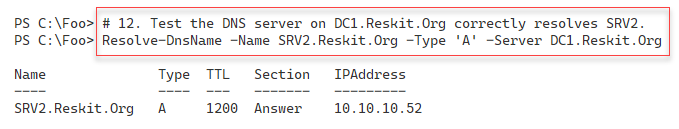
1. Insert image B1672\_01\_05.png

To enable SRV2, a workgroup computer to run commands on DC1, you need correct Windows credentials. In step 7, which creates no output, you create credentials for the Administrator user in Reskit.Org.

With step 8, you configure the WinRM service to trust DC1 explicitly. This step creates no output.

In step 9, you reconfigure the DNS Service on DC1 to enable non-secure updates to the Reskit.Org domain. In step 10, you configure SRV2 to register itself within the Reskit.Org zone on DC1. And then, in step 11, you register SRV2’s IP address within the DNS service on DC1. These three steps also produce no output.

In the final step, step 12, you query the DNS service to resolve the domain name SRV2.Reskit.Org. This step produces the following output:



1. Insert image B1672\_01\_06.png

In this recipe, you configured a workgroup server to have a static IP address. You also configure the DNS service to enable SRV2 to register a DNS record within the Reskit.Org domain. In most production scenarios, you would join SRV2 to the domain in which case DNS registration just works without needing step7 through step 11.

## There's more...

In step 5, you verify SRV2’s IP address. This test does not check SRV2’s DNS configuration. To check the DNS address as well, you could use Get-NetIPConfiguration.

In step 7, you create a credential to enable you to run commands on DC1. In this step, you use the enterprise/domain administrator account. In production, a better approach would be to create another user with a subset of the Enterprise Admin’s group's permissions then use that user to perform step 9.

In step 8, you configure WinRM to trust the DNS Server, DC1. This configuration is necessary for a workgroup environment because, by default, workgroup computers do not trust other servers when using PowerShell remoting (as you do in a later step). PowerShell remoting, by default, performs mutual authentication. Kerberos provides the necessary mutual authentication in a domain environment, while in a workgroup environment, you could use SSL to connect to DC1. By configuring SRV2 to trust DC1, you are disabling authentication of DC1 by SRV2. In a protected environment, like you have your set of Reskit.Org servers, this is acceptable. In production and especially in larger environments, a better approach is to enable SSL for remoting to hosts in separate security realms.

# Testing Network Connectivity

In today’s connected world, network connectivity is vital. When you add a new server to your infrastructure, it is useful to ensure that the server can connect to and use the network.

In this recipe, you perform necessary network connectivity tests on the newly installed SRV2 host. You should ensure that full connectivity exists before adding a server to the domain.

## Getting Ready

This recipe uses SRV2 a workgroup host. You gave this host a static IP address in “Configuring IP Addressing”.

## How to do it...

1. Verifying SRV2 itself is up, and that loopback is working

Test-Connection -ComputerName SRV2 -Count 1 -IPv4

1. Testing connection to local host's WinRM port

Test-NetConnection -ComputerName SRV2 -CommonTCPPort WinRM

1. Testing basic connectivity to DC1

Test-Connection -ComputerName DC1.Reskit.Org -Count 1

1. Checking  connectivity to SMB port on DC1

Test-NetConnection -ComputerName DC1.Reskit.Org -CommonTCPPort SMB

1. Check connectivity to the LDAP port on DC1

Test-NetConnection -ComputerName DC1.Reskit.Org -Port 389

1. Examining the path to a remote server on the Internet

$NCHT = @{

  ComputerName     = 'WWW.Packt.Com'

  TraceRoute       = $true

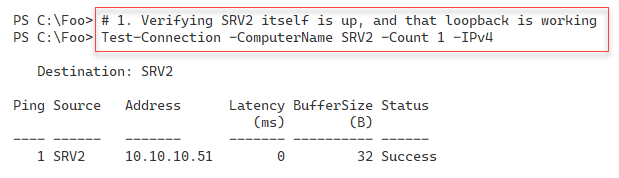
  InformationLevel = 'Detailed'

}

Test-NetConnection @NCHT    # Check our wonderful publisher

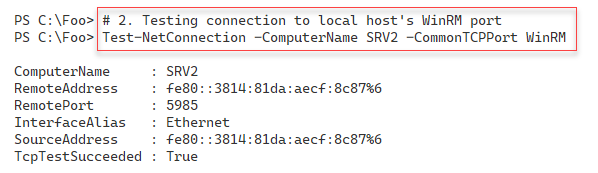
## How it works...

In step 1, you verify that SRV2’s Loopback adapter works and that the basic TCP/IP stack is up and working. The output looks like this:



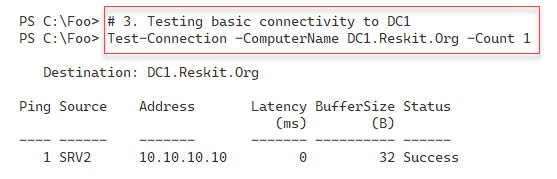
1. Insert image B1672\_01\_07.png

In step 2, you check to ensure that the WinRM port is open and working, with output like this:



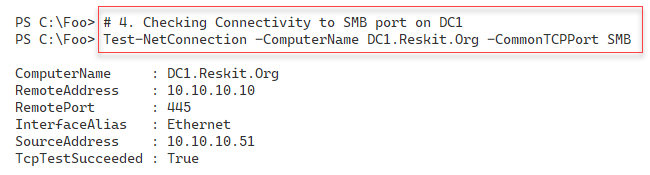
1. Insert image B1672\_01\_08.png

In the Reskit.Org network, DC1 is a domain controller and a DNS server. In step 3, you test the connectivity to this critical enterprise server, with output like this:



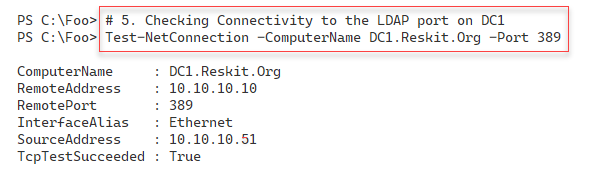
1. Insert image B1672\_01\_09.png

In any domain environment hosts need to access the Sysvol share on a DC to download group policy.POL files. In step 4, you test that SRV2 can access the SMB port, port 445, on the DC, with output like this:



