2

Managing Windows Networking

In this chapter, we cover the following recipes:

* New ways to do old things
* Configuring IP addressing
* Installing and authorizing a DHCP server
* Configuring DHCP scopes
* Configuring an IP address from static to DHCP
* Configuring DHCP failover and load balancing
* Configuring DNS servers, zones, and resource records

# Introduction

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

Many of the tools that IT pros use today have been around for a long time but have more recently been replaced by PowerShell cmdlets. In the New ways to do old things recipe, we look at some of the old commands and their replacement cmdlets.

Every server or workstation in your environment needs to have a correct IP configuration. In the Configuring IP addressing recipe, we look at how to set a network interface's IP configuration, including DNS settings.

As an alternative to creating static IP addresses, you can set up a DHCP server to issue IP address configuration to clients by using the Installing and authorizing a DHCP server recipe. Once your DHCP server is set up, 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 IP address from static to DHCP recipe, we set a network interface to get IP configuration from DHCP.

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 the final recipe of this chapter, Configuring DNS zones and resource records, we will configure the DNS server on DC1 with zones and additional resource records.

# New ways to do old things

Networking IT pros in the Windows Server space have been using a number of console applications to perform basic diagnostics for decades. Tools such as Ipconfig, Tracert, and NSlookup are used by IT pros all over the world. The network shell (netsh) is another veritable Swiss Army Knife set of tools to configure and manage Windows networking components.

PowerShell implements a number of cmdlets that do some of the tasks that older Win32 console applications provided. Cmdlets, such as Get-NetIPConfiguration and Resolve-DnsName, are newer alternatives to ipconfig.exe and nslookup.exe.

These cmdlets also add useful functionality. For example, using Test-NetConnection enables you to check whether a host that might block ICMP is supporting inbound traffic on a particular port. ping.exe only uses ICMP, which may be blocked somewhere in the path to the server.

One administrative benefit of using cmdlets rather than older console applications relates to remoting security. With JEA, as discussed in the Implementing Just Enough Administration recipe in [Chapter 1](file:///C:\Users\siddh\OneDrive\Desktop\PEN\September%2022%20-%20Parvathy%20-%20Chapter%201-12%20STY\9781789808537\OEBPS\ch01.html), Establishing a PowerShell Administrative Environment, you can constrain a user to only be able to use certain cmdlets and parameter values. In general, cmdlets make it easier for you to secure servers that are open for remoting.

This recipe shows you some of the new cmdlets that are available with PowerShell and Windows Server 2019.

## Getting ready

This recipe uses two servers: DC1.Reskit.Org and SRV1.Reskit.Org. DC1 is a domain controller in the Reskit.Org domain and SRV1 is a member server. See the recipe Installing Active Directory with DNS for details on how to set up DC1 as a domain controller. You must run this recipe on SRV1.

## How to do it...

1. Examine two ways to retrieve the IP address configuration (ipconfig versus a new cmdlet):

# Two variations on the old way

ipconfig.exe

ipconfig.exe /all

# The new Way

Get-NetIPConfiguration

1. Ping a computer:

# The old way

Ping DC1.Reskit.Org -4

# The New way

Test-NetConnection -ComputerName DC1.Reskit.Org

# And some new things Ping does not do!

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

$ILHT = @{InformationLevel = 'Detailed'}

Test-NetConnection -ComputerName DC1.Reskit.Org -port 389 @ILHT

1. Use the sharing folder from DC1:

# The old way to use a shared folder

net use X: \\DC1.Reskit.Org\C$

# The new way using an SMB cmdlet

New-SMBMapping -LocalPath 'Y:' -RemotePath '\\DC1.Reskit.Org\C$'

# See what is shared the old way:

net use

# And the new way

Get-SMBMapping

1. Share a folder from SRV1:

# Now share the old way

net share Windows=C:\windows

# and the new way

New-SmbShare -Path C:\Windows -Name Windows2

# And see what has been shared the old way

net share

# and the new way

Get-SmbShare

1. Display the contents of the DNS client cache:

# The old way to see the DNS Client Cache

ipconfig /displaydns

# Vs

Get-DnsClientCache

1. Clear the DNS client cache using old and new methods:

ipconfig /flushdns

# Vs the new way

Clear-DnsClientCache

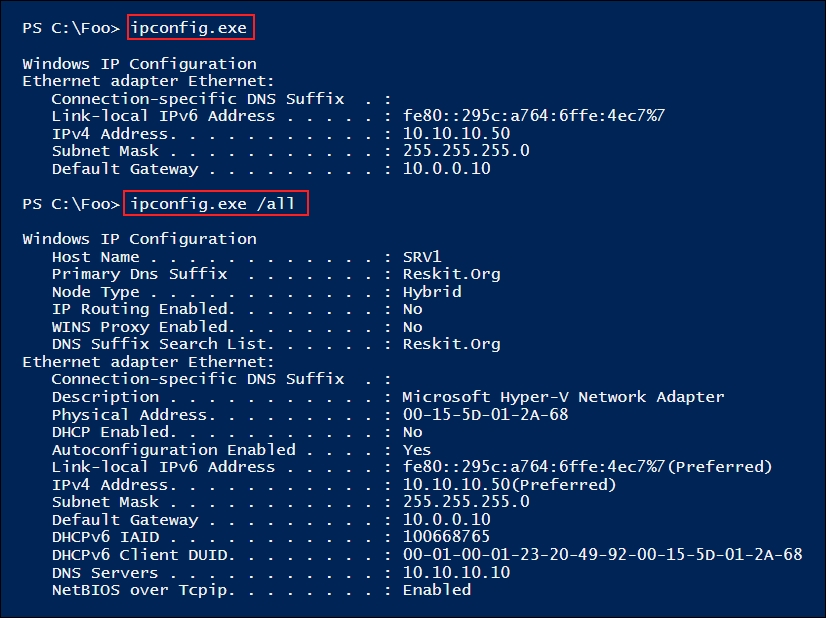
1. Perform DNS lookups:

nslookup DC1.Reskit.Org

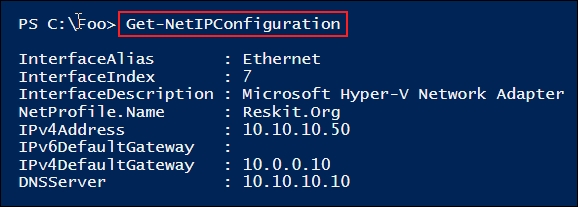
Resolve-DnsName -Name DC1.Reskit.Org -Type ALL

