Mastering Windows Server 2019

Second Edition

The complete guide for IT professionals to install and manage Windows Server 2019 and deploy new capabilities



Jordan Krause

Packt
www.packt.com

Mastering Windows Server 2019
Second Edition

The complete guide for IT professionals to install and manage Windows Server 2019 and deploy new capabilities

Jordan Krause

HivaNetwork.Com



BIRMINGHAM - MUMBAI

Mastering Windows Server 2019 Second Edition

Copyright © 2019 Packt Publishing

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

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

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

Commissioning Editor: Vijin Boricha
Acquisition Editor: Meeta Rajani
Content Development Editor: Abhijit Sreedharan
Technical Editor: Aditya Khadye
Copy Editor: Safis Editing
Project Coordinator: Jagdish Prabhu
Proofreader: Safis Editing
Indexer: Pratik Shirodkar
Graphics: Tom Scaria
Production Coordinator: Jayalaxmi Raja

First published: October 2016
Second edition: March 2019

Production reference: 1150319

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78980-453-9

www.packtpub.com



mapt.io

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

Why subscribe?

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

Packt.com

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

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

Contributors

About the author

Jordan Krause is a six-time Microsoft MVP, currently awarded in the Cloud and Datacenter Management category. He has the unique opportunity of working daily with Microsoft networking and remote access technologies. Jordan specializes in Microsoft DirectAccess and Always On VPN. Committed to continuous learning, Jordan holds Microsoft certifications as an MCP, MCTS, MCSA, and MCITP Enterprise Administrator, and regularly writes articles reflecting his experiences with these technologies. Jordan lives in beautiful West Michigan (USA), but works daily with companies around the world.

About the reviewers

Anderson Patricio is a Canadian Microsoft MVP and an IT consultant based in Toronto. His areas of expertise are Microsoft Exchange, Skype for Business, Azure, System Center, and Active Directory. Anderson is an active member of the Exchange Community and he contributes in forums, blogs, articles, and videos. In Portuguese, his website contains thousands of Microsoft tutorials to help the local community, as well as his speaking engagements at TechED in South America and MVA Academy training courses.

Premnath Sambasivam is a Technical Analyst with 6 years of experience in Windows, VMWare, and SCCM administration. He is a MCSE Cloud Platform and Infrastructure certified professional. He has developed and deployed the Microsoft System Center Configuration Manager solution to manage more than 6,000 assets in his client's environment. He loves learning more about and exploring Azure. He is a Microsoft enthusiast.

It was a very pleasant experience overall. Thank you, Sunanda, for choosing me for this project.

HivaNetwork.Com

What this book covers

`Chapter 1`, *Getting Started with Windows Server 2019*, gives us an introduction to the new operating system and an overview of the new technologies and capabilities that it can provide. We will also spend a little bit of time exploring the updated interface for those who may not be comfortable with it yet.

`Chapter 2`, *Installing and Managing Windows Server 2019*, dives right into the very first thing we will have to do when working with Server 2019; installing it! While this seems like a simple task, there are a number of versioning and licensing variables that need to be understood before you proceed with your own install. From there, we will start to expand upon Microsoft's centralized management mentality, exploring the ways in which we can now manage and interact with our servers without ever having to log into them.

`Chapter 3`, *Core Infrastructure Services*, gives us a solid baseline on the technologies that make up the infrastructure of any Microsoft-centric network. We will discuss the big three—Active Directory (AD), Domain Name System (DNS), and Dynamic Host Configuration Protocol (DHCP)—and also address some server backup capabilities, as well as a cheat-sheet list of Microsoft Management Console (MMC) and Microsoft Configuration (MSC) shortcuts to make your day job easier.

`Chapter 4`, *Certificates in Windows Server 2019*, jumps into one of the pieces of Windows Server that has existed for many years and yet, the majority of server administrators that I meet are unfamiliar with it. Let's take a closer look at certificates as they become more and more

commonly required for the new technologies that we roll out. By the end of this chapter, you should be able to spin up your own PKI and start issuing certificates for free!

Chapter 5, *Networking with Windows Server 2019*, begins with an introduction to that big, scary IPv6, and continues from there into building a toolbox of items that are baked into Windows Server 2019 and can be used in your daily networking tasks. We will also discuss Software-Defined Networking.

Chapter 6, *Enabling Your Mobile Workforce*, takes a look at the different remote access technologies that are built into Windows Server 2019. Follow along as we explore the capabilities provided by VPN, DirectAccess, Web Application Proxy, and the brand new Always On VPN.

Chapter 7, *Hardening and Security*, gives some insight into security and encryption functions that are built into Windows Server 2019. Security is the priority focus of CIOs everywhere this year, so let's explore what protection mechanisms are available to us out of the box.

Chapter 8, *Server Core*, throws us into the shrinking world of headless servers. Server Core has flown under the radar for a number of years, but is critical to understand as we bring our infrastructures into a more security-conscious mindset. Let's make sure you have the information necessary to make your environment more secure and more efficient, all while lowering the amount of space and resources that are consumed by those servers.

Chapter 9, *Redundancy in Windows Server 2019*, takes a look at some platforms in Server 2019 that provide powerful data and computing redundancy. Follow along as we discuss Network Load Balancing, Failover Clustering, and information on the updated Storage Spaces

Direct.

Chapter 10, *PowerShell*, gets us into the new, blue command-line interface so that we can become comfortable using it, and also learn why it is so much more powerful than Command Prompt. PowerShell is quickly becoming an indispensable tool for administering servers, especially in cases where you are adopting a centralized management and administration mindset.

Chapter 11, *Containers and Nano Server*, incorporates the terms *open source* and *Linux* in a Microsoft book! Application containers are quickly becoming the new standard for hosting modern, scalable applications. Learn how to start enhancing your DevOps story through the use of tools such as Windows Server Containers, Hyper-V Containers, Docker, and Kubernetes.

Chapter 12, *Virtualizing Your Data Center with Hyper-V*, covers a topic that every server administrator should be very familiar with. Organizations have been moving their servers over to virtual machines in mass quantities for many years. Let's use this chapter to make sure you understand how that hypervisor works, and give you the resources required to build and manage one if and when you have the need.

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

Title Page

Copyright and Credits

Mastering Windows Server 2019 Second Edition

About Packt

Why subscribe?

Packt.com

Contributors

About the author

About the reviewers

Packt is searching for authors like you

Preface

Who this book is for

What this book covers

To get the most out of this book

Conventions used

Get in touch

Reviews

1. Getting Started with Windows Server 2019

The purpose of Windows Server
It's getting cloudy out there

Public cloud

Private cloud

Windows Server versions and licensing

Standard versus Datacenter
Desktop Experience/Server Core/Nano Server

Desktop Experience

Server Core

Nano Server

Licensing models - SAC and LTSC

Semi-Annual Channel (SAC)

Long-Term Servicing Channel (LTSC)

Overview of new and updated features

- The Windows 10 experience continued
- Hyper-Converged Infrastructure
- Windows Admin Center
- Windows Defender Advanced Threat Protection
- Banned Passwords
- Soft restart
- Integration with Linux
- Enhanced Shielded Virtual Machines
- Azure Network Adapter
- Always On VPN

Navigating the interface

- The updated Start menu
- The Quick Admin Tasks menu
- Using the Search function
- Pinning programs to the taskbar
  - The power of right-clicking

Using the newer Settings screen

- Two ways to do the same thing
  - Creating a new user through Control Panel
  - Creating a new user through the Settings menu

Task Manager

Task View

Summary

Questions

2. Installing and Managing Windows Server 2019

Technical requirements
Installing Windows Server 2019

Burning that ISO

Creating a bootable USB stick

Running the installer

Installing roles and features

Installing a role using the wizard

Installing a feature using PowerShell

Centralized management and monitoring

Server Manager

Remote Server Administration Tools (RSAT)
Does this mean RDP is dead?

Remote Desktop Connection Manager

Windows Admin Center (WAC)

Installing Windows Admin Center

Launching Windows Admin Center

Adding more servers to Windows Admin Center

Managing a server with Windows Admin Center

Enabling quick server rollouts with Sysprep

Installing Windows Server 2019 onto a new server

Configuring customizations and updates onto your new server

Running Sysprep to prepare and shut down your master server

Creating your master image of the drive

Building new servers using copies of the master image

Summary

Questions

3. Core Infrastructure Services

What is a Domain Controller?

Active Directory Domain Services

Using AD DS to organize your network

Active Directory Users and Computers

- User accounts
- Security Groups
- Prestaging computer accounts

- Active Directory Domains and Trusts
- Active Directory Sites and Services
- Active Directory Administrative Center

- Dynamic Access Control

- Read-Only Domain Controllers (RODC)

The power of Group Policy

- The Default Domain Policy
- Creating and linking a new GPO
- Filtering GPOs to particular devices

Domain Name System (DNS)

- Different kinds of DNS records

  - Host record (A or AAAA)
  - ALIAS record - CNAME
  - Mail Exchanger record (MX)
  - Name Server (NS) record
  - ipconfig /flushdns

DHCP versus static addressing

The DHCP scope

DHCP reservations

Back up and restore

Schedule regular backups

Restoring from Windows

Restoring from the installer disc

MMC and MSC shortcuts

Summary

Questions

4. Certificates in Windows Server 2019

Common certificate types

User certificates

Computer certificates

SSL certificates

Single-name certificates

Subject Alternative Name certificates

Wildcard certificates

Planning your PKI

- Role services
- Enterprise versus Standalone
- Root versus Subordinate (issuing)
- Naming your CA server
- Can I install the CA role onto a domain controller?

Creating a new certificate template
Issuing your new certificates

- Publishing the template
- Requesting a cert from MMC
- Requesting a cert from the Web interface

Creating an auto-enrollment policy
Obtaining a public-authority SSL certificate

- Public/private key pair
- Creating a Certificate Signing Request
- Submitting the certificate request
- Downloading and installing your certificate

Exporting and importing certificates

- Exporting from MMC
- Exporting from IIS

Importing into a second server

Summary

Questions

5. Networking with Windows Server 2019

Introduction to IPv6

Understanding IPv6 IP addresses

Your networking toolbox

ping

tracert

pathping

Test-Connection

telnet

Test-NetConnection

Packet tracing with Wireshark or Message Analyzer

TCPView

Building a routing table

Multi-homed servers

Only one default gateway

Building a route

Adding a route with the Command Prompt

Deleting a route

Adding a route with PowerShell

NIC Teaming

Software-defined networking

Hyper-V Network Virtualization

Private clouds

Hybrid clouds

How does it work?

System Center Virtual Machine Manager

Network controller

Generic Routing Encapsulation

Microsoft Azure Virtual Network

Windows Server Gateway/SDN Gateway

Virtual network encryption

Bridging the gap to Azure

Azure Network Adapter

Summary

Questions

6. Enabling Your Mobile Workforce

Always On VPN

Types of AOVPN tunnel

User Tunnels

Device Tunnels

Device Tunnel requirements

AOVPN client requirements

Domain-joined

Rolling out the settings

AOVPN server components

Remote Access Server

IKEv2

SSTP

L2TP

PPTP

Certification Authority (CA)

Network Policy Server (NPS)

- DirectAccess
  - The truth about DirectAccess and IPv6
  - Prerequisites for DirectAccess
    - Domain-joined
    - Supported client operating systems
    - DirectAccess servers get one or two NICs
      - Single NIC Mode
      - Dual NICs
      - More than two NICs
    - To NAT or not to NAT?
      - 6to4
      - Teredo
      - IP-HTTPS
      - Installing on the true edge – on the i
      - nternet
      - Installing behind a NAT
    - Network Location Server
    - Certificates used with DirectAccess
      - SSL certificate on the NLS web server
      - SSL certificate on the DirectAccess server
      - Machine certificates on the DA server and all

DA clients

Do not use the Getting Started Wizard (GSW)!

Remote Access Management Console

Configuration

Dashboard

Operations Status

Remote Client Status

Reporting

Tasks

DA, VPN, or AOVPN? Which is best?

Domain-joined or not?

Auto or manual launch

Software versus built-in

Password and login issues with traditional VPNs

Port-restricted firewalls

Manual disconnect

Native load-balancing capabilities

Distribution of client configurations

Web Application Proxy

WAP as AD FS Proxy

Requirements for WAP
Latest improvements to WAP

Preauthentication for HTTP Basic

HTTP to HTTPS redirection

Client IP addresses forwarded to applications

Publishing Remote Desktop Gateway

Improved administrative console

Summary

Questions

7. Hardening and Security

Windows Defender Advanced Threat Protection

Installing Windows Defender AV

Exploring the user interface

Disabling Windows Defender

What is ATP, anyway?

Windows Defender ATP Exploit Guard

Windows Defender Firewall – no laughing matter

Three Windows Firewall administrative consoles

Technet24

Windows Defender Firewall (Control Panel)

Firewall & network protection (Windows Security Settings)

Windows Defender Firewall with Advanced Security (WFAS)

Three different firewall profiles

Building a new inbound firewall rule

Creating a rule to allow pings (ICMP)

Managing WFAS with Group Policy

Encryption technologies

BitLocker and the virtual TPM

Shielded VMs

Encrypted virtual networks

Encrypting File System

IPsec

Configuring IPsec

Server policy

Secure Server policy

Client policy

IPsec Security Policy snap-in

Using WFAS instead

Banned passwords

Advanced Threat Analytics
General security best practices

Getting rid of perpetual administrators
Using distinct accounts for administrative access
Using a different computer to accomplish administrative tasks
Never browse the internet from servers
Role-Based Access Control (RBAC)
Just Enough Administration (JEA)

Summary
Questions

8. Server Core

Why use Server Core?

No more switching back and forth

Interfacing with Server Core

PowerShell

Using cmdlets to manage IP addresses
Setting the server hostname
Joining your domain

HivaNetwork.Com

Remote PowerShell

Server Manager

Remote Server Administration Tools

Accidentally closing Command Prompt

Windows Admin Center for managing Server Core

The Sconfig utility

Roles available in Server Core

What happened to Nano Server?

Summary

Questions

9. Redundancy in Windows Server 2019

Network Load Balancing (NLB)

Not the same as round-robin DNS

What roles can use NLB?

Virtual and dedicated IP addresses

NLB modes

Unicast

Multicast

Multicast IGMP

Configuring a load-balanced website

Enabling NLB

Enabling MAC address spoofing on VMs

Configuring NLB

Configuring IIS and DNS

Testing it out

Flushing the ARP cache

Failover clustering

Clustering Hyper-V hosts

Virtual machine load balancing

Clustering for file services

Scale-out file server

Clustering tiers

Application-layer clustering

Host-layer clustering

A combination of both

How does failover work?

Setting up a failover cluster

Building the servers

Installing the feature

Running the failover cluster manager

Running cluster validation

Running the Create Cluster wizard

Recent clustering improvements in Windows Server

True two-node clusters with USB witnesses

Higher security for clusters

Multi-site clustering
Cross-domain or workgroup clustering

Migrating cross-domain clusters

Cluster operating-system rolling upgrades

Virtual machine resiliency

Storage Replica (SR)

Storage Spaces Direct (S2D)

New in Server 2019

Summary

Questions

10. PowerShell

Why move to PowerShell?

Cmdlets

PowerShell is the backbone

Scripting

Server Core

Working within PowerShell

Launching PowerShell

Default Execution Policy

Restricted

AllSigned

RemoteSigned

Unrestricted

The Bypass mode

Using the Tab key

Useful cmdlets for daily tasks

Using Get-Help

Formatting the output

Format-Table

Format-List

PowerShell Integrated Scripting Environment

PS1 files

PowerShell Integrated Scripting Environment

Remotely managing a server

Preparing the remote server

The WinRM service

Enable-PSRemoting

Allowing machines from other domains or workgroups

Connecting to the remote server

Using -ComputerName

Using Enter-PSSession

Desired State Configuration

Summary

Questions

11. Containers and Nano Server

Understanding application containers

- Sharing resources
- Isolation
- Scalability

Containers and Nano Server
Windows Server containers versus Hyper-V containers

- Windows Server Containers
- Hyper-V Containers

Docker and Kubernetes

- Linux containers
- Docker Hub
- Docker Trusted Registry
- Kubernetes

Working with containers

- Installing the role and feature
- Installing Docker for Windows

  - Docker commands

    - docker --help
    - docker images

docker search

docker pull

docker run

docker ps -a

docker info

Downloading a container image

Running a container

Summary

Questions

12. Virtualizing Your Data Center with Hyper-V

Designing and implementing your Hyper-V Server

Installing the Hyper-V role

Using virtual switches

The external virtual switch

The internal virtual switch

The private virtual switch

Creating a new virtual switch

Implementing a new virtual server

Starting and connecting to the VM

Installing the operating system

Managing a virtual server

Hyper-V Manager

The Settings menu

Checkpoints

Hyper-V Console, Remote Desktop Protocol (RDP), or PowerShell

Windows Admin Center (WAC)

Shielded VMs

Encrypting VHDs

Infrastructure requirements for shielded VMs

Guarded hosts

Host Guardian Service (HGS)

Host attestations

TPM-trusted attestations

Host key attestations

Admin-trusted attestation – deprecated in 2

019

Integrating with Linux
ReFS deduplication

ReFS

Data deduplication

Why is this important to Hyper-V?

Hyper-V Server 2019

Summary

Questions

Assessments

Chapter 1: Getting Started with Windows Server 2019

Chapter 2: Installing and Managing Windows Server 2019

Chapter 3: Core Infrastructure Services

Chapter 4: Certificates in Windows Server 2019

Chapter 5: Networking with Windows Server 2019

Chapter 6: Enabling Your Mobile Workforce

Chapter 7: Hardening and Security

Chapter 8: Server Core

Chapter 9: Redundancy in Windows Server 2019

Chapter 10: PowerShell

Chapter 11: Containers and Nano Server

Chapter 12: Virtualizing Your Data Center with Hyper-V

Another Book You May Enjoy

Leave a review - let other readers know what you think

Preface

I'm really not sure how or when it happened, but we are almost at the year 2020! In fact, part of me *really* wishes that Microsoft had held out on releasing this new version of Windows Server, just so that we could call it Server 2020. Alas, we will have to make do with the far less exotic sounding Server 2019. How amazing to look back and reflect on all of the big changes that have happened in terms of technology over the past 20 years. In some ways, it seems that Y2K has just happened and everyone has been scrambling to make sure their DOS-based and green screen applications are prepared to handle four-digit date ranges. It seems unthinkable to us now that these systems could have been created in a way that was so short-sighted. Did we not think the world would make it to the year 2000? Today, we build technology with such a different perspective and focus. Everything is centralized, redundant, global, and cloud-driven. Users expect 100% uptime, from wherever they are, on whatever device that happens to be sitting in front of them. The world has truly changed.

And, as the world has changed, so has the world of technology infrastructure. This year, we are being introduced to Microsoft's Windows Server 2019. Before we know it, we will be sitting in the year 2020. We are now living in and beyond Doc and Marty's future. My kids have actually ridden around on something called a hoverboard, for crying out loud!

From a user's perspective, as a consumer of data, backend computing requirements are becoming almost irrelevant. Things such as maintenance windows, scheduled downtime, system upgrades, slowness

due to weak infrastructure—these items have to become invisible to the workforce. We are building our networks in ways that allow knowledge workers and developers to do their jobs without consideration for what is supporting their job functions. What do we use to support that level of reliability and resiliency? Our data centers haven't disappeared. Just because we use the words "cloud" and "private cloud" so often doesn't make them magic. What makes all of this centralized, "spin up what you need" mentality a reality is still physical servers running inside physical data centers.

And what is it that drives the processing power of these data centers for the majority of companies in the world? Windows Server. In fact, even if you have gone all-in for cloud adoption and host 100% of your serving resources in the Azure Cloud, you are still making use of Windows Server 2019. It is the operating system that underpins all of Azure! Server 2019 is truly ready to service even the heaviest workloads, in the newest cloud-centric ways.

Over the last few years, we have all become familiar with Software-Defined Computing, using virtualization technology to turn our server workloads into a software layer. Now, Microsoft is expanding on this idea with new terms such as Software-Defined Networking, and even an entire Software-Defined Data Center. The technologies that make these happen allow us to virtualize and share resources on a grand scale.

In order to make our workloads more flexible and cloud-ready, Microsoft is taking major steps in shrinking the server compute platform and creating new ways of interfacing with those servers. There is an underlying preference for new Windows Servers to be running the smaller, efficient, and more secure Server Core interface. Additionally, application containers have made huge advancements over the past year, and Server 2019 now allows us to transition our applications into containers in order to run them in isolation from each other and on a

mass scale. We also have new centralized management tools for administering our servers and networks, namely, the brand new Windows Admin Center that we will be discussing in the forthcoming pages.

Let's take some time together to explore the inner workings of the newest version of this server operating system, which will drive and support so many of our business infrastructures over the coming years. Windows servers have dominated our data centers' rackspaces for more than two decades. Will this newest iteration in the form of Windows Server 2019 continue that trend?

Who this book is for

Anyone interested in Windows Server 2019 or in learning more in general about a Microsoft-centric data center will benefit from this book. An important deciding factor when choosing which content was appropriate for such a volume was making sure that anyone who had a baseline in working with computers could pick this up and start making use of it within their own networks. If you are already proficient in Microsoft infrastructure technologies and have worked with prior versions of Windows Server, then there are some focused topics on the aspects and parts that are brand new and only available in Server 2019. On the other hand, if you are currently in a desktop support role, or if you are coming fresh into the IT workforce, care was taken in the pages of this book to ensure that you will receive a rounded understanding, not only of what is brand new in Server 2019, but also what core capabilities it includes as carryovers from previous versions of the operating system, and that are still crucial to be aware of when working in a Microsoft-driven data center.

To get the most out of this book

Each technology that we discuss within the pages of this book is included in, or relates directly to, Windows Server 2019. If you can get your hands on a piece of server hardware and the Server 2019 installer files, you will be equipped to follow along and try these things out for yourself. We will talk about and reference some enterprise-class technologies that come with stiffer infrastructure requirements in order to make them work fully, and so you may have to put the actual testing of those items on hold until you are working in a more comprehensive test lab or environment, but the concepts are all still included in this book.

We will also discuss some items that are not included in Server 2019 itself, but that are used to extend the capabilities and features of it. Some of these items help tie us into an Azure Cloud environment, and some are provided by third parties, such as using Docker and Kubernetes on your Server 2019 in order to interact with application containers. Ultimately, you do not need to use these tools in order to manage your new Windows Server 2019 environment, but they do facilitate some pretty cool things that I think you will want to look into.

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: "Inside DNS, I am going to create an alias record that redirects `intranet` to `web1`."

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

```
Uninstall-WindowsFeature -Name Windows-Defender
```

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Simply find the appropriate OU for his account to reside within, right-click on the OU, and navigate to **New** | **User**."

> *Warnings or important notes appear like this.*
>
> *Tips and tricks appear like this.*

Get in touch

Feedback from our readers is always welcome.

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

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

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

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

Reviews

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

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

Getting Started with Windows Server 2019

About 10 years ago, Microsoft adjusted its operating system release ideology so that the latest Windows Server operating system is always structured very similarly to the latest Windows client operating system. This has been the trend for some time now, with Server 2008 R2 closely reflecting Windows 7, Server 2012 feeling a lot like Windows 8, and many of the same usability features that came with the Windows 8.1 update are also included with Server 2012 R2. This, of course, carried over to Server 2016 as well—giving it the same look and feel as if you were logged into a Windows 10 workstation.

Now that we are all familiar and comfortable with the Windows 10 interface, we typically have no problems jumping right into the Server 2016 interface and giving it a test drive. Windows Server 2019 is once again no exception to this rule, except that the release of client-side operating systems has shifted a little bit. Now, instead of releasing new versions of Windows (11, 12, 13, and so on), we are, for the time being, simply sticking with Windows 10 and giving it sub-version numbers, indicative of the dates when that operating system was released. For example, Windows 10 version 1703 released around March of 2017. Windows 10 version 1709 was released in September of 2017. Then, we have had 1803 and 1809 as well—although 1809 was delayed a little and didn't release until somewhere closer to November, but that wasn't the original plan. The current plan is Windows OS releases every six months or so, but expecting IT departments to lift and shift all of their servers just for the purposes of moving to an OS that is six months newer is

crazy; sometimes it takes longer than that just to *plan* a migration.

Anyway, I'm getting ahead of myself a little, as we will be discussing versioning of Windows Server later in this chapter, during our *Windows Server versions and licensing* section. The point here is that Windows Server 2019 looks and feels like the latest version of the Windows client operating system that was released at about the same time—that OS being Windows 10 1809. Before we get started talking about the features of Windows Server, it is important to establish a baseline for usability and familiarity in the operating system itself before diving deeper into the technologies running under the hood.

Let's spend a few minutes exploring the new graphical interface and options that are available for finding your way around this latest release of Windows Server, with a view to covering the following topics in this chapter:

- The purpose of Windows Server
- It's getting cloudy out there
- Windows Server versions and licensing
- Overview of new and updated features
- Navigating the interface
- Using the newer Settings screen
- Task Manager
- Task View

HivaNetwork.Com

The purpose of Windows Server

Is asking what the purpose of Windows Server a silly question? I don't think so. It's a good question to ponder, especially now that the definition for servers and server workloads is changing on a regular basis. The answer to this question for Windows clients is simpler. A Windows client machine is a requester, consumer, and contributor of data.

From where is this data being pushed and pulled? What enables the mechanisms and applications running on the client operating systems to interface with this data? What secures these users and their data? The answers to these questions reveal the purpose of servers in general. They house, protect, and serve up the data to be consumed by clients.

Everything revolves around data in business today. Our email, documents, databases, customer lists—everything that we need to do business well, is data. That data is critical to us. Servers are what we use to build the fabric upon which we trust our data to reside.

We traditionally think about servers using a client-server interface mentality. A user opens a program on their client computer, this program reaches out to a server in order to retrieve something, and the server responds as needed. This idea can be correctly applied to just about every transaction you may have with a server. When your domain-joined computer needs to authenticate you as a user, it reaches out to Active Directory on the server to validate your credentials and get an authentication token. When you need to contact a resource by name, your computer asks a DNS server how to get there. If you need to open a

file, you ask the file server to send it your way.

Servers are designed to be the brains of our operation, and often by doing so transparently. In recent years, large strides have been taken to ensure resources are always available and accessible in ways that don't require training or a large effort on the part of our employees.

In most organizations, many different servers are needed in order to provide your workforce with the capabilities they require. Each service inside Windows Server is provided as, or as part of, a **role**. When you talk about needing new servers or configuring a new server for any particular task, what you are really referring to is the individual role or roles that are going to be configured on that server in order to get the work done. A server without any roles installed is useless, though depending on the chassis, can make an excellent paperweight. A 3U SAN device could weigh upwards of 100 pounds and keep your desk orderly even in the middle of a hurricane!

If you think of roles as the meat and potatoes of a server, then the next bit we will discuss is sort of like adding salt and pepper. Beyond the overhead roles you will install and configure on your servers, Windows also contains many features that can be installed, which sometimes stand alone, but more often complement specific roles in the operating system. Features may be something that complement and add functionality to the base operating system such as **Telnet Client**, or a feature may be added to a server in order to enhance an existing role, such as adding the **Network Load Balancing** feature to an already-equipped remote access or IIS server. The combination of roles and features inside Windows Server is what equips that piece of metal to do work.

This book will, quite obviously, focus on a Microsoft-centric infrastructure. In these environments, Windows Server operating

system is king, and is prevalent across all facets of technology. There are alternatives to Windows Server, and different products which can provide some of the same functions to an organization, but it is quite rare to find a business environment anywhere that is running without some semblance of a Microsoft infrastructure.

Windows Server contains an incredible amount of technology, all wrapped up in one small installation disk. With Windows Server 2019, Microsoft has gotten us thinking out of the box about what it means to be a server in the first place, and comes with some exciting new capabilities that we will spend some time covering in these pages. Things such as PowerShell, Windows Admin Center, and Storage Spaces Direct are changing the way that we manage and size our computing environments; these are exciting times to be or to become a server administrator!

It's getting cloudy out there

There's this new term out there, you may have even heard of it...*cloud*. While the word "cloud" has certainly turned into a buzzword that is often misused and spoken of inappropriately, the idea of cloud infrastructure is an incredibly powerful one. A cloud fabric is one that revolves around virtual resources—virtual machines, virtual disks, and even virtual networks. Being plugged into the cloud typically enables things like the ability to spin up new servers on a whim, or even the ability for particular services themselves to increase or decrease their needed resources automatically, based on utilization.

Think of a simple e-commerce website where a consumer can go to order goods. Perhaps 75% of the year, they can operate this website on a single web server with limited resources, resulting in a fairly low cost of service. But, the other 25% of the year, maybe around the holiday seasons, utilization ramps way up, requiring much more computing power. Prior to cloud mentality, this would mean that the company would need to size their environment to fit the maximum requirements all the time, in case it was ever needed. They would be paying for more servers and much more computing power than was needed for the majority of the year. With a cloud fabric, giving the website the ability to increase or decrease the number of servers it has at its disposal as needed, the total cost of such a website or service can be drastically decreased. This is a major driving factor of cloud in business today.

Public cloud

Most of the time, when your neighbor Suzzi Knowitall talks to you about the cloud, she is simply talking about the internet. Well, more accurately she is talking about some service that she uses, which she connects to by using the internet. For example, Office 365, Google Drive, OneDrive, Dropbox—these are all public cloud resources, as they are storing your data *in the cloud*. In reality, your data is just sitting on servers which you access via the internet, but you can't see those servers and you don't have to administer and maintain those servers, which is why it feels like magic and is then referred to as the cloud.

To IT departments, the term "cloud" more often means one of the big three cloud hosting providers. Since this is a Microsoft-driven book, and since I truly feel this way anyway, Azure is top-notch in this category. Azure itself is another topic for another (or many other) book, but is a centralized cloud compute architecture that can host your data, your services, or even your entire network of servers.

Moving your datacenter to Azure enables you to stop worrying or caring about server hardware, replacing hard drives, and much more. Rather than purchasing servers, unboxing them, racking them, installing Windows on them, and then setting up the roles you want configured, you simply click a few buttons to spin up new virtual servers that can be resized at any time for growth. You then pay smaller op-ex costs for these servers—monthly or annual fees for running systems inside the cloud, rather than the big cap-ex costs for server hardware in the first place.

Other cloud providers with similar capabilities are numerous, but the big three are Azure, Amazon (AWS), and Google. As far as enterprise is concerned, Azure simply takes the cake and eats it too. I'm not sure that the others will ever be able to catch up with all of the changes and updates that Microsoft constantly makes to the Azure infrastructure.

Private cloud

While most people working in the IT sector these days have a pretty good understanding of what it means to be part of a cloud service, and many are indeed doing so today, a term which is being pushed into enterprises everywhere and is still many times misunderstood is **private cloud**. At first, I took this to be a silly marketing ploy, a gross misuse of the term "cloud" to try and appeal to those hooked by buzzwords. Boy was I wrong. In the early days of private clouds, the technology wasn't quite ready to stand up to what was being advertised.

Today, however, that story has changed. It is now entirely possible to take the same fabric that is running up in the true, public cloud, and install that fabric right inside your data center. This enables you to provide your company with cloud benefits such as the ability to spin resources up and down, and to run everything virtualized, and to implement all of the neat tips and tricks of cloud environments, with all of the serving power and data storage remaining locally owned and secured by you. Trusting cloud storage companies to keep data safe and secure is absolutely one of the biggest blockers to implementation on the true public cloud, but, by installing your own private cloud, you get the best of both worlds, specifically stretchable compute environments with the security of knowing you still control and own all of your own data.

This is not a book about clouds, public or private. I mention this to give a baseline for some of the items we will discuss in later chapters, and also to get your mouth watering a little bit to dig in and do a little reading yourself on cloud technology. You will see Windows Server 2019

interface in many new ways with the cloud, and will notice that so many of the underlying systems available in Server 2019 are similar to, if not the same as, those becoming available inside Microsoft Azure.

In these pages, we will not focus on the capabilities of Azure, but rather a more traditional sense of Windows Server that would be utilized on-premise. With the big push toward cloud technologies, it's easy to get caught with blinders on and think that everything and everyone is quickly running to the cloud for all of their technology needs, but it simply isn't true. Most companies will have the need for many on-premise servers for many years to come; in fact, many may never put full trust in the cloud and will forever maintain their own data centers. These data centers will have local servers that will require server administrators to manage them. That is where you come in.

Windows Server versions and licensing

Anyone who has worked with the design or installation of a Windows Server in recent years is probably wondering which direction we are taking within this book. You see, there are different capability editions, different technical versions, plus different licensing models of Windows Server. Let's take a few minutes to cover those differences so that you can have a well-rounded knowledge of the different options, and so that we can define which portions we plan to discuss over the course of this book.

Standard versus Datacenter

When installing the Windows Server 2019 operating system onto a piece of hardware, as you will experience in `Chapter 2`, *Installing and Managing Windows Server 2019*, you will have two different choices on server capability. The first is Server 2019 Standard, which is the default option and one that includes most of your traditional Windows Server roles. While I cannot give you details on pricing because that could potentially be different for every company depending on your agreements with Microsoft, Standard is the cheaper option and is used most commonly for installations of Windows Server 2019.

Datacenter, on the other hand, is the luxury model. There are some roles and features within Windows Server 2019 that only work with the Datacenter version of the operating system, and they are not available inside Standard. If ever you are looking to a new piece of Microsoft technology to serve a purpose in your environment, make sure to check over the requirements to find out whether you will have to build a Datacenter server. Keep in mind that Datacenter can cost significantly more money than Standard, so you generally only use it in places where it is actually required. For example, if you are interested in hosting Shielded VMs or working with Storage Spaces Direct, you will be required to run the Server 2019 Datacenter edition on the servers related to those technologies.

One of the biggest functional differences between Standard and Datacenter is the number of **virtual machines** (**VMs**) that they can host. Server 2019 Standard can only run two VMs on it at any given time, which is a pretty limiting factor if you were looking to build out a

Hyper-V server. Datacenter allows you to run unlimited numbers of VMs, which makes it a no-brainer when building your virtualization host servers. For running Hyper-V, Datacenter is the way to go.

Desktop Experience/Server Core/Nano Server

Next up are the different footprints and user interfaces that you can run on your Windows Server 2019 machines. There are three different versions of Windows Server that can be used, and the correct one for you depends on what capabilities and security you are looking for.

Desktop Experience

This is the most common choice among Windows Servers everywhere. Whether you are building a Windows Server 2019 Standard or Datacenter, you have a choice of running Server with or without a graphical user interface. The traditional look and feel, point-and-click interface is called **Desktop Experience**. This allows things such as RDPing into your servers, having a traditional desktop, being able to use the graphical Server Manager right from your logged-in server, and all in all is the best way to go if you are new to server administration.

If you are familiar with navigating around inside Windows 10, then you should be able to at least make your way around in Windows Server 2019 running Desktop Experience. This is the version of Windows Server 2019 that we will be focusing on for the majority of this book, and almost all of the screenshots will be taken from within a Desktop Experience environment.

Server Core

As you will see when we install Windows Server 2019 together, the default option for installation is *not* Desktop Experience. What this means is that choosing the default install path would instead place a headless version of Windows Server onto your machine, most commonly referred to as Server Core. The nature of being headless makes Server Core faster and more efficient than Desktop Version, which makes sense because it doesn't have to run all of that extra code and consume all of those extra resources for launching and displaying a huge graphical interface.

Almost anything that you want to do within Windows Server is possible to do on either Server Core or Desktop Experience, the main differences being interface and security. To be able to use Server Core, you definitely have to be comfortable with a command-line interface (namely PowerShell), and you also have to consider remote server management to be a reliable way of interacting with your servers. We will talk much more about Server Core in `Chapter 8`, *Server Core*.

The largest benefit that Server Core brings to the table, other than performance, is security. Most malware that attempts to attack Windows Servers is reliant upon items that exist inside the GUI of Desktop Experience. Since those things aren't even running inside Server Core—alas, you couldn't get to a *desktop* even if you wanted to—attacks against Server Core machines are much, much less successful.

Nano Server

A third platform for Windows Server 2019 does exist, known as **Nano Server**. This is a tiny version of Windows Server, headless like Server Core but running an even smaller footprint. The last time I booted up a Nano Server, it consumed less than 500 MB of data for the complete operating system, which is incredible.

It seemed like Nano Server was discussed much more surrounding the release of Server 2016, because at that time Microsoft was pressing forward with plans to include a whole bunch of roles inside Nano Server so that we could start replacing some of our bloated, oversized everyday servers with Nano, but that mentality has since gone by the wayside.

As of this writing, Nano Server is pretty well married to the use of containers. In fact, I believe the only supported way to run Nano Server right now is to run it as an image inside a container. We will discuss both in more detail inside `Chapter 11`, *Containers and Nano Server*, but, for the purposes of this summary, it is safe to say that, if you know what containers are, and are interested in using them, then you will benefit from learning all there is to know about Nano Server. If you are not in a position to work with containers, you will probably never run into Nano Server in your environment.

Licensing models - SAC and LTSC

Another decision about how to set up your Windows Servers is what licensing/support model and release cadence you would like to follow. There are two different paths that you can take. It is possible to have a mix of these in a single environment, if you have need for both.

Semi-Annual Channel (SAC)

If you opt to run SAC releases of Windows Server, your naming convention for the operating system changes. Rather than calling it *Server 2019*, you are really running Windows Server 1803, 1809, and so on. It follows the same mentality that Windows 10 does. What that implies is that these new versions of Windows Server SAC are released at much shorter intervals than we have ever seen for servers in the past. The SAC channel is planned to receive two major releases every year—generally in the spring and the fall. Because of the fast release cadence, support for SAC versions of Windows Server lasts for a short 18 months. If you use SAC, you had better get used to always jumping on the latest version shortly after it releases.

If swapping out your server operating systems twice a year sounds daunting, you're not alone. Thankfully, Microsoft recognizes this and realizes that the general server administrator population is not going to use this model for their regular, everyday servers. Rather, SAC-versions of Windows Server are really only going to be used for running containers. In this new world of flexible application hosting, where applications are being written in ways that the infrastructure resources behind those applications can be spun up or spun down as needed, containers are a very important piece of that DevOps puzzle. If you host or build these kinds of applications, you will almost certainly be using containers—now or in the future. When you find yourself in the position of researching and figuring out containers, you will then probably find that the best way to accomplish a highly-performant container environment is by hosting it on SAC server releases.

Long-Term Servicing Channel (LTSC)

Some of you probably think that LTSC is a typo, as in previous years this model was called **Long-Term Servicing Branch** (**LTSB**). While you can go with either and people will generally know what you are talking about, LTSC is now the proper term.

Windows Server 2019 is an LTSC release. Essentially, LTSC releases are what we have always thought of as our traditional Windows Server operating system releases. Server 2008, Server 2008 R2, Server 2012, Server 2012 R2, Server 2016, and now Server 2019 are all LTSC releases. What has changed is that the LTSC releases will now be coming with fewer things that are *wow, that's so awesome and brand-new*, because we will be seeing and getting hints about those brand new things as they are created and rolled out in a more short-term fashion through the SAC releases. So, your SAC releases will come out roughly every six months, and then every two to three years we will experience a new LTSC release that rolls up all of those changes.

While SAC is generally all about DevOps and containers, LTSC servers are for running pretty much everything else. You wouldn't want to install a domain controller, certificate server, or file server and have to replace that server every six months. So, for any of these scenarios, you will always look to LTSC.

One other major difference between the two is that, if you want to use the Desktop Experience version of Windows Server (having a graphical

interface to interact with)—then you're looking at LTSC. The SAC versions of Windows Server do NOT include Desktop Experience—you are limited to only Server Core or Nano Server.

With LTSC versions of Windows Server, you continue to get the same support we are used to: five years of mainstream support followed by five years of available extended support.

Throughout this book, we will be working and gaining experience with Windows Server 2019 - LTSC release.

Overview of new and updated features

The newest version of the Windows Server operating system is always an evolution of its predecessor. There are certainly pieces of technology contained inside that are brand new, but there are even more places where existing technologies have been updated to include new features and functionality. Let's spend a few minutes providing an overview of some of the new capabilities that exist in Windows Server 2019.

The Windows 10 experience continued

Historically, a new release of any Microsoft operating system has meant learning a new user interface, but Server 2019 is the first exception to this rule. Windows 10's release gave us the first look at the current graphical platform, which then rolled into Windows Server 2016, and that was the first time we had seen the current interface on a server platform. Now that Windows 10 updates are releasing but continuing on with essentially the same desktop interface, the same is true for Server 2019. Logging in and using Windows Server 2019 is, in a lot of ways, the same experience that you have had inside Windows Server 2016. Even so, some reading this book have never experienced logging into a server of any kind before, and so we will certainly be looking over that interface, and learning some tips and tricks for navigating around smoothly and efficiently within Server 2019.

Hyper-Converged Infrastructure

When you see the phrase **Hyper-Converged Infrastructure** (**HCI**), it is important to understand that we are not talking about a specific technology that exists within your server environment. Rather, HCI is a culmination of a number of different technologies that can work together and be managed together, all for the purposes of creating the mentality of a **Software-Defined Datacenter** (**SDDC** as it is sometimes referred to). Specifically, HCI is most often referred to as the combination of Hyper-V and **Storage Spaces Direct** (**S2D**) on the same cluster of servers. Clustering these services together enables some big speed and reliability benefits over hosting these roles separately, and on their own systems.

Another component that is part of, or related to, a software-defined data center is **Software Defined Networking** (**SDN**). Similar to how compute virtualization platforms (like Hyper-V) completely changed the landscape of what server computing looked like ten or so years ago, we are now finding ourselves capable of lifting the network layer away from physical hardware, and shifting the design and administration of our networks to be virtual, and managed by Windows Server platform.

A newly available tool that helps configure, manage, and maintain clusters as well as HCI clusters is the new **Windows Admin Center** (**WAC**). WAC can be a hub from which to interface with your Hyper-Converged Infrastructure.

Windows Admin Center

Finally releasing in an official capacity, WAC is one of the coolest things I've seen yet as part of the Server 2019 release. This is a free tool, available to anyone, that you can use to start centrally managing your server infrastructure. While not fully capable of replacing all of the traditional PowerShell, RDP, and MMC console administration tools, it enables you to do a lot of normal everyday tasks with your servers, all from a single interface.

If this capability sounds at all familiar to you, it may be because you tested something called Project Honolulu at some point over the past year. Yes, Windows Admin Center is Project Honolulu, now in full production capacity.

We will take a closer look at the Windows Admin Center in *Chapter 2, Installing and Managing Windows Server 2019*.

Windows Defender Advanced Threat Protection

If you haven't done any reading on **Advanced Threat Protection** (**ATP**), you may see the words Windows Defender and assume I am simply talking about the antivirus/anti-malware capabilities that are now built into both Windows client operating systems, as well as Windows Servers starting with 2016. While it is true that Windows Server 2019 does come out of the box with built-in antivirus, the ATP service is much, much more.

We'll discuss it in more depth in `Chapter 7`, *Hardening and Security*, but the short summary is that Windows Defender Advanced Threat Protection is a cloud-based service that you tap your machines into. The power of ATP is that many thousands, or perhaps even millions, of devices are submitting data and creating an enormous information store that can then be used with some AI and machine learning to generate comprehensive data about new threats, viruses, and intrusions, in real time. ATP customers then receive the benefits of protection as those new threats arise. It's almost like crowd-sourced anti-threat capabilities, with Azure handling all of the backend processing.

Banned Passwords

Active Directory has stored all of our user account information, including passwords, for many years. The last few releases of Windows Server operating system have not included many updates or new features within AD, but Microsoft is now working with many customers inside their cloud-based Azure AD environment, and new features are always being worked on in the cloud. Banned Passwords is one of those things. Natively an Azure AD capability, it can now be synchronized back to your on-premise domain controller servers, giving you the ability to create a list of passwords that cannot be used in any fashion by your users. For example, the word *password*. By banning *password* as a password, you effectively ban any password that includes the word *password*. For example, P@ssword, Password123!, or anything else of similar bearing.

Soft restart

The ability to perform a soft restart was actually new with Server 2016, but it had to be manually added into Server 2016 and I don't think anybody really ever started using it. In the past three years, I have never seen a single person initiate a soft restart, so I assume it is not well-known and I will include it here in our list of features. In an effort to speed up reboots, there is an optional reboot switch called **soft restart,** which is now included automatically inside Server 2019. So, what is a soft restart? It is a restart without hardware initialization.

In other words, it restarts the operating system without restarting the whole machine. It is invoked during a restart by adding a special switch to the `shutdown` command. Interestingly, in Server 2016 you could also invoke a soft restart with the `Restart-Computer` cmdlet in PowerShell, but that option seems to have fallen away in Server 2019. So, if you want to speed up your reboots, you'll have to turn back to good old Command Prompt, as seen in the following:

- Note the following using the `shutdown` command:

```
shutdown /r /soft /t 0
```

Here `/r` is for restart, `/soft` is for soft restart, and `/t 0` is for zero seconds until reboot initiates.

Integration with Linux

Heresy! Under whose authority did I type the word *Linux* inside a book about Windows Server?! Historically, corporate computing environments have run Windows, or they have run Linux, or maybe they have run both but with a very clear separation between the two. Windows Server 2019 blurs that line of separation. We now have the ability to run Linux VMs within our Microsoft Hyper-V, and to even be able to interface with them properly. Did you know some Linux operating systems actually know how to interact with a mouse? Before now, you didn't have much chance of that when trying to run a Linux-based VM on top of a Windows Server, but we now have some compatibility implemented in Hyper-V.

Linux-based containers can also be run on top of Server 2019, which is a big deal for anyone looking to implement scaling applications via containers.

You can even protect your Linux virtual machines by encrypting them, through the use of Shielded Virtual Machines!

Enhanced Shielded Virtual Machines

So many companies are running a majority of their servers as virtual machines today. One of the big problems with this is that there are some inherent security loopholes that exist in the virtualization host platforms of today. One of those holes is backdoor access to the hard disk files of your virtual machines. It is quite easy for anyone with administrative rights on the virtual host to be able to see, modify, or break any virtual machine that is running within that host. And, these modifications can be made in almost untraceable ways. Take a look inside Chapter 12, *Virtualizing Your Data Center with Hyper-V*, to learn how the new capability to create **Shielded Virtual Machines** closes up this security hole by implementing full disk encryption on those VHD files.

Server 2019 brings some specific benefits to the Shielded VM world: we can now protect both Windows-based and Linux-based virtual machines by shielding them, and we are no longer so reliant on communication with the Host Guardian Service when trying to boot protected VMs from our Guarded Host servers. We will discuss this further in Chapter 12, *Virtualizing Your Data Center with Hyper-V*.

Azure Network Adapter

Hybrid Cloud—isn't it great when you can take two separate buzzwords, and combine them to make an even larger and more powerful buzzword? Hybrid Cloud is the thing of CIO's dreams. I hope you know I am jesting on this; the idea of hybrid cloud is incredibly powerful and is the bridge which is making cloud utilization possible. We can have both on-premise servers, and servers hosted in Azure, and make it all one big happy network where you can access any resource from anywhere.

Now, there are already a myriad of technologies that allow you to tap your local network into your Azure network—namely site-to-site VPNs and Azure Express Route. However another option never hurts, especially for small companies that don't want the complexity of building a site-to-site VPN, nor the cost of Express Route.

Enter the Azure Network Adapter. This new capability allows you to very quickly and easily add a virtual network adapter to a Windows Server (even one as far back as 2012 R2), and then connect that virtual NIC straight to your Azure network! Windows Admin Center is required for this transaction to take place; we will take a closer look in `Chapter 5`, *Networking with Windows Server 2019*.

Always On VPN

Users hate launching VPN connections. I know this because I hear that kind of feedback every day. Having to manually make a connection to their work network is wasting time that they could otherwise spend doing actual work. In `Chapter 6`, *Enabling Your Mobile Workforce*, we will discuss the different remote access technologies available inside Windows Server 2019. There are actually two different technologies that allow for a fully automatic connection back to the corporate network, where the users don't have to take any manual action to enact those connections. One of those technologies is DirectAccess and has been around since Server 2008 R2. We will detail DirectAccess because it is still a viable and popular connectivity option, and we will also cover the newest version of automated remote connectivity—Always On VPN.

Navigating the interface

Unfortunately, Microsoft turned a lot of people off with the introduction of Windows 8 and Server 2012, not because functionality or reliability was lacking, but because the interface was so vastly different than it had been before. It was almost like running two separate operating systems at the same time. You had the normal desktop experience, in which all of us spent 99.9% of our time, but then there were also those few moments where you found yourself needing to visit the full page Start menu. More likely, you stumbled into it without wanting to. However you ended up there, inside that fullscreen tablet-like interface, for the remaining 0.01% of your Server 2012 experience you were left confused, disturbed, and wishing you were back in the traditional desktop. I am, of course, speaking purely from experience here. There may be variations in your personal percentages of time spent, but, based on the conversations I have been involved with, I am not alone in these views. And, I haven't even mentioned the magical self-appearing **Charms bar**. Some bad memories are better left in the recesses of the brain.

The major update of Windows 8.1 and Server 2012 R2 came with welcome relief to these symptoms. There was an actual Start button in the corner again, and you could choose to boot primarily into the normal desktop mode. However, should you ever have the need to click on that Start button, you found yourself right back in the full page Start screen, which I still find almost all server admins trying their best to avoid at all costs.

Well, it turns out that Microsoft listened and brought some much-needed relief in Windows 10 and Windows Server 2016. While not quite

back to the traditional Start menu that existed back in 2008, we have a good mix of both old ways and new ways of launching the tools that we need to access on our server platforms.

As far as the graphical interface goes, Windows Server 2019 is mostly unchanged from Server 2016, because we have not seen a major interface update on the client operating system. As you already know, each new version of Windows Server has received updates to the point-and-click interface based on what the latest Windows client operating system is at the time, and this is the first time in many years that a new server operating system has been released while the client operating system is still hanging out on the same version—Windows 10. If you are comfortable navigating around in Windows 10, you will be well-suited to Windows Server 2019.

For anyone who is new to working within Windows, or is just looking for some tips and tricks to get you rolling, this section is for you.

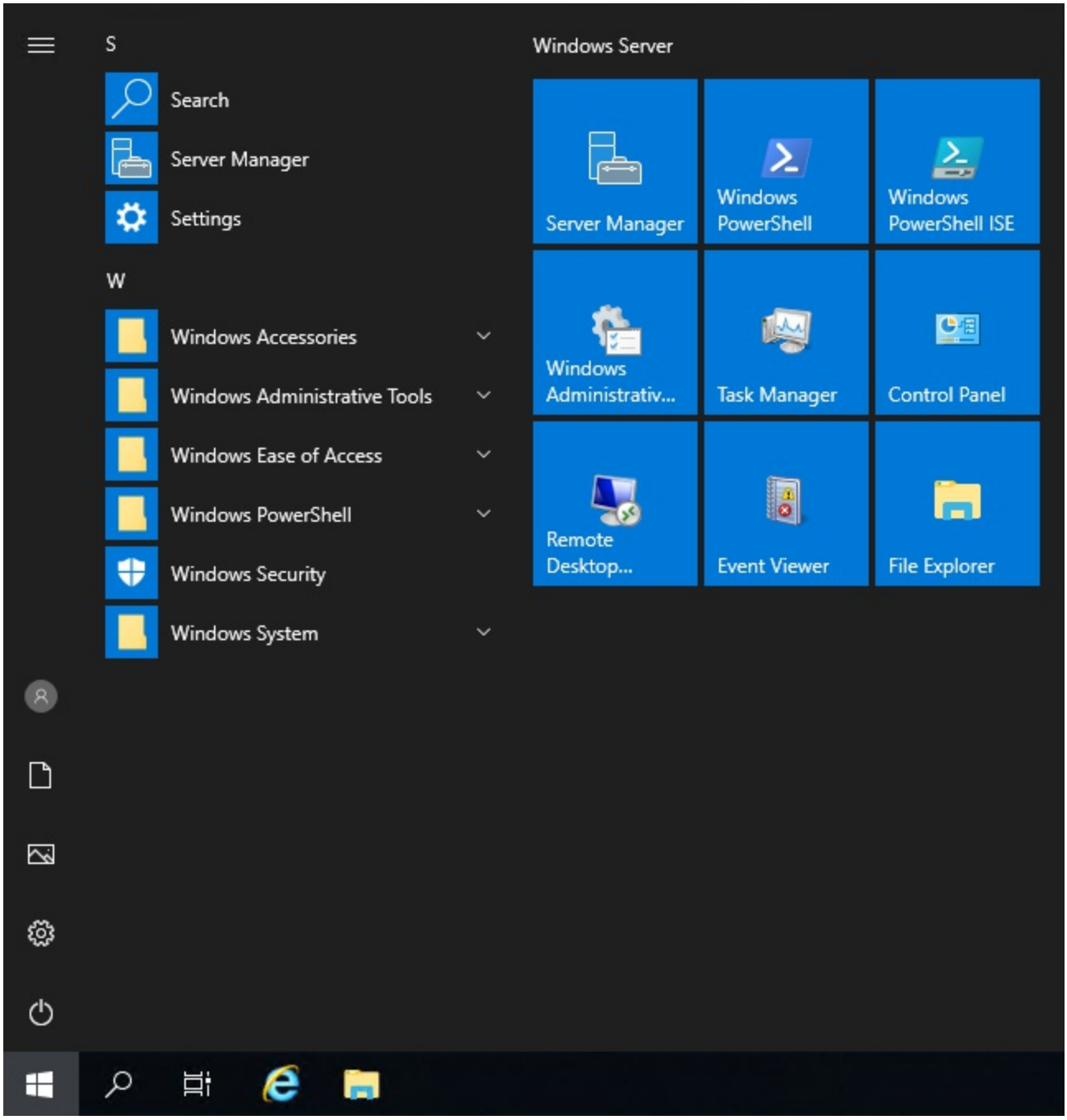
The updated Start menu

As sub-versions of Windows 10 have been released, there have been small ongoing changes to the Start menu. All in all, I consider many of the changes to be backpedalling from the Windows 8 fiasco. We are now back to a real Start button that launches a real Start menu, one that doesn't take over the entire desktop. To be honest, personally I almost never open the Start menu at all, other than to search for the application or feature that I want. We will cover more on that very soon. However, when I do open up the Start menu and look at it, there are a few nice things that stand out.

- All of the applications installed on the server are listed here, in alphabetical order. This is very useful for launching an application, or for doing a quick check to find out whether or not a particular app or feature is installed on your server.
- The left side of the Start menu includes a few buttons for quick access to items. Probably the most useful buttons here are power controls for shutting down or restarting the server, and the **Settings** gear that launches system settings.
- By default, the right side of the Start menu shows some bigger buttons, sometimes called **live tiles**. Pinning items to be shown here, gives you an easy-access location for items that you commonly launch on your server, and having the larger buttons

is useful when you are controlling your server from a touchscreen laptop or something similar.

You can see all three of these functions in the following screenshot:

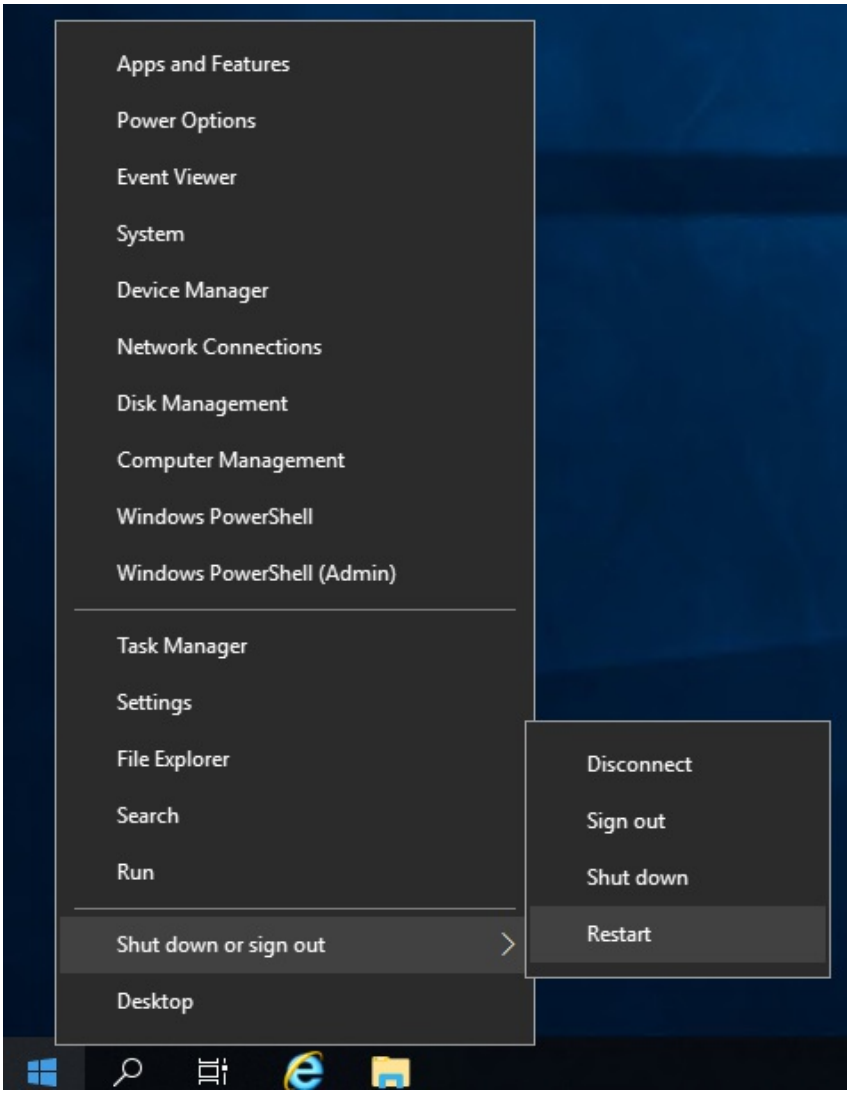


Now, that is a breath of fresh air. A simple but useful Start menu, and more importantly one that loads quickly over remote connections such as RDP or Hyper-V consoles.

The Quick Admin Tasks menu

As nice as it is to have a functional Start menu, as a server administrator I still very rarely find myself needing to access the traditional menu for my day-to-day functions. This is because many items that I need to access are quickly available to me inside the quick tasks menu, which opens by simply right-clicking on the Start button. This menu has been available to us since the release of Windows 8, but many IT professionals are still unaware of this functionality. This menu has become an important part of my interaction with Windows Server operating systems, and hopefully it will be for you as well. Right-clicking on the Start button shows us immediate quick links to do things like open the **Event Viewer**, view the **System** properties, check **Device Manager**, and even **Shut down** or **Restart** the server. The two most common functions that I call for in this context menu are the **Run** function and using it to quickly launch a PowerShell prompt. Even better is the ability from this menu to open either a regular user context PowerShell prompt, or an elevated/administrative PowerShell prompt. Using this menu properly saves many mouse clicks and shortens troubleshooting time.

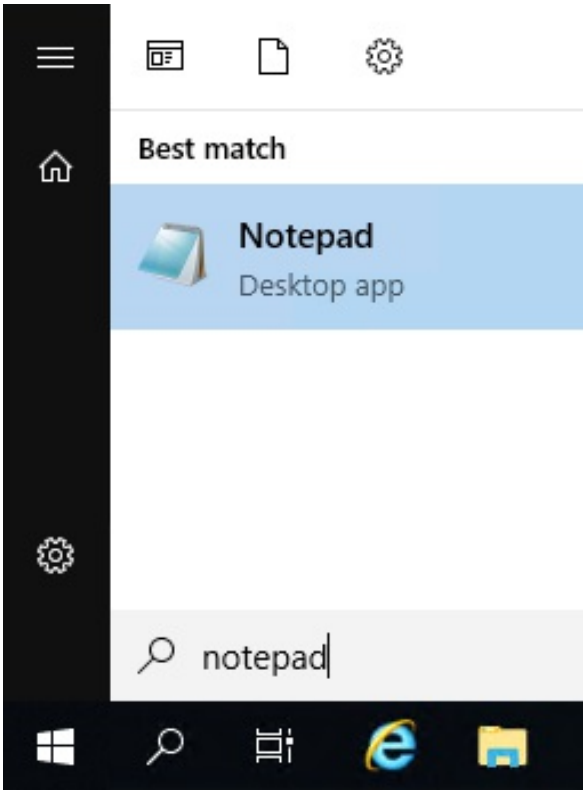
Alternatively, this menu can be invoked using the WinKey + X keyboard shortcut!


Apps and Features
Power Options
Event Viewer
System
Device Manager
Network Connections
Disk Management
Computer Management
Windows PowerShell
Windows PowerShell (Admin)
Task Manager
Settings
File Explorer
Search
Run
Shut down or sign out
Desktop
Disconnect
Sign out
Shut down
Restart

Using the Search function

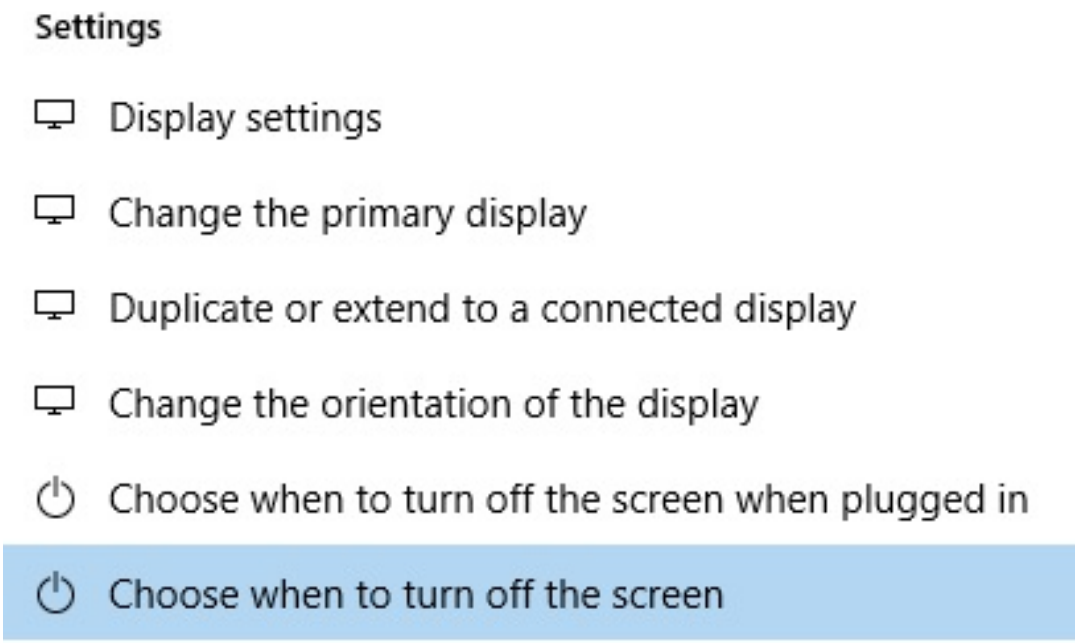
While the Quick Admin menu hidden behind the Start button is useful for calling common administrative tasks, using the Search function inside the Start menu is a powerful tool for interfacing with literally anything on your Windows Server. Depending on who installed applications and roles to your servers, you may or may not have shortcuts available to launch them inside the Start menu. You also may or may not have Desktop shortcuts, or links to open these programs from the taskbar. I find that it is often difficult to find specific settings that may need to be tweaked in order to make our servers run like we want them to. The Control Panel is slowly being replaced by the newer Settings menu in newer versions of Windows, and sometimes this results in the discovery of particular settings being difficult. All of these troubles are alleviated with the search bar inside the Start menu. By simply clicking on the Start button, or even easier by pressing the Windows key (*WinKey*) on your keyboard, you can simply start typing the name of whatever program or setting or document that you want to open up. The search bar will search everything on your local server, and present options to you for which application, setting, or even document, to open.

As a most basic example, press WinKey on your keyboard, then type `notepad` and press the *Enter* key. You will see that good old Notepad opens right up for us. We never had to navigate anywhere in the `Programs` folder in order to find and open it. In fact, we never even had to touch the mouse, which is music to the ears for someone like me who loves doing everything he possibly can via the keyboard:



An even better example is to pick something that would be buried fairly deep inside Settings or the Control Panel. How about changing the amount of time before the screen goes to power save and turns itself off? The traditional server admin will open Control Panel (if you can find it), probably navigate to the Appearance and Personalization section because nothing else looks obviously correct, and still not find what they were looking for. After poking around for a few more minutes, they would start to think that Microsoft forgot to add in this setting altogether. But alas, these power settings are simply moved to a new container, and are no longer accessible through Control Panel at all. We will discuss the new Settings screen momentarily in this chapter, but ultimately for the purposes of this example you are currently stuck at the point where you cannot find the setting you want to change. What is a quick solution? Press your *WinKey* to open the Start menu, and type `monitor` (or `power`, or just about anything else that would relate to the setting you are looking for). You see in the list of available options

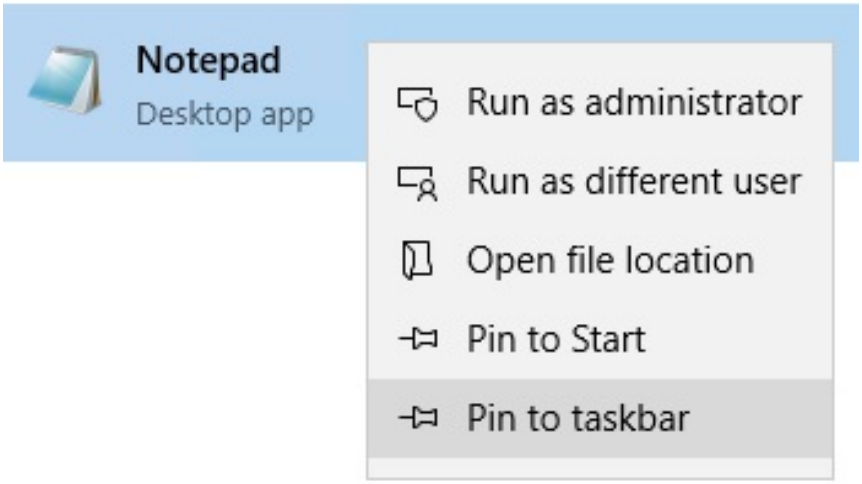
showing in the search menu one called Choose when to turn off the screen. Click on that, and you have found the setting you were looking for all along:



You will also notice that you have many more options on this Search screen than what you were originally searching for. Search has provided me with many different items that I could accomplish, all relating to the word `monitor` that I typed in. I don't know of a more powerful way to open applications or settings on Windows Server 2019 than using the search bar inside the Start menu. Give it a try today!

Pinning programs to the taskbar

While Windows Server 2019 provides great searching capabilities so that launching hard-to-find applications is very easy, sometimes it's easier to have quick shortcuts for commonly used items to be available with a single click, down in the traditional taskbar. Whether you have sought out a particular application by browsing manually through the Start menu, or have used the Search function to pull up the program that you want, you can simply right-click on the program and choose Pin to taskbar in order to stick a permanent shortcut to that application in the taskbar at the bottom of your screen. Once you have done this, during future logins to your session on the server, your favorite and most-used applications will be waiting for you with a single click. As you can see in the following screenshot, you also have the ability to pin programs to the Start menu, which of course is another useful place from which to launch them regularly:

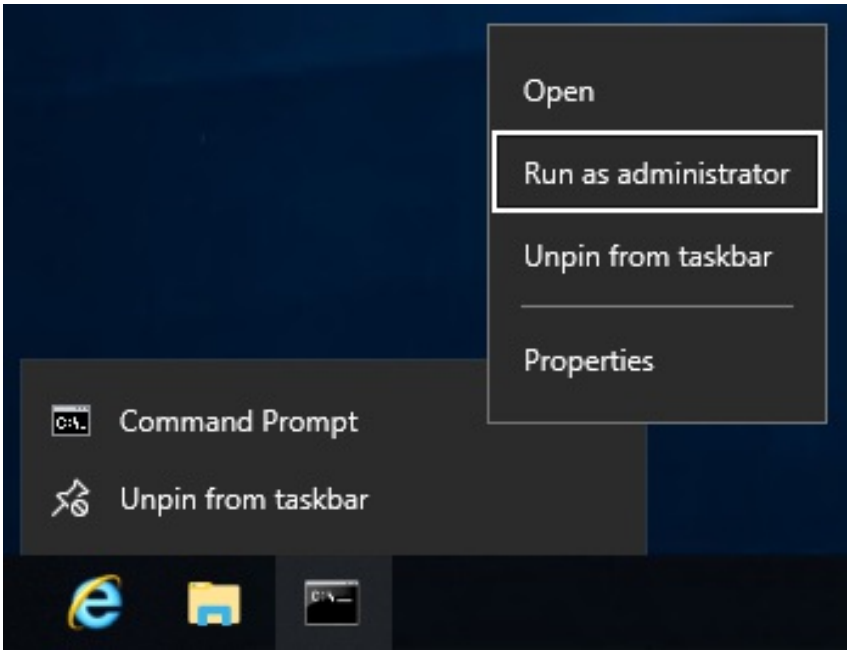


Many readers will already be very familiar with the process of pinning programs to the taskbar, so let's take it one step further to portray an additional function you may not be aware is available to you when you have applications pinned.

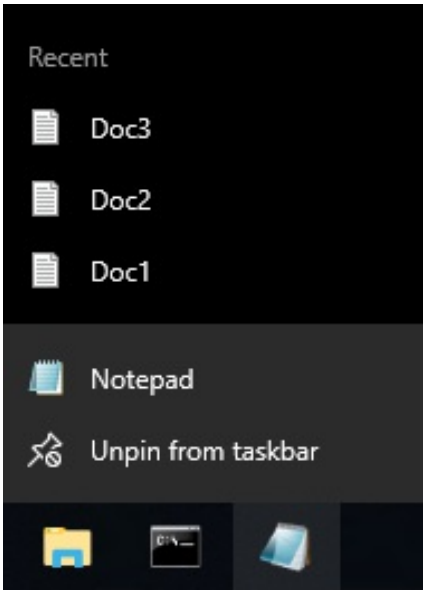
The power of right-clicking

We are all pretty familiar with right-clicking in any given area of a Windows operating system in order to do some more advanced functions. Small context menus displayed upon a right-click have existed since the mouse rolled off the assembly line. We often right-click in order to copy text, copy documents, paste the same, or get into a deeper set of properties for a particular file or folder. Many day-to-day tasks are accomplished with that mouse button. What I want to take a minute to point out is that software makers, Microsoft and otherwise, have been adding even more right-click functionality into application launchers themselves, which makes it even more advantageous to have them close at hand, such as inside the taskbar.

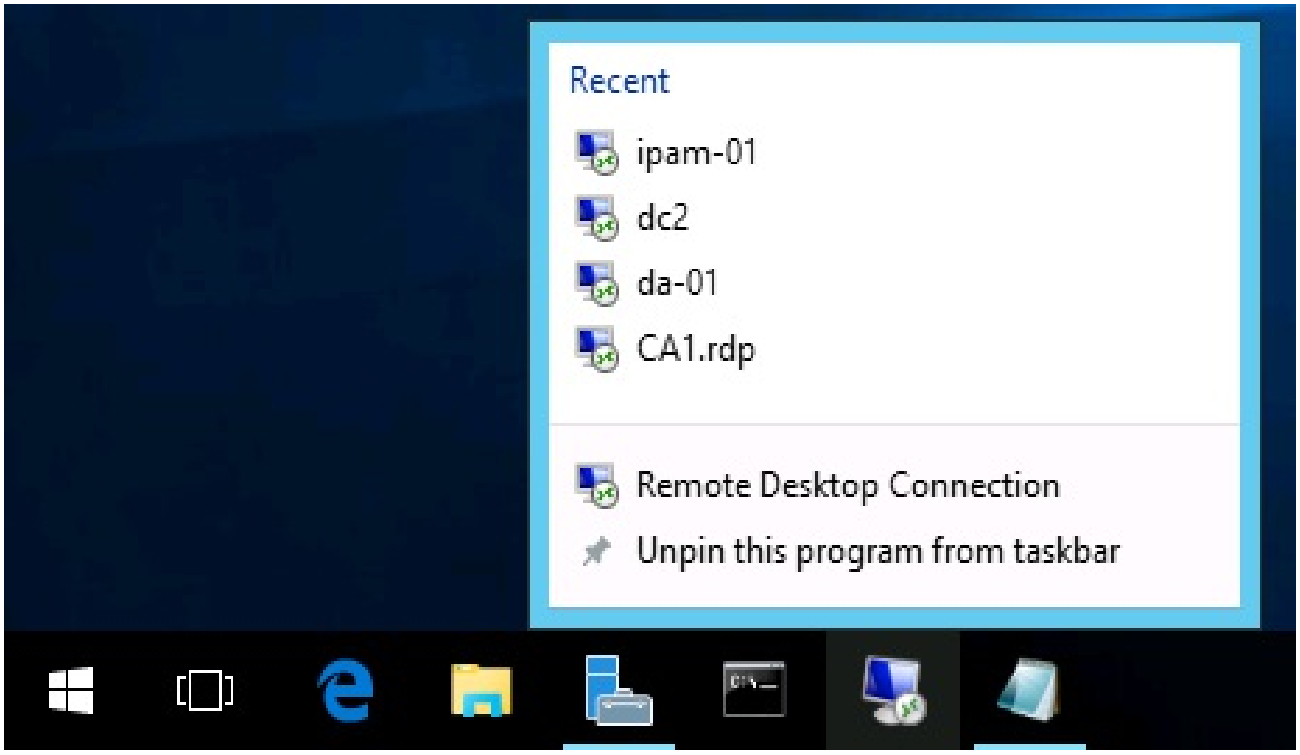
The amount of functionality provided to you when right-clicking on an application in the taskbar differs depending on the application itself. For example, if I were to right-click on Command Prompt, I have options to either open Command Prompt, or to Unpin from taskbar. Very simple stuff. If I right-click again on the smaller menu option for Command Prompt, I have the ability to perform the same functions, but I could also get further into Properties, or Run as administrator. So, I get a little more enhanced functionality the deeper I go:



However, with other programs you will see more results. And, the more you utilize your servers, the more data and options you will start to see in these right-click context menus. Two great examples are Notepad and the Remote Desktop Client. On my server, I have been working in a few text configuration files, and I have been using my server in order to jump into other servers to perform some remote tasks. I have been doing this using the Remote Desktop Client. Now, when I right-click on Notepad listed in my taskbar, I have quick links to the most recent documents that I have worked on:



When right-clicking on my RDP icon, I now have quick links listed right here for the recent servers that I have connected to. I don't know about you, but I RDP into a lot of different servers on a daily basis. Having a link for the Remote Desktop Client in the taskbar automatically keeping track of the most recent servers I have visited, definitely saves me time and mouse clicks as I work through my daily tasks:

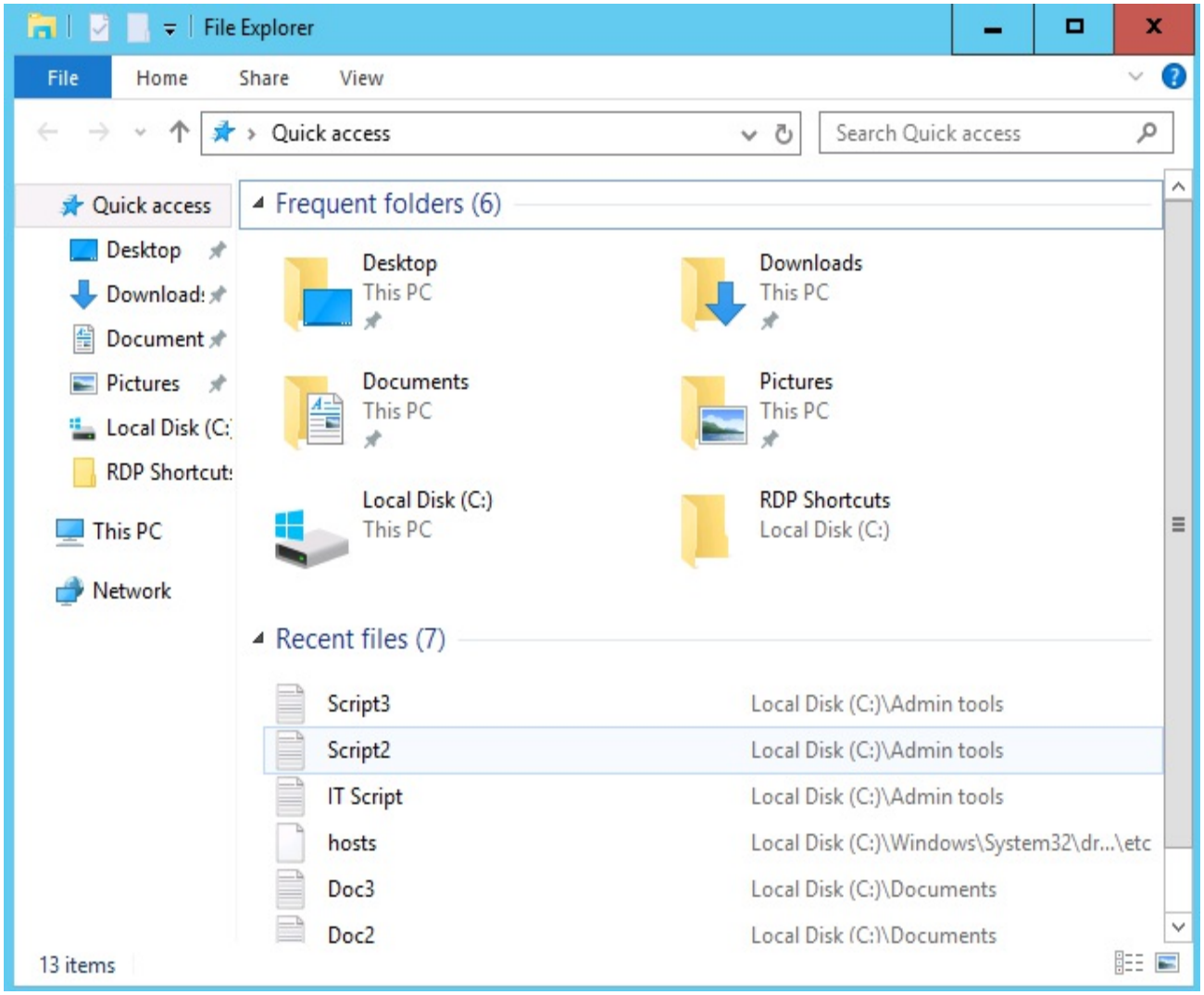


These right-click functions have existed for a couple of operating system versions now, so it's not new technology, but it is being expanded upon regularly as new versions of the applications are released. It is also a functionality that I don't witness many server administrators utilizing, but perhaps they should start doing so in order to work more efficiently, which is why we are discussing it here.

Something that is enhanced in the Windows 10 and Server 2019 platforms that is also very useful on a day-to-day basis is the Quick access view that is presented by default when you open File Explorer. We all know and use File Explorer and have for a long time, but typically when you want to get to a particular place on the hard drive or to a specific file, you have many mouse clicks to go through in order to reach your destination. Windows Server 2019's Quick access view immediately shows us both recent and frequent files and folders, which we commonly access from the server. We, as admins, often have to visit the same places on the hard drive and open the same files time and time

again. Wouldn't it be great if **File Explorer** would lump all of those common locations and file links in one place? That is exactly what **Quick access** does.

You can see in the following screenshot that opening File Explorer gives you quick links to open both frequently accessed folders as well as links to your recent files. A feature like this can be a real time-saver, and regularly making use of these little bits and pieces available to you in order to increase your efficiency, demonstrates to colleagues and those around you that you have a real familiarity and comfort level with this latest round of operating systems:


File Explorer
File
Home
Share
View
Quick access
Search Quick access
Quick access
Desktop
Download
Document
Pictures
Local Disk (C:
RDP Shortcut
This PC
Network
Frequent folders (6)
Desktop
This PC
Downloads
This PC
Documents
This PC
Pictures
This PC
Local Disk (C:)
This PC
RDP Shortcuts
Local Disk (C:)
Recent files (7)
Script3
Local Disk (C:)\Admin tools
Script2
Local Disk (C:)\Admin tools
IT Script
Local Disk (C:)\Admin tools
hosts
Local Disk (C:)\Windows\System32\dr...\etc
Doc3
Local Disk (C:)\Documents
Doc2
Local Disk (C:)\Documents
13 items

Using the newer Settings screen

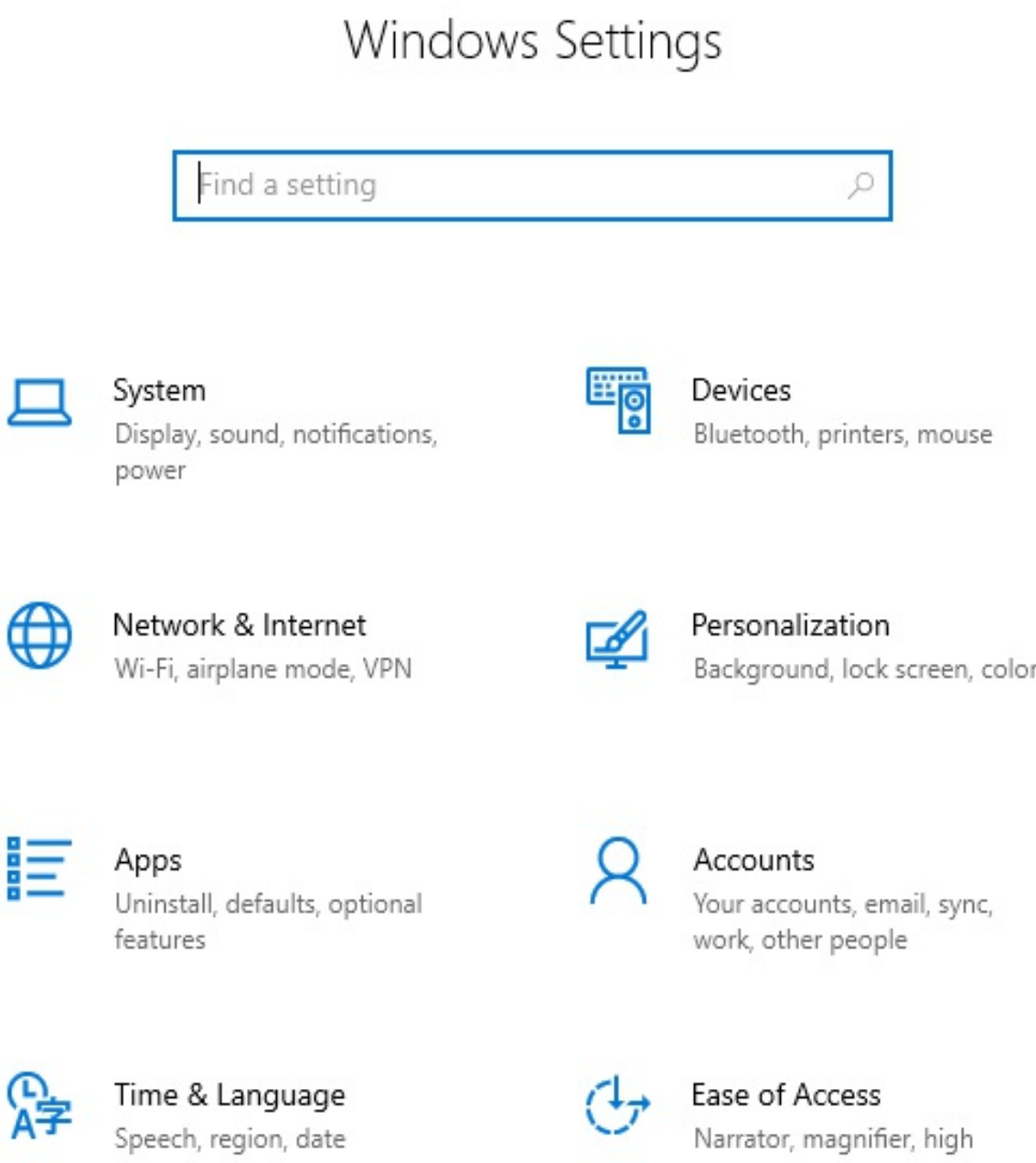
If you work in IT and have been using Windows 10 on a client machine for any period of time, it's a sure bet that you have stumbled across the new Settings interface—perhaps accidentally, as was the case for me, the first time I saw it. I have watched a number of people now bump into the Settings interface for the first time when trying to view or configure Windows Updates. You see, Settings in Windows Server 2019 are just what the name implies, an interface from which you configure various settings within the operating system. What's so hard or confusing about that? Well, we already have a landing platform for all of the settings contained inside Windows that has been around for a zillion years. It's called Control Panel.

The Settings menu inside Windows isn't a brand new idea, but looks and feels quite new when compared to Control Panel. Windows Server 2012 and 2012 R2 had a quasi-presence of settings that as far as I know went largely unused by systems administrators. I believe that to be the effect of poor execution as the Settings menu in 2012 was accessed and hidden behind the Charms bar, which most folks have decided was a terrible idea. We will not spend too much time on technology of the past, but the Charms bar in Server 2012 was a menu that presented itself when you swiped your finger in from the right edge of the screen. Yes, you are correct, servers don't usually have touchscreens. Not any that I have ever worked on, anyway. So, the Charms bar also presented when you hovered the mouse up near the top-right of the screen. It was quite difficult to access, yet seemed to show up whenever you didn't want it to, like when you were trying to click on something near the right of the desktop and instead you clicked on something inside the Charms bar

that suddenly appeared.

I am only giving you this background information in order to segue into this next idea. Much of the user interface in Windows 10, and therefore, Windows Server 2016 and 2019, can be considered a small step backward from the realm of finger swipes and touch screens. Windows 8 and Server 2012 were so focused on big app buttons and finger swipes that a lot of people got lost in the shuffle. It was so different than what we had ever seen before and difficult to use at an administrative level. Because of feedback received from that release, the graphical interface and user controls, including both the Start menu and the Settings menu in Windows Server 2019, are sort of smack-dab in the middle between Server 2008 and Server 2012. This backwards step was the right one to take, and I have heard nothing but praise so far on the new user interface.

So, getting back to the Settings menu, if you click on your Start button, then click on that little gear button just above the power controls, you will see this new interface:

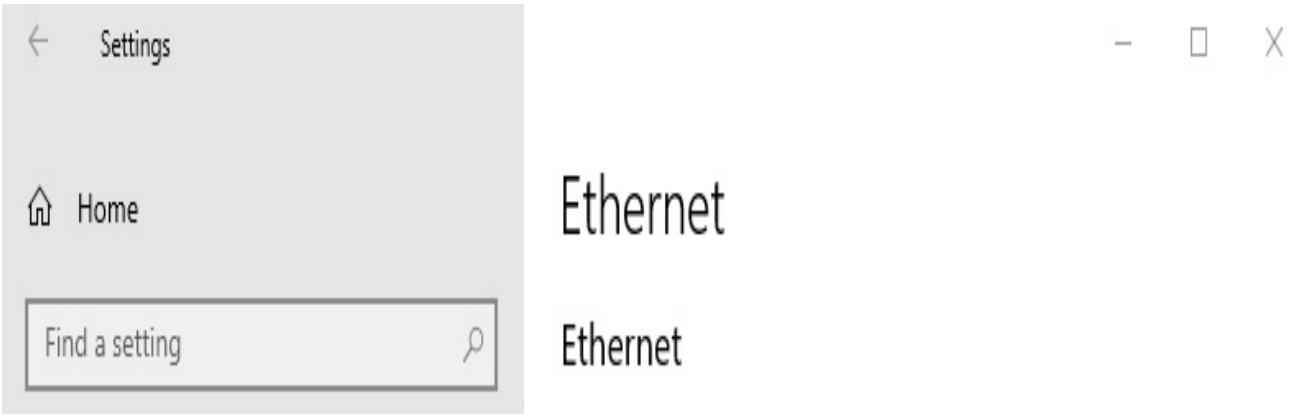


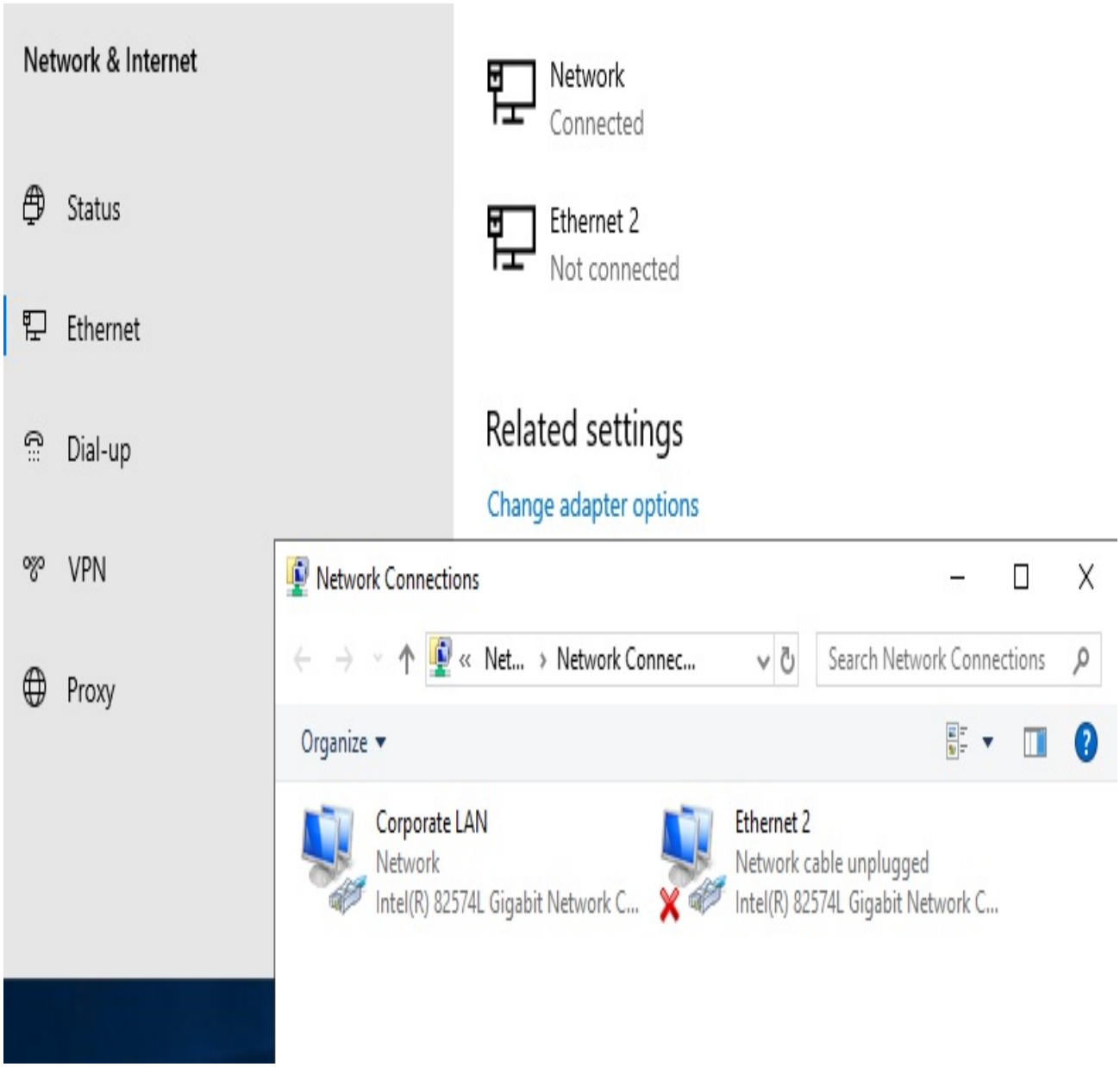
There are many settings and pieces of the operating system that you can configure in this new Settings menu. Some settings in Windows now only exist in this interface, but many can still be accessed either here or through the traditional Control Panel. The goal seems to be a shift toward all configurations being done through the new menu in future

releases, but, for now, we can still administer most setting changes through our traditional methods if we so choose. I mentioned Windows Updates earlier, and that is a good example to look over. Traditionally, we would configure our Windows Update settings via the Control Panel, but they have now been completely migrated over to the new Settings menu in Windows Server 2019. Search Control Panel for `Windows Update`, and the only result is that you can view currently installed updates. But, if you search the new Settings menu for `Windows Update`, you'll find it right away.

> *Remember, you can always use the Windows search feature to look for any setting! Hit your WinKey and type* `Windows Update`*, and you'll be given quick links that take you straight into the appropriate Settings menus.*

For the moment, you will have to use a combination of Control Panel and the Settings menu in order to do your work. It gets confusing occasionally. Sometimes, you will even click on something inside the **Settings** menu, and it will launch a Control Panel window! Try it out. Open up the **Settings** menu and click on Network & Internet. Click on Ethernet in the left column. Here, you can see the status of your network cards, but you can't change anything, such as changing an IP address. Then, you notice the link for Change adapter options. Oh yeah, that sounds like what I want to do. Click on Change adapter options, and you are taken right into the traditional Network Connections screen with the **Control Panel** look-and-feel:



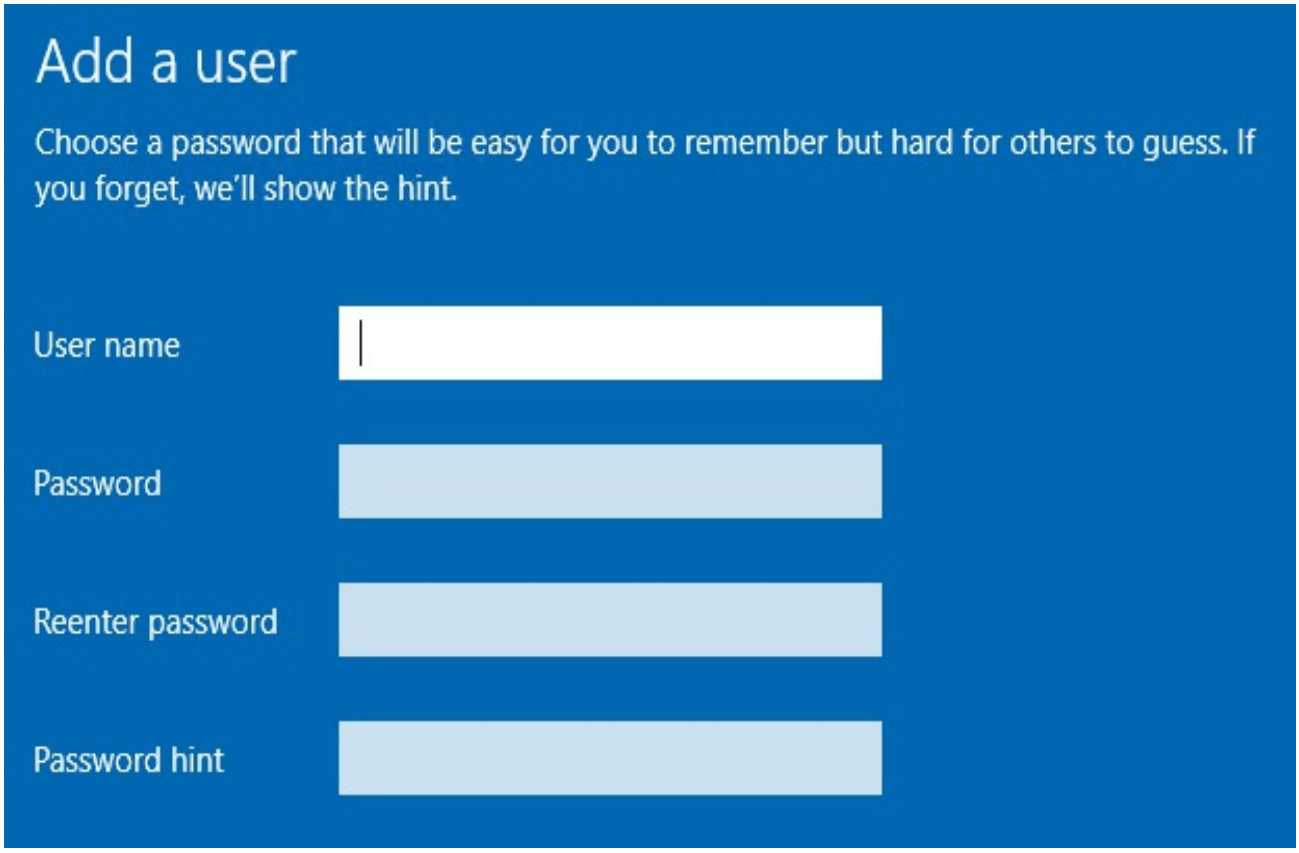

Network & Internet
Status
Ethernet
Dial-up
VPN
Proxy
Network
Connected
Ethernet 2
Not connected
Related settings
Change adapter options
Network Connections
Net... › Network Connec...
Search Network Connections
Organize
Corporate LAN
Network
Intel(R) 82574L Gigabit Network C...
Ethernet 2
Network cable unplugged
Intel(R) 82574L Gigabit Network C...

Two ways to do the same thing

Potentially confusing as well, until you get used to navigating around in here, is that you can sometimes accomplish the same task in either Control Panel or the Settings menu, but the process that you take in each interface can have a vastly different look and feel. Let's take a look at that firsthand by trying to create a new user account on our server, once via **Control Panel**, and again via **Settings**.

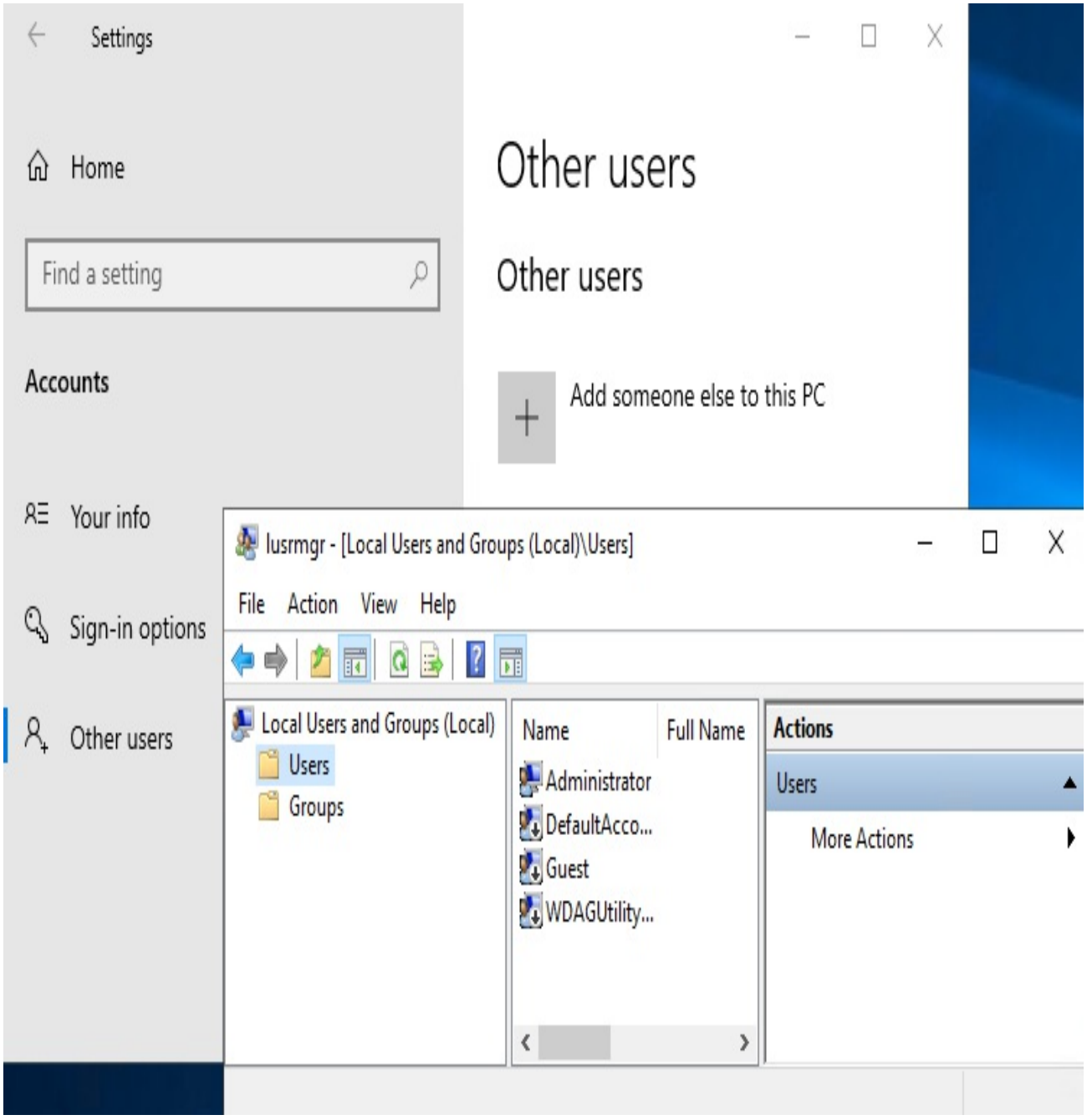
Creating a new user through Control Panel

You are probably pretty familiar with this. Open Control Panel and click on User Accounts. Then, click on the User Accounts heading. Now, click on the link to Manage another account. Inside this screen is your option to Add a user account. Click on that and you get the dialog box where you enter a username and password for your new user:



Creating a new user through the Settings menu

Let's take this newer **Settings** interface for a test drive. Open the Settings menu, and click on Accounts. Now, click on Other users in the left column. There is an option here to Add someone else to this PC; go ahead and click on that:



What in the world is that? Not what I expected, unfortunately. To my surprise, the old **Control Panel** user account launches a nice, fresh-looking interface from which I can create new user accounts. Accessing user accounts via the newer **Settings** console launches me into the old **Local Users and Groups** manager. Technically, from here I could

definitely go ahead and create new user accounts, but it seems like there is some sort of a disconnect here. You would naturally think that the new **Settings** would initiate the newer, nicer screen for adding new user accounts, but we found the opposite to be true.