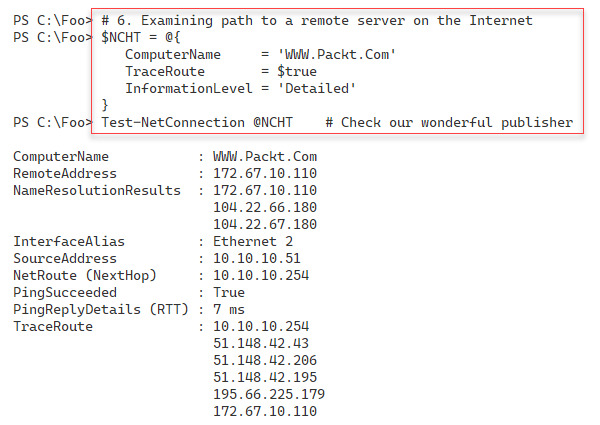
1. Insert image B1672\_01\_10.png

In step 5, you test that SRV2 can access DC1 on the LDAP port, port 389, with the following output:



1. Insert image B1672\_01\_11.png

In step 6, you check connectivity with the Internet and test the network path to the publisher’s web site at www.packt.com. The output is:



1. Insert image B1672\_01\_12.png

## There's more...

This recipe's steps confirm the host can accept connections over WinRM and can contact the DC for core activities. You could add several additional tests, such as testing you can access the DNS server and resolve DNS queries.

In step 6, you test the Internet connectivity to our publisher’s web site (www.packt.com). Since we are just testing the connectivity to this site, you only need to specify the actual computer name and do not need the HTTP or HTTPS prefix.

# Installing DHCP

In previous recipes, you configured SRV2 with a static IP address and tested its connectivity. Each server needs a unique IP address and other configuration options which you can configure on a server by server basis. You can also configure client computers running Windows 10 or other OSs manually as well although in large organisations this can be a huge and challenging task.

Dynamic Host Configuration Protocol (DHCP) is a solution to enable a DHCP client to obtain it’s IP configuration and other networking details automagically from a DHCP server. DHCP automates IP configuration and avoids the work and avoids the inevitable issues involved with manual IP configuration.

Windows and most other client operating systems, including Linux and Apple Macs, have a built-in DHCP client. Windows Server also includes a DHCP Server service you can install to provide DHCP services to the clients. You can install DHCP using Server Manager and configure the service using the DHCP GUI application. Alternatively, you can automate the installation of DHCP as you can see in this recipe. In the next recipe, “Configure DHCP Scopes and Options, you configure the DHCP service to issue IP addresses in a specific range. You also configure DHCP to provide DHCP clients with other IP address configuration options, such as the subnet mask, default gateway, and the DNS server IP Address(es).

## Getting Ready

This recipe uses DC1, a domain controller in the Reskit.Org domain. You should have installed AD on this host and configured it as per earlier recipes in Chapter 5 and Chapter 6.

## How to do it...

1. Install the DHCP Feature on DC1 and add the Management tools

Import-Module -Name ServerManager -WarningAction SilentlyContinue

Install-WindowsFeature -Name DHCP -IncludeManagementTools

1. Adding DC1 to trusted DHCP Servers and add the DHCP Security Group

Import-Module -Name DHCPServer -WarningAction SilentlyContinue

Add-DhcpServerInDC

Add-DHCPServerSecurityGroup

1. Letting DHCP know it's all configured

$DHCPHT = @{

Path = 'HKLM:\SOFTWARE\Microsoft\ServerManager\Roles\12'

Name = 'ConfigurationState'

Value = 2

}

Set-ItemProperty @DHCPHT

1. Restarting DHCP Server

Restart-Service -Name DHCPServer –Force

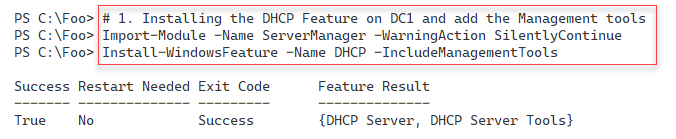
1. Testing service availability

Get-Service -Name DHCPServer |

Format-List -Property \*

## How it works...

In step 1, you import the ServerManager module and use Install-WindowsFeature to add the DHCP server service to DC1. The output from this step looks like this:



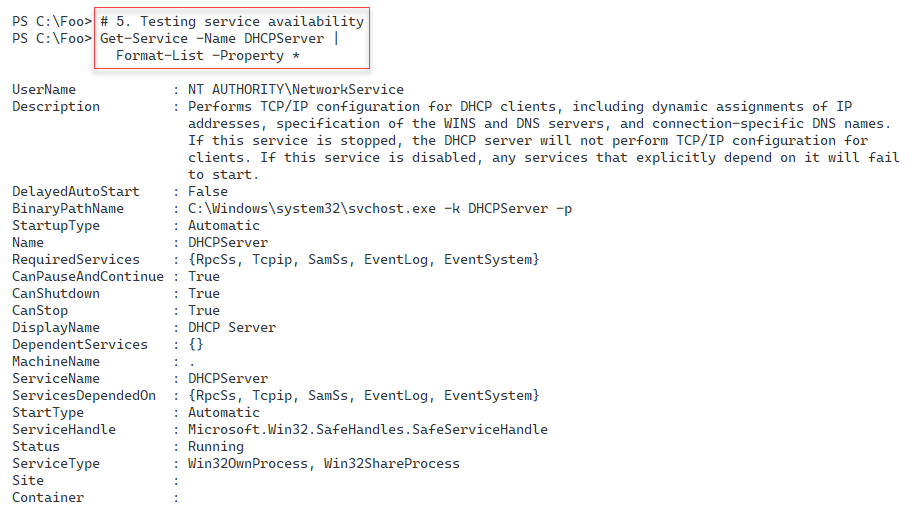
1. Insert image B1672\_01\_13.png

In step 2, you add DC1 to the set of authorized DHCP servers in the domain and add the DHCP security groups to the DCHP server. The groups that this command adds are DHCP Users and DHCP Administrators security groups. For more details on these groups, see https://secureidentity.se/delegate-dhcp-admins-in-the-domain/.