## How it works...

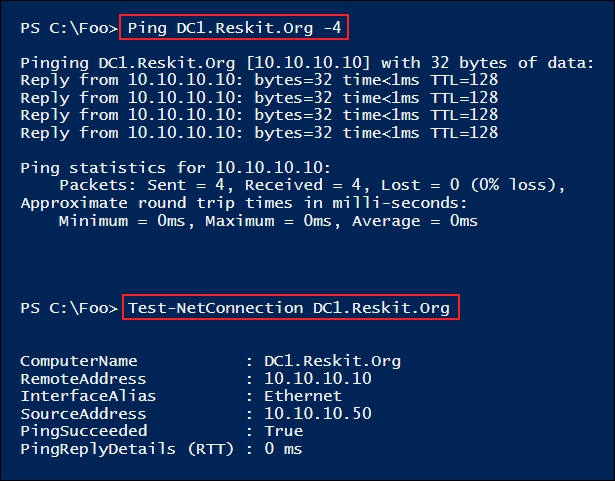
In step 1, you examined the old/new way to view the IP configuration of a Windows host using ipconfig.exe and the Get-NetIPConfiguration cmdlet. First, you looked at two variations of using ipconfig.exe, which looks like this:



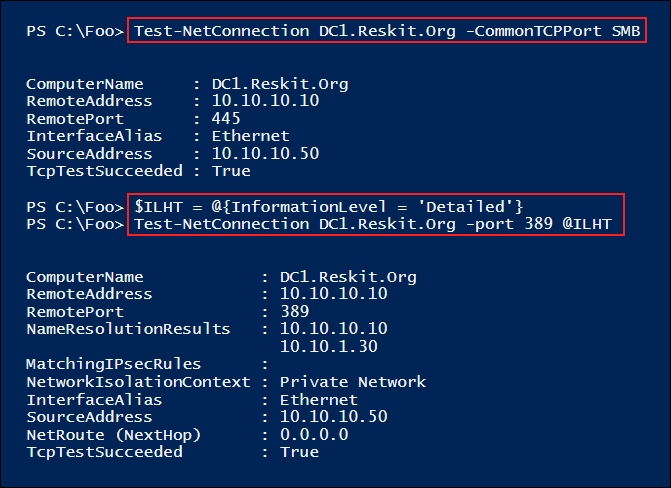
The Get-NetIPConfiguration cmdlet returns similar information, as follows:



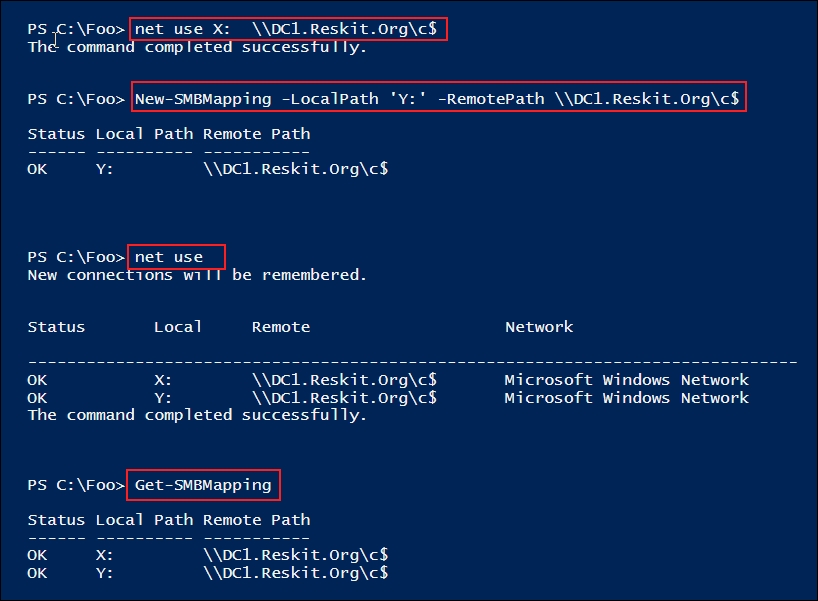
In step 2, you examined the ping.exe command and the newer Test-NetConnection cmdlet. Using these two commands to ping DC1 (from SRV1) looks like this:



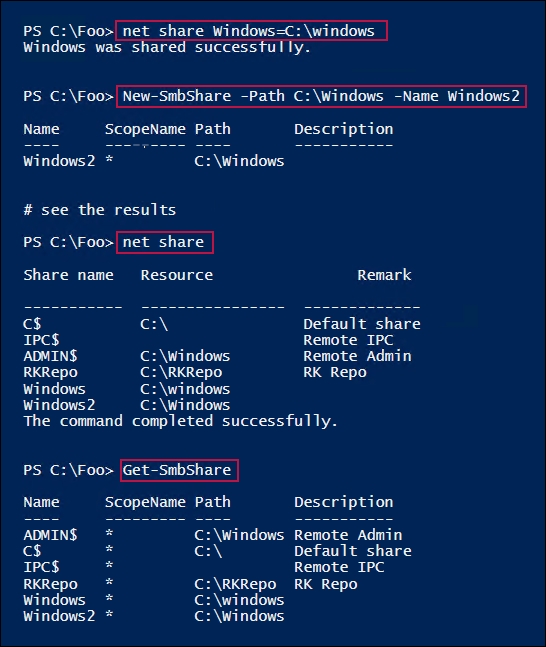
The Test-NetConnection cmdlet is also able to do some things that ping.exe cannot do, including testing access to a specific port (as opposed to just using ICMP) on the target host and providing more detailed information about connecting to that remote port, as you can see here:



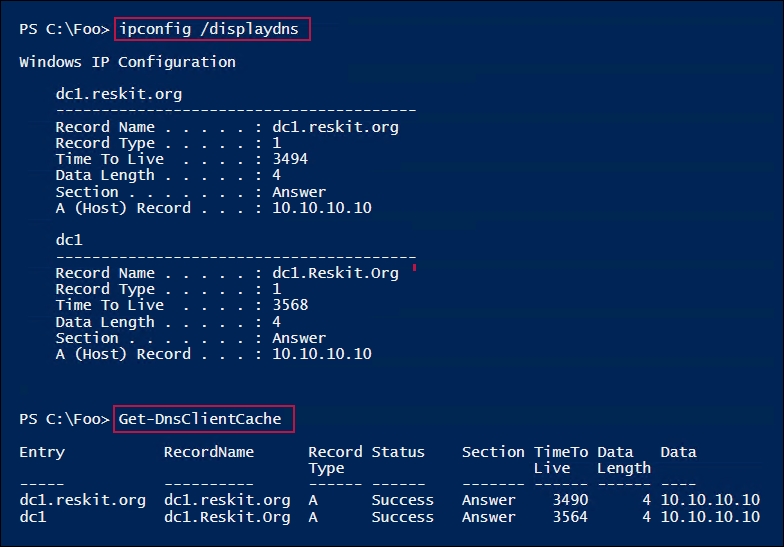
In step 3, you examined new and old ways to create a drive mapping on the local host (that points to a remotely shared folder). The net.exe command, which has been around since the days of Microsoft LAN Manager, enables you to create and view drive mappings. The SMB cmdlets perform similar functions, as you can see here:



In step 4, you created and viewed an SMB share on SRV1, using both net.exe and the SMB cmdlets. This step looks like this:

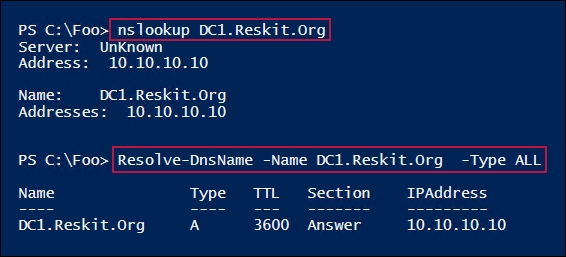


DNS is all too often the focus of network troubleshooting activity. The Windows DNS client holds a cache of previously resolved network names and their IP addresses. This avoids Windows systems from having to perform DNS lookups every time a network host name is used. In step 5, you looked at the old and new ways to view the local DNS cache, which looks like this:



One often-used network troubleshooting technique involves clearing the DNS client cache. You can use ipconfig.exe or the Clear-DNSClientCache cmdlet, as shown in step 6. Neither the ipconfig.exe command or the Clear-DNSClientCache cmdlet produce any output.

Another troubleshooting technique involves asking the DNS server to resolve a DNS name. Traditionally, you would have used nslookup.exe. This is replaced with the Resolve-DNSName cmdlet. The two methods that you used in step 7 look like this:



## There's more...

In step 1, you looked at two ways of discovering a host's IP configuration. Get-NetIPconfiguration, by default, returns the host's DNS server IP address, whereas ipconfig.exe doesn't. On the other hand, ipconfig.exe is considerably quicker.

Ping is meant to stand for Packet InterNetwork Groper and has been an important tool to determine whether a remote system is online. ping.exe uses ICMP echo request/reply, but many firewalls block ICMP (it has been an attack vector in the past). The Test-NetConnection cmdlet has the significant benefit that it can test whether the remote host has a particular port open. On the other hand, the host might block ICMP, if the host is to provide a service, for example, SMB shares, then the relevant port has to be open. Thus, Test-NetConnection is a lot more useful for network troubleshooting.

In step 2, you pinged a server. In addition to ping.exe, there are numerous third-party tools that can help you determine whether a server is online. The TCPing application, for example, pings a server on a specific port using TCP/IP by opening and closing a connection on the specified port. You can download this free utility from <https://www.elifulkerson.com/projects/tcping.php>.

# Configuring IP addressing

By default, Windows uses DHCP to configure any NICs that are found during the installation process. Once you complete the installation of Windows, you can use the settings application netsh.exe, or, of course, PowerShell to set IP configuration manually.

## Getting ready

This recipe runs on SRV2.Reskit.Org. This host is a domain-joined system with an NIC that is initially set up to be configured from DHCP.

## How to do it...

1. Get existing IP address information for SRV2:

$IPType = 'IPv4'

$Adapter = Get-NetAdapter |

Where-Object Status -eq 'Up'

$Interface = $Adapter |

Get-NetIPInterface -AddressFamily $IPType

$IfIndex = $Interface.ifIndex

$IfAlias = $Interface.Interfacealias

Get-NetIPAddress -InterfaceIndex $Ifindex -AddressFamily $IPType

1. Set the IP address for SRV2:

$IPHT = @{

InterfaceAlias = $IfAlias

PrefixLength = 24

IPAddress = '10.10.10.51'

DefaultGateway = '10.10.10.254'

AddressFamily = $IPType

}

New-NetIPAddress @IPHT | Out-Null

1. Set the DNS server details:

$CAHT = @{

InterfaceIndex = $IfIndex

ServerAddresses = '10.10.10.10'

}

Set-DnsClientServerAddress @CAHT

1. Test the new configuration:

Get-NetIPAddress -InterfaceIndex $IfIndex -AddressFamily IPv4

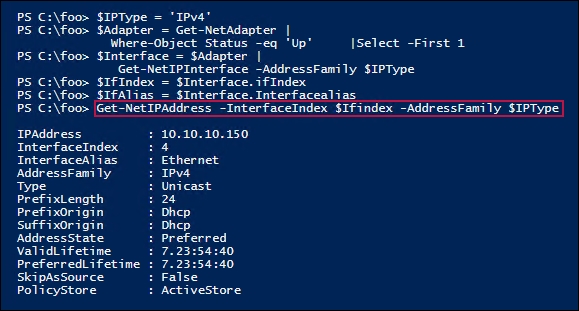
Test-NetConnection -ComputerName DC1.Reskit.Org