We walked through this simple example of attempting to perform the same function through two different interfaces to showcase that there are some items which can and must be performed within the new Settings menu context, but there are many functions within Windows that still need to be accomplished through our traditional interfaces. While Control Panel continues to exist, and probably will for a very long time, you should start navigating your way around the Settings menu and figure out what is available inside, so that you can start to shape your ideas for the best combination of both worlds in order to manage your servers effectively.

Just one last thing to point out as we start getting comfortable with the way that the new Settings menus look: many of the settings that we configure in our servers are on/off types of settings. By that I mean we are setting something to either one option or another. Historically, these kinds of configurations were handled by either drop-down menus or by radio buttons. That is normal; that is expected; that is Windows. Now, you will start to see little swipe bars, or sliders, that allow you to switch settings on or off, like a light switch. Anyone who has used the settings interface of any smart phone knows exactly what I am talking about. This user interface behavior has now made its way into the full Windows operating systems, and is probably here to stay. Just to give you an idea of what it looks like inside the context of the new Settings menu, here is a screenshot of the current **Windows Update** settings page inside the Update & Security settings.

This is a good example of those on/off slider buttons:

Settings

Advanced options

Update options

Give me updates for other Microsoft products when I update Windows.

Off

Automatically download updates, even over metered data connections (charges may apply)

Off

Update notifications

Show a notification when your PC requires a restart to finish updating

Off

Pause updates

Temporarily pause updates from being installed on this device for up to 35 days. When updates resume, this device will need to get the latest updates before it can be paused again.

Off

Pausing now will pause updates until 2/13/2019.

Delivery Optimization

Task Manager

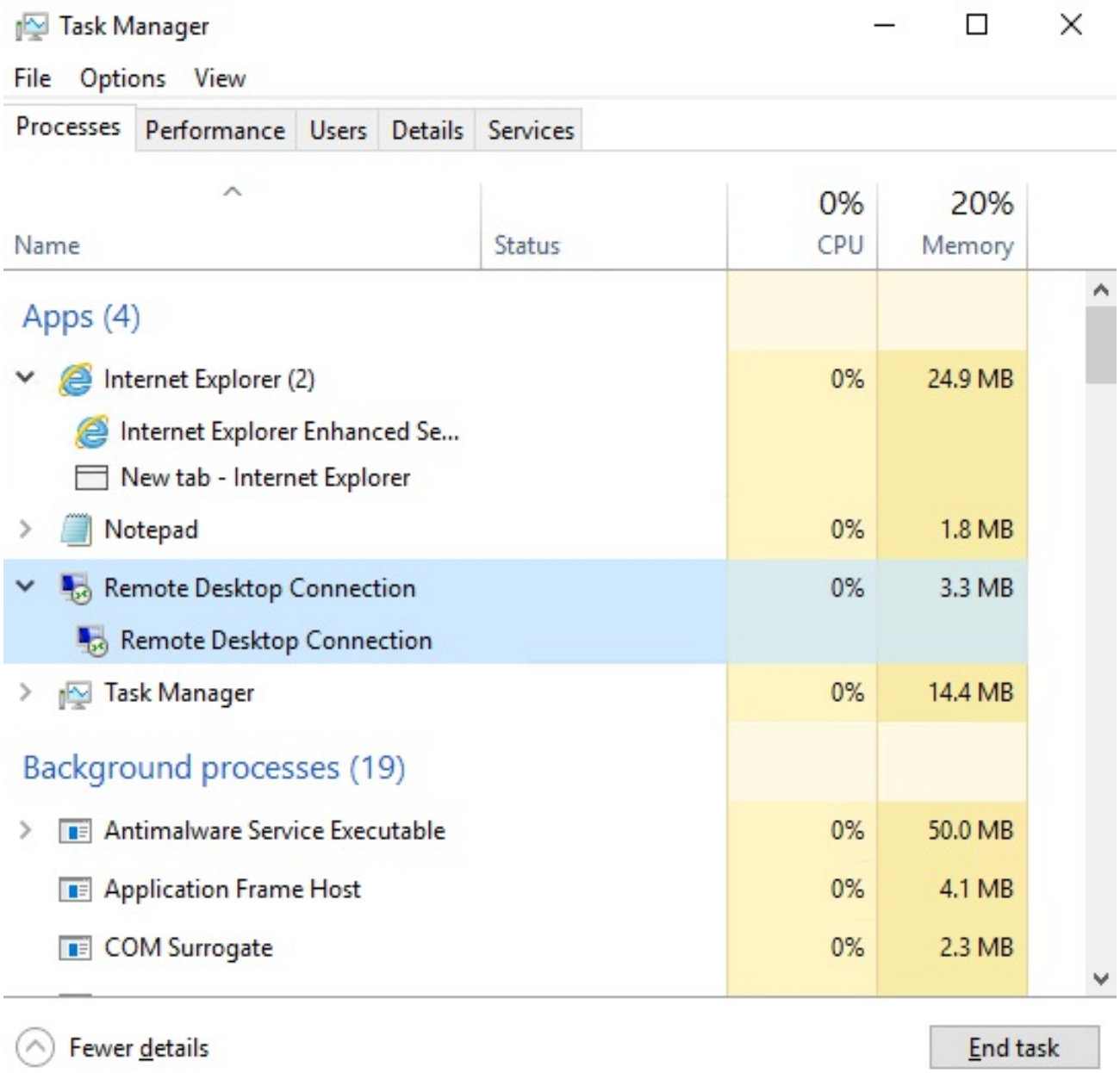
Task Manager is a tool that has existed in all Windows operating systems since the first days of the graphical interface, but it has evolved quite a bit over the years. One of the goals for Windows Server 2019 is to be even more useful and reliable than any previous version of Windows Server has been. So, it only makes sense that we finally remove Task Manager altogether, since it simply won't be needed anymore, right?

I'm kidding, of course! While Server 2019 will hopefully prove itself to indeed be the most stable and least needy operating system we have ever seen from Microsoft, Task Manager still exists and will still be needed by server administrators everywhere. If you haven't taken a close look at Task Manager in a while, it has changed significantly over the past few releases.

Task Manager is still typically invoked by either a *Ctrl* + *Alt* + *Del* on your keyboard then clicking on Task Manager, or by right-clicking on the taskbar and then choosing Task Manager. You can also launch Task Manager with the key combination *Ctrl* + *Shift* + *Esc*, or typing `taskmgr` inside the Run or Search dialog boxes. The first thing you'll notice is that very little information exists in this default view, only a simple list of applications that are currently running. This is a useful interface for forcing an application to close which may be hung up, but not for much else. Go ahead and click on the More details link, and you will start to see the real information provided in this powerful interface.

We immediately notice that the displayed information is more user-

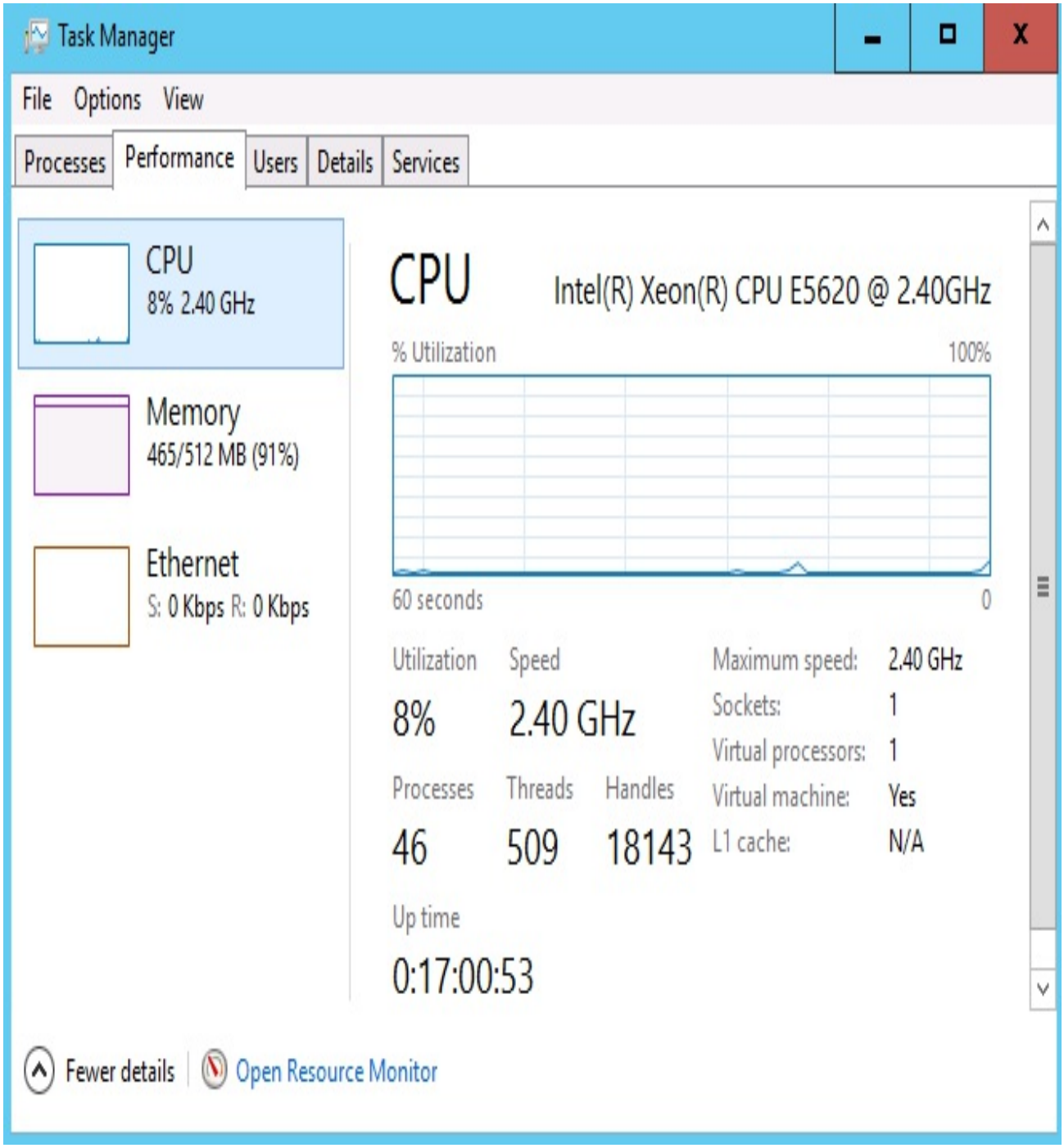
friendly than in previous years, with both Apps and Background processes being categorized in a more intuitive way, and multiple instances of the same application being condensed down for easy viewing. This gives a faster overhead view at what is going on with our system, while still giving the ability to expand each application or process to see what individual components or windows are running within the application, such as in the following screenshot:



Make sure to check out the other tabs available inside **Task Manager** as well. Users will show us a list of currently logged-in users and the amounts of hardware resources that their user sessions are consuming. This is a nice way to identify on a Remote Desktop Session Host server, for example, an individual who might be causing a slowdown on the server. The Details tab is a little bit more traditional view of the

Processes tab, splitting out much of the same information but in the older style way we were used to seeing in versions of the operating system long ago. Then, the Services tab is pretty self-explanatory; it shows you the Windows services currently installed on the server, their status, and the ability to start or stop these services as needed, without having to open the Services console separately.

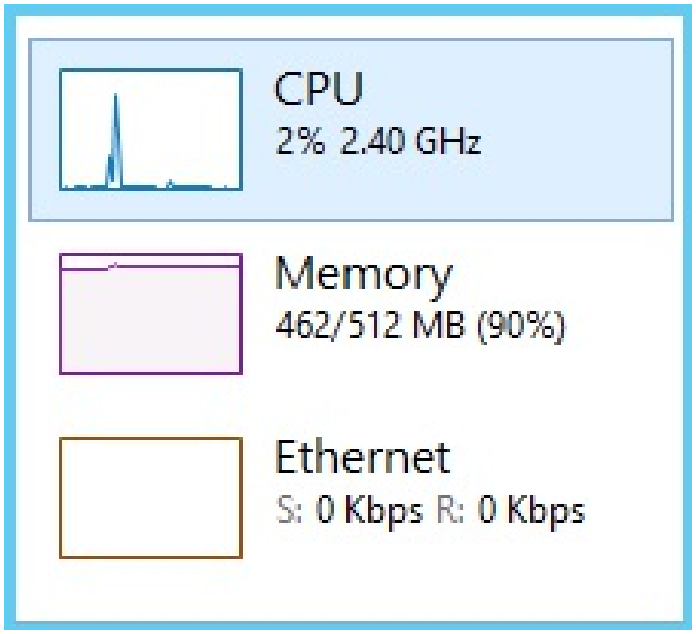
The tab that I skipped over so that I could mention it more specifically here is the Performance tab. This is a pretty powerful one. Inside, you can quickly monitor CPU, memory, and Ethernet utilization. As you can see in the following screenshot, I haven't done a very good job of planning resources on this particular virtual machine, as my CPU is hardly being touched but I am almost out of system memory:



Another useful piece of information available inside this screen is server uptime. Finding this information can be critical when troubleshooting an issue, and I watch admins time and time again calculating system uptime based on log timestamps. Using Task Manager is a much easier way to find that information!

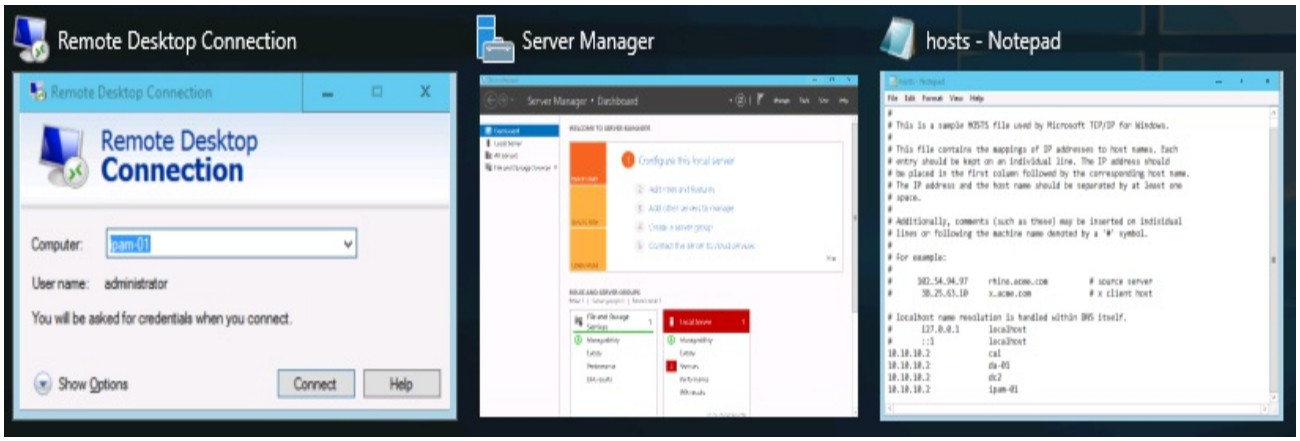
If you are interested in viewing more in-depth data about server performance, there is a link at the bottom of this Task Manager window where you can Open Resource Monitor. Two technologies provided inside Server 2019 for monitoring system status, particularly for hardware performance, are **Resource Monitor** and **Performance Monitor**. Definitely open up these tools and start testing them out, as they can provide both troubleshooting information and essential baseline data when you spin up a new server. This baseline can then be compared against future testing data so that you can monitor how new applications or services installed onto a particular server have affected their resource consumption.

Moving back to Task Manager, there is just one other little neat trick I would like to test. Still inside the Performance tab, go ahead and right-click on any particular piece of data that you are interested in. I will right-click on the CPU information near the left side of the window. This opens up a dialog box with a few options, of which I am going to click on Summary view. This condenses the data that was previously taking up about half of my screen real-estate, into a tiny little window, which I can move to the corner of my screen. This is a nice way to keep hardware utilization data on the screen at all times as you navigate through and work on your server so that you can watch for any spikes or increases in resource consumption when making changes to the system:


CPU
2% 2.40 GHz
Memory
462/512 MB (90%)
Ethernet
S: 0 Kbps R: 0 Kbps

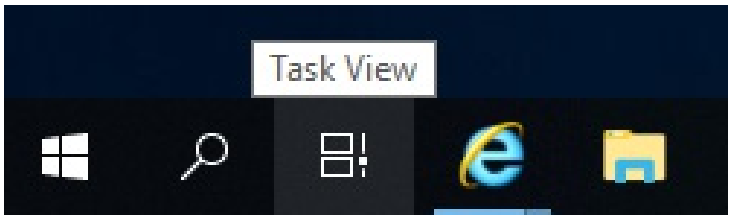
Task View

Task View is a new feature as of Windows 10 and Windows Server 2016, which carries over to Server 2019. It is a similar idea as that of holding down the *Alt* key and then pressing *Tab* in order to cycle through the applications that you currently have running. For anyone who has never tried that, go ahead and hold down those two keys on your keyboard right now. Depending on what version of Windows you are running, your screen might look slightly different than this, but, in effect, it's the same information. You can see all of the programs you currently have open, and you can cycle through them from left to right using additional presses of the *Tab* button. Alternatively, use *Alt* + *Shift* + *Tab* in order to cycle through them in reverse order. When you have many windows open, it is perhaps easier to simply use the mouse to jump to any specific window:



Task View is quite a bit more powerful than this, because it adds the capability of managing multiple full-desktops' worth of windows and applications. For example, if you were working on two different projects

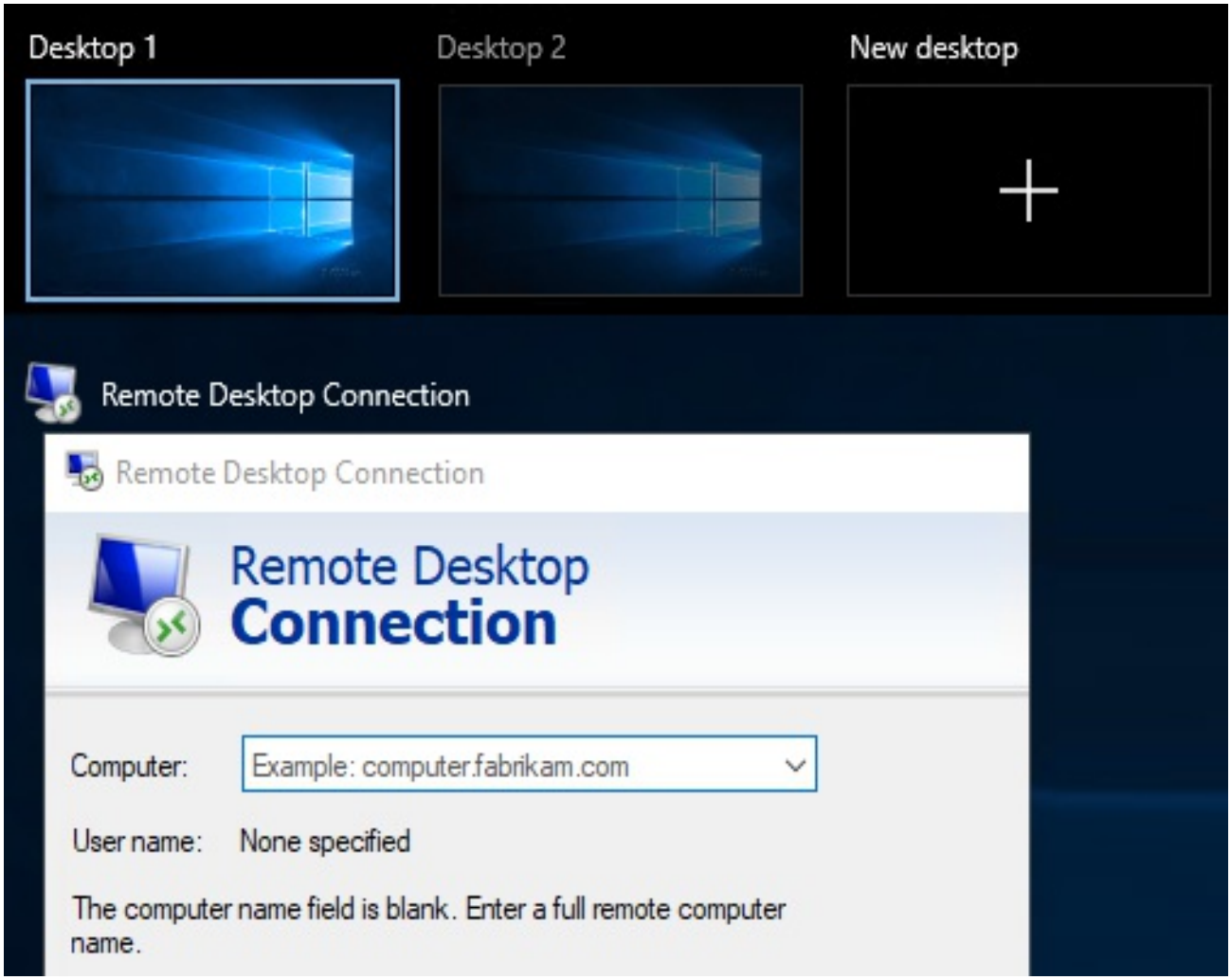
on the same server, and each project required you to have many different windows open at the same time, you would start to burn a lot of time switching back and forth between all of your different apps and windows in order to find what you were looking for. Using Task View, you could leave all of your open windows for the first project on your first desktop, and open all of the windows dealing with the second project on a second desktop. Then, with two clicks you can easily switch back and forth between the different desktops, using the Task View button. By default, Task View is the little button down in the taskbar, immediately to the right of the Search magnifying glass near the Start button. Go ahead and click on it now, it looks like this:



You now see a listing of your currently open windows; this looks very similar to the *Alt + Tab* functionality we looked at earlier. The difference is the little button near the top-left corner that says New desktop. Go ahead and click on that now:



Now, you will see Desktop 1 and Desktop 2 available for you to use. You can click on Desktop 2 and open some new programs, or you can even drag and drop existing windows between different desktops, right on this Task View screen:



Task View is a great way to stay organized and efficient by utilizing multiple desktops on the same server. I suppose it is kind of like running dual monitors, or three or four or more, all from a single physical monitor screen.

> *If you want to avoid having to click on the icon for Task View, pressing WinKey + Tab on your keyboard does the same thing!*

Summary

This first chapter on the new Windows Server 2019 is all about getting familiar and comfortable navigating around in the interface. There are various ways to interact with Server 2019 and we will discuss many of them throughout this book, but the majority of server administrators will be interfacing with this new operating system through the full graphical interface, using both mouse and keyboard to perform their tasks. If you have worked with previous versions of the Windows Server operating system, then a lot of the tools that you will use to drive this new platform will be the same, or at least similar, to the ones that you have used in the past. New operating systems should always be an evolution of their predecessors, and never all new. I think this was a lesson learned with the release of Windows 8 and Server 2012.

With Server 2019, we find a great compromise between the traditional familiarity of the prior versions of Windows, and the new benefits that come with rounded edges and touch-friendly screens that will be used more and more often as we move toward the future of Windows-based devices. In the next chapter, we will look into installing and managing the
Windows Server.

Questions

1. In Windows Server 2019, how can you launch an elevated PowerShell prompt with two mouse clicks?
2. What is the keyboard combination to open this Quick Admin Tasks menu?
3. What is the name of Microsoft's cloud service offering?
4. What are the two licensing versions of Windows Server 2019?

5. How many virtual machines can run on top of a Windows Server 2019 Standard host?
6. What installation option for Windows Server 2019 does not have a graphical user interface?
7. Which is the correct verbiage for the latest release of Windows Server 2019, Long-Term Servicing Branch (LTSB) or Long-Term Servicing Channel (LTSC)?
8. What is the correct tool from which to change configurations on a Windows Server 2019, Windows Settings or Control Panel?

Installing and Managing Windows Server 2019

Now that we have taken a look at some of the features inside the graphical interface of Windows Server 2019, I realize that some of you may be sitting back thinking *That's great to read about, but how do I really get started playing around with this for myself?* Reading about technology is never as good as experiencing it for yourself, so we want some rubber to meet the road in this chapter. One of the biggest goals of this book is to make sure we enable you to *use* the product. Rattling off facts about new features and efficiencies is fine and dandy, but ultimately worthless if you aren't able to make it work in real life. So, let's make this chunk of raw server metal do some work for us.

In this chapter, we will be covering the following:

- Requirements for installation
- Installing Windows Server 2019
- Installing roles and features
- Centralized management and monitoring
- **Windows Admin Center** (**WAC**)
- Enabling quick server rollouts with Sysprep

Technical requirements

When planning the build of a new server, many of the decisions that you need to make reflect licensing-type decisions. *What roles do you intend to install on this server? Can Server 2019 Standard handle it, or do we need Datacenter Edition for this guy? Is Server Core going to be beneficial from a security perspective, or do we need the full Desktop Experience?* In these days of Hyper-V servers with the ability to spin up virtual machines on a whim, we oftentimes proceed without much consideration of the hardware of a server, but there are certainly still instances where physical equipment will be hosting the Windows Server 2019 operating system. In these cases you need to be aware of the requirements for this new platform, so let us take a minute to list out those specifics. This information is available in longer form on the Microsoft Docs website if you need to double-check any specifics, but here are your summarized minimum system requirements (`https://docs.microsoft.com/en-us/windows-server/get-started-19/sys-reqs-19`):

- **CPU**: 1.4 GHz 64-bit that supports a number of things—NX, DEP, CMPXCHG16b, LAHF/SAHF, PrefetchW, and SLAT.
- **RAM**: 512 MB ECC memory minimum, or a recommended 2 GB minimum for a server running Desktop Experience. I can tell you that it is possible to install and run Desktop Experience with far fewer than 2 GB (such as inside a test lab), but you have to understand that performance will not be on par with what it could be.

- **Disk**: Server 2019 requires a **PCI Express** (**PCIe**) storage adapter. ATA/PATA/IDE are not allowed for boot drives. The minimum storage space requirement is 32 GB, but Desktop Experience consumes about 4 GB more space than Server Core, so take that into consideration.

Those are sort of the bare minimum specs, if you just want to spin up Server 2019 and poke around at it. For production systems, increase these numbers by a lot. There is no magic answer here, the specs you need depend on the workloads you expect to throw at your server. There are additional components that would be good to look for when building a new system that are required for particular roles and features as well. Things such as UEFI and a TPM chip are quickly becoming mainstream and used by more and more services with every operating system update. In particular, if you are interested in security and protection via BitLocker, or working with strong certificates or the new Shielded VMs, you will want to make sure that your systems include TPM 2.0 chips.

Installing Windows Server 2019

The installation process for Microsoft operating systems in general has improved dramatically over the past 15 years. I assume that a lot of you,, as IT professionals, are also the de facto *neighborhood computer guy*, being constantly asked by friends and family to fix or rebuild their computers. If you're anything like me, this means you are still occasionally rebuilding operating systems such as Windows XP. Looking at the bright blue setup screens and finding a keyboard with the *F8* key are imperative to this process. To spend 2 hours simply installing the base operating system and bringing it up to the highest service pack level is pretty normal. Compared to that timeline, installation of a modern operating system such as Windows Server 2019 is almost unbelievably fast and simple.

It is very likely that the majority of readers have completed this process numerous times already, and, if that is the case, feel free to skip ahead a couple of pages. But for anyone who is new to the Microsoft world, or new to IT in general, I'd like to take just a couple of quick pages to make sure you have a baseline to get started with. Without earning your **Installing an OS 101** badge on your tool belt, you will get nowhere in a hurry.

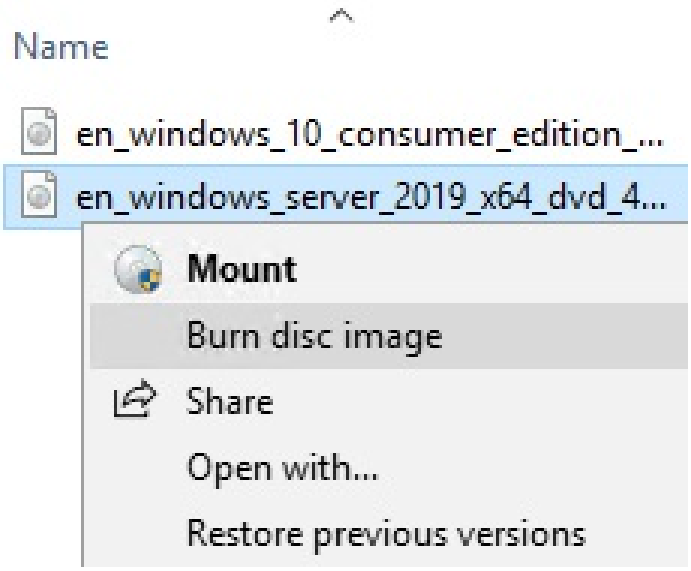
Burning that ISO

The first thing you must do is acquire some installation media. The most straightforward way to implement a single new server is to download an `.ISO` file from Microsoft, burn that `.ISO` to a DVD disc, and slide that DVD in to be used for installation. Since the website links and URLs are subject to change over time, the most trustworthy way to acquire your `.ISO` file to be used for installation is to open a search engine, such as **Bing**, and type `Download Windows Server 2019`. Once you have landed on the official Microsoft downloads page, click on the link to download your `.ISO` file and save it onto the hard drive of your computer.

The trickiest part of getting an `.ISO` file to be a workable DVD used to be the need for downloading some kind of third-party tool in order to burn it to a disc while making it bootable. If you are running an older client operating system on your computer, this may still be the case for you. I have watched many who are new to this process take the `.ISO` file, drag it over to their disc drive, and start burning the disc. This creates a DVD with the `.ISO` file on it, but that `.ISO` is still packaged up and not bootable in any way, so the disc would be worthless to your new piece of server hardware. Luckily, the newer versions of the Windows client operating systems have built-in functions for dealing with `.ISO` files that make the correct burning process very simple.

Once you have your `.ISO` file for the Windows Server 2019 installation downloaded onto your computer, insert a fresh DVD into your disc drive and browse to the new file. Simply right-click on the `.ISO` file, and then choose your menu option for Burn disc image. This launches a simple wizard that will extract and burn your new `.ISO` file the correct way onto

the DVD, making it a bootable installation media for your new server. This is shown in the following screenshot:



It is probable, if you attempt to download Windows Server 2019 and use this Windows Disc Image Burner utility with a DVD that you grabbed off your stack of standard blank DVDs, that you will receive the following error message: The disc image file is too large and will not fit on the recordable disc.

This should come as no surprise, because our operating system installer files have been getting larger and larger over the years. We have now reached the critical tipping point where the standard Server 2019 ISO installer is larger than a standard 4.7 GB DVD disc. In order to burn this ISO onto a DVD, you will need to hit the store and find some dual-layer discs that can handle more data.

Creating a bootable USB stick

DVDs can be cumbersome and annoying, and now they are also too small for our purposes. Therefore, when installing the newer, larger operating systems it is becoming commonplace to prep a USB stick to use for installation of the operating system, rather than relying on a DVD.

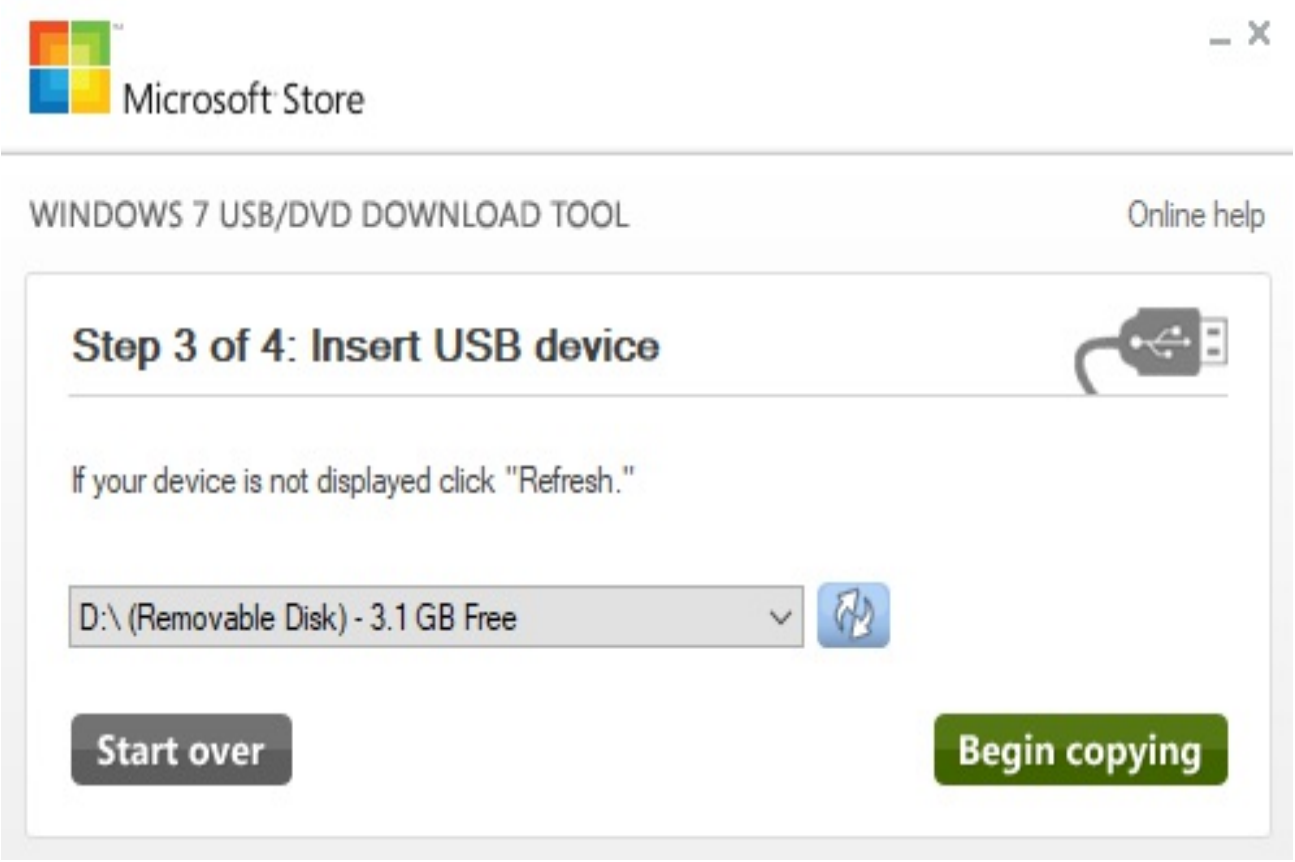
To do this, all you need is a Windows computer, a USB stick that is at least 8 GB, and access to the internet. You will need to download the same ISO that we discussed earlier, as that contains all of the installation files for Server 2019. Then you will also need to download and install some kind of bootable USB creation tool. There are various free ones available (Rufus is pretty popular), but the one straight from Microsoft is called the **Windows 7 USB/DVD Download Tool**. Why does it have this crazy name that includes the words *Windows 7* right in the name? Don't ask me. But, it works nonetheless and is a quick, easy, and a free way to prep your bootable USB sticks for fresh operating installations. I should point out that this tool has nothing to do with Windows 7. It will take any ISO file and turn it into a bootable USB stick. That ISO can definitely be a Windows 10 or Server 2019 ISO file and it still works just fine.

Once the USB DVD Download Tool is installed, launch the application and simply walk through the steps.

> *This process will erase and format your USB stick. Make sure nothing important is stored there!*

You will need to identify the ISO that you want the tool to grab

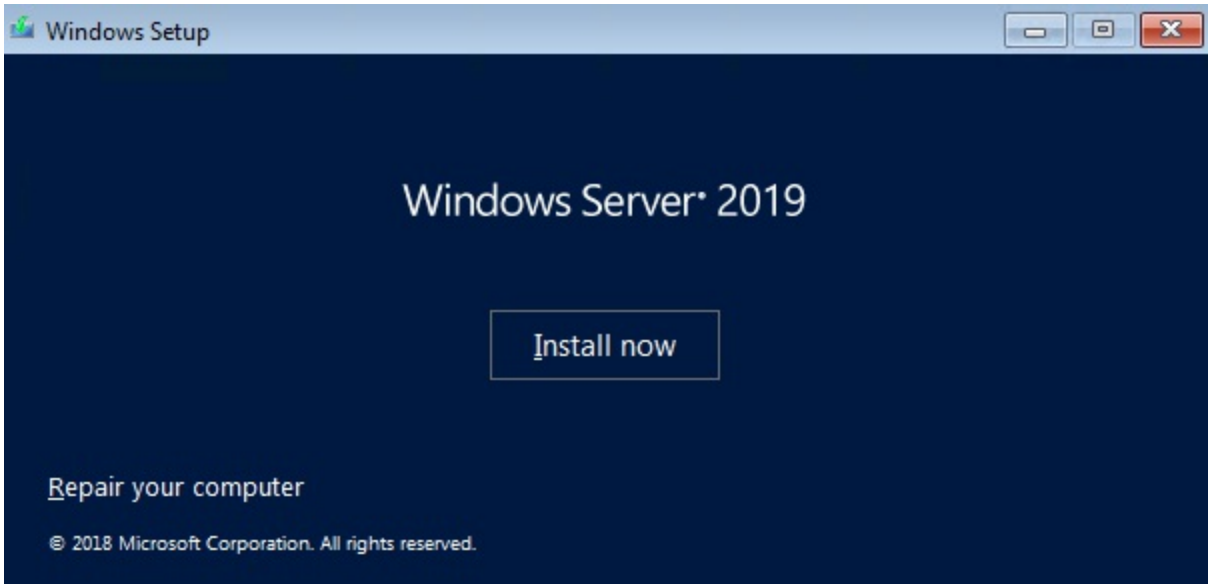
information from, then choose your USB stick from a drop-down list. After that, simply press the Begin copying button and this tool will turn your USB stick into a bootable stick capable of installing the entire Windows Server 2019 OS, as shown in the following screenshot:



Running the installer

Now go ahead and plug your newly-created DVD or bootable USB into the new server hardware. Boot to it, and you will finally see the installation wizard for Windows Server 2019. Now, there really are not that many options for you to choose from within these wizards, so we won't spend a lot of time here. For the most part, you are simply clicking on the Next button in order to progress through the screens, but there are a few specific places where you will need to make decisions along the way.

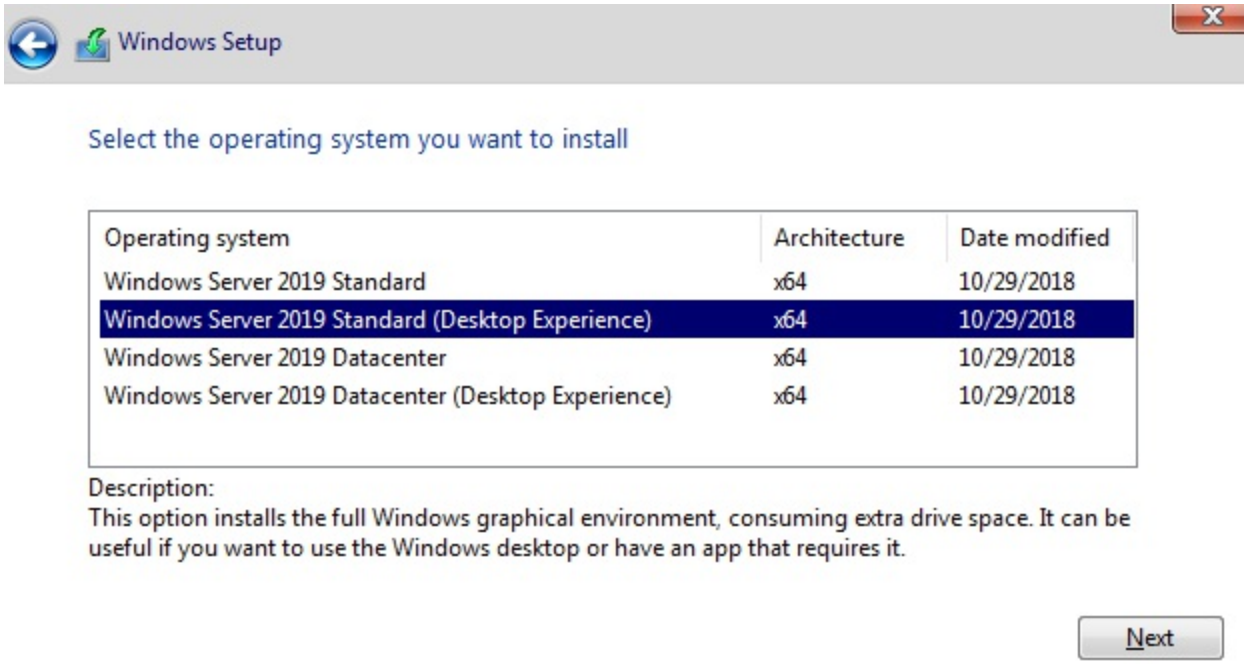
After choosing your installation language, the next screen seems pretty easy. There's just a single button that says Install now. Yes, that is what you want to click on, but I want you to notice the text in the lower-left corner of your screen. If you are ever in a position where you have a server that cannot boot and you are trying to run some recovery or diagnostic functions in order to resolve that issue, you can click on Repair your computer in order to launch into that recovery console. But for our fresh server installation, go ahead and click on Install now. This is shown in the following screenshot:



You will now be asked to input a product key in order to activate Windows. If you have your keys already available, go ahead and enter one now. Otherwise if you are simply installing this to test Server 2019 and want to run in trial mode for a while, you can click on the link that says I don't have a product key in order to bypass this screen.

The next screen is an interesting one, and the first place that you really need to start paying attention. You will see four different installation options for Windows Server 2019. There are what seem to be the *regular* installers for both Server 2019 Standard as well as Server 2019 Datacenter, and then a second option for each that includes the words **Desktop Experience**. Typically, in the Microsoft installer world, clicking on Next through every option gives you the most typical and common installation path for whatever it is that you are installing. *Not so with this wizard*. If you simply glide by this screen by clicking on Next, you will find yourself at the end with an installation of **Server Core**. We will talk more about Server Core in a later chapter of the book, but for now I will just say that if you are expecting to have a server that looks and feels like what we talked about in `Chapter 1`, *Getting Started with Windows Server 2019*, this default option is not going to

be the one that gets you there. The Desktop Experience that the wizard is talking about with the second option is the full Windows Server graphical interface, which you are more than likely expecting to see once we are done with our installation. So, for the purposes of our installation here, where we want to interact with the server using full color and our mouse, go ahead and decide whether you want the Standard or Datacenter edition, but make sure you choose the option that includes **Desktop Experience** before clicking on the Next button. This is shown in the following screenshot:



In some previous versions of Windows Server, we had the ability to migrate back and forth from a full Desktop Experience to Server Core and back again, even after the operating system was installed. This does not work in Windows Server 2019! The ability to transition between the two modes has disappeared, so it is even more important that you plan your servers properly from the beginning.

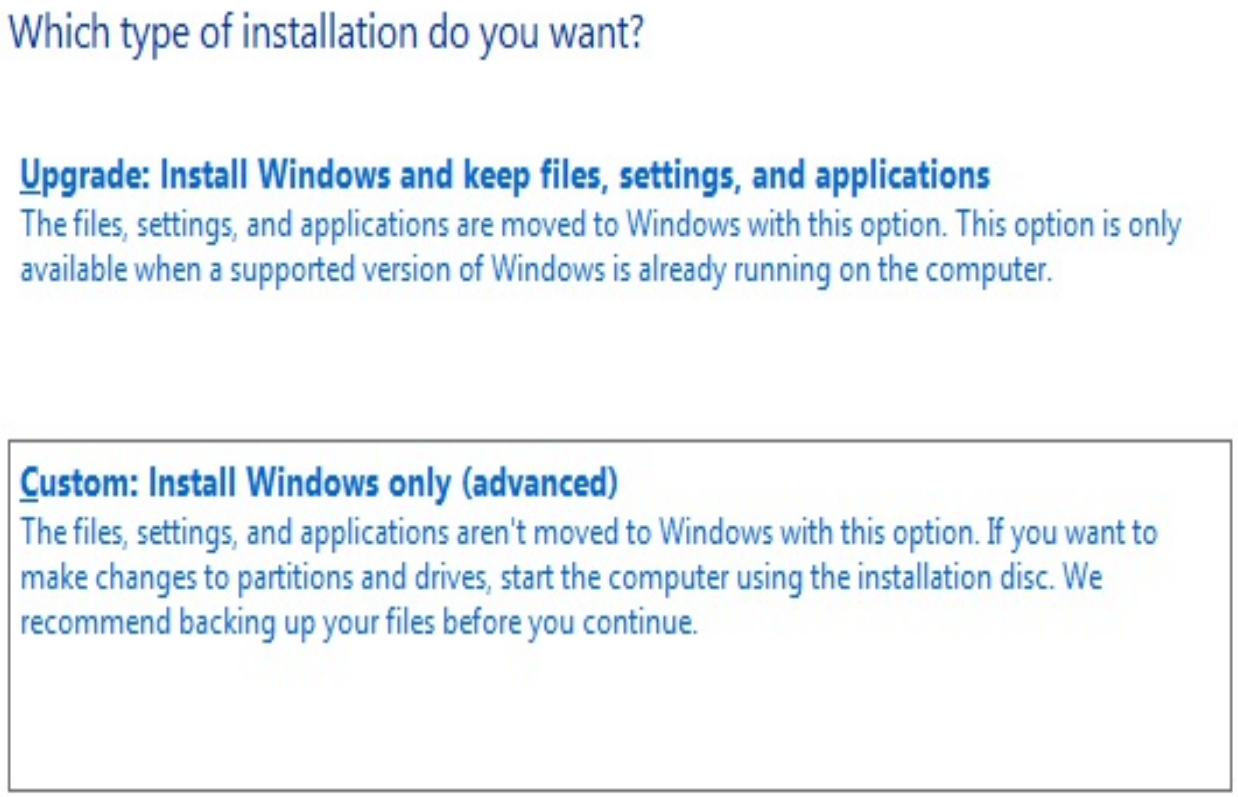
The next screen details licensing terms to which you need to agree, and then we come to another screen where the top option is most likely not the one that you intend to click on. I do understand why the Upgrade function is listed first for a consumer-class Windows 10 machine, but

nobody does in-place upgrades to Windows Servers. In a perfect world where everything always works flawlessly following upgrades, this would be a great way to go. You could have many servers all doing their jobs, and every time that a new operating system releases, you simply run the installer and upgrade them. Voila, magic! Unfortunately, it doesn't quite work like that, and I almost never see server administrators willing to take the risks in doing an in-place upgrade to an existing production server. It is much more common that we are always building brand new servers alongside the currently running production servers. Once the new server is configured and ready to accept its responsibilities, then, and only then, does the actual workload migrate over to the new server from the old one. In a planned, carefully sculpted migration process, once the migration of duties is finished, then the old server is shut down and taken away. If we were able to simply upgrade the existing servers to the newest operating system, it would save an awful lot of time and planning. But this is only feasible when you know that the upgrade is actually going to work without hiccups, and most of the time we are not prepared to take that risk. If an upgrade process goes sideways and you end up with a broken server, now you are looking at a costly repair and recovery process on a business-critical production server. You may very well be looking at working through the night or weekend as well. Would you rather spend your time planning a carefully-formed cutover, or recovering a critical server with the business breathing down your neck because they cannot work? My money's on the former.

> *Microsoft has announced that the Windows Server 2019 installer handles upgrades from Windows Server 2016 much better than any other Windows Server in-place-upgrade path in history. Upgrading from any earlier version of the operating system is still recommended to be a lift and shift, prepping a brand new server and moving the workload, but apparently they are now suggesting that people start testing in-place-upgrades from 2016 to 2019. Will that happen in the real world? I guess that's up to you...*

Given that, back to the topic at hand. In the Windows Server world, we

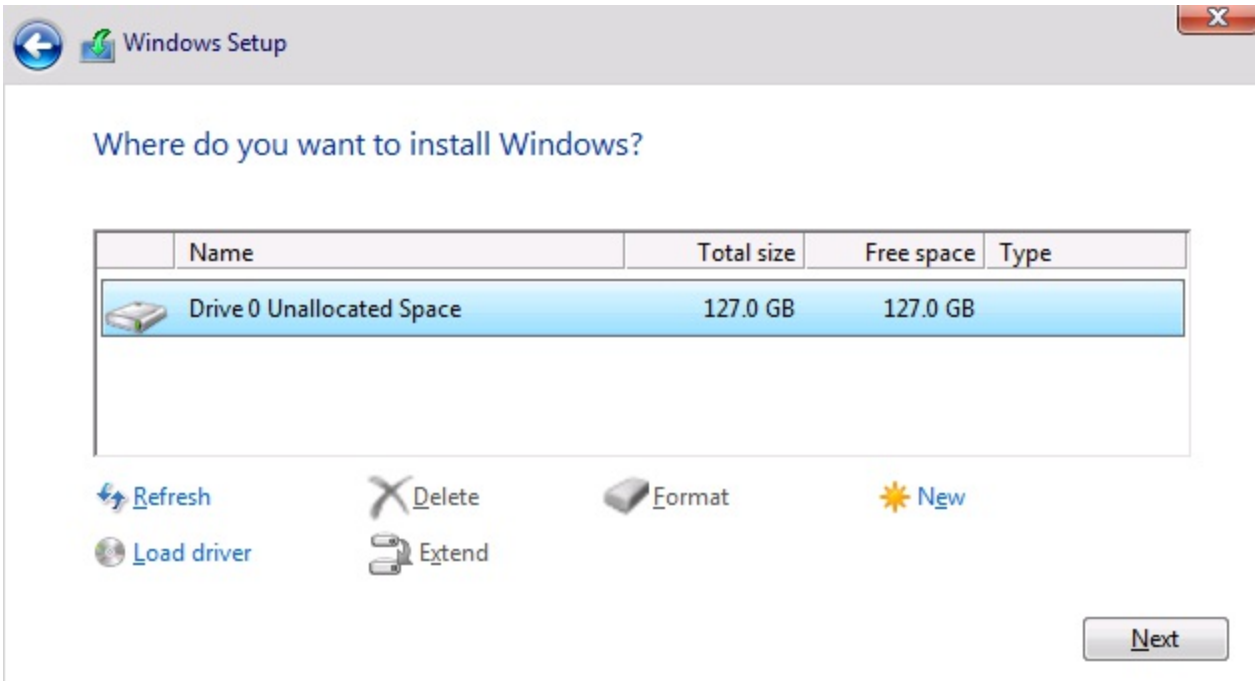
rarely touch the Upgrade option. So go ahead and choose the Custom: Install Windows only (advanced) option, which is where we will get into our options for installing this copy of Windows Server 2019 fresh into a new location on the hard drive. This is shown in the following screenshot:



Now we decide where we want to install our new copy of Windows Server 2019. In many cases, you will simply click on Next here, because your server will have just a single hard disk drive, or maybe a single RAID array of disks, and, in either case, you will see a single pool of free space onto which you can install the operating system. If you have multiple hard drives installed on your server and they have not been tied together in any way yet, then you will have multiple choices here of where to install Windows Server. We have just a single hard disk

attached here, which has never been used, so I can simply click on Next to continue. Note here that if your drives had existing or old data on them, you have the opportunity here, with some disk management tools, to format the disk, or delete individual partitions. If you are using some specialized disks that take specific drivers, there is also a Load driver button that you can use in order to inject these special drivers into the installation wizard in order to view these kinds of disks.

Also, it is important to note on this screen that there is no need to do anything here with most new server installations. You can see, in the following screenshot, that there is a New button that can be used to manually create hard disk partitions, and so many administrators assume they need to do that in order to install their new operating system.



This is not the case. There is no need to create partitions unless you want to set them up manually for some specific reason. If your hard drive is just a bunch of blank, unallocated spaces, all you need to do is

click on Next and the Windows Server 2019 installation will set up the partitions for you.

That's it! You will see the server installer start going to town copying files, installing features, and getting everything ready on the hard drive. This part of the installer runs on its own for a few minutes, and the next time you need to interact with the server it will be within the graphical interface where you get to define the administrator password. Once you have chosen a password, you will find yourself on the Windows desktop. Now you are really ready to start making use of your new Windows Server 2019.

Installing roles and features

Installing the operating system gets your foot in the door, so to speak, using your server as a server. However, you can't actually do anything useful with your server at this point. On a client desktop system, the base operating system is generally all that is needed in order to start working and consuming data. The server's job is to serve up that data in the first place, and, until you tell the server what its purpose is in life, there really isn't anything useful happening in that base operating system. This is where we need to utilize **roles** and **features**. Windows Server 2019 contains many different options for roles. A role is just what the name implies: the installation of a particular role onto a server defines that server's role in the network. In other words, a role gives a server some purpose in life. A feature, on the other hand, is more of a subset of functions that you can install onto a server. Features can complement particular roles, or stand on their own. There are pieces of technology available in Windows Server 2019 that are not installed or turned on by default, because these features wouldn't be used in all circumstances. Everything in the later chapters of this book revolves around the functionality provided by roles and features. They are the bread and butter of a Windows Server, and, without their installation, your servers make good paperweights, but not much else. As we will not be taking the time in each chapter to cover the installation of every particular role or feature that will be used within the chapter, let's take some time right now to cover the most common paths that admins can take in order to get these roles and features installed onto their own servers.

Installing a role using the wizard

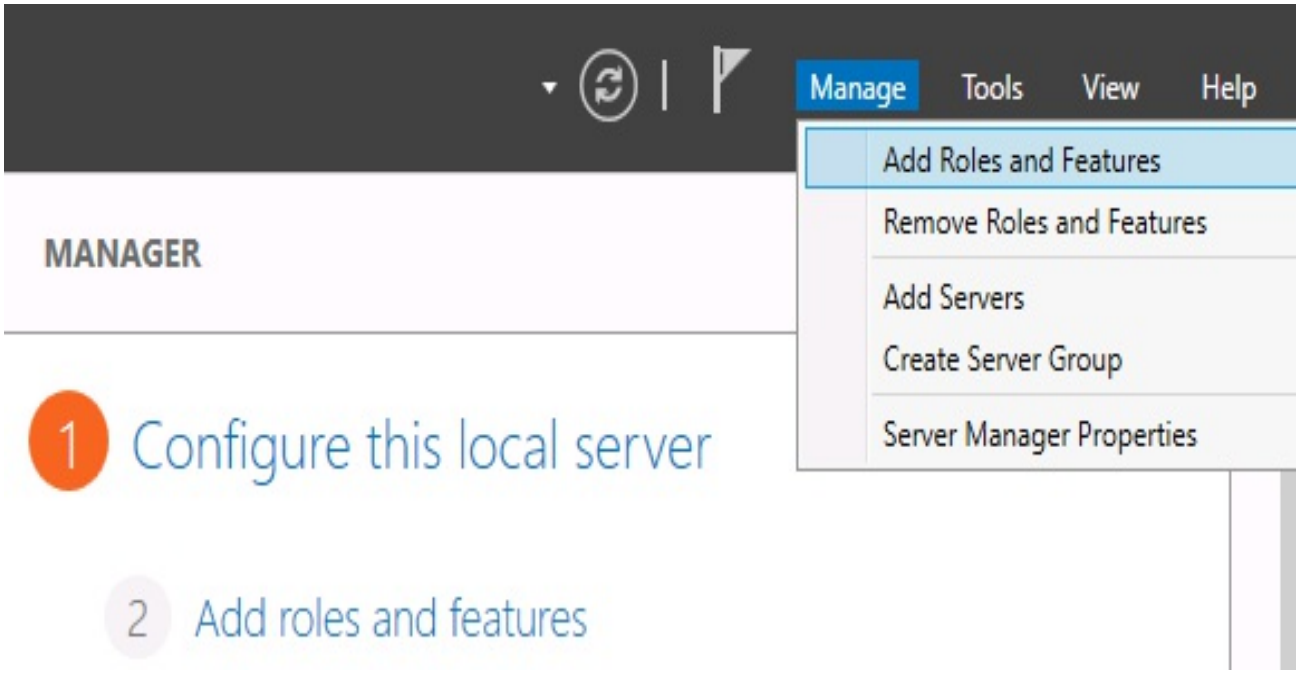
Without a doubt, the most common place that roles and features get installed is right inside the graphical wizards available as soon as your operating system has been installed. By default, a tool called **Server Manager** launches automatically every time you log into Windows Server 2019. We will take a closer look at Server Manager itself later in this chapter, but, for our purposes here, we will simply use it as a launching platform in order to get to our wizard which will guide us through the installation of our first role on this new server we are putting together.

Since you have just logged into this new server, you should be staring at the Server Manager Dashboard. Right in the middle of the Dashboard, you will see some links available to click on, a quick start list of action items numbered one through five. If you haven't already done so, put into place any local server configuration that you may need on this machine through the first link which is called Configure this local server.

Items that you will likely want in place are things such as a permanent hostname for the server, IP addressing, and, if you are joining this server to an existing domain, you typically handle that process prior to implementing any new roles on the server. But, in our case, we are more specifically interested in the role installation itself, so we will assume that you have already configured these little bits and pieces in order to have your server identified and routing on your network.

Go ahead and click on step 2, Add roles and features. Another way you

can launch the same wizard is by clicking on the Manage menu from the top bar inside Server Manager, and then choosing Add Roles and Features from the drop-down list. Selecting either link will bring you into our wizard for installation of the role itself. This is shown in the following screenshot:



You are first taken to a summary screen about installing roles. Go ahead and click on Next to bypass this screen. Now we get into our first option, which is an interesting one. We are first asked if we want to continue on with a Role-based or feature-based installation, which is exactly what we have been talking about doing. But the second option here, Remote Desktop Services installation, is important to note. Most of us consider the **Remote Desktop Services** (**RDS**) components of a Windows Server to be just another role that we can choose when setting up our server, similar to the installation of any other role. While that is basically true, it is important to note that RDS is so functionally different from the other kinds of roles that the entry path into installation of any of the RDS components invokes its own wizard, by

choosing the second option here. So, if you ever find yourself looking for the option to install RDS, and you have glazed over this screen because you are so used to clicking Next through it like I am, remember that you need to head back there in order to tell the wizard that you want to deal with an RDS component, and the remainder of the screens will adjust accordingly.

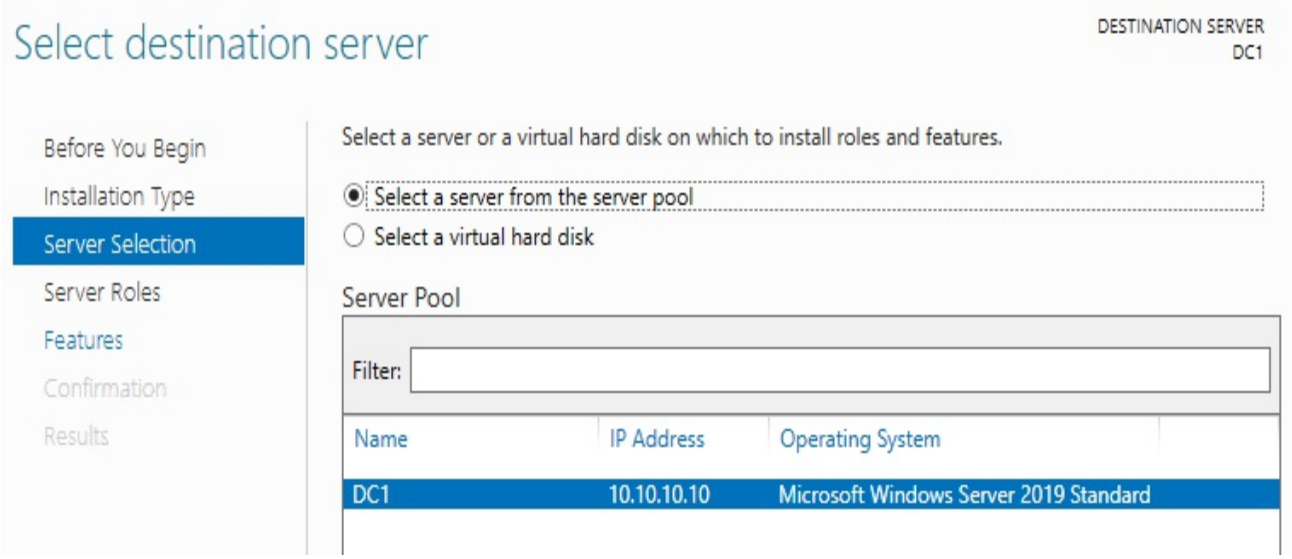
At the moment, I am working on building out a new test lab full of Windows Server 2019 boxes, and I am still in need of a Domain Controller to manage Active Directory in my environment. Prior to installing Active Directory on a server, it is critical that I have a few prerequisites in place, so I have already accomplished those items on my new server. The items that I need to have in place prior to the AD DS role installation are: having a static IP address assigned, and making sure that the DNS server setting in my NIC properties points somewhere, even if only to this server's own IP address. I also need to make sure that the hostname of my server is set to its final name, because once you turn it into a Domain Controller it is not supported to change the hostname. I have already accomplished these items on my server, so I will continue on through my role installation wizard here by leaving the option on for Role-based or feature-based installation, and clicking on Next, as shown in the following screenshot:



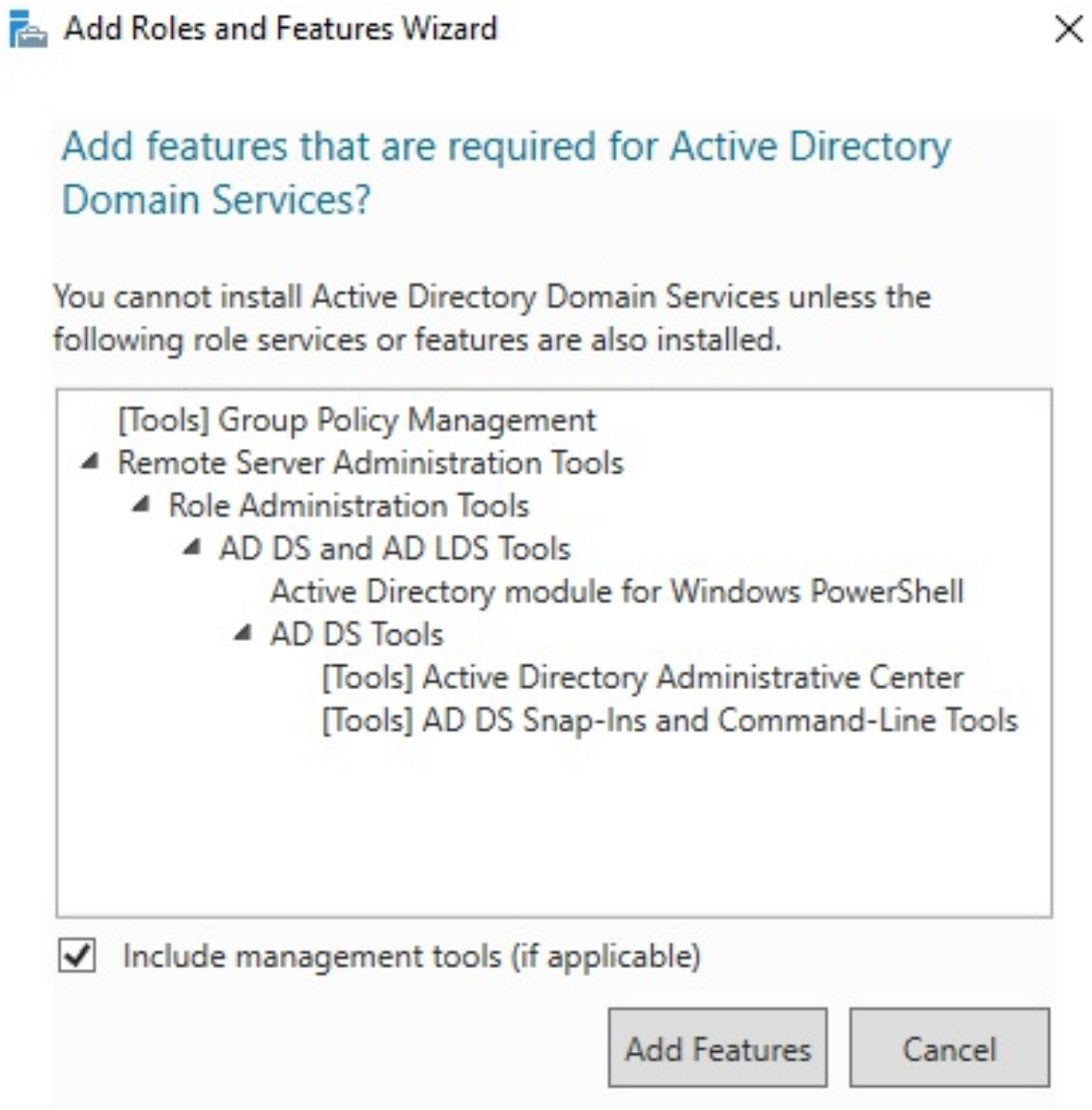
Our Server Selection screen is a very powerful one. If you've been through this process before, you have likely glazed over this screen, simply clicking on the Next button in order to progress through it. But,

essentially, what this screen is doing is asking you where you would like to install this new role or feature. By default, each server will only have itself listed on this screen, and so clicking on Next to continue is more than likely what you will be doing. But there are a couple of neat options here. First of all, if your Server Manager is aware of other servers in your network and has been configured to monitor them, you will have the option here to install a role or feature remotely onto one of the other servers. We will dig a little deeper in to this capability shortly. Another feature on this page, which I haven't seen many people utilize, is the ability to specify that you want to install a role or feature onto a virtual hard disk. Many of us work with a majority of virtual servers in this day and age, and you don't even need your virtual server to be running in order to install a role or feature to it! If you have access to the `.VHDX` file, the hard disk file, from where you are running Server Manager, you can choose this option that will allow you to inject the new role or feature directly into the hard drive. But, as is the case with 99% of the times that you will wander through this screen, we are logged directly into the server where we intend to install the role, and so we simply click on Next.

This is shown in the following screenshot:

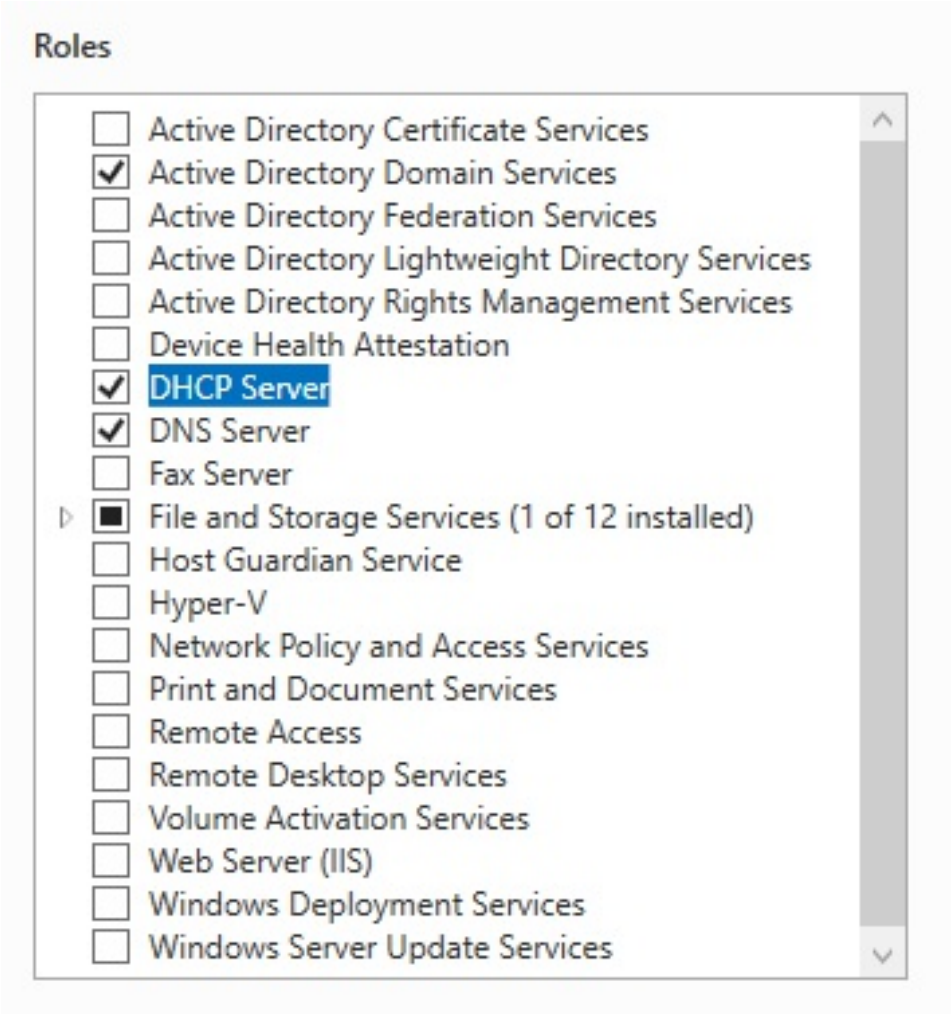


Now we have our list of roles that are available to be installed. Clicking on each role will give you a short description of the purpose of that role if you have any questions, and we will also be talking more about the core infrastructural pieces in our next chapter to give you even more information about what the roles do. All we need to do here in order to install a role onto our new server is check the box, and click on Next. Since this is going to be a Domain Controller, I will choose the Active Directory Domain Services role, and I will multipurpose this server to also be a DNS Server and a DHCP Server. With these roles, there is no need to re-run through this wizard three separate times in order to install all of these roles, I can simply check them all here and let the wizard run the installers together. Whoops, when I clicked on my first checkbox, I got a pop-up message that the Active Directory Domain Services role requires some additional features in order to work properly. This is normal behavior, and you will notice that many of the roles that you install will require some additional components or features to be installed. All you need to do is click on the Add Features button, and it will automatically add in these extra pieces for you during the installation process. An example of this is shown in the following screenshot:



Now that we have all three of our roles checked, it's time to click on Next. And, just to make it clear to all of you readers, I was not required to install all of these roles at the same time, they are not all dependent on each other. It is very common to see these roles all installed onto the same server, but I could split them up onto their own servers if I so desired. In a larger environment, you may have AD DS and DNS installed together, but you might choose to put the DHCP role onto its own servers, and that is just fine.

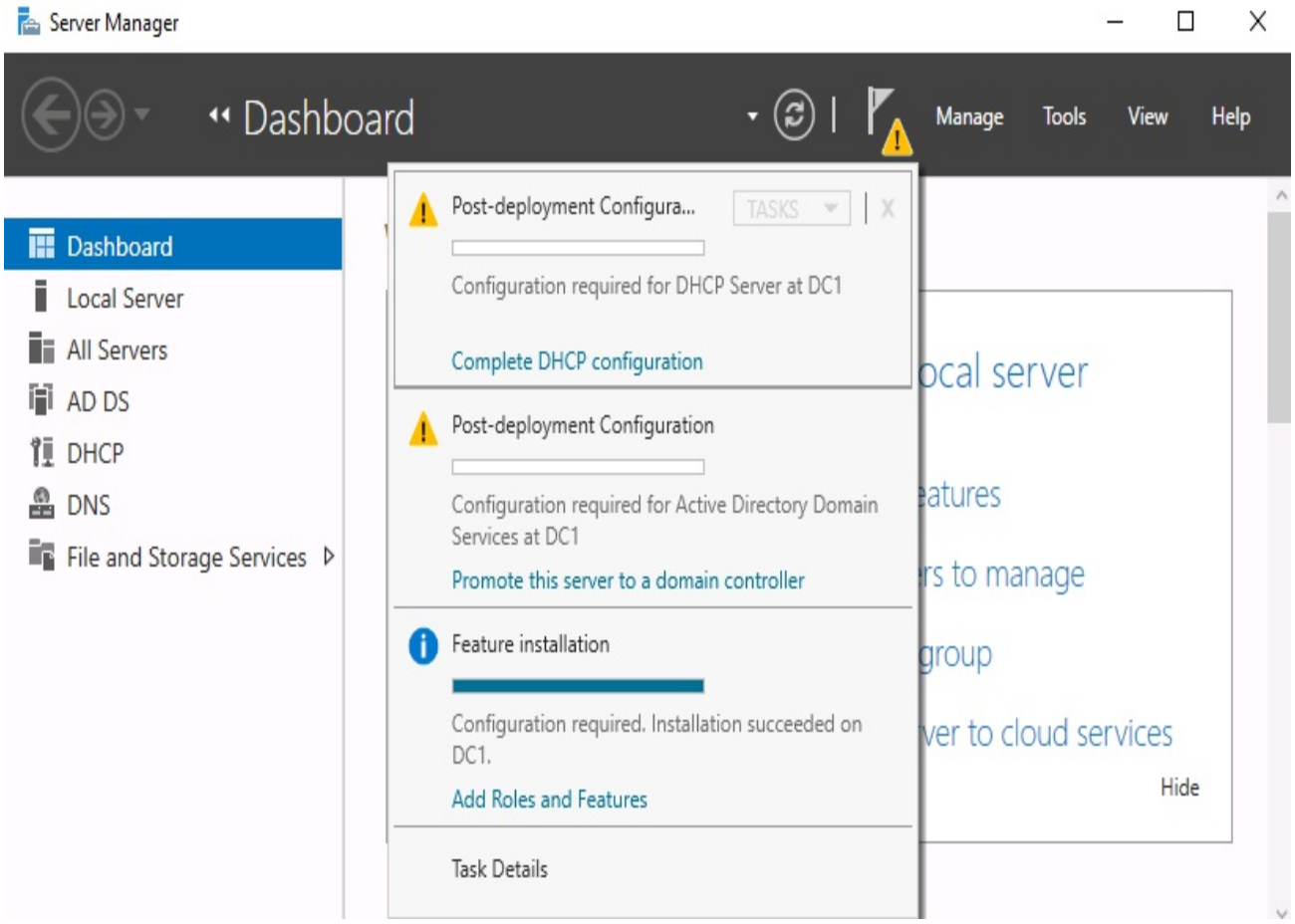
I am configuring this server to support a small lab environment, so, for me, it makes sense to put these core infrastructure services together in the same box, as shown in the following screenshot:



After clicking on Next, we have now landed on the page where we can install additional features to Windows Server 2019. In some cases, you may have originally intended only to add a particular feature, and in these cases, you would have bypassed the Server Roles screen altogether, and gone immediately to the Features installation screen. Just like with the role installation screen, go ahead and check off any features that you would like to install, and click on Next again. For our

new Domain Controller, we do not currently require any additional features to be specifically added, so I will just finish out the wizard which starts the installation of our new roles.

After the installation process has been completed, you may or may not be prompted to restart the server, depending on which roles or features you installed and whether or not they require a restart. Once you have landed back inside Server Manager, you will notice that you are now being prompted near the top with a yellow exclamation mark. Clicking here displays messages about further configurations that may be required in order to complete the setup of your new roles and make them live on the server. The roles for AD DS, DNS, and DHCP are now successfully installed, but there is some additional configuration that is now required for those roles to do their work. For example, in order to finish turning my server into a Domain Controller, I need to run through a promotion process to define my domain, or to specify a domain which I want to join. There are also some loose ends that I need to wrap before for putting DHCP into action:

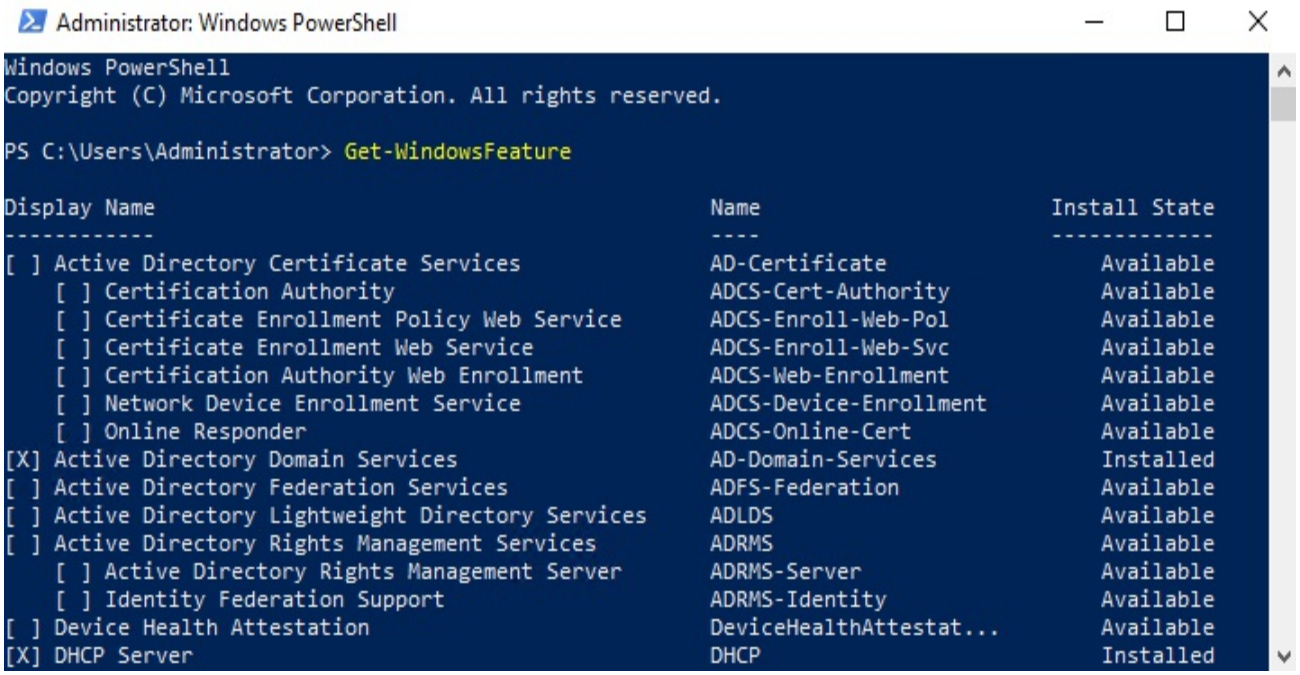

Server Manager
Dashboard
Manage
Tools
View
Help
Dashboard
Local Server
All Servers
AD DS
DHCP
DNS
File and Storage Services
Post-deployment Configura...
TASKS
Configuration required for DHCP Server at DC1
Complete DHCP configuration
Post-deployment Configuration
Configuration required for Active Directory Domain Services at DC1
Promote this server to a domain controller
Feature installation
Configuration required. Installation succeeded on DC1.
Add Roles and Features
Task Details
ocal server
atures
rs to manage
group
ver to cloud services
Hide

Installing a feature using PowerShell

Now that you have seen the graphical wizards for installing roles and features, you could certainly always use them in order to put these components into place on your servers. But Microsoft has put much effort into creating a Windows Server environment where almost anything within the operating system can be manipulated using PowerShell, and the addition of roles and features is certainly included in those capabilities. Let's take a look at the appropriate commands we can use to manipulate roles and features on our server right from a PowerShell prompt. We will view the available list of roles and features, and we will also issue a command to install a quick feature onto our server.

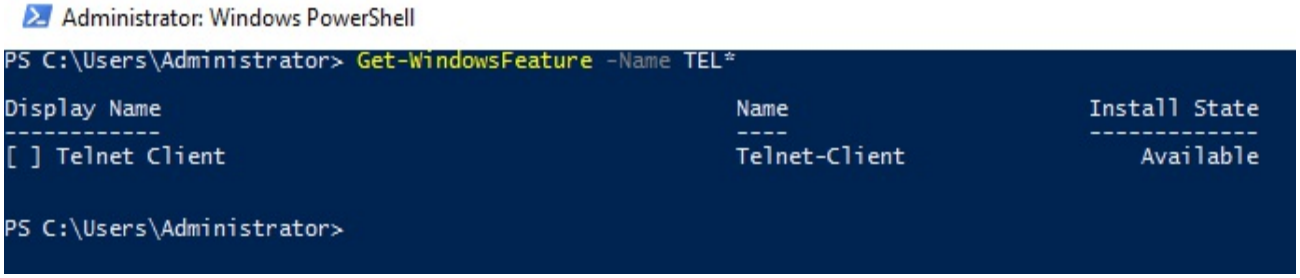
Open up an elevated PowerShell prompt, most easily accomplished via the quick admin tasks menu, accessed by right-clicking on the Start button. Then use the following command to view all of the available roles and features that we can install onto our server. It will also show you which ones are currently installed:

```
Get-WindowsFeature
```



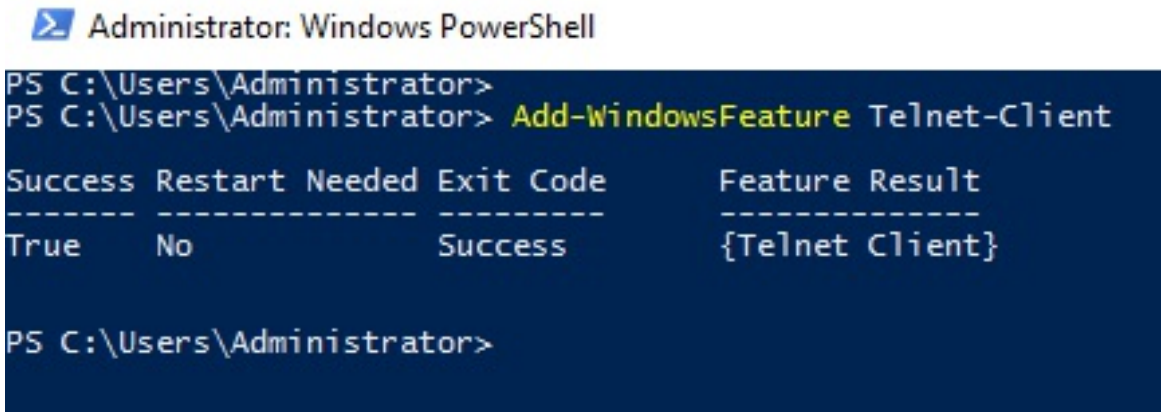
What I would like to do on this server is install the Telnet Client feature. I use Telnet Client pretty regularly for testing network connections, so it is helpful to have on this machine. Unfortunately, my PowerShell window currently has pages and pages of different roles and features in it, and I'm not sure what the exact name of the Telnet Client feature is in order to install it. So, let's run `Get-WindowsFeature` again, but, this time, let's use some additional syntax in the command to pare down the amount of information being displayed. I want to see only the features which begin with the letters `TEL`, as shown in the following examples:

```
Get-WindowsFeature -Name TEL*
```

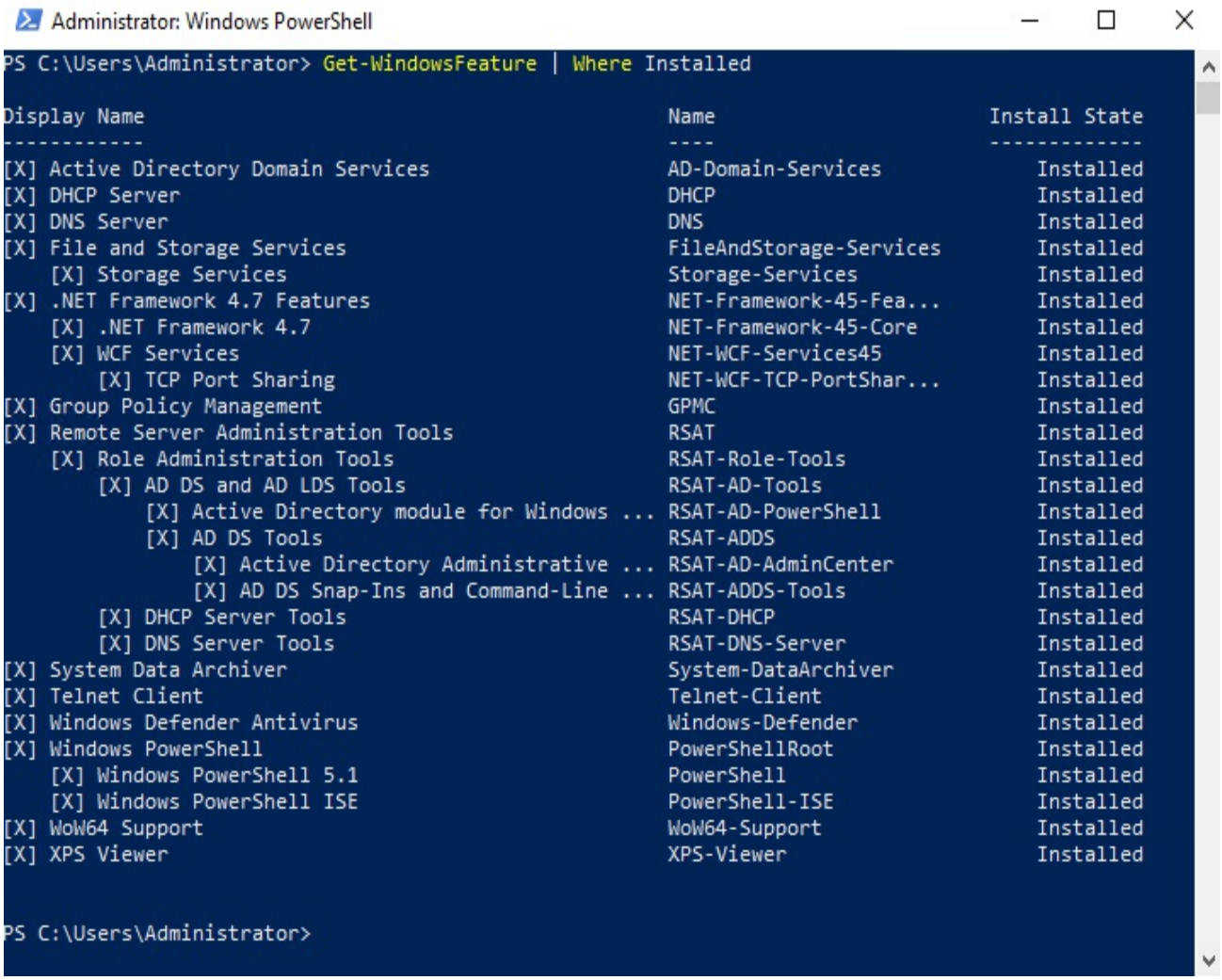


There it is! Okay, so now that I know the correct name of the feature, let's run the command to install it, as shown in the following examples:

```
Add-WindowsFeature Telnet-Client
```



One last thing to show you here, there is also a way to manipulate the `Get-WindowsFeature` cmdlet in order to quickly show only the roles and features currently installed on a server. Typing `Get-WindowsFeature | Where Installed` presents us with a list of the currently installed components. If I run that on my Domain Controller, you can see all the parts and pieces of my roles for AD DS, DNS, and DHCP, as shown in the following screenshot:


Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-WindowsFeature | Where Installed
Display Name Name Install State
[X] Active Directory Domain Services AD-Domain-Services Installed
[X] DHCP Server DHCP Installed
[X] DNS Server DNS Installed
[X] File and Storage Services FileAndStorage-Services Installed
[X] Storage Services Storage-Services Installed
[X] .NET Framework 4.7 Features NET-Framework-45-Fea... Installed
[X] .NET Framework 4.7 NET-Framework-45-Core Installed
[X] WCF Services NET-WCF-Services45 Installed
[X] TCP Port Sharing NET-WCF-TCP-PortShar... Installed
[X] Group Policy Management GPMC Installed
[X] Remote Server Administration Tools RSAT Installed
[X] Role Administration Tools RSAT-Role-Tools Installed
[X] AD DS and AD LDS Tools RSAT-AD-Tools Installed
[X] Active Directory module for Windows ... RSAT-AD-PowerShell Installed
[X] AD DS Tools RSAT-ADDS Installed
[X] Active Directory Administrative ... RSAT-AD-AdminCenter Installed
[X] AD DS Snap-Ins and Command-Line ... RSAT-ADDS-Tools Installed
[X] DHCP Server Tools RSAT-DHCP Installed
[X] DNS Server Tools RSAT-DNS-Server Installed
[X] System Data Archiver System-DataArchiver Installed
[X] Telnet Client Telnet-Client Installed
[X] Windows Defender Antivirus Windows-Defender Installed
[X] Windows PowerShell PowerShellRoot Installed
[X] Windows PowerShell 5.1 PowerShell Installed
[X] Windows PowerShell ISE PowerShell-ISE Installed
[X] WoW64 Support WoW64-Support Installed
[X] XPS Viewer XPS-Viewer Installed
PS C:\Users\Administrator>

Centralized management and monitoring

Whether you are installing new roles, running backups and maintenance programs, or troubleshooting and repairing a server, it makes common sense that you would log into the specific server that you will be working on. Long ago this meant walking up to the server itself and logging on with the keyboard and mouse which were plugged right into that hardware. Then, quite a number of years ago, this became cumbersome and technology advanced to the point where we now had the **Remote Desktop Protocol** (**RDP**) available to us. We quickly transitioned over to log into our servers remotely using RDP. Even though it's been around for many years, RDP is still an incredibly powerful and secure protocol, giving us the ability to quickly connect to servers from the comfort of our desk. And, as long as you have proper network topology and routing in place, you can work on a server halfway around the world just as quickly as one sitting in the cubicle next to you. In fact, I recently read that mining rights were being granted in outer space. Talk about a co-location for your datacenter! Maybe someday we will be using RDP to connect to servers in outer space. While this might be a stretch in our lifetimes, I do have the opportunity to work with dozens of new companies every year, and, while there are some other tools available for remotely managing your server infrastructure, RDP is the platform of choice for 99% of us out there.

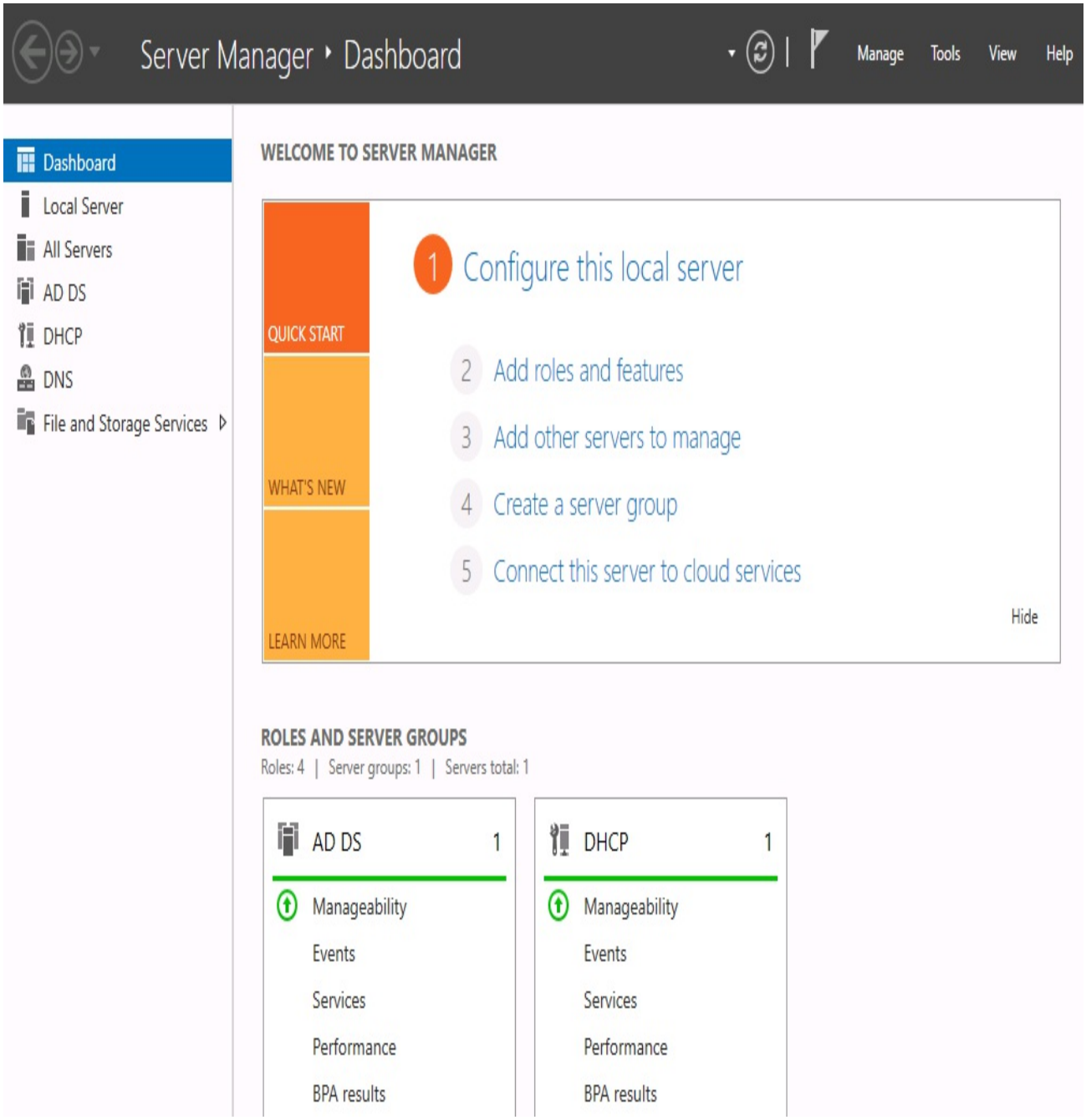
Why talk about RDP? Because I will now tell you that Windows Server 2019 includes some tools which make it much less necessary to our day-to-day workflow. The idea of centralized management in the server

world has been growing through the last few Windows Server operating system rollouts. Most of us have so many servers running that checking in with them all daily would consume way too much time. We need some tools that we can utilize to make our management and monitoring, and even configuration processes, more efficient in order to free up time for more important projects.

Server Manager

If you have worked on a Windows Server recently, you are familiar with the idea that logging into one of your servers automatically invokes this big window on top of the desktop. This auto-launching program is **Server Manager**. As the name implies, it's here to help you manage your server. However, in my experience, the majority of server administrators do not utilize Server Manager. Instead, they close it as fast as they can and curse at it under their breath, because it's been popping up and annoying them during every server login for the past 10 years.

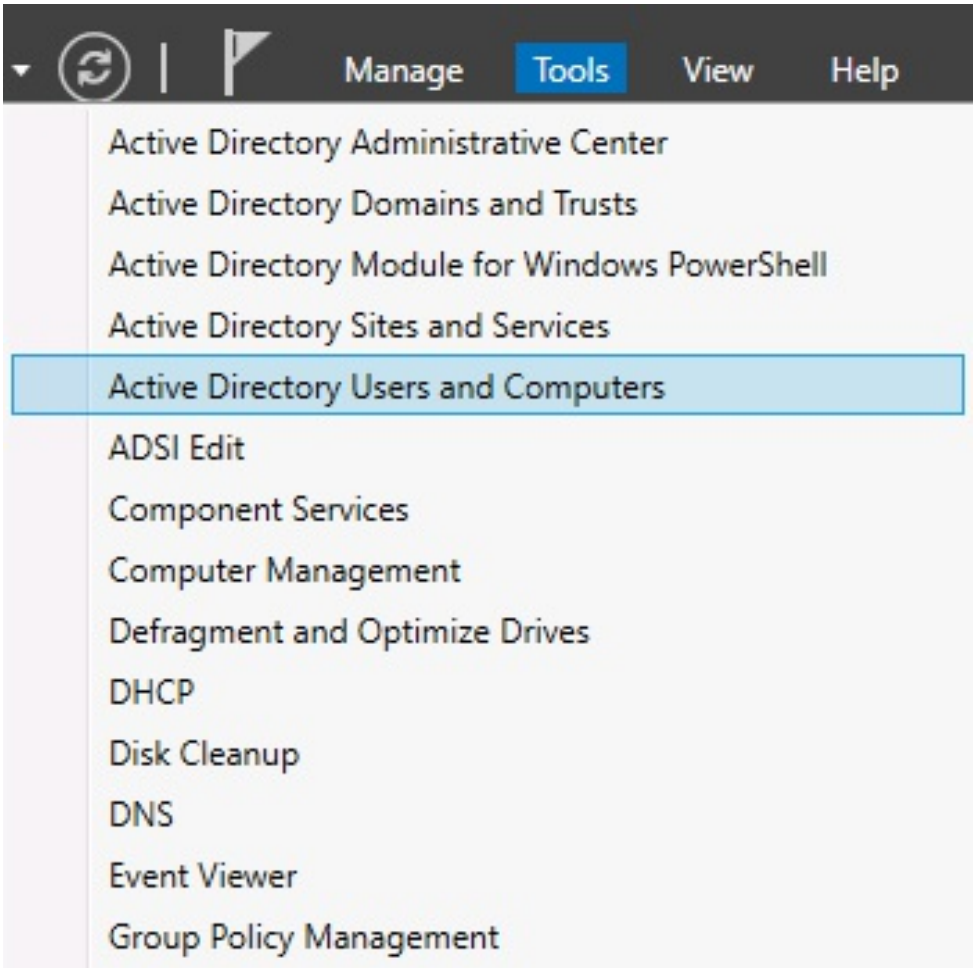
Stop doing that! It's here to help, I promise. The following screenshot is of the default view of Server Manager on my new Domain Controller:



What I like about this opening automatically is that it gives me a quick look into what is currently installed on the server. Looking at the column on the left side shows you the list of roles installed and available for management. Clicking on each of these roles brings you into some more particular configuration and options for the role itself. I

often find myself hopping back and forth between many different servers while working on a project, and by leaving Server Manager open it gives me a quick way of double-checking that I am working on the correct server. The Roles and Server Groups section at the bottom is also very interesting. You might not be able to see the colors in the picture, but this gives you a very quick view into whether or not the services running on this server are functioning properly. Right now, both my AD DS and DHCP functions are running normally, I have a nice green bar running through them. But, if anything was amiss with either of these roles, it would be flagged bright red, and I could click on any of the links listed under those role headings in order to track down what the trouble is.

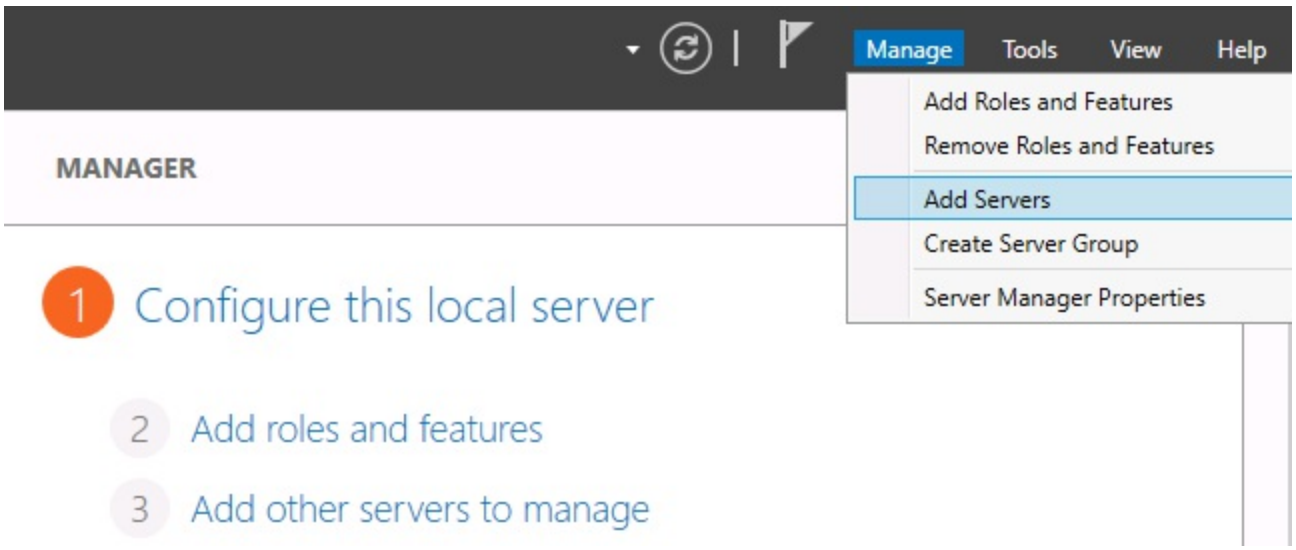
Up near the top-right corner you see a few menus, the most useful of which, to me, is the Tools menu. Click on that, and you see a list of all the available **Administrative Tools** to launch on this server. Yes, this is essentially the same Administrative Tools folder that has existed in each of the previous versions of Windows Server, now stored in a different location. Based on my experience, Server Manager is now the easiest way to access this myriad of tools all from a single location:



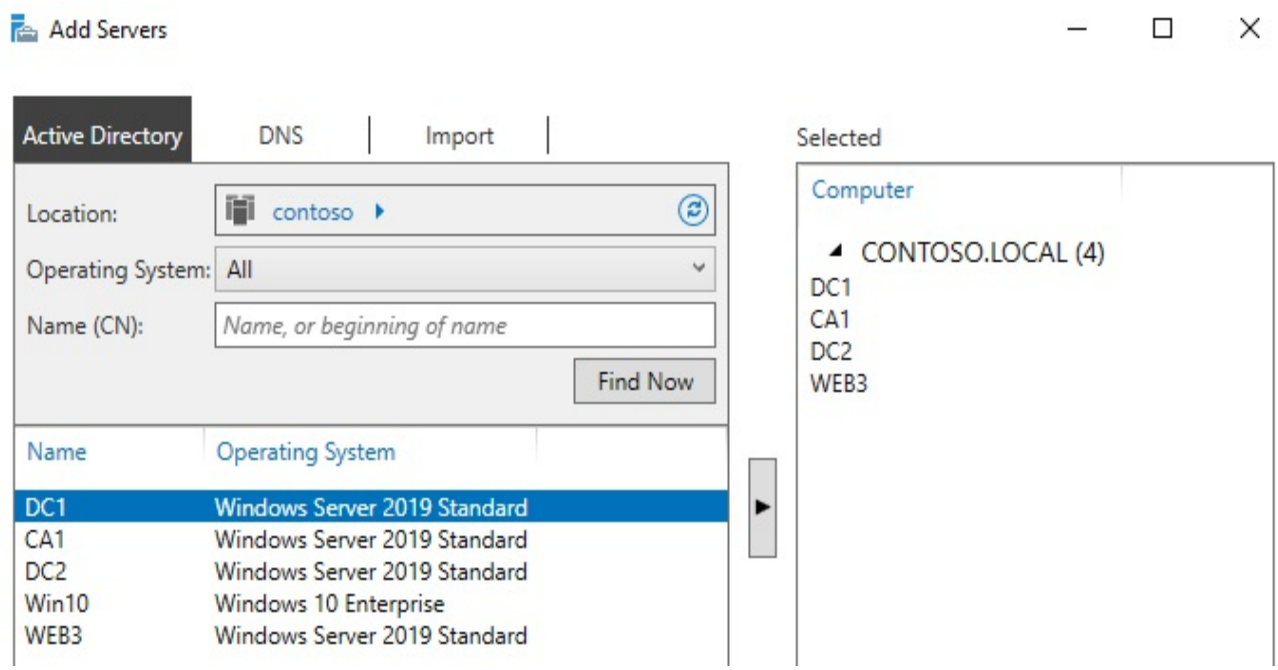
So far the functions inside Server Manager that we have discussed are available on any Windows Server 2019, whether it is standalone or part of a domain. Everything we have been doing is only dealing with the local server that we are logged into. Now, let's explore what options are available to us in Server Manager for centralization of management across multiple servers. The new mentality of managing many servers from a single server is often referred to as *managing from a single pane of glass*. We will use Server Manager on one of our servers in the network in order to make connections to additional servers, and after doing that we should have much more information inside Server Manager that we can use to keep tabs on all of those servers.