In step 3, you set a registry entry to tell Windows that all post-deployment DHCP configuration activities are complete. The GUI installation process takes you through this automatically. When installing via PowerShell, you need to set the registry entry to complete the configuration

With you have completed the configuration activities, you restart the DHCP service. Once restarted, the DHCP service can issue IP configuration to DHCP clients. For this to happen, you must also have specified the configuration information you specify in the “Configure DHCP Scopes and Options” recipe. Step 2, step 3. and step 4 produce no output.

In step 5, you complete this recipe by ensuring that the DHCP service has started. The output of this step looks like this:



1. Insert image B1672\_01\_14.png

## There's more...

When the Windows DHCP service starts, it checks to ensure the server is on the DHCP server list authorized in the domain. If it is not authorized, the DHCP service does not start. By adding DC1 to the list of authorized servers can help to guard against rogue DHCP Servers.

In step 5, you check the DHCP Service. Get-Service's output details the service, including a description and the pathname to the actual service executable. The DHCP service does not run in its own process as such. Instead, it runs inside svchost.exe. It is for this reason that you do not see the service explicitly when you use Get-Process.

# Configure DHCP Scopes and Options

Installing DHCP is simple, as you see in “Installing DHCP”. You add the Windows feature and then carry out two small configuration steps. Those extra steps are, strictly speaking, not necessary. The extra steps enable you to use the relevant security groups and avoid the Server Manager GUI message that there are configuration steps not yet performed.

Before your DHCP server can provide IP address configuration information to DHCP clients, you need to create a DHCP scope and DHCP options. A DHCP scope is a range of DHCP addresses that your DHCP server can give out for a given IP subnet. DHCP options are specific configuration options your DHCP server provides, such as the DNS server's IP address and the IPv4 default gateway.

You can set DHCP options a scope level or at a server level, depending on your organisation's needs. For example, you would most likely specify a default gateway in the Scope options, with DNS server address(es) set at the server level.

In this recipe, you create a new scope for the 10.10.10.0/24 subnet and specify both scopes and server level

## Getting Ready

You run this recipe on DC1, a domain controller in the Reskit.Org domain after installing the DHCP server service. You must have installed PowerShell 7 and VS Code on this host.

## How to do it...

1. Importing the DHCP Server module

Import-Module DHCPServer -WarningAction SilentlyContinue

1. Creating an IPv4 scope

$SCOPEHT = @{

Name = 'ReskitOrg'

StartRange = '10.10.10.150'

EndRange = '10.10.10.199'

SubnetMask = '255.255.255.0'

ComputerName = 'DC1.Reskit.Org'

}

Add-DhcpServerV4Scope @SCOPEHT

1. Getting IPV4 Scopes from the server

Get-DhcpServerv4Scope -ComputerName DC1.Reskit.Org

1. Setting server-wide option values

$OPTION1HT = @{

ComputerName = 'DC1.Reskit.Org' # DHCP Server to Configure

DnsDomain = 'Reskit.Org' # Client DNS Domain

DnsServer = '10.10.10.10' # Client DNS Server

}

Set-DhcpServerV4OptionValue @OPTION1HT

1. Setting a scope specific option

$OPTION2HT = @{

ComputerName = 'DC1.Reskit.Org' # DHCP Server to Configure

Router = '10.10.10.254'

ScopeID = '10.10.10.0'

}

Set-DhcpServerV4OptionValue @OPTION2HT

1. Viewing DHCP server options

Get-DhcpServerv4OptionValue | Format-Table -AutoSize

1. Viewing scope specific options

Get-DhcpServerv4OptionValue -ScopeId '10.10.10.10' |

Format-Table -AutoSize

1. Viewing DHCPv4 option definitions

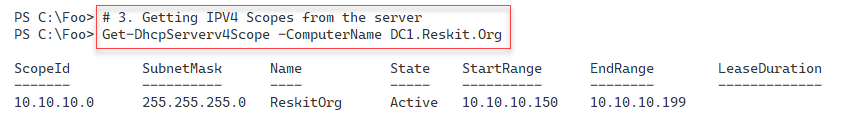
Get-DhcpServerv4OptionDefinition | Format-Table -AutoSize

## How it works...

In step 1, you import the DHCPServer module. When you installed DHCP (in “Installing DHCP”), you added the management tools, including this module. However, the DHCP team have not yet made this module compatible with PowerShell 7. This step, which produces no output, loads the module using the Windows PowerShell compatibility solution.

In step 2, you create a new DHCP scope for IPV4 addresses. The scope enables the DHCP server to issue IP addresses in the range of 10.10.10.150 - 10.10.10.199 range. This step produces no output.

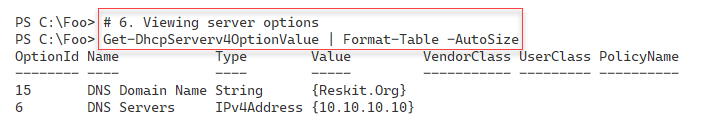
In step 3, you use Get-DHCPServerIPV4Scope to retrieve details of all the DHCP scopes you have defined on DC1. The output of this step looks like this:



1. Insert image B1672\_01\_15.png

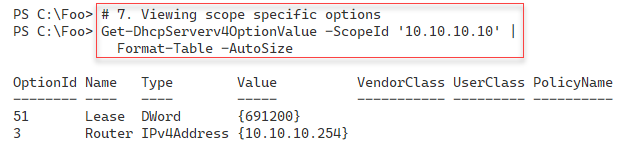
In step 4, you set two server-wide options, creating no output. These are options and values offered to all clients of any DHCP scope defined on this server. In step 5, you specify a scope option. This step also produces no output.