Resolve-DnsName -Name SRV2.Reskit.Org -Server DC1.Reskit.Org |

Where-Object Type -eq 'A'

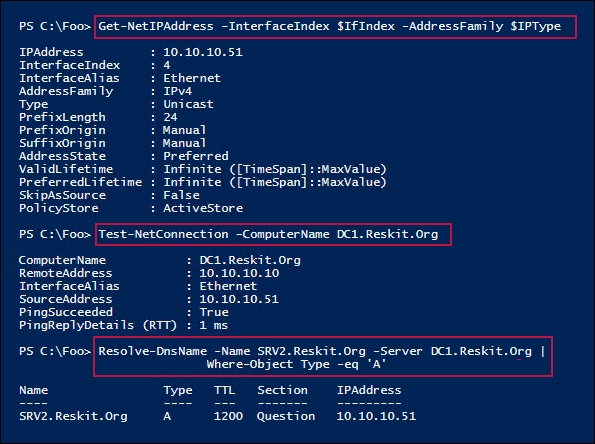
## How it works...

In step 1, you examined the current IP configuration for SRV2, which looks like this:



In step 2, you set a static IP address for the NIC in SRV2, which produces no output. In step 3, you set the DNS server IP address, which also produces no output.

In step 4, you tested the new IP configuration, which looks like this:



## There's more...

In step 1, you used the Get-NetIPConfiguration cmdlet. Two other closely related cmdlets that were not shown in the recipe are Get-NetIPInterface and Get-NetAdapter. Both provide additional information about the network adapter/network interface.

In step 4, you checked the IP configuration by using Get-NetIPAddress to show the IP address and subnet mask. You could have used the Get-NetIPConfiguration cmdlet to return the IP address and subnet mask, plus details of the default gateway and your DNS server IP address.

# Installing and authorizing a DHCP server

In most organizations, your servers are configured with a static IP address configuration, but client computers can get IP addresses from a DHCP server. In Windows (and with most Linux, Macintosh, and mobile phones), the operating system contains a DHCP client that communicates with a DHCP server to obtain an IP address configuration (including the IP address, subnet mask, default gateway, and DNS server IP address).

Installing and authorizing a DHCP server is easy and straightforward. You can use Server Manager to achieve this. Server Manager, though, is a GUI that's layered on top of PowerShell. Alternatively, as you see in this recipe, you can use PowerShell to automate the installation and configuration of DHCP.

## Getting ready

This recipe installs the DHCP service and the related management tools on the DC1.Reskit.Org computer. DC1 is a domain controller in the Reskit.Org domain and is also a DNS server.

## How to do it...

1. Install the DHCP server feature on DC1:

Install-WindowsFeature -Name DHCP -IncludeManagementTools

1. Add the DHCP server's security groups:

Add-DHCPServerSecurityGroup -Verbose

1. Let DHCP know that it's all configured:

$RegHT = @{

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

Name = 'ConfigurationState'

Value = 2

}

Set-ItemProperty @RegHT

1. Authorize the DHCP server in AD:

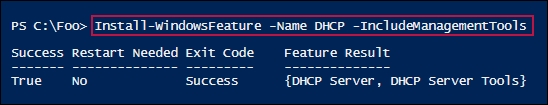
Add-DhcpServerInDC -DnsName DC1.Reskit.Org

1. Restart DHCP:

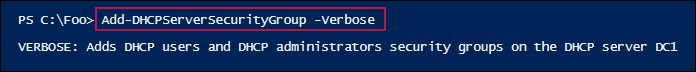
Restart-Service -Name DHCPServer –Force

## How it works...

In step 1, you used the Install-WindowsFeature cmdlet to add the DHCP server and the DHCP management tools, which looks like this:



In step 2, you added the necessary DHCP security groups. By default, this cmdlet does not produce any output. If you want to see some additional output, you can use the –Verbose switch. If you do, the cmdlet produces a bit of output, as follows:



In step 3, you told Windows that the configuration of DHCP is complete. This step produces no output, but is needed to let DHCP know that the necessary security groups are complete.

Before a DHCP server is able to provide IP address information to DHCP client hosts, you need to authorize it in AD. You performed this in step 4, which produces no output.

With the last step, step 5, you restarted the service. Since you authorized the DHCP server in the AD, the DHCP service can now start. After restarting, you can configure the DHCP server with address details and DHCP option values to distribute to DHCP clients.

## There's more...

In step 1, you installed the DHCP server service on your system using the Install-WindowsFeature cmdlet. In earlier versions of the Server Manager PowerShell module, the cmdlet was named Add-WindowsFeature. In Windows Server 2019, Add-WindowsFeature is an alias for Install-WindowsFeature.

In step 2, you used the -Verbose switch. When you use the -Verbose switch with a cmdlet, you can get some additional output that shows you what the cmdlet (or function) is doing. The amount of extra information returned when using the -Verbose switch depends on the cmdlet. Some cmdlets are remarkably terse and provide little or no extra verbose output. Other cmdlets provide more detailed verbose output.

In step 4, you authorized the DHCP server explicitly in the Active Directory. Authorization helps your organization avoid the potential for a rogue user setting up a DHCP server and handing out inappropriate IP address details. If you have multiple domain controllers, you may wish to force AD replication so that all DCs show this server as authorized. While the replication should occur pretty quickly, it never hurts to check the replication status before enabling the DHCP service (as you do in step 5).

# Configuring DHCP scopes

In the previous recipe, Installing and authorizing a DHCP server, you installed and authorized a DHCP server on DC1.Reskit.Org. Before that server can begin to 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 IP address of the DNS server. DHCP options can be set at a scope level or at a server level, depending on the needs of your organization.