Front and center inside the **Server Manager** console is the section entitled Welcome to Server Manager. Under that we have a series of steps or links that can be clicked on. The first one lets you configure settings that are specific only to this local server. We have already done some work with the second step when we added a new role to our server. Now we will test out the third step, Add other servers to manage.

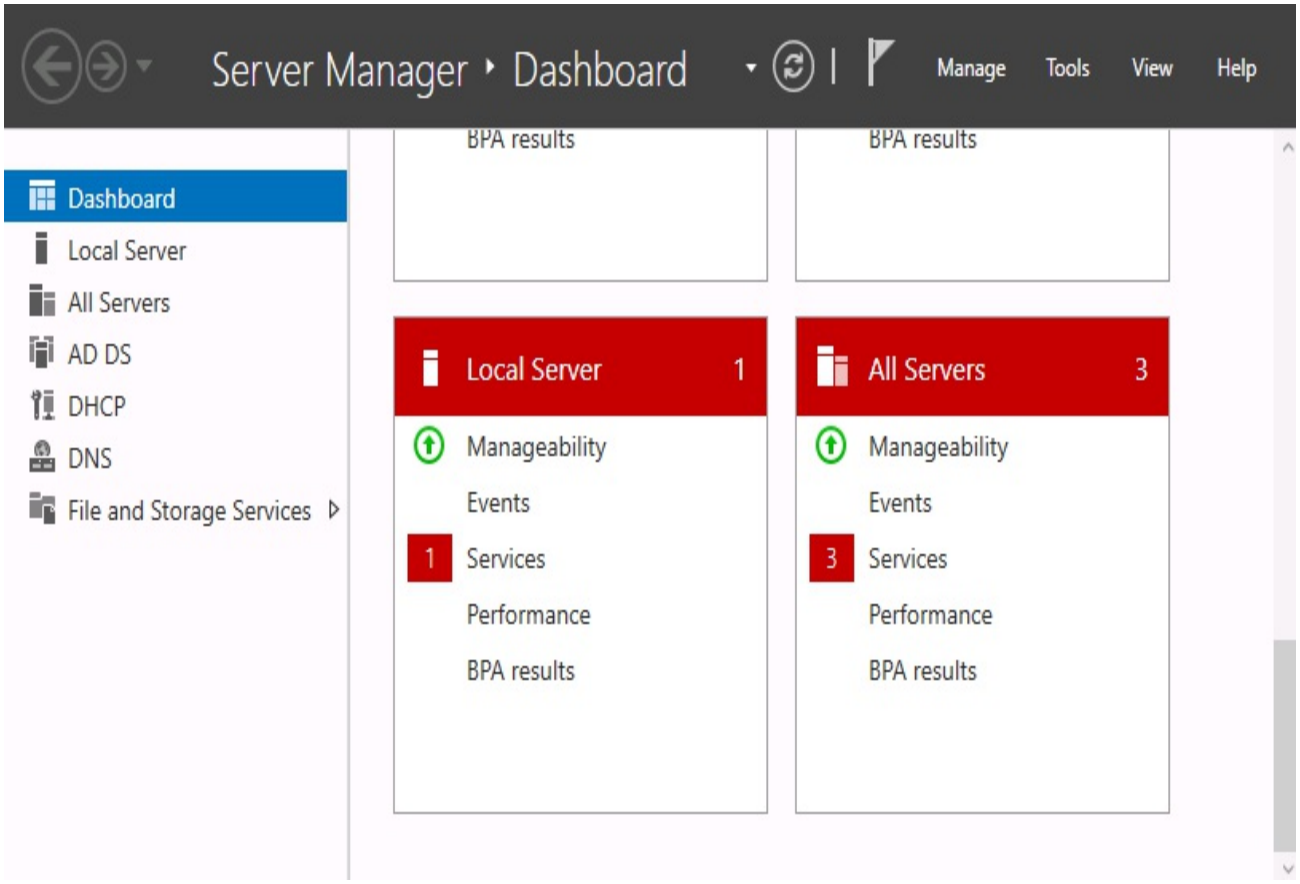
By the way, this same function can also be called by clicking on the Manage menu at the top, and then choosing Add Servers. This is shown in the following screenshot:



Most of you will be working within a domain environment where the servers are all domain joined, which makes this next part really easy. Simply click on the Find Now button, and the machines available within your network will be displayed. From here, you can choose the servers that you want to manage, and move them over to the Selected column on the right, as shown in the following screenshot:



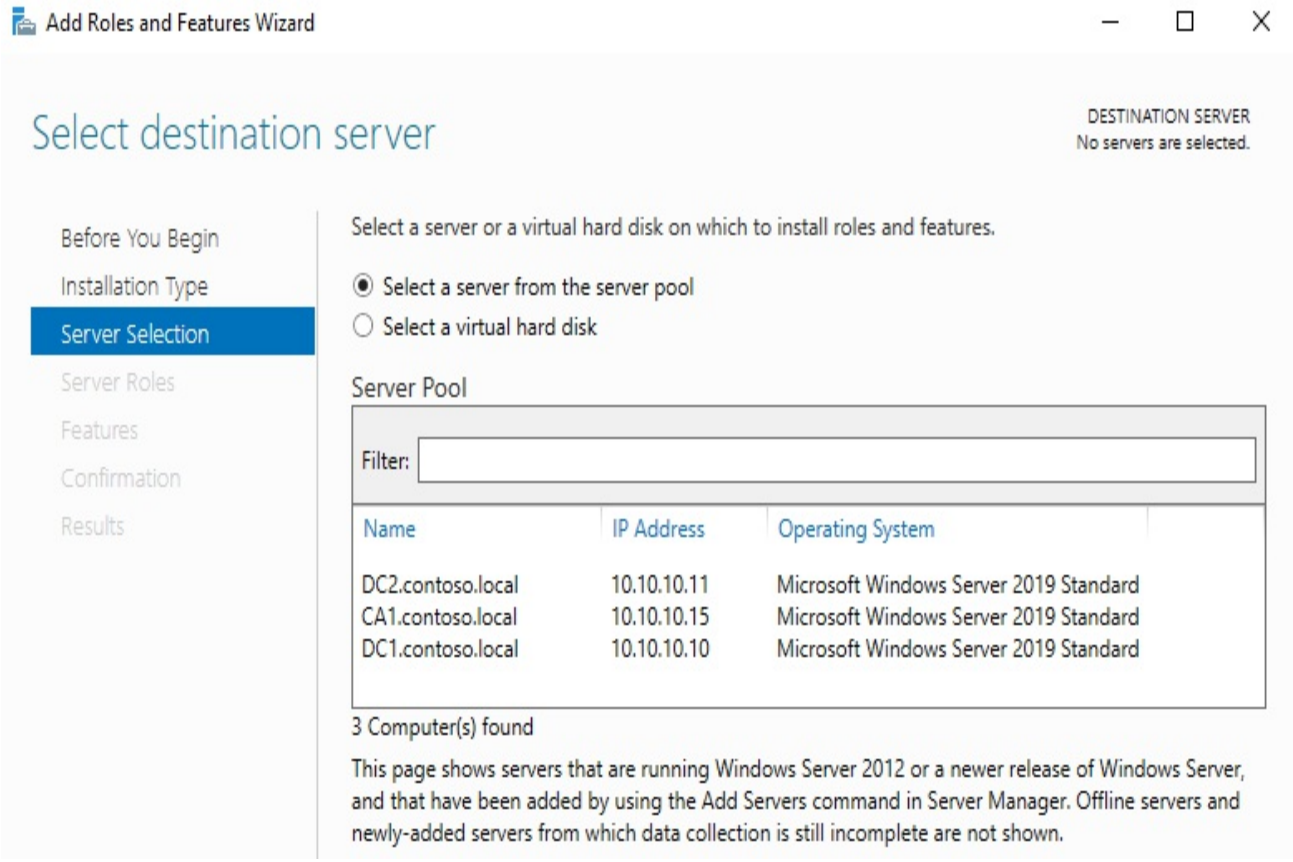
After clicking OK, you will see that Server Manager has transformed in order to give you more information about all of these servers and roles that are installed on them. Now, when you log into this single server, you immediately see critical maintenance information about all of the systems that you have chosen to add in here. You could even use a separate server, which is only intended for the purposes of this management. For example, I am currently logged into a brand new server called **CA1**. I do not have any roles installed onto this server, so, by default, Server Manager looks pretty basic. As soon as I add other servers (my Domain Controllers) to be managed, my Server Manager on the CA1 server now contains all of the detail about CA1 and my Domain Controllers, so I can view all facets of my infrastructure from this single pane. As you can see in the following screenshot, I even have some flags here indicating that some services are not running properly within my infrastructure:



Clicking on the All Servers link, or into one of the specific roles, gives you even more comprehensive information collected from these remote servers. Adding multiple servers into Server Manager is not only useful for monitoring, but for future configurations as well. You remember a few pages ago when we added a new role using the wizard? That process has now evolved to become more comprehensive, since we have now *tapped* this server into our other servers in the network.

If I now choose to add a new role, when I get to the screen asking me where I want to install that role, I see that I can choose to install a new role or feature onto one of my other servers, even though I am not working from the console of those servers, as shown in the following screenshot:

HivaNetwork.Com



If I wanted to install the AD DS role onto DC2, a server I'm prepping as a second domain controller in my environment, I would *not* have to log into the DC2 server. Right here, from Server Manager that is running on CA1, I could run through the Add Roles wizard, define DC2 as the server that I want to manipulate, and could install the role directly from here.

Remote Server Administration Tools (RSAT)

Using Server Manager in order to log into a single server and have access to manage and monitor all of your servers is pretty handy, but what if we could take one more step out of that process? What if I told you that you didn't have to log into *any* of your servers, but could perform all of these tasks from the computer sitting on your desk?

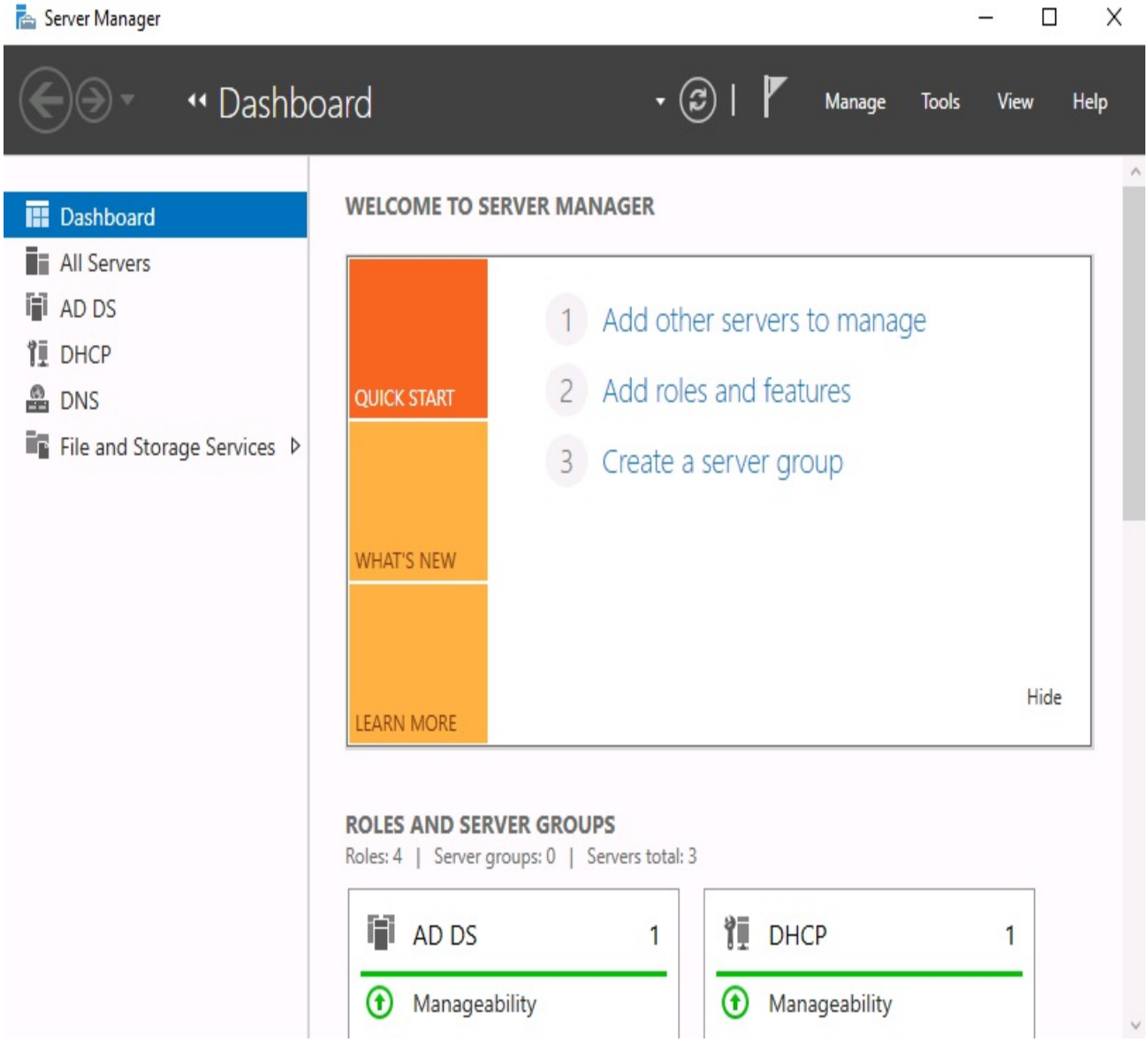
This is possible by installing a toolset from Microsoft called the **Remote Server Administration Tools** (**RSAT**). I have a regular Windows 10 client computer online and running in our network, also domain joined. On this computer I want to download and install the RSAT tools from the following location: https://www.microsoft.com/en-us/download/details.aspx?id=45520.

> *If you happen to be running Windows 10 1809 or newer, the RSAT toolset is now included within Windows, but is an optional feature that you must enable. This is accomplished from Windows Settings, inside the Apps category via a button called Manage optional features.*

After running the installer on my Windows 10 client computer, I can't seem to find any program that is called the Remote Server Administration Tool. That would be correct. Even though the name of this when downloading and installing is RSAT, after installation the program that is actually placed on your computer is called **Server Manager**. This makes sense, except that if you don't realize the name discrepancy, it can take you a few minutes to figure out why you cannot find what you just installed.

So, go ahead and launch Server Manager by finding it in the Start menu, or by using the search bar, or even by saying *Hey Cortana, open Server Manager*. Sorry, I couldn't resist. But whatever your method, open up Server Manager on your desktop computer and you will see that it looks and feels just like Server Manager in Windows Server 2019. And, in the same way that you work with and manipulate it within the server operating system, you can take the same steps here in order to add your servers for management.

In the following screenshot, you can see that, within my Windows 10 Server Manager, I now have access to manage and monitor all of the servers in my lab, without even having to log into them:

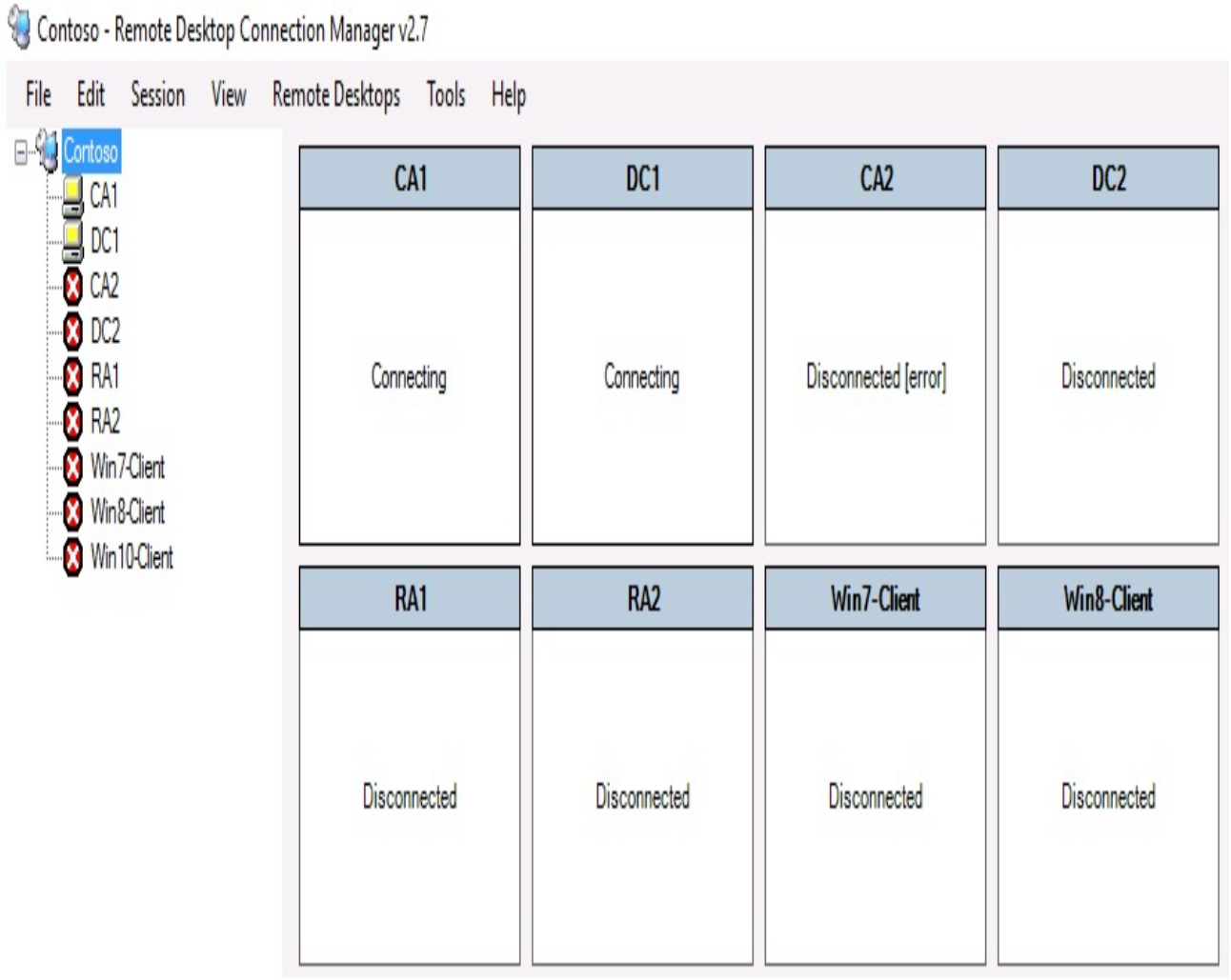

Server Manager
Dashboard
Manage
Tools
View
Help
Dashboard
All Servers
AD DS
DHCP
DNS
File and Storage Services
WELCOME TO SERVER MANAGER
QUICK START
WHAT'S NEW
LEARN MORE
1 Add other servers to manage
2 Add roles and features
3 Create a server group
Hide
ROLES AND SERVER GROUPS
Roles: 4 | Server groups: 0 | Servers total: 3
AD DS 1
Manageability
DHCP 1
Manageability

Does this mean RDP is dead?

With these new and improved ways to manage the underlying components of your servers without having to log into them directly, does this mean that our age-old friend RDP is going away? Certainly not! We will still have the need for accessing our servers directly sometimes, even if we go all-in with using the newer management tools. And I also expect that many administrators out there will continue using RDP and full desktop-based access for all management and monitoring of their servers simply because that is what they are more comfortable with, even if newer, more efficient ways now exist to accomplish the same tasks.

Remote Desktop Connection Manager

Since most of us do still utilize RDP occasionally (or often) when bouncing around between our servers, let's take a quick look at a tool that can at least make this task more manageable and centralized. I won't spend a lot of time looking over individual features or capabilities of this tool, since it is a client-side tool and not something that is specific to Windows Server 2019. You can use this to handle RDP connections for any and all of your servers, or even all of the client computers in your network. However, the **Remote Desktop Connection Manager** is an incredibly useful platform for storing all of the different RDP connections that you make within your environment. You can save connections so that you don't have to spend time trying to remember server names, sort servers into categories, and even store credentials so that you don't have to type passwords when connecting to servers. Though a disclaimer should come with that one, your security folks may not be happy if you choose to employ the password storing feature. I will leave you with a link for downloading the application, https://www.microsoft.com/en-us/download/details.aspx?id=44989, as well as the following screenshot, and then leave it up to you to decide whether or not this tool would be helpful in your daily tasks:


Contoso - Remote Desktop Connection Manager v2.7
File
Edit
Session
View
Remote Desktops
Tools
Help
Contoso
CA1
DC1
CA2
DC2
RA1
RA2
Win7-Client
Win8-Client
Win10-Client
CA1
Connecting
DC1
Connecting
CA2
Disconnected [error]
DC2
Disconnected
RA1
Disconnected
RA2
Disconnected
Win7-Client
Disconnected
Win8-Client
Disconnected

Windows Admin Center (WAC)

Now forget everything I just told you about remote server management, and focus here instead. I'm just kidding, sort of. All of the tools we have already discussed are still stable, relevant, and great ways to interact with and manage your bunches of Windows Servers. However, there's a new kid in town, and Microsoft expects him to be very popular.

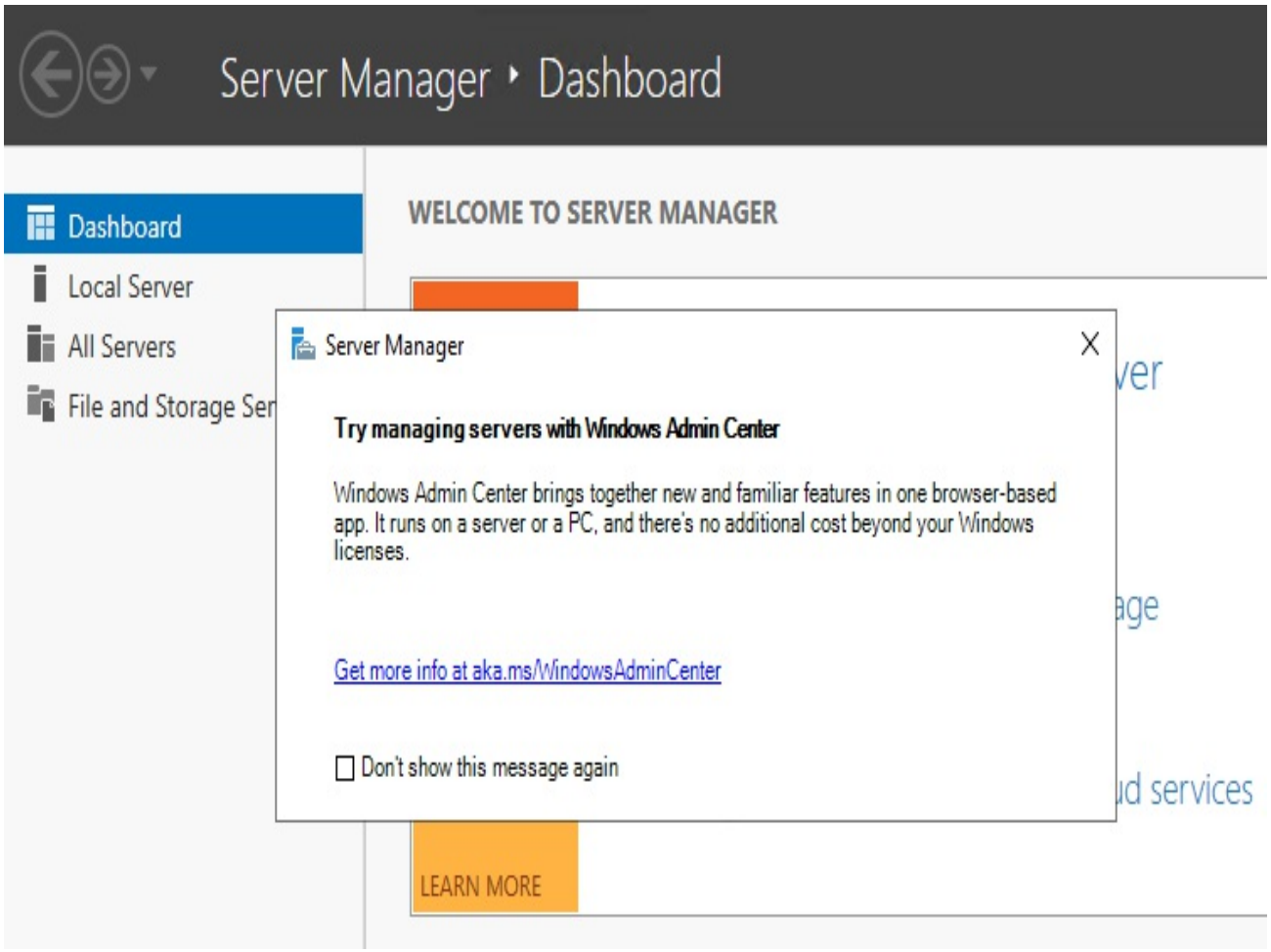
Windows Admin Center (**WAC**) is a server and client management platform that is designed to help you administer your machines in a more efficient manner. This is a browser-based tool, meaning that, once installed, you access WAC from a web browser, which is great. No need to install a management tool or application onto your workstation, simply sit down and tap into it with a URL.

WAC can manage your servers (all the way back to Server 2008 R2), your server clusters, and even has some special functionality for managing hyper-converged infrastructure clusters. You can also manage client machines in the Windows 10 flavor.

> *What's the cost for such an amazing, powerful tool? FREE! And chances are that if you have been following Microsoft activity over the past year, you may have even seen it already in one form or another. Do the words* **Project Honolulu** *sound familiar? Yes, Windows Admin Center is a renamed Project Honolulu, finally ready for production use.*

Windows Admin Center even has support for third-party vendors to be able to create extensions for the WAC interface, so this tool is going to continue growing. If you have been following along with the test lab configuration in the book so far, you will recognize Windows Admin Center from a pop-up window that displays itself every time that Server

Manager is opened. Microsoft wants administrators to know about WAC so badly that they are reminding you that you should start using it every time that you log into a Server 2019 box, as shown in the following screenshot:

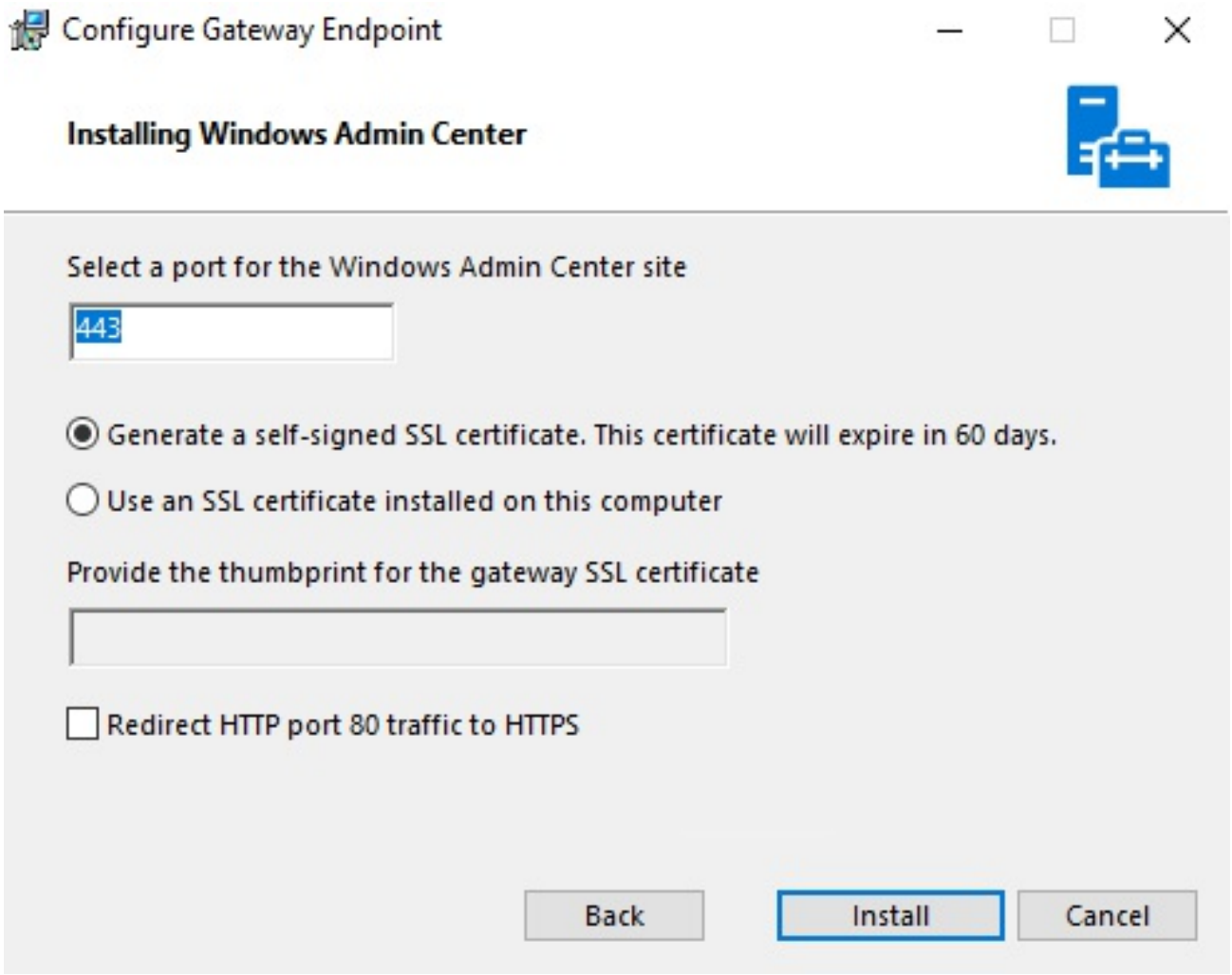


Installing Windows Admin Center

Enough talk, let's try it out! First we need to choose a location to install the components of WAC. True, I did say that one of the benefits was that we didn't need to install it, but what I meant was that once WAC is implemented, then tapping into it is as easy as opening up a browser. That website needs to be installed and running somewhere, right? While you could throw the whole WAC system onto a Windows 10 client, let's take the approach that will be more commonly utilized in the field and install it onto a server in our network. I have a system running called WEB3 which is not yet hosting any websites, sounds like a good place for something like this.

Download WAC from here: `https://www.microsoft.com/en-us/cloud-platform/windows-admin-center`.

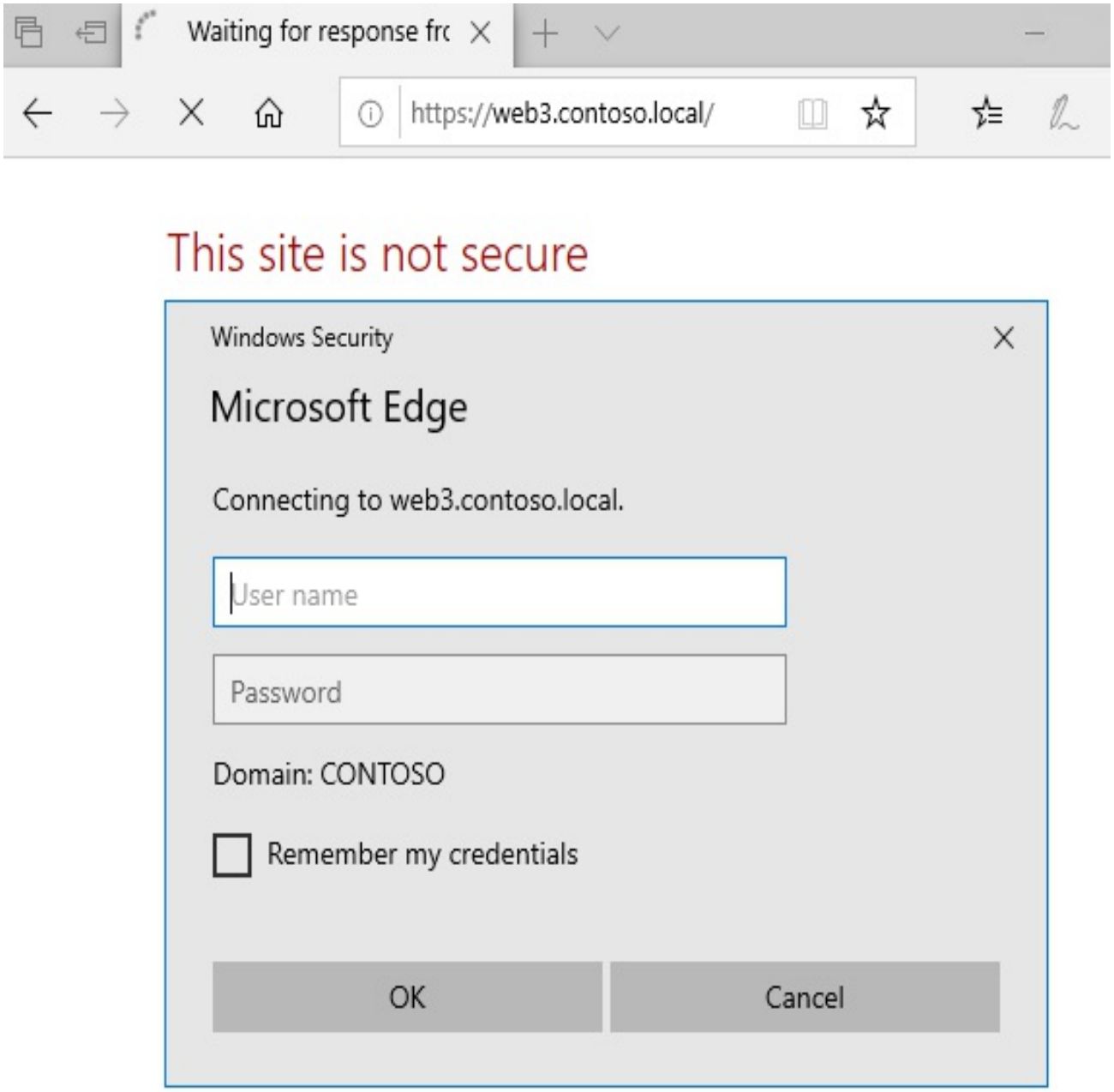
Once downloaded, simply run the installer on the host machine. There are a few simple decisions you need to make during the wizard, the most noticeable is the screen where you define port and certificate settings. In a production environment, it would be best to run port `443` and provide a valid SSL certificate here so that traffic to and from this website is properly protected via HTTPS, but, for my little test lab, I am going to run `443` with a self-signed certificate, just for testing purposes. Don't use self-signed certificates in production!



Once the installer is finished, you will now be hosting the Windows Admin Center website on this server. For my particular installation, that web address is: `https://WEB3.contoso.local`.

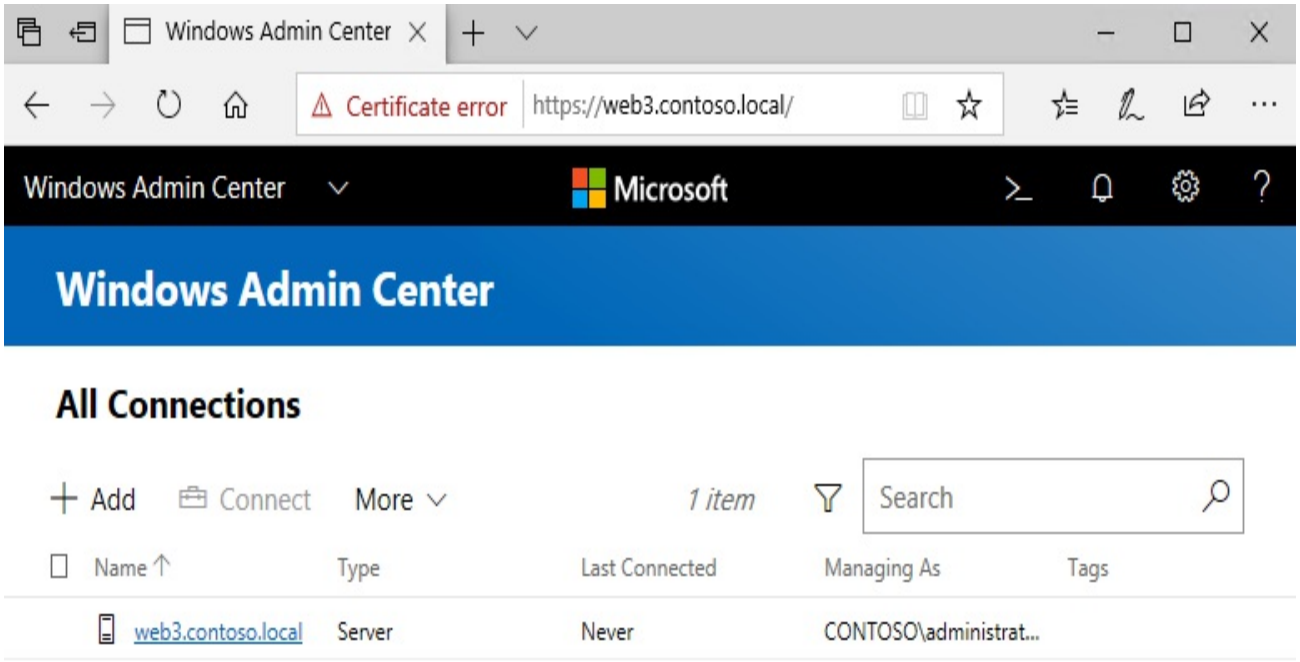
Launching Windows Admin Center

Now for the fun part, checking this thing out. To tap into Windows Admin Center, you simply open up a supported browser from any machine in your network, and browse to the WAC URL. Once again, mine is `https://WEB3.contoso.local`. Interestingly, Internet Explorer is not a supported browser, Microsoft recommends Edge but also works with Chrome. I am logged into my Windows 10 workstation, and will simply open up the Edge browser and try to hit my new site, as shown in the following screenshot:



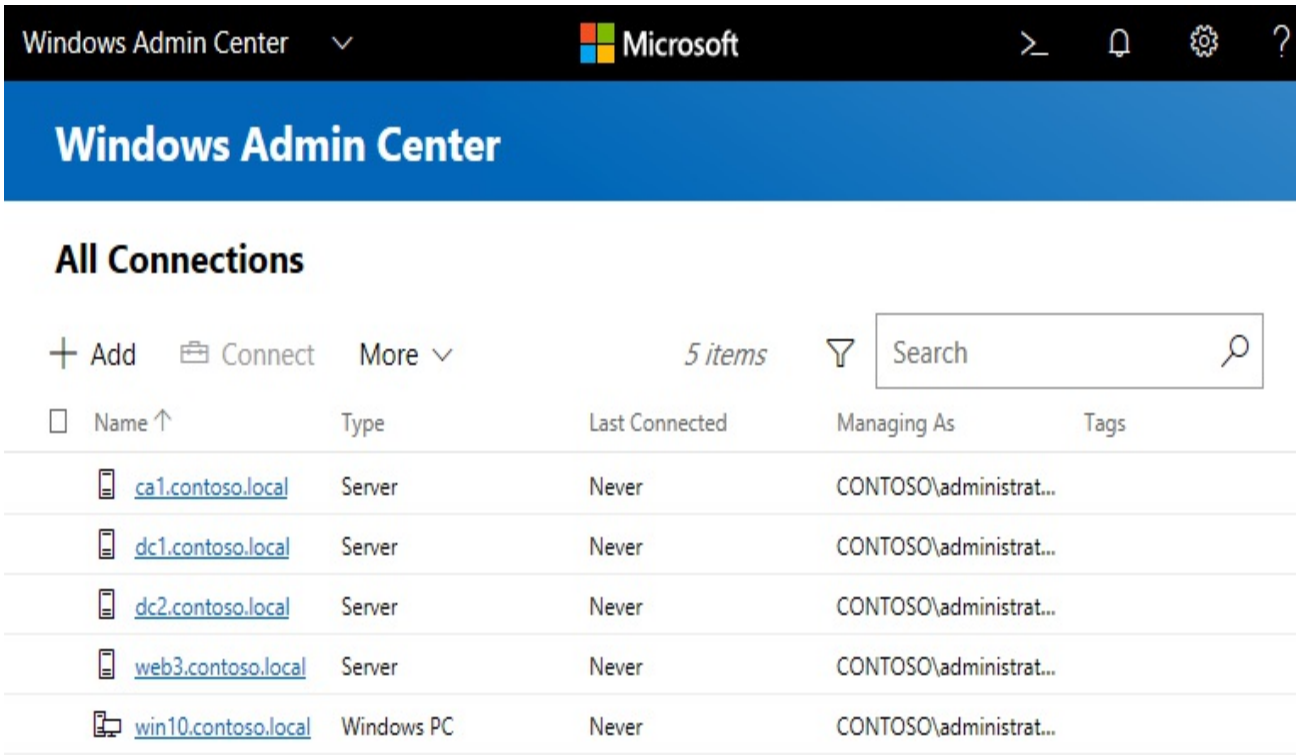
As you can see, I am dealing with a certificate warning. This is to be expected because I am using a self-signed certificate, which, once again, is a bad idea. I only justify it because I'm running in a test lab. But the more interesting part of the earlier screenshot is that I am being presented with a credentials prompt. Even though I am logged into a

Windows 10 computer that is domain-joined and I am logged in with domain credentials, the WAC website does not automatically try to inject those credentials for its own use, but rather pauses to ask who you are. If I simply input my domain credentials here, I am now presented with the Windows Admin Center interface, as shown in the following screenshot:



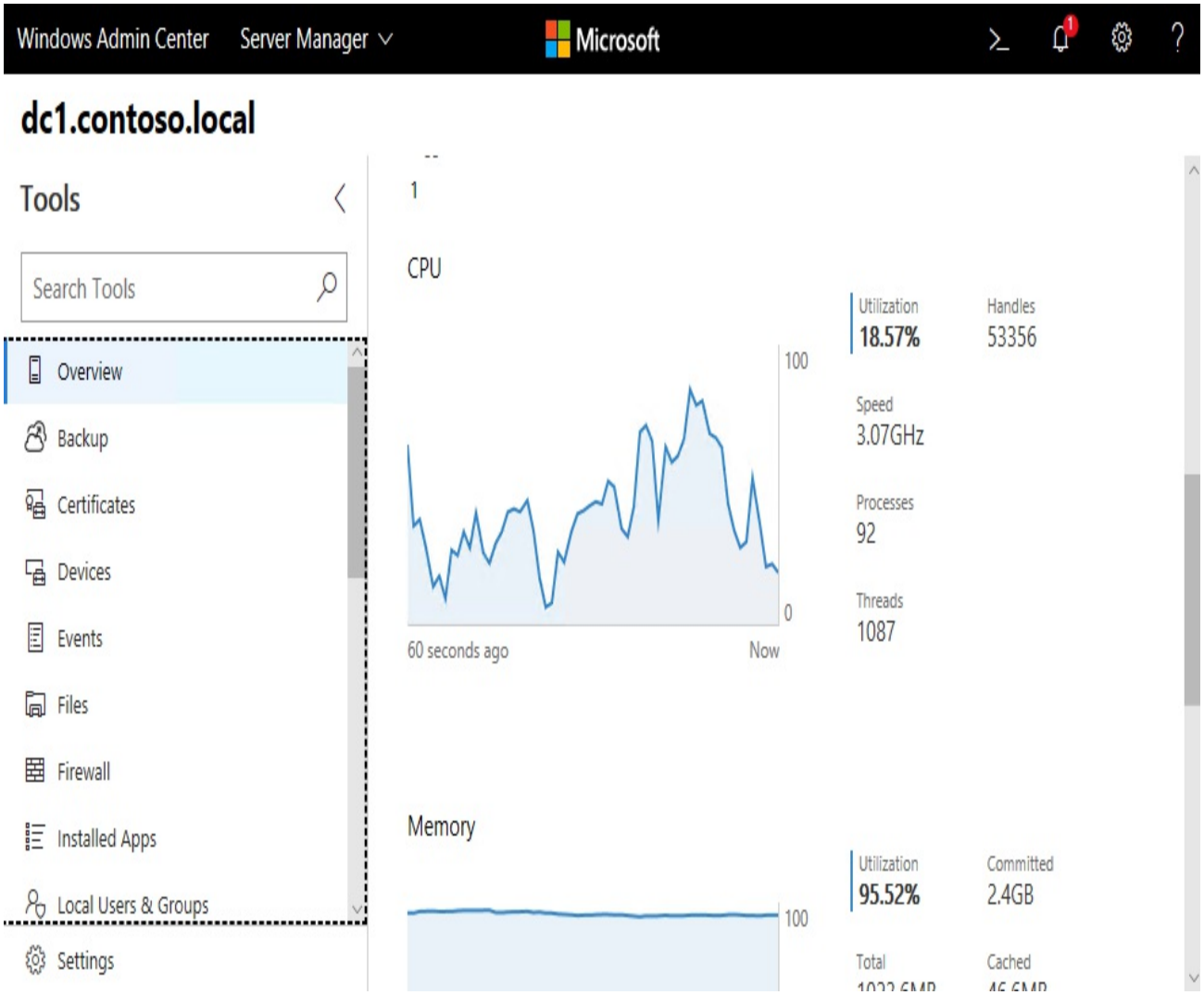
Adding more servers to Windows Admin Center

Logging into WAC is great, but not very useful until you add a bunch of machines that you want to manage. To do that, simply click the +Add button that is shown in the previous screenshot. You will be presented with choices to add a new server, a new PC, a failover cluster, or even a hyper-converged cluster. Make your selection and input the required information. I don't have any clusters in my test lab, not yet anyway, so I am going to add in connections to the standard servers that I have been running in the environment, as shown in the following screenshot:



Managing a server with Windows Admin Center

Beginning management of a server from within WAC is as simple as clicking on the server name. As you can see in the following screenshot, I have selected my DC1 server, as it is currently the only machine with some real roles installed and running:



From this interface, I can manage many different aspects of my DC1 server's operating system. There are power control functions, the ability to run backups on my server, I can even install certificates from here! You can monitor performance of the server, view its event logs, manipulate the local Windows Firewall, and launch a remote PowerShell connection to the server. The goal with Windows Admin Center is for it to be your one-stop shop for remotely managing your servers, and I would say it is well on its way to accomplishing that goal.

I don't yet have any Server Core instances running in my lab, but rest

assured that WAC can be used to manage Server Core instances just as well as servers running Desktop Experience. This makes Windows Admin Center even more potent and intriguing to server administrators.

Enabling quick server rollouts with Sysprep

At the beginning of this chapter, we walked through the process for installing the Windows Server 2019 operating system onto your new server. Whether this was a physical piece of hardware or a virtual machine that we were working with, the installation process was essentially the same. Plugging in the DVD or USB stick, booting to it, and letting the installer run its course is an easy enough thing to do, but what if you need to build out ten new servers instead of just one? This process would soon start to get tedious, and it would seem like you were wasting a lot of time having to do the exact same thing over and over again. You would be correct, this does waste a lot of time, and there is an easier and faster way to roll out new servers as long as you are building them all from a relatively similar hardware platform. If you are building out your servers as virtual machines, which is so often the case these days, then this process works great and can save you quite a bit of time with new server builds.

Now, before I go too far down this road of describing the Sysprep process, I will also note that there are more involved technologies available within the Windows infrastructure that allow automated operating system and server rollouts that can make the new server rollout process even easier than what I am describing here. The problem with some of the automated technologies is that the infrastructure required to make them work properly is more advanced than many folks will have access to if they are just learning the ropes with Windows Server. In other words, to have a fully automated server rollout

mechanism isn't very feasible for small environments or test labs, which is where a lot of us live while we are learning these new technologies.

So, anyway, we will not be focusing on an automation kind of approach to server rollouts, but rather we will be doing a few minutes of extra work on our very first server, which then results in saving numerous minutes of setup work on every server that we build afterwards. The core of this process is the **Sysprep** tool, which is baked into all versions of Windows, so you can take this same process on any current Windows machine, whether it be a client or a server.

Sysprep is a tool that prepares your system for duplication. Its official name is the **Microsoft System Preparation Tool**, and to sum up what it does in one line, it allows you to create a master *image* of your server that you can reuse as many times as you want in order to roll out additional servers. A key benefit to using Sysprep is that you can put customized settings onto your master server and install things such as Windows Updates prior to Sysprep, and all of these settings and patches will then exist inside your master image. Using Sysprep saves you time by not having to walk through the operating system installation process, but it saves you even more time with not having to wait for Windows Updates to roll all of the current patches down onto every new system that you create. Now, some of you might be wondering why Sysprep is even necessary.

If you wanted to clone your master server, you could simply use a hard disk imaging tool, or, if you were dealing with virtual machines, you could simply copy and paste the `.VHDX` file itself in order to make a copy of your new server, right? The answer is yes, but the big problem is that the new image or hard drive that you just created would be an exact replica of the original one. The hostname would be the same, and, more importantly, some core identification information in Windows, such as the operating system's **security identifier** (**SID**) number, would be

exactly the same. If you were to turn on both the original master server and a new server based off this exact replica, you would cause conflicts and collisions on the network as these two servers fight for their right to be the one and only server with that unique name and SID. This problem exacerbates itself in domain environments, where it is even more important that each system within your network has a unique SID/GUID—their identifier within Active Directory. If you create exact copies of servers and have them both online, let's just say neither one is going to be happy about it.

Sysprep fixes all of these inherent problems to the system duplication process by randomizing the unique identifiers in the operating system. In order to prepare ourselves to roll out many servers using a master image we create with Sysprep, here is a quick-reference summary of the steps we will be taking:

1. Install Windows Server 2019 onto a new server
2. Configure customizations and updates onto your new server
3. Run Sysprep to prepare and shut down your master server
4. Create your master image of the drive
5. Build new servers using copies of the master image

And now let's cover these steps in a little more detail.

Installing Windows Server 2019 onto a new server

First, just like you have already done, we need to prepare our first server by getting the Windows Server 2019 operating system installed. Refrain from installing any full roles onto the server, because, depending on the role or its unique configuration, the Sysprep process that we run shortly could cause problems for individual role configurations. Install the operating system and make sure device drivers are all squared away, and you're ready for the next step.

Configuring customizations and updates onto your new server

Next, you want to configure customizations and install updates onto your new server. Each setting or installation that you can do now that is universal to your batch of servers will save you from having to take that step on your servers in the future. This portion may be slightly confusing because I just told you a minute ago not to install roles onto the master server. This is because a role installation makes numerous changes in the operating system, and some of the roles that you can install lock themselves down to a particular hostname running on the system. If you were to do something like that to a master server, that role would more than likely break when brought up on a new server. Customizations that you can put into place on the master server are things such as plugging in files and folders that you might want on all of your servers, such as an `Admin Tools` folder or something like that. You could also start or stop services that you may or may not want running on each of your servers, and change settings in the registry if that is part of your normal server prep or hardening process. Whatever changes or customizations you put into place, it's not a bad idea to run a full slew of tests against the first new server that you build from this master image, just to make sure all of your changes made it through the Sysprep process.

Now is also the time to let Windows Updates install and to put any patches on this new server that you want to have installed on all of your new servers in the future. There is nothing more frustrating than installing a new operating system in 5 minutes, only to have to sit

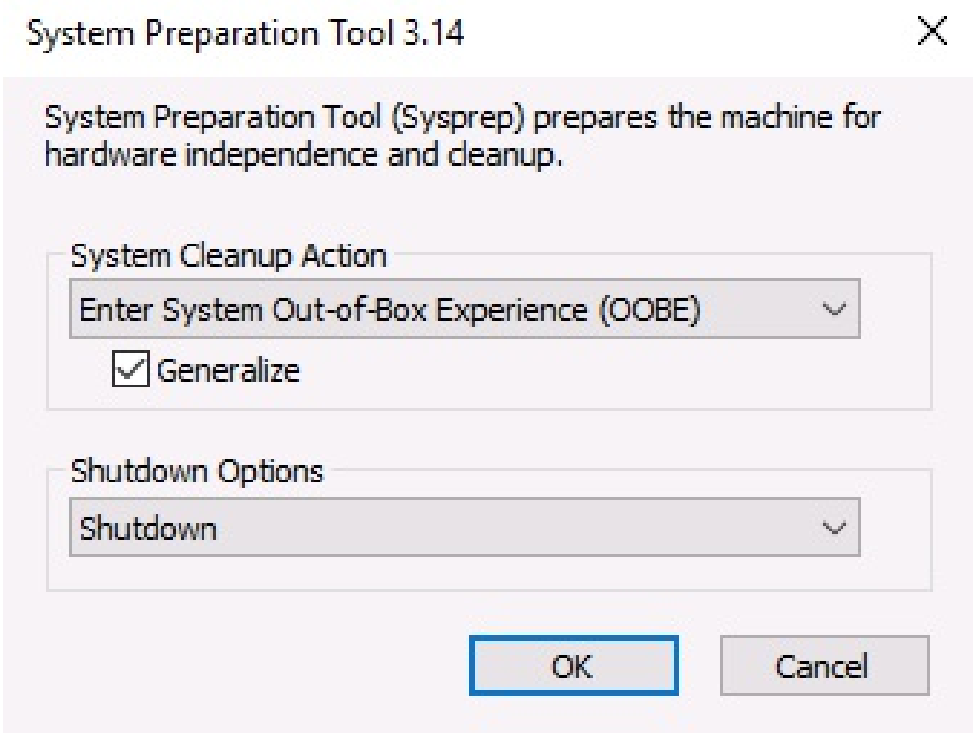
around and wait 4 hours for all of the current updates and patches to be installed before you can use the new server. By including all of these updates and patches in the master image, you save all of that download and installation time for each new server that you spin up.

> *Continue to save yourself time and effort by creating new copies of your master images every few months. This way the newest patches are always included in your master image and it continues to save you more and more time throughout the life of Windows Server 2019.*

Running Sysprep to prepare and shut down your master server

Now that our master server is prepped how we want, it is time to run the Sysprep tool itself. To do that, open up an administrative command prompt and browse to `C:\Windows\System32\Sysprep`. Now you can make use of the `Sysprep.exe` utility inside that folder in order to launch Sysprep itself.

As with many executables that you run from a command prompt, there are a variety of optional switches that you can tag onto the end of your command in order to make it do specific tasks. From your command prompt window, if you simply run the `sysprep.exe` command, you will see a graphical interface for Sysprep, where you can choose between the available options, as shown in the following screenshot:



Since I always use the same set of options for Sysprep, I find it easier to simply include all of my optional switches right from the command-line input, therefore bypassing the graphical screen altogether. Here is some information on the different switches that are available to use with `sysprep.exe`:

- `/quiet`: This tells Sysprep to run without status messages on the screen.
- `/generalize`: This specifies that Sysprep is to remove all of the unique system information (SID) from the Windows installation, making the final image usable on multiple machines in your network, because each new one spun up from the image will get a new, unique SID.
- `/audit`: This restarts the machine into a special audit mode, where

you have the option of adding additional drivers into Windows before the final image gets taken.

- `/oobe`: This tells the machine to launch the mini-setup wizard when Windows next boots.
- `/reboot`: This restarts when Sysprep is finished.

- `/shutdown`: This shuts down the system (not a restart) when Sysprep is finished. This is an important one and is one that I typically use.
- `/quit`: This closes Sysprep after it finishes.
- `/unattend`: There is a special `answerfile` that you can create that, when specified, will be used in conjunction with the Sysprep process to further configure your new servers as they come online. For example, you can specify in this `answerfile` that a particular installer or batch file is to be launched upon first Windows boot following Sysprep. This can be useful for any kind of cleanup tasks that you might want to perform, for example, if you had a batch file on your system that you used to flush out the log files following the first boot of new servers.

The two that are most important to our purposes of wanting to create a master image file that we can use for quick server rollouts in the future are the `/generalize` switch and the `/shutdown` switch. Generalize is very important because it replaces all of the unique identification information, the SID info, in the new copies of Windows that come

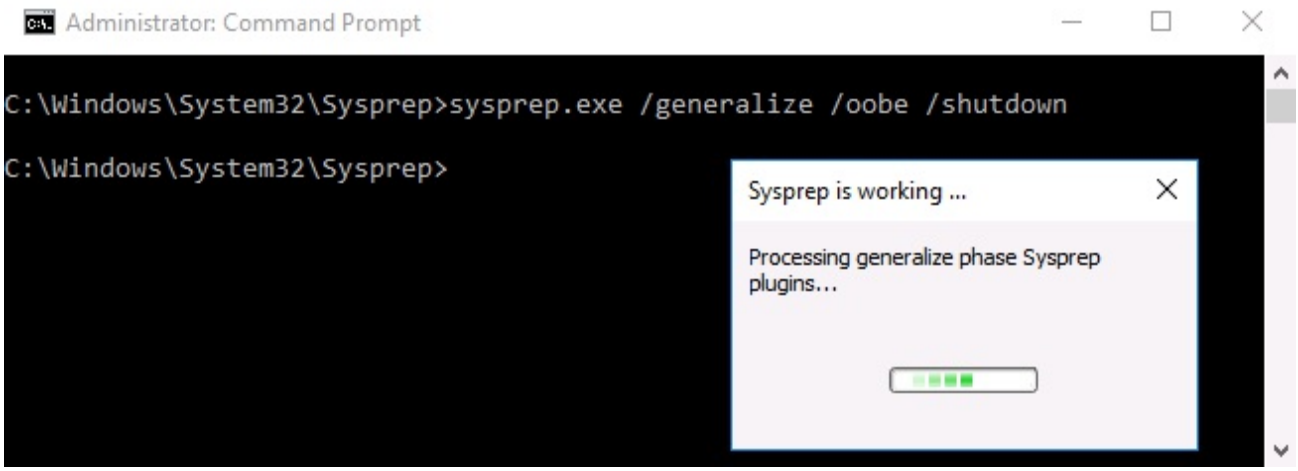
online. This allows your new servers to co-exist on the network with your original server, and with other new servers that you bring online. The shutdown switch is also very important, because we want this master server to become sysprepped and then immediately shutdown so that we can create our master image from it.

> *Make sure that your server does NOT boot into Windows again until after you have created your master image, or taken your master copy of the* `.VHDX` *file. The first time that Windows boots it will inject the new SID information, and you want that only to happen on new servers that you have created based off your new image.*

So, rather than simply throwing all of the switches at you and letting you decide, let's take a look at the ones that I typically use. I will make use of `/generalize` so that I make my new servers unique, and I also like to use `/oobe` so that the mini-setup wizard launches during the first boot of Windows on any of my new systems. Then, I will of course also use `/shutdown`, because I need this server to be offline immediately following Sysprep so that I can take a copy of the hard drive to be used as my master image. So, my fully groomed `sysprep` command is shown in the following code:

```
sysprep.exe /generalize /oobe /shutdown
```

After launching this command, you will see Sysprep moving through some processes within Windows, and after a couple of minutes, your server will shut itself down, as shown in the following screenshot:



You are now ready to create your master image from this hard disk.

Creating your master image of the drive

Our master server is now shutdown, and we are ready to create our master image from this server. If it is a physical server, you can use any hard disk imaging utility in order to create an image file from the drive. An imaging utility like those from the company Acronis will create a single file from your drive. This file contains an image of the entire disk that you can use to restore down onto fresh hard drives in new servers in the future. On the other hand, most of you are probably dealing with virtual servers most often in your day-to-day work lives, and prepping new servers in the virtual world is even easier. Once our master server has been sysprepped and shutdown, you simply create a copy of the `.VHDX` file. Log into your Hyper-V server, copy and paste the hard disk file, and you're done. This new file can be renamed `WS2019_Master_withUpdates.VHDX` , or whatever you would like it to be named, in order to help you keep track of the current status on this image file. Save this image file or copy of the `.VHDX` somewhere safe on your network, where you will be able to quickly grab copies of it whenever you have the need to spin up a new Windows Server 2019.

Building new servers using copies of the master image

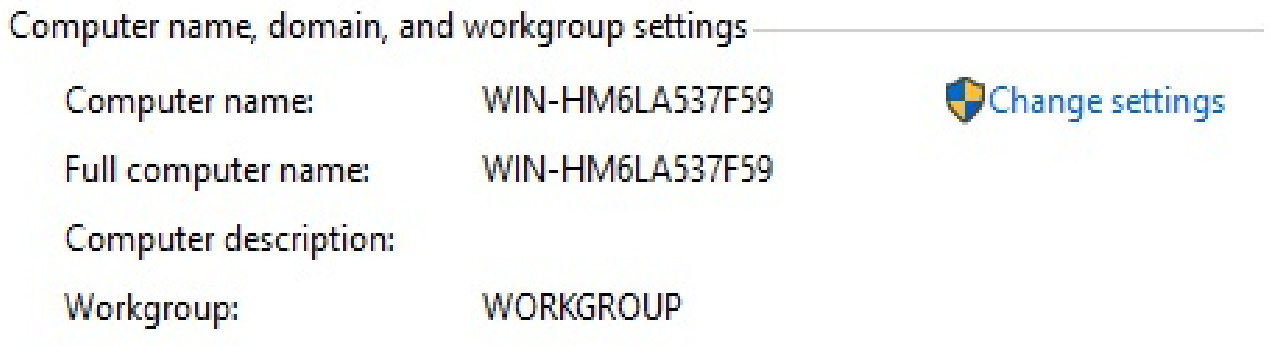
Now we get to the easy part. When you want to create new servers in the future, you simply copy and paste your master file into a new location for the new server, rename the drive file to be something appropriate for the server you are creating, and boot your new virtual machine from it. Here is where you see the real benefit from the time that Sysprep saves, as you can now spin up many new servers all at the same time, by doing a quick copy and paste of the master image file and booting all of your new servers from these new files. No need to install Windows or pull out that dusty installation DVD!

As the new servers turn on for the first time and boot into Windows, they will run through the out-of-box-experience, mini-setup wizard. Also, in the background, the operating system gives itself a new random and unique hostname and SID information so that you can be sure you do not have conflicts on your network with these new servers.

> *New servers created from a sysprepped image file always receive a new hostname when they boot. This often confuses admins who might have named their master server something such as* `MASTER`*. After booting your new servers, you can expect to see randomized names on your new servers and you will have to rename them according to their new duties in life.*

For example, before running Sysprep and creating my master image, the server that I was working on was named DC1 because I had originally intended to use it as a Domain Controller in my network. However, because I had not installed the role or configured anything domain related on it, this server was a perfect candidate for displaying the

Sysprep process and so I used it in our text today. I sysprepped it, shut it down, made a copy of its VHDX file (to be my master image file), and then I started DC1 back up. You can now see inside the system properties that I am back to having a randomized hostname, and so if I still want to use this server as `DC1`, I will have to rename it again now that it has finished booting through mini-setup, as shown in the following screenshot:



Hopefully, this process is helpful information that can save you time when building out new servers in your own environments. Get out there and give it a try the next time you have a new server to build! You can further benefit yourself with the Sysprep tool by keeping many different master image files. Perhaps you have a handful of different kinds of servers that you prep regularly, there is nothing stopping you from creating a number of different master servers, and creating multiple master images from these servers.

Summary

Anyone interested in being a Windows Server administrator needs to be comfortable with installing and managing their servers, and covering those topics establishes an important baseline for moving forward. It is quite common in today's IT world for new operating system releases to be thoroughly tested, both because server hardware resources are so easily available to us through virtualization technologies, and because most business systems are now being designed for 100% uptime. This kind of reliability requires very thorough testing of any platform changes, and, in order to accomplish such testing of the Windows Server 2019 operating system in your environment, you will be burning quite a bit of time spinning through the basic installation processes numerous times. I hope that you can put the suggestions provided in this chapter to good use in saving you precious extra minutes when dealing with these tasks in your Windows Server world.

Years ago, quite a bit of effort was regularly put into figuring out which roles and services could co-exist, because the number of servers available to us was limited. With the new virtualization and cloud paradigm shift, many companies have a virtually unlimited number of servers that can be running, and this means we are running much larger quantities of servers to perform the same jobs and functions. Management and administration of these servers then becomes an IT burden, and adopting the centralized administration tools and ideas available within Windows Server 2019 will also save you considerable time and effort in your daily workload. In the next chapter we will look into the core infrastructure services.

Questions

1. What is the name of the new web-based, centralized server management tool from Microsoft (formerly known as Project Honolulu)?
2. True or False—Windows Server 2019 needs to be installed onto rack-mount server hardware.
3. True or False—By choosing the default installation option for Windows Server 2019, you will end up with a user interface that looks quite like Windows 10.
4. What is the PowerShell cmdlet that displays currently installed roles and features in Windows Server 2019?
5. True or False—Server Manager can be used to manage many different servers at the same time?
6. What is the name of the toolset that can be installed onto a Windows 10 computer, in order to run Server Manager on that client workstation?
7. What are the supported web browsers that can be used to interact with Windows Admin Center?

Core Infrastructure Services

Each of you reading this book is going to have a different acquired skillset and level of experience with the Windows Server environment. As I mentioned previously, being able to make servers run the operating system is great and a very important first step for doing real work in your environment. But until you know and understand what the purposes behind the main roles available to run on Windows Server 2019 are, the only thing your new server does is consume electricity.

A server is intended to serve data. The kinds of data that it serves and to what purpose depends entirely on what roles you determine the server must ... well ... serve. Appropriately, you must install roles within Windows Server 2019 to make it do something. We already know how to get roles installed onto our server, but have not talked about any of the purposes behind these roles. In this chapter, we will take a look at the core infrastructural roles available within Windows Server. This involves discussing the role's general purpose, as well as plugging in some particular tasks dealing with those roles that you will be responsible for doing in your daily tasks as a server administrator. In this chapter, we will learn the following:

- What is a Domain Controller?
- Using AD DS to organize your network
- The power of Group Policy
- **Domain Name System (DNS)**

- DHCP versus static addressing
- Backup and restore
- MMC and MSC shortcuts

What is a Domain Controller?

If we are going to discuss the core infrastructure services that you need in order to piece together your Microsoft-driven network, there is no better place to start than the Domain Controller role. A **Domain Controller**, commonly referred to as a **DC**, is the central point of contact, and is sort of a central hub, that is accessed prior to almost any network communication that takes place. The easiest way to describe it is as a storage container for all *identification* that happens on the network. Usernames, passwords, computer accounts, groups of computers, servers, groups and collections of servers, security policies, file replication services, and many more things are stored within and managed by DCs. If you are not planning to have a Domain Controller be one of the first servers in your Microsoft-centric network, you might as well not even start building that network. DCs are essential to the way that our computers and devices communicate with each other and with the server infrastructure inside our companies.

Active Directory Domain Services

If you've stopped reading at this point in order to install the *Domain Controller* role onto your server, welcome back because there is no role called Domain Controller. The role that provides all of these capabilities is called **Active Directory Domain Services**, or **AD DS**. This is the role that you need to install on a server. By installing that role, you will have turned your server into a Domain Controller. The purpose of running a DC really is to create a directory, or database, of objects in your network. This database is known as **Active Directory**, and is a platform inside which you build an hierarchical structure to store objects, such as usernames, passwords, and computer accounts.

Once you have created a domain in which you can store these accounts and devices, you can then create user accounts and passwords for your employees to utilize for authentication. You can then also join your other servers and computers to this domain so that they can accept and benefit from those user credentials. Having and joining a domain is the secret sauce that allows you to walk from computer to computer within your company and log onto each of them with your own username and password, even when you have never logged into that computer before. Even more powerful is the fact that it enables directory-capable applications to authenticate directly against Active Directory when they need authentication information. For example, when I, as a domain user, log in to my computer at work with my username and password, the Windows operating system running on my computer reaches out to a Domain Controller server and verifies that my password is correct.

Once it confirms that I really am who I say I am, it issues an

authentication token back to my computer and I am able to log in. Then, once I am into my desktop and open an application—let's say I open my Outlook to access my email—that email program is designed to reach out to my email server, called an **Exchange Server**, and authenticate against it to make sure that my own mailbox is displayed and not somebody else's. Does this mean I need to re-enter my username and password for Outlook, or for any other application that I open from my computer? Generally not. And the reason I do not have to re-enter my credentials over and over again is because my username, my computer, and the application servers, are all part of the domain. When this is true, and it is for most networks, my authentication token can be shared among my programs. So, once I log in to the computer itself, my applications are able to launch and open, and pass my credentials through to the application server, without any further input from me as a user. It would be quite a frustrating experience indeed if we required our users to enter passwords all day, every day as they opened up the programs that they need in order to do their work.

The first Domain Controller you set up in your network will be a fully-writable one, able to accept data from the domain-joined users and computers working within your network. In fact, most DCs in your network will likely be fully functional. However, it's worth taking a quick minute to point out a limited-scope DC that can be installed called a **Read-Only Domain Controller** (**RODC**). Just like the name implies, an RODC can only have its directory data read from it. Writes that might try to be accomplished to the domain from a user's computer, such as a password change or new user account creation, are impossible with an RODC. Instead, RODCs receive their directory data from other more traditional Domain Controllers, and then utilize that data to verify authentication requests from users and computers. Where would a limited-access Domain Controller like this be beneficial? Many companies are installing them into smaller branch offices or less secure sites, so that the local computers onsite in those smaller offices have

quick and easy access to read from and authenticate to the domain, without the potential security risk of an unauthorized user gaining access to the physical server and manipulating the entire domain in bad ways.

Active Directory itself is a broad enough topic to warrant its own book, and indeed there have been many written on the topic. Now that we have a basic understanding of what it is and why it's critical to have in our Windows Server environment, let's get our hands dirty using some of the tools that get installed onto your Domain Controller during the AD DS role installation process.

Using AD DS to organize your network

There is not a single tool that is used to manage all facets of Active Directory. Since it is such an expansive technology, our configuration of the directory is spread across a number of different management consoles. Let's take a look at each of them, and a couple of the most common tasks that you will be performing inside these tools. Any of these management consoles can be launched from any of your Domain Controller servers, and just as we saw in a previous chapter, the easiest way to launch these consoles is right from the Tools menu in the upper-right corner of Server Manager.

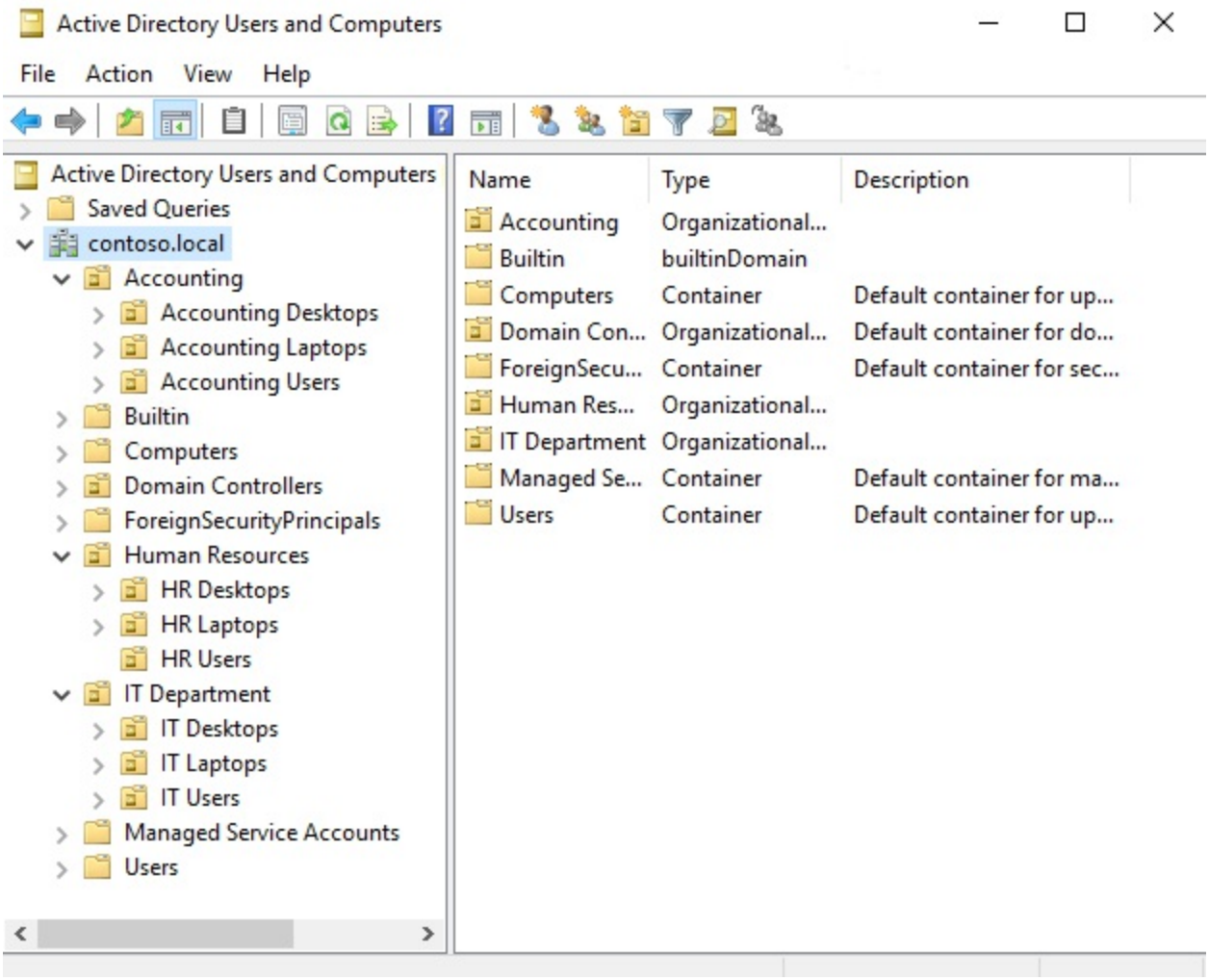
Active Directory Users and Computers

I'll start with the tool that is alphabetically last in the list of our Active Directory tools, because this is by far the one which the everyday server administrator will use most often. AD Users and Computers is the console from which all of the user accounts and computer accounts are created and managed. Open it up, and you will see the name of your domain listed in the left-hand column. Expand your domain name, and you will see a number of folders listed here. If you are opening this on an existing Domain Controller in a well-grown network, you may have pages and pages of folders listed here. If this is a new environment, there are only a handful. The most important pieces to point out here are Computers and Users. As common sense would dictate, these are the default containers in which new computer accounts and user accounts that join the domain will be located.

While this window looks quite a bit like File Explorer with a tree of folders, these *folders* really aren't folders at all. Most of the manila-colored folder icons that you see here are known as **Organizational Units** (**OUs**). I say *most of* because there are a few containers that exist out of the box that are legitimate storage containers for holding objects, but they are not official OUs. The ones we pointed out earlier, Users and Computers, are actually these generic storage containers and are not real Organizational Units. However, any new folders that you create for yourself inside AD are going to be OUs. The difference is depicted in the manila folder icon. You can see in the upcoming screenshots that some of the manila folders have an extra little graphic on top of the folder

itself. Only those folders that have the extra little yellow thing are real OUs.

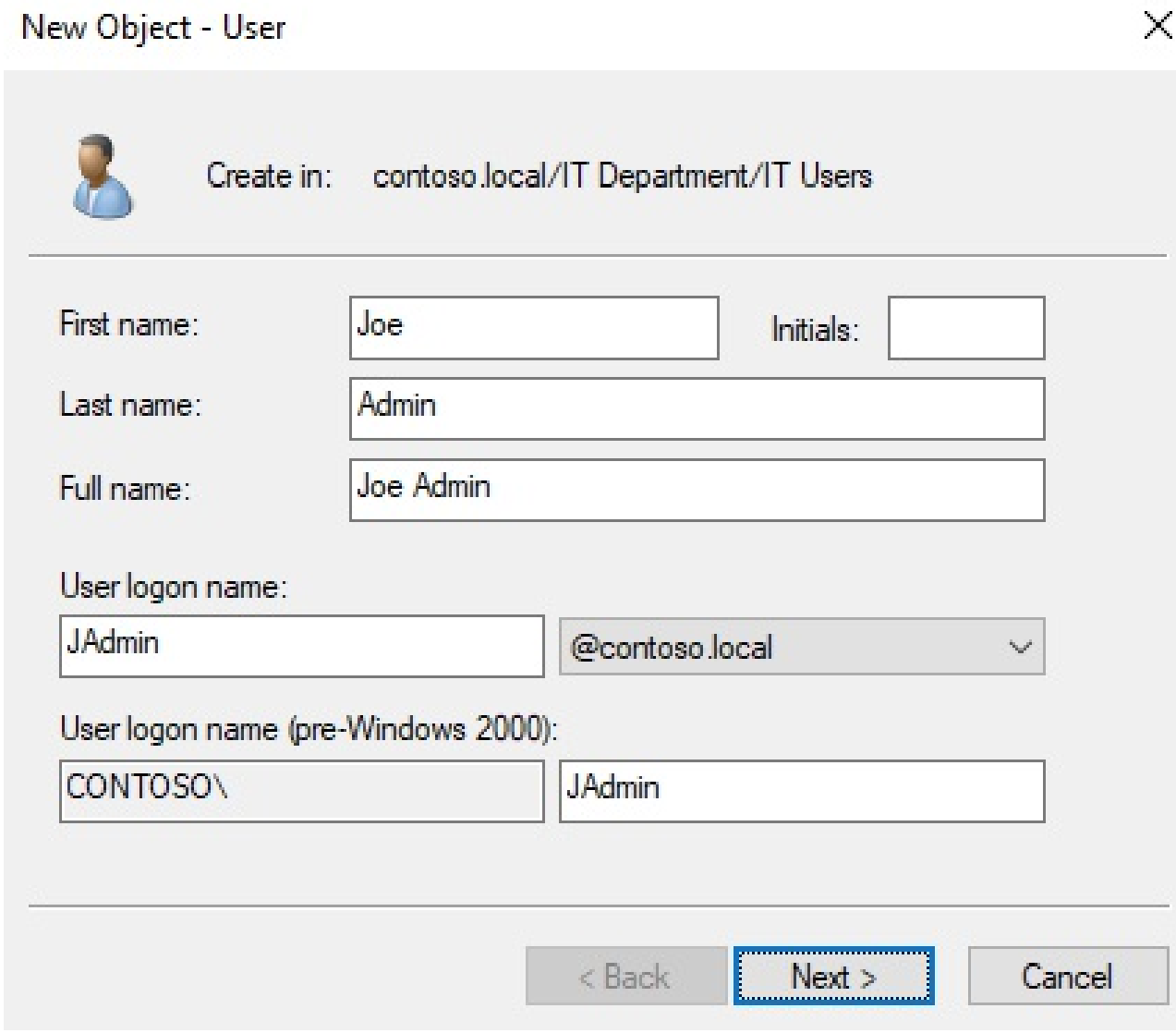
OUs are the structural containers that we use inside Active Directory in order to organize our objects and keep them all in useful places. Just like with folders on a file server, you can create your own hierarchy of organizational units here, in order to sort and manipulate the location inside Active Directory of all your domain-joined network objects and devices. In the following screenshot, you can see that instead of having just a plain Users and Computers folder, I have created some new OUs including subcategories (more officially known as **nested OUs**) so that as I grow my environment, I will have a more structured and organized directory:


Active Directory Users and Computers
File
Action
View
Help
Active Directory Users and Computers
Saved Queries
contoso.local
Accounting
Accounting Desktops
Accounting Laptops
Accounting Users
Builtin
Computers
Domain Controllers
ForeignSecurityPrincipals
Human Resources
HR Desktops
HR Laptops
HR Users
IT Department
IT Desktops
IT Laptops
IT Users
Managed Service Accounts
Users
Name
Type
Description
Accounting
Organizational...
Builtin
builtinDomain
Computers
Container
Default container for up...
Domain Con...
Organizational...
Default container for do...
ForeignSecu...
Container
Default container for sec...
Human Res...
Organizational...
IT Department
Organizational...
Managed Se...
Container
Default container for ma...
Users
Container
Default container for up...

User accounts

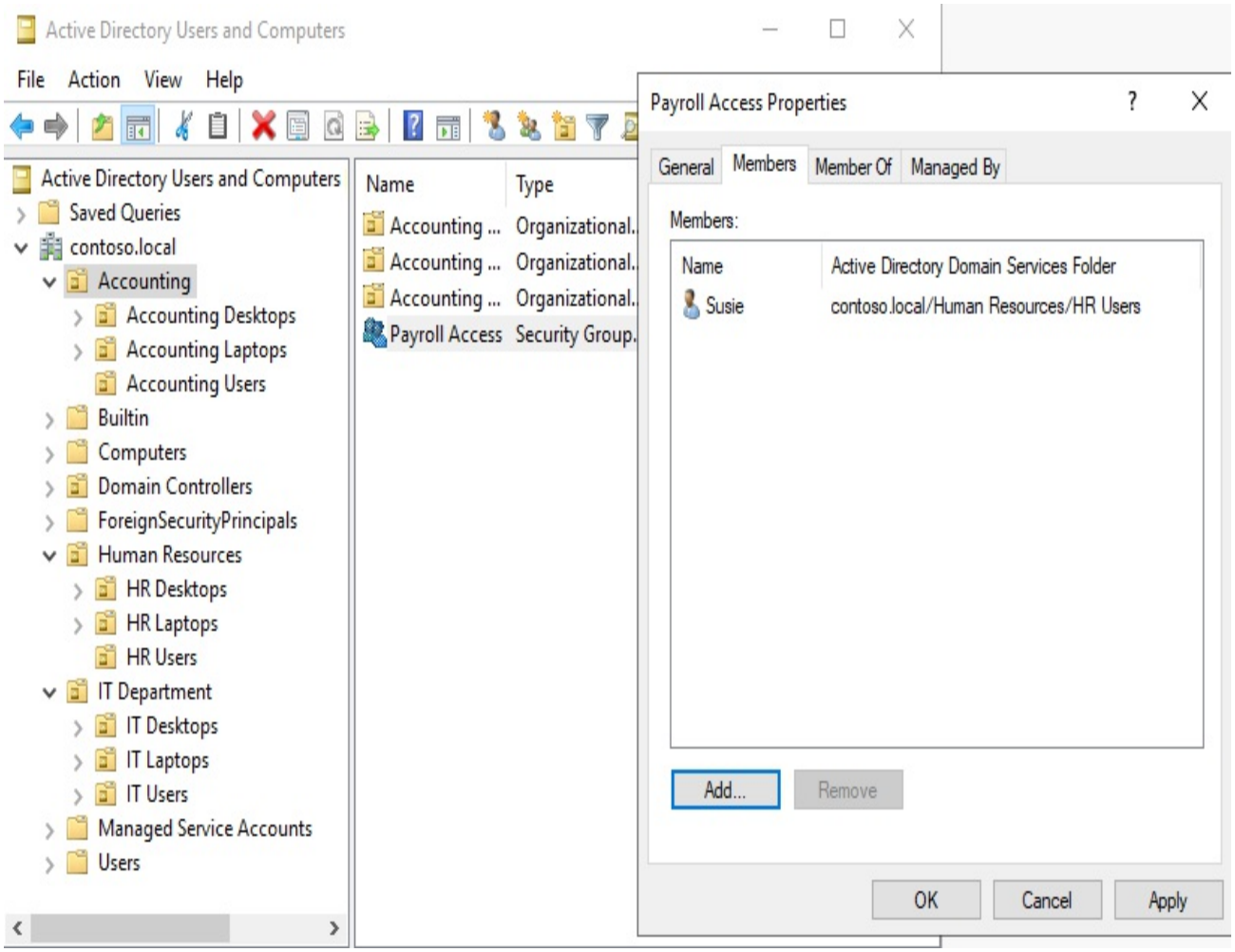
Now that we have some OUs ready to contain our objects, let's create a new user. Say we have a new Server Administrator coming onboard, and we need to get him an Active Directory login so that he can start his job. Simply find the appropriate OU for his account to reside within, right-click on the OU, and navigate to New | User. We are then presented with an information-gathering screen about all the things that AD needs in order to create this new account. Most of the information here is self-explanatory, but if you are new to Active Directory, the one field I will point out is User logon name. Whatever information is put in this field is the user's official **username** on the network. Whenever they log into a computer or server, this is the name they will input as their login.

When finished, our new admin will be able to utilize the new username and password in order to log in to computers and servers on the network, within the security boundaries we have established on those machines, of course. But that is another topic for another chapter.


New Object - User
Create in: contoso.local/IT Department/IT Users
First name:
Joe
Initials:
Last name:
Admin
Full name:
Joe Admin
User logon name:
JAdmin
@contoso.local
User logon name (pre-Windows 2000):
CONTOSO\
JAdmin
< Back
Next >
Cancel

Security Groups

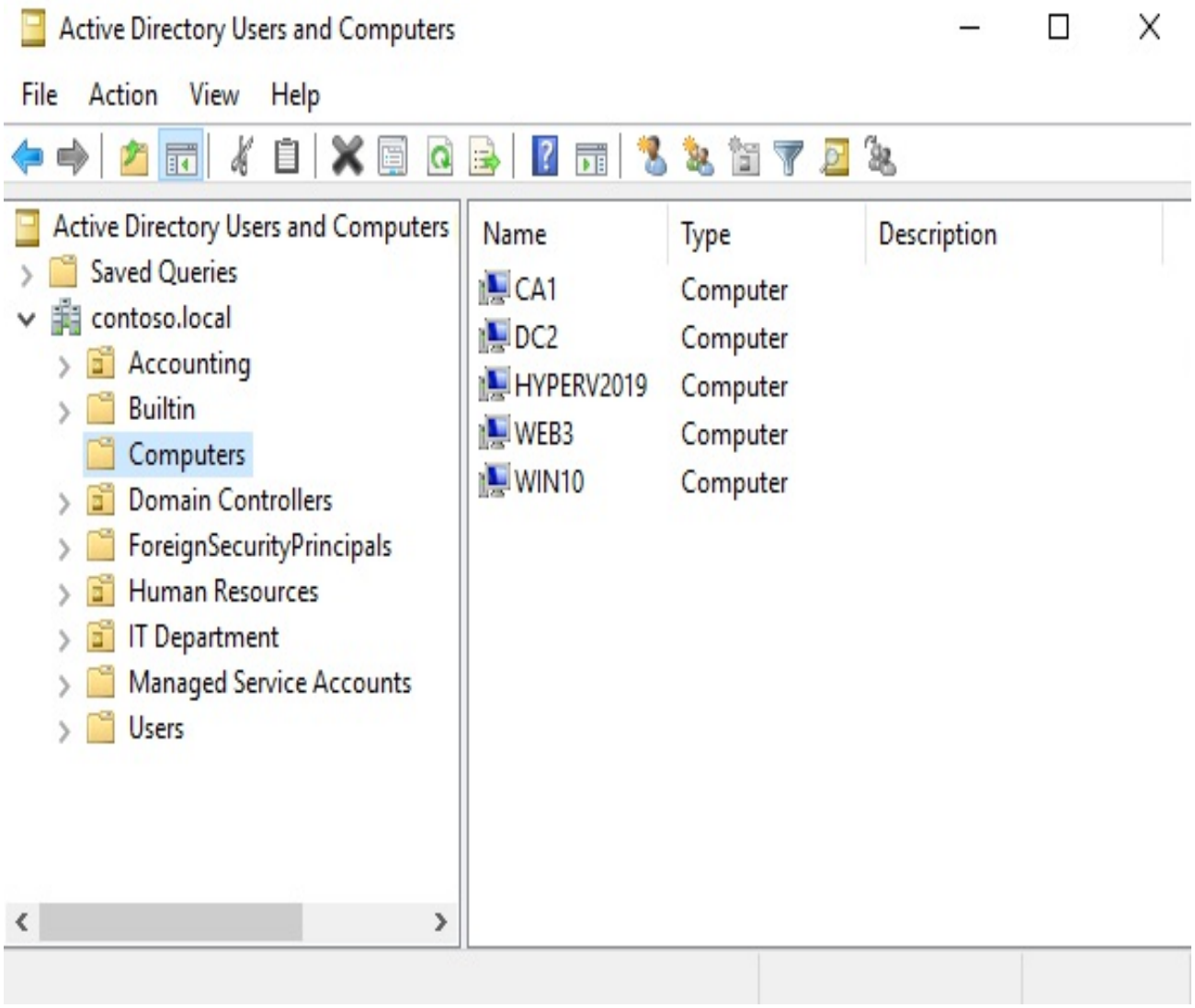
Another useful unit of organization inside Active Directory is **Security Groups**. We can do quite a bit to distinguish between different types and kinds of users and computer accounts using organizational units, but what about when we need a little cross-contamination in this structure? Perhaps we have an employee that handles some HR and some accounting responsibilities. Maybe it is more likely that we have configured file and folder permissions on our file servers so that only people who are part of certain groups have access to read and write into particular folders. Susie from HR needs to have access to the payroll folder, but Jim from HR does not. Both Susie and Jim reside inside the same OU, so at that level they will have the same permissions and capabilities, but we clearly need a different way to distinguish between them so that only Susie gets access to payroll information. By creating Security Groups inside Active Directory, we enable users to add and remove specific user accounts, computer accounts, or even other groups so that we can granularly define access to our resources. You create new groups in the same way that you create user accounts, by choosing the OU where you want the new group to reside, and then right-clicking on that OU and navigating to New | Group. Once your group has been created, right-click on it and head into Properties. You can then click on the Members tab; this is where you add in all of the users that you want to be a part of this new group:



HivaNetwork.Com

Prestaging computer accounts

It is very common to utilize Active Directory Users and Computers for creating new user accounts, because without the manual creation of a user account that new person is going to be completely unable to login on your network. It is far less common, however, to think about opening this tool when joining new computers to your domain. This is because most domains are configured so that new computers are allowed to join the domain by actions performed on the computer itself, without any work being done inside Active Directory beforehand. In other words, as long as someone knows a username and password that has administrative rights within the domain, they can sit down at any computer connected to the network and walk through the domain-join process on that local computer. It will successfully join the domain, and Active Directory will create a new computer object for it automatically. These auto-generating computer objects place themselves inside the default Computers container, so in many networks, if you click on that `Computers` folder, you will see a number of different machines listed, and they might even be a mix of both desktop computers and servers that were recently joined to the domain and haven't been moved to an appropriate, more specific OU yet. In my growing lab environment, I have recently joined a number of machines to the domain. I did this without ever opening AD Users and Computers, and you can see that my new computer objects are still sitting inside that default Computers container:



Allowing new computer accounts to place themselves inside the default **Computers** container is generally not a big problem for client systems, but if you allow servers to be autogenerated in that folder, it can cause you big issues. Many companies have security policies in place across the network, and these policies are often created in a way that they will be automatically applied to any computer account residing in one of the generalized OUs. Using security policies can be a great way to lock down parts of client machines that the user doesn't need to access or utilize, but if you inadvertently cause these **lockdown** policies to apply to your

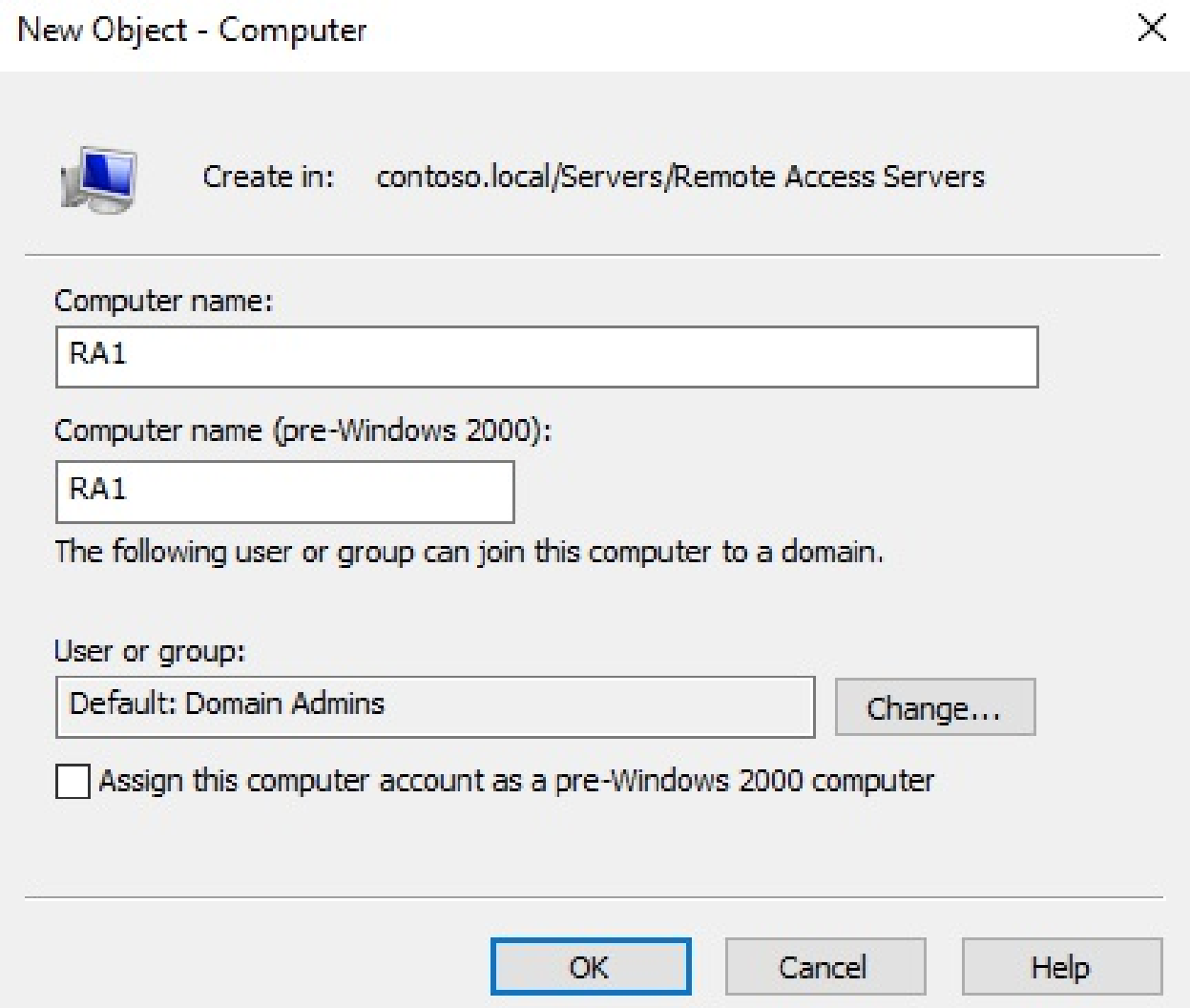
new servers as soon as they join the domain, you can effectively break the server before you even start configuring it. Trust me, I've done it. And, unfortunately, your new server accounts that get added to Active Directory will be identified and categorized the same as any client workstation that is added to the domain, so you cannot specify a different default container for servers simply because they are a server and not a regular workstation.

So, what can be done to alleviate this potential problem? The answer is to prestage the domain accounts for your new servers. You can even prestage all new computer accounts as a matter of principle, but I typically only see that requirement in large enterprises. Prestaging a computer account is very much like creating a new user account. Prior to joining the computer to the domain, you create the object for it inside Active Directory. By accomplishing the creation of the object before the domain-join process, you get to choose which OU the computer will reside in when it joins the domain. You can then ensure that this is an OU that will or will not receive the security settings and policies that you intend to have in place on this new computer or server. I highly recommend prestaging all computer accounts in Active Directory for any new servers that you bring online. If you make it a practice, even if it's not absolutely required all the time, you will create a good habit that may someday save you from having to rebuild a server that you broke simply by joining it to your domain.

Prestaging a computer object is extremely fast and simple; let's do one together. In the future, I plan to build a Windows Server that hosts the Remote Access role in order to connect my roaming users into the network from their homes, coffee shops, and so on. Some components in the Remote Access role are finicky when it comes to network security policies, and so I would rather ensure that my new RA1 server will not receive a whole bunch of lockdown settings as soon as I join it to the domain. I have created an OU called Remote Access Servers, and I will

now prestage a computer object inside that OU for my RA1 server.

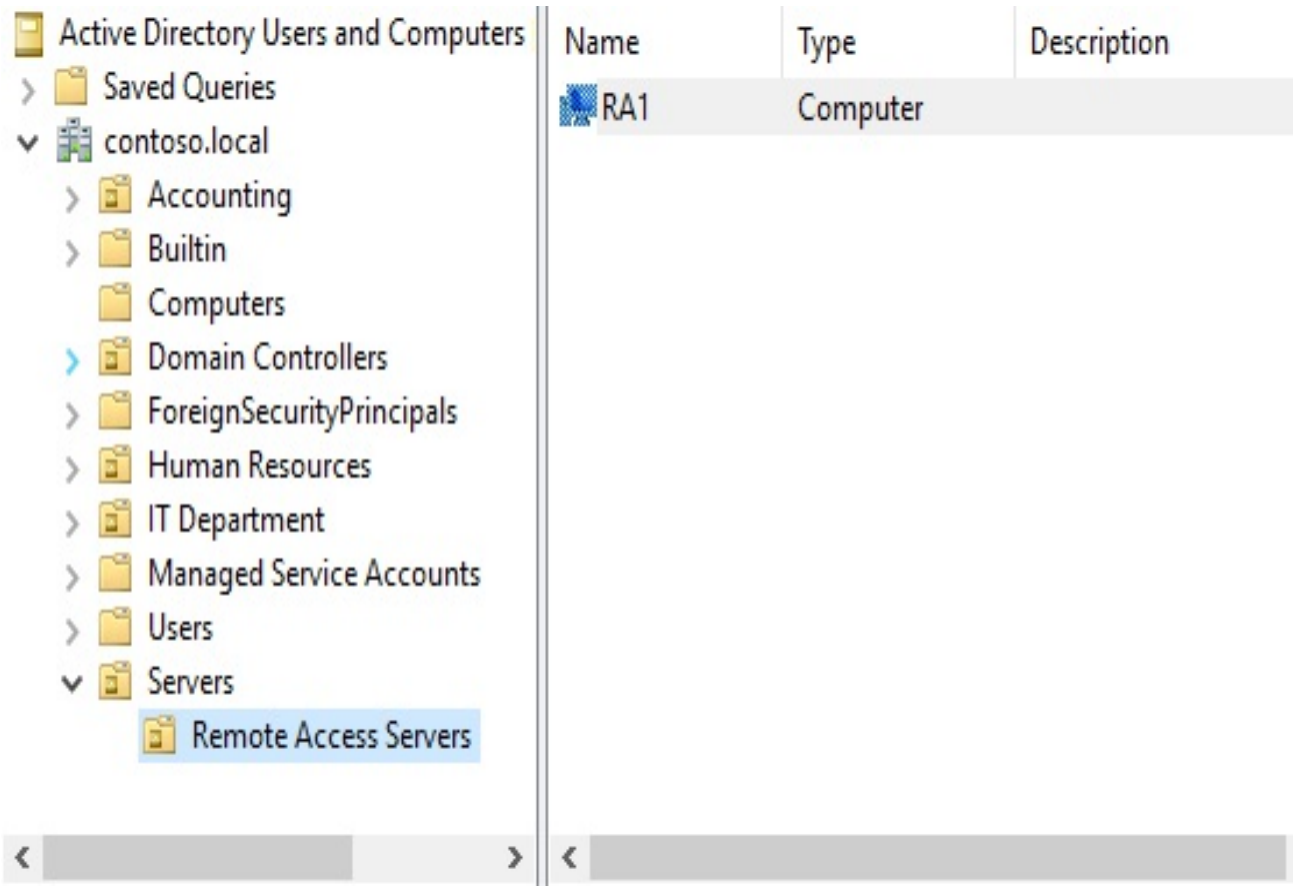
Right-click on the Remote Access Servers OU, and choose New | Computer. Then simply populate the **Computer name** field with the name of your server. Even though I have not built this server yet, I plan to name it `RA1`, so I simply type that into the field:



That's it! With a couple of simple mouse clicks and typing in one server name, I have now prestaged (pre-created) my computer object for the RA1 server. If you look closely at the previous screenshot, you will

notice that you could also adjust which users or groups are allowed to join this particular computer to the domain. If you plan to build a new server and want to make sure that you are the only person allowed to join it to the domain, this field is easily updated to accommodate that requirement.