In step 6, you view the DHCP Server-wide options, with output looks like this;



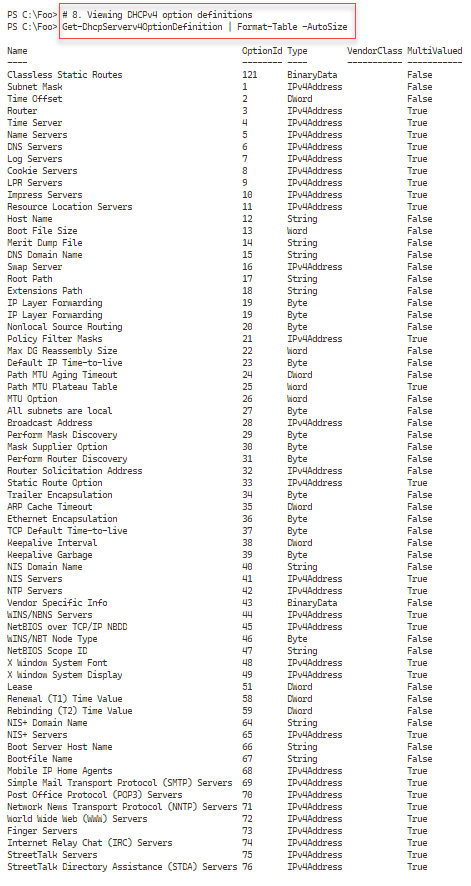
1. Insert image B1672\_01\_16.png

With step 7, you view the options you have set on the 10.10.10.10 scope, which looks like this:



1. Insert image B1672\_01\_17.png

There are 66 DHCP options you can use to provide option values to DHCP Clients. Most of these options are of little use in most cases but provide support for niche and uncommon scenarios. To view the set of options defined by default, in step 8, which looks like this:



1. Insert image B1672\_01\_18.png

## There's more...

In step 2, you create a new scope and give it a scope name. However, as you can see in step 5 and elsewhere, the DHCP cmdlets do not provide a -DHCPScopeName parameter. Instead, you specify a “ScopeID”. In general, this is the subnet for the IP addresses in the scope, 10.10.10.0/24. But even then, as you can see in step 8, the cmdlet accepts any IP address in that subnet as the subnet ID, including 10.10.10.10 as shown.

# Using DHCP

After installing the DHCP service and configuring scope(s) and option values, your DHCP services can issue IP configuration data to any client. Since the DHCP protocol acts at the IP level, the protocol performs no authentication when any DHCP client uses the protocol to request IP configuration details. That means that any client you attach to the physical subnet can ask for and receive IP confirmation details.

In “Configure IP Addressing”, you set a static IP address for SRV2. In this recipe, you reconfigure this server to obtain a DHCP based IP address (and options you set in “Configure DHCP Scopes and Options”.

## Getting Ready

You run this recipe on SRV2 which you reconfigure to get its address via DHCP. You also need DC1, a domain controller for the reskit.org domain and a DHCP server which you set up and configured in earlier recipes in this chapter.

## How to do it...

1. Adding DHCP RSAT Tools

Import-Module -Name ServerManager -WarningAction SilentlyContinue

Install-WindowsFeature -Name RSAT-DHCP

1. Importing the DHCP module

Import-Module -Name DHCPServer -WarningAction SilentlyContinue

1. Viewing the scopes on DC1

Get-DhcpServerv4Scope -ComputerName DC1

1. Getting V4 scope statistics from DC1

Get-DhcpServerv4ScopeStatistics -ComputerName DC1

1. Discovering a free IP address

Get-DhcpServerv4FreeIPAddress -ComputerName dc1 -ScopeId 10.10.10.42

1. Getting SRV2 NIC Configuration

$NIC = Get-NetIPConfiguration -InterfaceIndex 6

1. Getting IP Interface details

$NIC |

  Get-NetIPInterface  |

    Where-Object AddressFamily -eq 'IPv4'

1. Enabling DHCP on the NIC

$NIC |

  Get-NetIPInterface  |

    Where-Object AddressFamily -eq 'IPv4' |

      Set-NetIPInterface -Dhcp Enabled

1. Checking IP address assigned

Get-NetIPAddress -InterfaceAlias "Ethernet"     |

  Where-Object AddressFamily -eq 'IPv4'

1. Getting updated V4 scope statistics from DC1

Get-DhcpServerv4ScopeStatistics -ComputerName DC1

1. Discovering the next free IP address

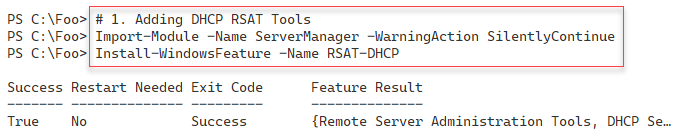
Get-DhcpServerv4FreeIPAddress -ComputerName dc1 -ScopeId 10.10.10.42

1. Checking IPv4 DNS Name Resolution

Resolve-DnsName -Name SRV2.reskit.org -Type A

## How it works...

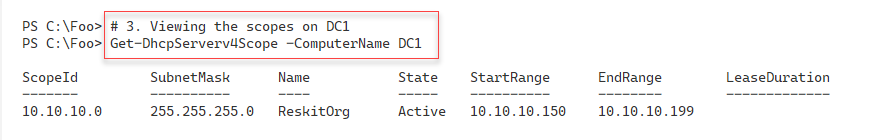
In step 1, you install the RSAT-DHCP feature to add the DHCP Server’s PowerShell module on SRV1, with output like this:



1. Insert image B1672\_01\_19.png

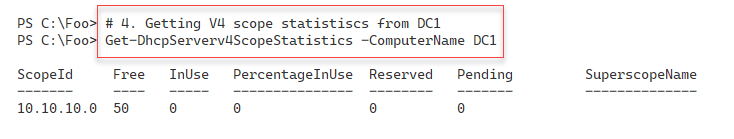
The DHCP Server module is not .NET Core compatible. In step 2, you explicitly load this module using Import-Module, which creates no output.