## Getting ready

This recipe adds and configures a DHCP scope for your DHCP service on DC1. You installed the DHCP service and authorized it by completing the Installing and authorizing a DHCP server recipe.

## How to do it...

1. Create a new DHCP scope:

$SHT = @{

Name = 'Reskit'

StartRange = '10.10.10.150'

EndRange = '10.10.10.199'

SubnetMask = '255.255.255.0'

ComputerName = 'DC1.Reskit.Org'

}

Add-DhcpServerV4Scope @SHT

1. Get scopes from the DHCP server:

Get-DhcpServerv4Scope -ComputerName DC1.Reskit.Org

1. Set DHCP option values for the DHCP server:

$OHT = @{

ComputerName = 'DC1.Reskit.Org'

DnsDomain = 'Reskit.Org'

DnsServer = '10.10.10.10'

}

Set-DhcpServerV4OptionValue @OHT

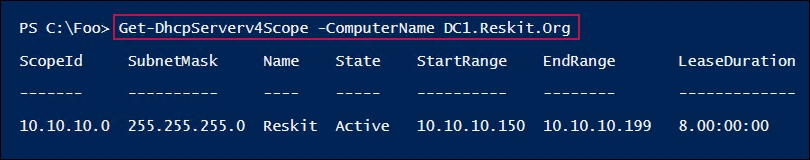
1. Get the options set:

Get-DhcpServerv4OptionValue -ComputerName DC1.Reskit.Org

## How it works...

In this recipe, which uses the DHCP server module, you did some basic DHCP scope management. In particular, you created and updated a DHCP scope. In step 1, you created a new scope for the 10.10.10.0/24 subnet. There is no output from this step.

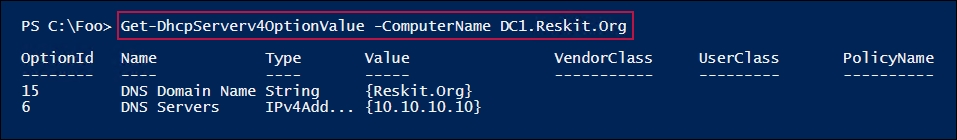
In step 2, you used the Get-DHCPServerV4Scope cmdlet to retrieve details of the scopes defined on DC1, which includes the scope that was set up in step 1. The output looks like this:



To enable a DHCP server to provide all the necessary IP configuration details to DHCP clients, you specified DHCP options. A DHCP option is a particular setting that the server can provide a client, for example, the IP address of a DNS server. You can set an option at the server level of a scope level, depending on your needs.

In step 3, you set two server-wide DHCP options, the DNS domain name (used in client DNS registrations), and the DNS server IP address. There is no output from this step.

In step 4, you used the Get-DHCPServerV4OptionValue cmdlet to see the server-wide DHCP options set on DC1, which looks like this:



## There's more...

In step 1, you created the scope using the New-DHCPServerV4Scope cmdlet. This creates a DHCP scope for the 10.10.10.0/24 subnet, which contains a range of IP addresses that the server can provide to DHCP clients coming from this subnet (that is, 10.10.10.150 – 10.10.10.199).

In step 3, you set an option and option value for the DNS server using the Set-DhcpServerV4OptionValue cmdlet. If you set a DNS server IP address, this cmdlet helpfully checks to see whether the IP address that's provided really is a DNS server (and returns an error message if so).

In this recipe, you created a simple scope for just one subnet that contained only two options. There is more complexity that you may encounter when scaling DHCP, including scope versus server options and client classes, which are outside the scope of this chapter. Nevertheless, the cmdlets used in this recipe form the core of what you might use in practice.

# Configuring IP addresses from static to DHCP

In some cases, you may need to switch the IP address of a server from static back to DHCP. The server may have had a static IP address based on the role it used to perform a role, but you plan to repurpose this server and want to reconfigure the server to obtain IP configuration from DHCP.

## Getting ready

This recipe uses SRV1 which, at the start of the recipe, has manual IP configuration, such as what you created in the Configure IP addressing recipe. In this recipe, you changed from static to DHCP configuration. Also, this recipe assumes that you have DHCP running based on the Installing and authorizing a DHCP server and the Configuring DHCP scopes recipes.

## How to do it...

1. Get the existing IP address' information:

$IPType = 'IPv4'

$Adapter = Get-NetAdapter |

Where-Object Status -eq 'up'

$Interface = $Adapter |

Get-NetIPInterface -AddressFamily $IPType

$IfIndex = $Interface.ifIndex

$IfAlias = $Interface.Interfacealias

Get-NetIPAddress -InterfaceIndex $Ifindex -AddressFamily $IPType

1. Set the interface to get its address from DHCP:

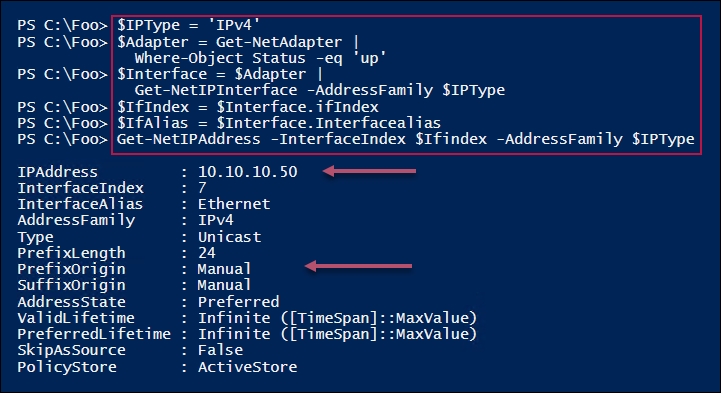
Set-NetIPInterface -InterfaceIndex $IfIndex -DHCP Enabled

1. Test the results:

Get-NetIPAddress -InterfaceIndex $Ifindex -AddressFamily $IPType

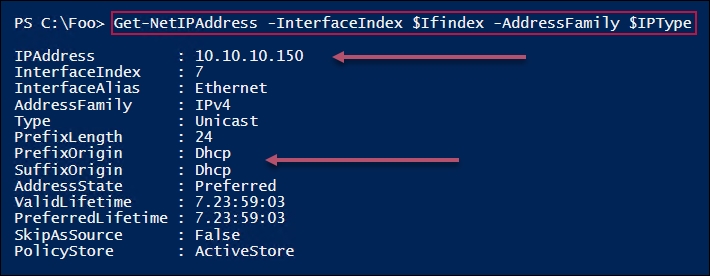
## How it works...