Once I actually get around to building that server, and I go ahead and walk through the steps of joining it to my domain, Active Directory will associate my new server with this prestaged `RA1` object instead of creating a brand new object inside the generic **Computers** container:



Active Directory Domains and Trusts

This tool is generally only used in larger environments that have more than one domain within the same network. A company may utilize multiple domain names in order to segregate resources or services, or for better organizational structure of their servers and namespaces within the company. There is also another tier in the hierarchical structure of Active Directory that we haven't talked about, and that is called a **forest**. The forest is basically the top level of your Active Directory structure, with domains and subdomains coming in under that forest umbrella. Another way to think of the forest is as the boundary of your AD structure. If you have multiple domains beneath a single forest, it does not necessarily mean that those domains trust each other. So, users from one domain may or may not have permissions to access resources on one of the other domains, based on the level of trust that exists between those domains. When you have a domain and are adding child domains under it, there are trusts placed automatically between those domains, but if you need to merge some domains together in a way other than the default permissions, **Active Directory Domains and Trusts** is the management tool you use in order to establish and modify those trust permissions.

Growing organizations often find themselves in a position where they need to regularly manage domain trusts as a result of business acquisitions. If Contoso acquires Fabrikam, and both companies have fully-functional domain environments, it is often advantageous to work through an extended migration process to bring the Fabrikam

employees over to Contoso's Active Directory, rather than suffer all the loss associated with simply turning off Fabrikam's network. So, for a certain period of time, you would want to run both domains simultaneously, and could establish a trust relationship between those domains in order to make that possible.

If you find yourself in a position where a domain migration of any sort is necessary, there is a tool available that you may want to try out. It is called the **Active Directory Migration Tool** (**ADMT**) and can be very useful in situations like the one described earlier. If you are interested in taking a closer look at this tool, you can download it from the following link: https://www.microsoft.com/en-us/download/details.aspx?id=19188.

Active Directory Sites and Services

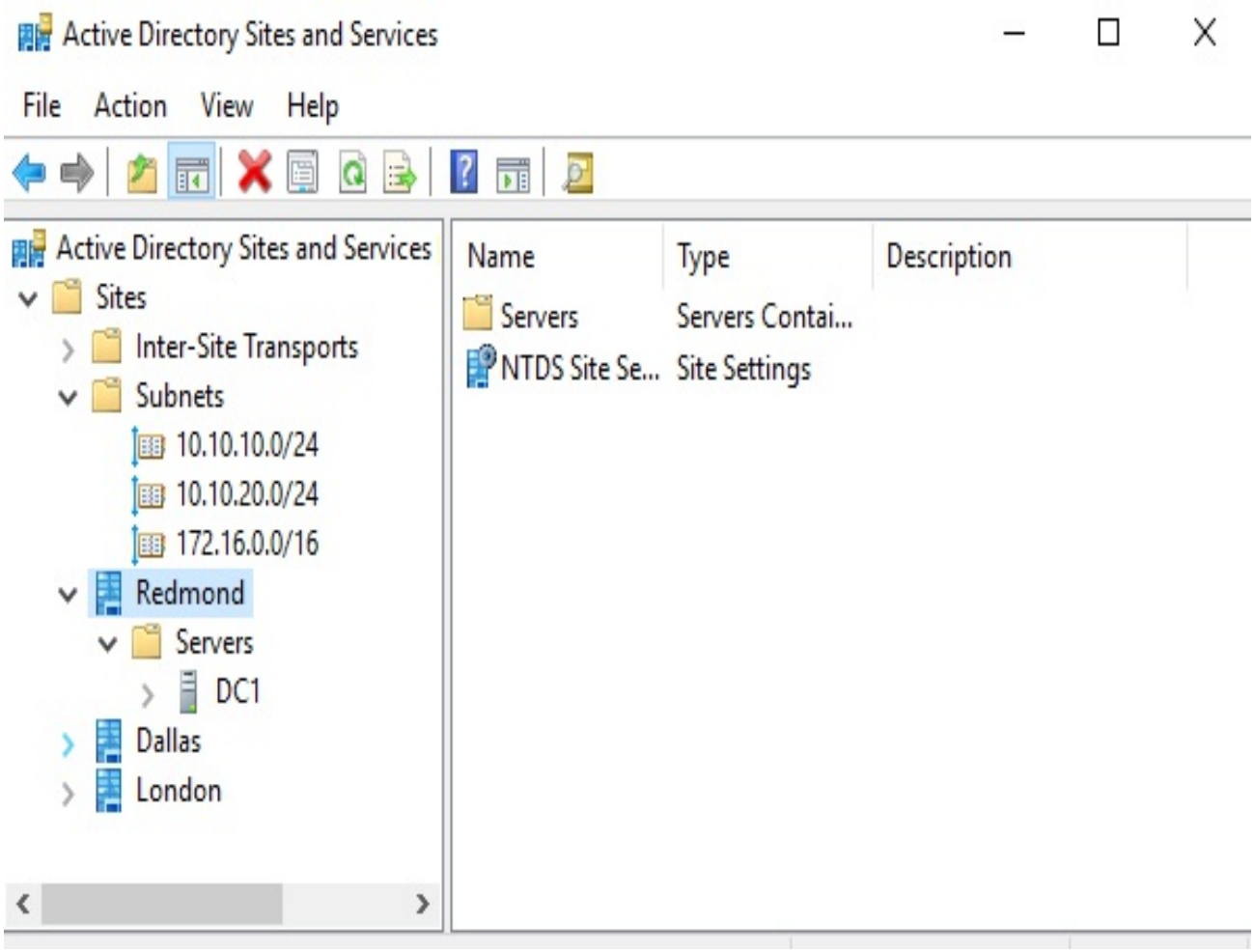
Sites and Services is another tool that is generally only employed by companies with larger Active Directory infrastructures. As is the case with any server, if having one Domain Controller is good, then having two Domain Controllers is even better. As your company grows larger, so does your Active Directory infrastructure. Before you know it, you will be looking into setting up servers in a second location, then a third, and so on. In a domain-centric network, having Domain Controller servers in each significant site is a general practice, and you could soon be looking at dozens of Domain Controller servers running in your network.

Turning on new Domain Controllers and joining them to your existing domain so that they start servicing users and computers is pretty easy. The harder part is keeping all of the traffic organized and flowing where you want it to. If you have a primary datacenter where the majority of your servers are located, you probably have multiple DCs onsite in that datacenter. In fact, in order to make your AD highly available, it is essential that you have at least two Domain Controllers. But let's pretend you then build a new office that is quite large, where it makes sense to install a local DC server in that office also, so that the computers in that office aren't reaching over the **Wide Area Network** (**WAN**) in order to authenticate all the time. If you were to spin up a server in the new office and turn it into a Domain Controller for your network, it would immediately start working. The problem is that the client computers aren't always smart enough to know which DC they

need to talk to. You may now have computers in the remote office that are still authenticating back to the main datacenter's DCs. Even worse, you probably also have computers in the main office that are now reaching over the WAN to authenticate with the new DC that is in the remote office, even though there are DCs right on the local network with them!

This is the situation where Active Directory Sites and Services become essential. In here, you build out your different physical sites and assign the Domain Controllers to these sites. Domain-joined users and computers within this network now follow the rules that you have put into place via Sites and Services, so that they are always talking to and authenticating from their local Domain Controller servers. This saves time, as the connections are faster and more efficient, and it also saves unnecessary bandwidth and data consumption on the WAN, which often saves you dollars.

Here's a quick look into Active Directory Sites and Services. As you can see, there are multiple sites listed here, and they correspond to network subnet information. This is the way that AD Sites and Services tracks which site is which. When a client computer comes online, it obviously knows what subnet it is part of, based on the IP address it is using. AD Sites and Services then knows, based on that IP address, which site the client now resides in. That site identification then helps Active Directory to steer authentication requests to the proper Domain Controllers, and also helps things like Group Policy (which we will talk about shortly) to be able to process site-specific information. There is a good chance you will have to make use of this tool someday if you are part of a growing organization:

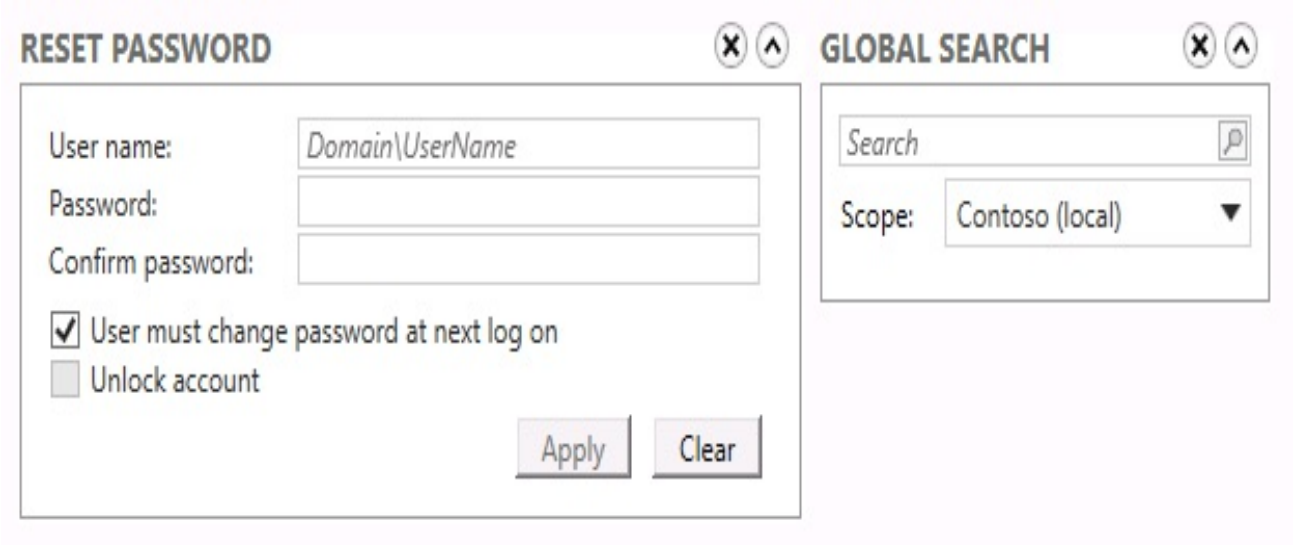


HivaNetwork.Com

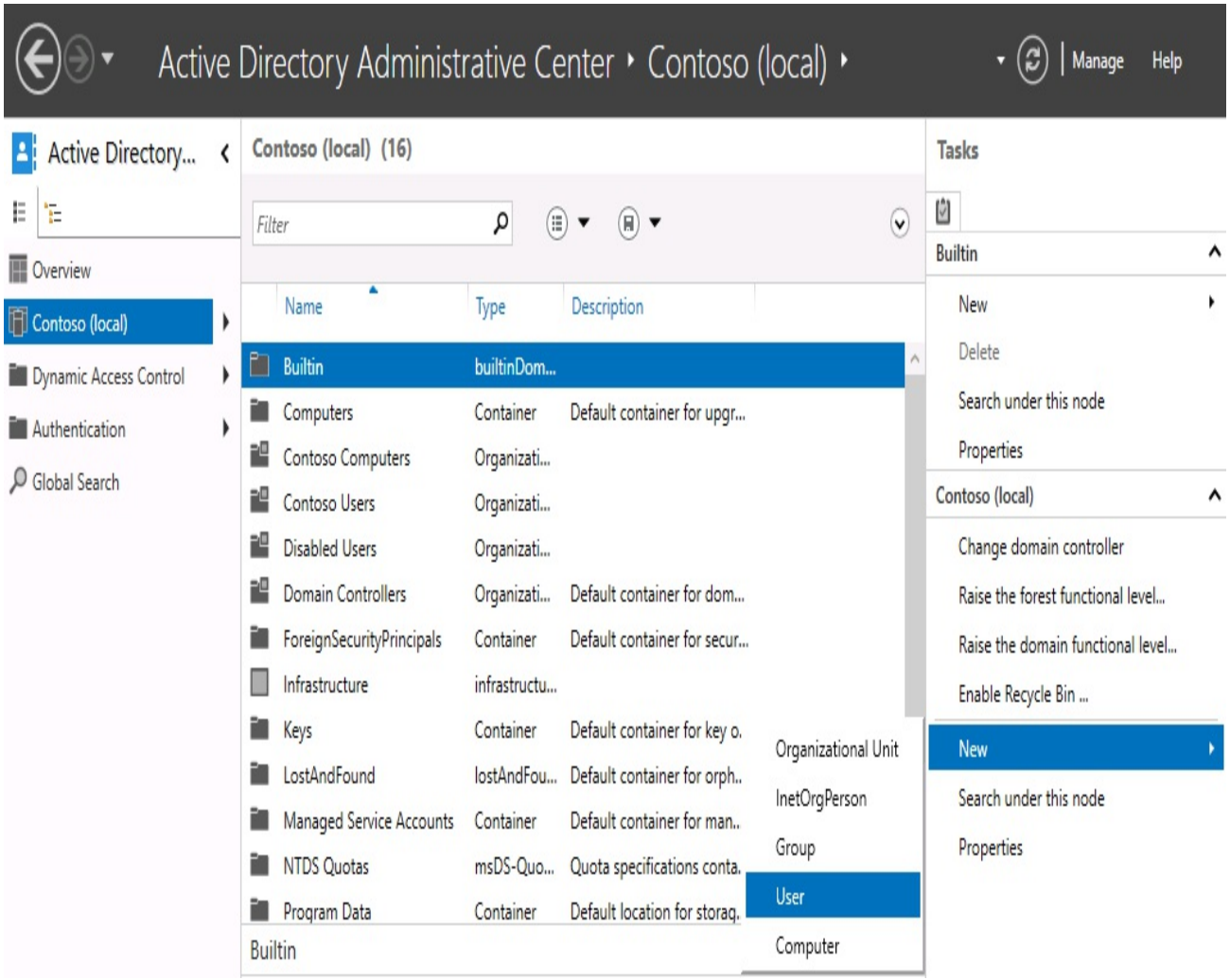
Active Directory Administrative Center

While it is critical to understand and be familiar with the tools we have looked at so far in order to manage Active Directory, you can tell that their aesthetics are a bit dated. The **Active Directory Administrative Center** (**ADAC**), on the other hand, has a much more streamlined interface that looks and feels like the newer Server Manager that we are all becoming more and more comfortable with. Many of the functions available within the ADAC accomplish the same things that we can do through the other tools already, but it pulls these functions into a more structured interface that brings some of the most commonly utilized functions up to the surface and makes them easier to run.

One great example is right on the landing page of ADAC. A common helpdesk task in any network is the resetting of passwords for user accounts. Whether the user forgot their password, changed it recently and mistyped it, or if you are resetting a password during some other sort of troubleshooting, resetting a password for a user account typically involves numerous mouse clicks inside AD Users and Computers in order to get the job done. Now, there is a quick link called Reset Password, shown right here on the main page of the Active Directory Administrative Center. Also useful is the Global Search feature right next to it, where you can type in anything to the search field and it will scour your entire directory for results relating to your search. This is another common task in AD that previously required multiple clicks to accomplish:



If you click on the name of your domain in the left navigational tree, you will dive a little deeper into the capabilities of ADAC. As you can see, the information listed here is being pulled from Active Directory and looks like the same information you would see in AD Users and Computers. That is correct, except instead of having to right-click for every function, such as new user creations or searches, you now have some quick Tasks available on the right that can quickly launch you into accomplishing these functions. Also interesting are the links for raising the forest or domain functional level on this screen. In order to do this using the classic tools, I see that most admins accomplish this by launching AD Domains and Trusts. So, one of the big benefits of the newer ADAC tool is that it is capable of giving you a centralized management window from which you can accomplish tasks that would normally have taken multiple windows and management consoles. Do you sense a common theme throughout Windows Server 2019 with the centralized management of everything?


Active Directory Administrative Center ▸ Contoso (local) ▸
Manage
Help
Active Directory...
Overview
Contoso (local)
Dynamic Access Control
Authentication
Global Search
Contoso (local) (16)
Filter
Name
Type
Description
Builtin
builtinDom...
Computers
Container
Default container for upgr...
Contoso Computers
Organizati...
Contoso Users
Organizati...
Disabled Users
Organizati...
Domain Controllers
Organizati...
Default container for dom...
ForeignSecurityPrincipals
Container
Default container for secur...
Infrastructure
infrastructu...
Keys
Container
Default container for key o.
LostAndFound
lostAndFou...
Default container for orph..
Managed Service Accounts
Container
Default container for man..
NTDS Quotas
msDS-Quo...
Quota specifications conta.
Program Data
Container
Default location for storag.
Builtin
Organizational Unit
InetOrgPerson
Group
User
Computer
Tasks
Builtin
New
Delete
Search under this node
Properties
Contoso (local)
Change domain controller
Raise the forest functional level...
Raise the domain functional level...
Enable Recycle Bin ...
New
Search under this node
Properties

Dynamic Access Control

In addition to teaching the old dogs new tricks, Active Directory Administrative Center also brings some new functionality to the table that is not available anywhere in the classic tools. If you once again take a look at the tree to the left, you will see that the next section in the list is **Dynamic Access Control** (**DAC**). This is a technology that is all about security and governance of your files, the company data that you need to hold onto tightly and make sure it's not falling into the wrong hands. DAC gives you the ability to tag files, thereby classifying them for particular groups or uses. Then you can create Access Control policies that define who has access to these particularly tagged files. Another powerful feature of Dynamic Access Control is the reporting functionality. Once DAC is established and running in your environment, you can do reporting and forensics on your files, such as finding a list of the people who have recently accessed a classified document.

DAC can also be used to modify users' permissions based on what kind of device they are currently using. If our user Susie logs in with her company desktop on the network, she should have access to those sensitive HR files. On the other hand, if she brings her personal laptop into the office and connects it to the network, we might not want to allow access to these same files, even when providing her domain user credentials, simply because we do not own the security over that laptop. These kinds of distinctions can be made using the Dynamic Access Control policies.

Read-Only Domain Controllers (RODC)

We can't wrap up our overview of the important Active Directory tools and components without covering **Read-Only Domain Controllers** (**RODC**) in a little more detail. Typically, when installing new Domain Controllers to your network, you add the role in a way that makes them a regular, writeable, fully functional DC on your network so that it can perform all aspects of the AD DS role. There are some circumstances in which this is not the best way to go, and that is what the RODC is here to help with. This is not a separate role, but rather a different configuration of the same AD DS role that you will see when spinning through the wizard screens during the configuration of your new Domain Controller. An RODC is a specialized Domain Controller, to which you cannot write new data. They contain a cached, read-only copy of only certain parts of the directory. You can tell an RODC to keep a copy of all the credentials within your domain, or you can even tell it to only keep a list of selective credentials that are going to be important to that particular RODC. What are the reasons for using an RODC? Branch offices and DMZs are the most common I see.

If you have a smaller branch office with a smaller number of people, it may be beneficial for them to have a local Domain Controller so that their login processing is fast and efficient, but because you don't have a good handle on physical security in that little office, you would rather not have a full-blown DC that someone might pick up and walk away with. This could be a good utilization for an RODC. Another is within a secure DMZ network. These are perimeter networks typically designed

for very limited access, because they are connected to the public internet. Some of your servers and services sitting inside the DMZ network might need access to Active Directory, but you don't want to open a communications channel from the DMZ to a full Domain Controller in your network. You could stand up an RODC inside the DMZ, have it cache the information that it needs in order to service those particular servers in the DMZ, and make a much more secure domain or subdomain environment within that DMZ network.

The power of Group Policy

In a network that is based upon Windows Servers and Active Directory, it is almost always the case that the primary set of client computers are also based upon the Microsoft Windows operating systems, and that these machines are all domain-joined. Setting everything up this way not only makes sense from an organizational perspective inside Active Directory, but also allows centralized authentication across devices and applications, like we have already talked about. I know that in a couple of the examples I gave earlier in the book that I said something like, *What about when a company has a security policy in place that...* or *Make sure your servers don't get those existing security policies because...* So what are these magical **security policies** anyway, and how do I set one up?

This is the power of Group Policy. It enables you to create **Group Policy Objects** (**GPOs**) that contain settings and configurations that you want to apply to either computers or users in your Active Directory domain. Once you have created and built out a GPO with a variety of settings, you then have the option to steer that GPO in whatever direction you choose. If you have a policy that you want applied to all desktop systems, you can point it at the appropriate OU or security group in Active Directory that houses all of your domain-joined desktop computers. Or maybe you created a GPO that only applies to your Windows 7 computers; you can filter it appropriately so that only those systems are receiving the policy. And the real magic is that the issuance of these settings happens automatically, simply by those computers being joined to your domain. You don't have to touch the client systems at all in order to push settings to them via a GPO. You can tweak or lock

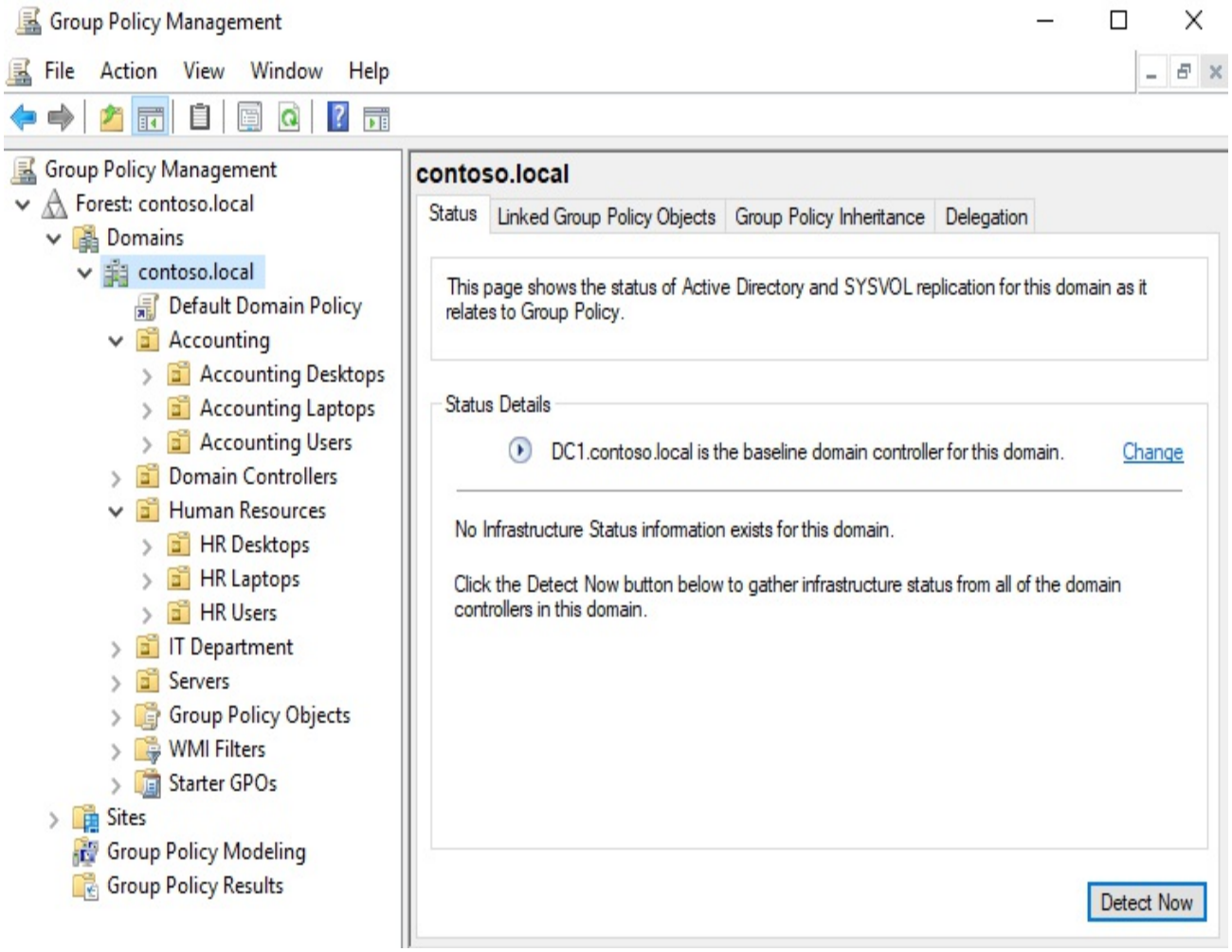
down almost anything within the Windows operating system by using Group Policy.

So, once again, I'm looking in the list of available roles on my Windows Server 2019, and I am just not seeing one called Group Policy. Correct again: there isn't one! In fact, if you have been following along with the lab setup in this book, you already have Group Policy fully functional in your network. Everything that Group Policy needs in order to work is part of Active Directory Domain Services. So, if you have a Domain Controller in your network, then you also have Group Policy on that same server, because all of the information Group Policy uses is stored inside the directory. Since the installation of the AD DS role is all we need in order to use Group Policy, and we have already done that on our Domain Controller, let's jump right in and take a look at a few things that will enable you to start utilizing Group Policy in your environment right away. I have worked with many small businesses over the years that were running a Windows Server simply because that's what everyone does, right? Whoever the IT guy or company was that set this server up for them certainly never showed them anything about GPOs, and so they have this powerful tool just sitting in the toolbox, unused and waiting to be unleashed. If you aren't already using GPOs, I want you to open that box and give it a shot.

The Default Domain Policy

First, we need to figure out where we go on our Domain Controller so that we can create and manipulate Group Policy Objects. As is the case with any of the administrative tools on a Windows Server 2019, Server Manager is the central platform for opening up your console. Click on the Tools menu from inside Server Manager, and choose Group Policy Management.

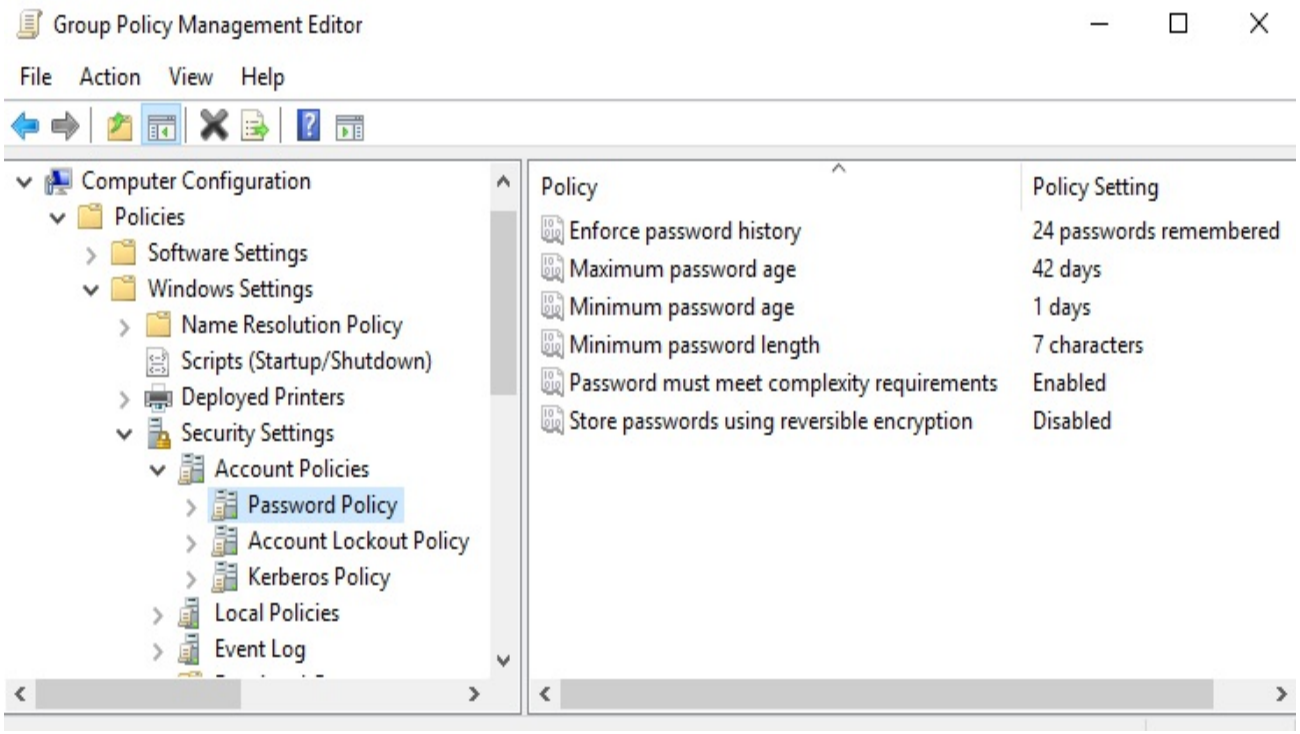
Once the console has opened, expand your Forest name from the navigational tree on the left, and then also expand out Domains and choose your domain name. Inside, you will see some familiar-looking objects. This is a list of the organizational units that you created earlier and a couple of other folders alongside your OUs:



We will talk about why the list of OUs exists here shortly, but, for now, we want to focus on a particular GPO that is typically near the top of this list, immediately underneath the name of your domain. It is called **Default Domain Policy**. This GPO is plugged into Active Directory by default during installation, and it applies to every user and computer that is part of your domain directory. Since this GPO is completely enabled right off the bat and applies to everyone, it is a common place for companies to enforce global password policies or security rules that need to apply to everyone.

With any GPO that you see in the management console, if you right-

click on that GPO and then choose Edit... you will see a new window open, and this GPO Editor contains all of the internals of that policy. This is where you make any settings or configurations that you want to be a part of that particular GPO. So, go ahead and edit your Default Domain Policy, and then navigate to Computer Configuration | Policies | Windows Settings | Security Settings | Account Policies | Password Policy:



Here, you can see a list of the different options available to you for configuring the Password Policy within your domain. Double-clicking on any of these settings lets you modify them, and that change immediately starts to take effect on all of your domain-joined computers in the network. For example, you can see that the default Minimum password length is set to **7 characters**. Many companies have already gone through much discussion about their own written policy on the standard length of passwords in the network, and in order to set up your new directory infrastructure to accept your decision, you

simply modify this field. Changing the minimum password length to 12 characters here would immediately require the change be made for all user accounts the next time they reset their passwords.

If you look along the left-hand tree of the Group Policy Management Editor, you can see that there are an incredibly large amount of settings and configurations that can be pushed out via Group Policy. While the Default Domain Policy is a very quick and easy way to get some settings configured and pushed out to everyone, tread carefully when making changes to this default policy. Every time that you make a setting change in here, remember that it is going to affect everyone in your domain, including yourself. Many times you will be creating policies that do not need to apply to everyone, and in those cases, it is highly recommended that you stay away from the Default Domain Policy, and instead set up a brand new GPO for accomplishing whatever task it is that you are trying to put into place. In fact, some administrators recommend never touching the Default Domain Policy at all, and making sure to always utilize a new GPO whenever you have new settings to push into place. In practice, I see many companies using the built-in Default Domain Policy for password complexity requirements, but that's it. All other changes or settings should be included in a new GPO.

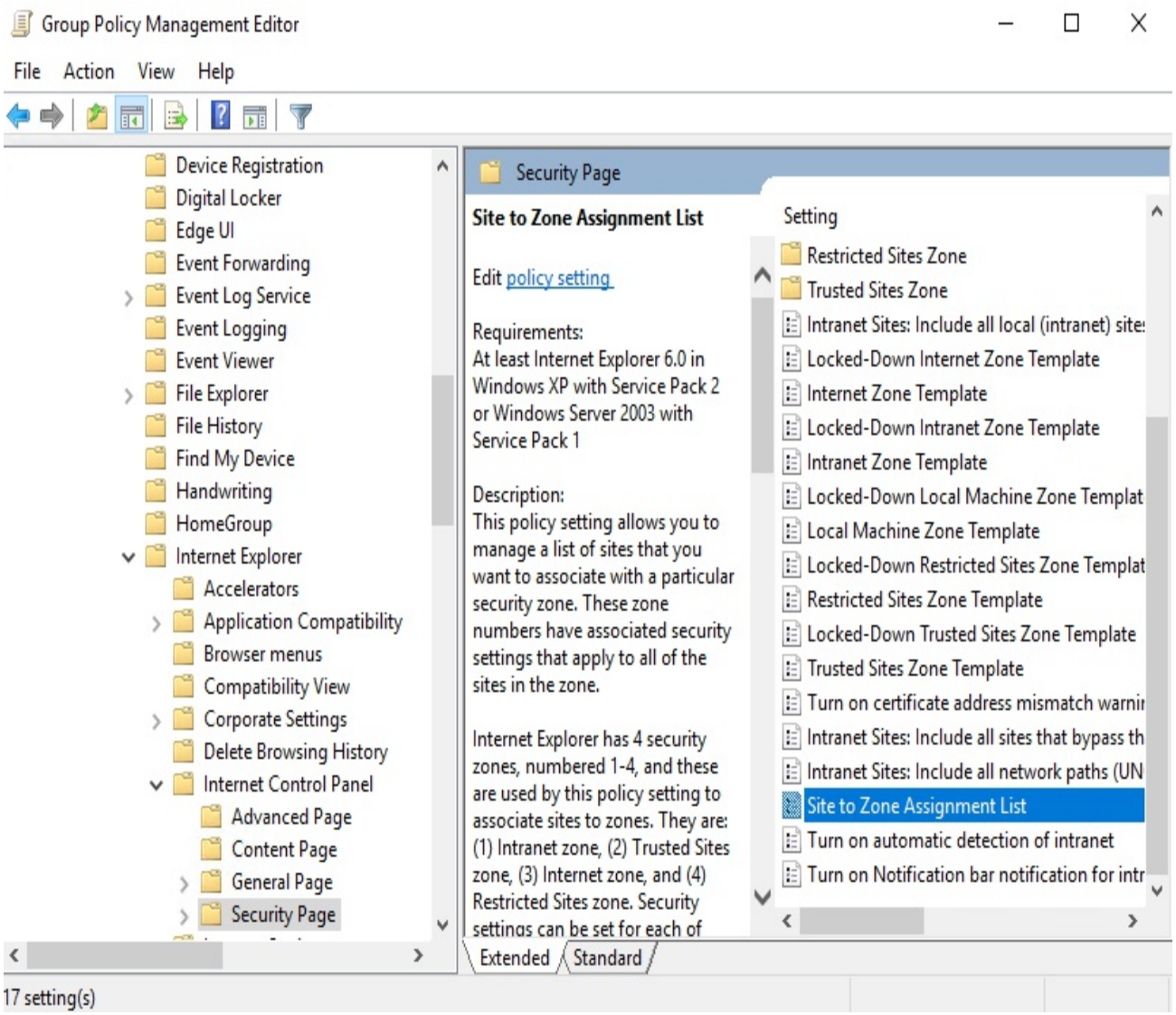
Creating and linking a new GPO

If the best practice in general is to build a new GPO when we need to apply some settings, we'd better take a minute and cover that process. For this example, we are going to create a new GPO that plugs a list of trusted sites into Internet Explorer on our desktop computers. If you run a web application in your network that needs to run JavaScript or ActiveX controls, or something like that, it may be required that the website is part of the trusted sites list inside Internet Explorer in order for it to run properly. You could print off an instructions page for the helpdesk on how to do this on each computer, and have them spend the time doing it for every user who calls in because they cannot access the application. Or you could simply create a GPO that makes these changes for you automatically on every workstation, and save all of those phone calls. This is just one tiny example of the power that Group Policy possesses, but it's a good example because it is something useful, and is a setting that is sort of buried way down in the GPO settings, so that you can get a feel for just how deep these capabilities go.

Inside the Group Policy Management console, right-click on the folder called Group Policy Objects, and choose New. Name your new GPO—mine is called Adding Trusted Sites—and then click on OK. Your new GPO now shows up in the list of available GPOs, but it is not yet applying to anyone or any computers. Before we assign this new GPO to anyone, let's plug in that list of trusted sites so that the policy contains our configuration information. We have a new policy, but it's currently void of any settings.

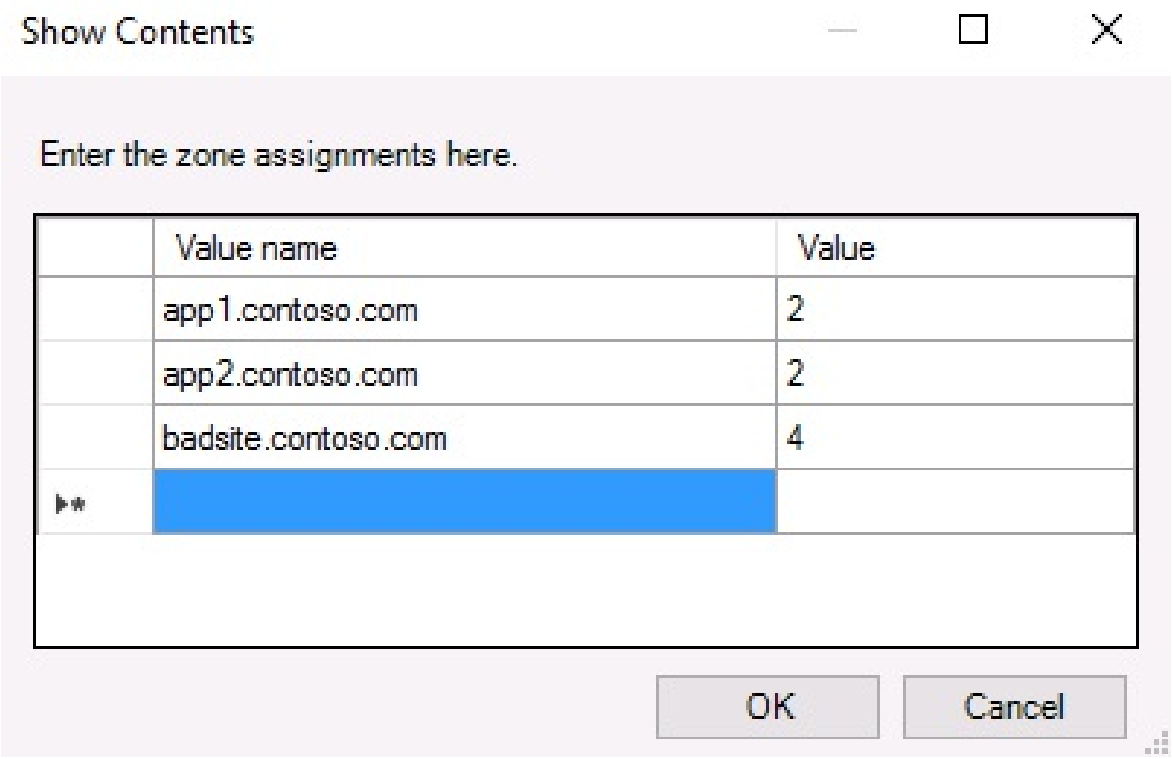
Right-click on the new GPO, and choose Edit.... Now navigate to

Computer Configuration | Policies | Administrative Templates | Windows Components | Internet Explorer | Internet Control Panel | Security Page. See, I told you it was buried in there!



Now, double-click on Site to Zone Assignment List, and set it to Enabled. This allows you to click on the Show... button, within which you can enter websites and give them zone assignments. Each GPO setting has a nice descriptive text to accompany it, telling you exactly what that particular setting is for and what the options mean. As you

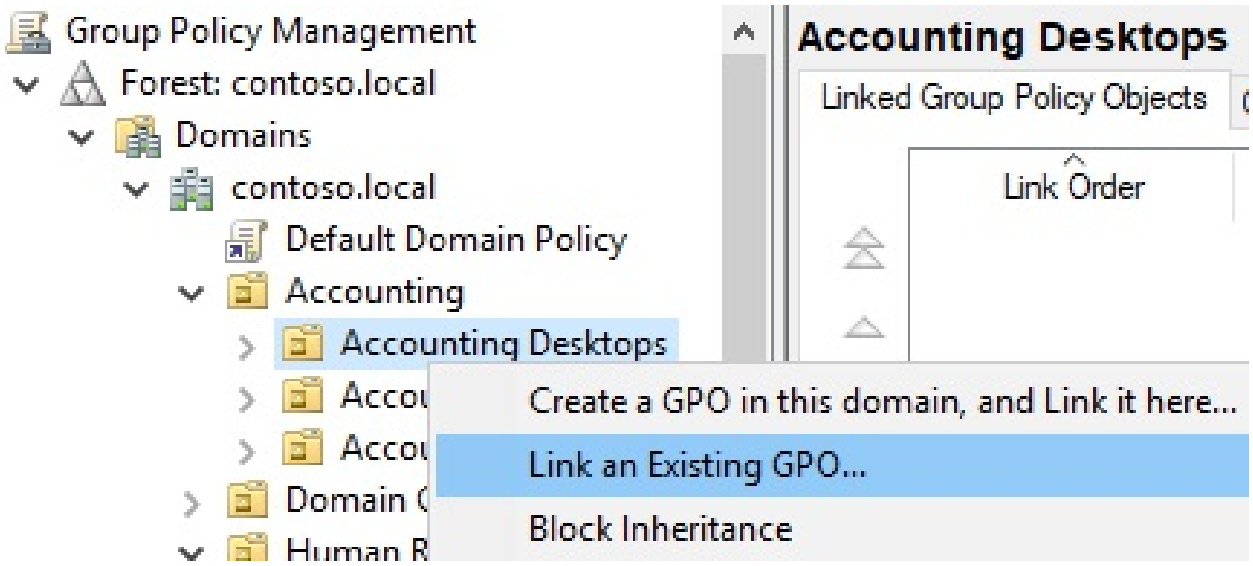
can see in the text for this one, in order to set my websites to be trusted sites, I need to give them a zone assignment value of 2. And, just for fun, I also added in a site that I do not want to be accessible to my users, and gave it a zone value of 4 so that badsite.contoso.com is a member of the restricted sites zone on all of my desktop computers. Here is my completed list:



Are we done? Almost. As soon as I click on the OK button, these settings are now stored in my Group Policy Object and are ready to be deployed, but at this point in time I have not assigned my new GPO to anything, so it is just sitting around waiting to be used.

Back inside the Group Policy Management console, find the location to which you want to **link** this new GPO. You can link a GPO to the top of the domain similar to the way that the Default Domain Policy works, and it would then apply to everything below that link. So, in effect, it

would start applying to every machine in your domain network. For this particular example, we don't want the Trusted Site's settings to be quite so global, so we are going to create our link to a particular OU instead. That way, this new GPO will only apply to the machines that are stored inside that OU. I want to assign this GPO to my Accounting Desktops OU that I created earlier. So I simply find that OU, right-click on it, and then choose Link an Existing GPO...:



I now see a list of the GPOs that are available for linking. Choose the new Adding Trusted Sites GPO, and click on OK, and that's it! The new GPO is now linked to the Desktop Computers OU, and will apply those settings to all machines that I place inside that OU.

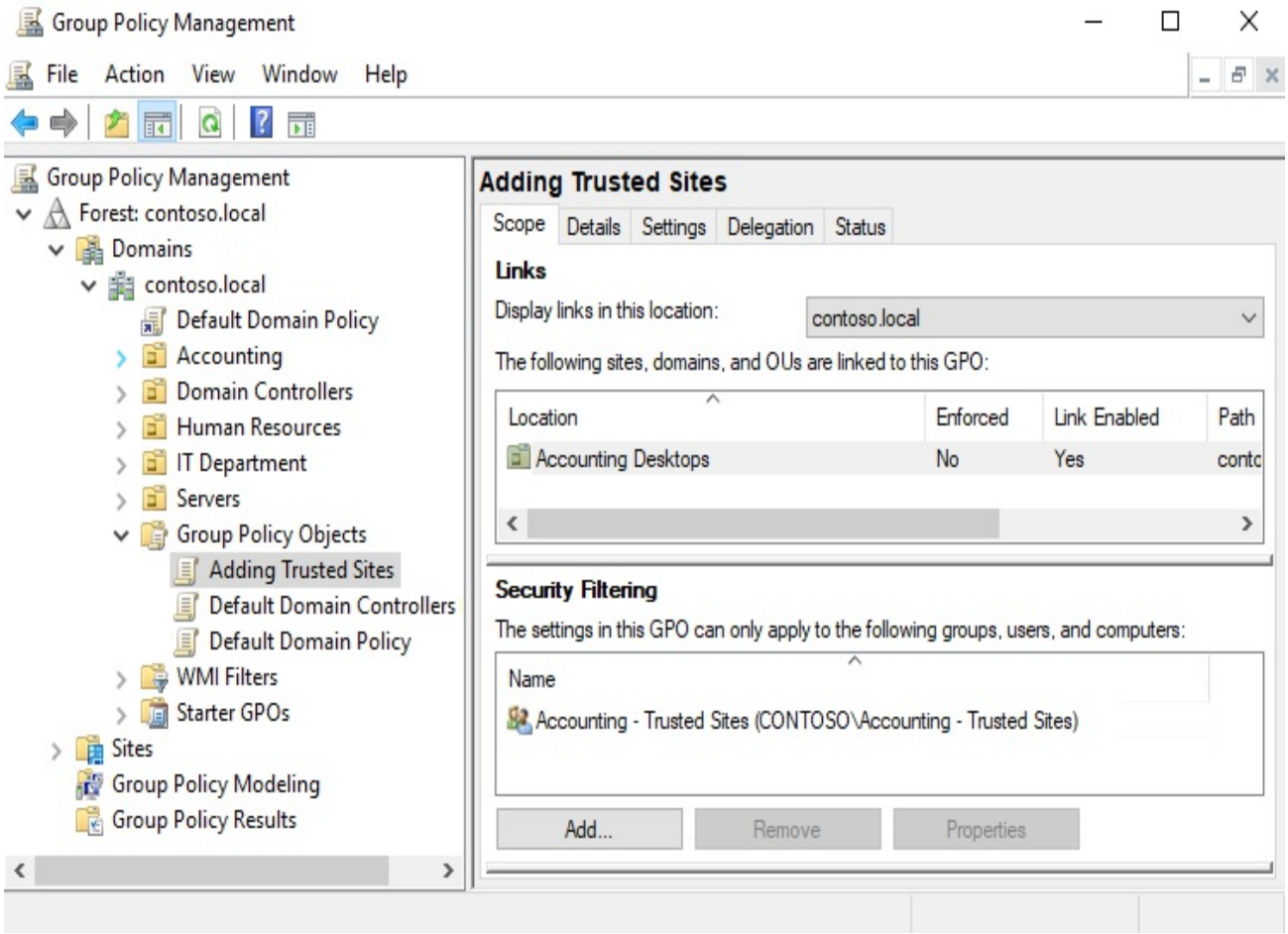
> *You can link a GPO to more than one OU. Just follow the same process again, this time choosing a different OU where you want to make the link, and that GPO will now apply to both OUs that have active links. You can also remove individual links by clicking on the Group Policy Object itself in order to view its properties.*

Filtering GPOs to particular devices

Now that you have created a GPO and linked it to particular OU, you have enough information in order to really start using Group Policy in your environment. Using links to determine what machines or users get what policies is the most common method that I see admins use, but there are many circumstances where you might want to take it a step further. What if you had a new GPO and had it linked to an OU that contained all of your desktop computers, but then decided that some of those machines needed the policy and some did not? It would be a headache to have to split those machines up into two separate OUs just for the purpose of this policy that you are building. This is where **GPO Security Filtering** comes into play.

Security filtering is the ability to filter a GPO down to particular Active Directory objects. On any given GPO in your directory, you can set filters so that the GPO only applies to particular users, particular computers, or even particular groups of users or computers. I find that using groups is especially useful. So, for our preceding example, where we have a policy that needs to apply only to some desktop computers, we could create a new Security Group inside Active Directory and add only those computers into the group. Once the GPO has been configured with that group listed in the filtering section, that policy would only apply to machines that are part of that group. In the future, if you needed to pull that policy away from some computers or add it to new computers, you simply add or remove machines from the group itself, and you don't have to modify the GPO at all.

The Security Filtering section is displayed when you click on any GPO from inside the Group Policy Management console. Go ahead and open GPMC, and simply click once on one of your GPOs. On the right-hand side, you see the Scope open for that policy. The section on top shows you what Links are currently active on the policy, and the bottom half of the screen displays the Security Filtering section. You can see here that I have linked my GPO to the Accounting Desktops OU, but I have set an additional security filter so that only machines that are part of the Accounting - Trusted Sites group will actually receive the settings from my GPO:



Another cool feature that is just a click away is the Settings tab on this same screen. Click on that tab, and it will display all of the configurations currently set inside your GPO. This is very useful for checking over GPOs that someone else may have

created, to see what settings are being modified.

As I mentioned earlier, you could take any one of these management consoles or topics regarding the core infrastructure services inside Windows Server and turn that topic into its own book. I actually had the opportunity to do exactly that with Group Policy. If you are interested in discovering more about Group Policy and all of the ways that it can be used to secure your infrastructure, check out Packt's *Mastering Windows Group Policy* (https://www.packtpub.com/networking-and-servers/mastering-windows-group-policy).

HivaNetwork.Com

Domain Name System (DNS)

If we consider Active Directory Domain Services to be the most common and central role in making our Microsoft-centric networks function, then the **Domain Name System** (**DNS**) role slides in at number two. I am yet to meet an admin who has chosen to deploy a domain without deploying DNS at the same time: they always go hand-in-hand.

> *DNS is a service that is typically provided by a Windows Server, but doesn't have to be. There are many different platforms available all the way from Linux servers to specialized hardware appliances designed specifically for managing DNS within a network that can be utilized for this role. For most Microsoft-centric networks, and for the purposes of this book, we will assume that you want to use Windows Server 2019 for hosting the DNS role.*

DNS is similar to Active Directory in that it is a structured database that is often stored on Domain Controller servers, and distributed automatically around your network to other Domain Controller/DNS servers. Where an AD's database contains information about the domain objects themselves, DNS is responsible for storing and resolving all of the names on your network. What do I mean by names? Whenever a user or computer tries to contact any resource by calling for a name, DNS is the platform responsible for turning that name into something else in order to get the traffic to the correct destination. You see, the way that traffic gets from client to server is via networking, and typically via the TCP/IP stack, using an IP address to get to its destination. When I open an application on my computer to access some data that resides on a server, I could configure the application so that it communicates directly to my server by using the server's IP address on the network.

If I plug `10.10.10.15` into my application configuration, it would open successfully. If I set up hundreds of different computers this way, all pointing to IP addresses, it would work fine for a while. But the day will come when, for whatever reason, that IP address might need to change. Or perhaps I add a second server to share the load and handle my increased user traffic. What to do now? Re-visit every client computer and update the IP address being used? Certainly not. This is one of the reasons that DNS is critical to the way that we design and manage our infrastructures. By using DNS we can employ names instead of IP addresses. With DNS, my application can be configured to talk to `Server01` or whatever my server name is, and if I need to change the IP address later, I simply change it inside the DNS console to the updated IP address and immediately all of my client computers will start resolving the `Server01` name to the new IP address. Or I can even use a more generic name, such as intranet, and have it resolve across multiple different servers. We will discuss that a little bit more shortly.

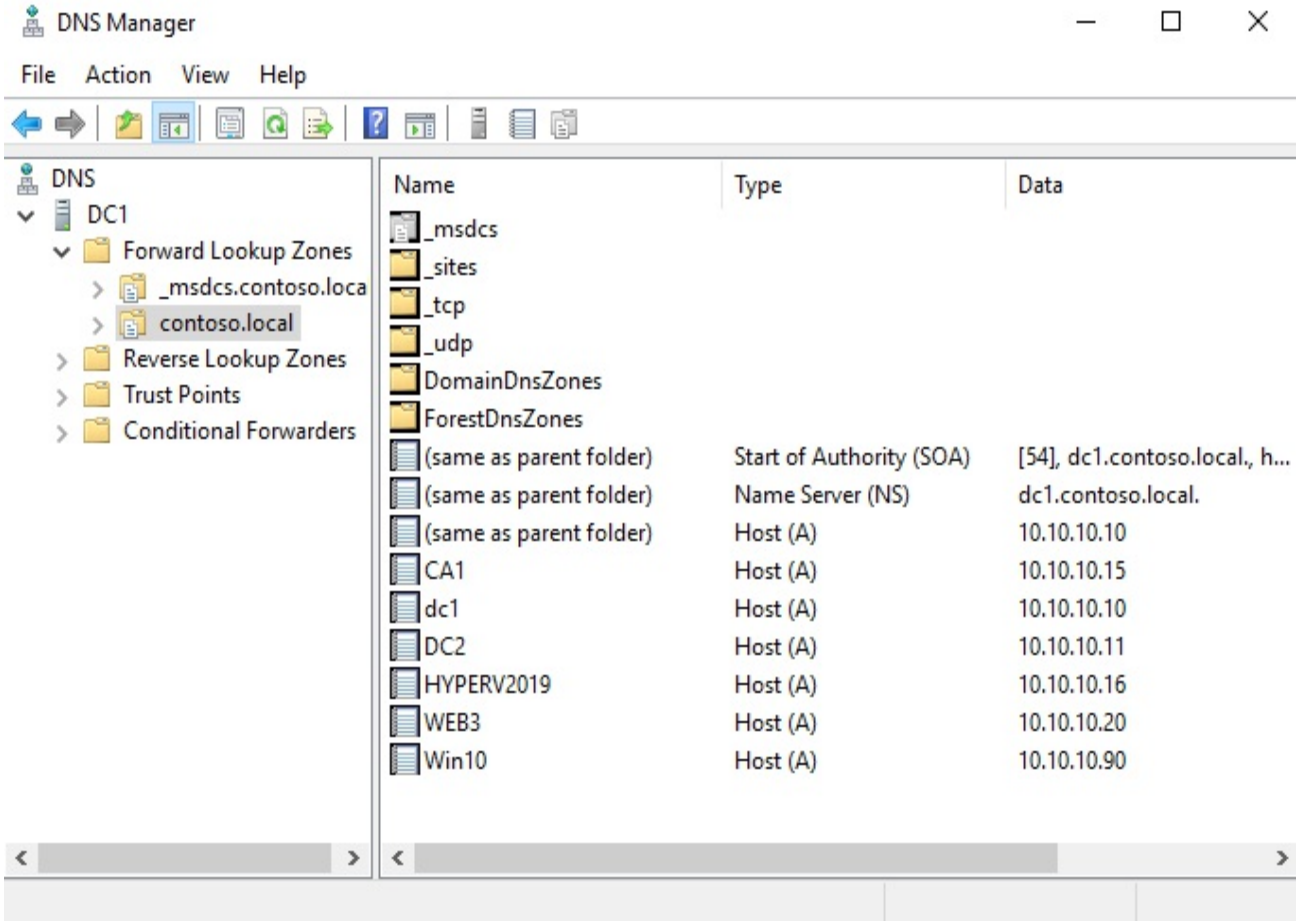
Any time that a computer makes a call to a server, or service, or website, it is using DNS in order to resolve that name to a more useful piece of information in order to make the network connection happen successfully. The same is true both inside and outside of corporate networks. On my personal laptop right now, if I open Internet Explorer and browse to `https://www.bing.com/`, my internet provider's DNS server is resolving `http://www.bing.com/` to an IP address on the internet, which is the address that my laptop communicates with and so that page opens successfully. When we are working inside our own corporate networks, we don't want to rely on or trust a public provider with our internal server name information, and so we build our own DNS servers inside the network. Since DNS records inside a domain network are almost always resolving names to objects that reside inside Active Directory, it makes sense then that DNS and AD DS would be tightly integrated. That rings true in the majority of Microsoft networks, where it is a very common practice to install both the AD DS role, plus the DNS role, on

your Domain Controller servers.

Different kinds of DNS records

Having installed our DNS role on a server in the network, we can start using it to create DNS records, which resolve names to their corresponding IP addresses, or other pieces of information needed in order to route our traffic around the network. Assuming that you are working in a domain network, you may be pleasantly surprised to see that a number of records already exist inside DNS, even though you haven't created any of them. When you are running Active Directory and DNS together, the domain-join process that you take with your computers and servers self-registers a DNS record during that process.

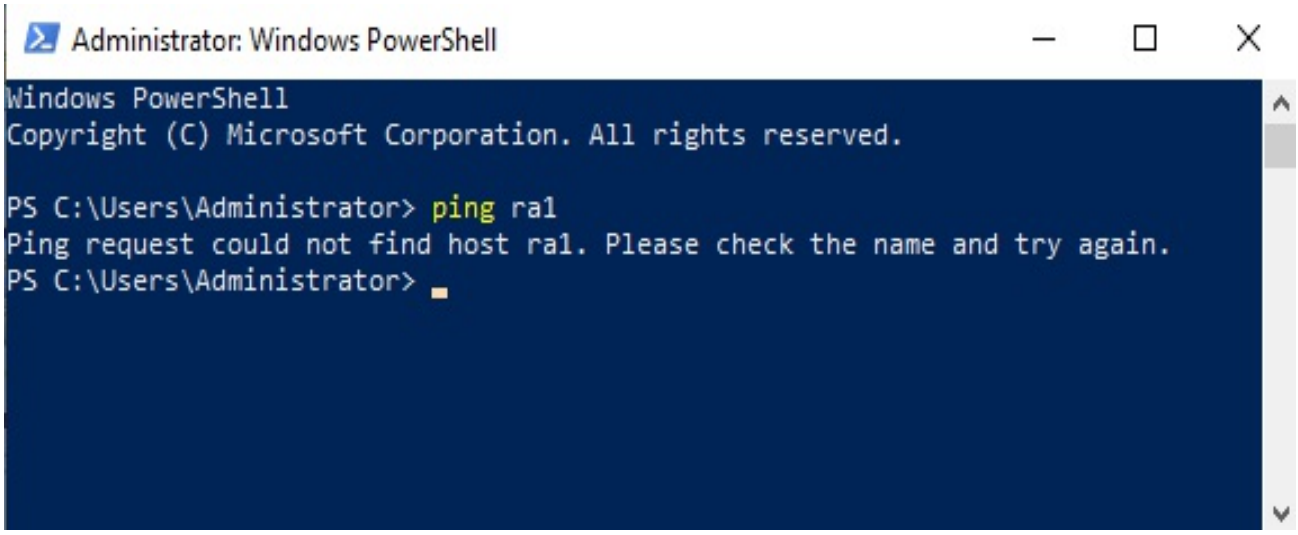
I have not yet created any DNS records in my new lab environment, not purposefully anyway, and yet when I open the DNS Manager console from inside the Tools menu of Server Manager, I can see a handful of records already existing. This is because when I joined each of these machines to the domain, it automatically registered these records for me so that the new servers and clients were immediately resolvable within our domain:


DNS Manager
File
Action
View
Help
DNS
DC1
Forward Lookup Zones
_msdcs.contoso.loca
contoso.local
Reverse Lookup Zones
Trust Points
Conditional Forwarders
Name
Type
Data
_msdcs
_sites
_tcp
_udp
DomainDnsZones
ForestDnsZones
(same as parent folder) Start of Authority (SOA) [54], dc1.contoso.local., h...
(same as parent folder) Name Server (NS) dc1.contoso.local.
(same as parent folder) Host (A) 10.10.10.10
CA1 Host (A) 10.10.10.15
dc1 Host (A) 10.10.10.10
DC2 Host (A) 10.10.10.11
HYPERV2019 Host (A) 10.10.10.16
WEB3 Host (A) 10.10.10.20
Win10 Host (A) 10.10.10.90

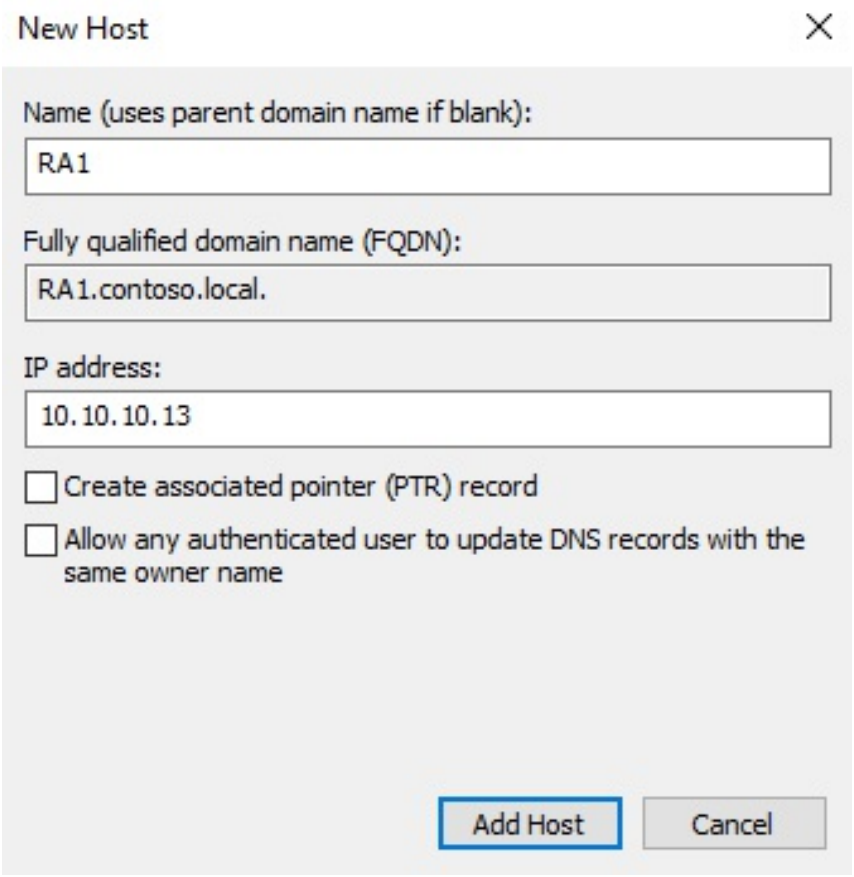
Host record (A or AAAA)

The first kind of DNS record we are looking at is the most common type that you will work with. A **host record** is the one that resolves a particular name to a particular IP address. It's pretty simple, and for most of the devices on your network this will be the only kind of record that exists for them inside DNS. There are two different classes of host record that you should be aware of, even though you will likely only be using one of them for at least a few more years. The two different kinds of host records are called an **A record**, and an **AAAA record**, which is pronounced **Quad A**. The difference between the two? A records are for IPv4 addresses and will be used in most companies for years to come. AAAA records serve the exact same purpose of resolving a name to an IP address, but are only for IPv6 addresses, and will only be useful if you use IPv6 in your network.

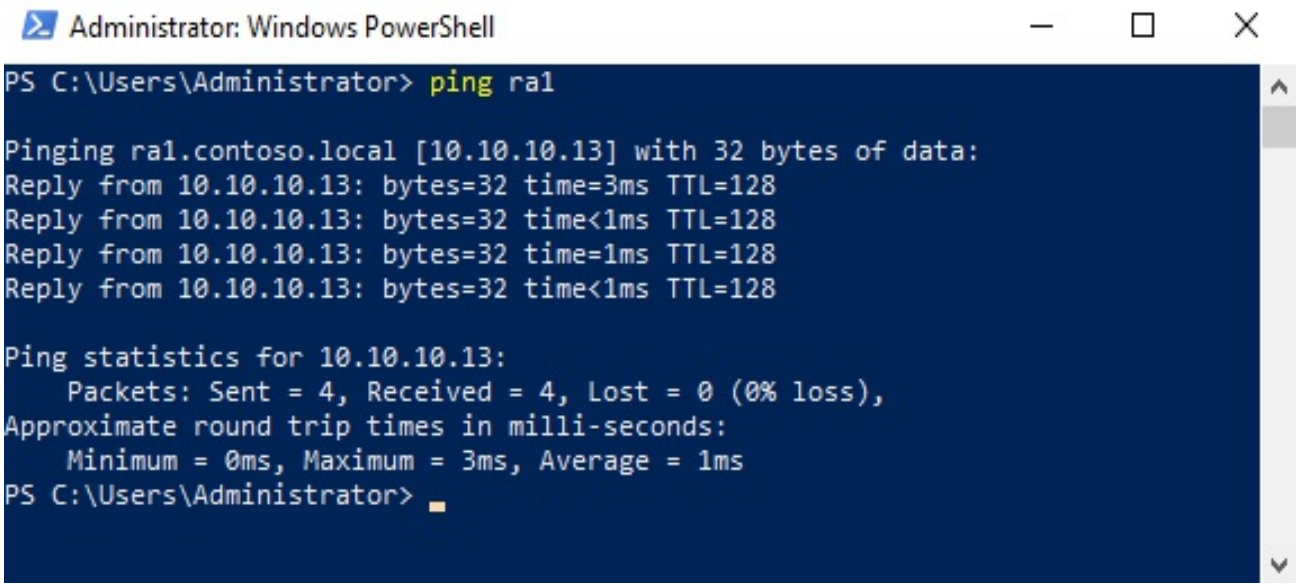
In the previous screenshot, you can see some Host (A) records that were self-created when those machines joined our domain. I also have another server running on my network that has not yet been domain joined, and so it has not self-registered into DNS. This server is called `RA1`, but if I log in to any other system on my network, I fail to contact my RA1 server, since that name is not yet plugged into DNS:



For now, I am going to choose not to join this server to the domain, so that we can manually create a DNS record for it and make sure that I am able to resolve the name properly after doing that. Back inside DNS Manager on my DNS server, right-click on the name of your domain listed under the Forward Lookup Zones folder, and then choose New Host (A or AAAA). Inside the screen to create a new host record, simply enter the name of your server, and the IP address that is configured on its network interface:



Now that we have created this new host record, we should immediately be able to start resolving this name inside our domain network. Moving back to the client machine from which I was trying to ping `RA1` earlier, I'll try the same command again, and this time it does resolve and reply successfully:

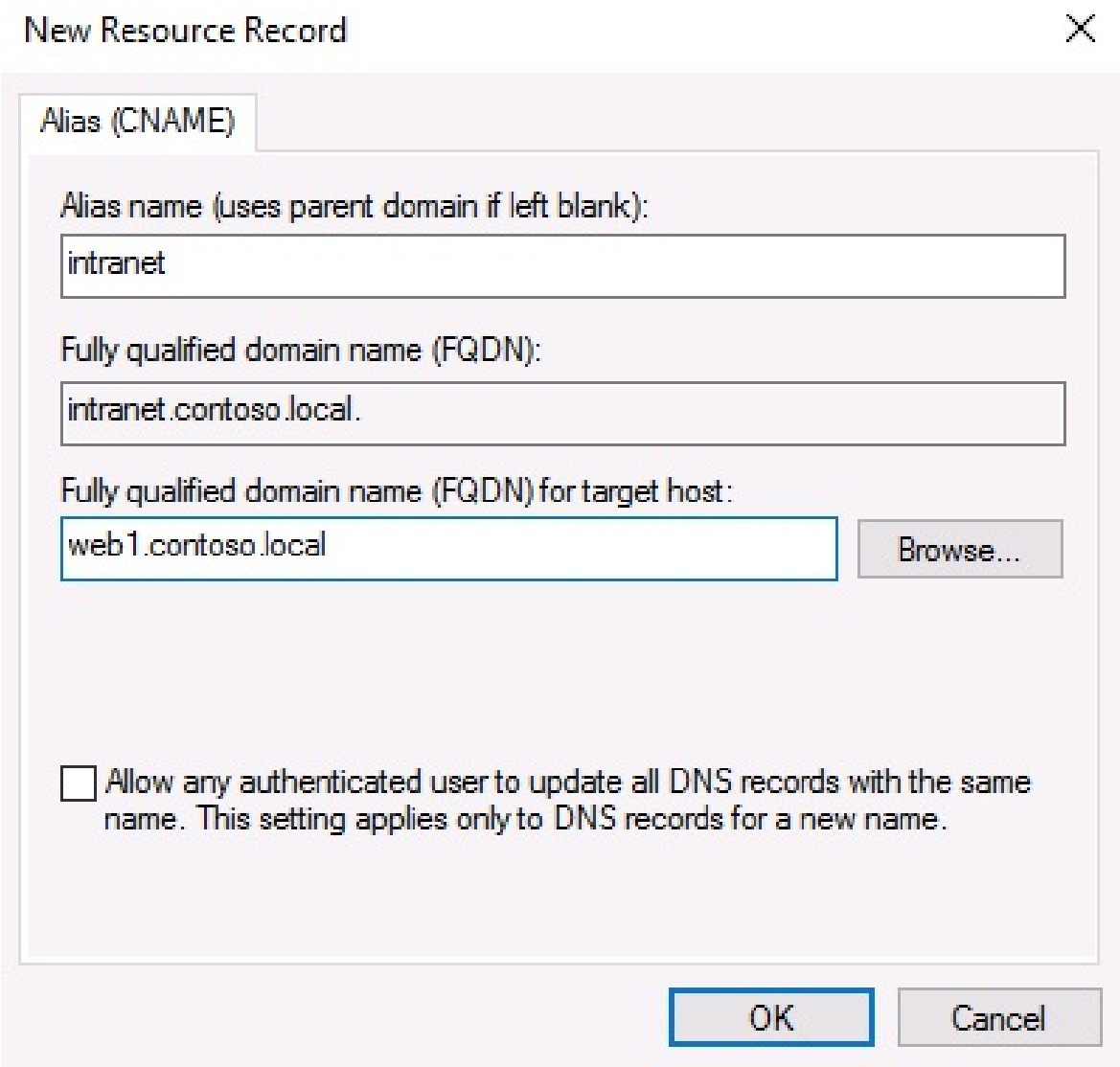

Administrator: Windows PowerShell
PS C:\Users\Administrator> ping ra1
Pinging ra1.contoso.local [10.10.10.13] with 32 bytes of data:
Reply from 10.10.10.13: bytes=32 time=3ms TTL=128
Reply from 10.10.10.13: bytes=32 time<1ms TTL=128
Reply from 10.10.10.13: bytes=32 time=1ms TTL=128
Reply from 10.10.10.13: bytes=32 time<1ms TTL=128
Ping statistics for 10.10.10.13:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 3ms, Average = 1ms
PS C:\Users\Administrator>

ALIAS record - CNAME

Another useful type of DNS record is **CNAME**, which more commonly these days is called the **ALIAS record**. This is a record that you can create that takes a name, and points it at another name. It sounds a little silly at first glance, because, in the end, you are still going to have to resolve your final name to an IP address in order to get the traffic where it needs to go, but the purposes of an ALIAS record can be vast. A good example to portray the usefulness of an ALIAS record is when you are running a web server that is serving up websites within your network. Rather than force all of your users to remember a URL like `http://web1.contoso.local` in order to access a website, we could create an ALIAS record called **intranet**, and point it at `web1`. This way, the more generalized intranet record can always be utilized by the client computers, which is a much friendlier name for your users to remember.

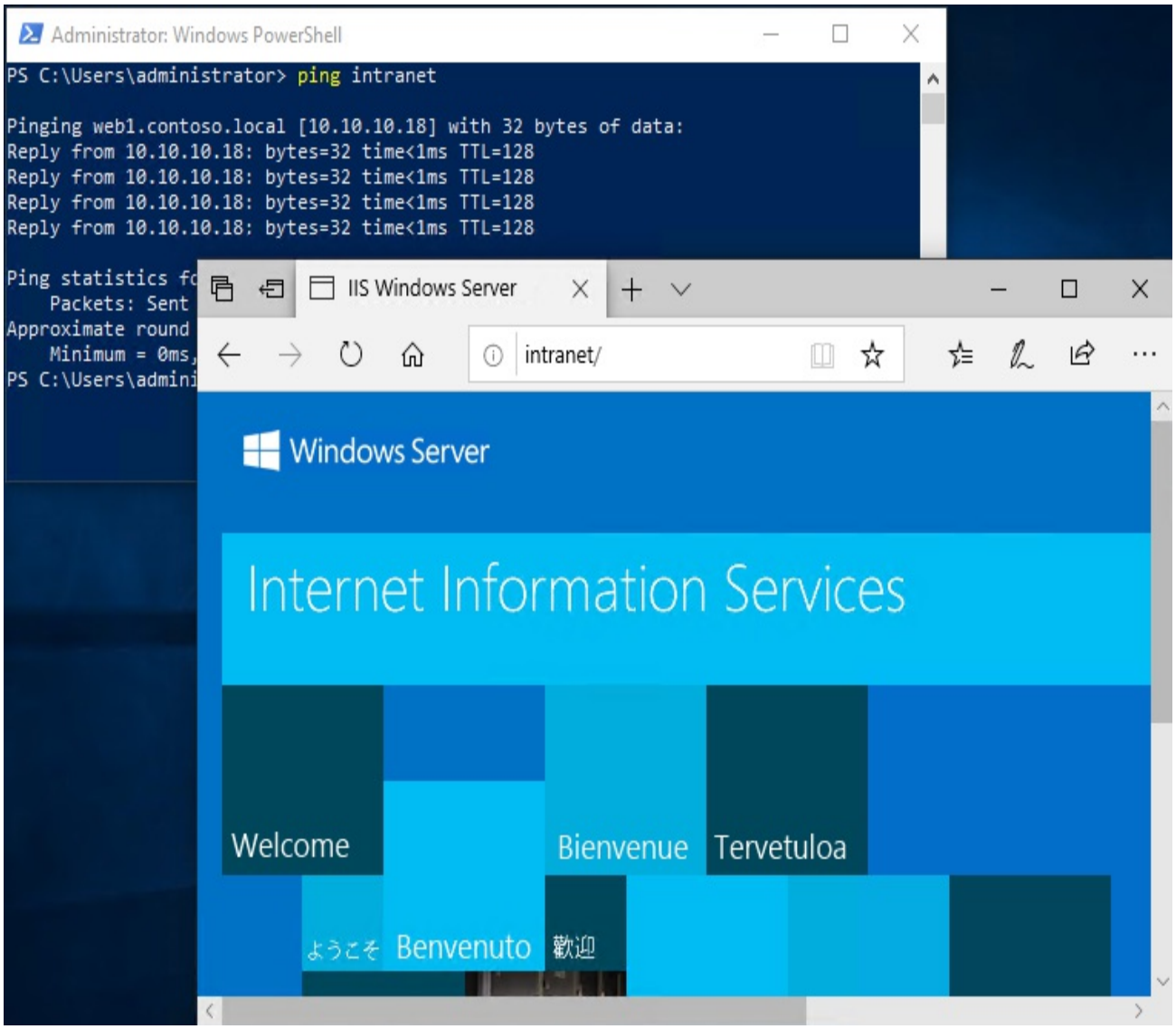
In addition to creating a happier user experience with this new DNS record, you have, at the same time, created some additional administrative flexibility because you can easily change the server components that are running beneath that record, without having to adjust any settings on the client machines or re-train employees on how to access the page. Need to replace a web server? No problem, just point the ALIAS record at the new one. Need to add another web server? That's easy too, as we can create multiple ALIAS records, all with the same intranet name, and point them at the different web servers that are in play within the environment. This creates a very simple form of load balancing, as DNS will start to round-robin the traffic among the different web servers, based on that intranet CNAME record.

In fact, rather than continue to talk about this, let's give it a try. I have a website running on exactly that URL in my environment, but currently I can only access it by typing in `http://web1.contoso.local`. Inside DNS, I am going to create an ALIAS record that redirects `intranet` to `web1`:



Now when I ping `intranet`, you can see that it resolves to my `web1` server. And when accessing the web page, I can simply type the word `intranet` into my address bar inside Internet Explorer in order to launch my page. The website itself is not aware of the name change being made, so I

didn't have to make any modifications to the website, only within DNS:

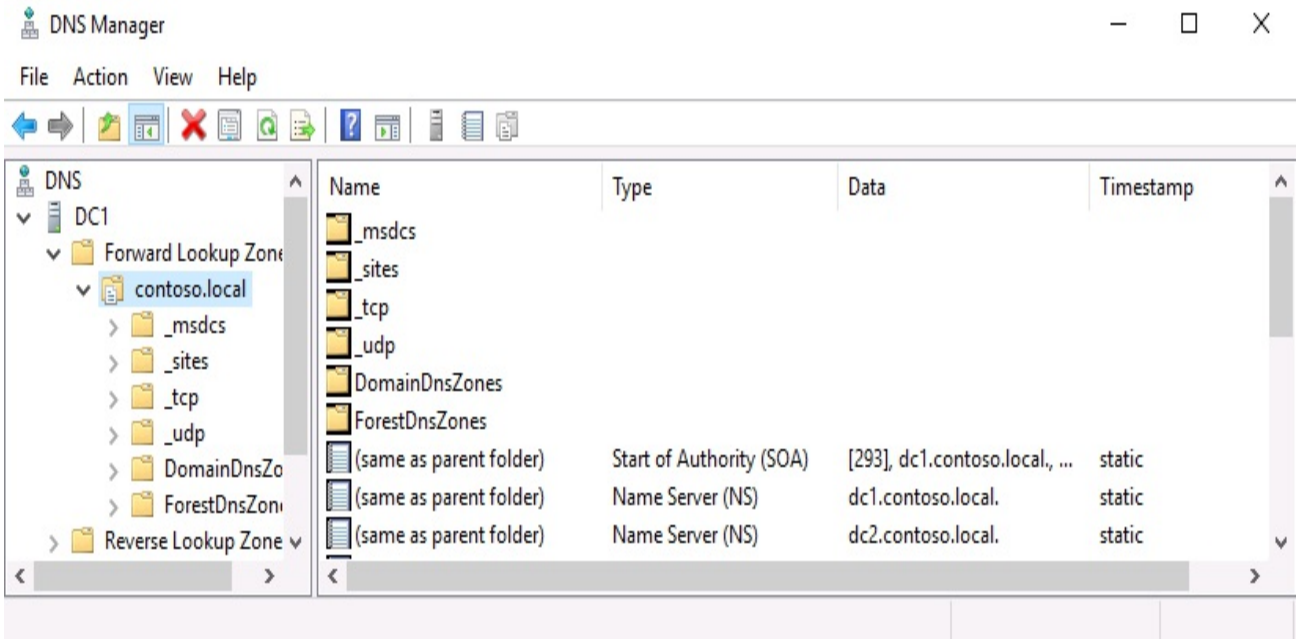


Mail Exchanger record (MX)

A third type of DNS record is called a **Mail Exchanger** (**MX**) record. In your everyday duties, you would not have to encounter or configure MX records nearly as much as A or CNAME records, but they are important to understand nonetheless. An MX record is all about email services and delivery. Whatever domain name follows the "@" in your email address, the DNS servers that are handling that domain name must contain an MX record telling the domain where to point for its mail services. MX records are only used with public DNS, for name resolutions happening over the internet. For companies hosting their own email on local Exchange servers, your public DNS servers will contain an MX record that points at your Exchange environment. For companies hosting their email in a cloud service, your public DNS records would need to contain an MX record that directs email traffic toward the cloud provider that is hosting your mailboxes.

Name Server (NS) record

Here is another that you don't have to deal with on a day-to-day basis, but you should still know what it's there for. An NS record is an identifier within a DNS zone that tells it which **Name Servers** (which are your DNS servers) to use as the authorities for that zone. If you look at the NS records listed in your DNS right now, you will recognize that it is calling out the names of your DNS servers on the network. When you add a new DC/DNS server to your domain, a new NS record for this server will be automatically added into your DNS zone:



There are many other possible types of records that can be stored and used within a DNS database, but they are not generally relevant to the common server administrator in a typical Microsoft-driven network.

ipconfig /flushdns

Just one final note to finish out this section. I have been saying things like *Now when I do this...* or *Immediately following this change...* and if you are creating some of your own records, you may have noticed that it sometimes takes a while for your client computers to recognize these new DNS records. That is normal behavior, and the time that it takes before your change rolls around to the whole network will depend entirely on how large your network is and how Active Directory replication is configured. When you create a new DNS record on one Domain Controller, your new record needs to replicate itself around to all of the other DCs in your network. This process alone can take upwards of a couple hours if AD is not configured for faster replication. Typically, it only takes a few minutes. And then, once the new record exists on all of your DC servers, your clients may still take a little bit of time to utilize the new record, because client computers in a domain network hold onto a cache of DNS data. This way, they don't have to reach out to the DNS server for every single name resolution request. They can more quickly refer to their local cache in order to see what the information was from the last time they checked in with the DNS server. If you are trying to immediately test out a new DNS record that you just created and it's not working, you may want to try to run the `ipconfig /flushdns` command on your client computer. This forces the client to dump its locally cached copies of DNS resolver records, and go grab new information that is current from the DNS server. After flushing your cache, the record will more than likely start working properly.

DHCP versus static addressing

IP addresses on your network are sort of like home addresses on your street. When you want to send a package to someone, you write their address on the front of the package and set it in the mailbox. In the same way, when your computer wants to send data to a server or another device on a network, each of those devices has an IP address that is used for the delivery of those packets. We know that DNS is responsible for telling the machines which name resolves to which IP address, but how do those IP addresses get put into place on the servers and computers in the first place?

Static addressing is simply the process of configuring IP addresses on your system manually, using your own hands as the configuration tool in order to plug all of your IP address information into the NIC settings on that device. While this is a quick and easy way to get network traffic flowing between a few endpoints, by giving them each an IP address, it is not scalable. We do often statically address our servers as a way of making sure that those IP addresses are not subject to change, but what about on the client and device side? Even in a small company with 10 employees, each person may have a desktop and a laptop, there are likely going to be print servers on the network also needing IP addresses, and you may have a wireless network where employees or even guests can connect phones and other devices in order to gain internet access. Are you going to assign IP addresses by hand to all of these devices? Certainly not.

Our answer to this problem is the **Dynamic Host Configuration Protocol** (**DHCP**). This is a protocol that is designed to solve our exact

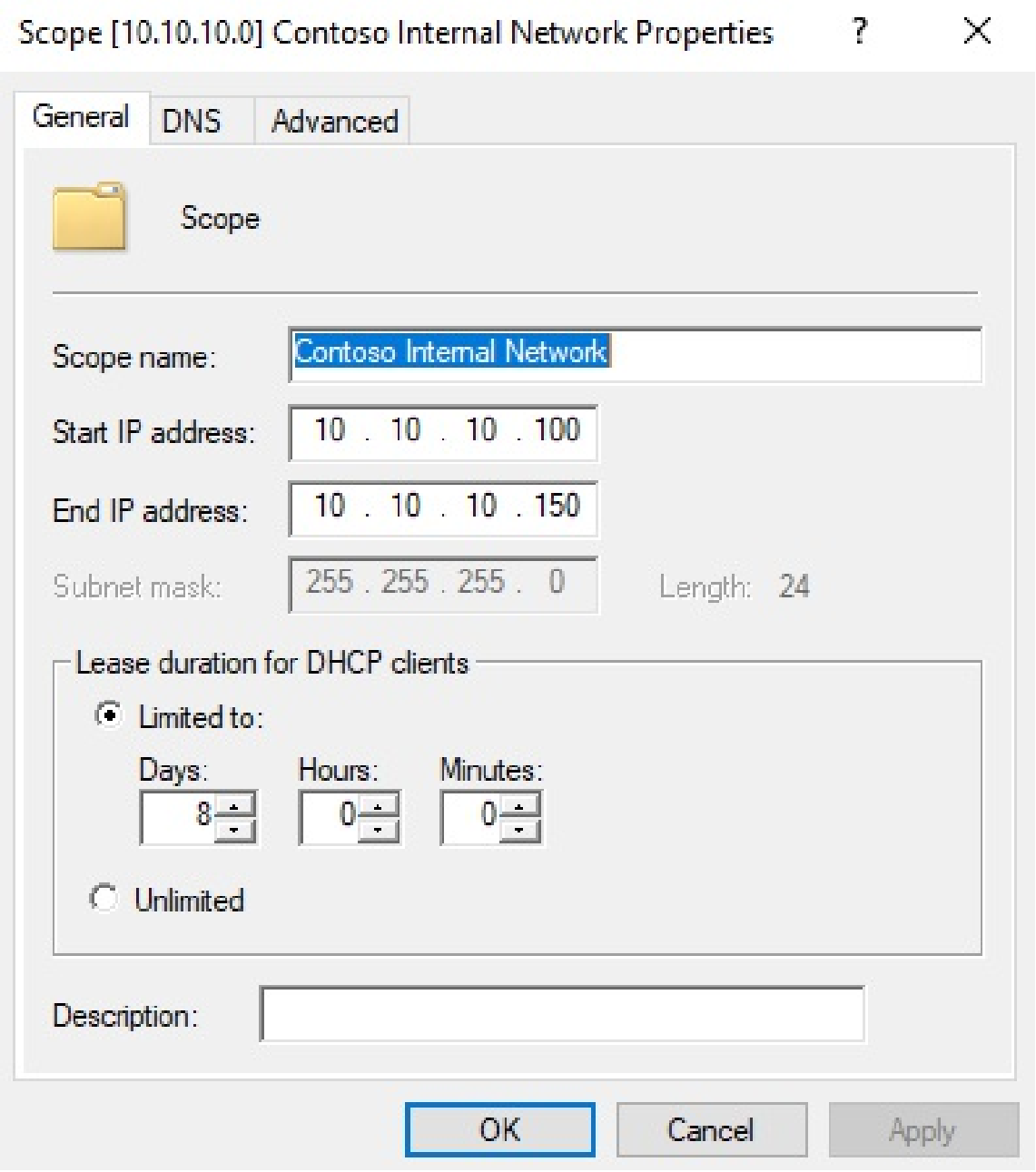
problem by providing the ability for machines and devices to be plugged into your network and automatically obtain IP addressing information. Almost any user on any device in the entire world uses DHCP every day without even realizing it. When you connect your laptop or smartphone to a Wi-Fi router to gain internet access, a DHCP server has given you the ability to route traffic on that Wi-Fi network by assigning you IP addressing information. Often, in the case of a public Wi-Fi, your DHCP server is actually running on the router itself, but in our businesses where Windows Server rules the datacenter, our DHCP services are most often hosted on one or more servers across the network.

The DHCP scope

In the new Windows Server 2019 lab environment I have been building, so far I have been statically assigning IP addresses to all of the servers that are being built. This is starting to get old, and hard to keep track of. When the first Domain Controller was configured, I actually installed the DHCP role onto it, but haven't told it to start doing anything yet. What does a DHCP server need in order to start handing out IP addresses? It needs to know what IP addresses, subnet mask, default gateway, and DNS server addresses are within your network so that it can package that up and start handing the information out to the computers who request it. This package of information inside the DHCP server is called a **DHCP scope**. Once we define our scope, the DHCP server will start handing out IP addresses from that scope to our new servers and computers which do not already have static addresses defined.

Once again, we need to launch a management tool on our Windows Server 2019, and once again, the easiest way to launch that is by using the Tools menu inside Server Manager. Go ahead and launch the DHCP console. Inside, you will see the name of your server where the DHCP server is running. Expand that, and you have options for both IPv4 and IPv6. Yes, this means that you can use this DHCP server to hand out both IPv4 addresses, as well as IPv6 addresses for those of you who are testing out IPv6, or have plans to in the future. For now, we are sticking with good old IPv4, and so I can right-click on IPv4 and choose to create a New Scope. This launches a New Scope Wizard that walks you through the few pieces of information that the DHCP server needs in order to create a scope that is ready to start handing out IP addresses inside your

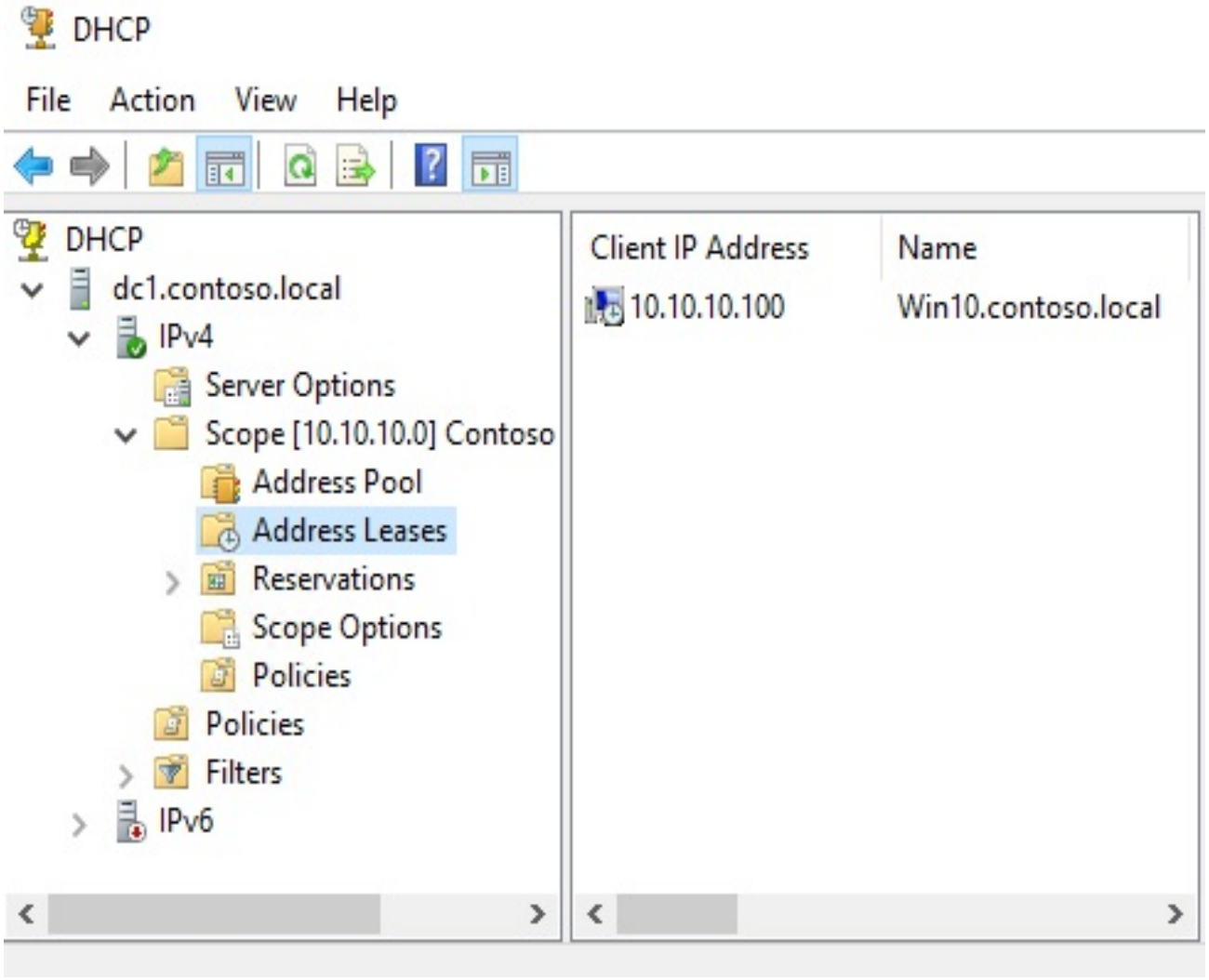
network. I am setting my new scope to hand out IP addresses from `10.10.10.100` through `10.10.10.150`:



As soon as you finish creating your scope, it is immediately active and any computer in your network whose NIC is configured to grab an

address automatically from a DHCP server will start doing so against this new DHCP server.

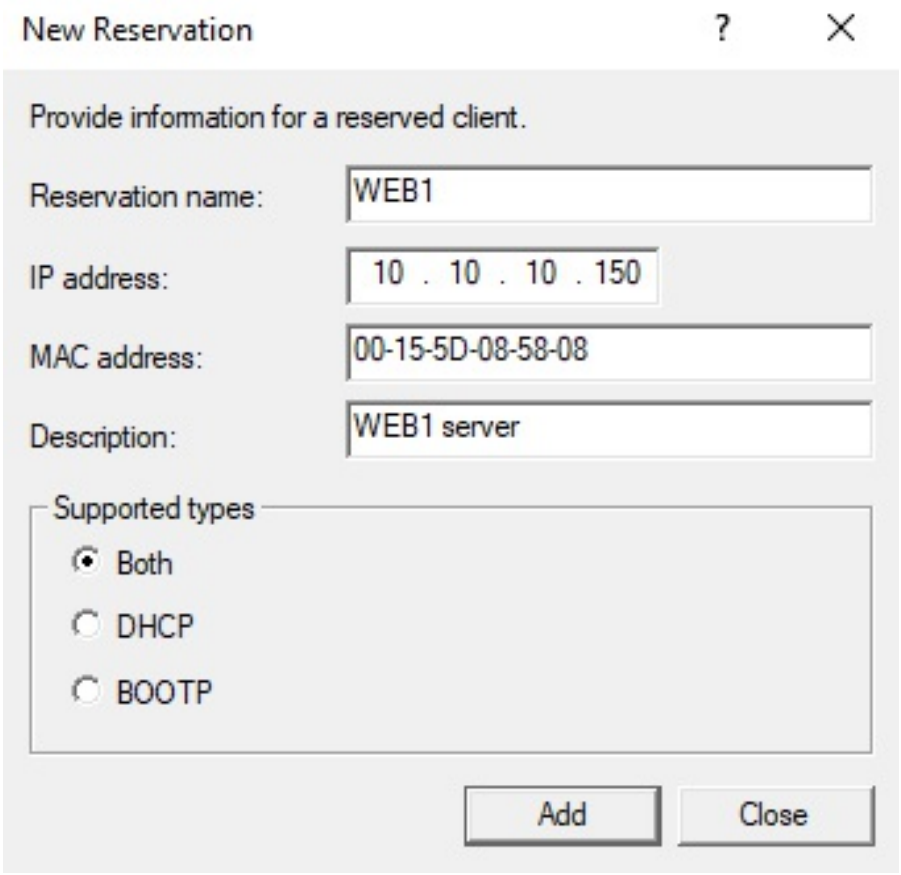
Now that our new scope has been created, you can expand the scope inside the DHCP console and see some additional information about this scope. By clicking on the Address Leases folder, you can see all of the DHCP addresses that have been handed out by this DHCP server. As you can see in the following screenshot, I have a Windows 10 client computer on the network, which does not have a static address, and so it has grabbed a DHCP address from my DHCP server. It has been given the first IP address that I defined in my scope, `10.10.10.100`. The next machine that reaches in to grab an IP address from this DHCP server will receive `10.10.10.101`, and so on from there:


DHCP
File
Action
View
Help
DHCP
dc1.contoso.local
IPv4
Server Options
Scope [10.10.10.0] Contoso
Address Pool
Address Leases
Reservations
Scope Options
Policies
Policies
Filters
IPv6
Client IP Address
Name
10.10.10.100
Win10.contoso.local

DHCP reservations

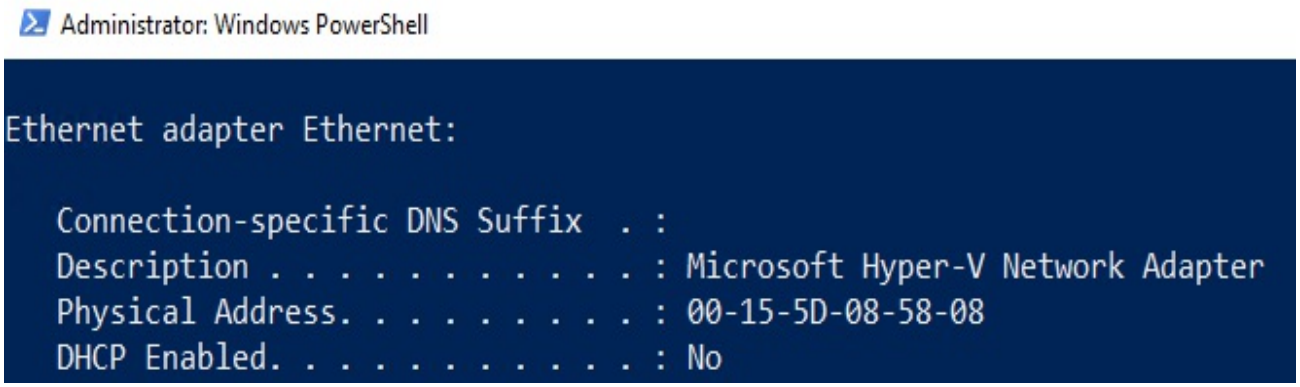
Assigning IP addresses from a big pool of available ones is great, but these address leases are subject to expiry and change. This means that a computer that has `10.10.10.100` today might receive `10.10.10.125` tomorrow. Typically, this is fine from a desktop computer perspective, as they don't generally care what IP address they have. Client computers are usually reaching outward on the network, but other devices are rarely trying to find and contact them. What if you have a more permanent fixture in your network, like a Windows Server, but you don't want to have to deal with statically addressing this server? This is where **DHCP reservations** come into play. A reservation is the act of taking a single IP address within your DHCP scope, and reserving it to a particular device. This device will receive the same IP address every time it connects through the DHCP server, and this particular IP address will not be handed out to any other device on your network. By using reservations inside DHCP, you can allow the DHCP server to handle the assigning of IP addresses even to your permanent servers, so that you do not have to manually configure the NICs of those servers, yet still maintain permanent IP addresses on those machines.

You can see the folder called Reservations in the DHCP console. Currently, there is nothing listed here, but by right-clicking on **Reservations** and choosing New Reservation... we will create one for ourselves. Let's work once again with that `web1` server. Right now, I have a static IP address assigned to `web1`, but I will instead create a reservation for it on the IP address `10.10.10.150`:

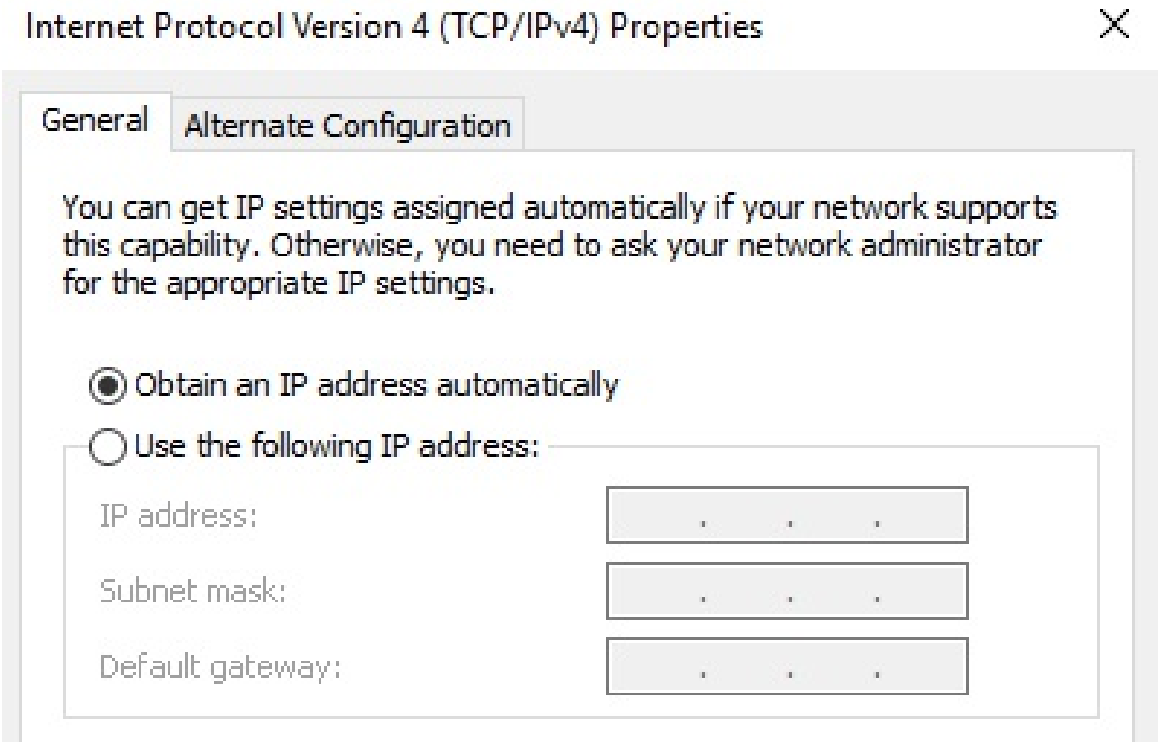


Whoa, whoa, whoa ... back the train up. Most of the information on this screen makes sense—a quick description of the server name and the IP address itself—but how did I come up with that MAC address? A **MAC address** is a network card's physical address on the network. When your networking equipment tries to send information to a certain IP address, or, in this case, when the DHCP server needs to hand a certain IP address to a particular NIC on a server, it needs a physical identifier for that network card. So, this MAC address is something that is unique to the NIC on my `web1` server. By logging into my `web1` server, I can run `ipconfig /all` and see the MAC address listed for my NIC right in that output, that goofy looking combination of letters and numbers shown as Physical Address. That is where I got this information. This is how DHCP decides when to invoke reservations. If a network interface asks it for a DHCP address, and that device's MAC address is listed here in

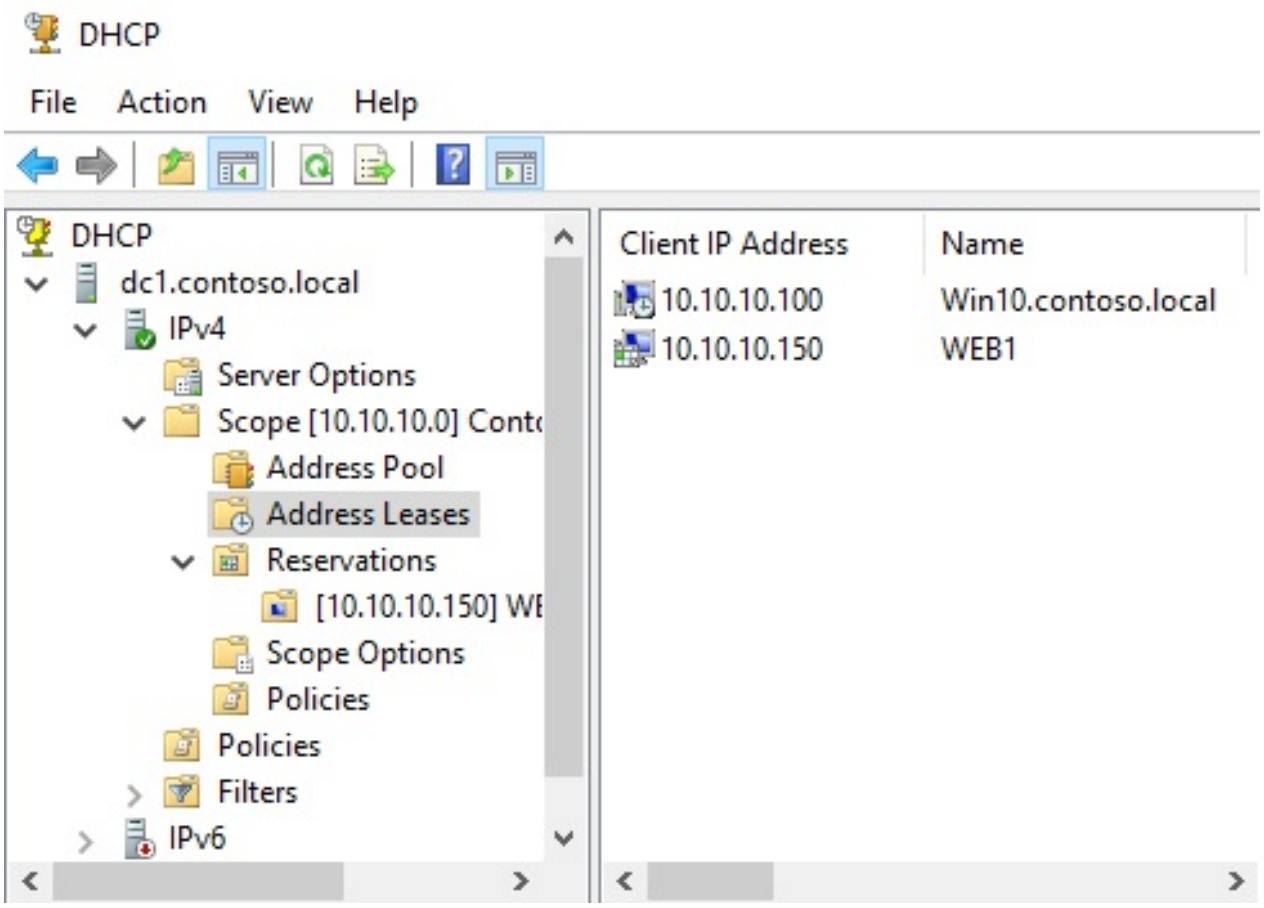
the reservations, then the DHCP server will hand the reserved address back to the device, rather than one from the general pool:



Now that our DHCP reservation has been created, I will head into the NIC settings on my `web1` server, and get rid of all the static IP addressing information by choosing the option to Obtain an IP address automatically:



After doing that, `web1` will reach over to the DHCP server and ask for an address, and you can see that I have now been assigned the reserved address of `10.10.10.150`. This will always be the IP address of the `web1` server from this point forward, unless I change my DHCP Reservation or somehow change the MAC address of `web1`. This could possibly happen if I were to install a new NIC into `web1`:



You can also create DHCP Reservations for objects other than Windows devices in your network. Since all you need is the MAC address of the device (and every device with a network adapter has a MAC address), it is easy to create reservations for devices such as print servers, copy machines, security alarm systems, and more.