In step 3, you look at the scopes available on DC1 (the DHCP server you installed in “Installing DHCP”). The output looks like this;



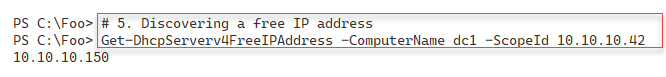
1. Insert image B1672\_01\_20.png

In step 4, you examine the scope statistics for the DHCP scope you created in “Configure DHCP Scopes and Options” which produces output like this:



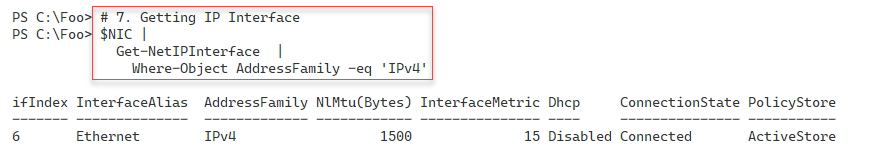
1. Insert image B1672\_01\_21.png

In step 5, you use the Get-DhcpServerv4FreeIPAddress cmdlet to find the first available free IP address in the scope. The output resembles this:



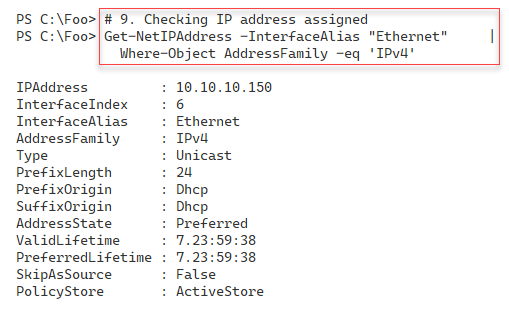
1. Insert image B1672\_01\_22.png

In step 6, you get the NIC details and store it in the $NIC variable producing no output. In step 7, you use that variable to get details of the NIC, with output like this:



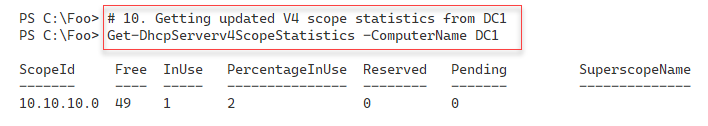
1. Insert image B1672\_01\_23.png

In step 8, you change the NIC to get configuration details from the DHCP server. This step creates no output. In step 9, you view the NIC’s IPV4 address, this time one assigned by DHCP. The output looks like this:



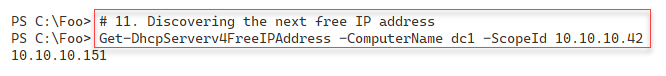
1. Insert image B1672\_01\_24.png

In step 10, you re-examine the scope statistics, with output like this:



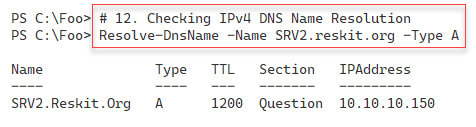
1. Insert image B1672\_01\_25.png

With step 11, you re-check to discover the next free IP address in the DHCP scope, with output like this:



1. Insert image B1672\_01\_26.png

In the final step, you check to ensure that SRV2 has registered its new IP address in the DNS Server on DC1. The output looks like:



1. Insert image B1672\_01\_27.png

## There's more...

In this recipe, you use the DHCP server cmdlets to get information from the DHCP server on DC1. These cmdlets show how you can obtain information from the DHCP server, including the next free IP address and statistics on the scope (free/used addresses etc.).

In step 6, you et the IP Interface details to allow you, in step 8, to convert the NIC from a static IP address to a dynamic one, based on DHCP.

In step 9, you view the DHCP supplied IP address information for SRV2. If you perform step 8 and immediately run step 9, you may find that the NIC shows an Automatic Private IP Addressing (APIPA) IP address in the 169.254.0.0/24 subnet This address is transient. When you change to DHCP (as you did on step 8), Windows removes the static address and creates an APIPA address. Once SRV2 contacts the DHCP server and negotiates an address, you see the output shown for step 9. Obtaining a lease can take a few seconds, so be patient.

# Implementing DHCP Fail-Over/Load Balancing

As shown in the two previous two recipes, the installation and configuration of a single DHCP server is straightforward. However, a single DHCP server represents a single point of failure, which is never a good thing. The solution is always to have a second DHCP server. In earlier versions of Windows, you could do this with two DHCP servers each with a scope. Typically you used to split the full set of IP addresses and allow each server to have part of that set. The traditional ‘wisdom’ was to do an 80/20 split (have 80% of the scope supplied by your primary DHCP server and 20% on the backup server).

Independent DHCP servers are an error-prone approach and were never ideal since these independent servers did not co-ordinate scope details. And that 80/20 “rule” was a recommendation for one specific customer scenario (a large firm in the Pacific Northwest), and possibly was not meant to become a best practice.

In Windows Server 2012, Microsoft added a DHCP failover and load balancing mechanism that simplified deploying DHCP in an organization. You can now set up two DHCP servers, define a DHCP scope, and allow both servers to work in tandem.

In this recipe, you install DHCP on a second server, DC2 and then configure and use the failover and load balancing capabilities of Windows Server.

## Getting Ready

This recipe uses the two DCs you have installed, DC1 and DC2. You should also have installed DHCP on DNS (“Installing DCHP Server”) and configured a DNS zone (“Configure DHCP Scopes and Options”). You run this recipe on DC2.

## How to do it...

1. Installing the DHCP Server feature on DC2

Import-Module -Name ServerManager -WarningAction SilentlyContinue

$FEATUREHT = @{

  Name                   = 'DHCP'

  IncludeManagementTools = $True

}

Install-WindowsFeature @FEATUREHT

1. Letting DHCP know it is fully configured

$IPHT = @{

  Path  = 'HKLM:\SOFTWARE\Microsoft\ServerManager\Roles\12'

  Name  = 'ConfigurationState'

  Value = 2

}

Set-ItemProperty @IPHT

1. Authorizing the DHCP server in AD

Import-Module -Name DHCPServer -WarningAction 'SilentlyContinue'

Add-DhcpServerInDC -DnsName DC2.Reskit.Org

1. Viewing Authorized DHCP servers in the Reskit domain

Get-DhcpServerInDC

1. Configure failover and load balancing

$FAILOVERHT = @{

  ComputerName       = 'DC1.Reskit.Org'

  PartnerServer      = 'DC2.Reskit.Org'

  Name               = 'DC1-DC2'

  ScopeID            = '10.10.10.0'

  LoadBalancePercent = 60

  SharedSecret       = 'j3RryIsTheB3est!'