In step 1, you checked the IP address assigned to SRV1, which is manually configured. The output of this step looks like this:



In step 2, you set the NIC to get its IP configuration via DHCP. This step produces no output.

In step 3, you checked the results of changing back to DHCP, which look like this:



## There's more...

In step 3, you obtained the IP address for SRV1 using Get-NetIPAddress. As noted in the New ways to do old things recipe, you could have used ipconfig.exe for a faster result.

# Configuring DHCP failover and load balancing

The basic installation and configuration of a single DHCP server, as shown in the two previous recipes, is straightforward. However, a single DHCP server represents a single point of failure. A standard solution to this shortcoming is to implement DHCP Failover and Load Balancing. Microsoft added this to DHCP with Windows 2012. This feature, and indeed DHCP, is still provided with Server 2019.

## Getting ready

This recipe requires two servers, with one server (DC1) set up with a working and configured DHCP scope. You achieved this by using the Configuring and authorizing a DHCP server and Configure DHCP scopes recipes. This recipe needs a second server (in this case, DC2.Reskit.Org).

## How to do it...

1. Install the DHCP server feature on DC2:

$FHT = @{

Name = 'DHCP','RSAT-DHCP'

ComputerName = 'DC2.Reskit.Org'

}

Install-WindowsFeature @FHT

1. Let DHCP know it's all configured on DC2:

$IPHT = @{

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

Name = 'ConfigurationState'

Value = 2

}

Set-ItemProperty @IPHT

1. Authorize the DHCP server in AD and view the results:

Add-DhcpServerInDC -DnsName DC2.Reskit.Org

1. View the DHCP servers that are authorized in the domain:

Get-DhcpServerInDC

1. Configure DHCP failover and load balancing between DC1 and DC2:

$FHT= @{

ComputerName = 'DC1.Reskit.Org'

PartnerServer = 'DC2.Reskit.Org'

Name = 'DC1-DC2'

ScopeID = '10.10.10.0'

LoadBalancePercent = 60

SharedSecret = 'j3RryIsG0d!'

Force = $true

}

Add-DhcpServerv4Failover @FHT

1. Get active leases in the scope (from both servers):

'DC1', 'DC2' |

ForEach-Object {Get-DhcpServerv4Scope -ComputerName $\_}

1. Now, get server statistics from both servers:

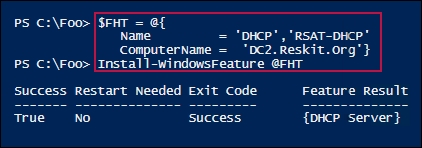
'DC1', 'DC2' |

ForEach-Object {

Get-DhcpServerv4ScopeStatistics -ComputerName $\_}

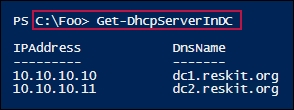
## How it works...

In step 1, you added the DHCP server feature to DC2.Reskit.org, which looks like this:



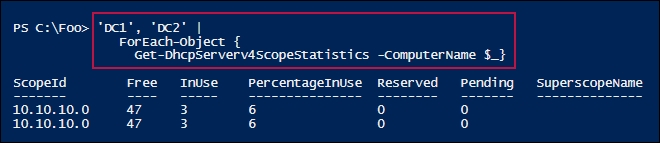
In step 2, you set a registry key to indicate to Windows that DHCP is fully configured. In step 3, you authorized this DHCP server in the AD. There is no output from either of these two steps.

In step 4, you viewed details about the authorized DHCP servers in the Reskit.Org domain, which looks like this:



In step 5, you configured DC1 and DC2 to be in a failover and load-balancing state. This step produces no output.

In step 6, you viewed the active leases on each DHCP server, which looks like this:



## There's more

In step 2, you set a registry key on the DHCP server that indicates that the DHCP server service is fully installed. If you install DHCP using the Server Manager (GUI), this step is performed automatically.

With step 3, you authorized this DHCP server in Active Directory. Without this step, the DHCP service on DC2 would never start up fully. This is intended to ensure that only authorized DHCP servers can hand out DHCP addresses. In step 4, you viewed the authorized servers in the domain.

In step 5, you set up DC2 as a failover and load-balancing DHCP server (with DC1 as the other partner in the relationship). As you can see in step 6, both DHCP servers are synchronized (with 3 addresses used and 47 free).

# Configuring DNS servers, zones, and resource records

In [Chapter 3](file:///C:\Users\siddh\OneDrive\Desktop\PEN\September%2022%20-%20Parvathy%20-%20Chapter%201-12%20STY\9781789808537\OEBPS\ch03.html), Managing Windows Active Directory, in the Installing Active Directory with DNS recipe, you installed a DNS server as part of the installation of AD. This enabled DC1 to be an initial DNS server that provided a home for the various DNS records that were created by AD for the Reskit.Org domain. Adding a DHCP scope with DHCP options that specify 10.10.10.10 (the IP address of DC1.Reskit.Org) means that DHCP clients use DC1 as their DNS server (and register their IP addresses with DC1).

After you perform these two recipes, DHCP clients receive IP address configuration, which includes a DNS server. Thus, DHCP clients can easily resolve IP address for each other and for the domain forest infrastructure (DNS resolution provides AD clients with IP address details for the domain controller and global catalog servers).