Back up and restore

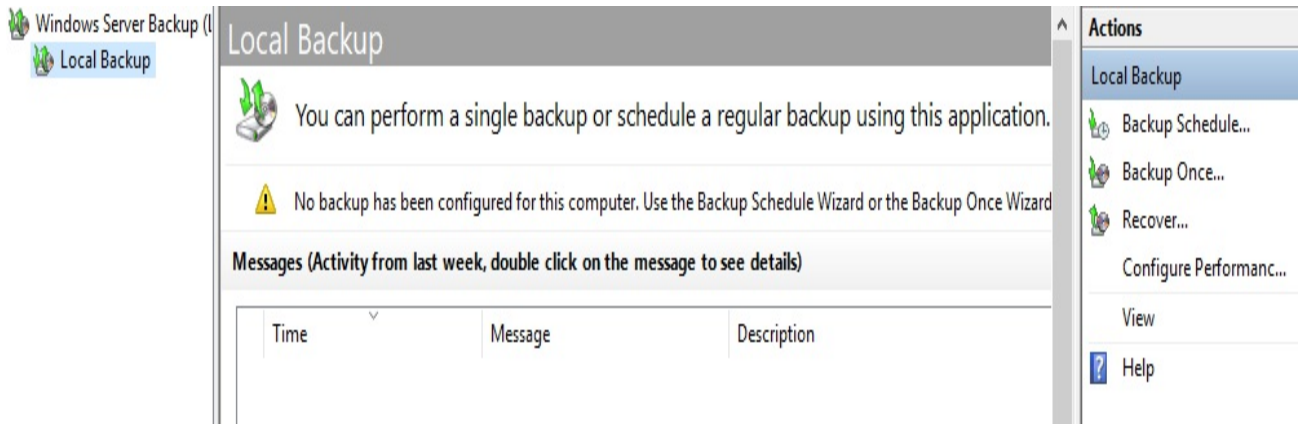
The need to back up and occasionally to restore your servers is, unfortunately, still present in Windows Server 2019. I dream of a day when servers are 100 percent reliable and stable throughout their lifetimes, unaffected by viruses and rogue software, but today is not that day. While there are many third-party tools available on the market that can improve and automate your backup experience when managing many servers, we do have these capabilities baked right into our own Server 2019 operating system, and we should all be familiar with how to utilize them.

Schedule regular backups

Logging into your servers and launching a manual backup task every day is obviously not feasible for most of our organizations, as the process of running backups would turn into our full time job. Thankfully, the Windows Server Backup feature gives us the option to create a backup schedule. This way, we can define what we want to back up, where we want to back it up to, and how often this backup should run. Then we can sit back, relax, and know that our systems are performing this task on their own.

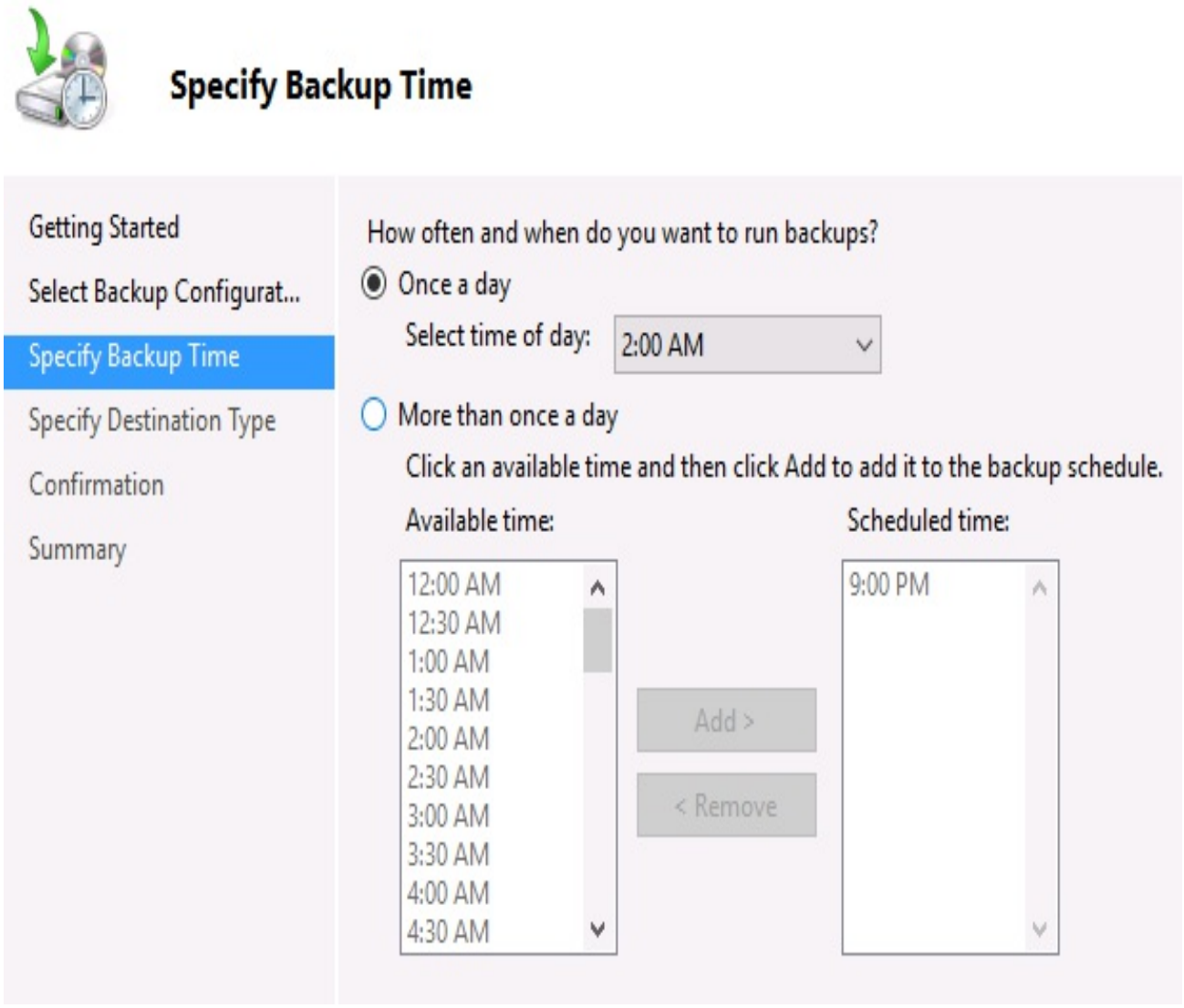
Before we can do anything with backups, we need to install the appropriate feature inside Windows. Using your Add roles and features link, go ahead and install the feature called Windows Server Backup. Remember that I said *feature*—you won't find Windows Server Backup on the primary **Server Roles** selection screen; you need to move ahead one screen further in the wizard to find Features. Once the feature has finished installing, you can launch the Windows Server Backup console that is available inside the Tools menu of Server Manager. Once inside, click on Local Backup in the left-side window pane and you will see some Actions appear on the right-hand side of your screen.

As you can see, there is an option listed here called **Backup Once…** that would, as the name implies, perform an ad hoc backup job. While this is a nice feature, there is no way that any server administrator is going to log into all of their servers and do this every day. Instead, clicking on the Backup Schedule... action will launch a configuration wizard for creating a scheduled, recurring backup job:



The first option you come across is deciding what it is that you want to back up. The default option is set for Full server, which will take a backup of everything in the operating system. If you want to customize the amount of data that is being backed up, you can choose the Custom option and proceed from there. Since I have lots of disk space available to me, I am going to stick with the recommended path of creating full server backups.

Next, we get to the real advantage of using the scheduling concept: choosing how often our backup is going to run. The most common way is to choose a particular time of day, and let the backup run every day at that allotted time. If you have a server whose data is being updated regularly throughout the days and you want to shorten your window of lost information in the event of needing to perform a restore, you can also specify to back up multiple times per day:

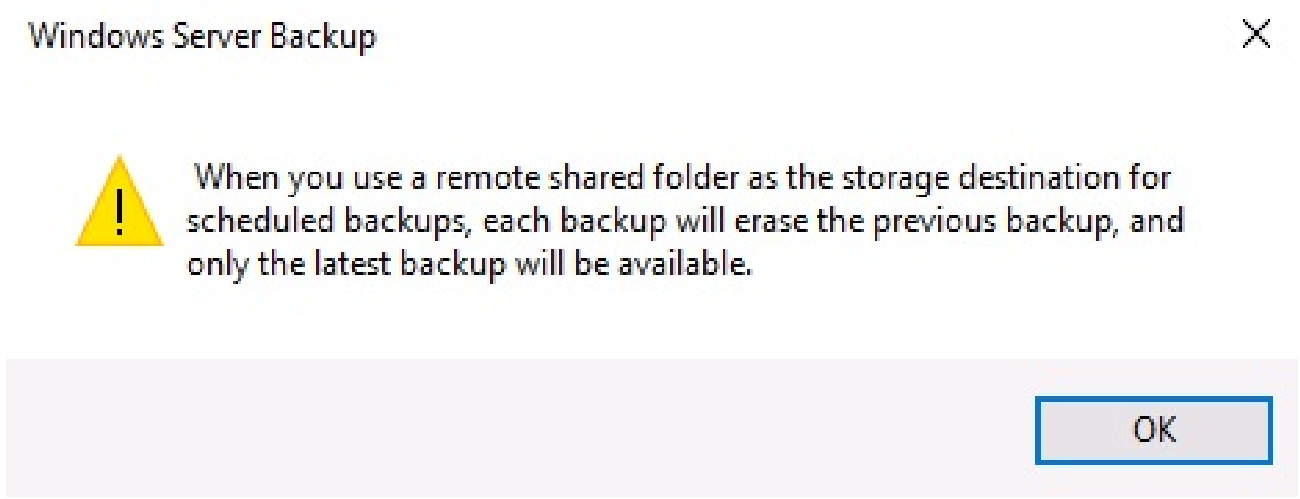


The last screen where we need to make a decision for our scheduled backups is the Specify Destination Type screen, where we determine the location that our backup files are going to be stored. You can see there are a couple of different options for storing the backup locally on the physical hard disks of the same server where you are configuring the backups. Storing backup files on a local, dedicated disk, or volume, can be advantageous because the speed of the backup process will be increased. For servers that you are trying to back up on workdays in order to continually back up data, you would likely want to choose a

local backup option so that those backups run quickly and smoothly. Another advantage to using a locally connected disk for backups is that you can create multiple rollback points within your backup schema, keeping multiple days' worth of backup information in case you need to roll back to a particular point in time.

However, I find that most admins prefer to keep all of their backup files in a centralized location, and that means choosing the third option on this screen, the one entitled Back up to a shared network folder. By choosing this option, we can specify a network location, such as a file server or drive mapping to a NAS, and we can set all of our different servers to back up to this same location. That way we have a central, standardized location where we know that all of our backup files are going to be sitting in the event that we need to pull one out and use it for a restoration.

I cannot tell you which option is best, because it depends on how you are planning to utilize backups in your own environment. The screen where we choose which destination type we want for our backups includes some good text to read over related to these options, such as the important note that when using a shared network folder for backups, only one backup file can be stored at a time for your server, because the process of creating a new backup on the following day will overwrite the previous backup:



Once you have chosen a destination for your backups, and specified a network share location if that is the option you have chosen, you are finished in the wizard. Your backup jobs will automatically kick off at the allocated time that you specified during the wizard, and tomorrow you will see a new backup file existing for your server. If you are impatient, like, me and want to see the backup job run right now, you can walk through the other Action available in the Windows Server Backup console called Backup Once... in order to run a manual backup right away:



Backup Progress

Backup Options

Confirmation

Backup Progress

Status: Backup in progress...

Status details

Backup location: \\WEB1\Backups

Data transferred: 4.05 GB

Items

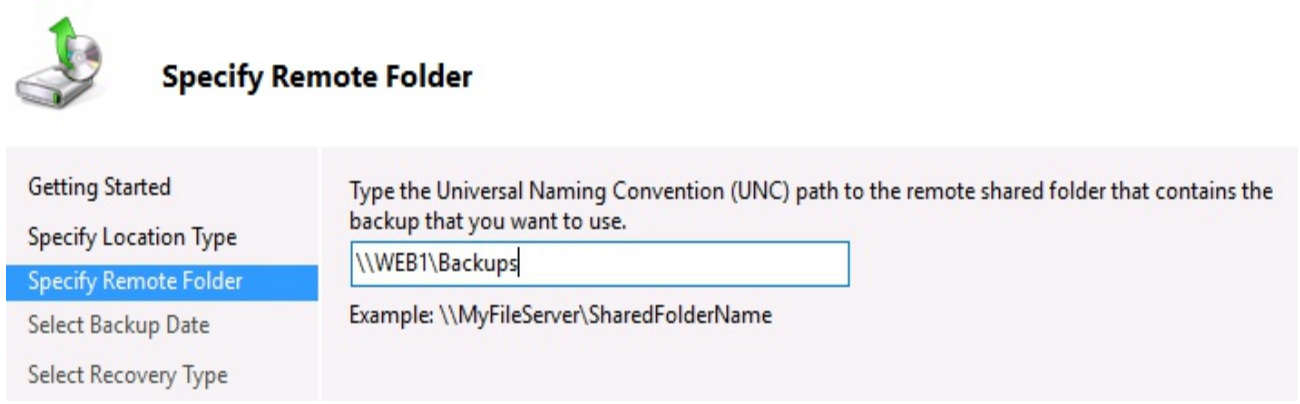
Item	Status	Data transferred
System Reserv...	Completed.	317.00 MB of 317.00...
Local disk (C:)	42% of backup don...	3.74 GB of 8.84 GB
System state	Backup in progress...	-
Bare metal rec...	Backup in progress...	-

You may close this wizard and the backup operation will continue to run in the background.

Restoring from Windows

Since you are being diligent and keeping good backups of your servers, the hope is then that you will never have to actually utilize those backup files in order to restore a server. But, alas, the time will probably come when you have a server that goes sideways, or some data is accidentally deleted, and you must revisit the process of restoring data or an entire server in your infrastructure. If your server is still online and running, the restore process is quite easy to invoke from the same Windows Server Backup console. Open up the console, and choose the Action that says Recover....

This invokes another wizard that walks us through the recovery process. First, we specify the location of our backup file. If you have a dedicated backup location on the local server, it is pretty simple to find; otherwise, like in my example, where we specified a network location, you choose A backup stored on another location, and then choose Remote shared folder in order to tell it where to find that backup file:

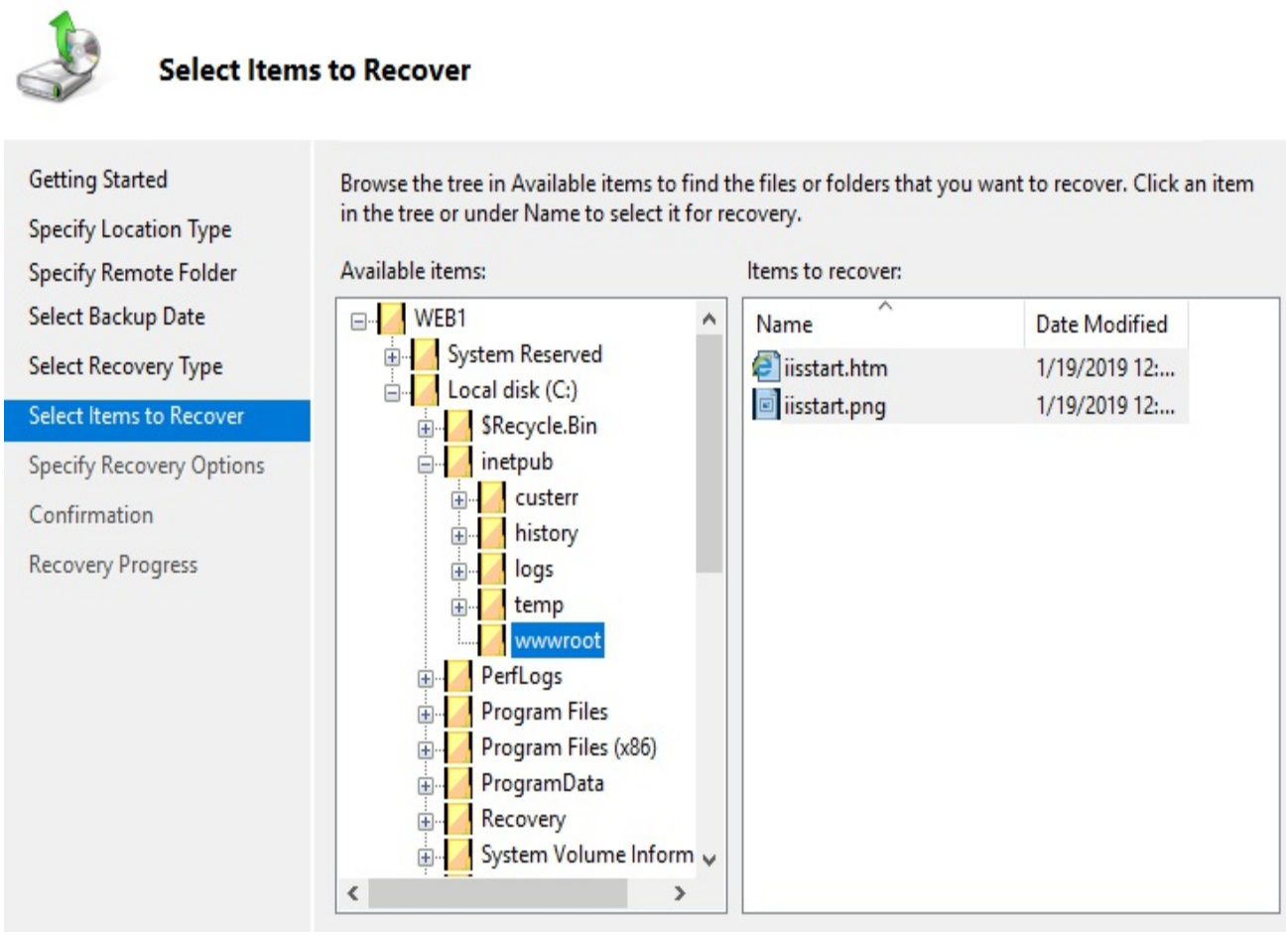


Based on the backup location that you have chosen, the wizard will now

identify all available rollback dates that are available within the backup files. If you have stored your backup files on a local disk so that multiple days' worth of rollback points are available, then you will see numerous dates available to click on. For me, since I chose to store my backups on a network location, that means only one day's worth of backup information is available, and yesterday's date is the only one which I can choose. So I will choose to restore yesterday's backup, and continue on through the wizard.

Now that we have identified the specific backup file that is going to be used for recovery, we get to choose what information from that backup is going to be restored. This is a nice piece of the recovery platform, because, often, when we need to restore from backup, it is only for specific files and folders that may have been deleted or corrupted. If that is the case, you choose the top option, Files and folders. In other cases, you may want to roll the entire server back to a certain date, and for that functionality you would choose to recover an entire Volume. Right now, I am just missing a few files that somehow disappeared between yesterday and today, so I am going to choose the default Files and folders option.

The Select Items to Recover screen is now presented, which polls the backup file and displays to me the entire list of files and folders within the backup file, and I simply choose which ones I want to restore. This kind of recovery can be critical to your daily management of a file server, where the potential is high for users to accidentally delete information:



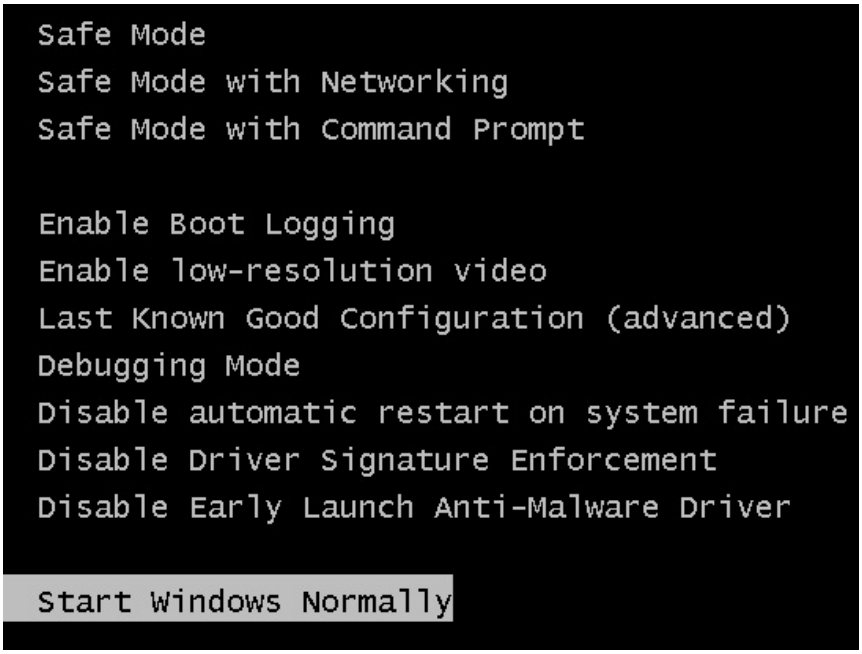
All that remains is to specify where you want these recovered files to be restored. You can choose for the recovered files to be placed back in their original location, or if you are running this recovery process on a different machine, you can choose to restore the files to a new location from which you can grab them and place them manually wherever they now need to reside.

Restoring from the installer disc

Recovery from the console inside Windows is a nice wizard-driven experience, but what about in the case where your server has crashed hard? If you cannot get into Windows on your server, you cannot run the Windows Server Backup console in order to initiate your recovery process. In this case, we can still utilize our backup file that has been created, but we need to use it in combination with a Windows Server 2019 installation disc, from which we can invoke the recovery process.

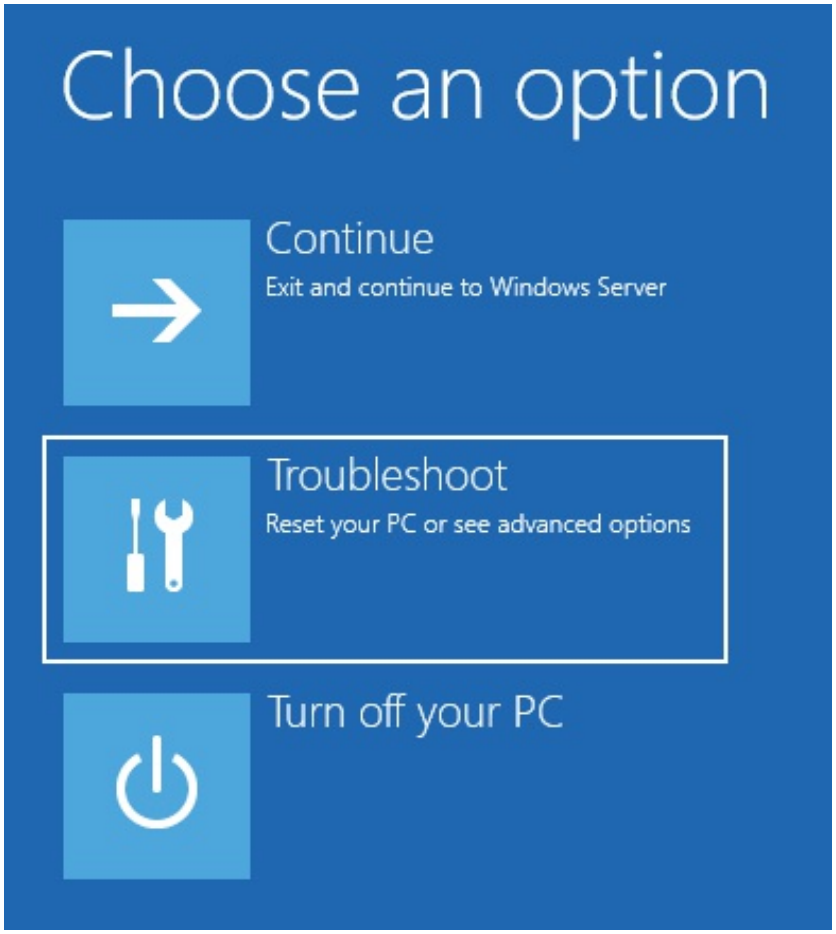
> *It is important to note that this recovery process cannot access locations on the network, and your backup file will have to be stored on a disk attached to your server. You can utilize a USB drive for this purpose during the recovery process, if you did not originally set up your backup job to store onto an existing locally attached disk.*

To make things interesting, I'm going to crash my own server. This is the server that we took a backup of a few minutes ago. I accidentally deleted some very important files in my `C:\Windows` directory. Whoops! Now this is all I see when I try to boot my server:

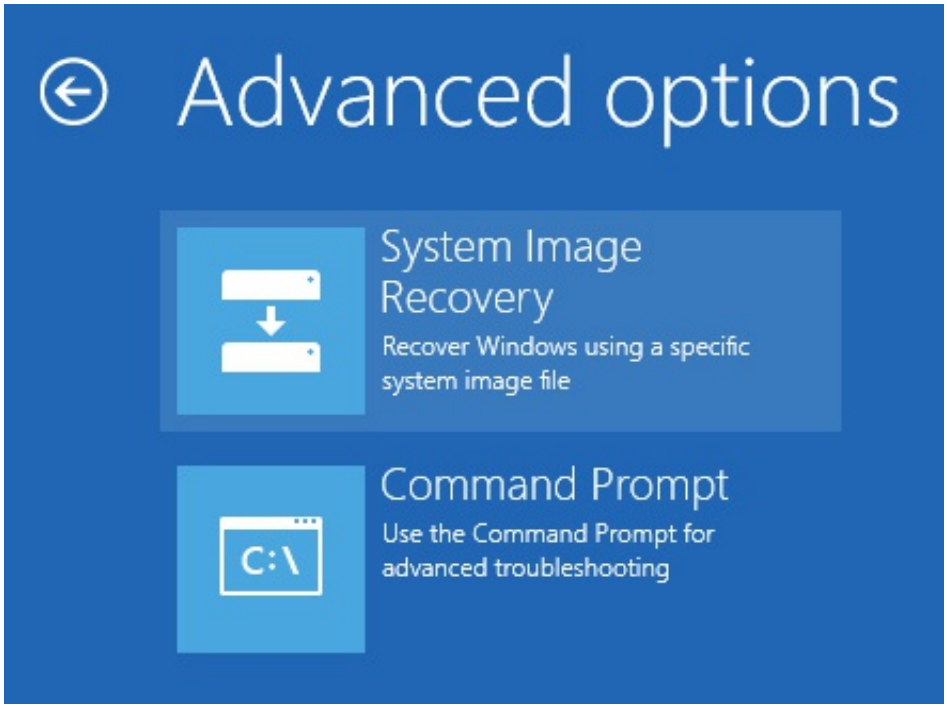


That's not a very friendly screen to see first thing in the morning! Since I seem to be stuck here and unable to boot into Windows, my chances of running the recovery wizard are nil. What to do? Boot to the Windows Server 2019 installer DVD? No, I do not want to install Windows afresh, as all of my programs and data could be overwritten in that scenario. Rather, once I get into the installer screens, you will notice that there is an option down in the corner for Repair your computer. Choose this option in order to open up the recovery options that are available to us on the installation DVD.

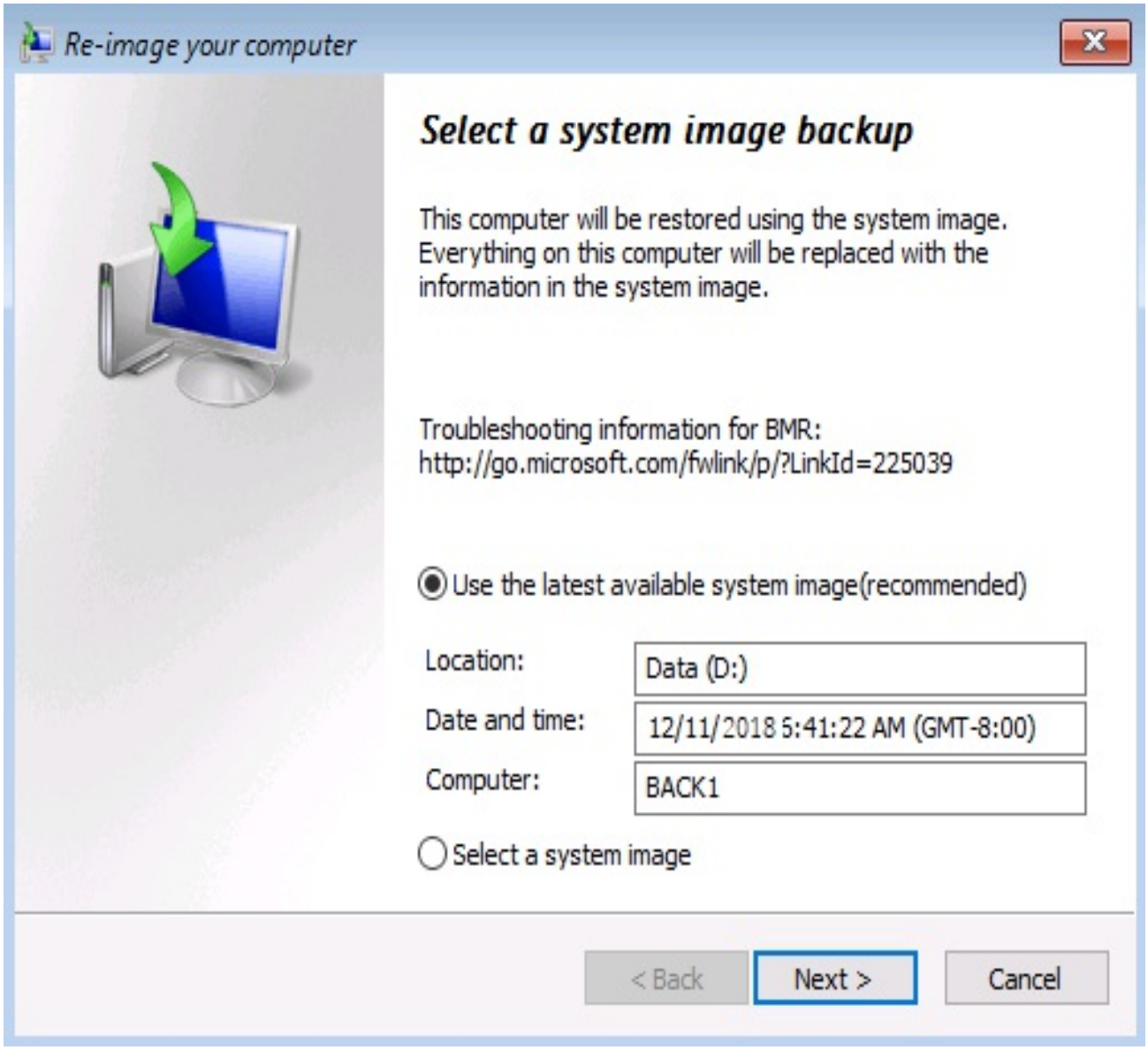
Now you see the screen adjust to a new blue hue, indicating that we have entered a special portion of the installer disc. If we click on the Troubleshoot button, we can see all of the options that we have available:



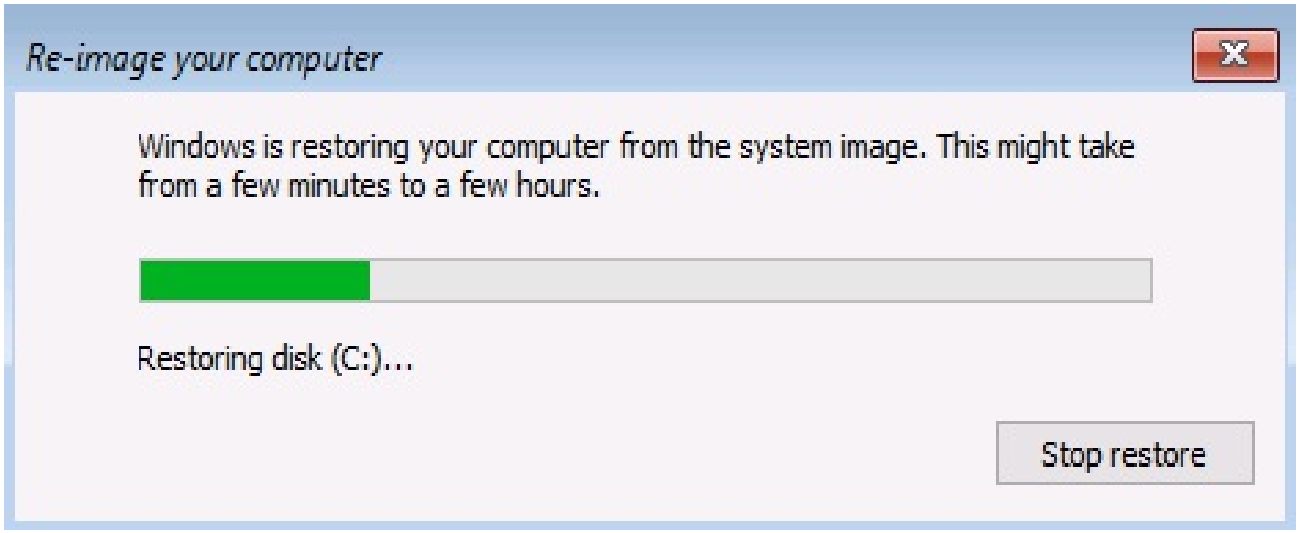
If you think you can fix whatever the issue is from Command Prompt, choose that option and try to fix it yourself. For our example, I am pretty sure that I significantly hosted the operating system, so I am going to do a full System Image Recovery and click on that button:



As long as you have a hard drive connected that contains a Windows Server Backup file, the wizard launches and pulls in the information about the backup. Since I had originally chosen to store my backup file on a network location, I copied the backup files down to a disk and connected it as a second disk into my server. The wizard automatically recognizes that backup file, and displays it in the Select a system image backup screen:



Now, by simply clicking on Next a few times to progress through the wizard, my backup image is restoring onto my server:



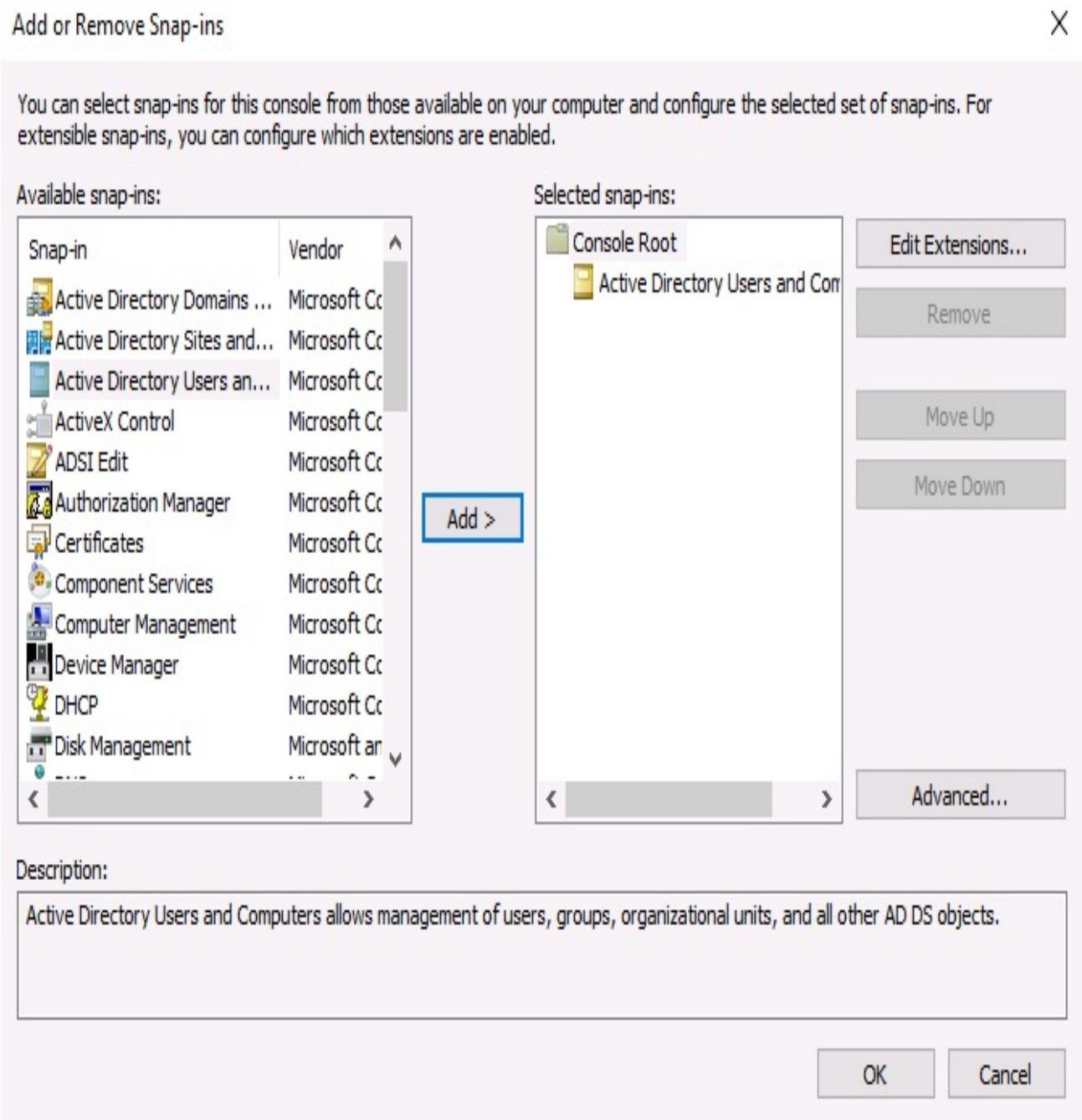
Once the restore process has completed, the system reboots and launches itself right back into Windows, where it is fully functional back to the restore point. My test server doesn't have much of anything running on it, so the time that it took to restore was pretty minimal and a production box may take a little longer, but I'd say that 20 minutes from blowing up the server to being fully recovered is a pretty awesome length of time!

Keeping good and recent backup files is critical to your operation's sustainability. I have worked on quite a few systems where the admins took some manual backups after getting their servers initially configured, but never set up a regular schedule. Even if the data on the server never changes, if you are part of a domain, you never want to do this. In the event that a server fails and you need to recover it, restoring a backup that is only a few days old will generally recover well. But if you restore an image that is 6 months old, Windows itself will come back online with no problems and all of your data will exist, but in that amount of time your computer account for that server would most certainly have fallen out of sync with the domain, causing you authentication errors against the Domain Controllers. In some cases, you may even have to do goofy things like disjoin and re-join the server

to the domain following the image restoration in order to recover communications to the domain. If you had kept regular backups from which to restore, you would not have to deal with those issues.

MMC and MSC shortcuts

You have probably noticed that many of the management consoles that we utilize to configure components inside Windows Server 2019 look pretty similar. What happens under the hood with a number of these consoles is that you are actually looking at a snap-in function, a specific set of tools that are snapped into a generic console tool called the **Microsoft Management Console**, more commonly referred to as **MMC**. In fact, rather than open all of these management functions from inside Server Manager, for many of them, you could simply type `MMC` by navigating to Start | Run or in Command Prompt, and invoke the generic MMC console. From here, you can click on the File menu and choose Add or Remove Snap-ins:

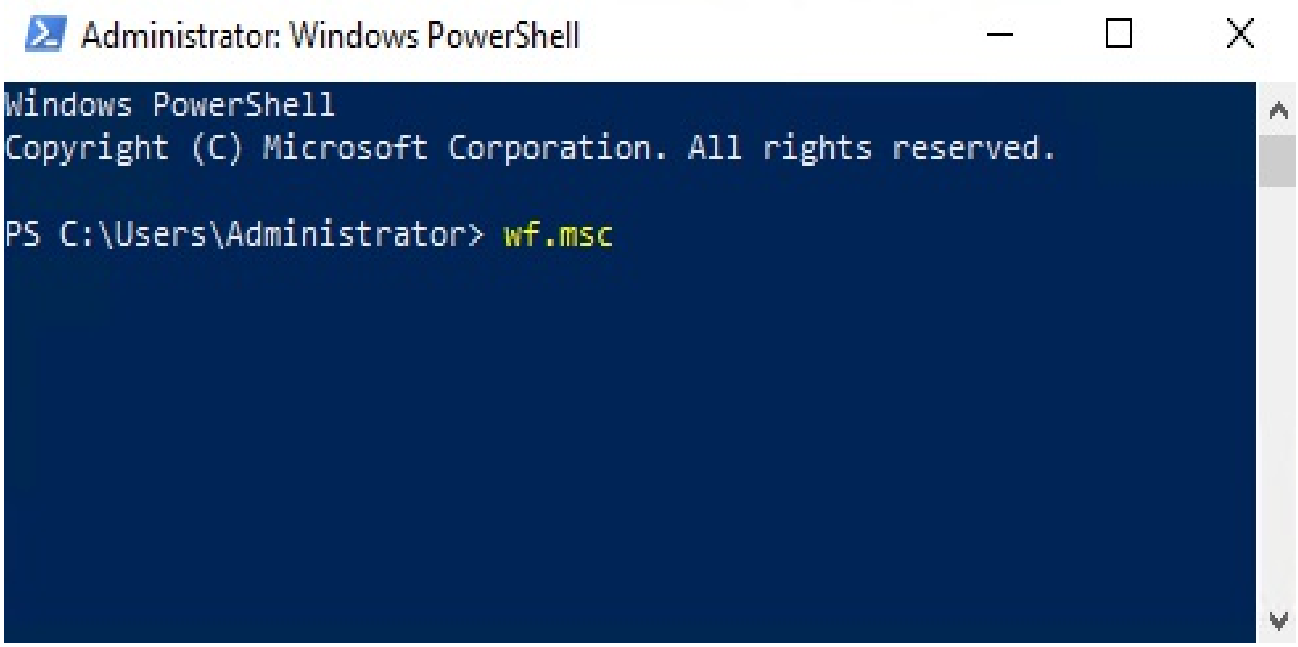


Choose the management snap-in that you would like to work in, and add it to the console. There are a great number of management functions that can be accessed through the standard MMC console, and even some particular functions where MMC is the preferred, or perhaps

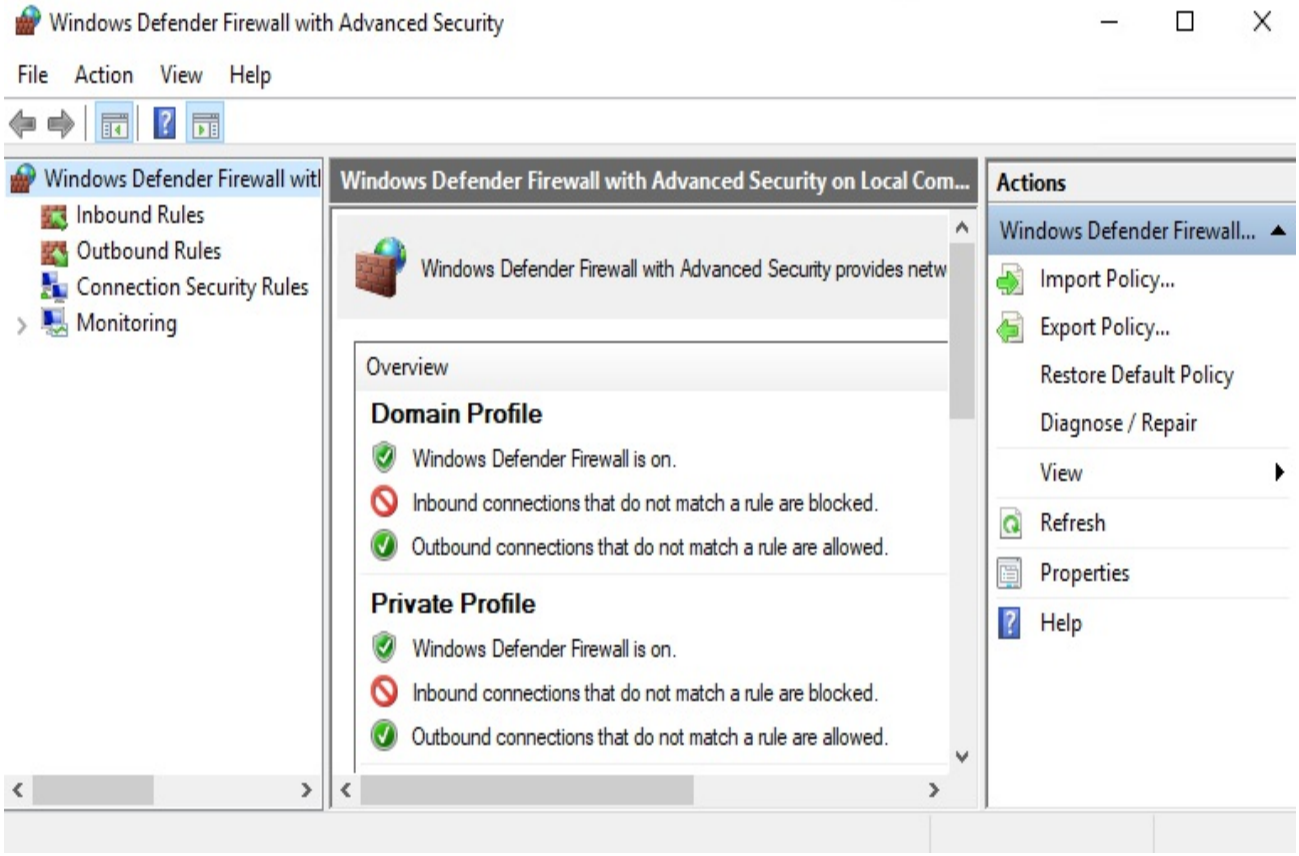
the only, method to interact with some components of Windows. For example, later in our book we will look into certificate stores within Windows Server 2019, and we will be utilizing MMC for some of that interaction.

Another interesting way to open up many of the management consoles is by using their direct **MSC** tool name. An MSC file is simply a saved configuration of an MMC console session. There are many MSC shortcuts stored in Windows Server 2019 out of the box. If a given management console provides the capability of being launched by an MSC, all you need to do is type in the name of the MSC by navigating to either Start | Run or in Command Prompt or a PowerShell window, and it will immediately launch into that particular management console without needing to snap anything in, and without needing to open Server Manager whatsoever. Since I tend to prefer using a keyboard over a mouse, I always have a PowerShell window or Command Prompt open on each system I'm working with, and I can very quickly use that window to open up any of my MSC administrative consoles. Let's show one example, so that you know exactly how to use this functionality, and then I will provide a list of the common MSCs which I find useful on a day-to-day basis.

Open an elevated PowerShell window, type `WF.MSC`, and press *Enter*:



The Windows Defender Firewall with Advanced Security window will open, and is ready to accept input from you. We didn't have to poke through Control Panel, or open the regular Windows Firewall and then click on the Advanced Settings link, which are the common ways to get into this console by using a mouse. By knowing our MSC shortcut name, we were able to take a direct route to opening the full WFAS console, which is where I often go to check over particular firewall rules or status:



Now that you've seen how an MSC command works, and again there are many different places where you can type in the name of an MSC and invoke it, I want to leave you with a list of common MSC consoles that you can use to quickly gain access to many administrative consoles on your servers:

- `DSA.MSC`: Active Directory Users and Computers
- `DSSITE.MSC`: Active Directory Sites and Services
- `DNSMGMT.MSC`: DNS Manager
- `GPEDIT.MSC`: Local Group Policy Editor
- `GPMC.MSC`: Group Policy Management Console

- `CERTSRV.MSC`: Certification Authority Management
- `CERTTMPL.MSC`: Certificate Template Management
- `CERTLM.MSC`: Local Computer Certificates Store
- `CERTMGR.MSC`: Current User Certificates Store
- `COMPMGMT.MSC`: Computer Management
- `DEVMGMT.MSC`: Device Manager
- `DHCPMGMT.MSC`: DHCP Manager
- `DISKMGMT.MSC`: Disk Management
- `EVENTVWR.MSC`: Event Viewer
- `PERFMON.MSC`: Performance Monitor
- `SECPOL.MSC`: Local Security Policy Console
- `FSMGMT.MSC`: Shared Folders
- `WF.MSC`: Windows Defender Firewall with Advanced Security

Summary

In this chapter, we discussed some of the roles and components inside Windows Server 2019 that you will need to utilize and be familiar with, if you want to run a truly Microsoft-centric infrastructure. Active Directory, DNS, and DHCP are the primary core services that underlie and support your entire infrastructure. A basic understanding of these technologies is essential to any Windows Server administrator, and an in-depth knowledge of the way these tools work together will open many doors to you as you progress through your IT career. Almost all of the other roles and features available in Windows Server are supported by or hinge on these core services. I hope that you are comfortable with navigating around inside them after following through the examples in this chapter.

Now we continue on our journey through Windows Server 2019 with the next chapter as we dive into one of the scarier topics—for many admins, anyway—certificates! In the next chapter, we will be looking at the certificates in Windows Server 2019.

Questions

1. Inside Active Directory, a container (folder) that holds computer and user accounts is called a...?
2. What is the term for creating a computer account inside Active Directory prior to that computer being joined to your domain?
3. Which management tool is used to specify that certain physical locations in your network are bound to particular IP subnets?

4. What is the name of a special Domain Controller that cannot accept new information, only synchronize from an existing Domain Controller?
5. Which GPO contains password complexity settings in a brand new installation of Group Policy?
6. What kind of DNS record resolves a name to an IPv6 address?
7. What is the MSC shortcut for opening Active Directory Users and Computers?

Certificates in Windows Server 2019

"Ugh, we need to use certificates to make this work."
- Quote from an anonymous admin who just discovered their latest technology purchase requires the use of certificates in their organization

If this sounds familiar, don't scrap that new project just yet! For some reason, the use of certificates seems like a daunting task to many of us, even those who have worked in IT for many years. I think this is probably because there are many different options available on a certificate server, but there is not a lot of common sense or user-friendliness built into the management console for dealing with certificates. This, combined with a general lack of requirements for certificates on servers for so many years, means that, even though this technology has existed for a long time, many server administrators have not had the opportunity to dig in and deploy certificates for themselves. I regularly deploy a couple of technologies that require a broad use of certificates in an organization, often needing to issue them to every workstation or user in the network, and I hear these kinds of concerns all the time. Issuing a certificate to a single business-critical web server sounds daunting enough if you don't have any experience with the process, let alone issuing hundreds or thousands of certificates all at once. Another common scenario is one where a company determined certificates to be in their best interests but lacked the on-staff resources to stand it up themselves, and so hired a third party to implement certificates within the network. While this gets certificates rolling, it often leaves a knowledge gap that never gets filled, so you may have a certificate server up and running, but not be at all comfortable

modifying or utilizing it.

The broad term for a certificate environment is known as **Public Key Infrastructure** (**PKI**). I call that out specifically because you will probably see PKI listed in documentation or requirements at some point, if you haven't already. Your PKI is provided by servers in your network, and configuring those servers to issue certificates for you is the purpose of this chapter. The servers that you determine to be your certificate servers are known as **certification authority** (**CA**) servers, and we will refer to them as CA servers throughout this book.

In order to get you rolling with certificates in your own network, here are the topics that we will cover in this chapter:

- Common certificate types
- Planning your PKI
- Creating a new certificate template
- Issuing your new certificates
- Creating an auto-enrollment policy
- Obtaining a public-authority SSL certificate
- Exporting and importing certificates

Common certificate types

There are a number of different types of certificates that you may find yourself needing to publish. As you will see, when you need a certificate that has a list of particular requirements, you can build a certificate template to whatever specifications you like. So, in a sense, there aren't really certificate *types* at all, but just certificate templates that you scope to contain whatever pieces of information are needed for that cert to do its job. While this holds true technically, it is generally easier to segment certificates into different groups, making them more distinguishable for the particular job that they are intended to perform.

User certificates

As the name implies, a user certificate is one used for purposes that are specific to the username itself. One of the platforms that is driving more certificate adoption is the network-authentication process. Companies that are looking into stronger authentication in their environments often look at certificates as part of that authentication process. Smart cards are one of the specific mechanisms that can be used for this purpose, specifically, some sort of physical card to be plugged into a computer in order for the user to gain access to that computer.

Smart cards can also be stored virtually, within a special place on newer machines called the TPM. But that is a discussion for a different day. The reason we mention smart cards here is because, often the core functionality of the smart-card authentication is provided by a user certificate that has been stored on that smart card. If you find yourself in the middle of a project to deploy smart cards, you will probably find yourself in need of a PKI.

Another popular strong authentication form is **one-time passwords** (**OTP**). This requires the user to enter a randomly-generated PIN in addition to their regular login criteria, and in some cases when the user enters their PIN, they are issued a temporary user certificate to be used as part of the authentication chain. Additional places that user certificates are commonly found include when companies employ file-encrypting technologies, such as EFS (short for Encrypting File System), or when building up **virtual private network** (**VPN**) systems to enable remote users to connect their laptops back to the corporate network. Many companies don't want to rely purely on a username and

a password for VPN authentication, so issuing user certificates and requiring that they be present in order to build that VPN tunnel is commonplace.

Computer certificates

Often referred to as computer certificates or machine certificates, these guys get issued to computers in order to assist with the interaction between the network and the computer account itself. Technologies, such as SCCM, that interact with and manage the computer systems regardless of which users are logged into those computers make use of computer certificates. These kinds of certificates are also used for encryption processing between systems on the network, for example, if you were interested in using IPsec to encrypt communications between clients and a highly secure file server. Issuing computer or machine certificates to the endpoints within this communication chain would be essential to making that work properly. I often find myself issuing computer certificates to a business' machines in order to authenticate DirectAccess tunnels, another form of automated remote access. There are many different reasons and technologies you may be interested in, which would require the issuance of certificates to the client workstations in your environment.

SSL certificates

If you find yourself in the middle of the certificate road, where you haven't really managed a CA server but you have at one point issued and installed some kind of certificate, the chances are that the certificate you worked with was an SSL certificate. This is by far the most common type of certificate used in today's technology infrastructure, and your company is more than likely using SSL certificates, even if you are not aware of them and do not have a single CA server running inside your network.

SSL certificates are most commonly used to secure website traffic. Any time you visit a website and see HTTPS in the address bar, your browser is using an SSL packet stream to send information back and forth between your computer and the web server that you are talking to. The web server has an SSL certificate on it, and your browser has checked over that certificate before allowing you onto the web page, to make sure that the certificate is valid and that the website really is who it says it is. You see, if we did not use SSL certificates on websites, anyone could impersonate our site and gain access to the information being passed to the website.

Let's provide a quick example. Let's say one of your users is at a coffee shop, using the public Wi-Fi. An attacker has figured out a way to manipulate DNS on that Wi-Fi network, and so when your user tries to visit `mail.contoso.com` in order to access their company's Outlook Web Access to check email, the attacker has hijacked that traffic and the user is now sitting on a website that looks like their company portal, but is actually a website hosted by the attacker. The user types in their

username and password, and *bingo* the attacker now has that user's credentials and can use them to access your real network. What prevents this from happening every day in the real world? **SSL certificates**. When you force your externally-facing websites, such as that email login page, to be HTTPS sites, it requires the client browsers to check over the SSL certificate that is presented with the website. That SSL certificate contains information that only you as a company have, it cannot be impersonated. This way, when your user accesses your real login page, the browser checks out the SSL certificate, finds it to be correct, and simply continues on its merry way. The user never even knows they are being protected except for the little lock symbol up near their browser's address bar. On the other hand, if their traffic is being intercepted and redirected to a fake website, the SSL certificate check will fail (because the attacker would not have a valid SSL certificate for your company website name), and the user will be stopped in their tracks, at least to read through a certificate warning page before being able to proceed. At this point, the user should back off and realize that something is wrong, and contact IT staff to look into the issue.

SSL certificates used by websites on the internet are almost always provided, not by your internal CA server, but by a public certification authority. You have probably heard of many of them, such as Verisign, Entrust, DigiCert, and GoDaddy. Companies generally purchase SSL certificates from these public authorities because those authorities are trusted by default on new computers that users might purchase in the field. When you buy a new computer, even straight from a retail store, if you were to open up the local store of certificates that exists out of the box on that system, you would find a list of trusted root authorities. When you visit a website protected by an SSL certificate issued from one of these public authorities, that certificate, and therefore the website, is automatically trusted by this computer. The public CAs are publicly-recognized entities, known for their capacity to securely issue SSL certificates.

When a company acquires an SSL certificate from one of these public authorities, there is an in-depth verification process that the authority takes in order to make sure that the person requesting the certificate (you) is really someone with the proper company, and authorized to issue these certificates. This is the basis of security in using SSL certificates from a public CA. All new computers know by default to trust certificates that have been issued by these authorities, and you don't have to take any special actions to make your websites function on the internet. On the other hand, it is possible to issue SSL certificates from a CA server that you built yourself and have running inside your network, but it requires a couple of things that make it difficult, because your CA server is obviously not trusted by all computers everywhere, nor should it be. First, if you want to issue your own SSL certificate for use on a public website, you need to externalize at least part of your internal PKI, known as the **Certificate Revocation List** (**CRL**), to the internet. Any time you take a component that is internal to your network, and publicize it on the internet, you are introducing a security risk, so unless you absolutely have to do this, it is generally not recommended. The second reason it is difficult to utilize your own SSL certificates on public websites is that only your own company's domain-joined computers will know how to trust this SSL certificate. So, if a user brings their company laptop home and uses it to access their email login page, it will probably work fine. But if a user tries to access the same email login page from their home computer, which is not part of your domain or network, they would get a certificate warning message and have to take special steps in order to gain access to the website. What a pain for the users. You should never encourage users to accept risk and proceed through a certificate warning message—this is a recipe for disaster, even if the certificate they are clicking through is one issued by your own CA. It's a matter of principle never to accept that risk.

These issues can be alleviated by purchasing an SSL certificate from one

of those public cert authorities, and so purchasing these kinds of certificates is the normal and recommended way to make use of SSL on your publicly-facing websites. Websites that are completely inside the network are a different story, since they are not facing the internet and their security footprint is much smaller. You can use your internal CA server to issue SSL certificates to your internal websites, and not have to incur the cost associated with purchasing certificates for all of those websites.

There are a few different tiers of SSL certificates that you can purchase from a public CA, information for which is listed on the authority's own websites. Essentially, the idea is that the more you pay, the more secure your certificate is. These tiers are related to the way that the authority validates against the certificate requester, since that is really where the security comes into play with SSL certificates. The authority is guaranteeing that when you access the page secured by their certificate, the cert was issued to the real company that owns that web page.

Other than the validation tier, which you get to choose when purchasing a certificate, there is another option you have to decide on as well, and this one is much more important to the technical aspect of the way that certificates work. There are different naming conventions available to you when you purchase a certificate, and there is no best answer for which one to choose. Every situation that requires a certificate will be unique, and will have to be evaluated individually to decide which naming scheme works best. Let's quickly cover three possibilities for an SSL-certificate naming convention.

Single-name certificates

This is the cheapest and most common route to take when purchasing a certificate for an individual website. A single-name certificate protects and contains information about a single DNS name. When you are setting up a new website at `portal.contoso.com` and you want this website to protect some traffic by using HTTPS, you would install an SSL certificate onto the website. When you issue the request to your certification authority for this new certificate, you would input the specific name of `portal.contoso.com` into the Common name field of the request form. This single DNS name is the only name that can be protected and validated by this certificate.

Subject Alternative Name certificates

Subject Alternative Name (**SAN**) certificates generally cost a little bit more than single-name certs, because they have more capabilities. When you request a SAN certificate, you have the option of defining multiple DNS names that the certificate can protect. Once issued, the SAN certificate will contain a primary DNS name, which is typically the main name of the website, and, further inside the cert properties, you will find listed the additional DNS names that you specified during your request. This single certificate can be installed on a web server and used to validate traffic for any of the DNS names that are contained in the certificate. A use-case example of a SAN certificate is when setting up a Lync (Skype for Business) server. Lync uses many different DNS names, but all names that are within the same DNS domain. This is an important note regarding SAN certificates: your names must be part of the same domain or subdomain. Here is an example list of the names we might include in a single SAN certificate for the purposes of Lync:

- `Lync.contoso.com` (the primary one)
- `Lyncdiscover.contoso.com`
- `Meet.contoso.com`
- `Dialin.contoso.com`
- `Admin.contoso.com`

These different websites/services used by Lync are then implemented across one or multiple servers, and you can utilize the same SAN certificate on all of those servers in order to validate traffic that is headed toward any of those DNS names.

Wildcard certificates

Last but certainly not least is the wildcard certificate. This is the luxury model, the one that has the most capabilities, gives you the most flexibility, and at the same time offers the easiest path to implementation on many servers. The name on a wildcard certificate begins with a **star** (*). This star means *any*, as in *anything preceding the DNS domain name*, is covered by this certificate. If you own `contoso.com` and plan to stand up many public DNS records that will flow to many different websites and web servers, you could purchase a single wildcard certificate with the name `*.contoso.com`, and it may cover all of your certificate needs.

Typically, wildcards can be installed on as many web servers as you need, with no limit on the number of different DNS names that it can validate. I have run across an exception to this once, when a particular customer's agreement with their certification authority specified that they had to report and pay for each instance of their wildcard certificate that was in use. So watch those agreements when you make them with your CA. Most of the time, a wildcard is meant to be a free-for-all within the company so that you can deploy many sites and services across many servers, and utilize your wildcard certificate everywhere.

The downside of a wildcard certificate is that it costs more, significantly more. But if you have large certificate needs or big plans for growth, it will make your certificate administration much easier, faster, and cost-effective in the long run.

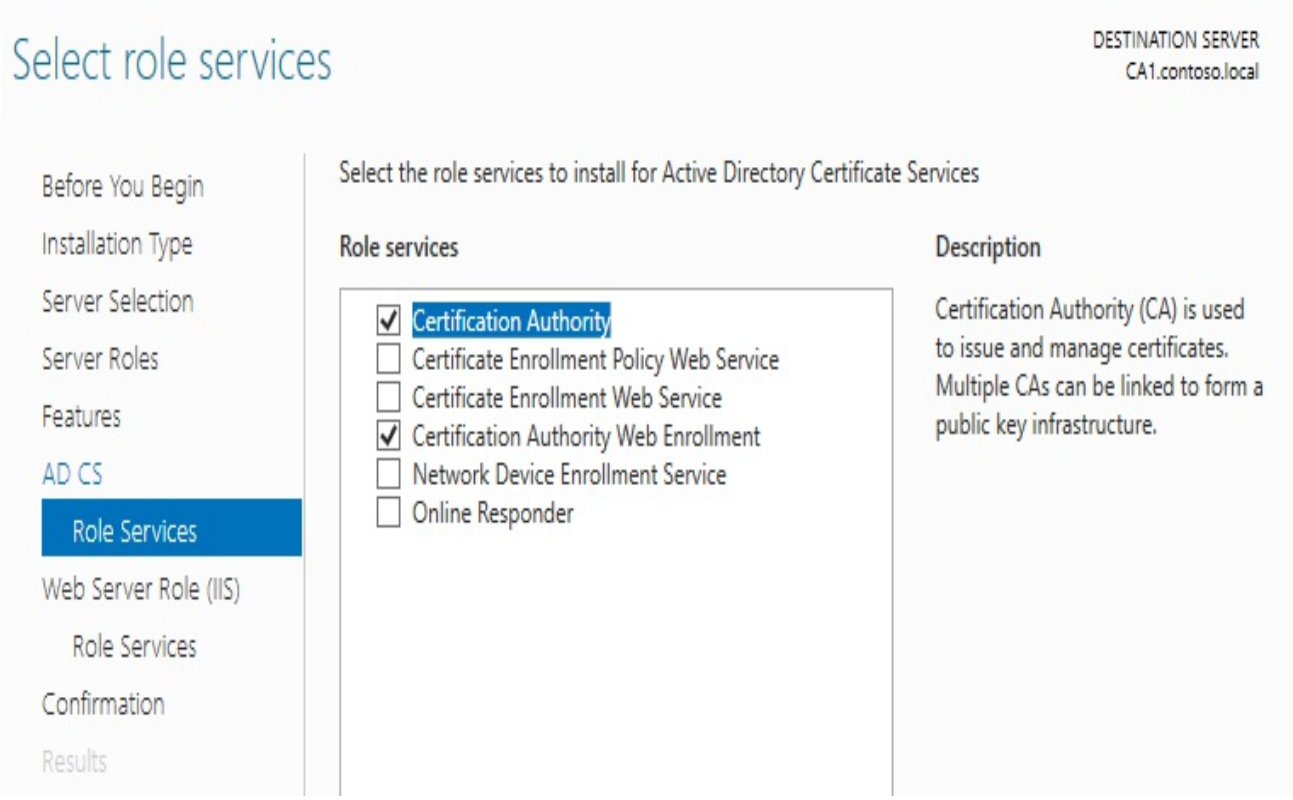
Planning your PKI

Since we are revolving all of our discussion in this book around Windows Server 2019, this means that your CA server can and should be one provided by this latest and greatest of operating systems. As with most capabilities in Server 2019, the creation of a certification authority server in your network is as simple as installing a Windows role. When you go to add the role to a new server, it is the very first role in the **Active Directory Certificate Services** (**AD CS**) list. When installing this role, you will be presented with a couple of important options and you must understand the meaning behind them before you create a solid PKI environment.

> *Your server's hostname and domain status cannot be changed after implementing the CA role. Make sure you have set your final hostname, and joined this server to the domain (if applicable), prior to installing the AD CS role. You won't be able to change those settings later!*

Role services

The first decision you need to make when installing the AD CS role is which role services you would like to install, as you can see in the following screenshot:



Clicking on each option will give you a description of its capabilities, so you can probably determine which pieces of the role that you need by poking around on this screen. Here also is a short summary of these options. Note that I am listing them out of order, because of the way that I typically see them configured in the field:

- **Certification Authority**: This is the primary certificate engine that needs to be installed in order for this server to officially become a CA.
- **Certification Authority Web Enrollment**: Often, this one gets installed as well, especially in environments that are small enough to be running a single CA server for the entire environment. The web-enrollment portion will install IIS (web server) capabilities on this server, and launch a small website that is used for the purposes of requesting certificates. We will discuss this further when we walk through issuing certificates from this web interface, later in the chapter.
- **Certificate Enrollment Web Service and Certificate Enrollment Policy Web Service**: Most of the time, we are only concerned with issuing certificates to our company-owned, domain-joined systems. In those cases, these two selections are not necessary. If you plan to issue certificates to non-domain-joined computers from this CA server, you want to select these options.

- **Network Device Enrollment Service**: As the name implies, this piece of the CA role provides the capability to issue certificates to routers and other kinds of networking devices.
- **Online Responder**: This is a special function reserved for larger environments. Inside every certificate is a specification for a **Certificate Revocation List** (**CRL**). When a client computer

utilizes a certificate, it reaches out and checks against this CRL in order to make sure that its certificate has not been revoked. The CRL is an important piece of the certificate security puzzle; in an environment with thousands of clients, your CRL may be very, very busy responding to all of these requests. You can deploy additional CA servers that are running Online Responder to help ease that load.

For the purposes of our lab, and to cover the required capabilities of most small-to-medium businesses out there, I am going to select the two options shown in the earlier screenshot: Certification Authority and Certification Authority Web Enrollment.

Enterprise versus Standalone

Following the installation of your AD CS role, Server Manager will notify you that certificate services needs some additional configuration, as is common with many role installations. When configuring your CA role for the first time, you will be presented with a big choice. Do you want this CA server to be an **Enterprise CA** or a **Standalone CA**?

Let's start with the enterprise CA. As the wizard will tell you, an enterprise CA server must be a member of your domain, and these certificate servers typically stay online so that they can issue certificates to computers and users who need them. Wait a minute! Why in the world would we want to turn a certificate server off anyway? We will discuss that in a minute, but if you intend to utilize this CA to issue certificates, it must obviously remain turned on. Most CA servers within a domain environment will be enterprise CAs. When creating an enterprise CA, your templates and some certificate-specific information is able to store itself within Active Directory, which makes integration between certificates and the domain tighter and more beneficial. If this is your first interaction with the CA role, I recommend you start with an enterprise CA because this better meets the needs of most organizations.

As you can correctly infer from the preceding text, this means that a standalone CA is less common to see in the wild. Standalones can be members of the domain, or they can remain out of that part of the network and reside on a local workgroup. If you had a security requirement that dictated that your certificate server could not be domain-joined, that might be a reason you would use a standalone CA.

Another reason might be because Active Directory simply does not exist in the chosen environment. In my eyes, it would be extremely rare to find a network where someone was trying to use Windows Server 2019 as their certification authority and at the same time was not running Active Directory Domain Services, but I'm sure there is a corner case somewhere that is doing exactly this. In that case, you would also need to choose standalone. A third example when you would choose standalone is the event we alluded to already, where you might have a reason to turn off your server. When you run this scenario, it is typically referred to as having an **offline root**. We haven't talked about root CAs yet, but we will in a minute. When you run an offline root, you create the top level of your PKI hierarchy as a standalone root CA, and then you build subordinate CAs underneath it. Your subordinate CAs are the ones doing the grunt work issuing certificates—which means that the root can be safely shut down since it doesn't have any ongoing duties. Why would you want to do this? Well, most companies don't, but I have worked with some that have very high-level security policies, and this is why you might visit this topic. If all of a company's CA servers are tied together as enterprise CAs with all of their information being stored inside Active Directory, a compromise to one of the subordinate issuing CAs could spell disaster for your entire PKI. It is possible that the only way to remediate an attack, would be to wipe out the whole PKI environment, all of the CA servers, and build them up again. If you had to do this, it would mean not only rebuilding your servers, but also re-issuing brand new copies of all your certificates to every user and device that has them.

On the other hand, if you were running a standalone root CA that was offline, it would not have been affected by the attack. In this case, you could tear down your affected certificate servers, but your core root server would have been safely hidden. You could then bring this root back online, rebuild new subordinates from it, and have an easier path to being 100% operational because your root keys that are stored within

the CA would not have to be re-issued, as they never would have been compromised in the attack.

Like I said, I do not see this very often in the field, but it is a possibility. If you are interested in learning more about offline root CAs and their uses, I highly recommend checking out the TechNet article at http://social.technet.microsoft.com/wiki/contents/articles/2900.offline-root-certification-authority-ca.aspx. If you're thinking about moving forward with an offline root CA only because it seems like it's more secure, but you don't have a specific reason for doing so, I recommend you change gears and go ahead with an online enterprise root CA. While there are some security advantages to the offline root, most companies do not find those advantages to be worth the extra hassle that accompanies using an offline root CA. There are definitely usability trade-offs when going that direction.

In most cases, you'll want to select Enterprise CA and proceed from there.

Root versus Subordinate (issuing)

This is the second big choice you need to make when building a new CA. Is your new server going to be a **Root CA** or a **Subordinate CA**? In some cases, even in a lot of Microsoft documentation, a subordinate CA is more often called an issuing CA. Generally, in a multi-tiered PKI, the subordinate/issuing CAs are the ones that do the issuing of certificates to users and devices in your network.

The difference really is just a matter of what you want your CA hierarchy to look like. In a PKI tree, there is a single high-level certificate, self-signed to itself by the root CA, that everything chains up to. A subordinate CA, on the other hand, is one that resides below a root CA in the tree, and it has been issued a certificate of its own from the root above it.

If your plans are to only run a single CA server, it must be a root. If you are creating a tiered approach to issuing certificates, the first CA in your environment needs to be a root, and you can slide subordinates in underneath it. You are allowed to have multiple roots, and therefore multiple trees, within a network. So your particular PKI can be structured however you see fit. In smaller companies, it is very common to see only a single CA server, an enterprise root. For the sake of simplicity in administration, these customers are willing to take the risk that, if something happens to that server, it won't be that big a deal to build a new one and re-issue certificates.

For larger networks, it is more common to see a single root with a couple of subordinates below it. Typically, in this case, the root is only

responsible for being the top dog, and the subordinate CAs are the ones doing the real work—issuing certificates to the clients.

Naming your CA server

At this point, now that you have installed the role, the hostname of the server itself is set in stone. You already knew this. But as you progress through the wizards to configure your CA for the first time, you will come across a screen called Specify the name of the CA. Huh? I thought we already did that when we set the hostname?

Nope, we do have our final hostname and that server name is plugged into Active Directory as my server is joined to the domain, but the actual "CA Name" is something else altogether. This is the name that will be identified inside the properties of every certificate that this CA issues. This is also a name that will be configured in various places inside Active Directory, since I am building an Enterprise CA. The wizard self-identifies a possible name for you to use, which many administrators simply take and use. If you want to configure your own name, this is where you should do it. Once you set the name here, this is the name of the CA forever:

CA Name

DESTINATION SERVER
CA1.contoso.local

- Credentials
- Role Services
- Setup Type
- CA Type
- Private Key
  - Cryptography
  - CA Name
  - Validity Period
- Certificate Database
- Confirmation
- Progress
- Results

Specify the name of the CA

Type a common name to identify this certification authority (CA). This name is added to all certificates issued by the CA. Distinguished name suffix values are automatically generated but can be modified.

Common name for this CA:

Contoso-CA1-CA

Distinguished name suffix:

DC=contoso,DC=local

Preview of distinguished name:

CN=Contoso-CA1-CA,DC=contoso,DC=local

Can I install the CA role onto a domain controller?

Since the role is officially called the **Active Directory Certificate Services** role, does that mean I should install this role onto one of my domain controller servers? No! Unfortunately, I have run across many small-to-medium businesses that have done exactly this, and luckily they don't have too many problems. So technically, it does work. However, it is not a Microsoft-recommended installation path and you should build your CAs on their own servers; try *not* to co-host them with *any* other roles whenever possible.

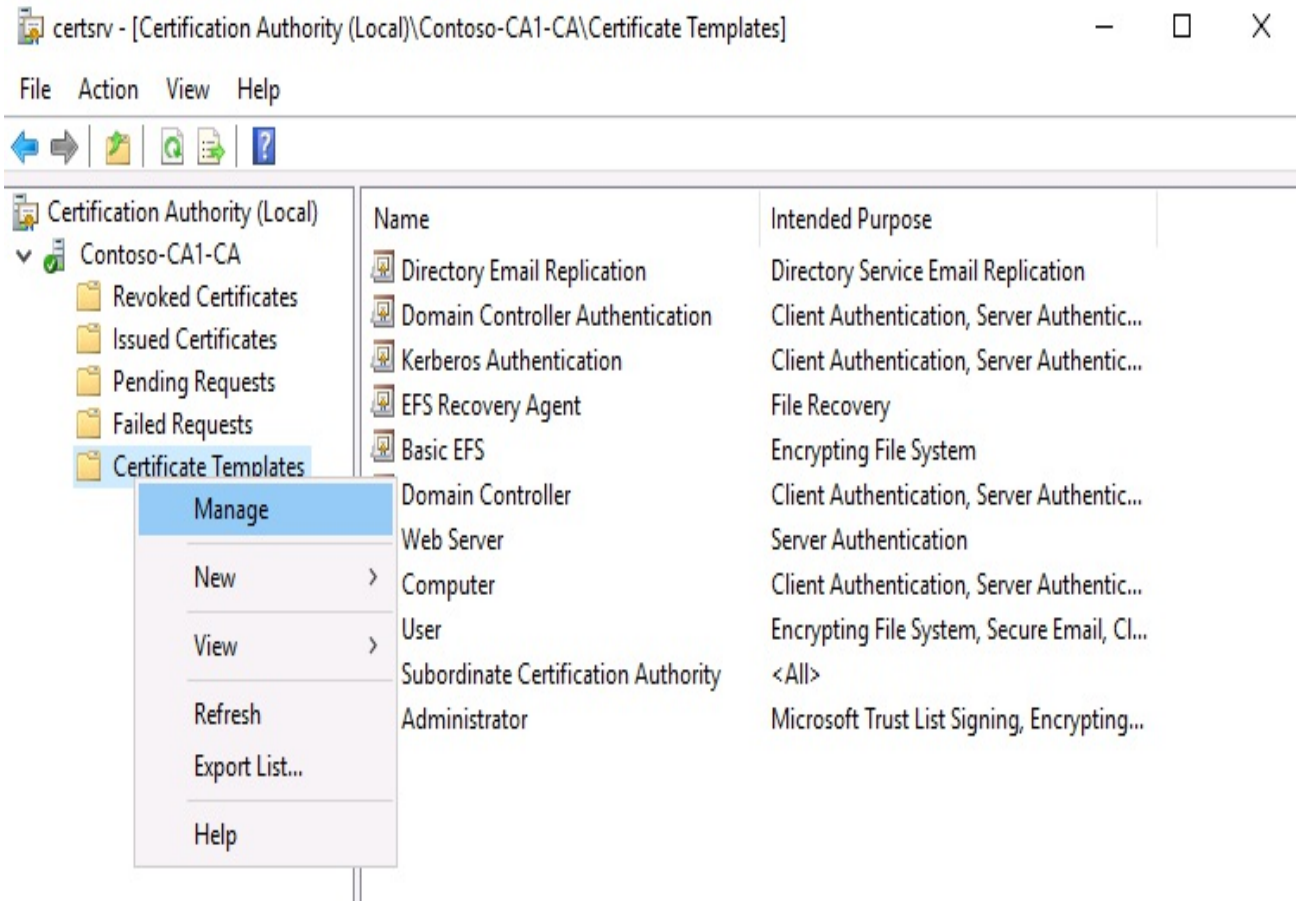
Creating a new certificate template

Enough talk, it's time to get some work done. Now that our CA role has been installed, let's make it do something! The purpose of a certificate server is to issue certificates, right? So, shall we do that? Not so fast. When you issue a certificate from a CA server to a device or user, you are not choosing which *certificate* you want to deploy; rather you are choosing which **certificate template** you want to utilize in order to deploy a certificate that is based upon the settings configured inside that template. Certificate templates are sort of like recipes for cooking. On the CA server, you build out your templates and include all of the particular ingredients, or settings, that you want to be incorporated into your final certificate. Then, when the users or computers come to request a certificate from the CA server, they are sort of baking a certificate onto their system by telling the CA which template recipe to follow when building that certificate. Certificates relating to food? Maybe that's a stretch, but it's getting pretty late at night and that's the first thing that came to mind.

When you walk through the steps to configure your first CA server, it comes with some pre-built certificate templates right in the console. In fact, one of those templates, called Computer, is typically preconfigured to the point where, if a client computer were to reach out and request a computer certificate from your new CA, it would be able to successfully issue one. However, where is the fun in using prebuilt templates and certificates? I would rather build my own template so that I can specify the particular configurations and settings inside that template. This

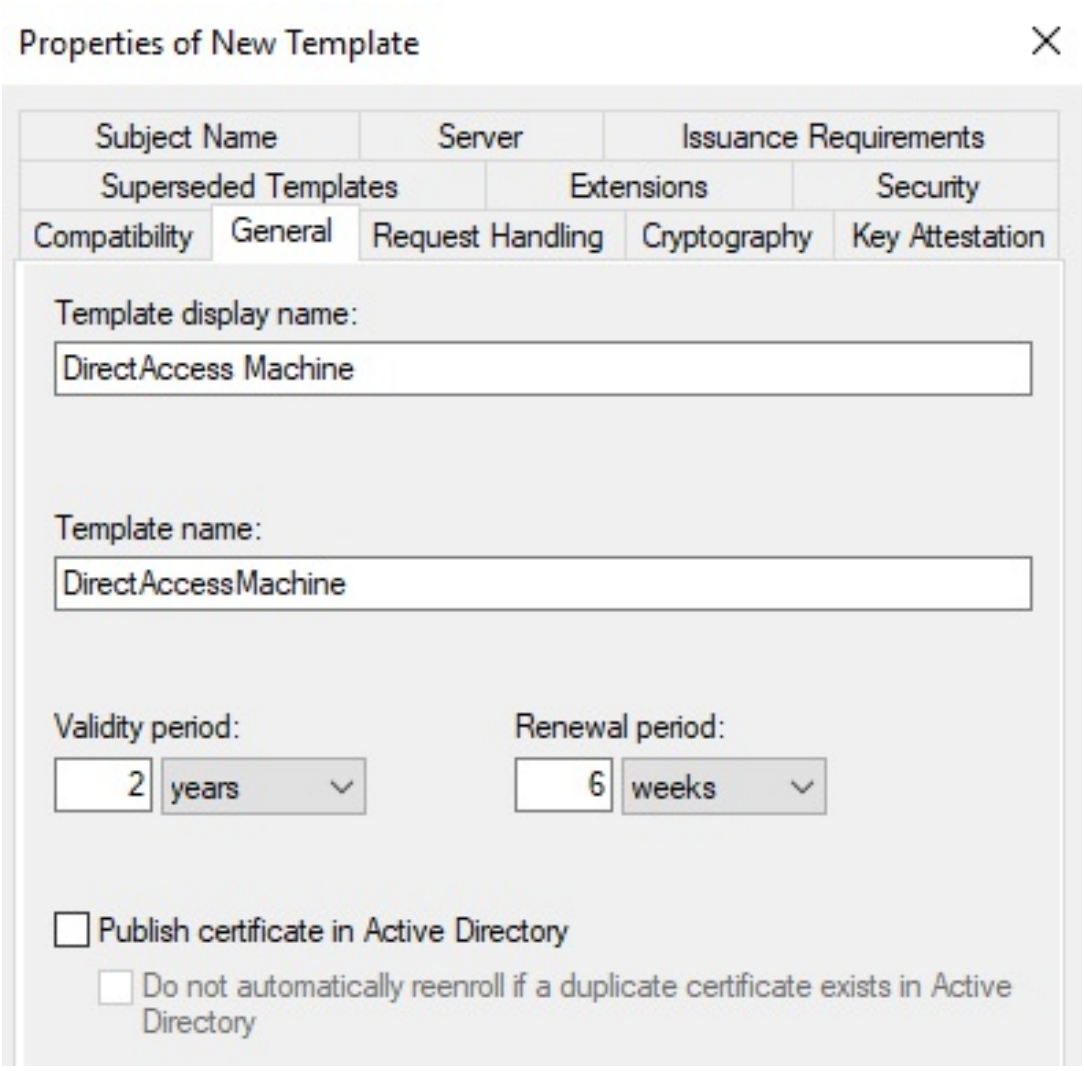
way, I know exactly what settings are contained within my certificates that will ultimately be issued to my computers in the network.

Once again, we need to launch the proper administrative console in order to do our work. Inside the Tools menu of Server Manager, click on Certification Authority. Once inside, you can expand the name of your certification authority and see some folders, including one on the bottom called Certificate Templates. If you click on this folder, you will see a list of the templates that are currently built into our CA server. Since we do not want to utilize one of these pre-existing templates, it is common sense that we would try to right-click in here and create a new template, but this is actually not the correct place to build a new template. The reason why new certificate templates are not built right from this screen must be above my pay grade, because it seems silly that it isn't, but, in order to get into a second screen where we need to go to actually manage and modify our templates, you need to right-click on the Certificate Templates folder, and then choose Manage:



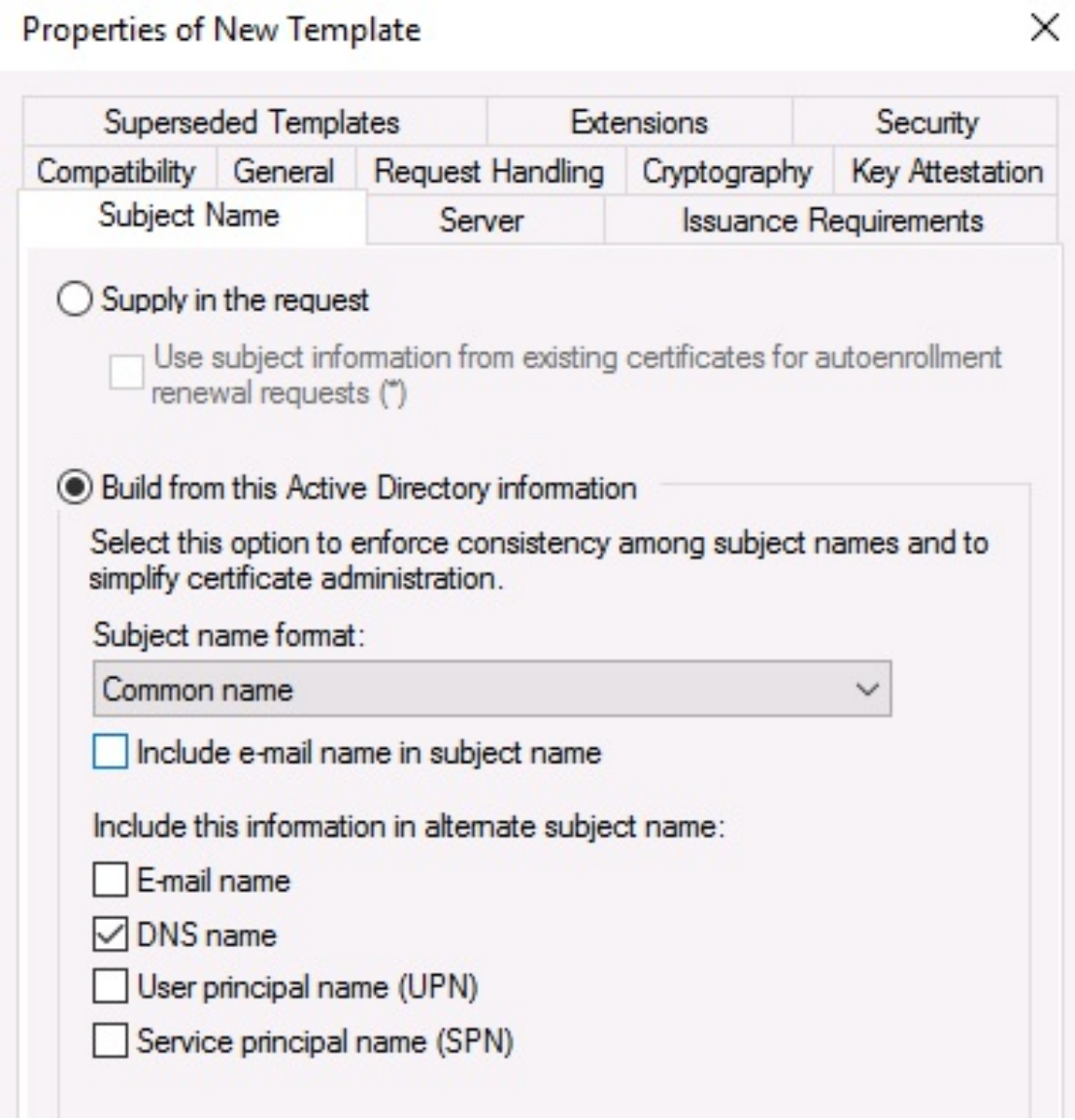
Now you see a much more comprehensive list of templates, including a number of them you couldn't view on the first screen. In order to build a new template, what we want to do is find a pre-existing template that functions similarly to the purpose that we want our new certificate template to serve. Computer templates are becoming commonly issued across many organizations due to more and more technologies requiring these certificates to exist, yet, as we said, we don't want to utilize that baked-in template, which is simply called Computer, because we want our template to have a more specific name and maybe we want the certificate's validity period to be longer than the default settings. Right-click on the built-in Computer template, and click on Duplicate Template. This opens the Properties screen for our new template, from which we first want to give our new template a unique name inside the General tab.

In an upcoming chapter, we will discuss DirectAccess, the remote access technology that will be used in our environment. A good implementation of DirectAccess includes machine certificates being issued to all of the mobile client workstations, so we will plan to make use of this new template for those purposes. The General tab is also the place where we get to define our validity period for this certificate, which we will set to 2 years:



If the certificates that you want to issue require any additional setting

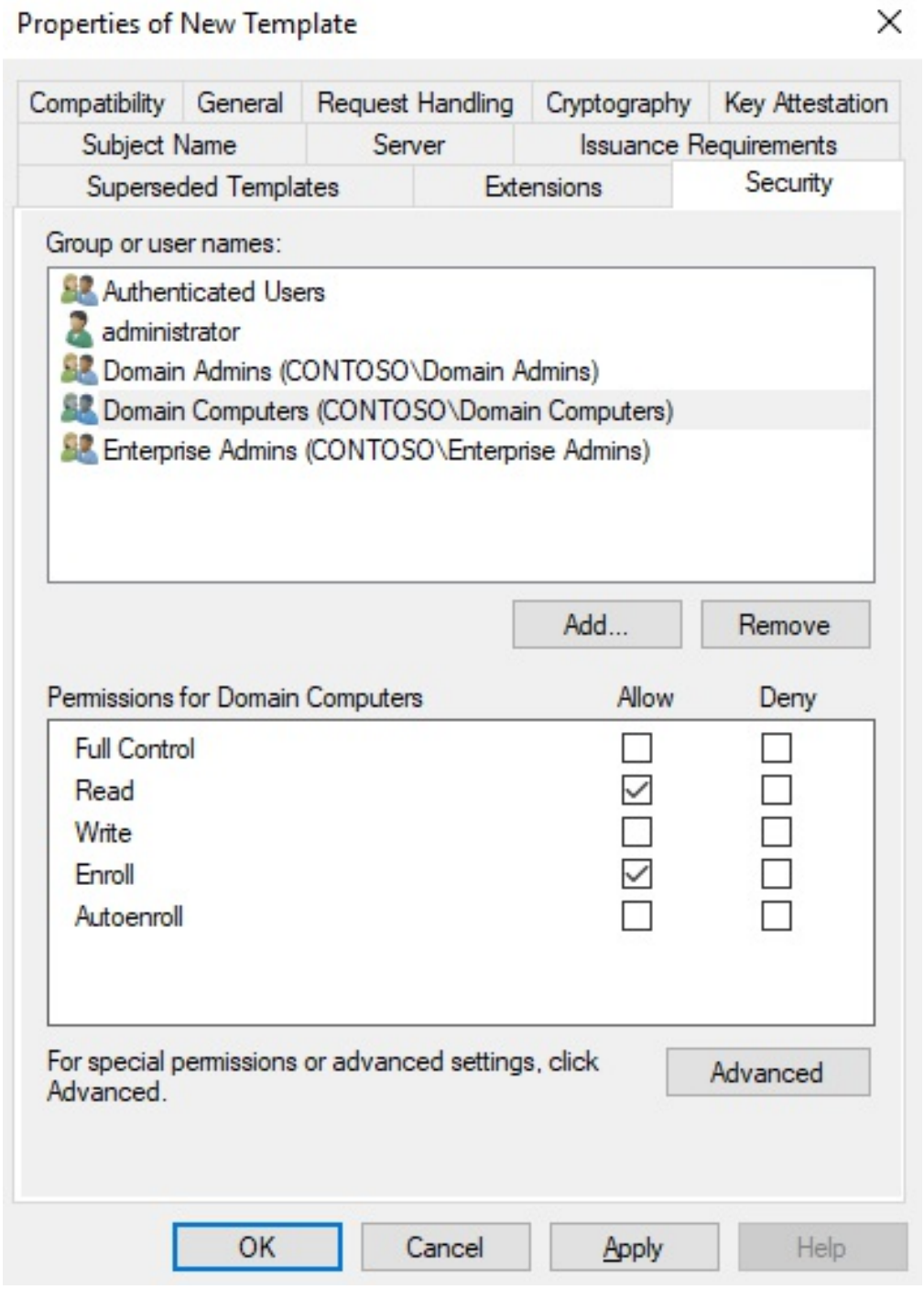
changes, you can flip through the available tabs inside properties and make the necessary adjustments. For our example, another setting I will change is inside the Subject Name tab. I want my new certificates to have a subject name that matches the common name of the computer where it is being issued, so I have chosen Common name from the drop-down list:



We have one more tab to visit, and this is something you should check

on every certificate template that you build: the Security tab. We want to check here to make sure that the security permissions for this template are set in a way that allows the certificate to be issued to the users or computers that we desire, and at the same time make sure that the template's security settings are not too loose, creating a situation where someone who doesn't need it might be able to get a certificate. For our example, I plan to issue these DirectAccess certificates to all of the computers in the domain, because the kind of machine certificate I have created could be used for general IPsec authentications as well, which I may someday configure.

So, I am just making sure that I have Domain Computers listed in the Security tab, and that they are set for Read and Enroll permissions, so that any computer that is joined to my domain will have the option of requesting a new certificate based on my new template:



Since that is everything I need inside my new certificate, I simply click on OK, and my new certificate template is now included in the list of templates on my CA server.

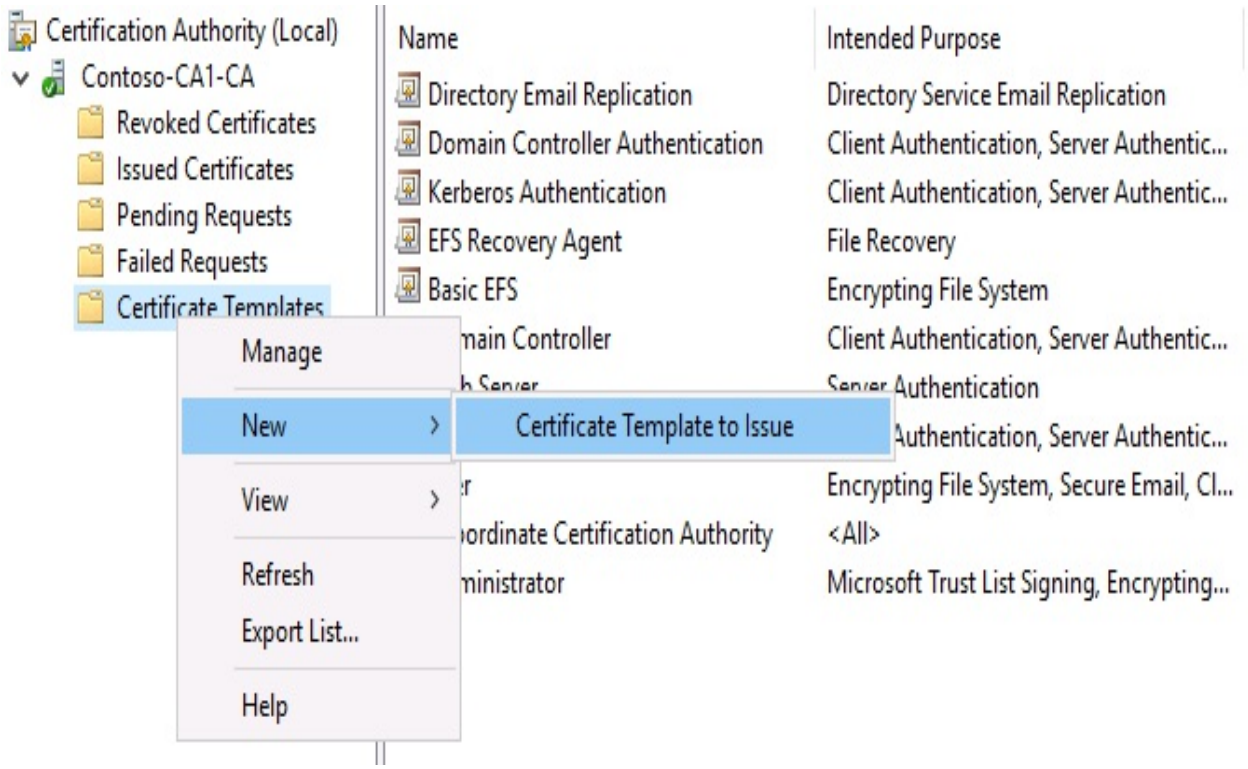
Issuing your new certificates

Next comes the part that trips up a lot of people on their first attempt. You now have a brand new template to issue, and we have verified that the permissions within that certificate template are appropriately configured so that any computer that is a member of our domain should be able to request one of these certificates, right? So our logical next step would be to jump onto a client computer and request a certificate, but there is first one additional task that needs to be accomplished in order to make that possible.

Even though the new template has been **created**, it has not yet been **published**. So at the moment, the CA server will not offer our new template as an option to the clients, even though security permissions are configured for it to do so. The process to publish a certificate template is very quick—only a couple of mouse clicks—but unless you know about the need to do this, it can be a very frustrating experience because nothing in the interface gives you a hint about this requirement.