  Force              = $true

  Verbose            = $True

}

Invoke-Command -ComputerName DC1.Reskit.org -ScriptBlock {

  Add-DhcpServerv4Failover @Using:FAILOVERHT

}

1. Getting active leases in the scope (from both servers!)

$DHCPServers = 'DC1.Reskit.Org', 'DC2.Reskit.Org'

$DHCPServers |

  ForEach-Object {

    "Server $\_" | Format-Table

    Get-DhcpServerv4Scope -ComputerName $\_ | Format-Table

  }

1. Viewing DHCP Server Statistics from both DHCP Servers

$DHCPServers |

  ForEach-Object {

    "Server $\_" | Format-Table

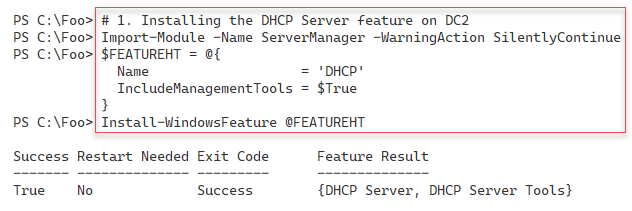
    Get-DhcpServerv4ScopeStatistics -ComputerName $\_  | Format-Table

  }

## How it works...

Using the first three steps in this recipe, you install the DHCP Server feature on DC2. These steps duplicate the approach you used in “Installing DHCP” to install DHCP on DC1.

In step 1, you import the Server Manager module and then install the DHCP feature, with output like this:



1. Insert image B1672\_01\_28.png

In step 2, you set a registry key to tell Windows that you have completed the DHCP feature installation. Without this step, any time you use the Server Manager GUI on DC2, you would see a message indicating that additional configuration was required. This step produces no output

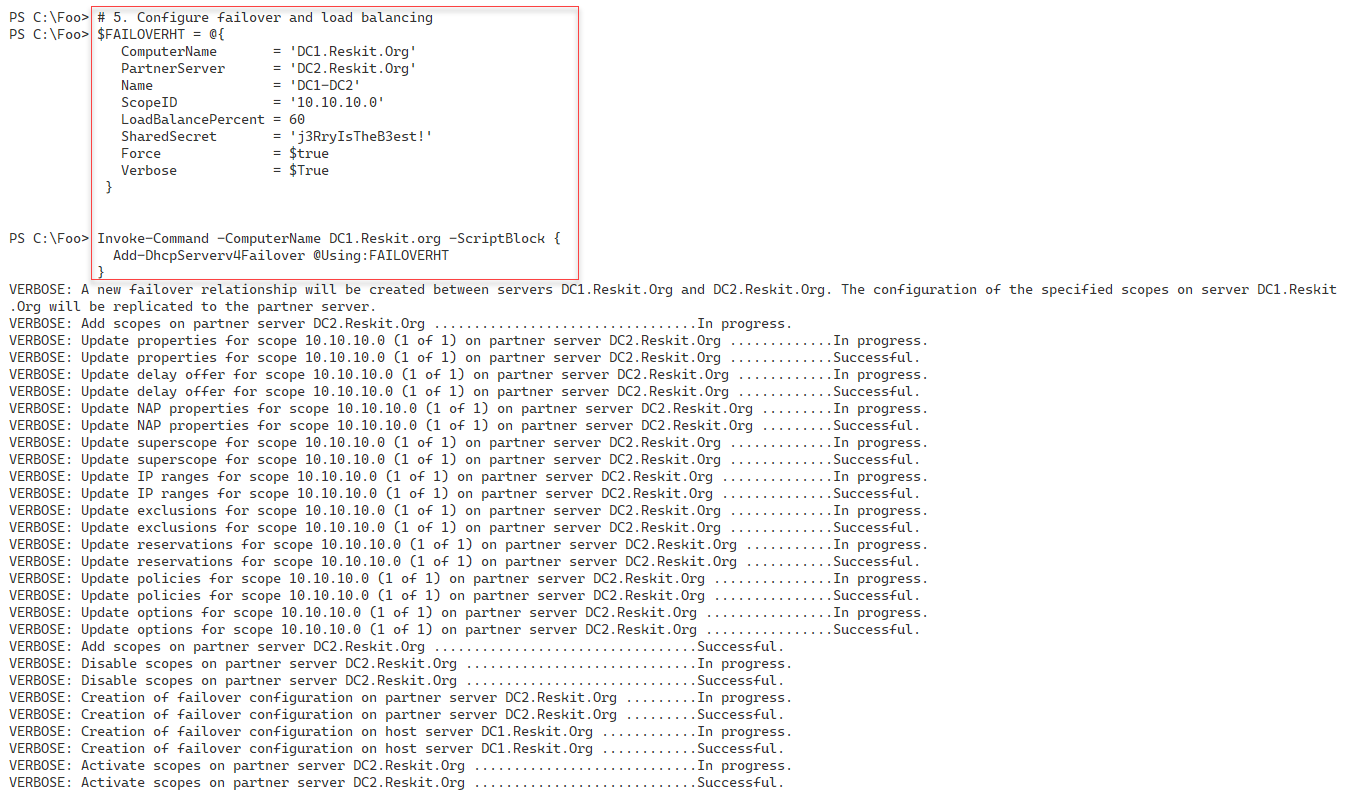
In step 3, you import the DHCP Server PowerShell module and add DC2 to the list of authorized DHCP servers in the Reskit domain. This step produces no output.

In step 4, you examine the set of authorized DHCP servers, which produces the following output:



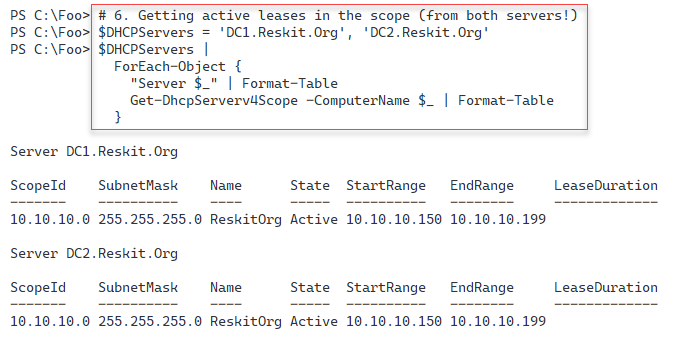
1. Insert image B1672\_01\_29.png

In step 5, you configure DHCP load balancing and failover. By using the -Verbose switch