The DC installation process, combined with DNS auto registration, means that basic DNS operations just work for DHCP configured clients (and DCs). Each Windows client and Windows server registers its details with the DNS servers on DC1 for others to resolve. This provides a good, basic DNS infrastructure. If you have statically configured servers, you need to ensure that each host has properly configured DNS settings that are pointing to both DNS servers. The IP addresses you assign statically must not interfere with the IP address range(s) provided by DHCP.

Once you have your first DNS server up and running (and AD installed), you should add both a second DC to the domain (outside the scope of this chapter) and add a second DNS server (and update DHCP to ensure that clients are configured with the IP addresses of both DNS servers). Adding a second DNS server (and a second DC) provides resilience and continuity, should a DC/DNS server fail.

In this recipe, you add a second DNS server, update DHCP, and then add a new DNS zone and new resource records.

## Getting ready

This recipe uses three systems: DC1 and DC2, and a client computer, CL1. DC1 is a domain controller with DNS installed, DC2 is a second domain controller, but without DNS installed, and CL1 is a Windows 10 system configured to be a DHCP client.

## How to do it...

1. Add the DNS server service to DC2:

Add-WindowsFeature -Name DNS -ComputerName DC2.Reskit.Org

1. Check that DC1 has replicated Reskit.Org to DC2 after installing DNS:

$DnsSrv = 'DC2.Reskit.Org'

Resolve-DnsName -Name DC1.Reskit.Org -Type A -Server $DnsSrv

1. Add the new DNS server to the DHCP scope:

$OHT = @{

ComputerName = 'DC1.Reskit.Org'

DnsDomain = 'Reskit.Org'

DnsServer = '10.10.10.10','10.10.10.11'

}

Set-DhcpServerV4OptionValue @OHT

1. Check the options on DC1:

Get-DhcpServerv4OptionValue | Format-Table -AutoSize

1. On CL1, check the IP configuration:

Get-DhcpServerv4OptionValue | Format-Table -AutoSize

1. Create a new primary forward DNS zone:

$ZHT = @{

Name = 'Cookham.Reskit.Org'

ReplicationScope = 'Forest'

DynamicUpdate = 'Secure'

ResponsiblePerson = 'DNSADMIN.Reskit.Org'

}

Add-DnsServerPrimaryZone @ZHT

1. Create a new IPv4 primary reverse lookup domain:

$PSHT = @{

Name = '10.in-addr.arpa'

ReplicationScope = 'Forest'

DynamicUpdate = 'Secure'

ResponsiblePerson = 'DNSADMIN.Reskit.Org'

}

Add-DnsServerPrimaryZone @PSHT

1. Check that both zones are available:

Get-DNSServerZone -Name 'Cookham.Reskit.Org', '10.in-addr.arpa'

1. Add an A resource record to the Cookham.Reskit.Org zone:

$RRHT1 = @{

ZoneName = 'Cookham.Reskit.Org'

A = $true

Name = 'Home'

AllowUpdateAny = $true

IPv4Address = '10.42.42.42'

TimeToLive = (30 \* (24 \* 60 \* 60)) # 30 days in seconds

}

Add-DnsServerResourceRecord @RRHT1

1. Check the results of the resource records in the Cookham.Reskit.Org zone:

$Zname = 'Cookham.Reskit.Org'

Get-DnsServerResourceRecord -ZoneName $Zname -Name 'Home'

1. Check the reverse lookup information for DC2:

$RRH = @{

ZoneName = '10.in-addr.arpa'

RRType = 'Ptr'

ComputerName = 'DC2'

}

Get-DnsServerResourceRecord @RRH

1. Add the resource records to the Reskit.Org zone:

$RRHT2 = @{

ZoneName = 'Reskit.Org'

A = $true

Name = 'Mail'

CreatePtr = $True

AllowUpdateAny = $True

IPv4Address = '10.10.10.42'

TimeToLive = '21:00:00'

}

Add-DnsServerResourceRecord @RRHT2

$MXHT = @{

Preference = 10

Name = '.'

TimeToLive = '1:00:00'

MailExchange = 'Mail.Reskit.Org'

ZoneName = 'Reskit.Org'

}

Add-DnsServerResourceRecordMX @MXHT

$GHT = @{

ZoneName = 'Reskit.Org'

Name = '@'

RRType = 'Mx'

}

Get-DnsServerResourceRecord @GHT

1. Test the DNS service on DC1:

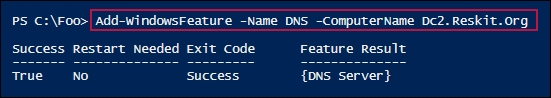
Test-DnsServer -IPAddress 10.10.10.10 -Context DnsServer

Test-DnsServer -IPAddress 10.10.10.10 -Context RootHints

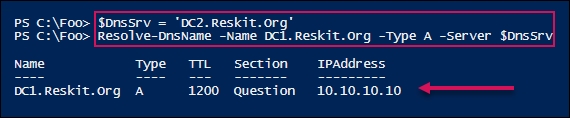
Test-DnsServer -IPAddress 10.10.10.10 -ZoneName 'Reskit.Org'

## How it works...

In step 1, we started by adding the DNS server feature to DC2. The output from this step looks like this:

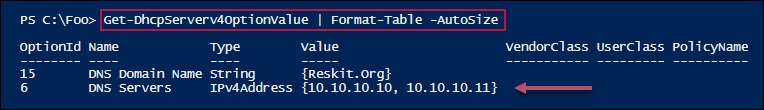


In step 2, you checked the DNS server on DC2 to ensure that it has replicated zone details from DC1 by checking to see whether DC2 can resolve DC1's IP address, which looks like this:

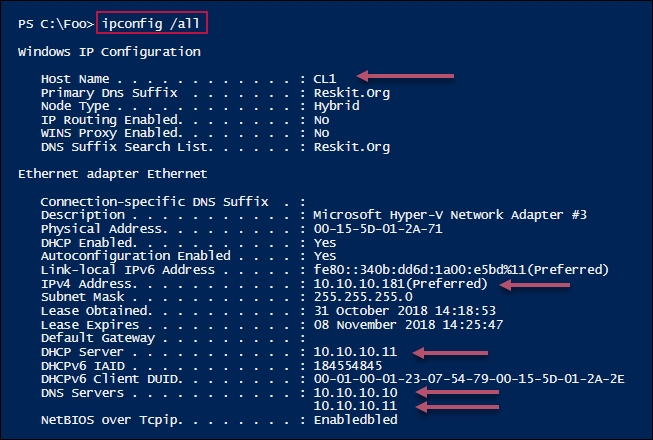


In step 3, which produces no output, you add DC2's IP address to the DHCP scope you created earlier. This enables DHCP clients to obtain the IP address of both DC1 and DC2.