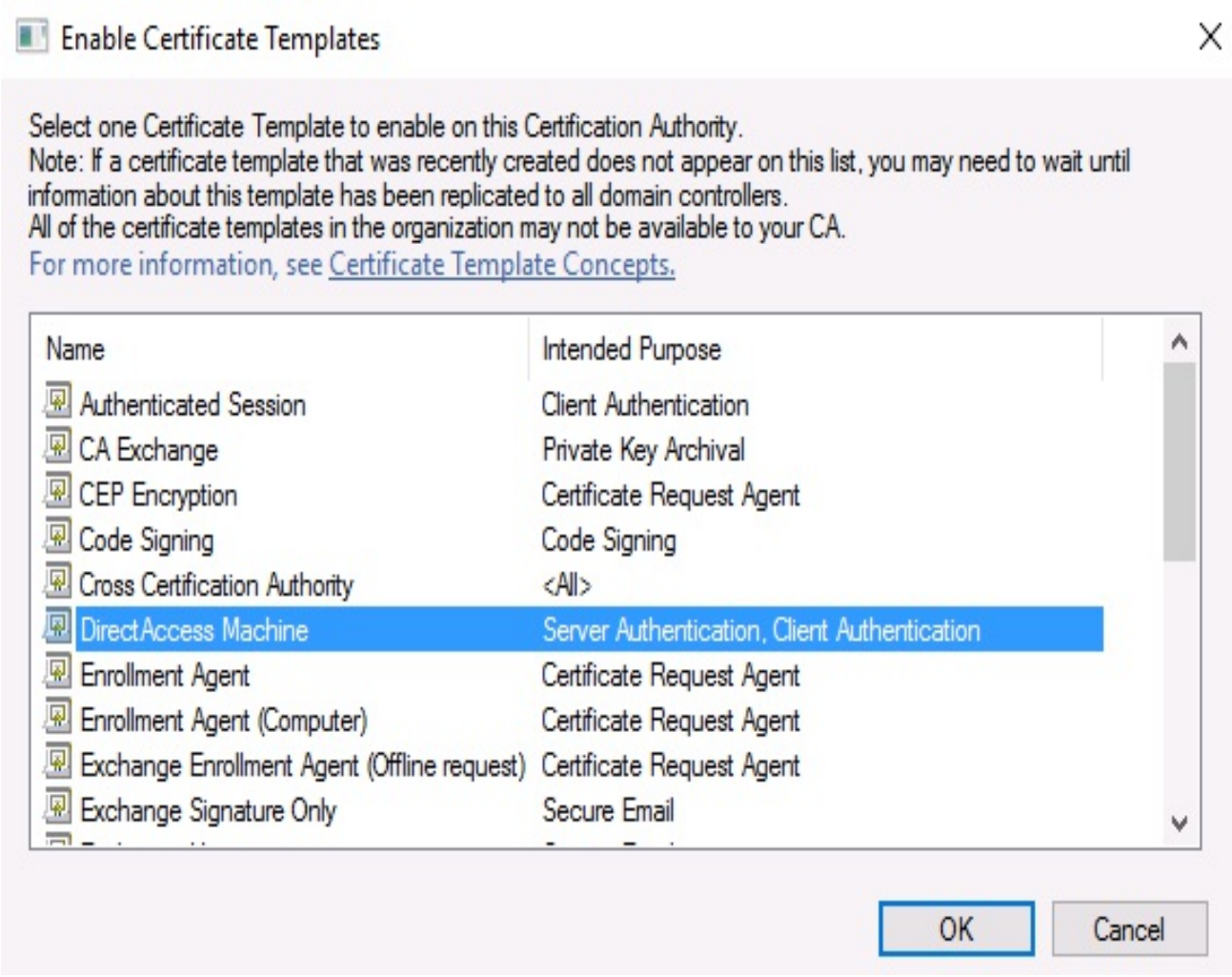
Publishing the template

If your Certificate Templates Console is still open (the one where we were managing our templates), close it so you are back at the main certification-authority management console. Remember how we noticed that the list of available certificate templates that shows up here is much shorter? This is because only these certificate templates are currently published and available to be issued. In order to add additional templates to the published list, including our new one, we simply right-click on the Certificate Templates folder and then navigate to New | Certificate Template to Issue:



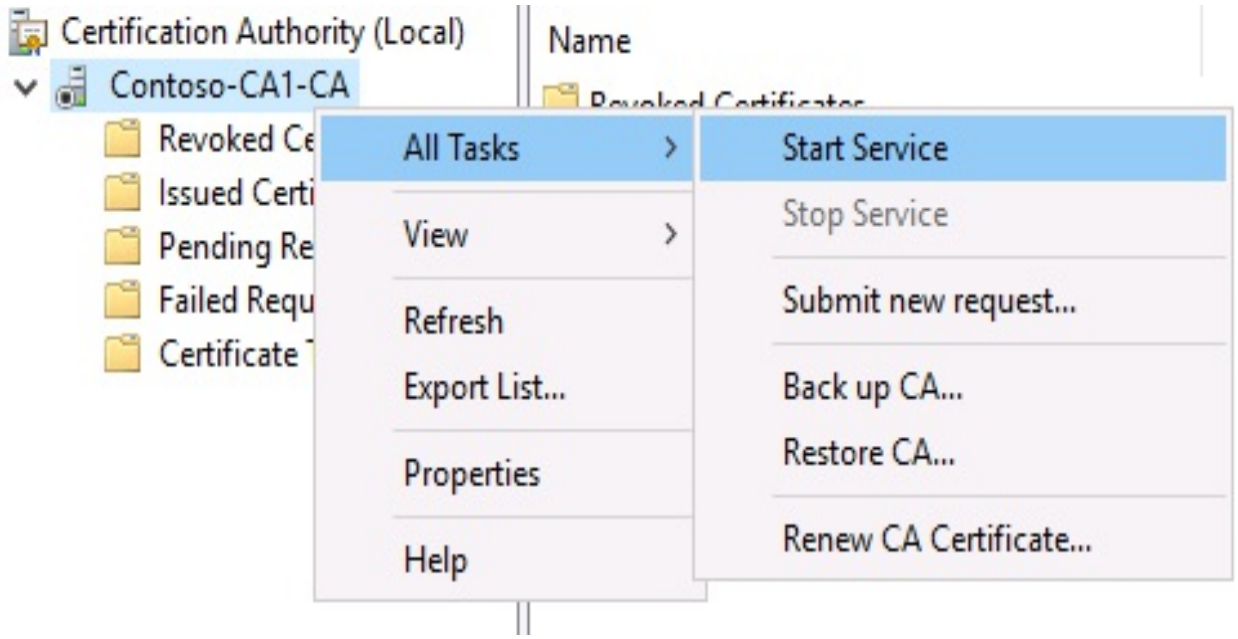
Now we are presented with a list of the available templates that are not

yet issued. All you need to do is choose your new template from the list, and click on OK. The new template is now included in the list of published certificate templates, and we are ready to request one from a client computer:



If you look through this list and do not see your newly-created template, you may have to take an additional step. Sometimes simply waiting will resolve this behavior, because occasionally the reason that the new template does not show up in the list is because you are waiting for your domain controllers to finish replicating. At other times, you will find that, even after waiting for a while, your new template is still not in this

list. In that case, you probably just need to restart the certification authority service to force it to pull in the new template information. To restart the CA service, you right-click on the CA's name near the top of the Certification Authority management console, and navigate to All Tasks | Stop Service. The stopping of that service typically only takes a second or two, and then you can immediately right-click on the CA name again, and this time navigate to All Tasks | Start Service. Now, try to publish your new template again, and you should see it in the list:



Requesting a cert from MMC

Our new certificate template has been created, and we have successfully published it within the CA console, thereby making it officially ready for issuing. It's time to test that out. Go ahead and log into a regular client computer on your network in order to do this. There are a couple of standard ways to request a new certificate on a client computer. The first is by using the good old MMC console. On your client computer, launch MMC and add the snap-in for Certificates. When you choose Certificates from the list of available snap-ins and click on the Add button, you are presented with some additional options for which certificate store you want to open. You get to choose between opening certificates for the User account, Service account, or Computer account. Since we are trying to issue a certificate that will be used by the computer itself, I want to choose Computer account from this list, and click on Finish:

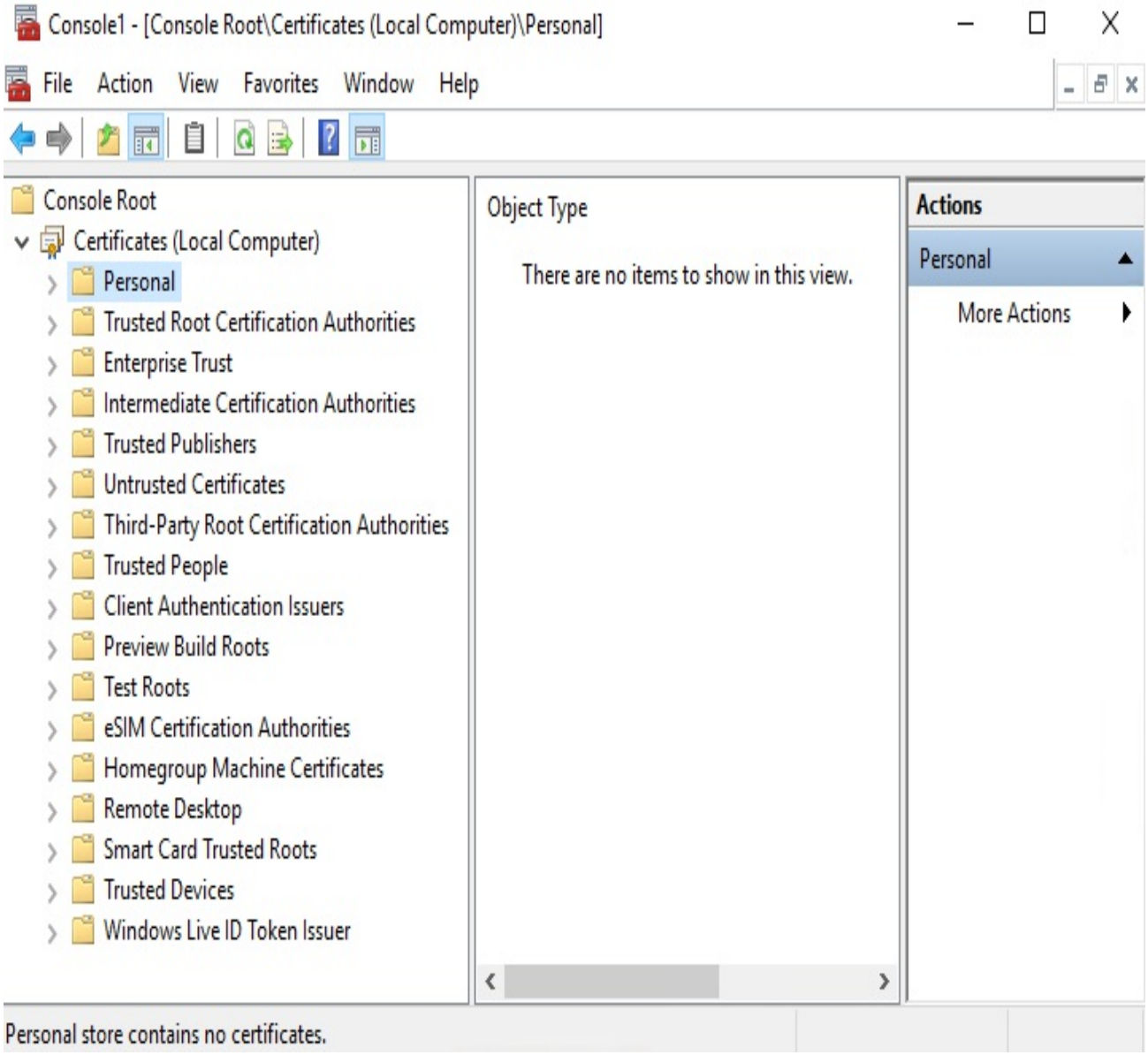


On the next page, click on the Finish button again in order to choose the default option, which is Local computer. This will snap in the local

machine's computer-based certificate store inside MMC.

> *On newer operating systems, such as Windows 8 and 10 and with Windows Server 2012, 2012R2, 2016, and 2019, there is an MSC shortcut for opening directly into the local computer's certificate store. Simply type* `CERTLM.MSC` *into a Run prompt, and MMC will automatically launch and create this snap-in for you.*

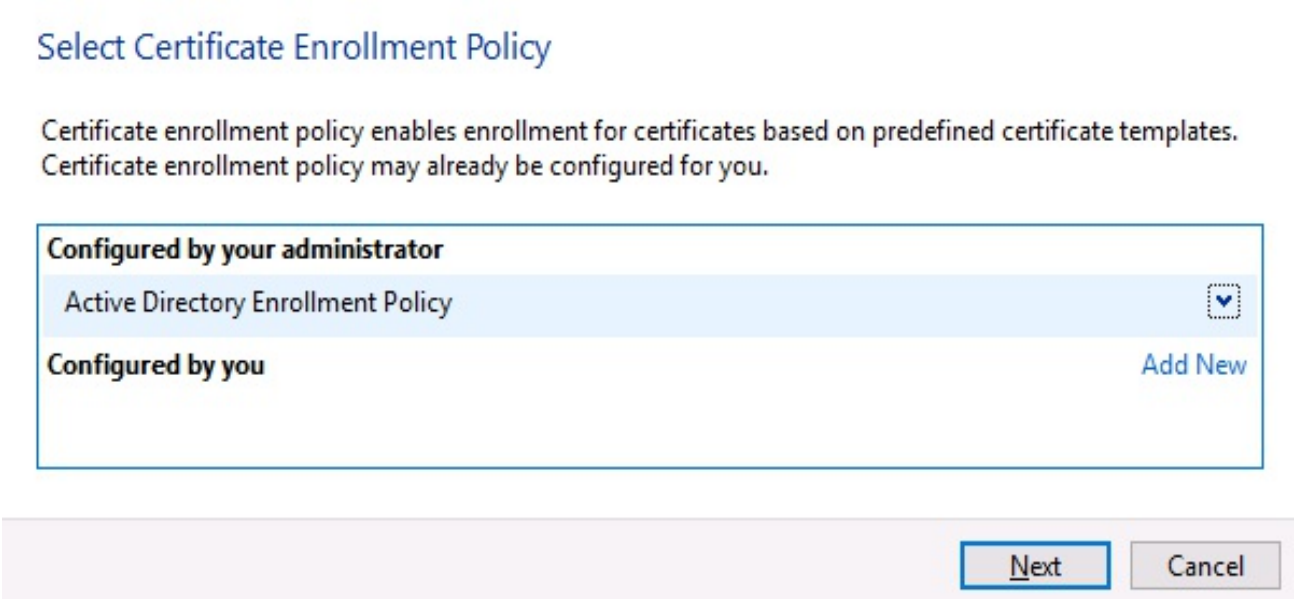
When you are installing certificates onto a computer or server, this is generally the place you want to visit. Inside this certificate store, the specific location that we want to install our certificate into is the Personal folder. This is true whether you would be installing a machine certificate as we are doing here, or if you were installing an SSL certificate onto a web server. The local computer's personal certificate folder is the correct location for both kinds of certificates. If you click on Personal, you can see that we do not currently have anything listed in there:



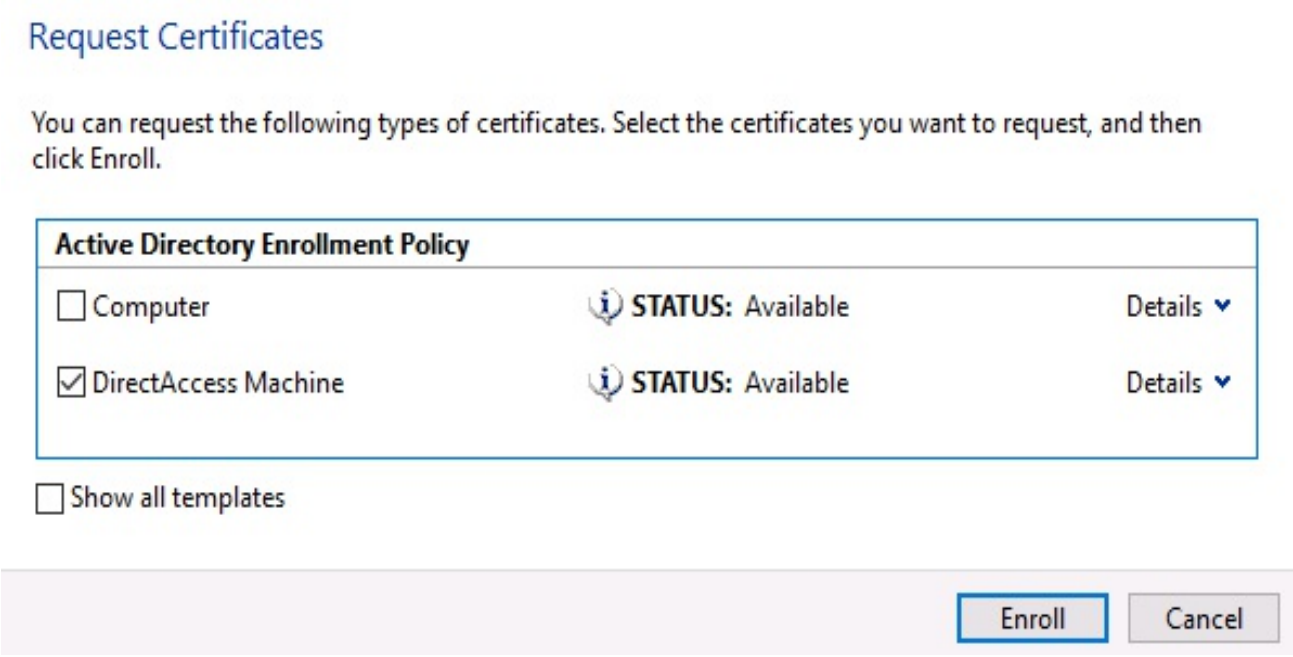
To request a new certificate from our CA server, we simply right-click on the Personal folder, and then navigate to All Tasks | Request New Certificate.... Doing so opens a wizard; go ahead and click on the Next button once.

Now you have a screen that looks like something needs to be done, but in most cases because we are requesting a certificate on one of our

corporate, domain-joined machines, we actually do not need to do anything on the screen presented in the following screenshot. Simply click on Next and the wizard will query Active Directory in order to show all of the certificate templates that are available to be issued:

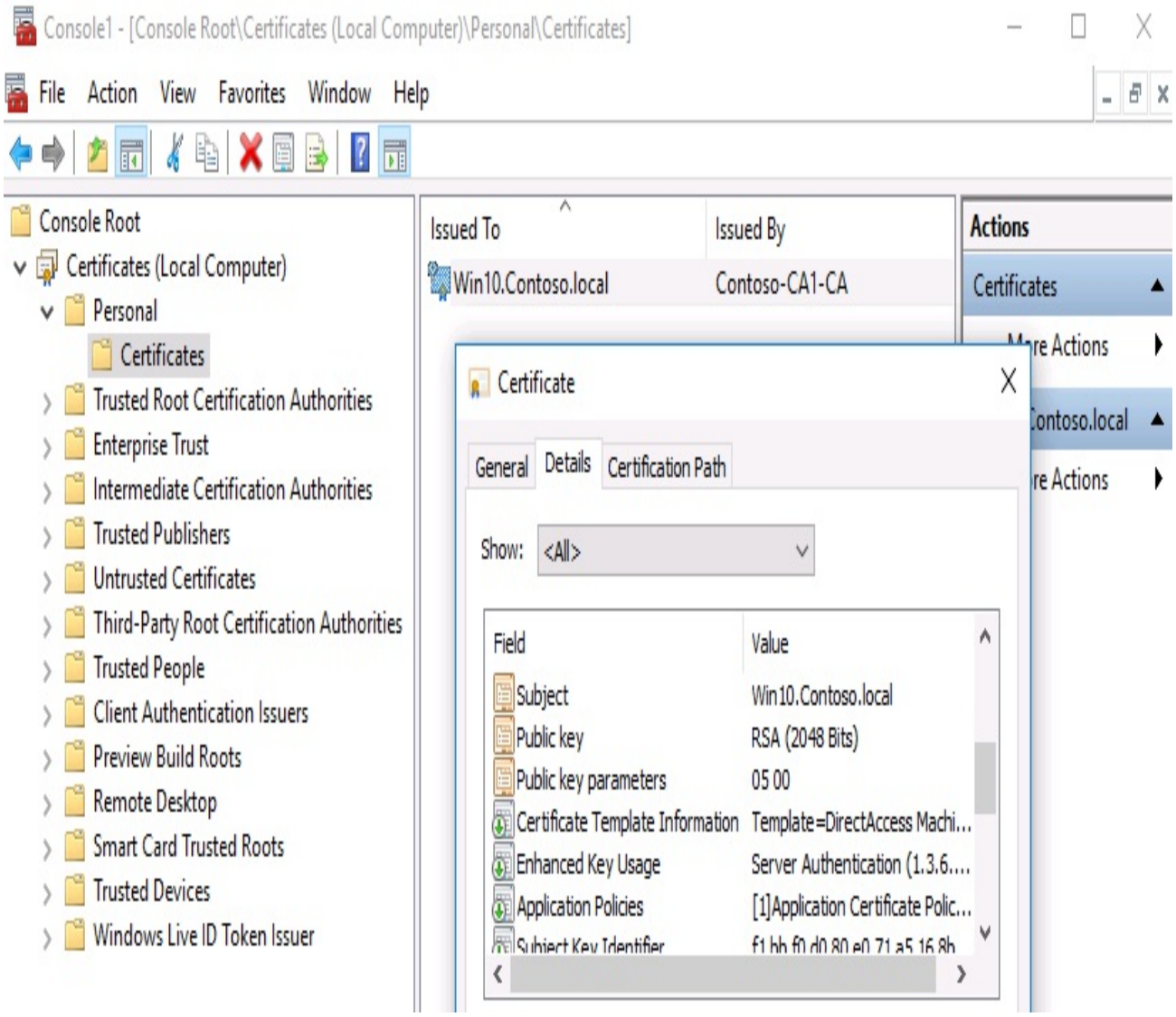


The Request Certificates screen is shown, which is the list of templates that are available to us. This list is dynamic; it is based on what computer you are logged into and what your user account permissions are. Remember when we set up the security tab of our new certificate template? It is there that we defined who and what could pull down new certificates based on that template, and, if I had defined a more particular group than domain computers, it is possible that my new DirectAccess Machine template would not be displayed in this list. However, since I did open up that template to be issuable to any computer within our domain, I can see it here:



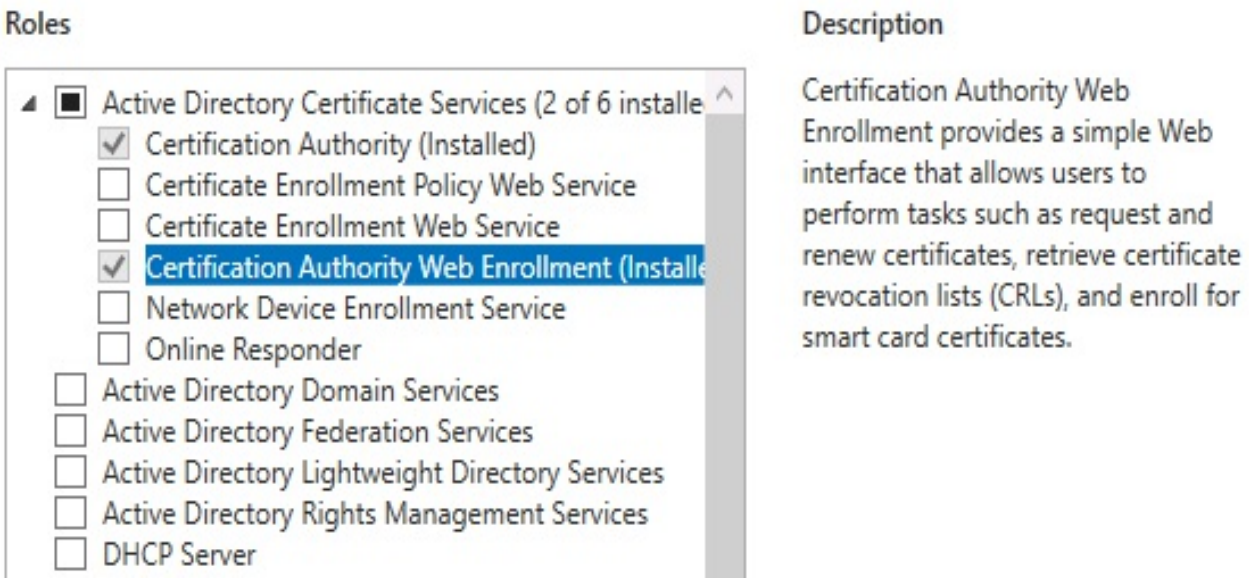
If you do not see your new template in the list, click on the checkbox for Show all templates. This will give you a full list of all the templates on the CA server, and a description on each one as to the reason that it is currently unavailable for issuing.

Put a checkmark next to any certificates that you want, and click on Enroll. Now the console spins for a few seconds while the CA server processes your request and issues a new certificate that is specific to your computer and the criteria that we placed inside the certificate template. Once finished, you can see that our brand new machine certificate is now inside Personal | Certificates in the MMC. If you double-click on the certificate, you can check over its properties to ensure all of the settings you wanted to be pushed into this cert exist:


Console1 - [Console Root\Certificates (Local Computer)\Personal\Certificates]
File Action View Favorites Window Help
Console Root
Certificates (Local Computer)
Personal
Certificates
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authorities
Trusted People
Client Authentication Issuers
Preview Build Roots
Remote Desktop
Smart Card Trusted Roots
Trusted Devices
Windows Live ID Token Issuer
Issued To
Issued By
Win10.Contoso.local
Contoso-CA1-CA
Actions
Certificates
Certificate
General
Details
Certification Path
Show:
<All>
Field
Value
Subject
Win10.Contoso.local
Public key
RSA (2048 Bits)
Public key parameters
05 00
Certificate Template Information
Template=DirectAccess Machi...
Enhanced Key Usage
Server Authentication (1.3.6....
Application Policies
[1]Application Certificate Polic...

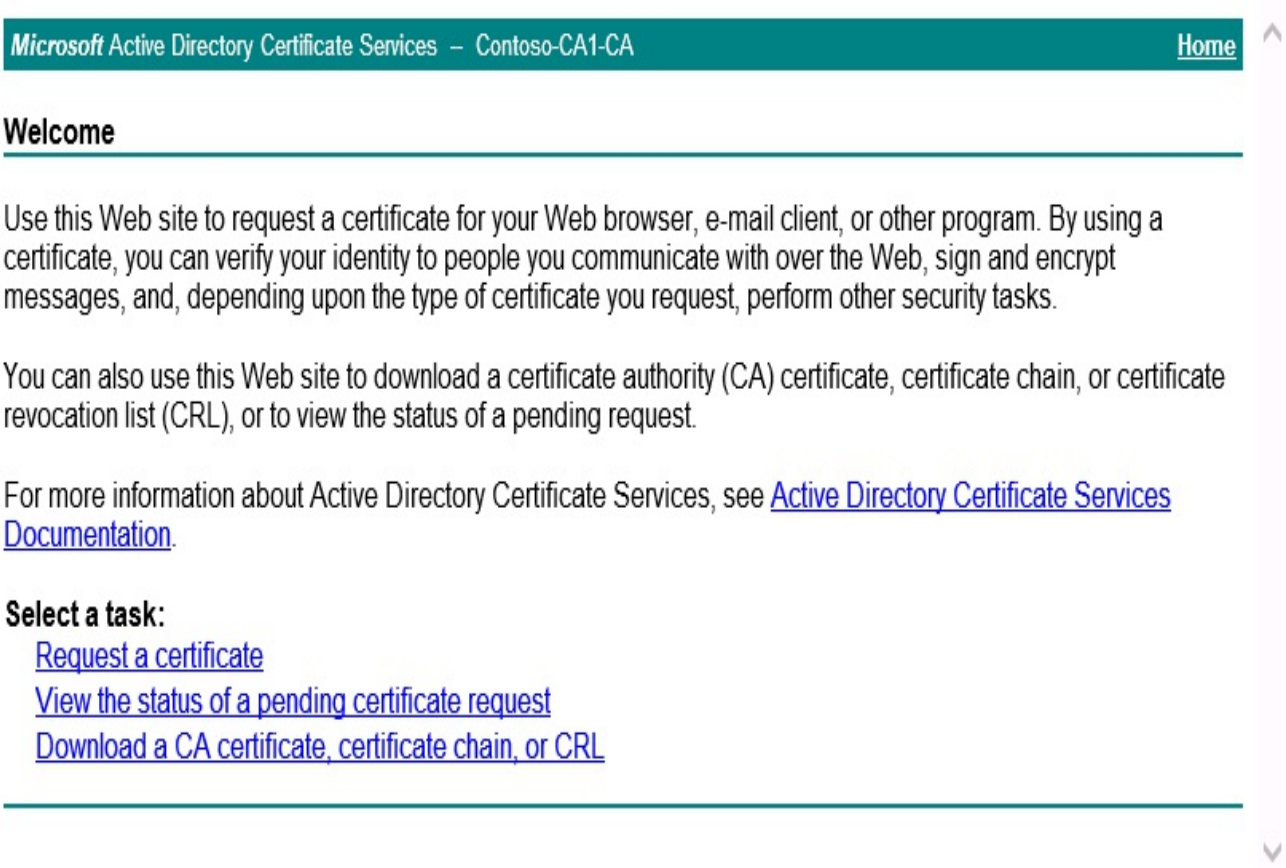
Requesting a cert from the Web interface

I typically use the MMC for requesting certificates whenever possible, but, in most cases, there is another platform from which you can request and issue certificates. I say *in most cases* because the existence of this option depends upon how the CA server was built in the first place. When I installed my AD CS role, I made sure to choose the options for both Certification Authority and Certification Authority Web Enrollment. This second option is important for our next section of text. Without the Web enrollment piece of the role, we would not have a web interface running on our CA server, and this part would not be available to us. If your CA server does not have Web enrollment turned on, you can revisit the role installation page in Server Manager and add it to the existing role:



Once Certification Authority Web Enrollment is installed on your CA, there is now a website running on that server that you can access via a browser from inside your network. Having this website is useful if you have the need for users to be able to issue their own certificates for some reason; it would be much easier to give them documentation or train them on the process of requesting a certificate from a website than expecting them to navigate the MMC console. Additionally, if you are trying to request certificates from computers that are not within the same network as the CA server, using MMC can be difficult. For example, if you have the need for a user at home to be able to request a new certificate, without a full VPN tunnel the MMC console is more than likely not going to be able to connect to the CA server in order to pull down that certificate. But since we have this certificate-enrollment website running, you could externally publish this website like you do with any other website in your network, using a reverse proxy or firewall in order to keep that traffic safe, and present users with the ability to hit this site and request certificates from wherever they are.

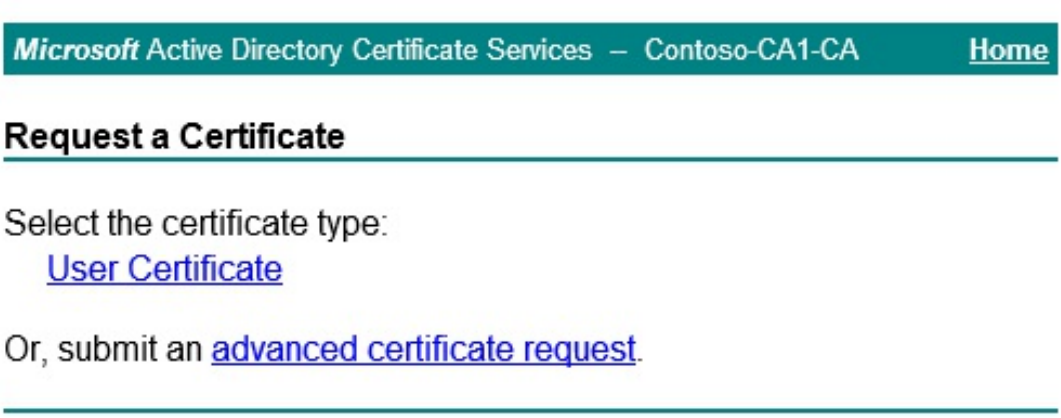
To access this website, let's use our regular client computer again. This time, instead of opening MMC, I will simply launch Internet Explorer, or any other browser, and log into the website running at `https://<CASERVER>/certsrv`. For my specific environment, that exact web address is `https://CA1/certsrv`:



Our URL starts with HTTPS. This website must be configured to run on HTTPS instead of regular HTTP in order to allow the website to request certificates. It does not allow issuing certificates over HTTP because that information would be traveling in cleartext to the client. Enabling the website on the CA server for HTTPS ensures that the certificate issued will be encrypted while it travels.

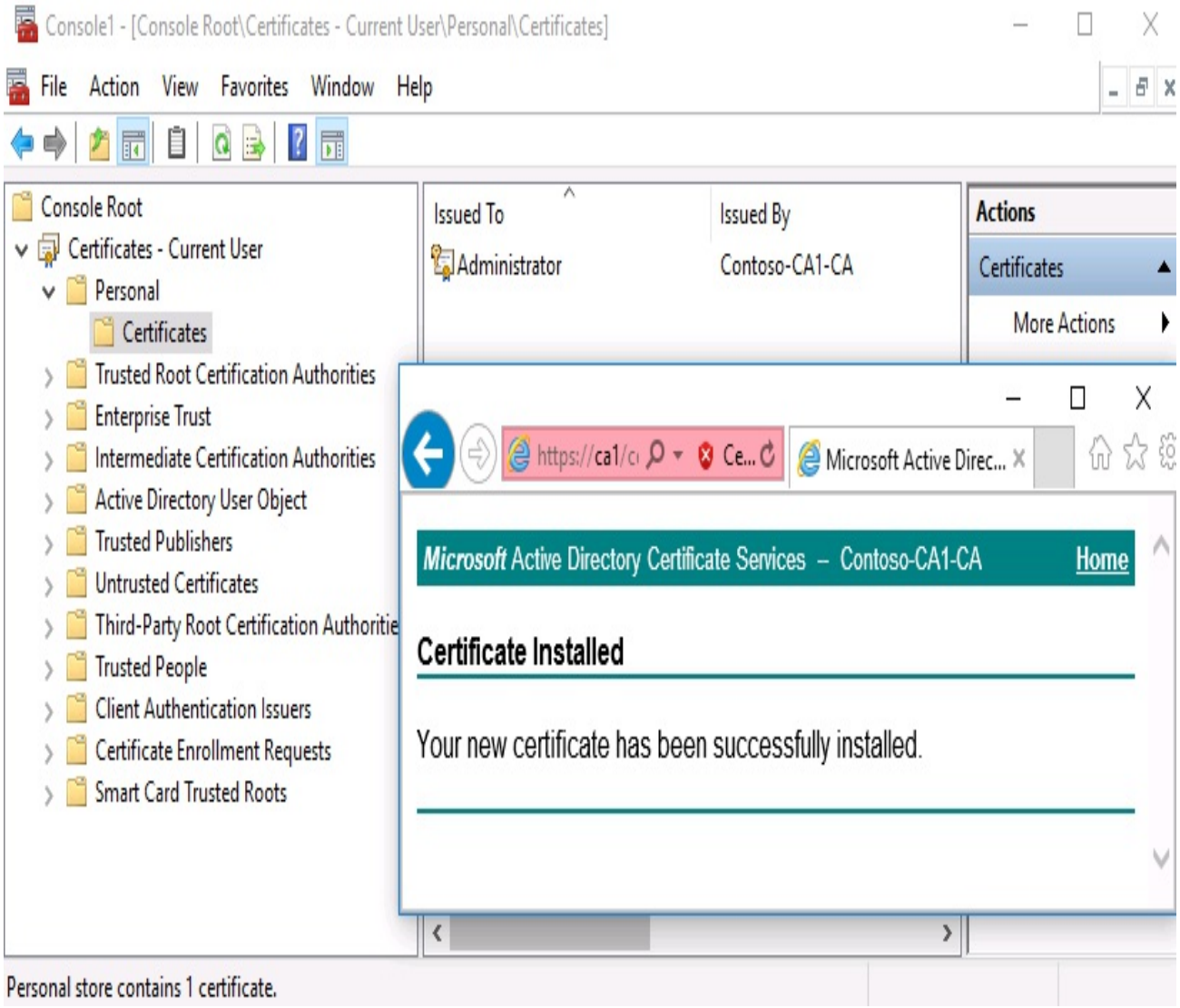
Clicking on the Request a certificate link brings you into our wizard in which we can request a new certificate from the CA server. When you have users driving their own way through this web interface, it is typically for the purpose of a user-based certificate, since we have some pretty easy ways of automatically distributing computer-level certificates without any user interaction. We will discuss that in a moment. However, for this example, since we are asking our users to log in here and request a new User Certificate, on the next page, I will choose that link:

HivaNetwork.Com



If you were not interested in a user certificate and wanted to use the web interface to request a machine certificate, a web server certificate, or any other kind of certificate, you could instead choose the link for advanced certificate request and follow the prompts to do so.

Next, press the Submit button, and, once the certificate has been generated, you will see a link that allows you to Install this certificate. Click on that link, and the new certificate that was just created for you has now been installed onto your computer. You can see in the following screenshot the response that the website gave me, indicating a successful installation, and you can also see I have opened up the current user certificates inside MMC in order to see and validate that the certificate really exists:


Console1 - [Console Root\Certificates - Current User\Personal\Certificates]
File Action View Favorites Window Help
Console Root
Certificates - Current User
Personal
Certificates
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Active Directory User Object
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authorities
Trusted People
Client Authentication Issuers
Certificate Enrollment Requests
Smart Card Trusted Roots
Issued To
Issued By
Administrator
Contoso-CA1-CA
Actions
Certificates
More Actions
Microsoft Active Direc...
Microsoft Active Directory Certificate Services – Contoso-CA1-CA
Home
Certificate Installed
Your new certificate has been successfully installed.
Personal store contains 1 certificate.

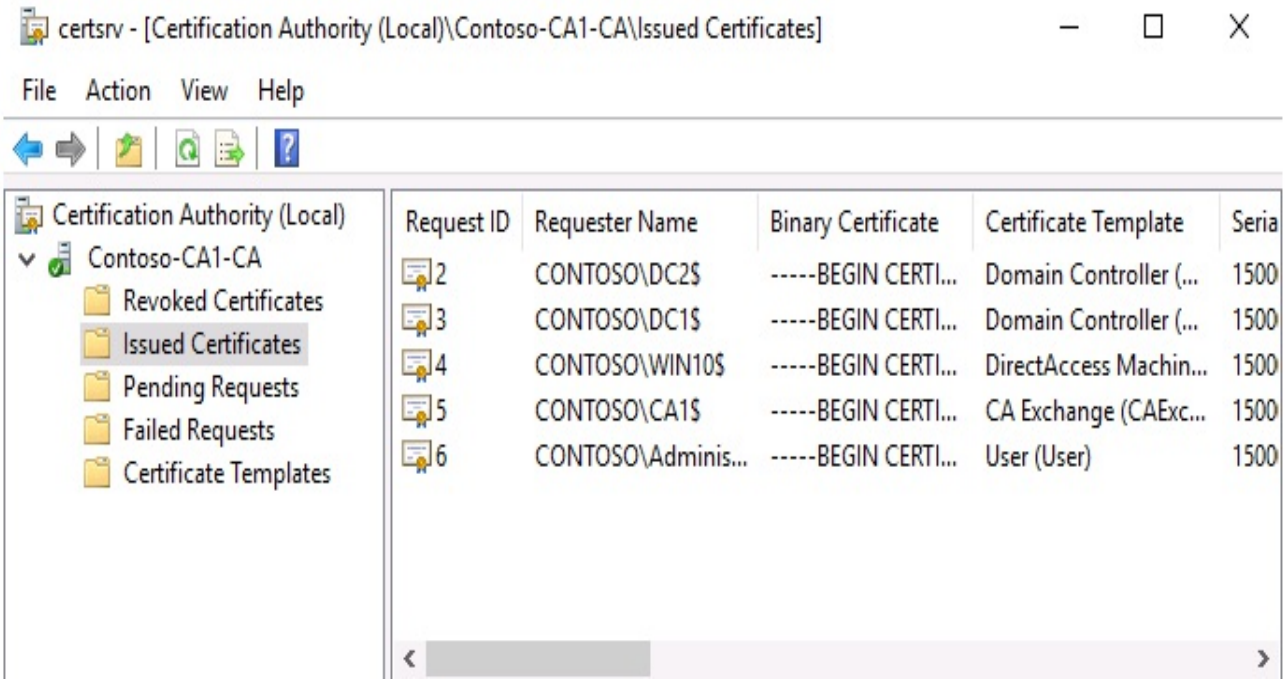
Creating an auto-enrollment policy

Our certification authority server is configured and running, and we can successfully issue certificates to the client machines. Great! Now let's pretend we have a new project on our plates, and one of the requirements for this project is that all of the computers in your network need to have a copy of this new machine certificate that we have created. Uh oh, that sounds like a lot of work. Even though the process for requesting one of these certificates is very quick—only a handful of seconds on each workstation—if you had to do that individually on a couple of thousand machines, you are talking about a serious period of time needing to be spent on this process. Furthermore, in many cases, the certificates that you issue will only be valid for one year. Does this mean I am facing an extreme amount of administrative work every single year to re-issue these certificates as they expire? Certainly not!

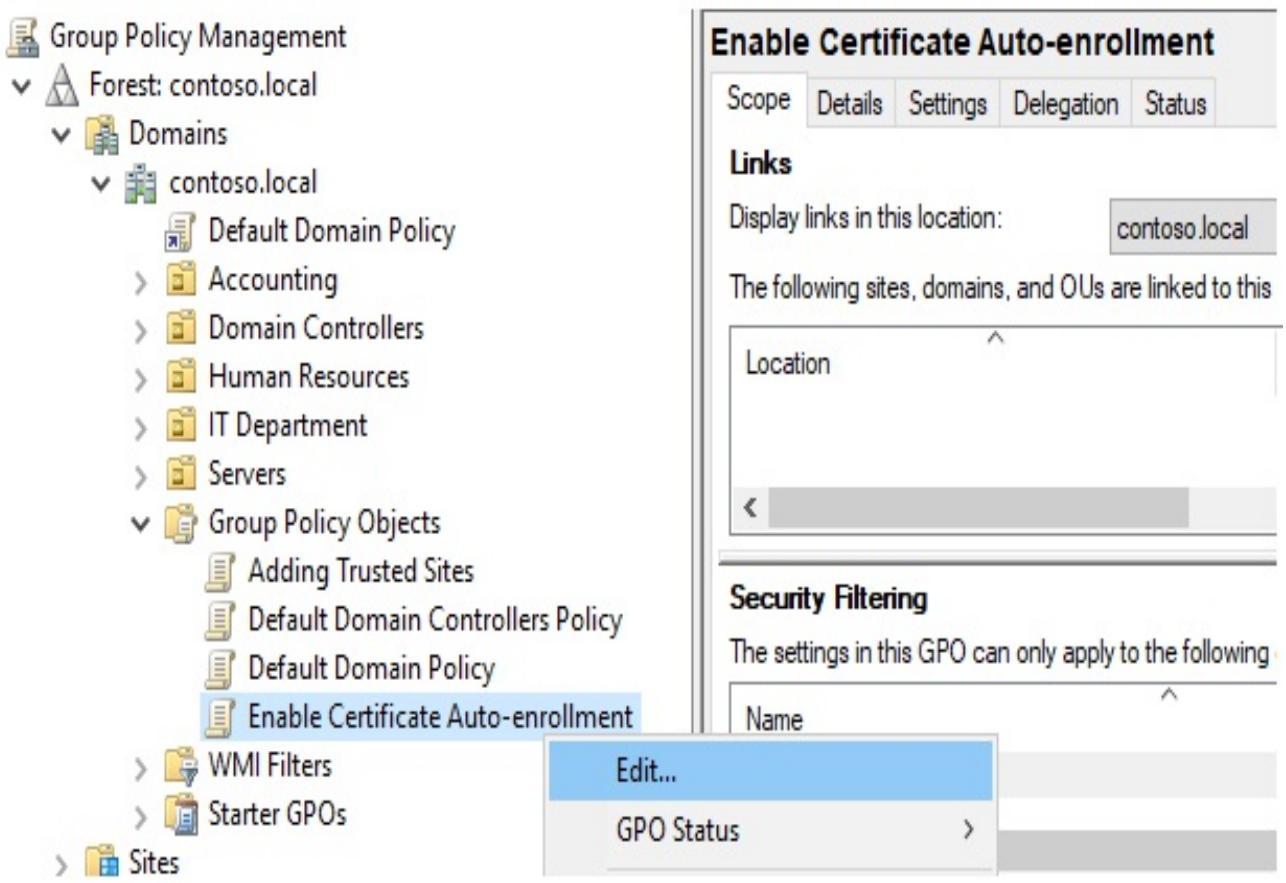
Let's figure out how to utilize Group Policy to create a GPO that will **auto-enroll** our new certificates to all of the machines in the network, and, while we are in there, also configure it so that when a certificate's expiration date comes up, the certificate will auto-renew at the appropriate intervals.

Let's pop into the Certification Authority management console on our CA server, and take a look inside the Issued Certificates folder. I only want to look here for a minute in order to see how many certificates we have issued so far in our network. It looks like just a handful of them,

so hopefully once we are done configuring our policy, if we have done it correctly, and it takes effect automatically, we should see more certificates starting to show up in this list:

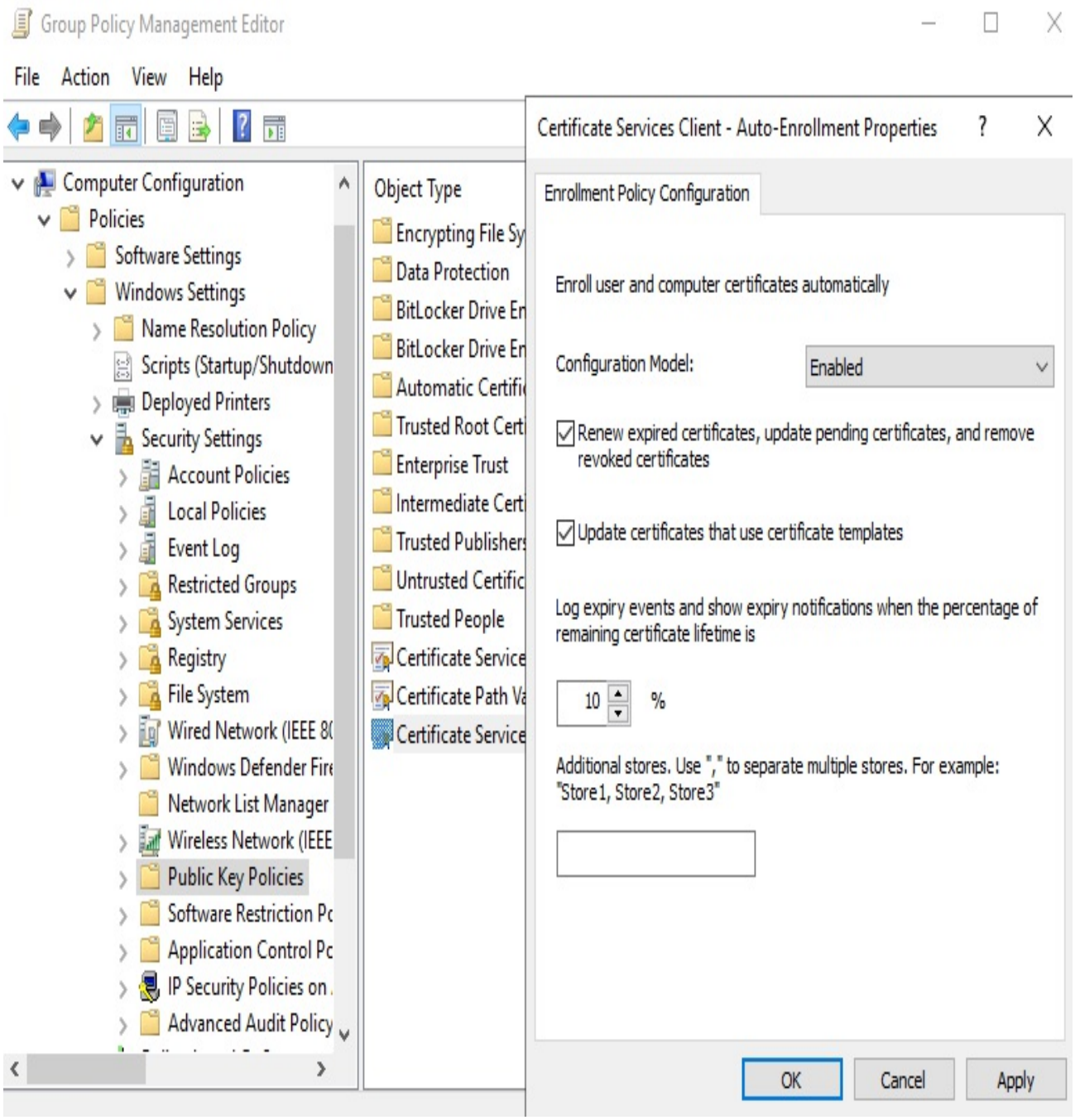


Log into a domain controller server, and then open up the Group Policy Management console. I have created a new GPO called Enable Certificate Auto-enrollment, and am now editing that GPO to find the settings I need to configure in order to make this GPO do its work:



The settings inside this GPO that we want to configure are located at Computer Configuration | Policies | Windows Settings | Security Settings | Public Key Policies | Certificate Services Client – Auto-Enrollment.

Double-click on this setting in order to view its properties. All we need to do is change Configuration Model to Enabled, and make sure to check the box that says Renew expired certificates, update pending certificates, and remove revoked certificates. Also check the box for Update certificates that use certificate templates. These settings will ensure that auto-renewal happens automatically when the certificates start running into their expiration dates over the next few years:



What is the last thing we need to do on our GPO in order to make it live? Create a link so that it starts applying! For your own environment, you will probably create a more specific link to a particular OU, as we discussed in the last chapter, but, for my lab, I want these certificates to

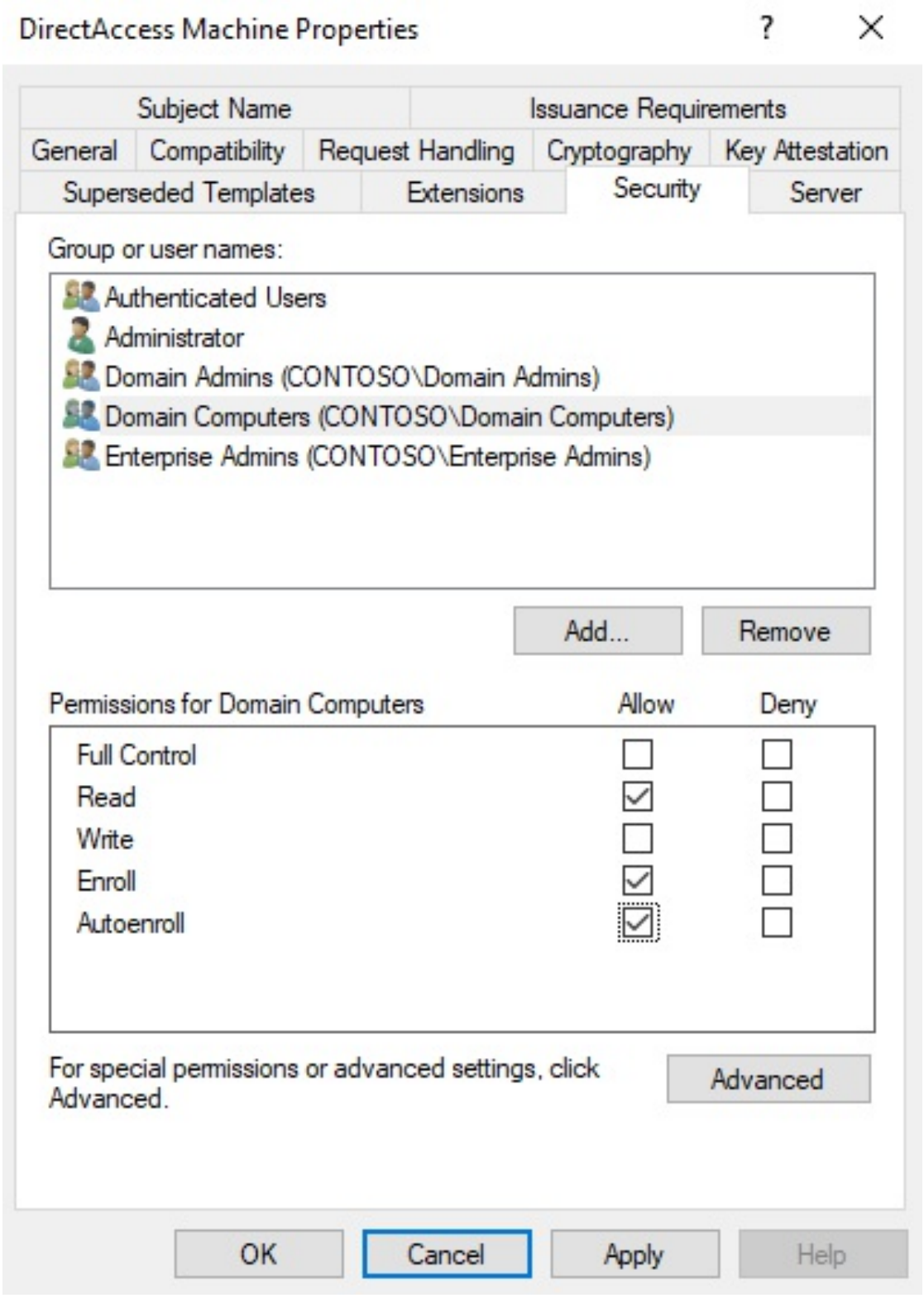
apply to every single machine that is joined to the domain, so I will link my new GPO at the root of the domain, so that it applies to all of my clients and servers.

Now that the GPO is created and configured, and we have linked it to the domain, I would think that some new certificates would be issued and there would be more names shown inside my Issued Certificates folder inside my certification authority console. But there are not. Wait a minute, in our GPO we didn't really specify anything particular to my DirectAccess Machine cert template, did we? Could that be the problem? No, there wasn't really an option for specifying which template I wanted to set up for auto-enrollment.

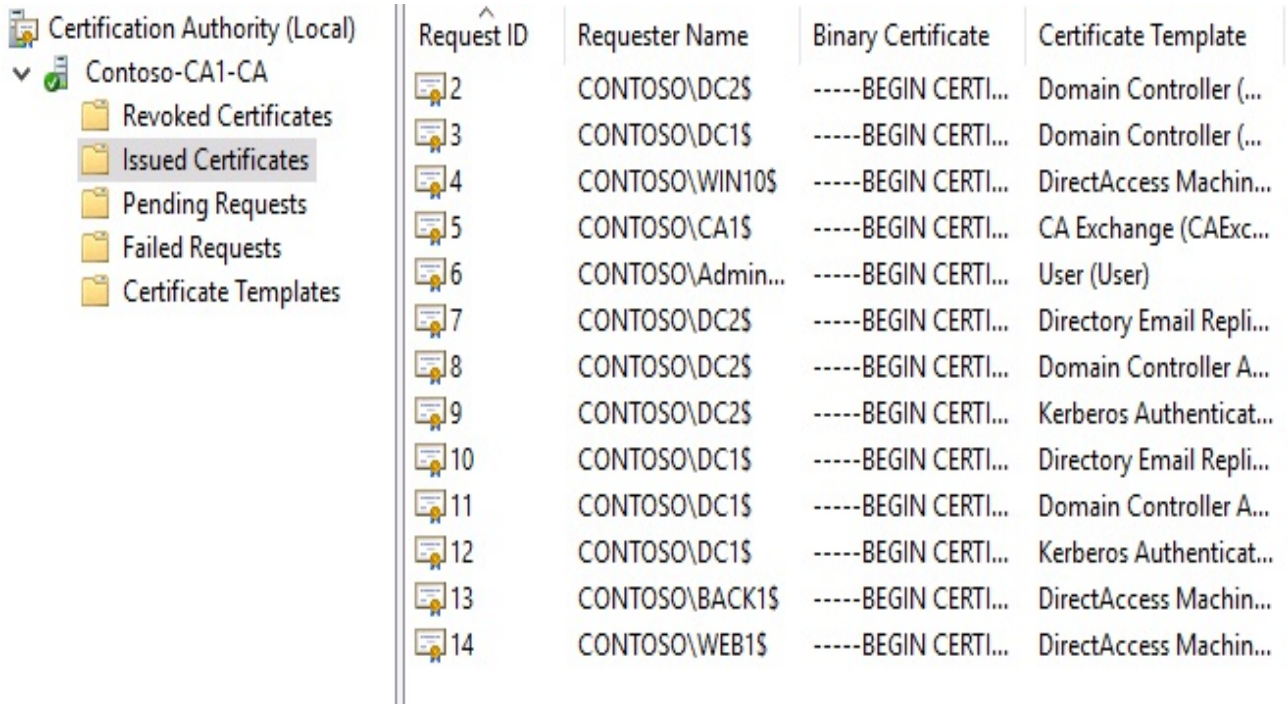
When you enable auto-enrollment in Group Policy, you are simply flipping an on/off switch and turning it on for *every* certificate template. So now that we have a policy that is configured to enable auto-enrollment and is linked to the domain, thus making it *live*, auto-enrollment has been enabled on every domain-joined computer, for **every certificate template** that is published on our CA server. Yet, none of them are issuing themselves to my computers. This is because we need to adjust the security settings on our new DirectAccess Machine template. Currently we have it configured so that all domain computers have Enroll permissions, but if you remember that security tab within the cert template's properties, there was an additional security identifier called Autoenroll. Every certificate template has the autoenroll permission identifier, and it is not allowed by default. Now that the light switch has been flipped ON for auto-enrollment in our domain, we need to enable the autoenroll permission on any template that we want to start distributing itself. As soon as we enable that permission, these certificates will start flowing around our network.

Head into the certificate-management section of your CA server and open the Properties of your new template, then make your way to the

Security tab and allow Autoenroll permissions for the Domain Computers group. This should tell the CA to start distributing these certificates accordingly:


DirectAccess Machine Properties
?
Subject Name
Issuance Requirements
General
Compatibility
Request Handling
Cryptography
Key Attestation
Superseded Templates
Extensions
Security
Server
Group or user names:
Authenticated Users
Administrator
Domain Admins (CONTOSO\Domain Admins)
Domain Computers (CONTOSO\Domain Computers)
Enterprise Admins (CONTOSO\Enterprise Admins)
Add...
Remove
Permissions for Domain Computers
Allow
Deny
Full Control
Read
Write
Enroll
Autoenroll
For special permissions or advanced settings, click Advanced.
Advanced
OK
Cancel
Apply
Help

And sure enough, if I let my environment sit for a little while, giving Active Directory and Group Policy a chance to update on all of my machines, I now see more certificates have been issued from my CA server:



In order to automatically issue certificates from any template you create, simply publish the template and make sure to configure the appropriate autoenroll permissions on that template. Once the auto-enrollment GPO is in place on those clients, they will reach out to your CA server and ask it for certificates from any template for which they have permissions to receive a certificate. In the future, when that certificate is about to expire and the machine needs a new copy, the auto-enrollment policy will issue a new one prior to the expiration date, based upon the timestamps you defined inside the GPO.

Certificate auto-enrollment can take what would normally be an enormous administrative burden, and turn it into a completely automated process!

Obtaining a public-authority SSL certificate

We are now pretty comfortable with grabbing certificates from our own CA server inside our own network, but what about handling those SSL certificates for our web servers that should be acquired from a public certification authority? For many of you, this will be the most common interaction that you have with certificates, and it's very important to understand this side of the coin as well. When you need to acquire an SSL certificate from your public authority of choice, there is a three-step process to do so: create a certificate request, submit the certificate request, and install the resulting certificate. We are going to use my WEB1 server, on which I have a website running. Currently the site is only capable of handling HTTP traffic, but when we turn it loose on the internet we need to enable HTTPS to keep the information that is being submitted onto the site encrypted.

In order to use HTTPS, we need to install an SSL certificate onto the WEB1 server. This web server is running the Microsoft web services platform, **Internet Information Services** (**IIS**). The three-step process we take is the same if you are running a different web server, such as Apache, but the particular things that you have to do to accomplish these three steps will be different, because Apache or any other web server will have a different user interface than IIS. Since we are working on a Windows Server 2019 web server, we are utilizing IIS 10.

Public/private key pair

Before we jump into performing those three steps, let's discuss why any of this even matters. You have probably heard of the term **private key**, but may not quite understand what that means. When we send traffic across the internet, from our client computers to an HTTPS website, we understand that the traffic is encrypted. This means that the packets are tied up in a nice little package before leaving my laptop so that nobody can see them while they travel, and are then unwrapped successfully when those packets reach the web server. My laptop uses a key to encrypt the traffic, and the server uses a key to decrypt that traffic, but how do they know what keys to use? There are two different kinds of encryption methodology that can be used here:

- **Symmetric encryption**: The simpler method of encryption, symmetric means that there is a single key, and both sides utilize it. Traffic is packaged up using a key, and the same key is used to unwrap that traffic when it reaches its destination. Since this single key is the all-powerful Oz, you wouldn't want it to get into the wrong hands, which means you would not be presenting it out on the Internet. Therefore, symmetric encryption is not generally used for protecting internet website traffic.
- **Asymmetric encryption**: This is our focus with HTTPS traffic. Asymmetric encryption utilizes two keys: a public key and a private key. The public key is included inside your SSL

certificate, and so anyone on the internet can contact your website and get the public key. Your laptop then uses that public key to encrypt the traffic, and sends it over to the web server. Why is this secure if the public key is broadcast to the entire internet? Because the traffic can only be decrypted by using a corresponding private key, which is securely stored on your web server. It is very important to maintain security over your private key and your web servers, and ensure that key doesn't fall into anyone else's pocket.

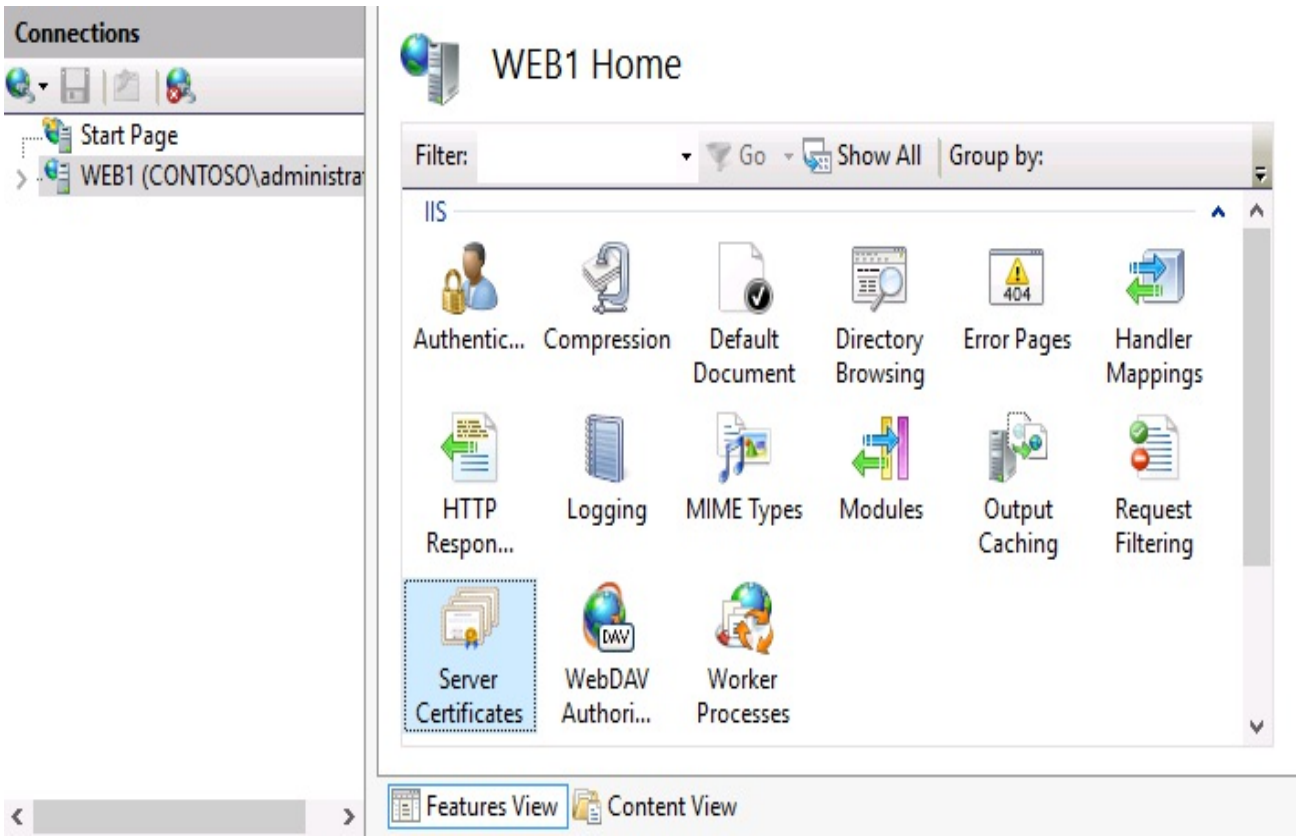
Creating a Certificate Signing Request

If your first step in acquiring an SSL certificate from your public CA entity was to log into their website, purchase a certificate, and immediately download it—you've already missed the boat. That certificate obviously would have no way of knowing about a private key that you might have sitting on your web server, and so that certificate would be effectively useless when installed anywhere.

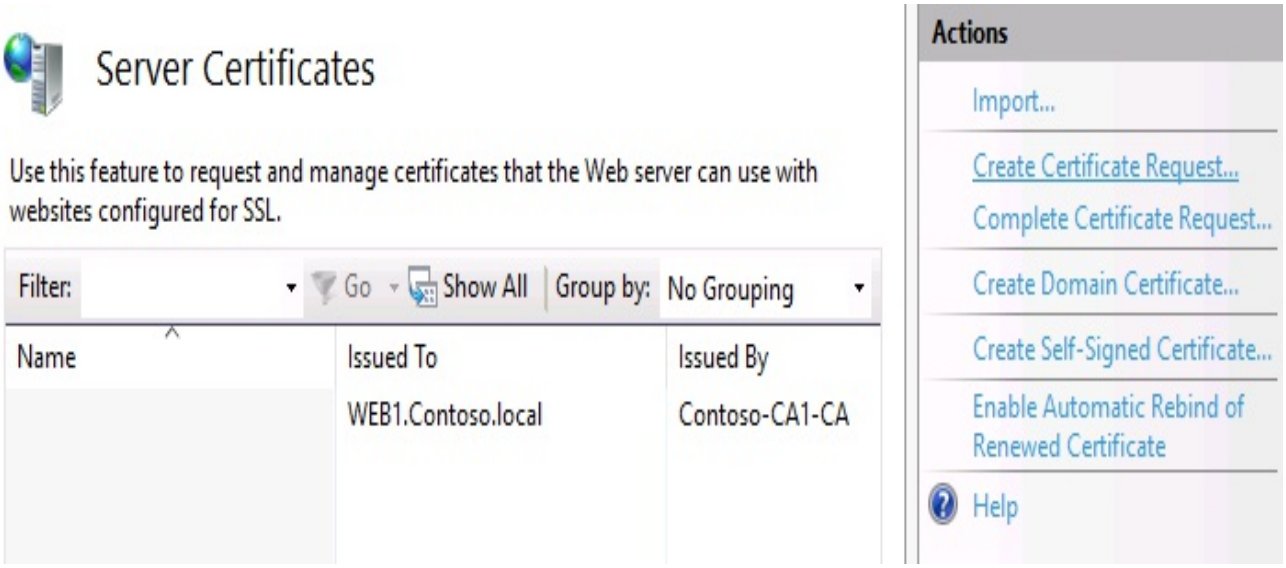
When you install an SSL certificate onto a web server, it is very important that the certificate knows about your private key. How do we make sure that happens? This is where the **Certificate Signing Request** (**CSR**) comes into play. The first step in correctly acquiring an SSL certificate is to generate a CSR. When you create that file, the web server platform creates the private key that is needed, and hides it away on your server. The CSR is then created in such a way that it knows exactly how to interact with that private key, and you then utilize the CSR when you log into the CA's website to request the certificate.

> *The private key is not inside the CSR, and your CA never knows what your private key is. This key is ultra important, and is only ever stored on your web server, inside your organization.*

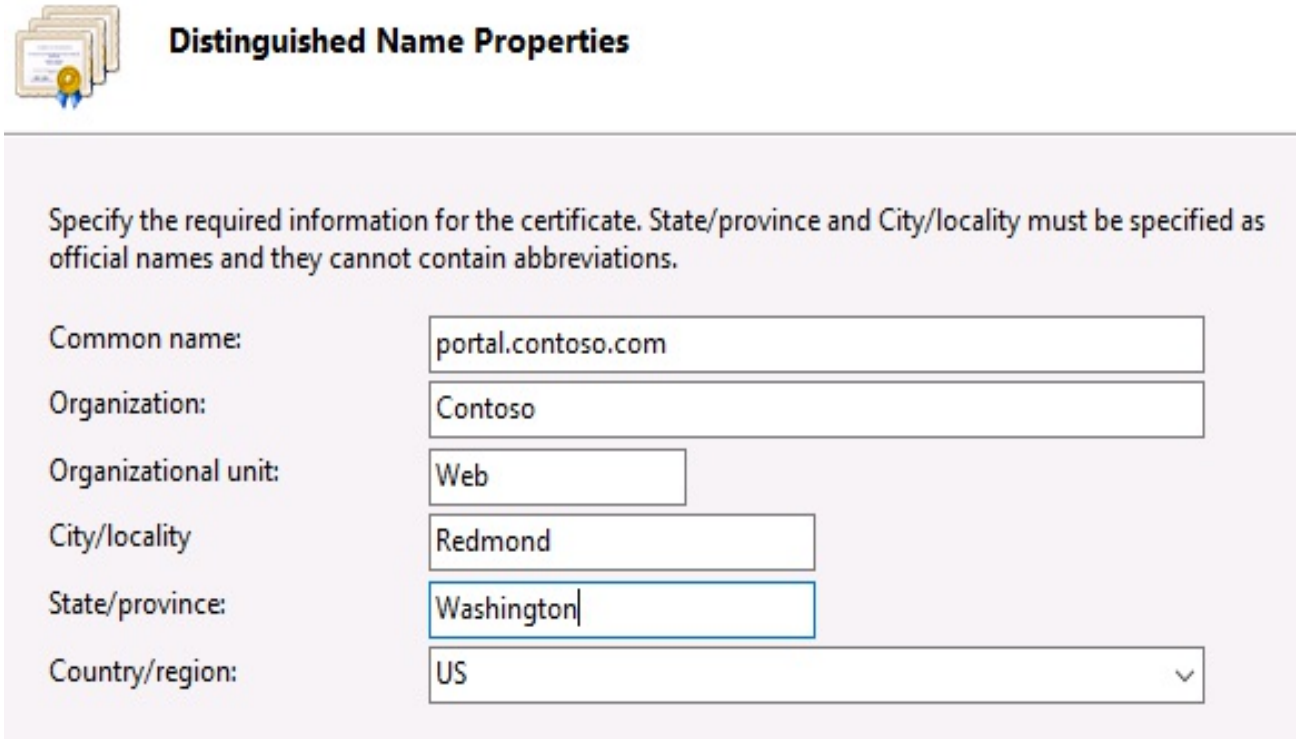
To generate a CSR, open up IIS from the Tools menu of Server Manager, and then click on the name of the web server from the navigational tree on the left side of your screen. This will populate a number of different applets into the center of the console. The one we want to work with is called Server Certificates. Go ahead and double-click on that:



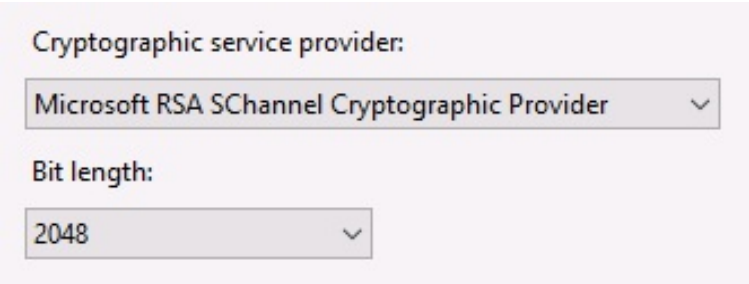
Now, inside the Server Certificates screen, you can see any existing certificates that reside on the server listed here. This is where we ultimately need to see our new SSL certificate, so that we can utilize it inside our website properties when we are ready to turn on HTTPS. The first step to acquiring our new certificate is creating the certificate request to be used with our CA, and, if you take a look on the right side of your screen, you will see an Actions section, under which is listed Create Certificate Request.... Go ahead and click on that Action:



In the resulting wizard, you need to populate the information that will be stored within your SSL certificate. The Common name field is the most important piece of information here, it needs to be the DNS name that this certificate is going to protect. So basically, you enter the name of your website here. Then continue with filling out the rest of your company-specific information. A couple of special notes here that often seem to trip up admins are that the Organizational unit can be anything at all; I usually just enter the word `Web`. Make sure to spell out the name of your State; do not use an abbreviation:



On the Cryptographic Service Provider Properties page, you typically want to leave the Cryptographic service provider set to its default, unless you have a specialized crypto card in your server and are planning to use it for handling encryption processing for this website. On an IIS server, you will almost always have Microsoft RSA SChannel Cryptographic Provider listed here. What you do want to change, however, is the Bit length. The standard bit length for many years was 1,024, and that continues to be the default choice in Windows Server 2019. The general industry for SSL encryption has decided that 1,024 is too weak, and the new standard is 2,048. When you head onto your CA's website to request a certificate, you will more than likely find that your request needs to have a minimum of 2,048 bits. So go ahead and change that drop-down setting to 2048:

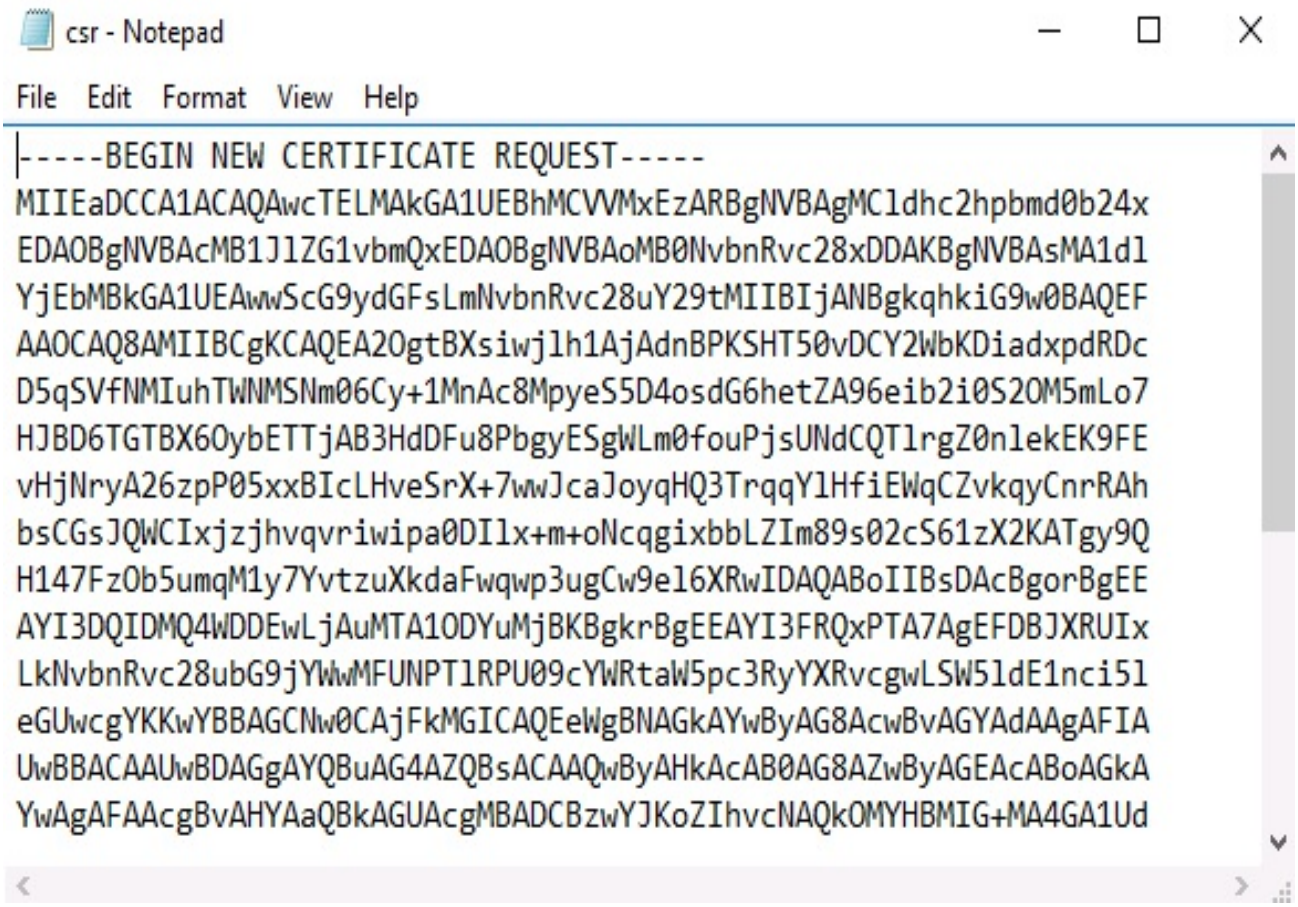


The only thing left to do for our CSR is to give it a location and filename. Saving this `csr` as a text file is the normal way to go, and serves our purposes well because all we need to do when we request our certificate is open the file and then copy and paste the contents. You have now created your `csr` file, and we can utilize this file to request the certificate from our public CA.

Submitting the certificate request

Now, head over to the website for your public certification authority. Again, any of the companies that we mentioned earlier, such as GoDaddy or Verisign, are appropriate for this purpose. Each authority has its own look and feel for their web interface, so I cannot give you exact steps that need to be taken for this process. Once you have an account and log into the authority's site, you should be able to find an option for purchasing an SSL certificate. Once that certificate has been purchased, there will be a process for requesting and deploying that certificate.

Once you enter the interface for building your new certificate, generally the only piece of information that the CA will ask you for is the contents of the `csr` file. If you open up the text file we saved earlier, you will see a big lump of nonsense:



This mess of data is exactly what the CA needs in order to create your new SSL certificate so that it knows how to interact with your web server's private key. Only the server that generated the CSR will be able to accept and properly utilize the SSL certificate that is based on this CSR. Typically, all you need to do is copy the entire content of the `csr` file and paste it into the CA's website.

Downloading and installing your certificate

Now you sit back and wait. Depending on which authority you are using and on how often your company purchases certificates from this authority, your certificate might be available for download almost immediately, or it could take a few hours before that certificate shows up in the available downloads list. The reason for this is that many of the CAs utilize human approval for new certificates, and you are literally waiting for someone to put eyes on the certificate request and your information to make sure you really work for the company and that you really own this domain name. Remember, the real benefit to a public SSL cert is that the CA is guaranteeing that the user of this certificate is the real deal, so they want to make sure they aren't issuing a certificate for `portal.contoso.com` to someone in the Fabrikam organization by mistake.

Once you are able to download the certificate from the CA website, go ahead and copy it over to the web server from which we generated the CSR. It is critically important that you install this new certificate onto the **same server**. If you were to install this new certificate onto a different web server, one that did not generate the CSR this certificate was built from, that certificate would import successfully, but would not be able to function. Once again, this is because the private key that the certificate is planning to interact with would not be present on a different server.

Back inside the IIS management console, we can now use the next Action in that list on the right, called Complete Certificate Request....

This launches a short little wizard in which you find the new certificate file that you just downloaded, and import it into our server. Now that the certificate resides on the server, it is ready to be used by your website.

There is one additional item that I always check after installing or importing an SSL certificate. You can now see your new certificate listed inside IIS, and if you double-click on your new certificate you will see the properties page for the cert. On the General tab of these properties, take a look near the bottom. Your certificate should display a little key icon and the You have a private key that corresponds to this certificate text. If you can see this message, your import was successful and the new certificate file matched up with the CSR perfectly. The server and certificate now share that critical private key information, and the SSL certificate will be able to work properly to protect our website. If you do not see this message, something went wrong in the process to request and download our certificate. If you do not see the message here, you need to start over by generating a new CSR, because the certificate file that you got back must not have been keyed appropriately against that CSR, or something along those lines. Without the *you have a private key* text at the bottom of this screen, your certificate will *not* validate traffic properly.

Here is an example of what it should look like when working correctly:



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer
- Proves your identity to a remote computer
- 2.16.840.1.114413.1.7.23.1

* Refer to the certification authority's statement for details.

Issued to:

Issued by: Go Daddy Secure Certificate Authority - G2

Valid from 6/26/2015 **to** 6/27/2016



You have a private key that corresponds to this certificate.

Exporting and importing certificates

I often find myself needing to use the same SSL certificate on multiple servers. This might happen in the case where I have more than one IIS server serving up the same website and I am using some form of load balancing to split the traffic between them. This need may also arise when working with any form of hardware load balancer, as you sometimes need to import certificates onto not only the web servers themselves, but into the load balancer box. Another example is when using wildcard certificates; when you purchase a wildcard, you typically intend to install it onto multiple servers.

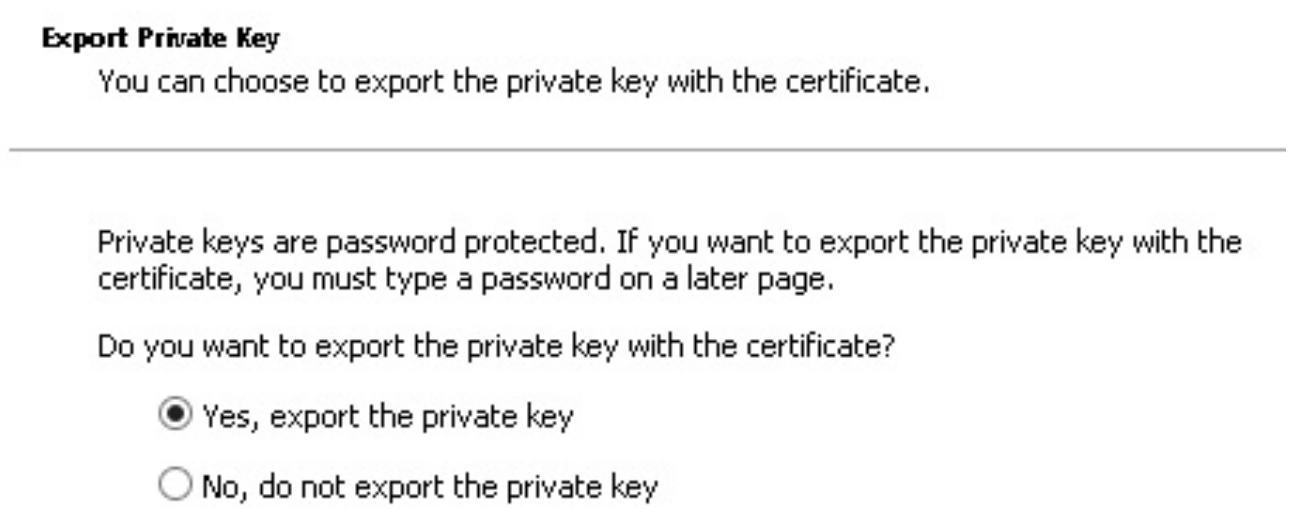
Does this mean that I need to generate a new CSR from each server, and request a new copy of the same certificate multiple times? Definitely not, and in fact doing so could cause you other problems: when a public CA **re-keys** a certificate—in other words, if you have already requested a certificate with a particular name and then come back again later to request another copy of the same certificate—that CA may invalidate the first one as it issues the second copy. This is not always immediately apparent, as there is usually a timer set on the invalidation of the first certificate. If you revisit the CA's web interface and request a new copy of the same certificate using a new CSR for your second web server, you might discover that everything works fine for a few days, but then suddenly the primary web server stops validating traffic because its SSL certificate has expired.

What should we do? When you need to reuse the same SSL certificate

on multiple servers, you can simply export it from one and import it on the next. There is no need to contact the CA at all. This process is quite straightforward, and there are two common places where you can do it: inside either the MMC snap-in for certificates, or from within IIS itself. It is important to note, though, that the process is slightly different depending on which avenue you take, and you have to be especially aware of what is happening with the private key as you step through these wizards.

Exporting from MMC

Head back into your Local Computer certificate store in the MMC, and navigate to Personal | Certificates so that you can see your SSL certificate listed. Right-click on the certificate, and then navigate to All Tasks | Export.... When you walk through this export wizard, the important part that I wanted to mention happens right away in the wizard steps. The first choice you have to make is whether to export the private key. Again, the private key is the secret sauce that allows the certificate to interact properly with the server on which it is installed. If you export without the private key, that certificate will not work on another server. So it is important here that, if you are exporting this certificate with the intention of installing it onto a second web server and using it for validating SSL traffic, you select the top option for Yes, export the private key:

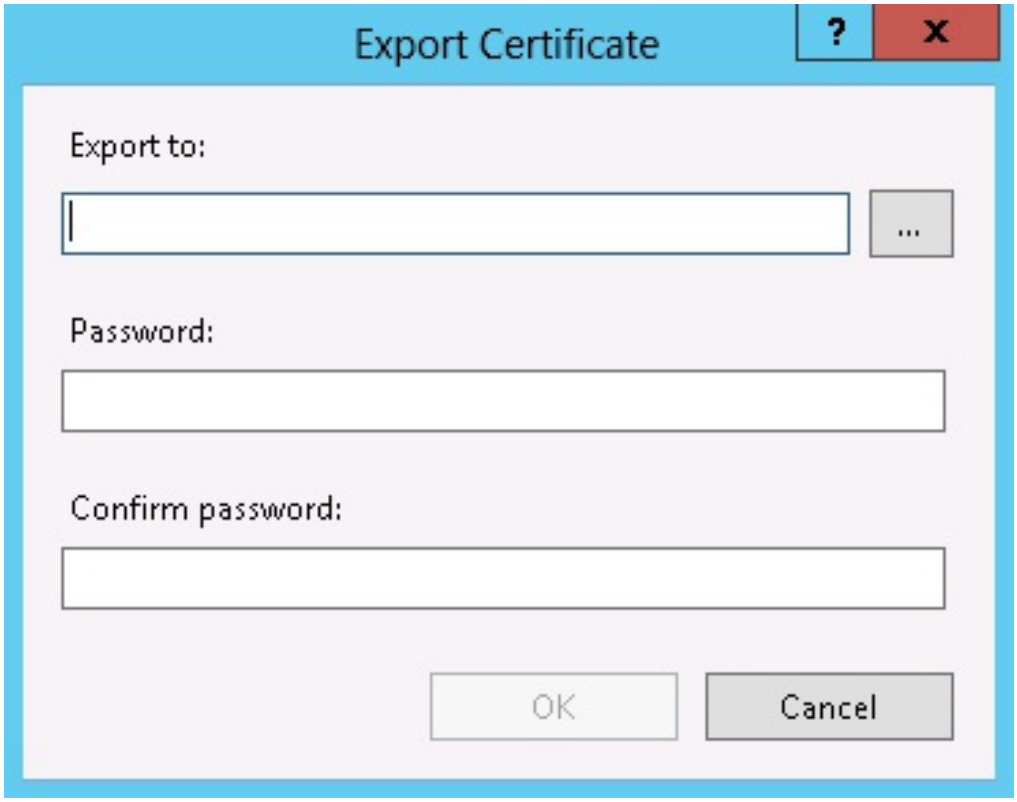


As the wizard sufficiently warns you, when you choose to export a

certificate that contains the private key information, you are required to supply a password, which will be used to protect the exported PFX file. It is important to choose a good password. If you forget it, your exported file will be completely useless. If you input a password that is very simple or is easy to guess, anyone who gets their hands on this PFX file may be able to use your certificate and private key on their own web servers, which would not be good.

Exporting from IIS

Inside the Server Certificates applet for IIS, just right-click on the certificate and choose Export.... This launches a single-page wizard that simply asks you for a location and password:



We had many more options that we could have chosen or denied when we exported using MMC, so why is this so short? IIS makes assumptions for the rest of the settings in order to speed up the export process. When you are exporting an SSL certificate, the chances are that you also intend to export the private key. Therefore, IIS simply makes that assumption and bypasses the rest of the choices. You are forced to

enter a password because you don't have a choice about the private key; it will be included with the certificate export automatically. So, if you had some reason to export a certificate that did *not* contain the private key info, you could not utilize the IIS console for this task. You would need to open up MMC and walk through the more extensive wizard found there.

Importing into a second server

Whichever direction you take for accomplishing the export, once you have the fully-fleshed PFX file available, importing into your second server is very easy. From within either console, MMC or IIS, you can right-click and choose the Import action. Walking through the steps, you simply choose the PFX file and then input the password that you used to protect the file. The certificate then imports, and, if you open the properties, you will see that the little key icon and the private key message are displayed properly at the bottom of the certificate properties screen. If you do not see the *you have a private key* message, you did something incorrectly during the export process and you'll need to try it again.

Go ahead and try it yourself; find a server with an SSL certificate and test exporting that cert with and without the private key. When you import into a new server, you will see that importing the certificate file without a private key does not contain this message at the bottom of the properties page, but the exported file that does contain the private key, results in the proper message here. To take it a step further, try utilizing both certificates on a non-important website and see what happens. You will find that the certificate that lacks the private key will fail to validate SSL traffic.

> *If you attempt to export an SSL certificate and the option to include the private key is grayed out, this means when the original administrator installed this certificate to the web server, they chose a special option that blocks the ability for the private key to be exported in the future. In this case, you will not be able to export the certificate with the private key.*

Summary

Certificates often get a bad rep, and I believe this is because people think they are a headache to deal with. I see their point. Without knowing how to navigate through the various administrative consoles that deal with your certificate infrastructure, it would be difficult to make even the simplest items function. By walking through the most common certificate-related tasks that any server admin will eventually have to tackle within their own networks, I hope that you have now found some comfort and confidence to progress with those projects that might be currently sitting on hold, waiting for the certificate infrastructure to be built. In the next chapter, we will study networking with Windows Server 2019.

Questions

1. What is the name of the role inside Windows Server 2019 that allows you to issue certificates from your server?
2. What kind of CA server is typically installed first in a domain environment?
3. Should you install the certification authority role onto a Domain Controller?
4. After creating a new certificate template, what next step needs to be taken before you can issue certificates to your computers or users from that new template?
5. What is the general name of the GPO setting that forces certificates to be issued without manual intervention by an administrator?
6. An SSL certificate will only be able to validate traffic properly if it shares _____ key information with the web server?
7. What is the primary piece of information that a public certification authority needs in order to issue you a new SSL certificate (hint: you generate this from your web server)?

Networking with Windows Server 2019

As we have been discussing so far in this book, servers are the tree trunks of our networks. They are the backbone infrastructure that enables us to get work done. If servers are the trunks, then the networks themselves must be the roots. Your network is the platform that supports the company infrastructure; it makes up the channels that any and all devices inside your company use to communicate with each other.

Traditionally, there have been *server professionals* and *network professionals* in the IT industry, and in many places that is still the case. An administrator who primarily works on servers does not generally have enough time in the day to also support the network infrastructure in an organization of any size, and the reverse is also true. Network administrators generally stick to their own equipment and management tools, and wouldn't be interested in diving too deeply into the Windows Server world. However, many of us work in smaller companies where many hats must be worn. Some days, both the server admin and the network admin hats sit on top of each other, and so we must understand at least a baseline of networking and tools that we can use to troubleshoot connections that are not working. In addition, Windows Server 2019 brings a new networking mindset into focus: the virtualization of your networks. There will always be some semblance of a physical network, using physical switches and routers to move the packets around between different rooms and buildings. But now we are also incorporating the idea of **software-defined networking** (**SDN**)

into our Windows Servers, which gives us the capability to virtualize some of that configuration. Not only the config itself, we are actually virtualizing the network traffic and building our networks from within a server console, rather than using command-line interfaces to configure our routers, which was always needed in the past.

Hold the phone; I am getting ahead of myself. First, let's talk about some of the new and useful things inside Windows Server 2019 that do involve working with physical networks, or any networks, because these are going to be important for any administrator in today's networking world. Later, we will take a few moments to further explore this new idea of network virtualization.

The following are the topics we plan to discuss in this chapter:

- Introduction to IPv6
- Your networking toolbox
- Building a routing table
- NIC Teaming
- Software-defined networking

Introduction to IPv6

Welcome to the dark side! Unfortunately, that is how many people think of IPv6 at the moment. While IPv6 is by no means a new thing, in my experience it is still something that almost no one has deployed in their own networks. While working with hundreds of different companies all over the world over the past few years, I have come across only one organization that was running IPv6 over their entire production network, and it wasn't even true native IPv6. Instead, they were using a tunneling technology, called ISATAP, over their whole network in order to make all of the servers and clients talk to each other using IPv6 packets, but these packets were still traversing an IPv4 physical network. Don't get me wrong; I have found plenty of cases where companies are toying around with IPv6 and have some semblance of it configured on a sectioned-off piece of their networks, but using it for the entire production network? Most of us just aren't ready for that big a change yet. Why is it so difficult to put IPv6 into place? Because we have been using IPv4 since basically the beginning of time, it's what we all know and understand, and there really isn't a great need to move to IPv6 inside our networks. Wait a minute; I thought there was some big scare about running out of IPv4 addresses? Yes, that is true for IP addresses on the public internet, but it has nothing to do with our internal networks. You see, even if we run out of public IPv4 addresses tomorrow, our internal networks at our companies are not going to be impacted. We can continue to run IPv4 inside the network for a long time to come, possibly forever and always, as long as we are comfortable using NAT technologies to translate the traffic down into IPv4 as it comes into us from the internet. We have all been using NAT in one form or another for almost as long as IPv4 has existed, so it is

obviously something people are very comfortable with.

Let me be clear: I am not trying to convince you that sticking with IPv4 is the way of the future. I am just laying out the fact that, for most organizations over the next few years, this will simply be the truth. The reason I want to discuss IPv6 here is that, eventually, you will have to deal with it. And once you do, you'll actually get excited about it! There are some huge advantages that IPv6 has over IPv4, namely, the enormous number of IP addresses that you can contain within a single network. Network teams in companies around the World struggle every day with the need to build more and more IPv4 networks and tie them together. Think about it: there are many companies now with employee counts in excess of 10,000. Some have many, many times that number. In today's world, everyone needs almost constant access to their data. Data is the new currency. Most users now have at least two physical devices they utilize for work, sometimes more than that: a laptop and a tablet, or a laptop and a smart phone, or a desktop and a laptop and a tablet and a smart phone; you get the idea. In the IPv4 world, where you are dealing with comparatively small IP address ranges, you have to get very creative with creating subnets in order to accommodate all of these physical devices, which each need a unique IP address in order to communicate on the network. The biggest advantage to IPv6 is that it resolves all of these problems immediately, and by default, by providing the capability to have a huge number of IP addresses within a single network. How many more addresses are we talking about? The following is some comparison data to give you a little perspective:

- An IPv4 address is a 32-bit length address that looks like this:

```
192.168.1.5
```

- An IPv6 address is a 128-bit length address that looks like this.

```
2001:AABB:CCDD:AB00:0123:4567:8901:ABCD
```

As you can see, the IPv4 address is much shorter, which obviously means there are fewer possibilities for unique IP addresses. What you don't see is how much longer IPv6 addresses really are. These examples portray IPv4 and IPv6 addresses as we are used to seeing them, in their finished forms. Really though, the IPv4 address is shown in decimal form, and the IPv6 in hexadecimal. IPv6 addresses are shown and used via hex so that the addresses are compressed as much as possible. In reality, if you dig around under the hood, an IPv6 address in its native 128-bit form might look something like this (and indeed, this is how it looks inside the actual packet):

```
00010000010000010000110110011000000000000000000010000000000000000
00010000000000000000000000000010000000000000000000000000000000001
```

That's an impressive set of digits, but not something that is very usable or friendly to the human eye. So rather than show the bits, what about an IPv6 address shown in decimal format, in the same way that IPv4 addresses have always been shown? In that case, an IPv6 address might look something like this:

```
192.16.1.2.34.0.0.1.0.0.0.0.0.0.0.1
```

Now we fully understand why IPv6 is always used and shown in hexadecimal; the addresses are long even in that compressed format!

Understanding IPv6 IP addresses

When we set up IPv4 networks, subnetting is extremely important because it is what enables us to have more than one collection of IP addresses within the same network. In the most basic form of networking, where you set up some IP addresses and run a `/24` subnet (a subnet mask of `255.255.255.0`)—which is very common on small networks such as inside a house or small business office—you are limited to 254 unique IP addresses. Ouch! Some companies have thousands of different servers, without accounting for all of their client computers and devices that also need to connect to the network. Thankfully, we can build out many different subnets within a single IPv4 network in order to increase our usable IP address scope, but this takes careful planning and calculation of those subnets and address spaces, and is the reason that we rely on experienced network administrators to manage this part of the network for us. One invalid subnet configuration in a routing table can tank the network traffic flow. The administration of subnets in a large IPv4 network is not for the faint of heart.

When we are talking about IPv6 addressing, the sky almost seems to be the limit. If you were to calculate all of the unique IPv6 addresses available in the preceding 128-bit space, you would find that there are more than 340 undecillion addresses available to create. In other words, 340 trillion, trillion, trillion addresses. This is the number being touted out there about how many available addresses there are on the IPv6 internet, but what does that mean for our internal networks?

To discuss the number of addresses we could have inside a typical internal network that runs IPv6, let's first step back and look at the

address itself. The address I showed earlier is just something I made up, but we will break down the parts of it here:

```
2001:AABB:CCDD:AB00:0123:4567:8901:ABCD
```

Compared to `192.168.1.5`, this thing looks like a monstrosity. That is because we are generally not used to dealing with the hexadecimal format; it is just a different way of looking at data. As we mentioned, this is a 128-bit address. It is broken up into 8 different sections, each section separated by a colon is made up of 16 bits. The first 64 bits (the first half) of the address are routing information, and the latter 64 bits are the unique device ID on the network. Within the first part, we have two different components. The first 48 bits (3 groups of hex) are an organizational prefix that will be the same on all of our devices in the network. Then the fourth set of information, the next 16 bits, can be our subnet ID. This gives us the flexibility of still having many different subnets if we desire in the future, by using multiple numbers here as the subnet ID. After dedicating the first half of the address, we now have the latter half to work with, the last 64 bits. These we can leave for device IDs. This part of the address will be different for every device on the network, and will define the individual static IPv6 addresses that will be used for communication. Let's lay it all out. We will take the example address from previously, and break it out into the following parts:

- **Organizational prefix**: `2001:AABB:CCDD:AB00:0123:4567:8901:ABCD` where `2001:AABB:CCDD` is the organizational prefix
- **Subnet ID**: `2001:AABB:CCDD:AB00:0123:4567:8901:ABCD` where `AB00` is the subnet ID
- **Device**

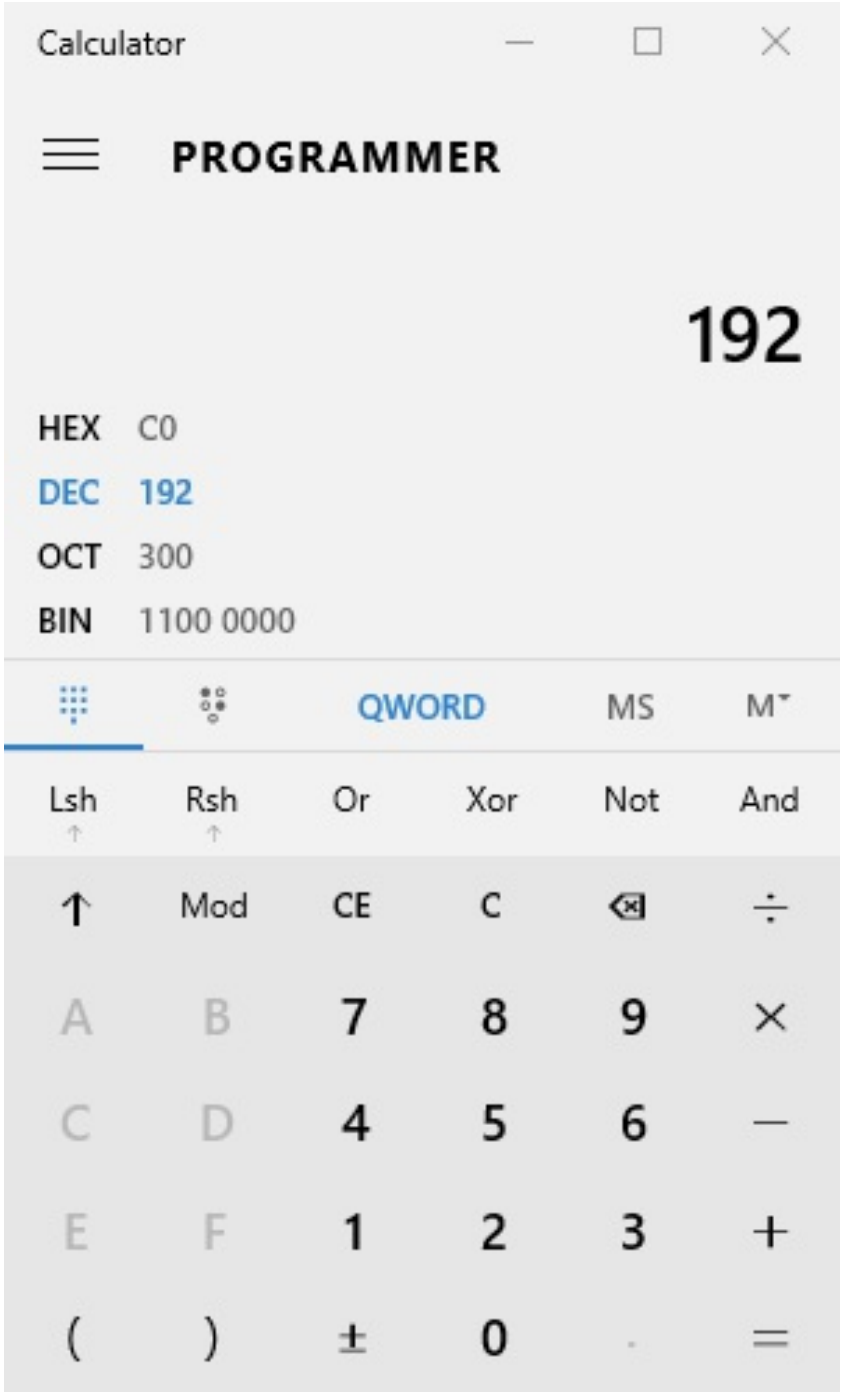
IDs: `2001:AABB:CCDD:AB00:0123:4567:8901:ABCD` where `0123:4567:8901:ABCD` is a unique device ID

How many devices can we have in our network with an IP schema such as this? Well, even in our example, where we only allocated one 16-bit section for subnetting, and 64 bits for actual IP addresses, that would provide us with the capability to have more than 65,000 subnets and quintillions of unique device IDs in our IP range. Impressive, isn't it?