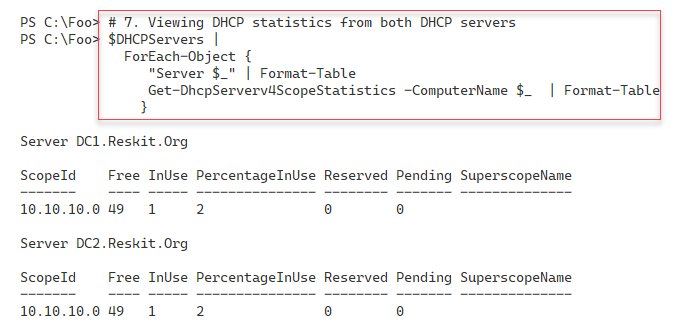
1. Insert image B1672\_01\_30.png

With step 6, you examine the scopes on both DHCP servers, with the following output:



1. Insert image B1672\_01\_31.png

In step 7, you view the DHCP server statistics from your two DHCP servers, with output like this;



1. Insert image B1672\_01\_32.png

## There's more...

Unlike in the “Installing DHCP”, in this recipe, you do not add the DHCP related local security groups on DC2. If you plan to delegate administrative privileges, you may wish to add those groups as you did in the earlier recipe.

In step 5, you establish a load balancing and failover relationship between the two DHCP Servers. By using the -Verbose switch to the Add-DhcpServerV4Failover cmdlet, you can see precisely what the command is doing, step by step. As you can see in the output, this command copies full details of all the scopes on DC1 to DC2. You should note, that the relationship is on a scope by scope basis.

Depending on your needs, you can configure different sorts of relationships between DHCP servers using the -ServerRole parameter. For more details on this parameter and others, you can use to fine-tune the relationship between the two partners, see https://docs.microsoft.com/powershell/module/dhcpserver/add-dhcpserverv4failover.

The DHCP Failover feature provides several additional settings to control precisely when to fail over and for how long (and when to failback). These settings allow you to control, for example, what might happen during a planned reboot of your DHCP server.

# Deploying DNS in the Enterprise

When you installed Active Directory, in chapter 6, you added a single DNS server on DC1. When you added a replica DC, DC2, you did not set up a second DNS Server on that DC. In an enterprise organization, this is not best practice. You always want to configure your clients and servers to use both DNS servers. You should update any static DNS settings and ensure that the DHCP servers also provide two DNS server entries.

In most organizations, there are several DNS service configuration options you may wish to set. These include whether to allow DNS server recursion on the server, the maximum size of the DNS cache, and whether to use Extended DNS (EDNS)

EDNS (also referred to as EDNS0 or more recently EDNS(0)), is an extension mechanism that enables more recent DNS servers to interact with older servers that may not be capable of specific actions. For more details on EDNS(0), see https://en.wikipedia.org/wiki/Extension\_mechanisms\_for\_DNS#:~:text=Extension%20mechanisms%20for%20DNS%20(EDNS,increasing%20functionality%20of%20the%20protocol.

This recipe is a starting point. There are other DNS server options you may wish to consider updating, such as DNS Security. Additionally, you may need to reconfigure other servers and update the static IP address settings (to point to both DNS Servers). And finally, in the Reskit.Org forest, you should also be updating the child domain UK.Reskit.Org.

In this recipe, you update the domain-wide DNS settings in the Reskit.Org domain. These settings include ensuring that you configure DC2’s DNS service the same as DC1, updating DNS server configuration settings, ensuring that both DC1 and DC2 point to the correct DNS servers and update DHCP issue both DNS server addresses. You complete the recipe by examining the settings.

## Getting Ready

You run this recipe on DC2, with DC1 also operational. You run this recipe after setting up DNS on both DC1 and DC2 and implementing DHCP.

## How to do it...

1. Installing the DNS feature on DC2

Import-Module -Name ServerManager -WarningAction SilentlyContinue

Install-WindowsFeature -Name DNS -IncludeManagementTools

1. Creating a script block to set DNS server options

$SB1 = {

  # Enable recursion on this server

  Set-DnsServerRecursion -Enable $true

  # Configure DNS Server cache maximum size

  Set-DnsServerCache  -MaxKBSize 20480  # 28 MB

  # Enable EDNS

  $EDNSHT = @{

    EnableProbes    = $true

    EnableReception = $true

  }

  Set-DnsServerEDns @EDNSHT

  # Enable Global Name Zone

  Set-DnsServerGlobalNameZone -Enable $true

}

1. Reconfiguring DNS on DC2, DC1

Invoke-Command -ScriptBlock $SB1

Invoke-Command -ScriptBlock $SB1 -ComputerName DC1

1. Creating script block to configure DC2 to have two DNS servers

$SB2 = {

  $NIC =

    Get-NetIPInterface -InterfaceAlias "Ethernet" -AddressFamily IPv4

  $DNSSERVERS = ('127.0.0.1','10.10.10.10')

  $DNSHT = @{

    InterfaceIndex  = $NIC.InterfaceIndex

    ServerAddresses = $DNSSERVERS

  }

  Set-DnsClientServerAddress @DNSHT

  Start-Service -Name DNS

}

1. Configuring DC2 to have two DNS Servers

Invoke-Command -ScriptBlock $SB2

1. Creating script block to configure DC1 to have two DNS servers

$SB3 = {

  $NIC =

    Get-NetIPInterface -InterfaceAlias "Ethernet" -AddressFamily IPv4

  $DNSSERVERS = ('127.0.0.1','10.10.10.11')

  $DNSHT = @{

    InterfaceIndex  = $NIC.InterfaceIndex

    ServerAddresses = $DNSSERVERS

  }

  Set-DnsClientServerAddress @DNSHT

  Start-Service -Name DNS

}

1. Configuring DC2 to have two DNS Servers

Invoke-Command -ScriptBlock $SB3 -ComputerName DC1

1. Updating DHCP scope to add 2 DNS entries

$DNSOPTIONHT = @{

  DnsServer    = '10.10.10.11',

                 '10.10.10.10'    # Client DNS Servers

  DnsDomain    = 'Reskit.Org'