With step 4, you checked on the DHCP options to ensure that the second DNS server address is configured as part of DHCP, which looks like this:

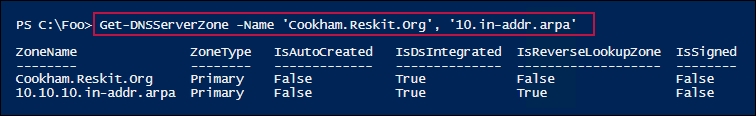


After configuring DHCP to issue both DNS server IP addresses with any leases from the DHCP service, you can validate this by running step 5 on CL1 (a domain joined Windows 10 host set up for DHCP). The output looks like this:

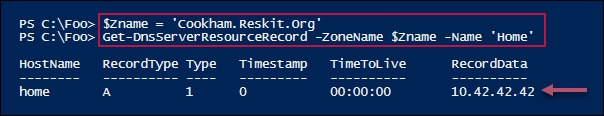


In step 6, you carried out some additional DNS maintenance by creating a forward lookup zone for Cookham.Reskit.Org. Then, in step 7, you created a new IPV4 reverse lookup domain. Neither step produces any output.

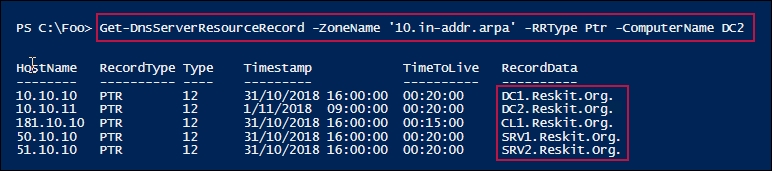
In step 8, you checked that these two zones are available, which looks like this:



In step 9, you added a resource record for Home.Cookham.Reskit.Org. This produces no output. In step 10, you checked that the resource record(s) are available, which looks like this:

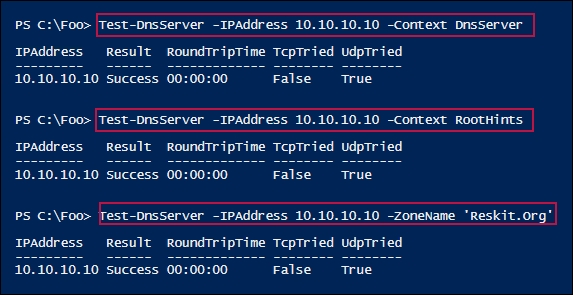


In step 11, you looked at the contents of the reverse lookup zone. Depending on how many hosts have registered with DNS, the output of this step may vary, but should look like this:



In step 12, you added an A resource record (for Mail.Reskit.Org and a mail exchanger (MX) resource record (pointing to Mail.Reskit.Org). Adding these two resource records to DNS creates no output.

In step 13, you tested the DNS service on DC1 by using the Test-DNSServer cmdlet. You used this cmdlet to test that the overall DNS service is up and running, is properly configured with root hints, and that the server is resolving addresses within the Reskit.Org domain. The output of this step appears as follows:



## There's more…

In step 1, you installed the DNS service on DC2, which is a domain controller. When you installed Active Directory on DC1 in the Installing Active Directory with DNS recipe in [Chapter 3](file:///C:\Users\siddh\OneDrive\Desktop\PEN\September%2022%20-%20Parvathy%20-%20Chapter%201-12%20STY\9781789808537\OEBPS\ch03.html), Managing Windows Active Directory, the installation process created a DNS zone for Reskit.Org, and set the replication to replicate to all DCs in the forest. Thus, when you install DNS on DC2, it should be able to immediately resolve the resource records for the Reskit.Org domain. As you can see, DC2 is able to resolve the addresses in the Reskit.Org zone on DC1.

In step 3, you adjusted the DHCP scope that was created in the Configuring DHCP Scopes recipe, which you then tested by first ensuring that the DHCP scope was configured with the IP addresses of both DNS servers and that a DHCP client was configured via DHCP with those addresses.

In step 6, you created a forward lookup DNS zone and, in step 7, you created a reverse lookup zone for the 10.0.0.0/8 set of IP addresses. In step 8, you checked that those two zones were up and running.

In step 9, you create an A resource record for Home.Cookham.Reskit.Org host and, as you can see in step 10, this resource record was successfully resolved.

In step 11, you examined the resource records in the 10.in-addr.arpa zone (that is, hosts with a 10.0.0.0/8 IP address).

In step 12, you added an A resource record for a mail server (Mail.Reskit.Org) and a MX resource record to point to the mail host. Hosts wishing to send mail to any user in the Reskit.Org domain (for example, ThomasLee@Reskit.Org) would be sent to Mail.Reskit.Org. Note that this DNS server (and the mail-related RRs) exist only inside the Reskit.Org network. If Reskit.Org is to receive mail from the internet, then you need to configure your external DNS zones with the appropriate mail server addresses.

These days, many organizations are moving to cloud mail, for example, Google's Gmail. In such cases, you should check with your cloud mail provider as to what resource records are to be set up and what they should contain. You also need to set up the Sender Protected Framework (SPF) to minimize spam coming from your domain. Setting up a mail server and configuring SPF records is outside the scope of this chapter. Look at <https://support.google.com/domains/answer/6304562?hl=en-GB> for an example of how you can set up mail security for G Suite (Google's cloud mail product).