If we stick with this and use just a single subnet to contain all of our machines, the first half of our addresses will always be the same, making these long addresses much easier to remember and deal with. It is surprising how quickly you will get used to seeing these large hex numbers in your environment, but even though you will start to recognize them, you still probably are not going to quickly jump into servers or computers in your network anymore by using the static IP addresses. I know a lot of us are still in the habit of saying: I need to quickly jump into my web server, I'll just RDP into `192.168.1.5`. Just the time that it takes to type out these IPv6 addresses, even if you do remember them, isn't generally worth it. IPv6 will bring with it a larger reliance on DHCP and DNS to make it more usable.

Now that we understand what sections of the address are going to be used for what purposes, how do we go about assigning the individual device ID numbers for all of the computers, servers, and other devices on our network? You could start with number 1 and go up from there. Another idea is to calculate out the old IPv4 addresses into hex and use this as the last 32 bits of the address—open up Windows Calculator on your computer, drop down the menu, and change it into Programmer mode. This is a quick and easy tool that you can use to convert decimal into hexadecimal, and vice versa. Let's take the example of my web server that is running on `192.168.1.5`. I want to implement IPv6 inside my

network, and I want my server's IPv6 addresses to reflect the original IPv4 address in the device ID section of the new address. In my calculator, if I type in `192` and then click on HEX, it will show me the corresponding hexadecimal to the decimal of 192, as shown in the following screenshot:



If we do that with each of the octets in our IPv4 address, we will see the following:

```
192 = C0
168 = A8
1 = 01
5 = 05
```

So my `192.168.1.5` factors out to `C0A8:0105`. I can now utilize that in combination with my organizational prefix and my subnet ID to create a static IPv6 address for my web server:

```
2001:AABB:CCDD:0001:0000:0000:C0A8:0105
```

You'll notice in the preceding IPv6 address that I input the hex for the device ID at the end, but I also made a couple of other changes. Since we are leaving the last 64 bits available for the device ID, but my old IPv4 address only consumes 32 bits, I am left with the 32 bits in the middle. It would be kind of weird to have data in there that didn't mean anything to us, so we will simply make it all zeros to simplify my addressing scheme. In addition to that change, I also adjusted my subnet ID to the number 1, since this is the first subnet in my network.

Our new addressing is starting to look a little cleaner, and makes more sense. Now that we see this new address for our web server laid out, I can see that there are some additional clean-up tasks we can perform on the address in order to make it look even better. Right now the address listed earlier is 100% accurate. I could plug this IP address into the NIC properties of my web server, and it would work. However, there are a whole lot of zeros in my address, and it isn't necessary that I keep them all. Any time you have unnecessary zeros within a 16-bit segment that are preceding the actual number, they can simply be removed. For example, our subnet ID and the first 32 bits of our device ID have a lot of unnecessary zeros, so I can consolidate the address down to the following:

```
```

```
2001:AABB:CCDD:1:0:0:C0A8:0105
```

Then, to take it even a step further, any time you have full 16-bit sections that are composed entirely of zeros, they can be fully consolidated into a double colon. So, the first 32 bits of my device ID that are all zeros, I can replace with `::`. The following is the full address, and the consolidated address. These numbers look quite different. My consolidated address is much easier on the eyes, but from a technological perspective, they are exactly the same number:

```
2001:AABB:CCDD:0001:0000:0000:C0A8:0105
2001:AABB:CCDD:1::C0A8:0105
```

In fact, if you are setting up a lab or want to quickly test IPv6, you could use addresses as simple as the following example. The two addresses that I will show you here are exactly the same:

```
2001:0000:0000:0000:0000:0000:0000:0001
2001::1
```

> *It is important to note that you can only use a double-colon* **once** *within an IP address. If you had two places where it could be applicable within the same address, you can only implement it in one of those places, and you will have to spell out the zeros in the other place.*

With the information provided here, you should be able to put together your own semblance of IPv6 and start issuing some IPv6 addresses to computers or servers in your network. There is so much more that could be learned on this subject that a full book could be written, and indeed many have.

Your networking toolbox

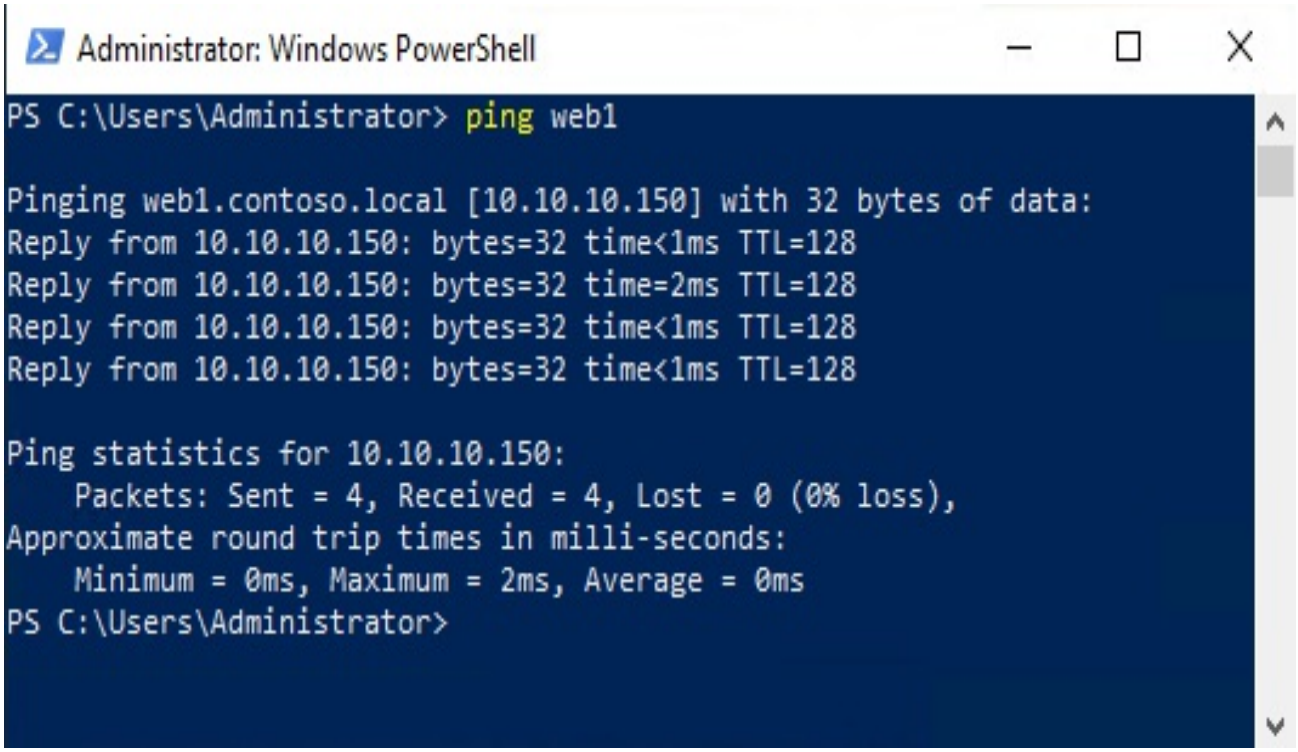
Whether you are a server administrator, a network administrator, or a combination of the two, there are a number of tools that are useful for testing and monitoring network connections within the Windows Server world. Some of these tools are baked right into the operating system and can be used from the Command Prompt or PowerShell, and others are more expansive graphical interfaces that require installation before running. The following are the network tools that we are going to look at:

- `ping`
- `tracert`
- `pathping`
- `Test-Connection`
- `telnet`
- `Test-NetConnection`

All the these tools are free, though, so you have no excuse to delay getting acquainted with these helpful utilities.

ping

Even the newest of IT pros are usually familiar with this one. `ping` is a command that you can utilize from a Command Prompt or PowerShell, and it is simply used to query a DNS name and/or IP address to find out whether it responds. Ping is and has always been our go-to tool for testing network connectivity between two devices on a network. From my Windows 10 client on the LAN, I can open a prompt and `ping <IP_ADDRESS>`. Alternatively, because I am using DNS in my environment, which will resolve names to IP addresses, I can also use `ping <SERVERNAME>`, as shown in the following example. You can see that my server replies to my ping, letting me know that it is online and responding:



Ping traffic is technically called **ICMP traffic**. This is important because ICMP is blocked by default more and more often these days, with firewalls being turned on by default on so many of our systems and devices. Historically, ping was always a tool that we could count on to tell us fairly surely whether connectivity was flowing between two devices, but that is not the case anymore. If you build a brand new Windows box and plug it into your network, that computer may be communicating with the internet and all of the servers on your network just fine, but if you try to ping that new computer from another machine on your network, the ping will probably fail. Why would that happen? Because Windows has some security measures built into it by default, including the blocking of ICMP traffic in the Windows Firewall. In that case, you would need to either turn off the firewall, or provide it with an access rule that allows ICMP traffic. Once such a rule is enabled, pings will start replying from this new computer. Keep in mind whenever building new systems or servers in your network that ping is not always the most trustworthy tool to depend upon in today's networking world.

It's easy to allow ICMP responses by plugging a rule into the Windows Defender Firewall with Advanced Security, though you still wouldn't want to have to remember to do this manually on every new system you introduce into a network. Thankfully, you already know how to utilize Group Policy in order to build a GPO and push it out to all machines on your network, and yes you can absolutely place firewall rules inside that GPO. This is a common way to allow or block ICMP throughout an entire organization, by issuing a firewall rule via Group Policy.

tracert

`tracert`, which is pronounced **Trace Route**, is a tool used to trace a network packet as it traverses your network. What it really does is watch all of the places the packet bumps into before hitting its destination. These bumps in the road that a network packet needs to get through are called **hops**. Trace route shows you all of the hops that your traffic is taking as it moves toward the destination server or whatever it is trying to contact. My test lab network is very flat and boring, so doing a `tracert` there wouldn't show us much of anything. However, if I open up a PowerShell prompt from an internet-connected machine and `tracert` to a web service, such as Bing, we get some interesting results:

```
PS C:\WINDOWS\system32> tracert www.bing.com

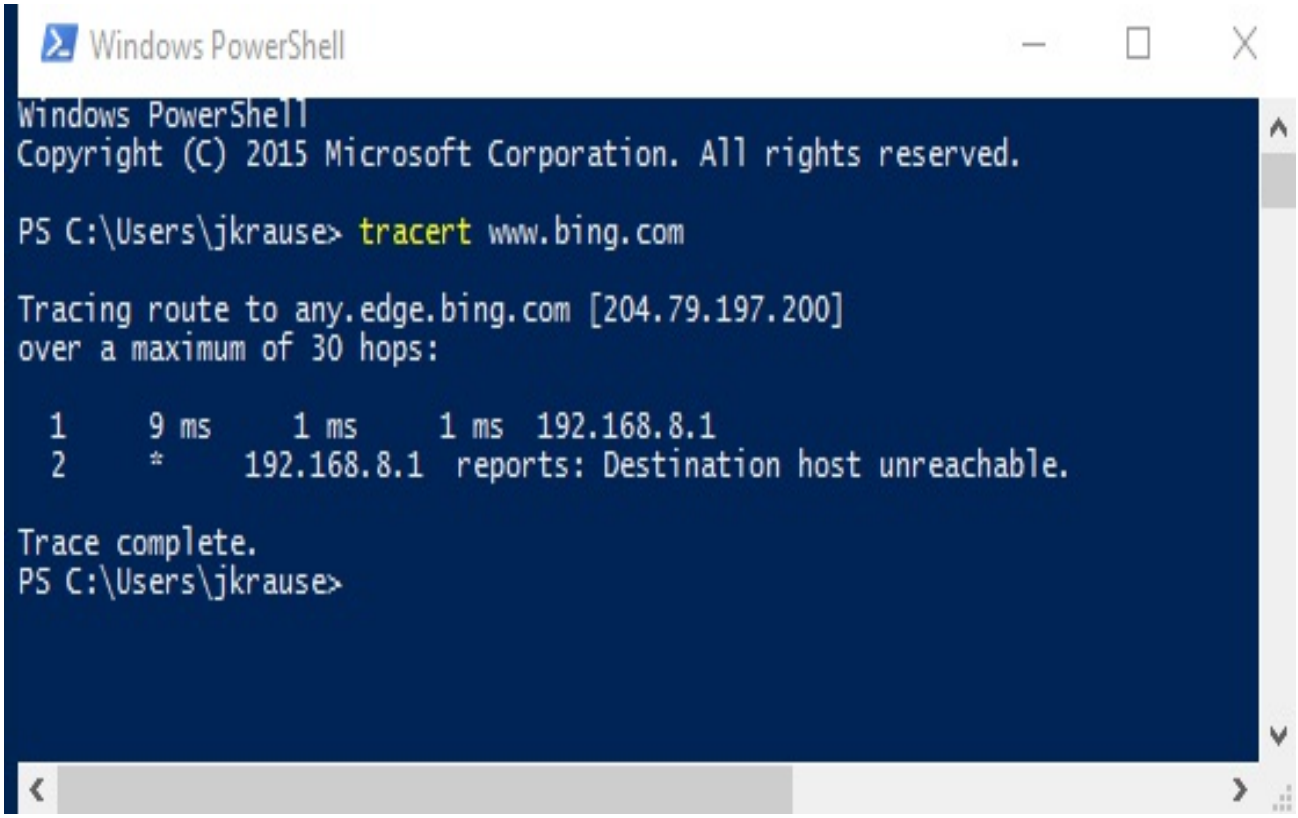
Tracing route to any.edge.bing.com [204.79.197.200]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.8.1
  2     1 ms    <1 ms    <1 ms  192.168.128.1
  3     8 ms     7 ms     5 ms  172.17.224.1
  4    11 ms     9 ms    15 ms  172.19.253.1
  5    10 ms     9 ms    11 ms  172.31.255.1
  6    20 ms     9 ms    13 ms  htl-max1-1.iserv.net [206.114.55.1]
  7    15 ms    12 ms     8 ms  69.87.144.9
  8    23 ms    18 ms    19 ms  888-2.iserv.net [206.114.40.2]
  9    23 ms    20 ms    15 ms  g5-0-0.core3.grr.iserv.net [206.114.51.20]
 10    19 ms    11 ms    19 ms  g5-0-0.core1.grr.iserv.net [206.114.51.2]
 11    21 ms    28 ms    19 ms  GigabitEthernet4-1.GW5.DET5.ALTER.NET [152.179.10.81]
 12    25 ms    28 ms    28 ms  0.ae1.XL3.CHI13.ALTER.NET [140.222.225.179]
 13    27 ms    37 ms    54 ms  TenGigE0-6-0-1.GW2.CHI13.ALTER.NET [152.63.65.133]
 14    36 ms    34 ms    34 ms  microsoft-gw.customer.alter.net [152.179.105.130]
 15    58 ms    50 ms    46 ms  104.44.81.58
 16    34 ms    33 ms    36 ms  10.201.194.219
 17    26 ms    29 ms    29 ms  a-0001.a-msedge.net [204.79.197.200]

Trace complete.
PS C:\WINDOWS\system32>
```

If you utilize `tracert` but are not interested in seeing all of the DNS information provided in the output, use `tracert -d` to focus only on the IP addresses.

This information can be extremely useful when trying to diagnose a connection that is not working. If your traffic is moving through multiple hops, such as routers and firewalls, before it gets to the destination, `tracert` can be essential in figuring out where in the connection stream things are going wrong. Given that the preceding screenshot shows a successful trace route to Bing, now let's see what that it looks like when things are broken. I will unplug my internet router and run the same `tracert www.bing.com` again, and now we can see that I am still able to communicate with my local router, but not beyond:


Windows PowerShell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.
PS C:\Users\jkrause> tracert www.bing.com
Tracing route to any.edge.bing.com [204.79.197.200]
over a maximum of 30 hops:
1 9 ms 1 ms 1 ms 192.168.8.1
2 * 192.168.8.1 reports: Destination host unreachable.
Trace complete.
PS C:\Users\jkrause>

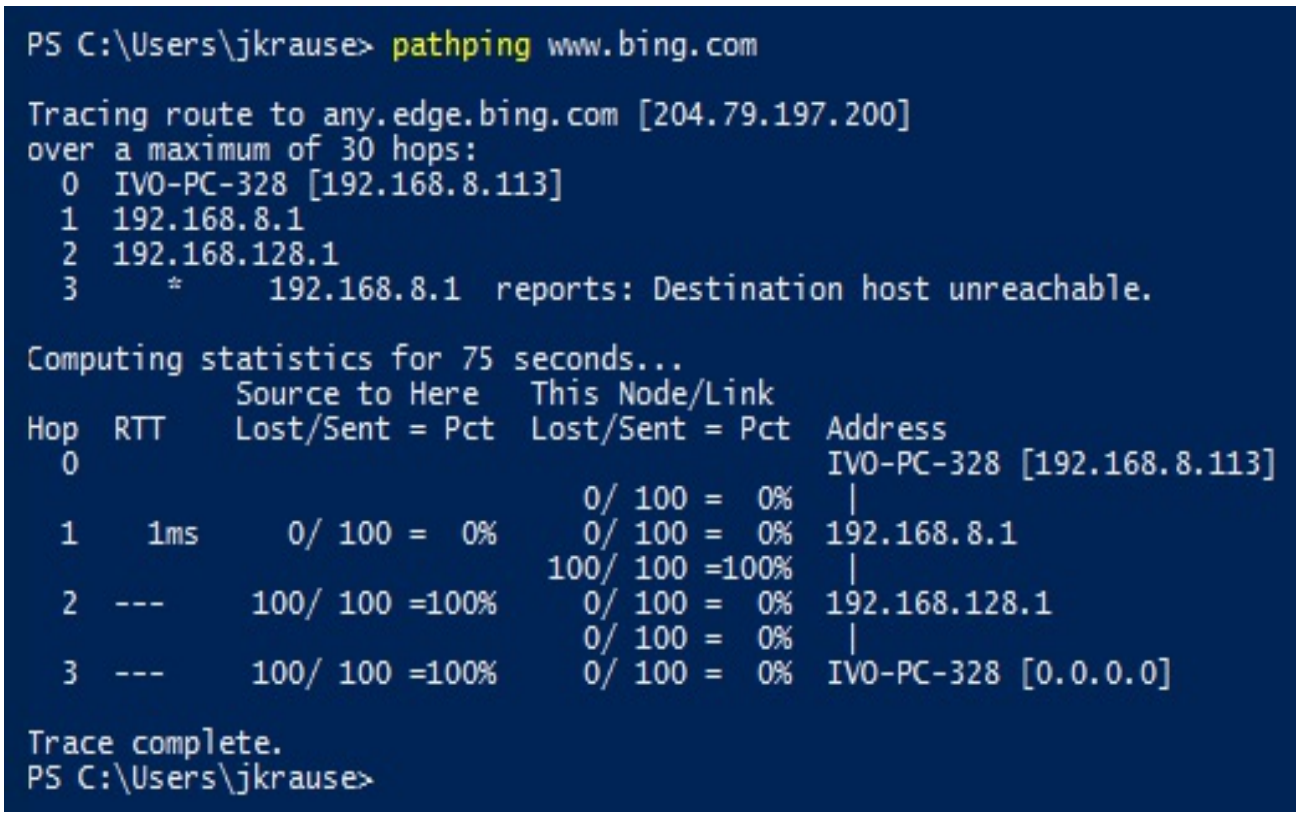
pathping

`tracert` is useful and seems to be the *de facto* standard for tracing packets around your network, but this next command is even more powerful in my opinion. `pathping` essentially does exactly the same thing as `tracert`, except that it provides one additional piece of crucial information. Most of the time, with either of these commands, you are only interested in figuring out where in the chain of hops something is broken, but often when I'm setting up servers for networking purposes, I am working with servers and hardware that have many different network cards. When dealing with multiple NICs in a system, the local routing table is just as important as the external routers and switches, and I often want to check out the path of a network packet in order to see which local NIC it is flowing out of. This is where `pathping` becomes more powerful than `tracert`. The first piece of information that `tracert` shows you is the first hop away from the local server that you are traversing. But `pathping` also shows you which local network interface your packets are flowing out of.

Let me give you an example: I often set up remote access servers with multiple NICs, and during this process we create many routes on the local server so that it knows what traffic needs to be sent in which direction, such as what traffic needs to go out the Internal NIC, and what traffic needs to go out the External NIC. After completing all of our routing statements for the Internal NIC, we test them out by pinging a server inside the network. Perhaps that ping fails, and we aren't sure why. I can try a `tracert` command, but it's not going to provide me with anything helpful because it simply cannot see the first hop, it just times out. However, if I try a `pathping` instead, the first hop will still time out,

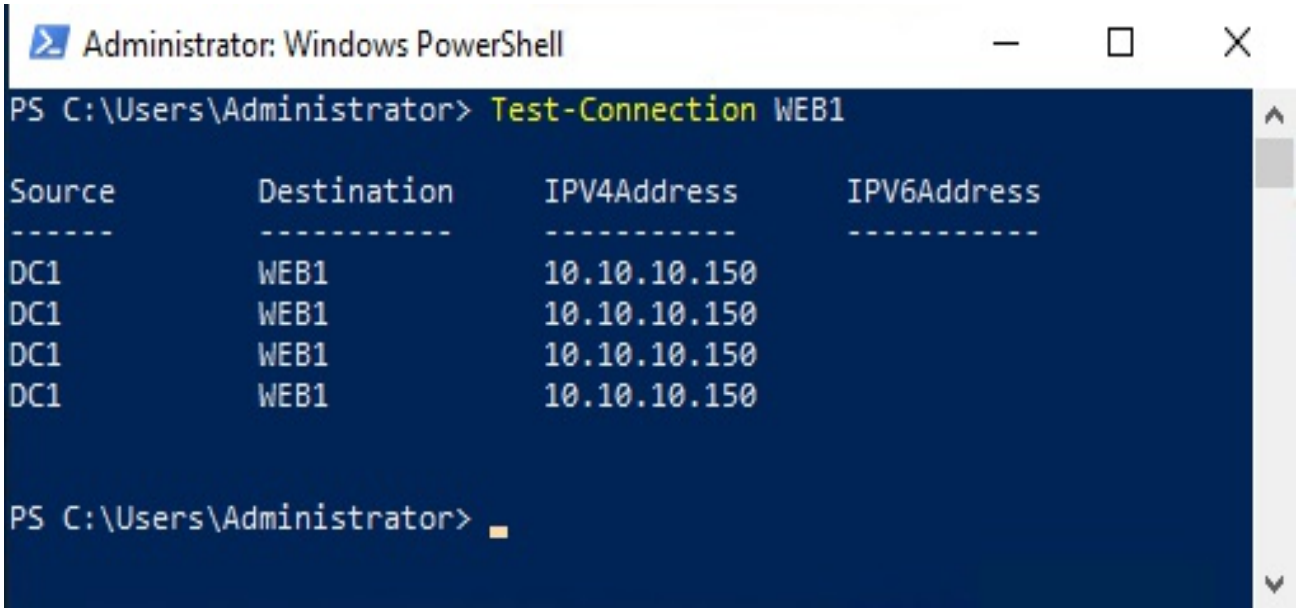
but I can now see that my traffic is attempting to flow out of my *EXTERNAL NIC*. *W*hoops! We must have set something up incorrectly with our static route on this server. So then I know that I need to delete that route and recreate it in order to make this traffic flow through the internal NIC instead.

The following is the same PowerShell prompt from the same computer that I used in my `tracert` screenshot. You can see that a `pathping` shows me the local IP address on my laptop where the traffic is attempting to leave the system, whereas the `tracert` command did not show this information:



Test-Connection

The commands we have discussed so far can be run from either the Command Prompt or PowerShell, but now it's time to dive into a newer one that can only be run from the PowerShell prompt: a cmdlet called `Test-Connection`; it is sort of like `ping` on steroids. If we open up a PowerShell prompt in the lab and run `Test-Connection WEB1`, we get very an output that is very similar to what we'd get with a regular `ping`, but the information is laid out in a way that I think is a little easier on the eyes. There is also an unexpected column of data here called `Source`:



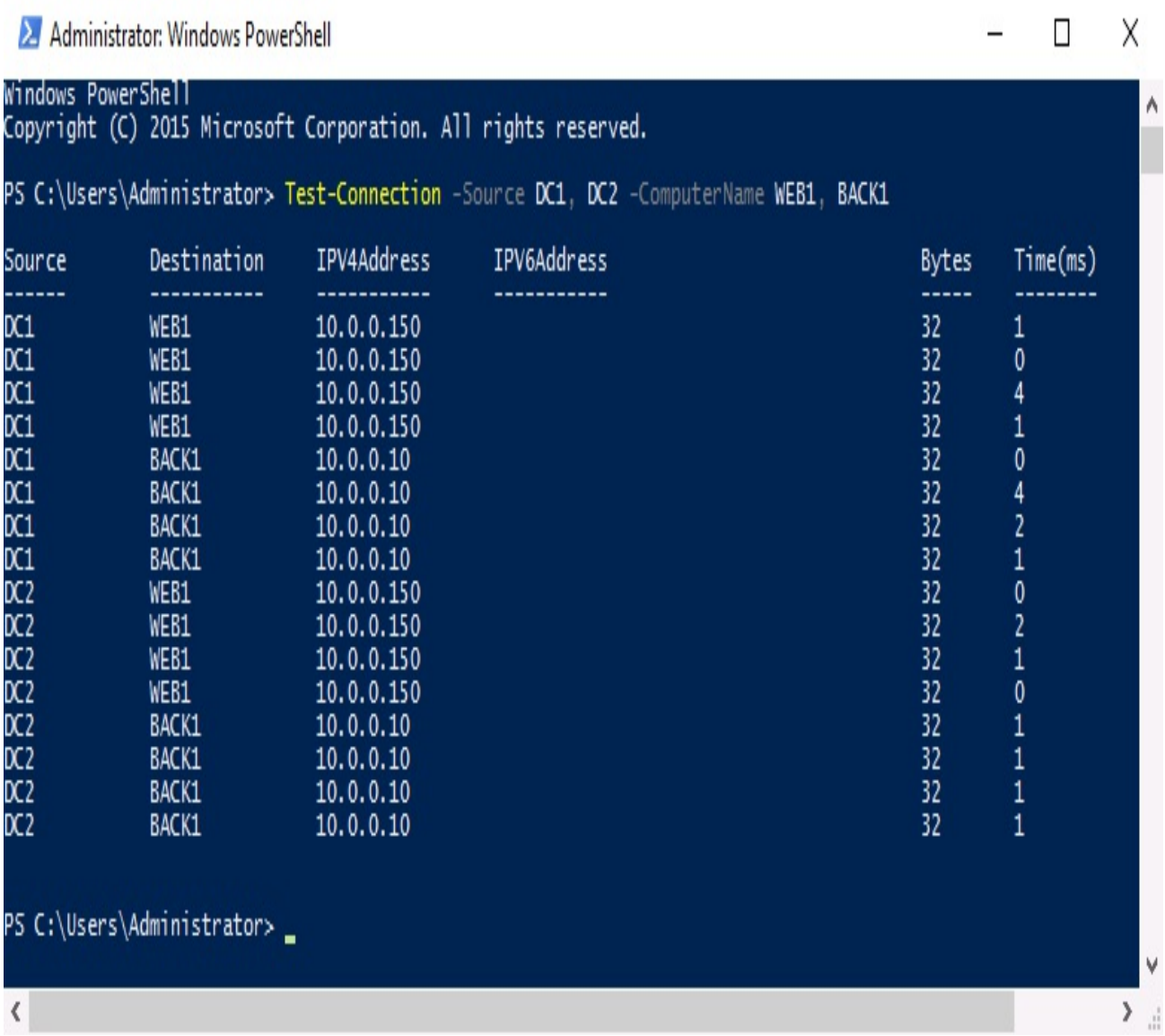
That is interesting. I was logged into my DC1 server when I ran this command so my source computer for this command was DC1. But does this mean that I have the ability to manipulate the source computer for the `Test-Connection` cmdlet? Yes, this is exactly what it means. As with everything in Windows Server 2019 management, the need to be logged

into a local server is decoupled. Specific to the `Test-Connection` cmdlet, this means you have the ability to open a PowerShell prompt anywhere on your network, and to test connections between two different endpoints, even if you are not logged into either of them. Let's test that out.

I am still logged into my DC1 server, but I am going to use a `Test-Connection` cmdlet in order to test connections between a number of my servers in the network. You see, not only can you specify a different source computer than the one you are currently logged into, you can take it a step further and specify multiple sources and destinations with this powerful cmdlet. So if I want to test connections from a couple of different source machines to a couple of different destinations, that is easily done with the following command:

```
Test-Connection -Source DC1, DC2 -ComputerName WEB1, BACK1
```

You can see in the following screenshot that I have ping statistics from both `DC1` and `DC2`, to each of the `WEB1` and `BACK1` servers in my network. `Test-Connection` has the potential to be a very powerful monitoring tool:

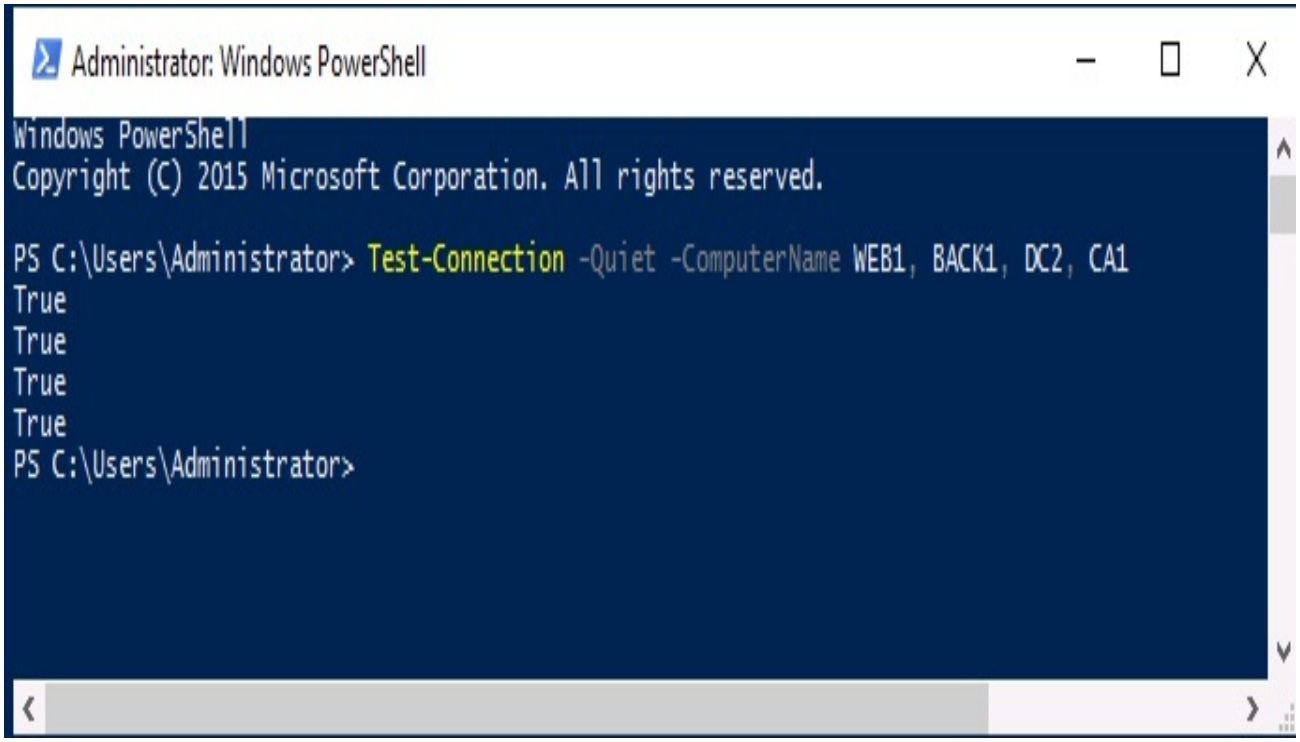


One more useful function to point out is that you can clean up the output of the command pretty easily by using the `-Quiet` switch. By adding `-Quiet` to a `Test-Connection` command, it sanitizes the output and only shows you a simple `True` or `False` for whether the connection was successful, instead of showing you each individual ICMP packet that was sent. Unfortunately, you cannot combine both the `-Source` switch and the `-Quiet` switch, but if you are using `Test-Connection` from the original source computer that you are logged into—like most of us will be doing

anyway—`-Quiet` works great. Most of the time, all we really care about is Yes or No as to whether these connections are working, and don't necessarily want to see all four attempts. By using `-Quiet` we get exactly that:

```
Test-Connection -Quiet -ComputerName WEB1, BACK1, DC2, CA1
```

If I were to use `Test-Connection` in the standard way to try to contact all of the servers in my network, that would turn into a whole lot of output. But by using the `-Quiet` switch, I get back a simple `True` or `False` on whether each individual server could be contacted:



telnet

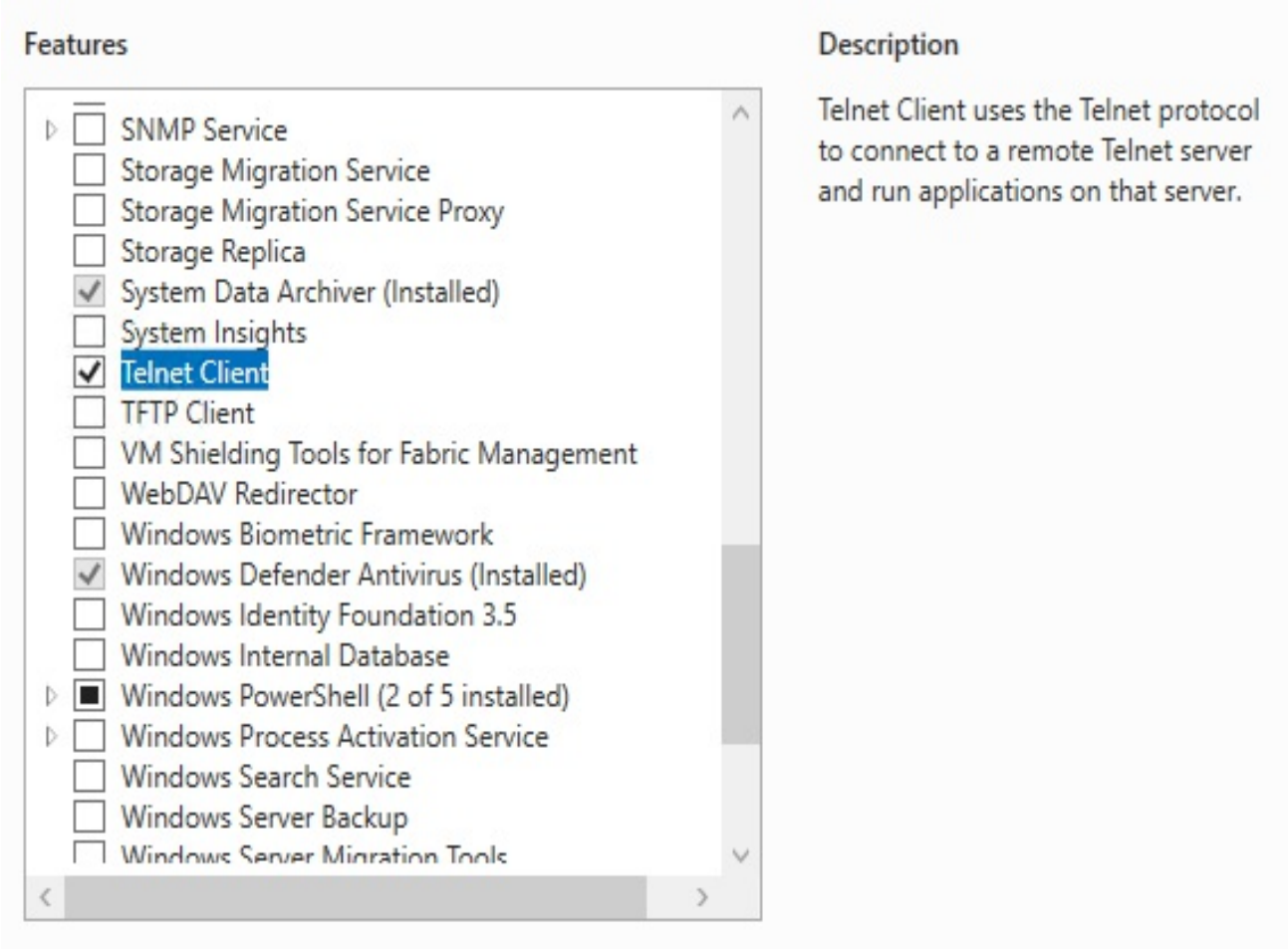
`telnet` provides quite a bit of remote management capability; it essentially offers the ability to make a connection between two computers in order to manipulate the remote machine through a virtual terminal connection. Surprisingly, we are not here to discuss any of the actual functionality that `telnet` provides, because with regard to networking I find it is quite useful as a simple connection-testing tool, without knowing anything about what functionality the `telnet` command itself actually provides.

When we discussed ping , we talked about the downside to ICMP: it is easily blocked, and it is becoming more common in today's networks not to allow pings to be successful. This is unfortunate since ping has always been the most common form of network-connection testing, but the reality is that, if ping makes our lives easier, it also makes the lives of hackers easier. If we cannot rely on ping to tell us with certainty whether we can contact a remote system, what do we use instead? Another case that I see often is where a server itself might be responding correctly, but a particular service running on that server has a problem and is not responding. A simple ping may show the server to be online, but it can't tell us anything about the service specifically. By using the **Telnet Client** commands, we can easily query a server remotely. Even more importantly, we can opt to query an individual service on that server, to make sure it is listening as it is designed to do. Let me give you an example that I use all the time. I often set up new internet-facing web servers.

After installing a new web server, it makes sense that I would want to

test access to it from the internet to make sure it's responding, right? But maybe the website itself isn't online and functional yet, so I can't browse to it with Internet Explorer. It is quite likely that we disabled pings on this server or at the firewall level, because blocking ICMP over the internet is very common to lower the security vulnerability footprint on the web. So my new server is running, and we think we have the networking all squared away, but I cannot test pinging my new server because, by design, it fails to reply. What can I use to test this? `telnet`. By issuing a simple `telnet` command, I can tell my computer to query a specific port on my new web server, and find out whether it connects to that port. Doing this establishes a socket connection to the port on that server, which is much more akin to real user traffic than a ping would be. If a `telnet` command connects successfully, you know your traffic is making its way to the server, and the server service running on the port we queried seems to be responding properly.

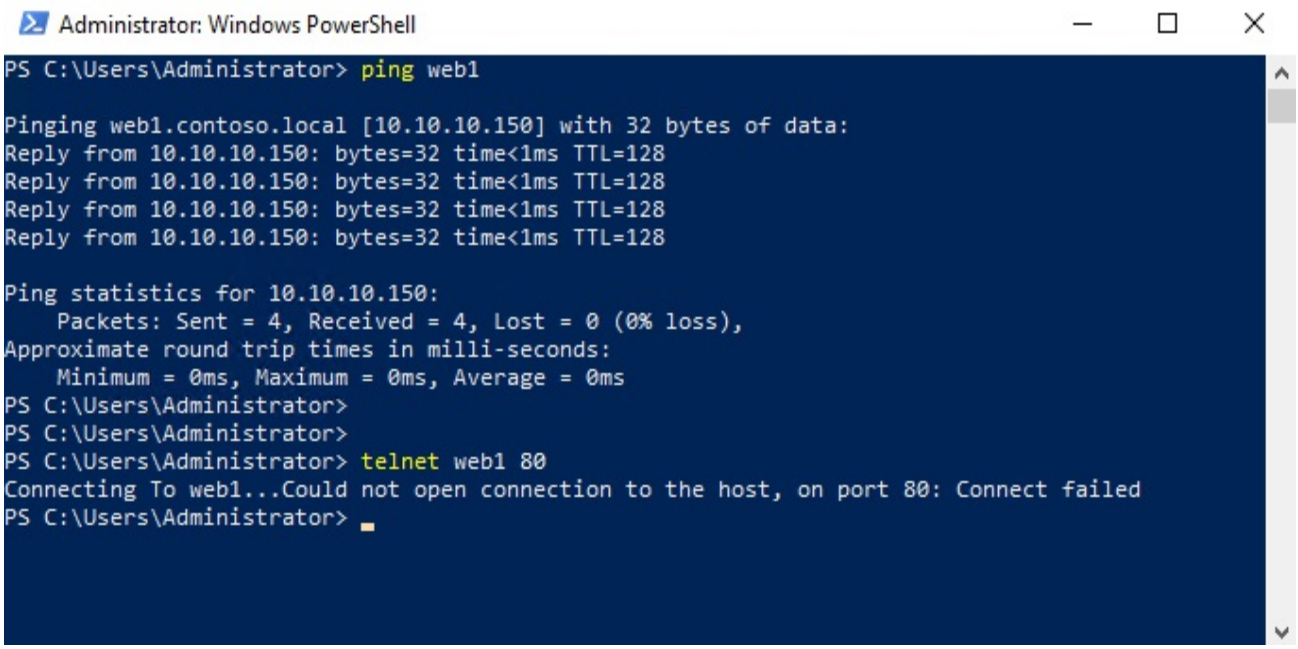
The ability to use Telnet is not installed by default in Windows Server 2019, or any Windows operating system, so we first need to head into Server Manager and Add Roles and Features in order to install the feature called Telnet Client:



You only need to install Telnet Client on the machine from which you want to do command-line testing. You do not have to do anything on the remote server that you are connecting to.

Now that the Telnet Client feature has been installed, we can utilize it from a Command Prompt or PowerShell in order to do work for us, by attempting to make socket connections from our computer to the remote service. All we need to do is tell it what server and port to query. Then telnet will simply connect or time out, and based on that result, we can see whether that particular service on the server is responding. Let's try it with our own web server. For our example I have turned off the website inside IIS, so we are now in the position where the server is online but the website is dead. If I ping `WEB1`, I can still see it happily responding. You can see where server-monitoring tools that rely on

ICMP would be showing false positives, indicating that the server was online and running, even though our website is inaccessible. Just below the successful ping in the following screenshot, you can see that I also tried querying port `80` on the `WEB1` server. The command that I used for that is `telnet web1 80`. That timed out. This shows us that the website, which is running on port `80`, is not responding:



If I turn the website back on, we can try `telnet web1 80` again, and this time I do not get a timeout message. This time, my PowerShell prompt wipes itself clean and sits waiting on a flashing cursor at the top. While it doesn't tell me *yay, I'm connected*!, this flashing - cursor indicates that a successful socket connection has been made to port `80` on my web server, indicating the website is online and responding:



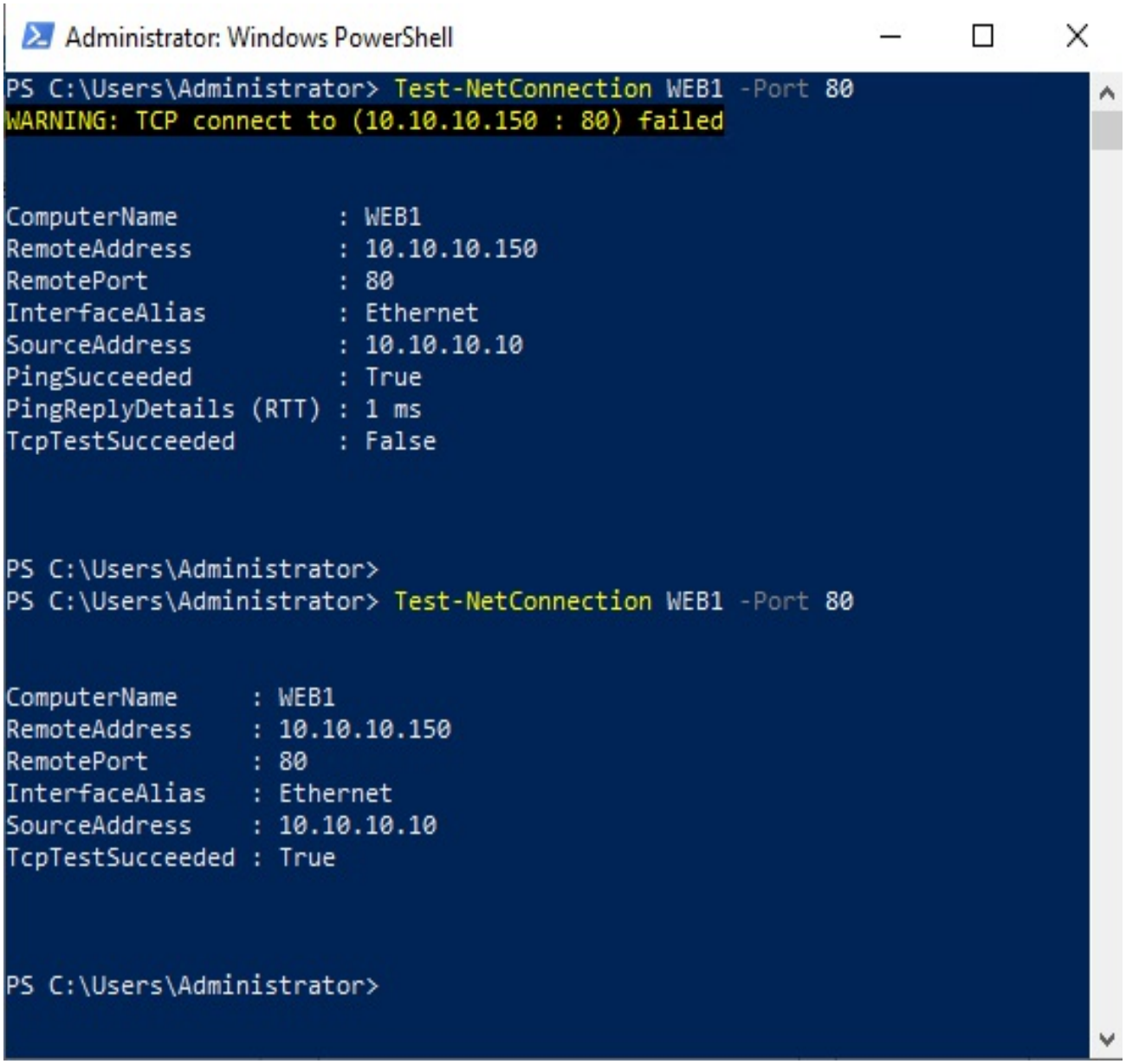
After creating a successful Telnet socket connection, you may be wondering how to get back to the regular PowerShell interface. Press the Ctrl+] keys together (that second one is a closed bracket key, usually next to the backslash key on your keyboard), type the word `quit`*, and then press Enter. This should return you to a regular prompt.*

Test-NetConnection

If ping has an equivalent and improved PowerShell cmdlet called `Test-Connection`, does PowerShell also contain an improved tool that works similarly to telnet for testing socket connections to resources? It sure does. `Test-NetConnection` is another way to query particular ports or services on a remote system, and the displayed output is friendlier than that of Telnet.

Let's walk through the same tests, once again querying port `80` on `WEB1`. You can see in the following screenshot that I have run the command twice. The first time the website on `WEB1` was disabled, and my connection to port `80` failed. The second time, I re-enabled the website, and I now see a successful connection.

```
Test-NetConnection WEB1 -Port 80
```


Administrator: Windows PowerShell
PS C:\Users\Administrator> Test-NetConnection WEB1 -Port 80
WARNING: TCP connect to (10.10.10.150 : 80) failed
ComputerName : WEB1
RemoteAddress : 10.10.10.150
RemotePort : 80
InterfaceAlias : Ethernet
SourceAddress : 10.10.10.10
PingSucceeded : True
PingReplyDetails (RTT) : 1 ms
TcpTestSucceeded : False
PS C:\Users\Administrator>
PS C:\Users\Administrator> Test-NetConnection WEB1 -Port 80
ComputerName : WEB1
RemoteAddress : 10.10.10.150
RemotePort : 80
InterfaceAlias : Ethernet
SourceAddress : 10.10.10.10
TcpTestSucceeded : True
PS C:\Users\Administrator>

Packet tracing with Wireshark or Message Analyzer

Eventually, you might need to look a little deeper into your network packets. Now we are entering territory where your network team may also be involved, but if you are familiar with these tools, you may be able to solve the problem before needing to call for assistance. Making use of command-line tools to check on the status of servers and services is very useful, but occasionally it may not be enough. For example, you have a client application that is not connecting to the application server, but you don't know why. Utilities such as ping and even telnet might be able to connect successfully, indicating that network routing is set up properly, yet the application fails to connect when it opens. If the application's own event logs don't help you troubleshoot what is going on, you might want to take a deeper look inside the network packets that the application is trying to push toward the server.

This is where the **Wireshark** and **Message Analyzer** tools come in handy. Both are free and easily downloadable, and they perform basically the same functions. They are designed to capture network traffic as it leaves from or arrives at a system, and they capture the information that is inside the packets themselves so that you can take a deeper look into what is going on. In our example of an application that cannot connect, you could run one of these tools on the client computer to watch the outgoing traffic, and also on the application server to watch for the incoming traffic from the client.

Each tool has quite a bit of individual functionality and we don't have

the space to cover all of it here, so I will leave you with links from which to get these tools so that you can test them out for yourself:

- **Wireshark**: https://www.wireshark.org/download.html
- **Microsoft Message Analyzer**: https://www.microsoft.com/en-us/download/details.aspx?id=44226

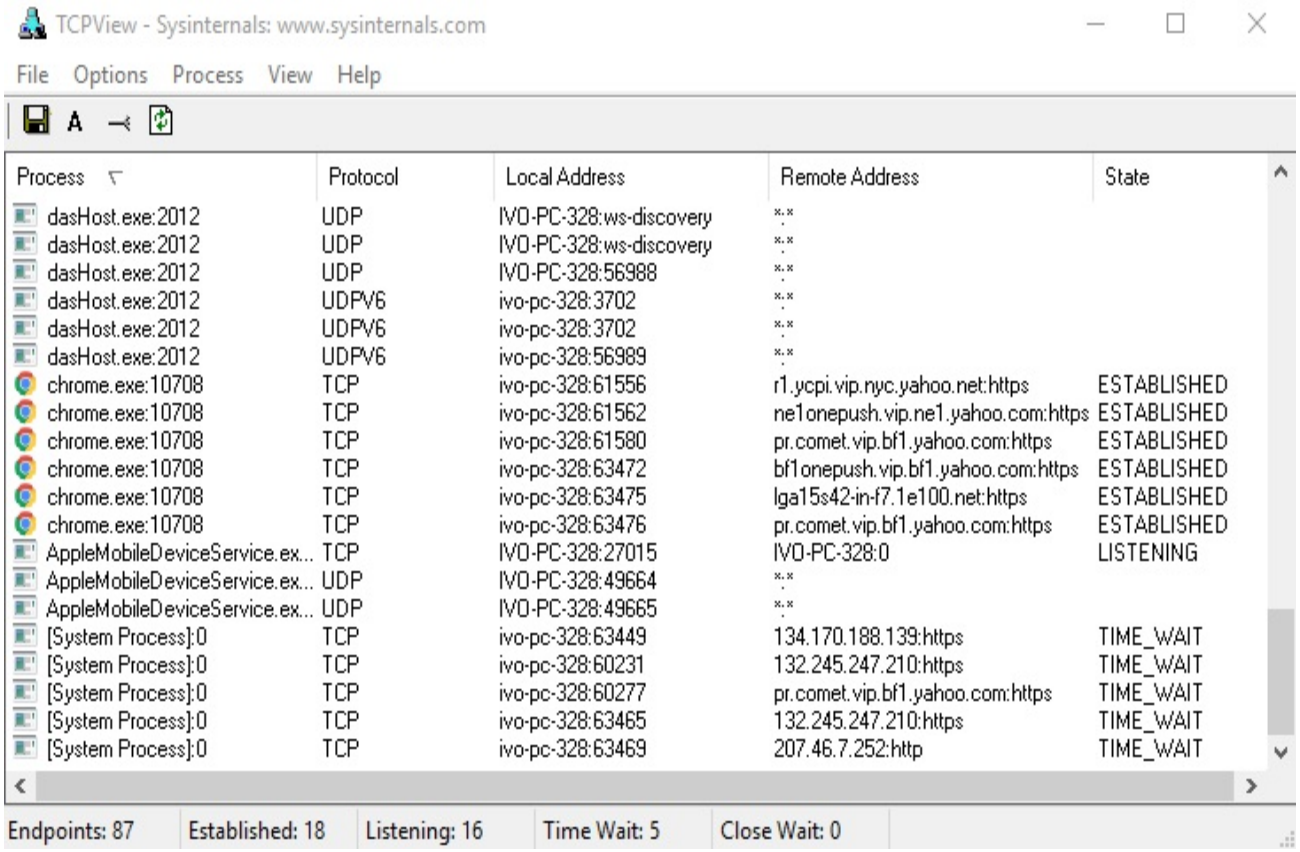
TCPView

The tools that we have discussed so far are great and can be used on a daily basis for poking and prodding individual resources that you want to test, but sometimes there are situations where you need to step back and figure out what it is you are looking for in the first place. Maybe you are working with an application on a computer and are not sure what server it is talking to. Or perhaps you suspect a machine of having a virus and trying to *phone home* to somewhere on the internet, and you would like to identify the location that it is trying to talk to or the process that is making the call. In these situations, it would be helpful if there was a tool that you could launch on the local computer that shows you all of the network traffic streams that are active on this computer or server, in a clear and concise way. That is exactly what **TCPView** does. TCPView is a tool originally created by Sysinternals; you may have heard of some of their other tools, such as ProcMon and FileMon. Running TCPView on a machine displays all of the active TCP and UDP connections happening on that computer in real-time. Also important is the fact that you do not need to install anything to make TCPView work; it is a standalone executable, making it extremely easy to use and clear off a machine when you are finished with it.

You can download TCPView from `https://technet.microsoft.com/en-us/sysinternals/tcpview.aspx`.

Simply copy the file onto a computer or server that you want to monitor, and double-click on it. The following is a screenshot of the TCPView interface running on my local computer, showing all of the connections that Windows and my applications are currently making. You can pause

this output to take a closer look, and you can also set filters to pare down the data and find what you are really looking for. Filters get rid of the *noise*, so to speak, and enable you to look more closely at a particular destination or a specific process ID:



Building a routing table

When you hear the term **routing table**, it is easy to pass that off as something the network folks need to deal with, something that is configured within the network routers and firewalls. It doesn't apply to the server admins, right? Networking servers together has been made pretty easy for us by only requiring an IP address, subnet mask, and default gateway, and we can instantly communicate with everything inside the rest of our network. While there is indeed a lot of networking magic going on under the hood that has been provided to us by networking equipment and network administrators, it is important to understand how routing inside Windows works because there will be some cases when you need to modify or build out a routing table right on a Windows Server itself.

Multi-homed servers

Running multi-homed servers is a case where you would certainly need to understand and work with a local Windows routing table, so let's start here. If you think this doesn't apply to you because you've never heard of multi-homed before, think again. Multi-homed is just a funny-looking word meaning your server has more than one NIC. This could certainly be the case for you, even if you are a small shop that doesn't have a lot of servers. Often, Small Business or Essentials Servers have multiple network interfaces, separating internal traffic from internet traffic. Another instance of a multi-homed server would be a remote access server that provides DirectAccess, VPN, or proxy capabilities at the edge of your network. Yet another reason to be interested and understand multi-homing is Hyper-V servers. It is very common for Hyper-V servers to have multiple NICs, because the VMs that are running on that server might need to tap into different physical networks within your organization.

Now that we have established what a multi-homed server is, you might still be wondering why we are discussing this. If I have more than one NIC, don't I simply configure each NIC individually inside Windows, giving each one an IP address, just like I would for any NIC on any server? Yes and no. Yes, you configure an IP address on each NIC, because it needs that for the identification and transport of packets on the network. No, you do not set up all of the NICs on your server in the same way. There is one critical item that you need to keep in mind and adhere to in order to make traffic flow properly on your multihomed server.

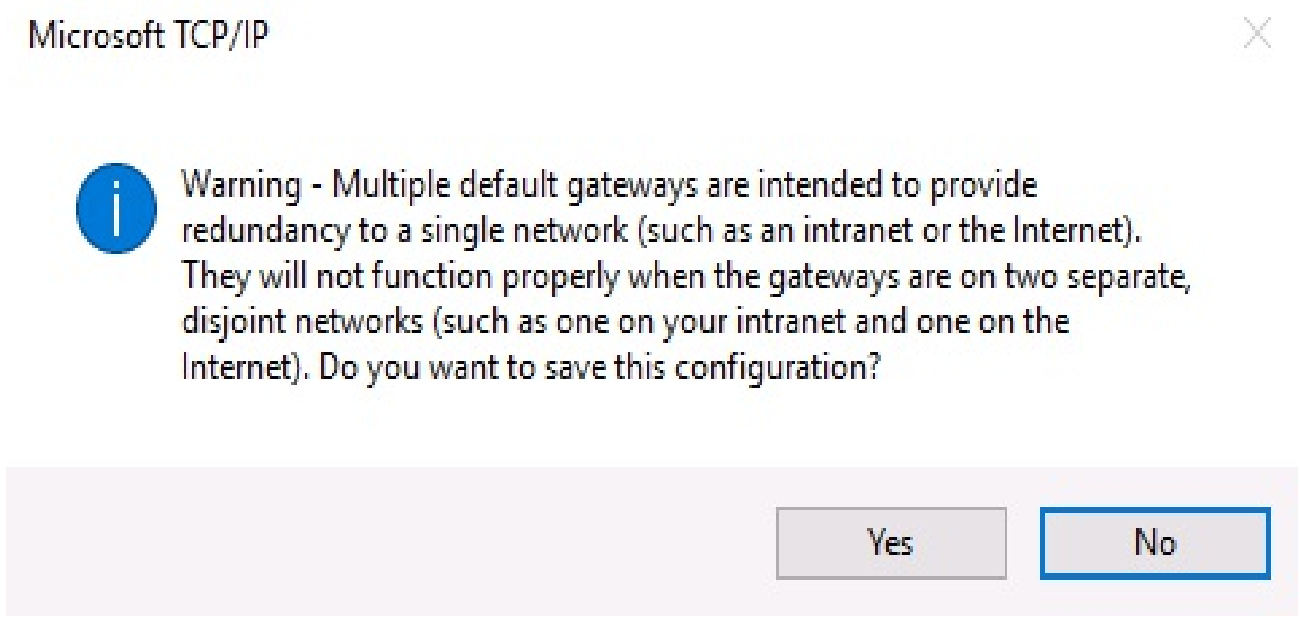
Only one default gateway

This is the golden ticket. When you multi-home a server by having multiple NICs, you can only have one default gateway. One for your entire server. This means you will have one NIC with a default gateway, and one or many NICs that do *NOT* have a default gateway inside their TCP/IP settings. This is extremely important. The purpose of a default gateway is to be the *path of last resort*. When Windows wants to send a packet to a destination, it browses over the local routing table—yes, there is a routing table even if you haven't configured it or ever looked at it—and checks to see whether a specific, static route exists for the destination subnet where this packet needs to go. If a route exists, it shoots the packet out that route and network interface to the destination. If no static route exists in the routing table, it falls back onto using the default gateway, and sends the traffic to that default gateway address. On all single NIC servers, the default gateway is a router that is designated with all of the routing information for your network, and so the server simply hands it to the router, and the router does the rest of the work.

When we have multiple NICs on a Windows Server, we cannot give each one a default gateway because it will confuse traffic flow from your server. It will be a crapshoot as to which default gateway traffic flows toward with every network transmission. I have helped many people troubleshoot servers in the field with exactly this problem. They needed to use their server as a bridge between two networks, or to have the server plugged into multiple different networks for whatever reason, and are now struggling because sometimes the traffic seems to work, and sometimes it doesn't. We start looking through the NIC properties

only to discover that every NIC has its own default gateway address in the TCP/IP properties. Bingo, that's our problem. The system is completely confused when it tries to send out traffic, because it doesn't know which gateway it needs to use at what times.

If you have ever tried adding default gateways to more than one NIC on the same server, you are probably familiar with the warning prompt that is displayed when you do this. Let's give it a try. I have added another NIC to one of my servers, and have IP settings configured on just one of the NICs. Now I will add a new IP address, subnet mask, and default gateway onto my second NIC. When I click on the OK button to save those changes, I am presented with the following popup:



This is one of those warnings that is easy to misread because of its slightly cryptic nature, but you get the essence of it: proceed at your own risk! And then what do most admins do at this point? Simply click through it and save the changes anyway. Then the routing problems start. Maybe not today, but perhaps the next time you reboot that server, or maybe three weeks down the road, but at some point your

server will start to send packets to the wrong destinations and cause you trouble.

Building a route

So what is our answer to all of this? Building a static routing table. When you have multiple NICs on a server, thereby making it multi-homed, you must tell Windows which NIC to use for what traffic inside the routing table. This way, when network traffic needs to leave the server for a particular destination, the routing table is aware of the different directions and paths that the traffic will need to take in order to get there, and will send it out accordingly. You will still be relying on routers to take the traffic the rest of the way, but getting the packets to the correct router by sending them out via the proper physical NIC is key to making sure that traffic flows quickly and appropriately from your multi-homed server.

Now that we understand why the routing table is important and conceptually how we need to use it, let's dig in and add a couple of routes on my dual-NIC server. We will add a route using the Command Prompt, and we will also add one using PowerShell, since you are able to accomplish this task from either platform, but the syntax used is different depending on which you prefer.

Adding a route with the Command Prompt

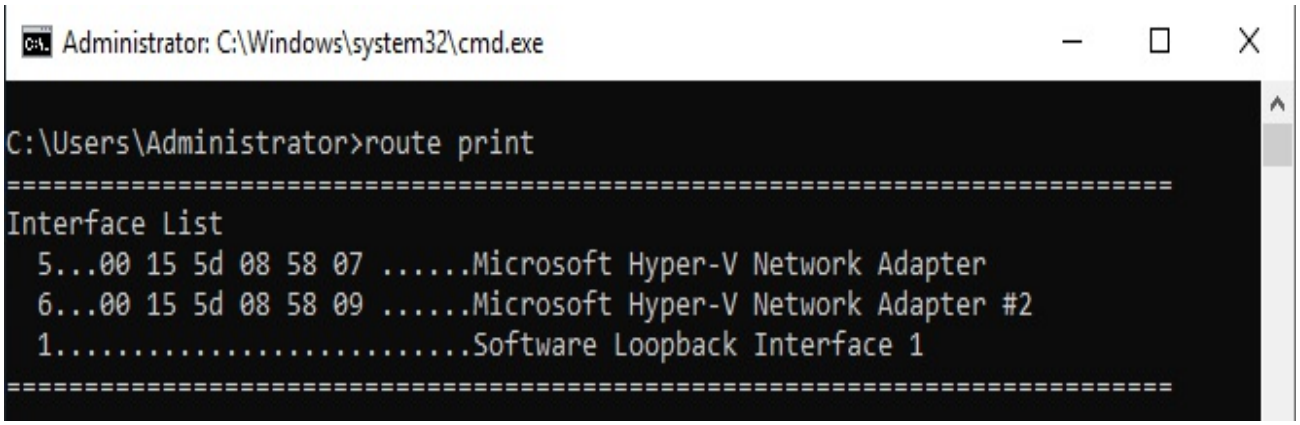
Before we can plan our new route, we need to get the lay of the land for the current networking configuration on this server. It has two NICs: one is plugged into my internal network and one is plugged into my DMZ that faces the internet. Since I can only have one default gateway address, it goes onto the DMZ NIC because there is no way that I could add routes for every subnet that might need to be contacted over the internet. By putting the default gateway on my DMZ NIC, the internal NIC does not have a default gateway, and is very limited in what it can contact at the moment. The internal subnet that I am physically plugged into is `10.10.10.0/24`, so I can currently contact anything in this small network from `10.10.10.1` through `10.10.10.254`. This is known as an **on-link** route; since I am plugged directly into this subnet, my server automatically knows how to route traffic inside this subnet. But I cannot contact anything else at the moment through my internal NIC, because the routing table knows nothing about the other subnets that I have inside my internal network. For example, I have an additional subnet, `192.168.16.0/24`, and there are some servers running within this subnet that I need to be able to contact from this new server. If I were to try to contact one of those servers right now, the packets would shoot out from my DMZ NIC, because the routing table on my server has no idea how to deal with `192.168` traffic, and so it would send it toward the default gateway. The following is the general syntax of the route statement we need to follow in order to make this traffic flow from our server into the new subnet:

```
Route add -p <SUBNET_ID> mask <SUBNET_MASK> <GATEWAY> IF <INTERFACE_ID>
```

Before we can type out our unique route statement for adding the `192.168` network, we need to do a little detective work and figure out what we are going to use in these fields. The following is a breakdown of the parts and pieces that are required to build a route statement:

- `-p`: This makes the command persistent. If you forget to put `-p` in the `route add` statement, this new route will disappear the next time you reboot the server. Not good.
- `SUBNET_ID`: This is the subnet we are adding; in our case it is `192.168.16.0`.
- `SUBNET_MASK`: This is the subnet mask number for the new route, `255.255.255.0`.
- `GATEWAY`: This one is a little confusing. It is very common to think it means you need to enter the gateway address for the new subnet, but that would be incorrect. What you are actually defining here is the first hop that the server needs to hit in order to send out this traffic. Or in other words, if you had configured a default gateway address on the internal NIC, what would that address be? For our network it is `10.10.10.1`.
- `INTERFACE_ID`: Specifying an interface ID number is not entirely necessary to create a route, but if you do not specify it, there is a chance that your route could bind itself to the wrong NIC and send traffic out in the wrong direction. I have seen it happen

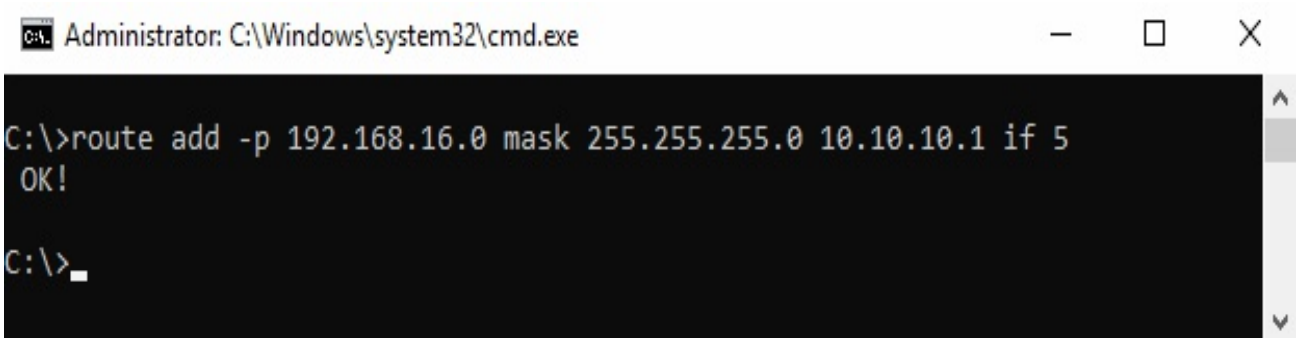
before, so I always specify a NIC interface ID number. This is typically a one- or two-digit number that is the Windows identifier for the internal NIC itself. We can figure out what the interface ID number is by looking at the `route print` command:



At the top of `route print` you see all of the NICs in a system listed. In our case, the internal NIC is the top one in the list; I identified it by looking at the MAC address for this NIC from the output of an `ipconfig /all` command. As you can see, my internal NIC's interface ID number is `5`. So in my `route add` statement, I am going to use IF 5 at the end of my statement to make sure my new route binds itself to that internal physical NIC.

The following is our completed `route add` statement:

```
route add -p 192.168.16.0 mask 255.255.255.0 10.10.10.1 if 5
```



If you now run a `route print` command, you can see our new `192.168.16.0` route listed in the `Persistent Routes` section of the routing table, and we can now send packets into that subnet from this new server. Whenever our server has traffic that needs to go into the `192.168.16.x` subnet, it will send that traffic out via the *internal* NIC, toward the router running on `10.10.10.1`. The router then picks up the traffic from there and brings it into the `192.168` subnet:

```
===========================================================================

Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
          0.0.0.0          0.0.0.0       10.10.10.1  Default
    192.168.16.0    255.255.255.0       10.10.10.1       1
===========================================================================
```

Deleting a route

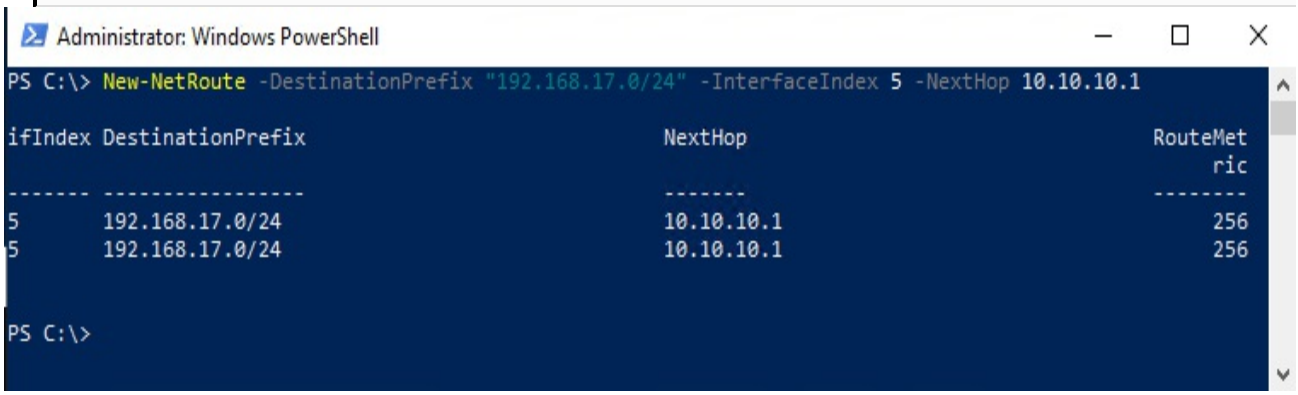
Occasionally, you may key in a route statement incorrectly. The best way to handle this is to simply delete the bad route, and then re-run your `route add` statement with the correct syntax. There are possibly other reasons why you might need to delete routes every now and then, so you'll want to be familiar with this command. Deleting routes is much simpler than adding new ones. All you need to know is the subnet ID for the route that you want to remove, then simply `route delete <SUBNET_ID>`. For example, to get rid of our `192.168.16.0` route that we created while we were working inside the Command Prompt, I would simply issue this command:

```
route delete 192.168.16.0
```

Adding a route with PowerShell

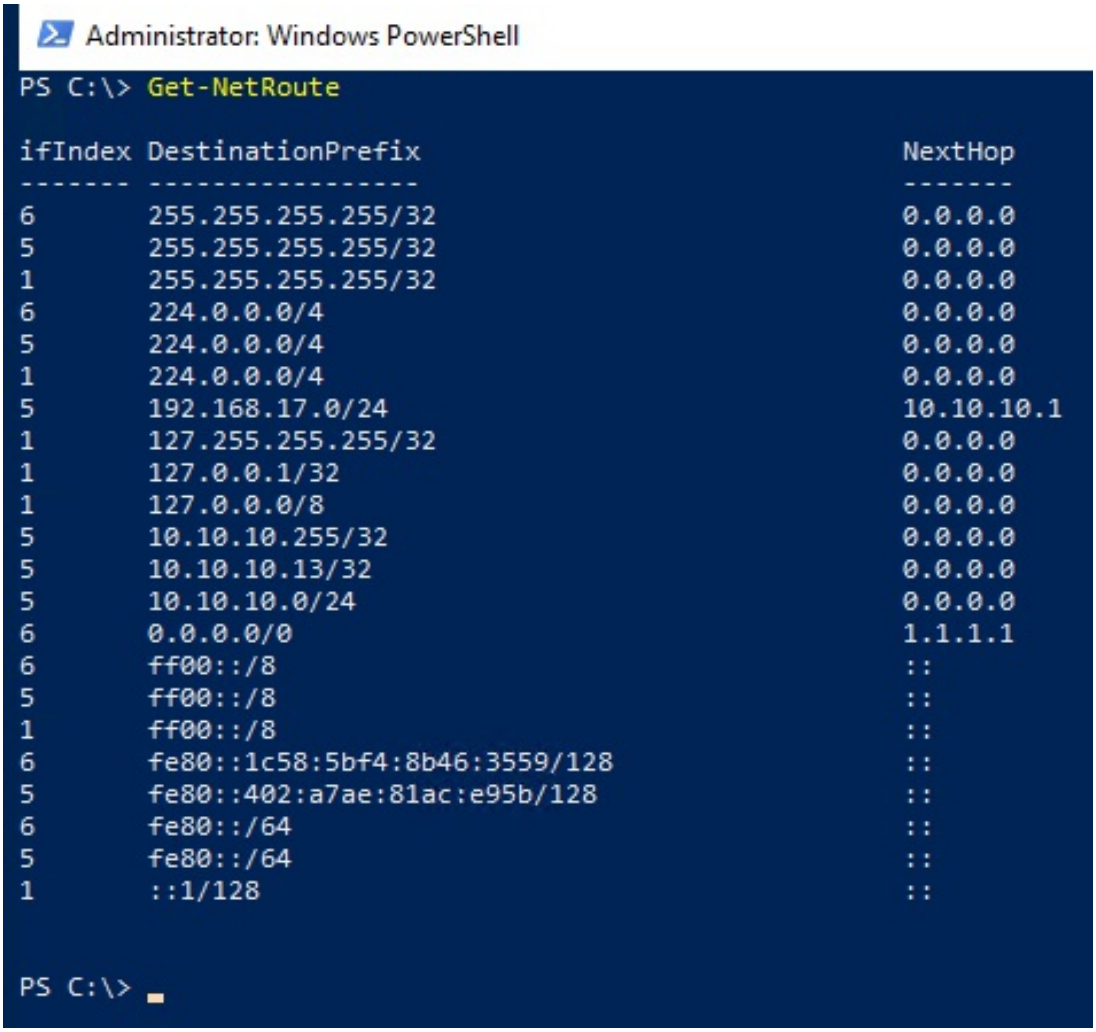
Since PowerShell is king when it comes to most command-line-oriented tasks inside Windows Server, we should accomplish the same mission from this interface as well. You can utilize the same `route add` command from inside the PowerShell prompt and that will work just fine, but there is also a specialized cmdlet that we can use. Let's utilize `New-NetRoute` to add yet another subnet to our routing table; this time we are going to add `192.168.17.0`. The following is a command we can utilize:

```
New-NetRoute -DestinationPrefix "192.168.17.0/24" -InterfaceIndex 5 -NextHop
10.10.10.1
```



You can see that the structure is similar, but a little bit friendlier. Instead of having to type the word `mask` and specify the whole subnet mask number, you can use the *slash* method to identify the subnet and mask within the same identifier. Also, where before we were specifying the *gateway*, which is always a little confusing, with the `New-NetRoute` cmdlet, we instead specify what is called the `NextHop`. This makes a little bit more sense to me.

Where we previously utilized `route print` in order to see our full routing table, the PowerShell cmdlet to display that table for us is simply `Get-NetRoute`:



NIC Teaming

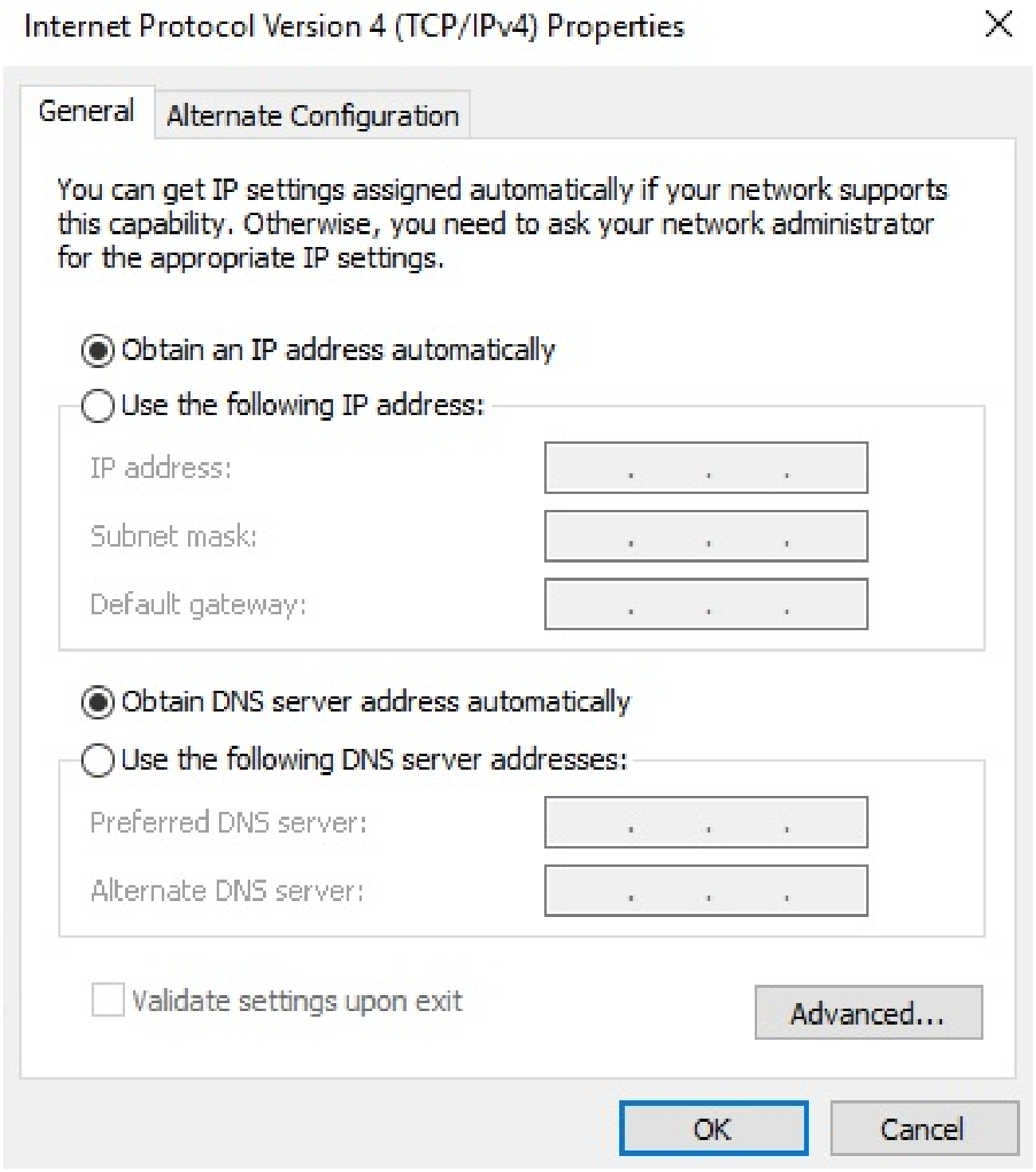
Moving on to another network topic that is becoming more and more popular on server hardware, let's walk through the steps to create NIC Teaming. The ability to team NICs together essentially consists of binding two or more physical network interfaces together, so that they behave as if they were a single network interface within Windows. This allows you to plug in two physical cables to two different switch ports, all using the same settings. That way, if one NIC port or switch port or patch cable goes bad, the server continues working and communicating without hesitation, because the teaming allows the NIC that is still working to handle the network traffic.

> *NIC Teaming itself is nothing new, it has been around for 10 years or more inside the Windows Server operating system. However, early versions were problematic, and in the field, I find that Server 2016 is the earliest server operating system most IT personnel consider to be stable enough to use NIC Teaming in production. So based on that, it is still relatively new to the wild.*

To begin teaming together your NICs, you need to make sure that you have multiple network cards on your server. I currently have four NIC ports on this machine. I have plans to create two teams: my first and second NICs will bind together to become an **Internal Network Team**, and my third and fourth NICs will become a **DMZ Network Team**. This way, I have network card redundancy on both sides of my network flow on this server.

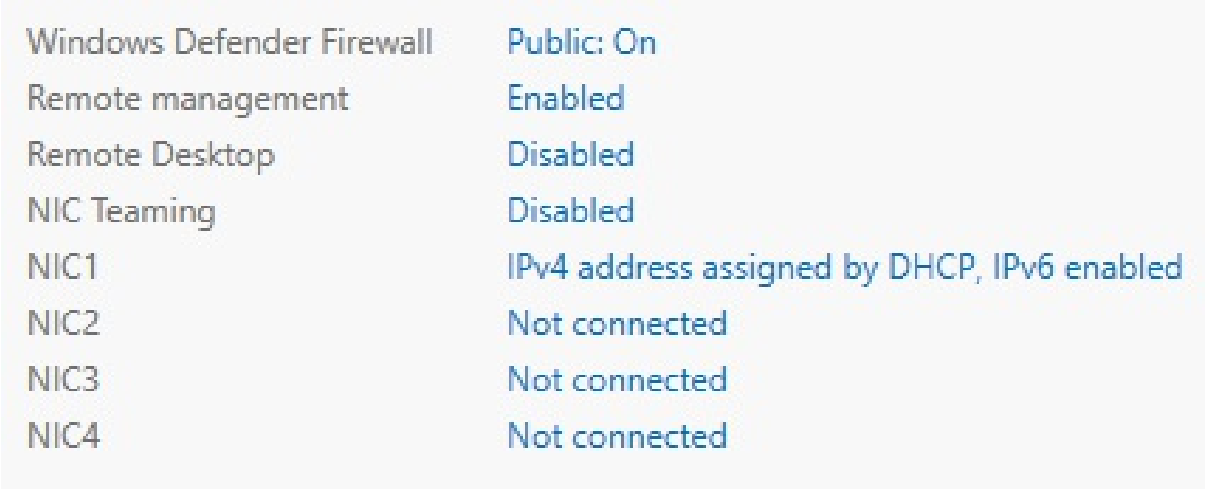
The first thing I want to do is clear out any IP addressing settings that might exist on my NICs. You see, once you tie together multiple NICs into a team, you will configure IP addressing settings on the team—you will no longer dive into individual NIC properties in order to assign IP

addresses. So open up the properties of each NIC, and make sure they are clear of static IP information, like so:



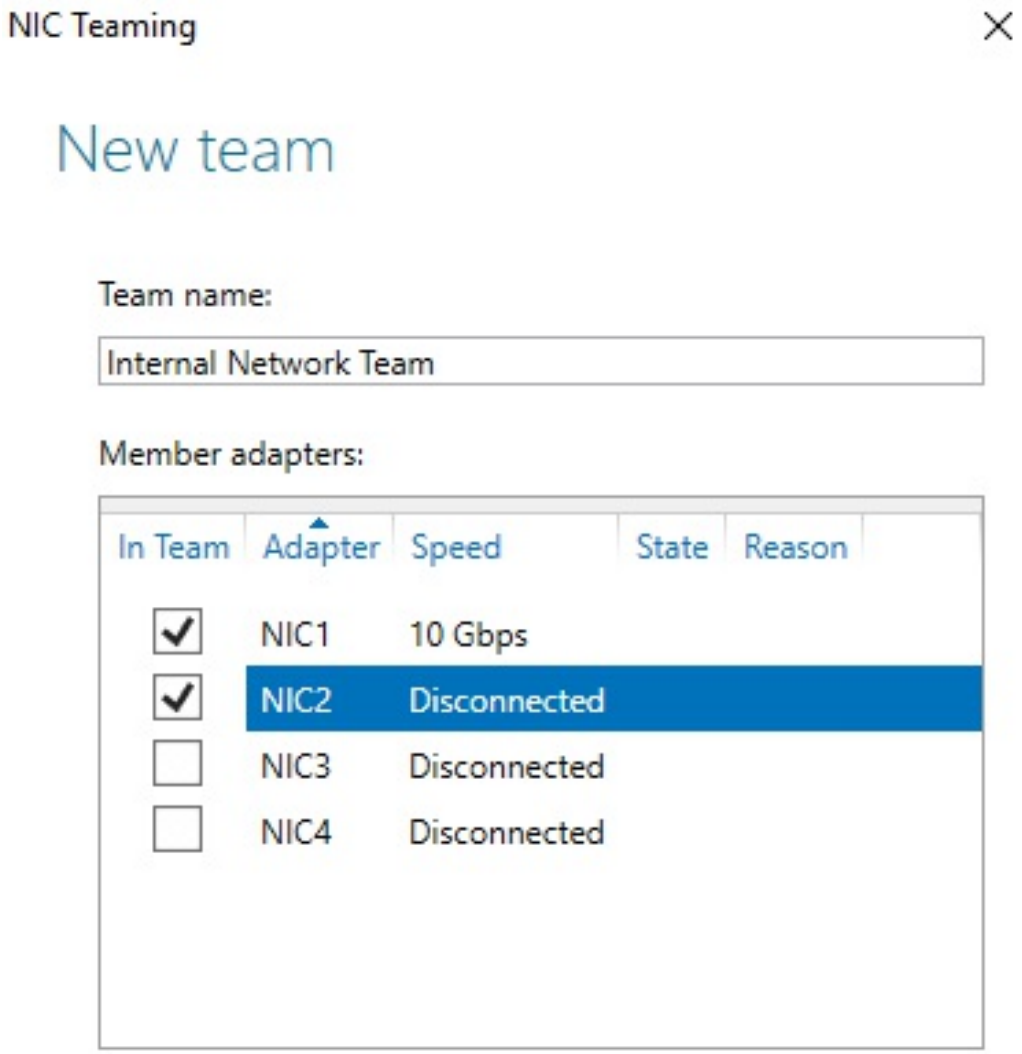
Now open up Server Manager, and click on Local Server. Looking inside the Properties information for your server, you will see listings for each

of your NICs, as well as an option called NIC Teaming, which is currently set to Disabled:

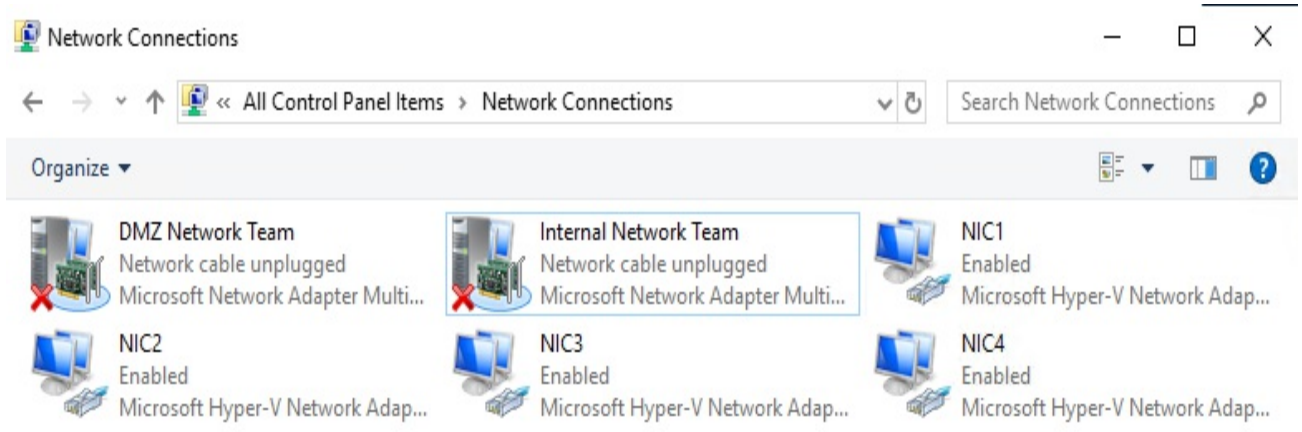


Go ahead and click on the word Disabled, and now look for a section entitled Teams. Click on the Tasks button, and choose to create a New Team.

Give your new team an appropriate name, and select the NICs that you want to be part of this team. Once finished, you can walk through the same steps as many times as you need in order to create additional teams with your remaining NICs:



Once finished, you will see your teams listed inside **Server Manager**, and if you open up the **Network Connections** screen inside Windows, you can see that, in addition to the four physical NICs, I now have two new entries listed here, which are the configuration locations for our new teams. From here I can right-click on each of my network teams, and configure IP addressing information just like I would have done on a single NIC. IPs input into the team properties will be in effect on all NICs that are part of the team:


Network Connections
« All Control Panel Items › Network Connections
Search Network Connections
Organize
DMZ Network Team
Network cable unplugged
Microsoft Network Adapter Multi...
Internal Network Team
Network cable unplugged
Microsoft Network Adapter Multi...
NIC1
Enabled
Microsoft Hyper-V Network Adap...
NIC2
Enabled
Microsoft Hyper-V Network Adap...
NIC3
Enabled
Microsoft Hyper-V Network Adap...
NIC4
Enabled
Microsoft Hyper-V Network Adap...

Software-defined networking

The flexibility and elasticity of cloud computing cannot be denied, and most technology executives are currently exploring their options for utilizing cloud technologies. One of the big stumbling blocks to adaptation is trust. Cloud services provide enormous computing power, all immediately accessible at the press of a button. In order for companies to store their data on these systems, the level of trust that your organization has in that cloud provider must be very high. After all, you don't own any of the hardware or networking infrastructure that your data is sitting on when it's in the cloud, and so your control of those resources is limited at best. Seeing this hurdle, Microsoft has made many efforts in recent updates to bring cloud-like technology into the local data center. Introducing server elasticity into our data centers means virtualization. We have been virtualizing servers for many years now, though the capabilities there are being continually improved. Now that we have the ability to spin up new servers so easily through virtualization technologies, it makes sense that the next hurdle would be our ability to easily move these virtual servers around whenever and wherever we need to.

Do you have a server that you want to move into a data center across the country? Are you thinking of migrating an entire data center into a new colocation across town? Maybe you have recently acquired a new company and need to bring its infrastructure into your network, but have overlapping network configurations. Have you bought into some space at a cloud service provider and are now trying to wade through the mess of planning the migration of all your servers into the cloud? These are all questions that needed an answer, and that answer is SDN.

SDN is a broad, general term that umbrellas many technologies working together to make this idea possible. Its purpose is to extend your network boundaries whenever and wherever you need. Let's take a look at some of the parts and pieces available in Windows Server 2019 that work in tandem to create a virtual networking environment, the first step in adopting our software-defined networking ideology.

Hyper-V Network Virtualization

The biggest component being focused on right now that brings the ability to pick up your networks and slide them around on a layer of virtualization lies within Hyper-V. This makes sense, because this is the same place you are touching and accessing to virtualize your servers. With Hyper-V Network Virtualization, we are creating a separation between the virtual networks and the physical networks. You no longer need to accommodate IP scheme limitations on the physical network when you set up new virtual networks, because the latter can ride on top of the physical network, even if the configurations of the two networks would normally be incompatible.

This concept is a little bit difficult to wrap your head around if this is the first time you are hearing about it, so let's discuss some real-world situations that would benefit from this kind of separation.

Private clouds

Private clouds are steamrolling through data centers around the World, because they make a tremendous number of sense. Anyone interested in bringing the big benefits of the cloud into their environments, while at the same time staying away from cloud negatives, can benefit from this. Building a private cloud gives you the ability to have dynamically-expanding and -shrinking compute resources and the ability to host multiple tenants or divisions within the same compute infrastructure. It provides management interfaces directly to those divisions so that the nitty-gritty setup and configuration work can be done by the tenant and you don't have to expend time and resources on the infrastructure-provider level making small, detailed configurations.

Private clouds enable all of these capabilities while staying away from the big scare of your data being hosted in a cloud service-provider's data center that you have no real control over, and all of the privacy concerns surrounding that.

In order to provide a private cloud inside your infrastructure, particularly one where you want to provide access to multiple tenants, the benefits of network virtualization become apparent, and even a requirement. Let's say you provide computing resources to two divisions of a company, and they each have their own needs for hosting some web servers. No big deal, but these two divisions both have administrative teams who want to use IP schemes that are within `10.0.0.0`. They both need to be able to use the same IP addresses, on the same core network that you are providing, yet you need to keep all of their traffic completely segregated and separated. These requirements would have

been impossible on a traditional physical network, but by employing the power of network virtualization, you can easily grant IP subnets and address schemes of whatever caliber each division chooses. They can run servers on whatever subnets and IP addresses they like, and all of the traffic is encapsulated uniquely so that it remains separated, completely unaware of the other traffic running around on the same physical core network that runs beneath the virtualization layer. This scenario also plays out well with corporate acquisitions. Two companies who are joining forces at the IT level often have conflicts with domains and network subnetting. With network virtualization, you can allow the existing infrastructure and servers to continue running with their current network config, but bring them within the same physical network by employing Hyper-V Network Virtualization.

Another simpler example is one where you simply want to move a server within a corporate network. Maybe you have a legacy line-of-business server that many employees still need access to, because their daily workload includes the LOB application to be working at all times. The problem with moving the server is that the LOB application on the client computers has a static IPv4 address configured by which it communicates with the server. When the user opens their app, it does something such as *talk to the server at* `10.10.10.10`. Traditionally, that could turn into a dealbreaker for moving the server, because moving that server from its current data center into a new location would mean changing the IP address of the server, and that would break everyone's ability to connect to it. With virtual networks, this is not an issue. With the ability to ride network traffic and IP subnets on the virtualization layer, that server can move from New York to San Diego and retain all of its IP address settings, because the physical network running underneath doesn't matter at all. All of the traffic is encapsulated before it is sent over the physical network, so the IP address of the legacy server can remain at `10.10.10.10`, and it can be picked up and moved anywhere in your environment without interruption.

Hybrid clouds

While adding flexibility to your corporate networks is already a huge benefit, the capabilities provided by virtualizing your networks expands exponentially when you do finally decide to start delving into real cloud resources. If and when you make the decision to move some resources to be hosted by a public cloud service provider, you will likely run a hybrid cloud environment. This means that you will build some services in the cloud, but you will also retain some servers and services on-site. I foresee most companies staying in a hybrid cloud scenario for the rest of eternity, as a 100% movement to the cloud is simply not possible given the ways that many of our companies do business. So now that you want to set up a hybrid cloud, we are again looking at all kinds of headaches associated with the movement of resources between our physical and cloud networks. When I want to move a server from on-site into the cloud, I need to adjust everything so that the networking configuration is compatible with the cloud infrastructure, right? Won't I have to reconfigure the NIC on my server to match the subnet that is running in my cloud network? Nope, not if you have your network virtualization infrastructure up and running. Once again, software-defined networking saves the day, giving us the ability to retain the existing IP address information on our servers that are moving, and simply run them with those IP addresses in the cloud. Again, since all of the traffic is encapsulated before being transported, the physical network that is being provided by the cloud does not have to be compatible with or distinct from our virtual network, and this gives us the ability to seamlessly shuttle servers back and forth from on-premise to the cloud without having to make special accommodations for networking.

How does it work?

So far it all sounds like a little bit of magic; how does this actually work and what pieces need to fit together in order to make network virtualization a reality in our organization? Something this comprehensive surely has many moving parts, and cannot be turned on by simply flipping a switch. There are various technologies and components running within a network that has been enabled for network virtualization. Let's do a little explaining here so that you have a better understanding of the technologies and terminology that you will be dealing with once you start your work with software-defined networking.

System Center Virtual Machine Manager

Microsoft System Center is a key piece of the puzzle for creating your software-defined networking model, particularly the **Virtual Machine Manager** (**VMM**) component of System Center. The ability to pick up IP addresses and move them to other locations around the World requires some coordination of your networking devices, and VMM is here to help. This is the component that you interface with as your central management point to define and configure your virtual networks. System Center is an enormous topic with many options and data points that won't fit in this book, so I will leave you with a link as a starting point on VMM learning: https://docs.microsoft.com/en-us/previous-versions/system-center/system-center-2012-R2/gg610610(v=sc.12).

Network controller

Microsoft's Network controller is a role that was initially introduced in Windows Server 2016, and as the name implies, it is used for control over network resources inside your organization. In most cases, it will be working side by side with VMM in order to make network configurations as centralized and seamless as possible. Network Controller is a standalone role and can be installed onto Server 2016 or 2019 and then accessed directly, without VMM, but I don't foresee many deployments leaving it at that. Interfacing with Network Controller directly is possible by tapping into its APIs with PowerShell, but is made even better by adding on a graphical interface from which you configure new networks, monitor existing networks and devices, or troubleshoot problems within the virtual networking model. The graphical interface that can be used is System Center VMM.

Network controller can be used to configure many different aspects of your virtual and physical networks. You can configure IP subnets and addresses, configurations and VLANs on Hyper-V switches, and you can even use it to configure NICs on your VMs. Network controller also allows you to create and manage **Access Control List** (**ACL**) type rules within the Hyper-V switch so that you can build your own firewalling solution at this level, without needing to configure local firewalls on the VMs themselves or having dedicated firewall hardware. Network controller can even be used to configure load balancing and provide VPN access through RRAS servers.

Generic Routing Encapsulation

Generic Routing Encapsulation (**GRE**) is just a tunneling protocol, but it's imperative to making network virtualization happen successfully. Earlier, when we talked about moving IP subnets around and about how you can sit virtual networks on top of physical networks without regard for making sure that their IP configurations are compatible, we should add that all of that functionality is provided at the core by GRE. When your physical network is running `192.168.0.x` but you want to host some VMs on a subnet in that data center, you can create a virtual network of `10.10.10.x` without a problem, but that traffic needs to be able to traverse the physical `192.168` network in order for anything to work. This is where routing encapsulation comes into play. All of the packets from the `10.10.10.x` network are encapsulated before being transported across the physical `192.168.0.x` network.

There are two different specific routing-encapsulation protocols that are supported in our Microsoft Hyper-V Network Virtualization environment. In previous versions of the Windows Server operating system, we could only focus on **Network Virtualization Generic Routing Encapsulation** (**NVGRE**), since this was the only protocol that was supported by the Windows flavor of network virtualization. However, there is another protocol, called **Virtual Extensible Local Area Network** (**VXLAN**), that has existed for quite some time, and many of the network switches—particularly Cisco—that you have in your environment are more likely to support VXLAN than they are NVGRE. So for the new network-virtualization platforms provided within Windows Server 2016+, we are now able to support either NVGRE or VXLAN, whichever best fits the needs of your company.

You don't necessarily have to understand how these GRE protocols work in order to make them do work for you, since they will be configured for you by the management tools that exist in this Hyper-V Network Virtualization stack. But it is important to understand in the overall concept of this virtual networking environment that GRE exists, and that it is the secret to making all of this work.

Microsoft Azure Virtual Network

Once you have Hyper-V Network Virtualization running inside your corporate network and get comfortable with the mentality of separating the physical and virtual networks, you will more than likely want to explore the possibilities around interacting with cloud service-provider networks. When you utilize Microsoft Azure as your cloud service provider, you have the ability to build a hybrid cloud environment that bridges your on-premise physical networks with remote virtual networks hosted in Azure. Azure's virtual network is the component within Azure that allows you to bring your own IP addresses and subnets into the cloud. You can get more info (and even sign up for a free trial of Azure virtual network) here: https://azure.microsoft.com/en-us/services/virtual-network/.

Windows Server Gateway/SDN Gateway

When you are working with physical networks, virtual networks, and virtual networks that are stored in cloud environments, you need some component to bridge those gaps, enabling the networks to interact and communicate with each other. This is where a Windows Server Gateway (also called an **SDN Gateway**) comes into play. Windows Server Gateway is the newer term; it was previously and is sometimes still called the Hyper-V Network Virtualization Gateway, so you might see that lingo in some of the documentation. A Windows Server Gateway's purpose is pretty simple: to be the connection between virtual and physical networks. These virtual networks can be hosted in your local environment, or in the cloud. In either case, when you want to connect networks, you will need to employ a Windows Server Gateway. When you are creating a bridge between on-premise and the cloud, your cloud service provider will utilize a gateway on their side, which you would tap into from the physical network via a VPN tunnel.

A Windows Server Gateway is generally a virtual machine, and is integrated with Hyper-V Network Virtualization. A single gateway can be used to route traffic for many different customers, tenants, or divisions. Even though these different customers have separated networks that need to retain separation from traffic of the other customers, cloud provider—public or private—can still utilize a single gateway to manage this traffic, because the gateways retain complete isolation between those traffic streams.

The Windows Server Gateway functionality existed in Server 2016, but once it was put into practice, some performance limitations that restricted network traffic throughput were discovered. Those overheads have now been increased dramatically in Windows Server 2019, meaning that you can flow more traffic and additional tenants through a single gateway than was previously possible.

Virtual network encryption

Security teams are continually concerned with the encryption of data. Whether that data is stored or on the move, making sure that it is properly secured and safe from tampering is essential. Prior to Server 2019, getting inner-network traffic encrypted while it was moving was generally the responsibility of the software application itself, not a network's job. If your software has the ability to encrypt traffic while it is flowing between the client and server, or between the application server and the database server, great! If your application does not have native encryption capabilities, it is likely the communications from that application are flowing in cleartext between the client and server. Even for applications that do encrypt, encryption ciphers, and algorithms are sometimes cracked and compromised, and in the future as new vulnerabilities are discovered, hopefully the way that your application encrypts its traffic can be updated in order to support newer and better encryption methods.

Fortunately, Windows Server 2019 brings us a new capability within the boundaries of software-defined networking. This capability is called **virtual network encryption**, and it does just what the name implies. When traffic moves between virtual machines and between Hyper-V servers (within the same network), entire subnets can be flagged for encryption, which means that all traffic flowing around in those subnets is automatically encrypted at the virtual networking level. The VM servers and your applications that are running on those servers don't have to be configured or changed in any way to take advantage of this encryption, as it happens within the network itself, automatically encrypting all traffic that flows on that network.