  Force        = $true

}

Set-DhcpServerV4OptionValue @DNSOPTIONHT -ComputerName DC1

Set-DhcpServerV4OptionValue @DNSOPTIONHT -ComputerName DC2

1. Getting DNS Service details

$DNSRV = Get-DNSServer -ComputerName DC2.Reskit.Org

1. View recursion settings

$DNSRV |

  Select-Object -ExpandProperty ServerRecursion

1. View server cache settings

$DNSRV |

  Select-Object -ExpandProperty ServerCache

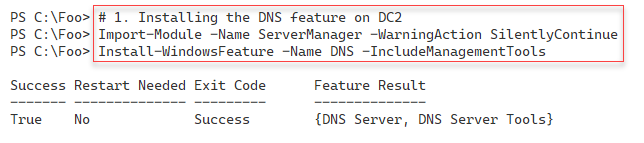
1. View EDNS settings

$DNSRV |

  Select-Object -ExpandProperty ServerEdns

## How it works...

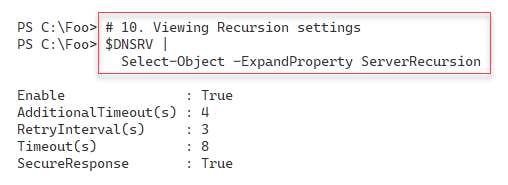
In step 1, you install the DNS feature on DC1, with output like this:



1. Insert image B1672\_01\_33.png

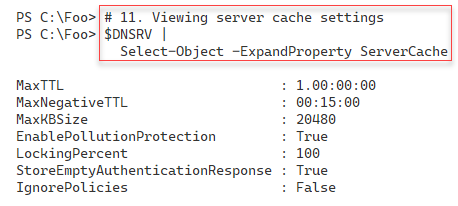
In step 2, you create a script block that contains the DNS server options. In step 3, you run this script block on both DC1 and DC2. In step 4, you create a script block to use to set the IP configuration on DC2. In step 5, you run this script block to reconfigure and DC2. In step 6 and step 7, you reconfigure the static IP setting on DC1. In step 8, you update the DHCP server DHCP options to ensure the DHCP server on both hosts supplies two DNS server IP addresses to clients of either DHCP server. These seven steps produce no output.

In step 9, you get the DNS Server details from DC2, producing no output. In step 10, you examine the DNS server recursion settings, with output like this:



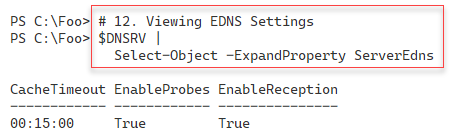
1. Insert image B1672\_01\_34.png

In step 11, you examine the DNS server cache settings on DC2, with output like this.



1. Insert image B1672\_01\_35.png

In step 12, you view the EDNS settings on DC2, with output like this



1. Insert image B1672\_01\_36.png

## There's more...

In this recipe, you install and configure DNS on DC2. You also set DNS server options on both DNS Servers (DC1, and DC2). In step 5 and step 7, you configure your domain controllers to point to themselves first and then to point to the other DNS servers second. Configuring DNS in this manner is considered a best practice.

In the earlier recipe, you configured your DHCP servers to provide a single DNS server IP address to DHCP clients. In step 8, you update DHCP to provide two DNS server IP addresses.

You complete the recipe by obtaining and then viewing the DNS server recursion, server cache, and EDNS settings on DC2. You could extend these steps by examining the settings on DC2 and comparing them to ensure consistent configuration.

# Configuring DNS Forwarding

When a DNS server gets a query for a resource record (RR) not held by the server, it can use recursion to discover a DNS server that can resolve the RR. If, for example, you use Resolve‑DNSName to attempt to resolve www.packt.com, the DNS server may not hold a zone that would help. Your DNS service then looks to the DNS root servers to discover a DNS server that can via the recursion process. Eventually, the process finds a DNS server that can and does resolve the RR. The details of the server and the resolved RR are then stored in your local DNS Servers cache.

If you are resolving publicly available names, this process works great. But you might have internally supplied DNS names that DNS can’t resolve via the mechanism. AN example might be when two companies merge. There may be internal hostnames (e.g. intranet.kapoho.com and intranet.reskit.org) which can be resolved by the organization’s internal DNS servers but are not available from publicly facing DNS servers. In that scenario, you can setup conditional forwarding. Conditional forwarding enables one DNS server to forward a query to a DNS server or set of servers and not use recursion. You can learn a bit more about conditional forwarding here: https://medium.com/tech-jobs-academy/dns-forwarding-and-conditional-forwarding-f3118bc93984.

An alternative to using conditional forwarding is to use stub zones. You can learn more about the differences between conditional forwarding and stub zones here: https://blogs.msmvps.com/acefekay/2018/03/20/what-should-i-use-a-stub-conditional-forwader-forwarder-or-secondary-zone/.

## Getting Ready

In this recipe, you use DC1, a domain controller and DNS server for reskit.org. You have previously promoted DC1 to be a DC and installed/configured DNS using recipes earlier in this chapter and book

## How to do it...

1. Obtaining the IP Addresses of DNS servers for Packt.Com

$NS = Resolve-DnsName -Name Packt.Com -Type NS |

  Where-Object Name -eq 'Packt.Com'

$NS

1. Obtaining the IPV4 addresses for these hosts

$NSIPS = foreach ($Server in $NS) {

  (Resolve-DnsName -Name $Server.NameHost -Type A).IPAddress

}

$NSIPS

1. Adding conditional forwarder on DC1

$CFHT = @{

  Name          = 'Packt.Com'

  MasterServers = $NSIPS

}DDDD

Add-DnsServerConditionalForwarderZone @CFHT

1. Checking zone on DC1

Get-DnsServerZone -Name Packt.Com

1. Testing conditional forwarding

Resolve-DNSName -Name WWW.Packt.Com -Server DC1 |

 Format-Table

## How it works...

In step 1

In step 2

In step 3

In step 4,

With the final step, step 5,]

Screenshots for each step that generates one

## There's more...

Some things of interest in this recipe

Repeat the recipe structure

# Managing DNS Zones and Resource Records

This recipe, blah blah

## Getting Ready

Specific stuff you need to do this recipe

## How to do it...

1. Step by step

## How it works...

Screenshots for each step that generates one

## There's more...

Some things of interest in this recipe

Repeat the recipe structure