With Server 2019 SDN, any subnet in a virtual network can be flagged for encryption by specifying a certificate to use for that encryption. If the future happens to bring the scenario where the current encryption standards are out of date or insecure, the SDN fabric can be updated to new encryption standards, and those subnets will continue to be encrypted using the new methods, once again without having to make changes to your VMs or applications. If you are using SDN and virtual networks in your environments, enabling encryption on those subnets is a no-brainer!

Bridging the gap to Azure

Most companies that host servers in Microsoft Azure still have physical, on-premise networks, and one of the big questions that always needs to be answered is *How are we going to connect our physical data center to our Azure data center?* Usually, companies will establish one of two different methods to make this happen. You can deploy gateway servers on the edges of both your onsite and Azure networks, and connect them using Site-to-Site VPN. This establishes a continuous tunnel between the two networks. Alternatively, Microsoft provides a service called **Azure Express Route** that does effectively the same thing— it creates a permanent tunnel between your physical network and that of your Azure virtual networks. Either of these methods works great once configured, but these solutions might be considered overkill by small organizations that only have a few on-premise servers that need to be connected to the Azure Cloud.

Azure Network Adapter

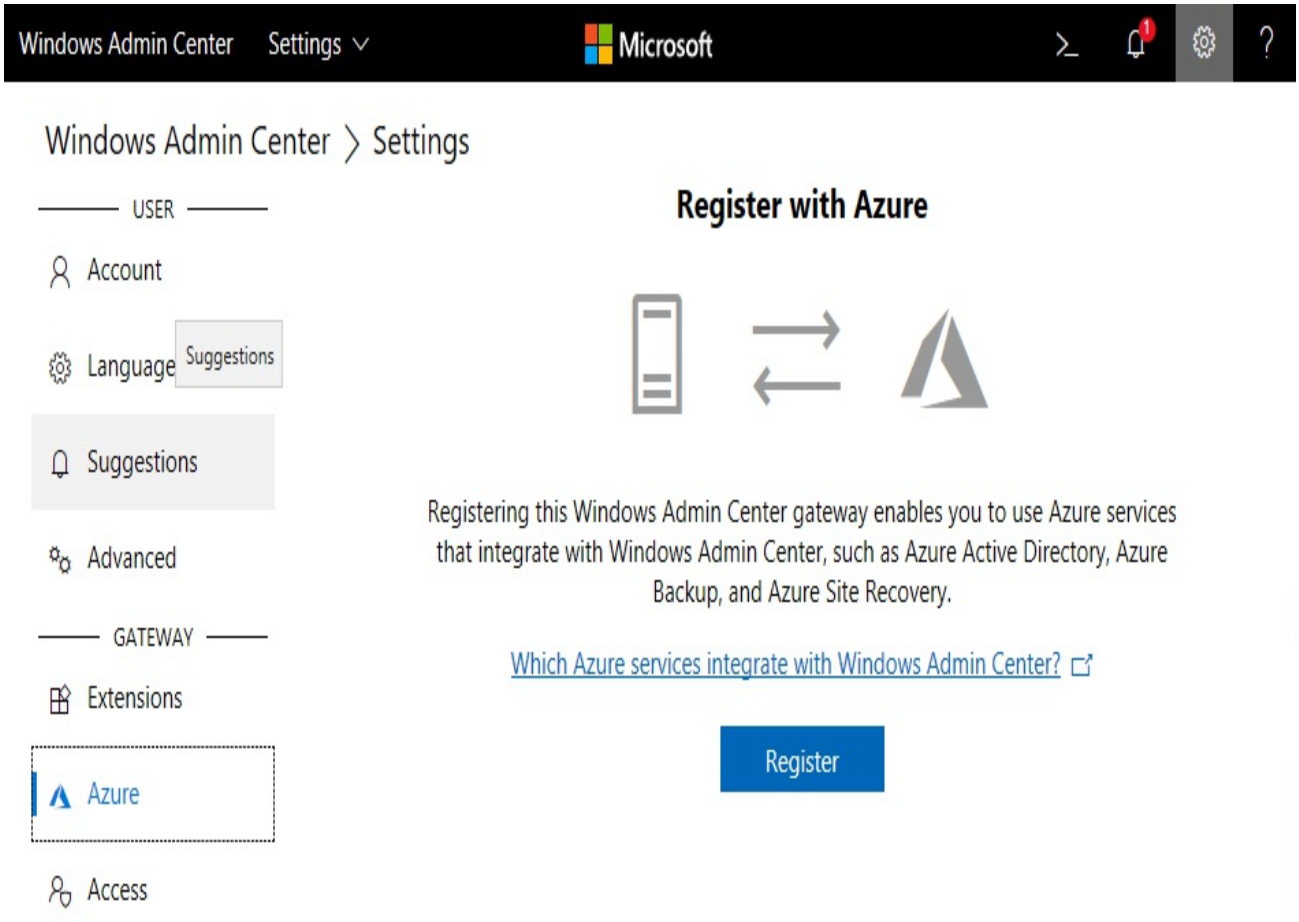
In the event that you have an on-premise server that you need to quickly connect to your Azure environment (and you don't already have a permanent connection established between your physical and Azure sites), there is a brand new hybrid cloud capability called **Azure Network Adapter**. In order to use one of these new network adapters, you must be utilizing the new Windows Admin Center to manage your servers.

Using Windows Admin Center, you can quickly add an Azure Network Adapter to an on-premise server, which connects it straight into your Azure network using a Point-to-Site VPN connection. Cool!

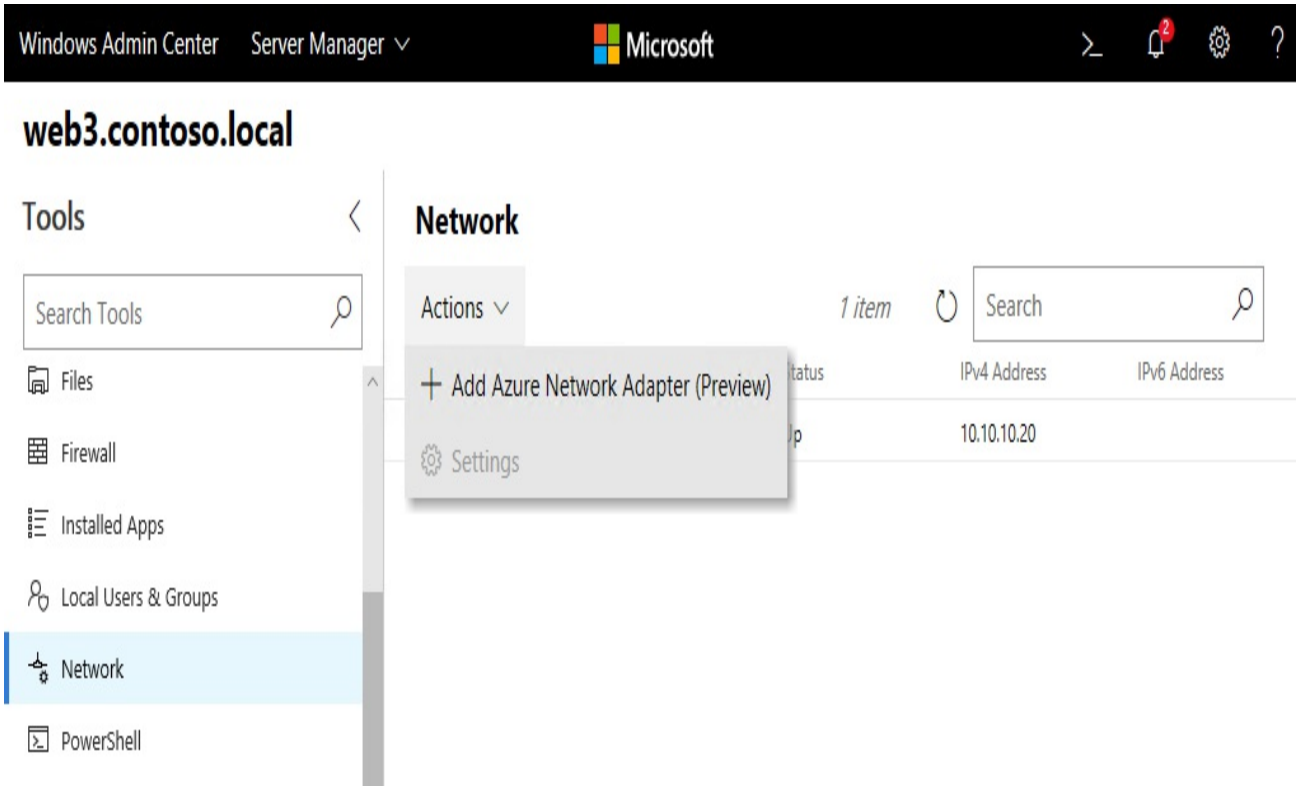
Even better, this capability has been back-ported so that you can add one of these adapters not only to Server 2019 machines, but also to Server 2016 and Server 2012 R2 machines as well.

To make this happen, there are a few requires that need to be in place: you must have an active Azure subscription, and you need to have at least one Azure Virtual Network configured.

Next, you need to register your Windows Admin Center with Azure. This is accomplished by opening up Windows Admin Center and visiting Settings. Once inside, navigate to GATEWAY | Azure and walk through the registration process:



Now that your WAC is registered with Azure, open up the server that you are managing from inside WAC and head over to the Network section. You will see listed here any NICs that are present on your server, and near the top of the window is a drop-down box for Actions. Inside, click on Add Azure Network Adapter.



You will find that all of the values that Azure needs in order to make this connection are auto-populated for you, based on your Azure network and subscription. If you don't already have an Azure Virtual Network, this wizard can even create one for you. You also get the opportunity to specify your own certificate for authenticating this connection, and doing so would be a good practice if you plan for this to be a long-term connection to Azure; otherwise, you can proceed without any input by allowing WAC/Azure to generate a self-signed certificate and simply click on the Create button. Windows Admin Center will go ahead and create a connection between your on-premise server and the Azure virtual network. That was only a couple of mouse clicks! This is an ingenious way of creating *ad hoc* connections between your servers and Azure very quickly and without complications.

If you later need to disconnect this server from the Azure network, you can open up Network Connections on that on-premise server, just like

you would when trying to modify NIC properties on your server, and you will find that what WAC has done under the hood is configure a Point-to-Site VPN connection, which is listed inside Network Connections. You can simply right-click on that Azure VPN connection, and disconnect it.

Summary

Server administration and network administration used to be segregated pretty clearly in most organizations, but over time those lines have blurred. There are numerous networking configurations and tasks that now need to be accomplished by Windows Server administrators, without needing to involve a networking team, so it is important that you have a good understanding of how your infrastructure connects together. Familiarity with the tools laid out in this chapter will provide you with the ability to configure, monitor, and troubleshoot the majority of Microsoft-centric networks.

Our introduction to software-defined networking may be a partially confusing section if you have never encountered this idea before, but hopefully it will prompt you to dig a little deeper and prepare yourself for dealing with this in the future. Whether you're ready or not, the cloud is here to stay. Microsoft on-premise networks now have numerous ways you can interact with Microsoft Azure, and it will soon be imperative that IT staff are familiar with these concepts. The idea of SDN will grow in popularity over the coming years; at the moment, it may seem daunting, but in five years, we may all look back and wonder how we ever made things work without virtual networks. There is much more information both in Microsoft Docs and in published books about Hyper-V Virtual Networking and System Center Virtual Machine Manager. I recommend a deeper familiarity this material if you are interested in trying this out for yourself. The next chapter deals with enabling your mobile workforce.

Questions

1. How many bits in length is an IPv6 address?
2. Re-write the following IPv6 address in condensed form: 2001:ABCD:0001:0002:0000:0000:0000:0001
3. What is the name of the command that is similar to trace route, but displays the local NIC which traffic is flowing out of?
4. True or False—On a server with multiple NICs, you can input a Default Gateway address onto each of those NICs.
5. What is the PowerShell cmdlet that can be used to create new routes on a Windows Server
6. Which Windows Server operating systems can be used with an Azure Network Adapter in order to connect them straight into Azure virtual networks?

Enabling Your Mobile Workforce

Giving employees the ability to remotely access corporate resources used to be a big benefit to most companies, but not necessarily a requirement. That has certainly changed in the past few years, where most companies and employees now have the expectation that they will be able to get their work done from wherever they happen to be. Cell phones are a big part of this equation, but are limited by the scope of what can be done with small screens and restricted operating systems. In order to grant remote workers the ability to do their jobs from home, coffee shops, or hotels, we have traditionally used **Virtual Private Networks** (**VPNs**).

Most VPNs in today's businesses are provided by products from companies other than Microsoft. The Remote Access role in Windows Server 2019 is here to change that. With many improvements having been made to the VPN components right in Windows Server, it is now a feasible and secure platform for providing access to corporate resources from remote computers. In addition to VPNs, we have a couple of newer technologies baked into Windows Server 2019 that are also designed to provide remote access to corporate resources, in a different way than a traditional VPN. The topics that we will cover in this chapter are the following:

- Always On VPN
- DirectAccess
- Remote Access Management Console

- DA, VPN, or AOVPN? Which is best?
- Web Application Proxy
- Requirements for WAP
- Latest improvements to WAP

Always On VPN

Giving a user access to a VPN connection traditionally means providing them with a special network connection link that they can launch, enter credentials to pass authentication, and then be connected to their work environment's network in order to communicate with company resources. After launching a VPN, users can open their email, find documents, launch their line-of-business applications, or otherwise work in the same ways that they can when physically sitting in their office. Also, when connected via a VPN, management of their laptop is possible, enabling successful a communication flow for systems such as Group Policy and SCCM. VPN connections offer great connectivity back to your network, but (remember, we are talking about traditional, regular VPN connections here) they only work when the user manually launches them and tells them to work. Anytime that a user has not connected to their VPN, they are navigating the internet with no connectivity back to the company data center. This also means that a traditional VPN connection obviously has no form of connectivity on the Windows login screen, because, until they get logged into the computer and find their way to the Windows desktop, users have no way of launching that VPN tunnel. This means that anything that might try to happen at the login screen, such as live authentication lookups, or during the login process, such as Group Policy processing or logon scripts, will not function via a traditional VPN.

Always On VPN (**AOVPN**), just as you have probably guessed based on the name, is simply the idea of making a VPN connection continuous and automatically connected. In other words, any time that the user has their laptop outside the office walls and is connected to the internet, a

VPN tunnel back to the corporate network is automatically established, ideally with zero user input to the process. This enables users to forget about the VPN altogether, as it is simply always connected and ready for them to use. They can log into their machines, launch their applications, and start working. It also means that IT management functions such as security policies, updates, and installer packages can push out to client machines a greater percentage of the time, since we no longer wait for the user to decide when they want to connect back to work; it happens automatically and pretty much all the time.

There are actually three different ways in which Always On VPN can be triggered on the client machine, and none of them involve the user having to launch a VPN connection:

- AOVPN can be configured to truly be Always On, meaning that as soon as internet access is available, it will always attempt to connect.
- Another option is **application triggering**, which means that you can configure AOVPN to launch itself only when specific applications are opened on the workstation.
- The third option is DNS name-based triggering. This calls the VPN connection into action when particular DNS names are called for, which generally happens when users launch specific applications.

Since you obviously don't need Always On VPN to be connected and working when your laptop is sitting *inside* the corporate network, we

should also discuss the fact that AOVPN is smart enough to turn itself off when the user walks through those glass doors. AOVPN-enabled computers will automatically decide when they are inside the network, therefore disabling VPN components, and when they are outside the network and need to launch the VPN tunnel connection. This detection process is known as **Trusted Network Detection**. When configured properly, Always On VPN components know what your company's internal DNS suffix is, and then it monitors your NIC and firewall profile settings in order to establish whether or not that same suffix has been assigned to those components. When it sees a match, it knows you are inside the network and then disables AOVPN.

Types of AOVPN tunnel

Before we get started on the particulars of the client and server components required to make AOVPN happen, there is an important core topic that needs to be understood in order to make appropriate decisions about how you want to utilize AOVPN in your company. There are two very different kinds of VPN tunnel that can be used with Always On VPN: a **User Tunnel** and a **Device Tunnel**. As you will learn later in this chapter, the ability to have two different kinds of tunnel is something included with AOVPN in order to bring it closer to feature parity with DirectAccess, which also has this dual-tunnel mentality. Let's take a minute and explore the purposes behind the two tunnels.

User Tunnels

The most common way of doing AOVPN in the wild (so far), a User Tunnel is authenticated on the user level. User certificates are issued from an internal PKI to your computers, and these certificates are then used as part of the authentication process during connection. User Tunnels carry all of the machine and user traffic, but it is very important to note that user Tunnels cannot be established while the computer is sitting on the login screen, because user authentication has not happened at that point. So, a User Tunnel will only launch itself once a user has successfully logged into the computer. With only a User Tunnel at play, the computer will not have connectivity back to the corporate network for management functions until someone has logged into the computer, and this also means that you will be relying on cached credentials in order to pass through the login prompt.

Device Tunnels

A Device Tunnel is intended to fill the gaps left by only running a User Tunnel. A Device Tunnel is authenticated via a machine certificate, also issued from your internal PKI. This means that the Device Tunnel can establish itself even prior to user authentication. In other words, it works even while sitting at the Windows login screen. This enables management tools such as Group Policy and SCCM to work regardless of user input, and it also enables real-time authentication against domain controllers, enabling users to log into the workstation who have never logged into it before. This also enables real-time password expiry resets.

Device Tunnel requirements

A User Tunnel can work with pretty much any Windows 10 machine, but there are some firm requirements in order to make use of a Device Tunnel. In order to roll out a Device Tunnel, you need to meet the following requirements:

- The client must be **domain-joined**.
- The client must be issued a **machine certificate**.
- The client must be running **Windows 10 1709 or newer**, and only **Enterprise** or **Education** SKUs have this capability.
- A Device Tunnel can only be **IKEv2**. This is not necessarily a requirement, but is important to understand once we get around to discussing what IKEv2 is and why it may or may not be the best connectivity method for your clients.

AOVPN client requirements

It is important to understand that the *Always On* part of Always On VPN is really client-side functionality. You can utilize AOVPN on a client computer in order to connect to many different kinds of VPN infrastructure on the backend. We will talk about that shortly, in the *AOVPN server components* section.

While creating regular, manual VPN connections has been possible on Windows client operating systems for 15 or 20 years, Always On VPN is quite new. Your workforce will need to be running Windows 10 in order to make this happen. Specifically, they will need to be running Windows 10 version 1607 or a more recent version.

The following are the supported SKUs:

- Windows 10 1607+
- Windows 10 1709+
- Windows 10 1803+

Wait a minute—that doesn't make any sense. Why list those three items separately if they are inclusive of one another? Because, while technically Always On VPN was officially introduced in Windows 10 1607, it has had some improvements along the way. Let's list those again, with a brief summary of what has changed over the years:

- **Windows 10 1607**: The original capability to auto-launch a VPN connection, thus enabling Always On VPN.
- **Windows 10 1709**: Updates and changes included the addition of Device Tunnel. If you intend to run a Device Tunnel for computer management purposes (and most enterprises will), then consider 1709 to be your minimum OS requirement.
- **Windows 10 1803**: Includes some fixes that were discovered from 1709. In reality, what this means is that I never see anyone implementing Always On VPN unless they are running 1803. Thankfully, the Windows 10 update platform is much improved, meaning that many more companies are rolling out newer versions of Win10 on an ongoing basis, and upgrading to 1803 is much less painful than, say, a Windows XP to Windows 7 migration.

Whether you are running 1607, 1709, 1803, or 1809—the particular SKU within those platforms does not matter. Well, it hardly matters. Always On VPN works with Windows 10 Home, Pro, Enterprise, and all of the other flavors. That is, the User Tunnel works with all of those.

It is important enough to point out once again: if you want to utilize a Device Tunnel with Always On VPN, using domain-joined, Windows 10 Enterprise or Education SKUs is a firm requirement.

Domain-joined

As we have already established, when you're interested in using the AOVPN Device Tunnel, your client computers must be domain-joined. However, if you are okay with only running the User Tunnel for AOVPN access, then there are no domain-membership requirements. Clients still need to be running Windows 10 1607 or newer, but they could be any SKU and could even be home computers that are joined to simple workgroups; no domain is required.

This is emphasized specifically in Microsoft documentation in many places, because it enables Always On VPN to be utilized (somewhat) with the **Bring Your Own Device** (**BYOD**) crowd. While this is interesting, I don't foresee it being at all common that companies would allow employees' personal computers to be connected to their VPN. Most organizations are trying to cater in a small way to the BYOD market by providing access to some resources via the cloud, such as Office 365 for email and documents. But connecting those personal computers and devices back to your network with a full-scale layer 3 VPN tunnel? I don't think so. That is the stuff of security administrators' nightmares.

Rolling out the settings

Let's say you have all of the server-side parts and pieces ready to roll for VPN connectivity, and in fact you have successfully established the fact that you can create ad hoc traditional VPN connections to your infrastructure with no problems. Great! Looks like you are ready from the infrastructural side. Now, what is necessary to get clients to start doing Always On connections?

This is currently a bit of a stiff requirement for some businesses. The configuration itself of the Always On VPN policy settings isn't terribly hard; you just have to be familiar with the different options that are available, decide on which ones are important to your deployment, and put together the configuration file/script. While we don't have the space here to cover all of those options in detail, the method for putting those settings together is generally to build out a manual-launch VPN connection, tweak it to the security and authentication settings you want for your workforce, and then run a utility that exports that configuration out to some configuration files. These VPN profile settings come in XML and PS1 (PowerShell script) flavors, and you may need one or both of these files in order to roll the settings around to your workforce. The following is a great starting point for working with these configurations: https://docs.microsoft.com/en-us/windows-server/remote/remote-access/vpn/always-on-vpn/deploy/vpn-deploy-client-vpn-connections.

Once you have created your configuration files, you then face the task of pushing that configuration out to the clients. You ideally need to have a **mobile device management** (**MDM**) solution of some kind in order to roll the settings out to your workforce. While many

technologies in the wild could be considered to be MDMs, the two that Microsoft is focused on are **System Center Configuration Manager** (**SCCM**) and **Microsoft Intune**.

If you have SCCM on-premise, great! You can easily configure and roll out PowerShell-based configuration settings to your client computers and enable them for Always On VPN.

Perhaps you don't have SCCM, but you are cloud-focused and you have all of your computers tapped into Intune? Wonderful! You could alternatively use Intune to roll out those AOVPN settings via XML configuration. One of the benefits of taking the Intune route is that Intune can manage non-domain-joined computers, so you could theoretically include users' home and personal computers in your Intune managed infrastructure, and set them up to connect.

SCCM and Intune are great, but not everybody is running them. There is a third option for rolling out Always On VPN settings via PowerShell scripting. While this is *plan B* from Microsoft (they would really prefer you to roll out AOVPN via an MDM), I'm afraid that PowerShell will be the reality for many SMB customers who want to utilize AOVPN. The biggest downside to using PowerShell in order to put AOVPN settings in place is that PowerShell needs to be run in elevated mode, meaning that it's difficult to automate because the logged on user (which is where you need to establish the VPN connection) needs to be a local administrator for the script to run properly.

I am hopefully and anxiously waiting for the day that they announce a Group Policy template for rolling out Always On VPN settings, but so far, there is no word on whether or not that will ever be an option. Everyone has Group Policy; not everyone has MDM. You will read in a few moments that the rollout of Microsoft DirectAccess connectivity settings (an alternative to AOVPN) is done via Group Policy, which is

incredibly easy to understand and manage. As far as I'm concerned, at the time of writing, DirectAccess holds a major advantage over AOVPN in the way that it handles the client-side rollout of settings. But, make sure you check out Microsoft Docs online to find the latest information on this topic, as AOVPN is being continuously improved and there will likely be some changes coming to this area of the technology.

AOVPN server components

Now that we understand what is needed on the client side to make Always On VPN happen, what parts and pieces are necessary on the server/infrastructure side in order to allow these connections to happen? Interestingly, the Always On component of AOVPN has nothing to do with server infrastructure; the Always On part is handled completely on the client side. Therefore, all we need to do on the server side is make sure that we can receive incoming VPN connections. If you currently have a workforce who are making successful VPN connections, then there is a good chance that you already have the server infrastructure necessary for bringing AOVPN into your environment.

Remote Access Server

Obviously, you need a VPN server to be able to host VPN connections, right? Well, not so obviously. In Windows Server, the role that hosts VPN, AOVPN, and DirectAccess connections is called the Remote Access role, but you can actually get Always On VPN working without a Windows Server as your Remote Access Server. Since the *Always On* part is client-side functionality, this enables VPN server-side infrastructures to be hosted by third-party vendors. Even though that is technically accurate, it's not really what Microsoft expects; nor is it what I find in the field. In reality, those of us interested in using Microsoft Always On VPN will be using Microsoft Windows Server in order to host the Remote Access role, which will be the inbound system that our remote clients connect to.

A lot of people automatically assume that AOVPN is married to Windows Server 2019 because it's a brand-new technology and Server 2019 was just released, but that is actually not the case at all. You can host your VPN infrastructure (the Remote Access role) on Server 2019, Server 2016, or even Server 2012 R2. It works the same on the backend, giving clients a place to tap into with their VPN connections.

After installing the Remote Access role on your new Windows Server, you will find that the majority of the VPN configuration happens from the **Routing and Remote Access** (**RRAS**) console. While configuring your VPN, you will find that there are multiple protocols that can be used in order to establish a VPN connection between client and server, and you should have at least a brief understanding of what those different protocols are. I'll list them here in order of *strongest and most*

secure, all the way down to **don't touch this one!**

IKEv2

The newest, strongest and, all-in-all, best way to connect your client computers via VPN or AOVPN, IKEv2 is the only way to connect the AOVPN Device Tunnel. IKEv2 requires machine certificates to be issued to your client computers in order to authenticate. This generally means that if you want clients to connect via IKEv2, those clients will be domain-joined. It is very important to note that IKEv2 uses UDP ports `500` and `4500` to make its connection.

SSTP

Considered to be the *fallback method* for connecting AOVPN connections, SSTP uses an SSL stream in order to connect. Because of this, it requires an SSL certificate to be installed on the Remote Access Server, but does not require machine certificates on the client computers. SSTP uses TCP port `443`, and so it is able to connect even from inside very restrictive networks where IKEv2 may fail (because of IKEv2's reliance on UDP).

L2TP

Not generally used for AOVPN deployments, L2TP is able to establish VPN connections by using certificates or a pre-shared key. Since you already have two better protocols at your disposal, you shouldn't be using this one.

PPTP

While still a valid configuration option inside RRAS, stay away from this guy! PPTP has essentially been cracked, and if you are still running VPN connections based on PPTP, you basically need to consider those traffic streams to be unencrypted and clear-text over the internet.

Certification Authority (CA)

Machine certificates, user certificates, SSL certificates... Oh, my! Clearly you need to be familiar with working with and deploying certificates in order to make use of Always On VPN. This is becoming more and more common with newer, well-secured technologies of any flavor. The major requirement here is that you will need to have PKI inside your environment and at least one Windows CA server in order to issue the necessary certificates. The following is a list of the places in which certificates could be used by an AOVPN infrastructure:

- **User certificates**: These are the certificates issued to your VPN users from an internal CA, used for authentication of the User Tunnel.
- **Machine certificates**: These are certificates issued to your workstations (mostly laptops) from an internal CA, used for authentication of the Device Tunnel.
- **SSL certificate**: Installed on your Remote Access Server in order to validate the incoming traffic for SSTP VPN connections.
- **VPN and NPS machine certificates**: Your Remote Access Server, as well as your NPS servers, which we will talk about in just a minute, require machine certificates issued from your internal CA.

Network Policy Server (NPS)

NPS is basically the authentication method for VPN connections. When a VPN connection request comes in, the Remote Access Server hands that authentication request over to an NPS server in order to validate who that user is, and also to verify that the user has permissions to log in via the VPN.

Most commonly when working with Microsoft VPN connections, we configure NPS so that it allows only users who are part of a certain Active Directory Security Group. For example, if you create a group called **VPN Users** and then point NPS to that group, it will only allow users whom you have placed inside that group make successful VPN connections.

NPS is another Windows Server role that can be hosted on its own system or spread across multiple servers for redundancy. As with the Remote Access role itself, there is no Server 2019 requirement for NPS. You could easily deploy it on previous versions of Windows Server just as well.

In small environments that have just a single Remote Access Server, it is common to co-host the NPS role right on the same server that is providing VPN connectivity.

DirectAccess

Throughout our discussion about Always On VPN, I mentioned Microsoft DirectAccess a couple of times. DirectAccess is another form of automatic VPN-like connectivity, but it takes a different approach than that of Always On VPN. Where AOVPN simply uses expected, well-known VPN protocols and does some crafty magic to automatically launch those otherwise traditional VPN tunnels, DirectAccess tunnels are quite proprietary. Tunnels are protected by IPsec, and are essentially impenetrable and also impersonable. I find that security teams love the protections and complexity surrounding DA tunnels because it is a connection platform that attackers have no idea how to tamper with, or how to replicate.

In my experience, at this point in the game, Microsoft DirectAccess is the most common reason that administrators deploy the Remote Access role on a Windows Server. As stated, the easiest way to think about DirectAccess is to think of it as an automatic VPN. Similar to VPN, its purpose is to connect users' computers to the corporate network when they are outside the office. Different from VPN, however, is the method that employees use in order to make this connection possible. DirectAccess is not a software component, it is a series of components that are already baked into the Windows operating system, working in tandem to provide completely seamless access for the user. What do I mean by seamless? In the same way that AOVPN connects without user interaction, there is nothing the user needs to do in order to make DirectAccess connect. It does that all by itself. As soon as the mobile computer receives an internet connection, whether that connection is a home Wi-Fi, public internet at a coffee shop, or a cell phone hotspot

connection, DirectAccess tunnels automatically build themselves using whatever internet connection is available, without any user input.

Whether using Always On VPN or DirectAccess, when your computer connects automatically it saves you time and money. Time is saved because the user no longer has to launch a VPN connection. Money is saved because time equals money, but also because having an *always-on* connection means that patching, security policies, and the management of those remote computers always happens, even when the user is working remotely. You no longer have to wait for users to get back into the office or for them to choose to launch their manual VPN connection in order to push new settings and policies down to their computers; it all happens wherever they are, as long as they have internet access. Clearly, with the advent of two different remote access technologies that are both focused on automatic connectivity for remote users, Microsoft is clearing the way for a more productive workforce. The terms **user-friendly** and **VPN** have never gone hand-in-hand before, but in the latest versions of the Windows operating systems, that is exactly the goal.

DirectAccess has actually been around since the release of Windows Server 2008 R2, and yet I regularly bump into people who have never heard of it. In the early days, it was quite difficult to deploy and came with a lot of awkward requirements, but much has changed over the past few years and DirectAccess is now easier than ever to deploy, and more beneficial than ever to have running in your environment.

The truth about DirectAccess and IPv6

One of the awkward requirements I mentioned *used* to be the need for IPv6 inside your network. With the first version of DirectAccess, this was an unfortunate requirement. I say unfortunate because, even today, in the year 2019, almost nobody is running IPv6 inside their corporate networks, let alone years ago when this technology was released—a lot of admins didn't even know what IPv6 was. Fortunately, the requirement for IPv6 inside your networks is no more. I repeat, just in case anybody wasn't paying attention or is reading old, outdated TechNet documents—*you do not need IPv6 to use DirectAccess*! I have seen too many cases where DirectAccess was considered by a company, but the project was tossed aside because reading on TechNet made them believe that IPv6 was a requirement, and they discarded DirectAccess as something that wouldn't work. You absolutely do not have to be running IPv6 in your network to make DirectAccess function, but it is important to understand how DirectAccess uses IPv6, because you will start to encounter it once your deployment gets underway.

When I am sitting at home, working on my company laptop, DirectAccess connects me to the corporate network. My internal network at work has absolutely no IPv6 running inside of it; we are a completely IPv4 network at the moment. This is true for most companies today. However, when I open Command Prompt and ping one of my servers from my DirectAccess laptop, this is what I see —apologies for the sanitized output of the screenshot:

```
Pinging    -vdt-02.    .local [fd63:c3  :4b8:7777::c0a8:  10]
ta:
Reply from fd63:c3  :4b8:7777::c0a8:  10: time=133ms
Reply from fd63:c3  :4b8:7777::c0a8:  10: time=59ms
Reply from fd63:c3  :4b8:7777::c0a8:  10: time=74ms
Reply from fd63:c3  :4b8:7777::c0a8:  10: time=54ms
```

What in the world is that? Looks like IPv6 to me. This is where IPv6 comes into play with DirectAccess. All of the traffic that moves over the internet part of the connection, between my laptop and the DirectAccess server that is sitting in my data center, is IPv6 traffic. My internal network is IPv4, and my DirectAccess server only has IPv4 addresses on it, and yet my DirectAccess tunnel is carrying my traffic using IPv6. This is the core of how DirectAccess works, and cannot be changed. Your DA laptop sends IPsec-encrypted IPv6 packets over the internet to the DA server, and when the DA server receives those packets, it has the capability to spin them down into IPv4 in order to send them to the destination server inside the corporate network. For example, when I open my Outlook and it tries to connect to my Exchange server, my Outlook packets flow over the DirectAccess tunnel as IPv6. Once these packets hit my DirectAccess server, that DA server reaches out to internal DNS to figure out whether my Exchange server is IPv4 or IPv6. If you are actually running IPv6 inside the network and the Exchange server is available via IPv6, the DA server will simply send the IPv6 packets along to the Exchange server. Connection complete! If, on the other hand, you are running IPv4 inside your network, the DA server will only see a single host record in DNS, meaning that the Exchange server is IPv4—only. The DirectAccess server will then manipulate the IPv6 packet, changing it down into IPv4, and then send it on its way to the Exchange server. The two technologies that handle this manipulation of packets are **DNS64** and **NAT64**, which you have probably seen in some of the documentation if you have read anything about DirectAccess online. The purposes of these technologies is to change the incoming IPv6 packet stream into IPv4 for networks where it

is required, which is pretty much every network at the moment, and spin the return traffic from IPv4 back up into IPv6 so that it can make its way back to the DirectAccess client computer over the IPv6-based IPsec tunnel that is connecting the DA client to the DA server over the internet.

It is important to understand that DirectAccess uses IPv6 in this capacity, because any security policies that you might have in place that squash IPv6 on the client computers by default will stop DirectAccess from working properly in your environment. You will have to reverse these policies in order to allow the clients to push out IPv6 packets and get their traffic across the internet. However, it is also very important to understand that you do *not* need any semblance of IPv6 running *inside* the corporate network to make this work, as the DirectAccess server can spin all of the traffic down into IPv4 before it hits that internal network, and most DA implementations that are active today run in exactly this fashion.

Prerequisites for DirectAccess

DirectAccess has a lot of moving parts, and there are many different ways in which you can set it up. However, not all of these ways are good ideas. So, in this section, we are going to discuss some of the big decisions that you will have to make when designing your own DirectAccess environment.

Domain-joined

The first big requirement is that the systems involved with DirectAccess need to be domain-joined. Your DA server, or servers, all need to be joined to your domain, and all of the client computers that you want to be DA-connected need to be joined to a domain as well. Domain membership is required for authentication purposes, and also because the DirectAccess client settings that need to be applied to mobile computers come down to these computers via Group Policy. I always like to point out this requirement early in the planning process, because it means that users who purchase their own laptops at a retail location are typically not going to be able to utilize DirectAccess—unless you are somehow okay with adding home computers to your domain—and so DA is really a technology that is designed for managing and connecting your corporate assets that you can join to the domain. It is also important to understand this requirement from a security perspective, since your DirectAccess server or servers will typically be facing the edge of your network. It is common for the external NIC on a DA server to sit inside a DMZ, but they also have to be domain-joined, which may not be something you are used to doing with systems in a perimeter network.

Supported client operating systems

Not all Windows client operating systems contain the components that are necessary to make a DirectAccess connection work. Enterprise does, which covers the majority of larger businesses who own Microsoft operating systems, but that certainly does not include everyone. I still see many small businesses using professional or even home SKUs on their client machines, and these versions do not include DirectAccess components. The following is a list of the operating systems that do support DirectAccess. During your planning, you will need to make sure that your mobile computers are running one of these:

- Windows 10 Enterprise
- Windows 10 Education
- Windows 8.0 or 8.1 Enterprise
- Windows 7 Enterprise
- Windows 7 Ultimate

DirectAccess servers get one or two NICs

One big question that needs to be answered even prior to installing the Remote Access role on your new server is: How many NICs are needed on this server? There are two supported methods for implementing DirectAccess.

Single NIC Mode

Your DirectAccess server can be installed with only a single NIC. In this case, you would typically plug that network connection directly into your internal network so that it had access to all of the internal resources that the client computers are going to need to contact during the user's DA sessions. In order to get traffic from the internet to your DirectAccess server, you would need to establish a **Network Address Translation** (**NAT**) from a public IP address into whatever internal IP address you have assigned to the DA server. Many network security administrators do not like this method, because it means creating a NAT that brings traffic straight into the corporate network without flowing through any kind of DMZ.

I can also tell you from experience that the single NIC mode does not always work properly. It does a fine job of spinning up a quick test lab or proof of concept, but I have seen too many problems in the field with people trying to run production DirectAccess environments on a single NIC. The ability to use only a single network card is something that was added to DirectAccess in more recent versions, so it was not originally designed to run like this. Therefore, I strongly recommend that, for your production DA install, you do it the right way and go with...

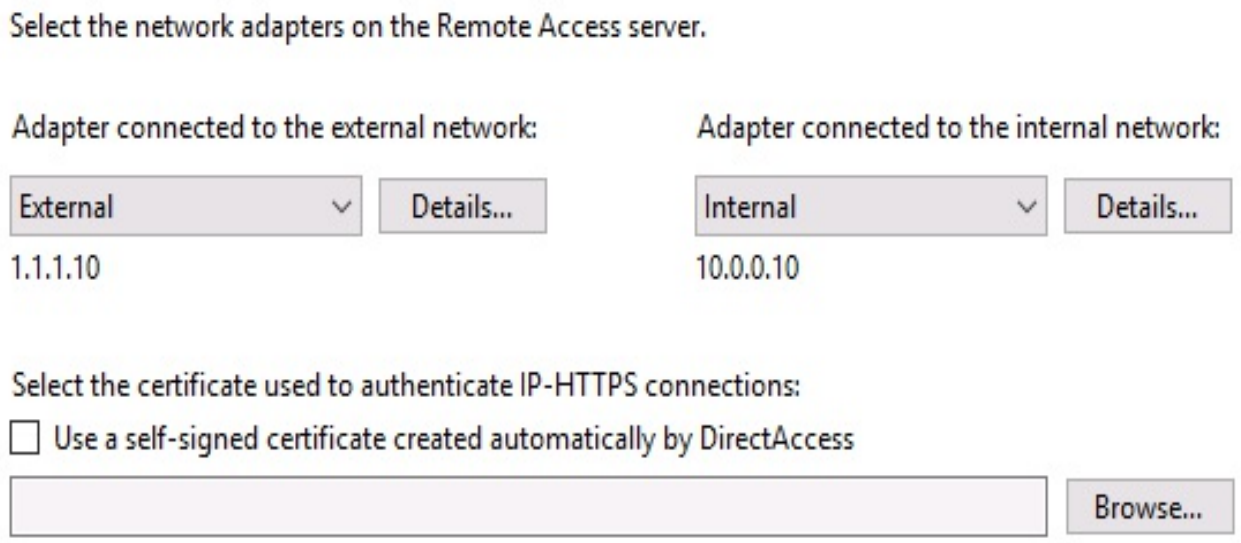
Dual NICs

Here, we have two network cards in the DirectAccess server. The internal NIC typically gets plugged right into the corporate network, and the external NIC's physical placement can vary depending on the organization. We will discuss the pros and cons of where to place the external NIC immediately after this section of the chapter. The edge mode with two NICs is the way that DirectAccess works best. As you may recall from earlier in the book, implementing a Windows Server with multiple NICs means that you will be multihoming this server, and you need to set up the network settings accordingly. With a Remote Access Server, your external NIC is always the one that receives the default gateway settings, so you need to make sure you follow this rule and *do not configure a default gateway on the internal NIC*. On the other hand, you do want to configure DNS server addresses into the internal NIC properties, but you do not want to configure DNS servers for the external NIC. Since this server is multihomed, you will likely need to create some route statements in order to add your corporate subnets into the Windows routing table of this server before it is able to successfully send and receive traffic. The only networks that would not need to accommodate adding static routes would be small networks, where all of your internal devices are on a single subnet. If this is the case, then you have no need to input static routes. But most corporate networks span multiple subnets, and in this case you should refer back to `Chapter 5`, *Networking with Windows Server 2019*, where we discussed multihoming and how to build out those route statements.

More than two NICs

Nope, don't go there. If you are familiar with configuring routers or firewalls, you know that you have the potential to install many different NICs on a server and plug them all into different subnets. While there are many reasons why splitting up network access like this on a Remote Access Server might be beneficial, it won't work how you want it to. The DirectAccess configuration itself is only capable of managing two different network interfaces.

As you can see in the following screenshot, in the course of the setup wizards, you will have to define one NIC as **External**, and the other as **Internal**. Any more NICs that exist in that server will not be used by DirectAccess, unfortunately. Maybe this is something that will change in future versions!



To NAT or not to NAT?

Now that you have decided to roll with two NICs in your DirectAccess server, where do we plug in the external NIC? There are two common places that this external network interface can be connected to, but depending on which you choose, the outcome of your DirectAccess environment can be vastly different. Before we talk about the actual placement of the NIC, I would like to define a couple of protocols that are important to understand, because they pertain very much to answering this question about NIC placement. When your DirectAccess laptop makes a connection to the DirectAccess server, it will do so using one of the three IPv6 transition tunneling protocols. These protocols are **6to4**, **Teredo**, and **IP-HTTPS**. When the DA client connects its DA tunnels, it will automatically choose which of these protocols is best to use, depending on the users current internet connection. All three protocols perform the same function for a DirectAccess connection: their job is to take the IPv6 packet stream coming out of the laptop, and encapsulate it inside IPv4 so that the traffic can make its way successfully across the IPv4 internet. When the packets get to the DirectAccess server, they are decapped so that the DA server can process these IPv6 packets.

6to4

DA clients will only attempt to connect using 6to4 when a remote laptop has a true public internet IP address. This hardly ever happens these days, with the shortage of available internet IPv4 addresses, and so 6to4 is typically not used by any DirectAccess client computers. In fact, it can present its own set of challenges when users are connecting with cell phone cards in their laptops, and so it is common practice to disable the 6to4 adapter on client computers as a DirectAccess best-practice setting.

Teredo

When DA clients are connected to the internet using a private IP address, such as behind a home router or a public Wi-Fi router, they will attempt to connect using the Teredo protocol. Teredo uses a UDP stream to encapsulate DA packets, and so, as long as the user's internet connection allows outbound UDP 3544, Teredo will generally connect and is the transition protocol of choice for that DirectAccess connection.

IP-HTTPS

If Teredo fails to connect, such as in the case where the user is sitting in a network that blocks outbound UDP, then the DirectAccess connection will fall back to using IP-HTTPS, pronounced *IP over HTTPS*. This protocol encapsulates the IPv6 packets inside IPv4 headers, but then wraps them up inside an HTTP header and encrypts them with TLS/SSL, before sending the packet out over the internet. This effectively makes the DirectAccess connection an SSL stream, just like when you browse an HTTPS website.

Installing on the true edge – on the internet

When you plug your DirectAccess server's external NIC directly into the internet, you grant yourself the ability to put true public IP addresses on that NIC. In doing this, you enable all three of the preceding transition tunneling protocols, so that DirectAccess client computers can choose between them for the best form of connectivity. When installing via the true edge method, you would put not only one, but two public IP addresses on that external NIC. Make sure that the public IP addresses are concurrent as this is a requirement for Teredo. When your DirectAccess server has two concurrent, public IP addresses assigned to the external NIC, it will enable the Teredo protocol to be available for connections.

> *The NIC does not necessarily have to be plugged directly into the internet for this to work. Depending on your firewall capabilities, you might have the option to establish a Bridged DMZ where no NATing is taking place. You need to check with your firewall vendor to find out whether or not that is an option for your organization. In this scenario, you are still able to configure true public IP addresses on the external NIC, but the traffic flows through a firewall first, in order to protect and manage that traffic.*

Installing behind a NAT

It is much more common for the networking team to want to place the external NIC of your DirectAccess server behind a firewall, inside a DMZ. This typically means creating a NAT in order to bring this traffic into the server. While this is entirely possible and better protects the DirectAccess server itself from the internet, it does come with a big downside. When you install a DA server behind a NAT, Teredo no longer works. In fact, the DirectAccess configuration wizards will recognize when you have a private IP address listed on the external NIC and will not even turn on Teredo.

When Teredo is not available, all of your DirectAccess clients will connect using IP-HTTPS. So why does it even matter if Teredo is unavailable? Because it is a more efficient protocol than IP-HTTPS. When Teredo tunnels packets, it simply encapsulates IPv6 inside IPv4. The DirectAccess traffic stream is already, and always, IPsec-encrypted, so there is no need for the Teredo tunnel to do any additional encryption. On the other hand, when IP-HTTPS tunnels packets, it takes the already-encrypted IPsec traffic stream and encrypts it a second time using SSL. This means all of the packets that come and go are subject to double encryption, which increases processing and CPU cycles, and makes for a slower connection. It also creates additional hardware load on the DirectAccess server itself, because it is handling twice the amount of encryption processing.

This is a particularly apparent problem when you are running Windows 7 on client computers, as double encryption processing will cause a noticeably slower connection for users. DirectAccess still works fine, but

if you sit a Teredo-connected laptop next to an IP-HTTPS-connected laptop, you will notice the speed difference between the two.

Thankfully, in Windows 8 and Windows 10, there have been some countermeasures added to help with this speed discrepancy. These newer client operating systems are now smart enough that they can negotiate the SSL part of the IP-HTTPS tunnel by using the NULL encryption algorithm, meaning that IP-HTTPS is not doing a second encryption and IP-HTTPS performance is now on a par with Teredo.

However, this only works for the newer client operating systems (Win7 will always double encrypt with IP-HTTPS), and it still doesn't work in some cases. For example, when you enable your DirectAccess server to also provide VPN connectivity, or if you choose to employ a **One-Time-Password** (**OTP**) system alongside DirectAccess, then the NULL algorithm will be disabled because it is a security risk in these situations, and so even your Windows 8 and Windows 10 computers will perform double encryption when they connect via IP-HTTPS. You can see where it would be beneficial to have Teredo enabled and available so that any computers that can connect via Teredo will do so.

To summarize, you can certainly install your DirectAccess server's external NIC behind a NAT, but be aware that all DA client computers will connect using the IP-HTTPS protocol, and it is important to understand the potential side-effect of implementing in this way.

Network Location Server

This major component in a DirectAccess infrastructure is something that does not even exist on the DA server itself, or at least it shouldn't if you are setting things up properly. **Network Location Server** (**NLS**) is simply a website that runs inside the corporate network. This website does not need to be available for access over the internet; in fact, it should not be. NLS is used as part of the inside/outside detection mechanism on DirectAccess client computers, similar to the way that Trusted Network Detection works for Always On VPN. Every time a DA client gets a network connection, it starts looking for the NLS website. If it can see the site, then it knows that you are inside the corporate network, and DirectAccess is not required, so it turns itself off. However, if your NLS website cannot be contacted, it means you are outside the corporate network, and DirectAccess components will start turning themselves on.

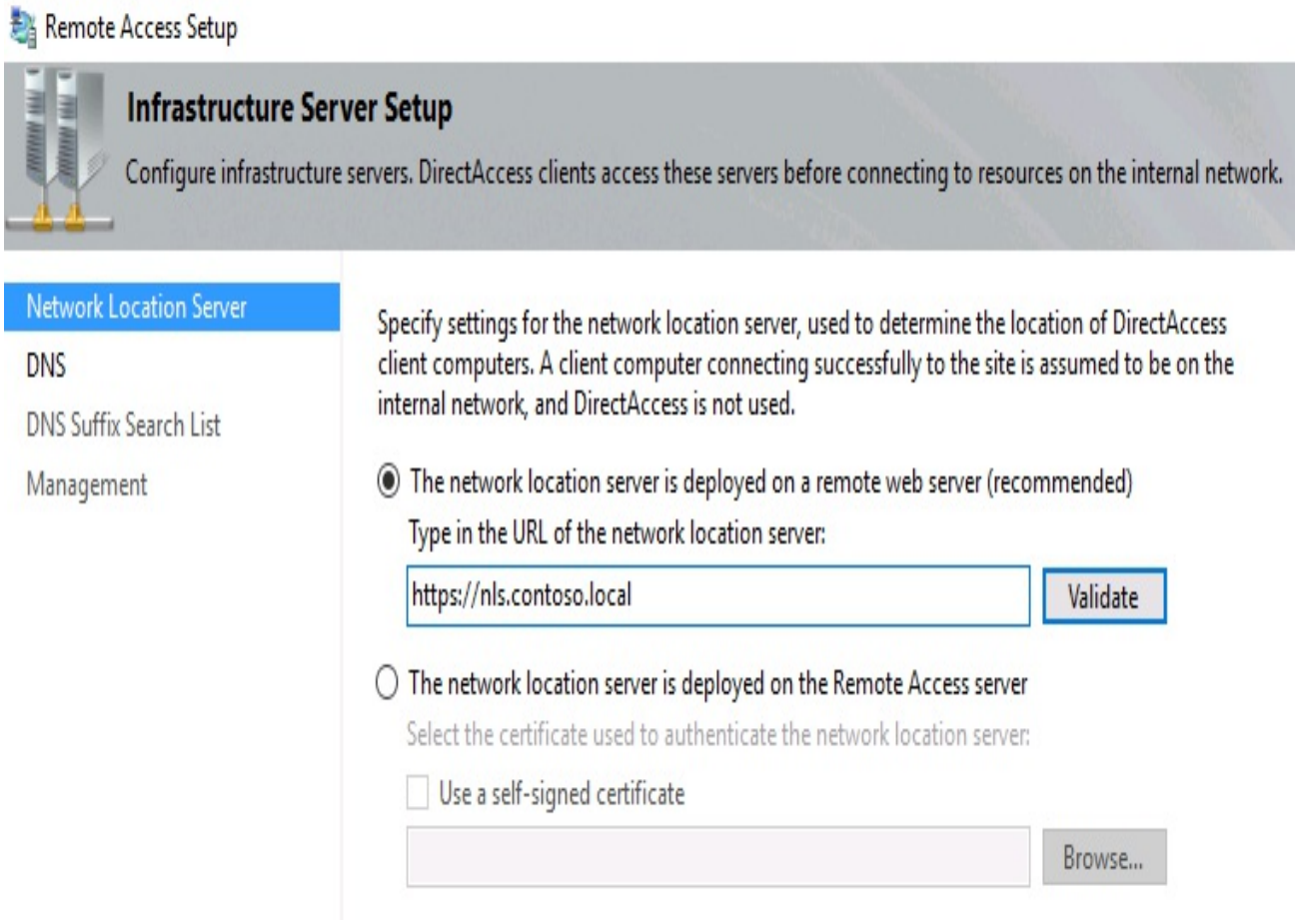
This prerequisite is easily met; all you need to do is spin up a VM and install IIS on it to host this new website, or you can even add a new website to an existing web server in your network. There are only two things to watch out for when setting up your NLS website. The first is that it must be an HTTPS site, and so it requires an SSL certificate. We will discuss the certificates used in DA, including this one, in the next section of this chapter. In addition to making sure that the website is accessible via HTTPS, you must also make sure that the DNS name you are using in order to contact this website is unique. You want to do this because, whatever name you choose for the NLS website, that name will not be resolvable when client computers are outside the corporate network. This is by design, because you obviously don't want your DA

clients to be able to successfully contact the NLS website when they are working remotely, as that would then disable their DirectAccess connection.

The reason I bring up the unique DNS name is that I often see new DirectAccess admins utilizing an existing internal website as their NLS website. For example, if you have `https://intranet` running as a SharePoint site, you could simply use this in the DA config as the NLS server definition. However, once you set it up this way, you will quickly realize that nobody who is working remotely can access the `https://intranet` website. This is by design, because the DA environment now considers your intranet website to be the NLS server, and you cannot resolve it while you are mobile. The solution to this problem? Make sure that you choose a new DNS name to use for this NLS website. Something like `https://nls.contoso.local` is appropriate.

The most important part about the Network Location Server that I want to stress is that you should absolutely implement this website on a server in your network that is *not* the DirectAccess server itself. When you are running through the DA config wizards, you will see on the screen where we define NLS that it is recommended to deploy NLS on a remote web server, but it also gives you the option to self-host the NLS website right on the DirectAccess server itself. Don't do it! There are many things that can go wrong when you co-host NLS on the DA server. Running NLS on your DA server also limits your DirectAccess potential in the future, because some of the advanced DA configurations require you to remove NLS from the DA server anyway, so you might as well do it correctly the first time you set it up. Changing your NLS website after you are running DA in production can be very tricky, and often goes sideways. I have helped numerous companies move their NLS website after realizing that they cannot co-host NLS on the DA server if and when they want to add a second DirectAccess server for growth or redundancy. The following a screenshot of the section in the DA config

wizard where you choose the location of NLS. Make sure you stick with the top box!



Certificates used with DirectAccess

Aside from reading and misunderstanding about how DirectAccess uses IPv6, here is the next biggest *turn off* for administrators who are interested in giving DirectAccess a try. Once you start to read about how DA works, you will quickly come to realize that certificates are required in a few different places. While VPNs generally also require the use of certificates, it is admittedly difficult to distinguish which certificates need to go where when you are wading through Microsoft Docs, so this section clears up any confusion that exists about DirectAccess certificates. It really is not very complicated, once you know what does and does not need to be done.

The core prerequisite is that you have a Windows CA server somewhere in your network. The stature of your PKI implementation is not that important to DirectAccess. We simply need the ability to issue certificates to our DA server and clients. There are only three places that certificates are used in DirectAccess, and two of them are SSL certificates.

SSL certificate on the NLS web server

As mentioned previously, your NLS website needs to be running HTTPS. This means that you will require an SSL certificate to be installed on the server that is hosting your NLS website. Assuming that you have an internal CA server, this certificate can be easily acquired from that internal CA, so there are no costs associated with this certificate. You do not need to purchase one from a public CA, because this certificate is only going to be accessed and verified from your domain-joined machines, the DirectAccess clients. Since domain-joined computers automatically trust the CA servers in your network, this certificate can simply be issued from your internal CA, and it will do exactly what we need it to do for the purposes of DirectAccess.

SSL certificate on the DirectAccess server

An SSL certificate is also required to be installed on the DirectAccess server itself, but this one should be purchased from your public certification authority. This certificate will be used to validate IP-HTTPS traffic streams coming in from the client computers, because that is SSL traffic and so we need an SSL certificate to validate it. Since the IP-HTTPS listener is facing the public internet, it is definitely recommended that you use a certificate from a public CA, rather than trying to use a cert from your internal PKI.

If your company already has a wildcard SSL certificate, use it here to save costs!

Machine certificates on the DA server and all DA clients

The last and most complicated part of the DirectAccess certificate puzzle is machine certificates. Once you know what is required though, it's really not hard at all. We simply require that a Computer or Machine certificate be installed on the DirectAccess server, as well as each of the DirectAccess client machines. This machine certificate is used as part of the authentication process for IPsec tunnels. It is a big part of the way in which DirectAccess verifies that you really are who you say you are when your computer makes that DA connection happen.

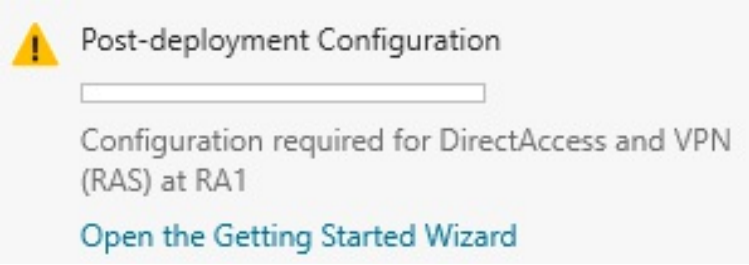
The best way to go about issuing these machine certificates is to log into your CA server and create a new certificate template that is duplicated from the built-in computer template. When setting up your new certificate template, just make sure that it meets the following criteria:

- The Common Name (subject) of the certificate should match the FQDN of the computer
- The **Subject Alternative Name** (**SAN**) of the certificate should equal the DNS Name of the computer
- The certificate should serve the intended purposes (Enhanced Key Usage) of both Client Authentication and Server Authentication

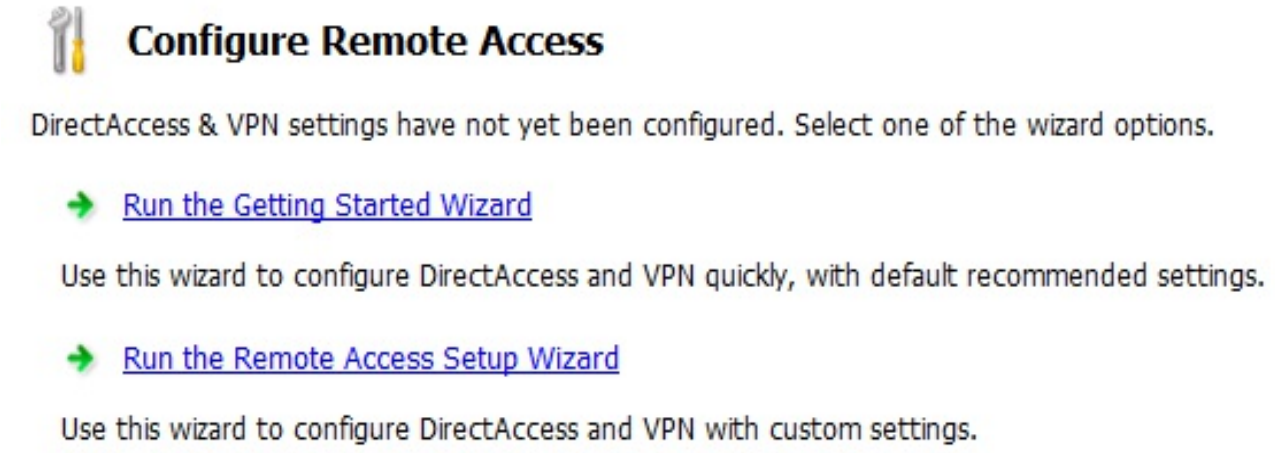
I should note here, though I don't really want to, that issuing these certificates is not absolutely necessary to make DirectAccess work. If you are running Windows 8 or higher on the client side, then it is possible to get DA working without machine certificates. Computers in the network can instead utilize something called **Kerberos Proxy** for their computer authentication when the IPsec tunnels are being built, but I highly recommend sticking with certificates. Using certificates as part of the authentication process makes the connection more stable and more secure. Additionally, as with the placement of NLS, if you want to perform any advanced functions with DirectAccess, such as load balancing or multisite, or even if you simply want to make some Windows 7 computers connect through DA, then you will be required to issue certificates anyway. So, just stick with the best-practice in the first place and issue these certificates before you even get started with testing DirectAccess.

Do not use the Getting Started Wizard (GSW)!

After making the necessary design decisions and implementing the prerequisites we have talked about so far, it is finally time to install the Remote Access role on your new DirectAccess server! After you have finished installing the role, similarly to many roles in Windows Server 2019, you will be shown a message informing you that additional configuration is required in order to use this role. In fact, if you follow the yellow exclamation mark inside Server Manager, the only option that you are presented with is Open the Getting Started Wizard. Ugh! This is *not* what you want to click on:



Your home for DirectAccess configurations is the Remote Access Management Console, which is available from inside the Tools menu of Server Manager now that our Remote Access role has been installed. Go ahead and launch that, and now we are presented with a choice:



Do *not* click on Run the Getting Started Wizard! The GSW is a shortcut method for implementing DirectAccess, designed only for getting DA up and running as quickly as possible, perhaps for a quick proof of concept. Under no circumstances should you trust the GSW for your production DA environment, because in an effort to make configuration quick and easy, many configuration decisions are made for you that are not best-practices.

You want to make sure and click on Run the Remote Access Setup Wizard instead when you are first prompted in the console; this will invoke the full set of DirectAccess configuration screens. DA setup consists of a series of four different steps, each containing a handful of screens that you will navigate through to choose your appropriate configuration options. There is a good amount of detail on these screens, as to what each one of them means and what your options are, so don't be afraid to dive in and set this up in the proper way. If you have already configured DirectAccess and used the Getting Started Wizard, DA may be working for you but it will not be running as efficiently or securely as it could be. The following is a quick list of the reasons why the Getting Started Wizard is not in your best interests. These are the things that it does that go directly against a best-practice DirectAccess install, with accompanying *peanut gallery* commentary

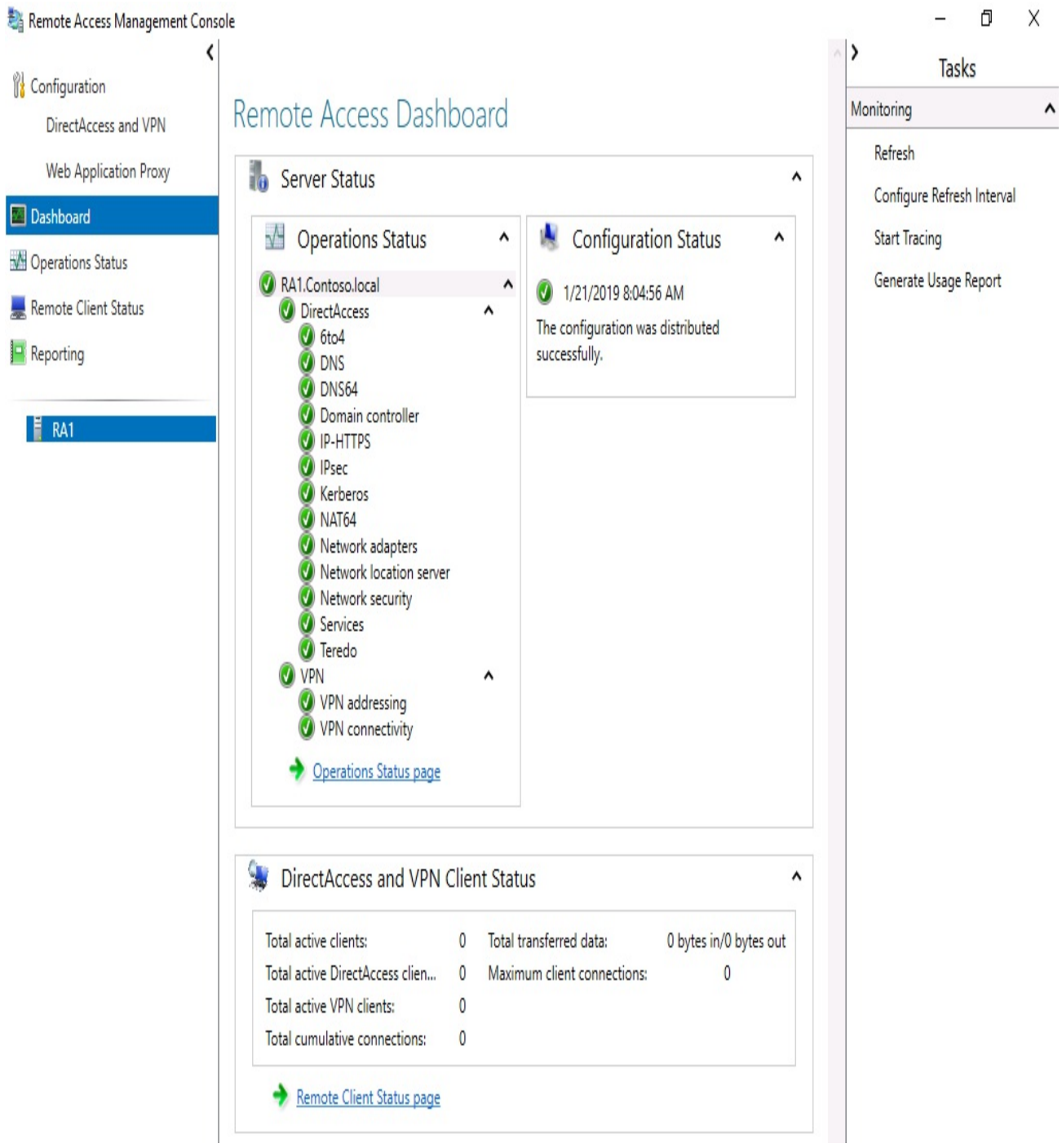
from yours truly:

- GSW co-hosts the NLS website on the DA server—*bad*
- GSW applies the DA client GPO settings to Domain Computers —*this is a terrible idea*
- GSW utilizes self-signed certificates—*a "Security 101 level" no-no*
- GSW automatically disables Teredo—*inefficient*
- GSW does not walk you through any of the advanced options for DirectAccess, probably because the way that it sets everything up invalidates your ability to even use the advanced functions —*lame*

Remote Access Management Console

You are well on your way to giving users remote access capabilities on this new server. As with many networking devices, once you have established all of your configurations on a Remote Access Server, it is pretty common for admins to walk away and let it run. There is no need for a lot of ongoing maintenance or changes to that configuration once you have it running well. However, Remote Access Management Console in Windows Server 2019 is useful not only for the configuration of remote access parts and pieces, but for monitoring and reporting as well. When working with DirectAccess, this is your home for pretty much everything: configuration, management, and monitoring. On the VPN/AOVPN side of the remote access toolset, you will be making many of the VPN configuration decisions inside RRAS, but RAMC is the place to go when checking over server-side monitoring, client-connection monitoring, and reporting statistics. Whether you use DA, VPN, or a combination of the two, RAMC is a tool you need to be comfortable with.

Let's take a look inside this console so that you are familiar with the different screens you will be interacting with:

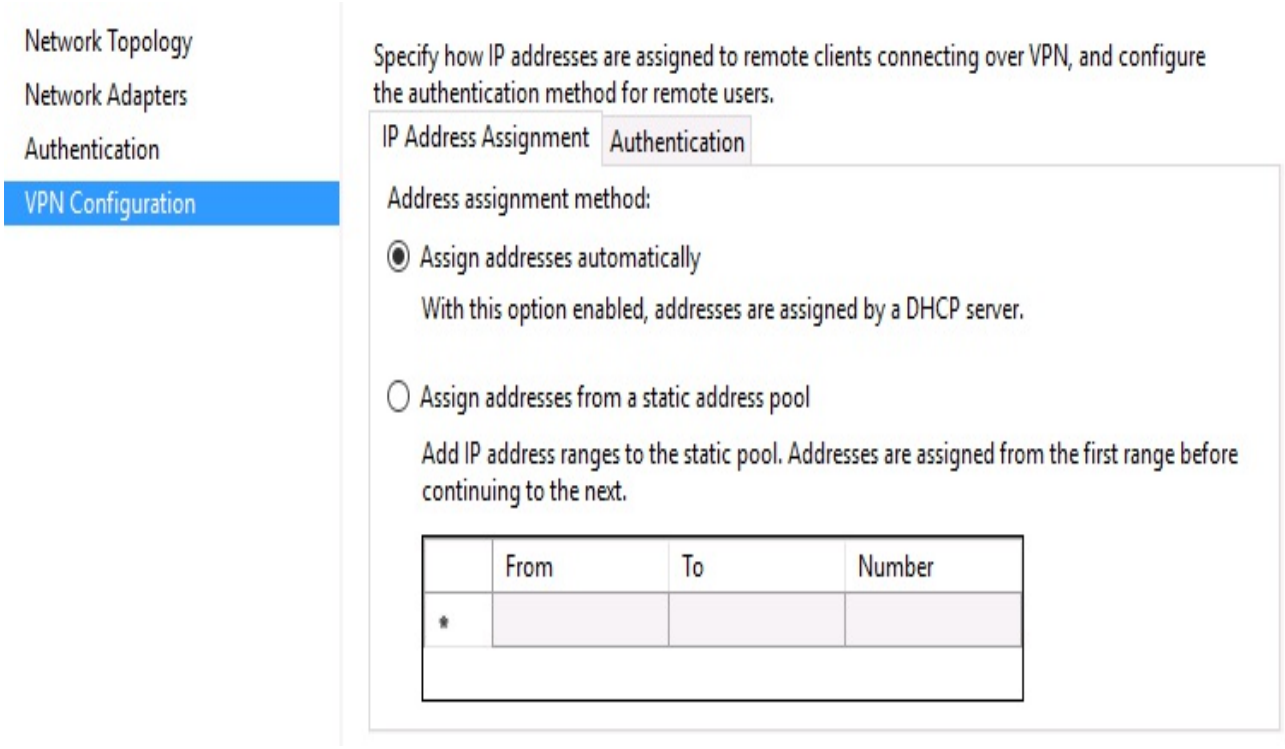

Remote Access Management Console
Configuration
DirectAccess and VPN
Web Application Proxy
Dashboard
Operations Status
Remote Client Status
Reporting
RA1
Remote Access Dashboard
Server Status
Operations Status
RA1.Contoso.local
DirectAccess
6to4
DNS
DNS64
Domain controller
IP-HTTPS
IPsec
Kerberos
NAT64
Network adapters
Network location server
Network security
Services
Teredo
VPN
VPN addressing
VPN connectivity
Operations Status page
Configuration Status
1/21/2019 8:04:56 AM
The configuration was distributed successfully.
DirectAccess and VPN Client Status
Total active clients: 0
Total active DirectAccess clien... 0
Total active VPN clients: 0
Total cumulative connections: 0
Total transferred data: 0 bytes in/0 bytes out
Maximum client connections: 0
Remote Client Status page
Tasks
Monitoring
Refresh
Configure Refresh Interval
Start Tracing
Generate Usage Report

Configuration

The configuration screen is pretty self-explanatory; this is where you create your initial remote access configuration, and where you update any settings in the future. As you can see in the screenshot, you are able to configure DirectAccess and VPN, and even the Web Application Proxy, right from this Remote Access Management Console.

> *Do not follow my lead with this screenshot. I have installed the DA/VPN portion of the Remote Access role alongside the Web Application Proxy portion of the same role, but it is not recommended to run both DA/VPN and WAP together on the same server. I did it simply for the purpose of creating screenshots in my test lab.*

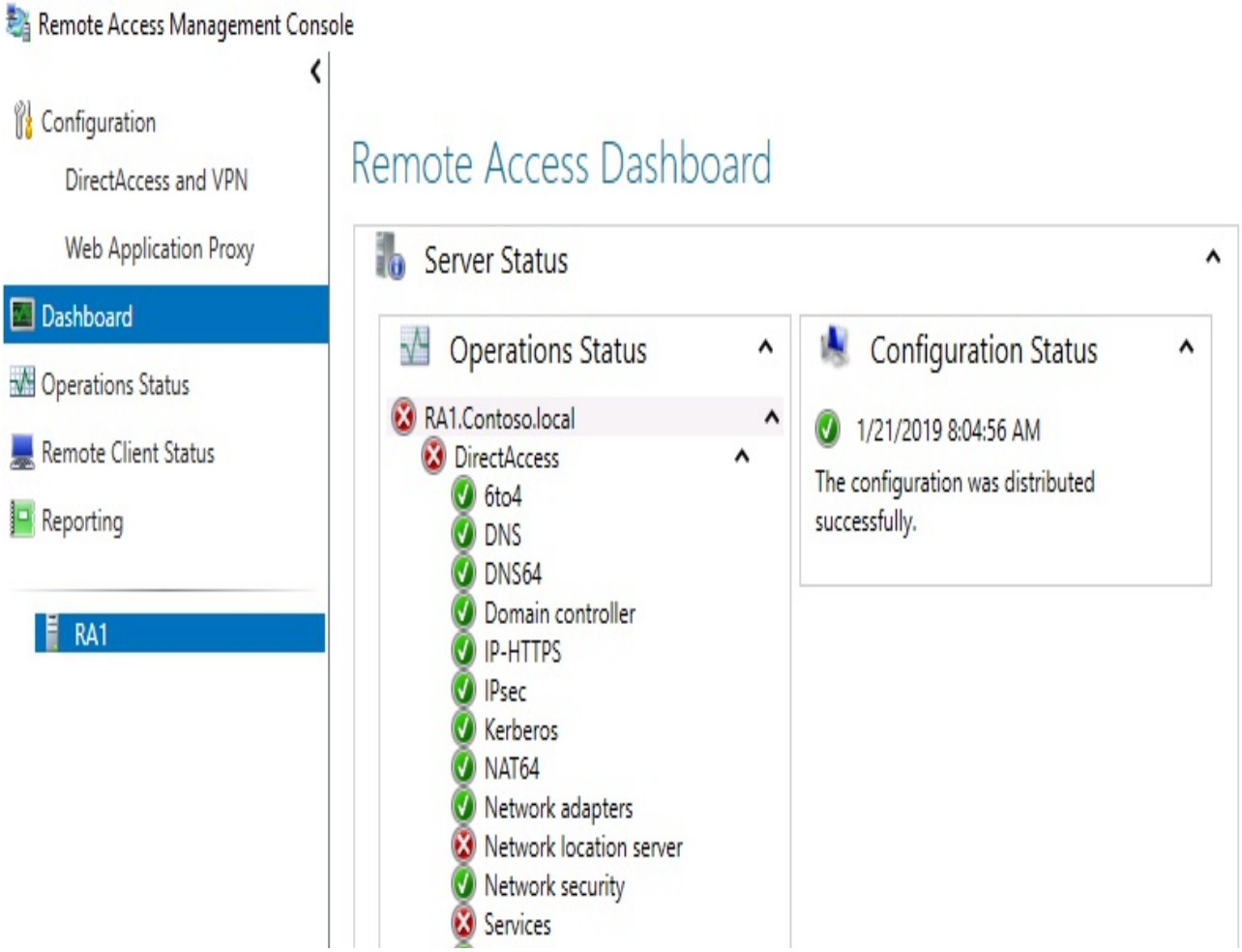
There is not a lot to configure as far as the VPN goes; you really only have one screen of options where you define what kinds of IP address are handed down to the VPN clients connecting in, and how to handle VPN authentication. It is not immediately obvious where this screen is, so I wanted to point it out. Inside the DirectAccess and VPN configuration section, if you click on Edit... under step 2, this will launch the step 2 mini-wizard. The last screen in this mini-wizard is called VPN Configuration. This is the screen where you can configure these IP address and authentication settings for your VPN connections. The remainder of your VPN configuration duties will fall within the traditional VPN configuration console, called RRAS. However, everything about DirectAccess connections is configured right from inside the **Remote Access Management Console** and those four mini-wizards:

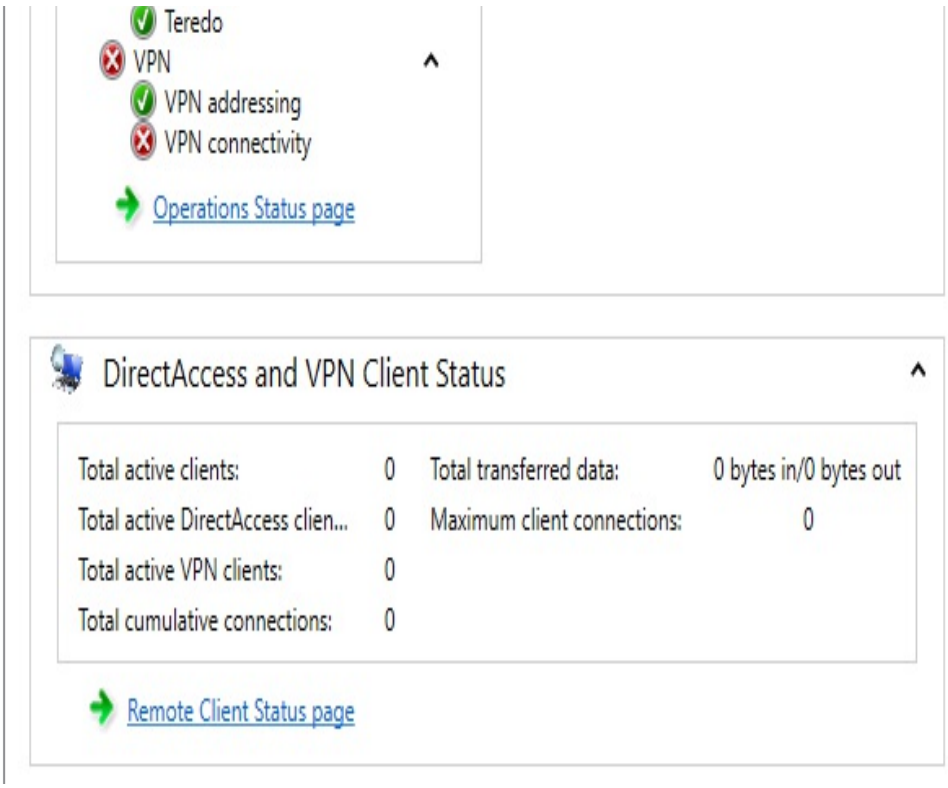


HivaNetwork.Com

Dashboard

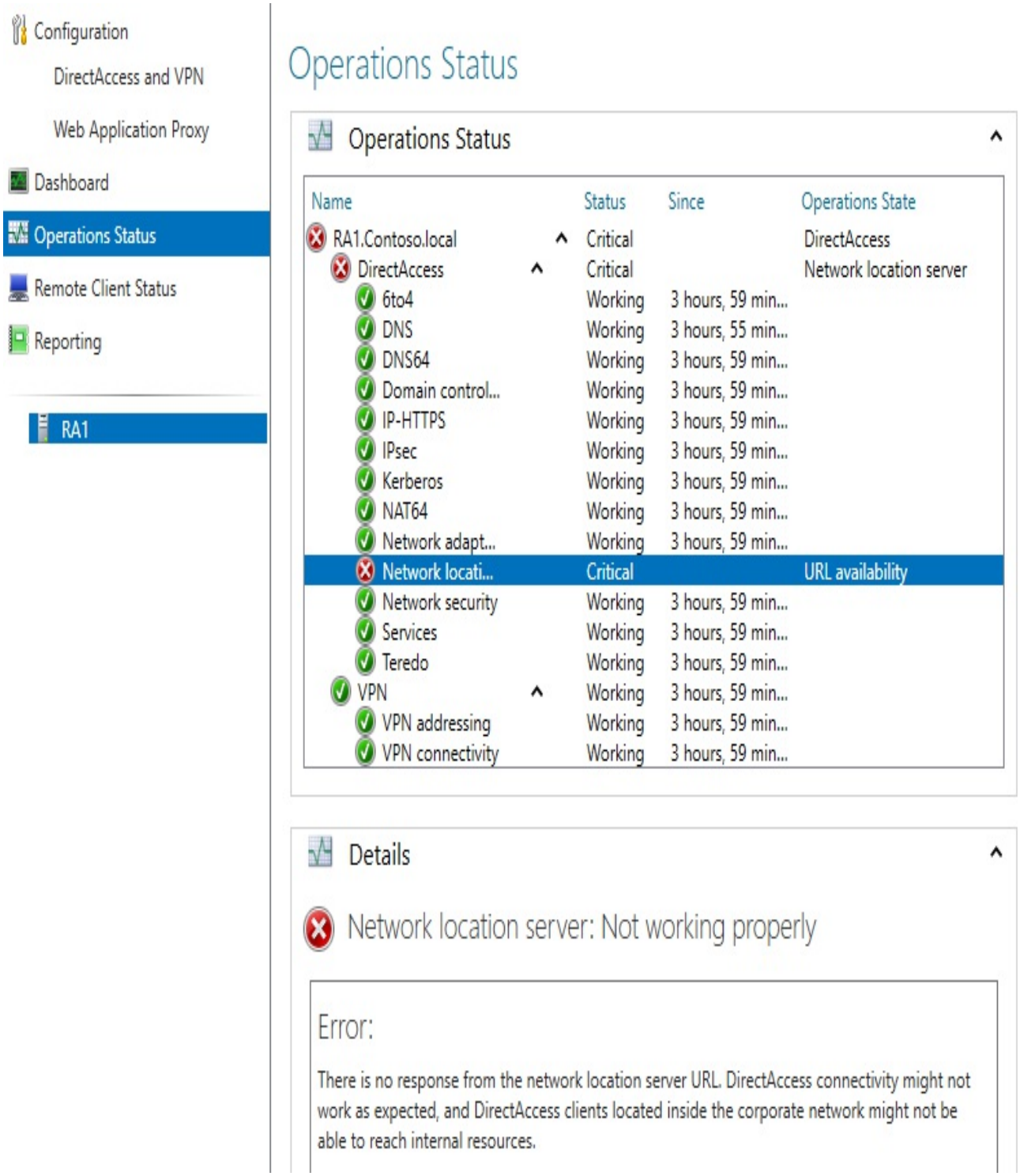
The Remote Access Dashboard gives you a 30,000-foot view of the Remote Access Server status. You are able to view a quick status of the components running on the server, whether or not the latest configuration changes have been rolled out, and some summary numbers near the bottom about how many DirectAccess and VPN connections are ongoing:




Teredo
VPN
VPN addressing
VPN connectivity
Operations Status page
DirectAccess and VPN Client Status
Total active clients: 0
Total active DirectAccess clien... 0
Total active VPN clients: 0
Total cumulative connections: 0
Total transferred data: 0 bytes in/0 bytes out
Maximum client connections: 0
Remote Client Status page

Operations Status

If you want to drill down further into what is happening on the server side of the connections, that is what the Operations Status page is all about. Here, you can see more detail on each of the components that are running under the hood to make your DA and VPN connections happen. If any of them have an issue, you can click on the specific component to get a little more information. For example, as a test, I have turned off the NLS web server in my lab network, and I can now see in the Operations Status page that NLS is flagged with an error:


Configuration
DirectAccess and VPN
Web Application Proxy
Dashboard
Operations Status
Remote Client Status
Reporting
RA1
Operations Status
Operations Status
Name
Status
Since
Operations State
RA1.Contoso.local
Critical
DirectAccess
DirectAccess
Critical
Network location server
6to4
Working
3 hours, 59 min...
DNS
Working
3 hours, 55 min...
DNS64
Working
3 hours, 59 min...
Domain control...
Working
3 hours, 59 min...
IP-HTTPS
Working
3 hours, 59 min...
IPsec
Working
3 hours, 59 min...
Kerberos
Working
3 hours, 59 min...
NAT64
Working
3 hours, 59 min...
Network adapt...
Working
3 hours, 59 min...
Network locati...
Critical
URL availability
Network security
Working
3 hours, 59 min...
Services
Working
3 hours, 59 min...
Teredo
Working
3 hours, 59 min...
VPN
Working
3 hours, 59 min...
VPN addressing
Working
3 hours, 59 min...
VPN connectivity
Working
3 hours, 59 min...
Details
Network location server: Not working properly
Error:
There is no response from the network location server URL. DirectAccess connectivity might not work as expected, and DirectAccess clients located inside the corporate network might not be able to reach internal resources.

Remote Client Status

Next up is the Remote Client Status screen. As indicated, this is the screen where we can monitor client computers that are connected. It will show us both DirectAccess and VPN connections here. We will be able to see computer names, usernames, and even the resources that they are utilizing during their connections. The information on this screen can be filtered by simply putting any criteria into the Search bar at the top of the window.

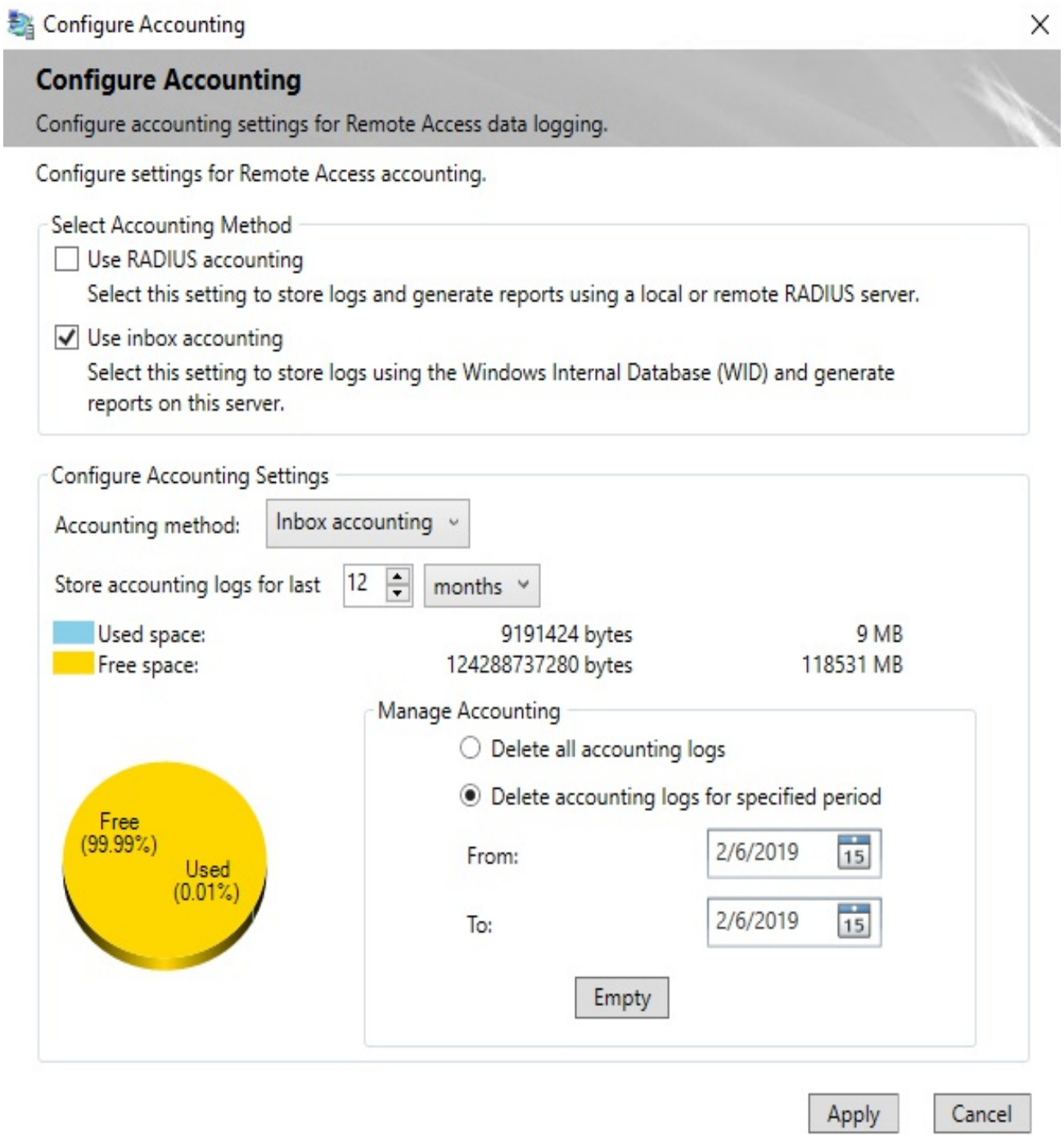
It is important to note that the Remote Client Status screen only shows live, active connections. There is no historical information stored here.

Reporting

You guessed it: this is the window you need to visit if you want to see historical remote access information. This screen is almost exactly the same as the Remote Client Status screen, except that you have the ability to generate reports for historical data pulled from date ranges of your choosing. Once the data is displayed, you have the same search and filtering capabilities that you had on the Remote Client Status screen.

Reporting is disabled by default, but you simply need to navigate to the Reporting page and click on Configure Accounting. Once that is enabled, you will be presented with options about storing the historical information. You can choose to store the data in the local WID, or on a remote RADIUS server.

You also have options here for how long to store logging data, and a mechanism that can be used to clear out old data:


Configure Accounting
Configure Accounting
Configure accounting settings for Remote Access data logging.
Configure settings for Remote Access accounting.
Select Accounting Method
Use RADIUS accounting
Select this setting to store logs and generate reports using a local or remote RADIUS server.
Use inbox accounting
Select this setting to store logs using the Windows Internal Database (WID) and generate reports on this server.
Configure Accounting Settings
Accounting method:
Inbox accounting
Store accounting logs for last
12
months
Used space:
9191424 bytes
9 MB
Free space:
124288737280 bytes
118531 MB
Free (99.99%)
Used (0.01%)
Manage Accounting
Delete all accounting logs
Delete accounting logs for specified period
From:
2/6/2019
To:
2/6/2019
Empty
Apply
Cancel

Tasks

The last window pane in the Remote Access Management Console that I want to point out is the Tasks bar on the right-hand side of your screen. The actions and options that are displayed in this taskbar change depending on what part of the console you are navigating through. Make sure you keep an eye on this side of your screen to set up some more advanced functions. Some examples of available tasks are creating usage reports, refreshing the screen, enabling or disabling VPNs, and configuring network load balancing or multisite configurations if you are running multiple Remote Access Server.

DA, VPN, or AOVPN? Which is best?

VPN has been around for a very long time, making it a pretty familiar idea to anyone working in IT. Always On VPN certainly brings its share of new capabilities, but under the hood what AOVPN is doing is launching a traditionally configured VPN connection, so the connection flow is similar to what we have always known. In this chapter, we have also discussed quite a bit about DirectAccess in order to bring you up to speed on this alternative method of automatically connecting your remote clients back to the data center. Now that you know there are two great connectivity platforms built into Windows Server 2019 for enabling your mobile workforce, which one is better?

You don't have to choose! You can actually run both of these technologies side-by-side, even on the same Remote Access Server. Each technology has its pros and cons, and the ways that you use each, or both, will depend upon many variables. Your users, your client computers, and your organization's individual needs will need to be factored into your decision making process. Let's discuss some of the differences between DirectAccess and VPN so that you can make intelligent decisions on which connectivity platforms fit into your organization well.

Domain-joined or not?

One of the biggest requirements for a DirectAccess client computer is that it must be domain-joined. While this requirement in and of itself doesn't seem so major, what it implies can have huge implications. Trusting a computer enough to be joined to your domain more than likely means that the laptop is owned by the company. It also probably means that this laptop was initially built and prepped by the IT team. Companies that are in the habit of allowing employees to purchase their own computers to be used for work purposes may not find that DirectAccess fist well with that model. DA is also not ideal for situations where employees use their existing home computers to connect into work remotely.

In these kinds of situation, such as home and personally-owned computers, VPN may be better suited to the task. You can connect to a VPN (including Always On VPN) from a non-domain-joined Windows 10 machine, and you can even establish VPN connections (manual connections) from many non-Microsoft devices. iOS, Android, Windows phones, and the Mac—all these platforms have a VPN client built into them that can be used to tap into a VPN listener on a Windows Server 2019 Remote Access Server. If your only remote access solution were DirectAccess, you would not be able to provide non-domain-joined devices with a connectivity platform.

Keep in mind that, while the Always On VPN User Tunnel is more flexible than DirectAccess in this regard, if you intend to use the AOVPN *Device Tunnel*, then your machines will still need to be domain-joined.

Auto or manual launch

There are multiple different ways to look at this one. When discussing whether DirectAccess or a traditional VPN is better, DirectAccess is the clear winner. Nobody wants to make their users open a connection and manually launch it in order to establish VPN connectivity when an automated platform is available to use.

Always On VPN, however, brings automated and seamless connectivity to the VPN world. AOVPN is almost as seamless as DirectAccess in this regard. I say *almost* because, at the time of writing, it's fairly difficult to make a Device Tunnel work well. This means that most companies rolling out AOVPN are only using the User Tunnel. In the User Tunnel scenario, the VPN does launch automatically, but not until the user has already passed the login screen. This means that, in these situations, DirectAccess still holds an advantage over AOVPN, because DA connects seamlessly at the login screen. This enables password resets and new domain users to log into DA-connected machines. My hope is that future improvements will enable AOVPN devices and User Tunnels to co-exist well, which will give true always-on connectivity to AOVPN clients.

Software versus built-in

I'm a fan of Ikea furniture. They do a great job of supplying quality products at a low cost, including packaging it up in incredibly small boxes. After you pay for the product, unbox it, put it together, and then test it to make sure it works—it's great. If you can't see where this is going, I'll give you a hint: it's an analogy for traditional, third-party VPNs. As in, you typically pay a vendor for their VPN product, unbox it, implement it at more expense, then test the product. That VPN software then has the potential to break and need reinstallation or reconfiguration, and will certainly come with software updates that need to be accomplished down the road. Maintenance, maintenance, maintenance.

Maybe I have been watching too many home improvement shows lately, but I am a fan of houses with built-ins. Built-ins are essentially furniture that is permanent to the house, built right into the walls, corners, or wherever it happens to be. It adds value, and it integrates into the overall house much better than furniture that was pieced together separately and then stuck against the wall in the corner.

DirectAccess and Always On VPN are like built-ins. They live inside the operating system. There is no software to install, no software to update, no software to reinstall when it breaks. Everything that DA and AOVPN need is already in Windows today, you just aren't using it. Oh, and it's free! Well, built into the cost of your Windows license anyway. There are no user CALs, and no ongoing licensing costs related to implementing one of Microsoft's remote access solutions.

If your workforce consists of Windows 10 machines, Microsoft DirectAccess or Microsoft Always On VPNs are clear winners when compared to any third-party VPN connectivity solution.

Password and login issues with traditional VPNs

If you have ever worked on the helpdesk for a company that uses a VPN, you know what I'm talking about. There are a series of common troubleshooting calls that happen in the VPN world related to passwords. Sometimes, the user forgets their password. Perhaps their password has expired and needs to be changed—ugh! VPN doesn't handle this scenario very well either. Or perhaps the employee changed their expired password on their desktop before they left work for the day, but are now trying to log in remotely from their laptop and it isn't working.

What is the solution to password problems with VPN? Reset the user's password and then make the user come into the office in order to make it work on their laptop. Yup, these kinds of phone call still happen every day. This is unfortunate, but a real potential problem with old-school VPNs.

What's the good news? New Microsoft remote access solutions don't have these kinds of problem! Since DA and AOVPN are part of the operating system, they have the capability to be connected anytime that Windows is online. This includes the login screen! Even if I am sitting on the login or lock screen, and the system is waiting for me to input my username and password, as long as I have internet access I also have a DirectAccess tunnel, or an Always On VPN Device Tunnel. This means that I can actively do password management tasks. If my password expires and I need to update it, it works. If I forgot my password and I

can't get into my laptop, I can call the helpdesk and simply ask them to reset my password. I can then immediately log in to my DA or AOVPN laptop with the new password, right from my house.

Another cool function that this seamlessness enables is the ability to log in with new user accounts. Have you ever logged into your laptop as a different user account in order to test something? Yup, that works over DA and AOVPN as well. For example, I am sitting at home and I need to help one of the sales guys troubleshoot some sort of file permission problem. I suspect it's got something to do with his user account, so I want to log in to my laptop as him in order to test it. The problem is that his user account has never logged into my laptop before. With VPN, not a chance; this would never work. With DirectAccess, piece of cake! I simply log off, type in his username and password, and bingo. I'm logged in, while still sitting at home in my pajamas.

> *It is important to note that you can run both DirectAccess and VPN connections on the same Windows Server 2019 Remote Access Server. This enables you to host clients who are connected via DA, via AOVPN, and also via traditional VPN connections if you have non-Win10 machines that need to connect. If any of these connectivity technologies have capabilities that you could benefit from, use them all!*

Port-restricted firewalls

One of the other common VPN-related helpdesk calls has always been *My VPN won't connect from this hotel*. Unfortunately, most protocols that VPNs use to connect are not firewall-friendly. Chances are that your router at home allows all outbound traffic, and so from your home internet connection everything is fine and dandy when connecting with a VPN protocol. But take that same laptop and connection over to a public coffee shop, or a hotel, or an airport, and suddenly the VPN fails to connect, with a strange error. This is usually caused by that public internet connection flowing its traffic through a port-restricting firewall. These firewalls restrict outbound access, oftentimes blocking things such as ICMP and UDP, which can interfere with VPN connections. In the most severe cases, these firewalls may only allow two outbound ports: TCP `80` for HTTP and TCP `443` for HTTPS website traffic. Then they block everything else.

In the event that you are sitting behind a port-restricted firewall, how do these newer remote access technologies handle connectivity?

DirectAccess is built to handle this scenario out of the box. Remember those three different protocols that DA can use to connect? The **fallback** option is called IP-HTTPS, and it flows its traffic inside TCP `443`. So, even while sitting behind the most severe firewalls, DA will generally connect automatically and without hesitation.

Always On VPN is generally deployed (as it should be) with best-practices in mind, which includes prioritizing IKEv2 as the VPN connectivity protocol. In fact, some companies deploy AOVPN with only

IKEv2. For these folks, a port-restricting firewall would be detrimental to that user's VPN connection, as IKEv2 uses UDP ports to connect. It wouldn't work. So, hopefully, the main point you take from this is that, when setting up AOVPN, make sure that you take the necessary steps to also enable SSTP VPN connectivity as a fallback method. SSTP also flows traffic inside TCP `443`, which can then get outbound traffic, even through hardcore firewalls.

> **Super important***:*
>
> *The AOVPN Device Tunnel can only use IKEv2. If you are behind a port-restricting firewall and are relying on a Device Tunnel for connectivity, it's not going to work. The AOVPN User Tunnel is the only one capable of doing SSTP fallback.*

In fact, I recently worked on this exact scenario with someone who was trying to decide whether they wanted to set up DirectAccess or Always On VPN for their remote machines. This was a company that manages computers for numerous hospitals and doctors' offices, and they did not have WAN links into those offices. The offices did have internet access though, and so we needed the ability to keep those computers automatically connected back to the main data center at all times. So far in the scenario, either DirectAccess or Always On VPN would fit the bill. Then, during testing, we discovered that many hospital networks restrict outbound internet access. The only way that DA would connect was via IP-HTTPS, and the only way that AOVPN would connect was via SSTP. Not a problem, right? Except that it was. You see, these remote workstations are often treated as kiosk, walk-up machines, where dozens of different employees could walk up at any moment and log into them. Oftentimes, this means that users are logging into these machines who have never logged into them before, so they don't have cached credentials on those computers.

If you haven't figured it out already, we had no choice but to go with DirectAccess in this scenario. DA is always connected at the login screen, even when using its "fallback" IP-HTTPS method. Always On

VPN, however, can only do IKEv2 at the logon screen, because the Device Tunnel requires IKEv2. This uses UDP and was blocked by the firewall, so the only way that AOVPN would connect was by using SSTP, but that wasn't available until the User Tunnel could launch, which was only after the user had logged into the machine. It was an extremely interesting real-world use case that helped shed some light on the decision-making process you may need to take for your own environments.

Manual disconnect

If you aren't already convinced that old-school, traditional VPNs are yesterday's news, let's throw another point at you. When you use VPNs that require the user to manually launch the connection, you are relying on the user themselves to keep that machine managed, patched, and up to date. Sure, you may have automated systems that do these things for you, such as WSUS, SCCM, and Group Policy. But when the laptop is out and about, roaming around away from the LAN, those management systems can only do their jobs when the user decides to establish a VPN connection. It's very possible that a laptop could spend weeks completely outside the corporate network, connecting to dozens of insecure hotspots while that employee works their way around the Caribbean on a cruise ship. After weeks of partying and Netflix, they then connect back into the LAN or via VPN to do some work, and wouldn't you know it, that machine has been offline for so long that it's picked up all kinds of fun and creative new software that you now have to deal with.

Not so with the Microsoft remote access tools! Providing an automatic connectivity option such as Always On VPN or DirectAccess means that the laptop would have been connected and receiving all of its security policies and patches during that entire vacation.

In fact, to take it one step further, on a DirectAccess-connected machine, the user cannot disable their DA tunnels even if they want to. You do have the ability to provide them with a **Disconnect** button, but that basically just fakes out the connection from the user's perspective to make it feel to them like DA is offline. In reality, the IPsec tunnels

are still flowing in the background, always allowing management-style tasks to happen.

Native load-balancing capabilities

To cut a long story short, DirectAccess is the winner here. The Remote Access Management Console in Windows Server 2019 has built-in capabilities for configuring and managing arrays of DA servers. You can stack multiple DA servers on top of each other, tie them together in load-balanced arrays, and provide redundancy and resiliency right from inside the console, without any extra hardware or traditional load-balancer consideration. You can even configure something called **DirectAccess multisite**, where you can configure DirectAccess servers that reside in different geographical locations together in arrays, giving cross-site resiliency. Almost every company that runs DirectAccess configures a redundant environment, providing either inner-site load balancing or multisite, or sometimes both, because these capabilities are built in and easy to configure.

Unfortunately, these capabilities are not (not yet, anyway) ported over into the Microsoft VPN world. Whether you are connecting Windows 7 clients via traditional VPN connectivity or getting Windows 10 clients to connect using Always On VPN, the backend infrastructure of RRAS VPN is the same, and has no built-in accommodation for multiple servers or sites. It is certainly possible to do so, making that VPN system redundant, but that would require you to set it up on your own by using external load balancers and, oftentimes, would require the use of global site/server load balancers to make that traffic flow properly.

Anyone who has set up load-balanced VPNs of any flavor in the past may be well aware of this process and be able to configure that easily, and that is great. But this is definitely a limiting factor for small

business customers who have a limited number of servers, network equipment, and IT experience. All in all, the extra capabilities built into the console related to DirectAccess put it a step ahead of any VPN solution in terms of building up your remote access infrastructure to be resilient to failure.

Distribution of client configurations

The last primary consideration that you need to take into account when deciding which direction you want to go in for remote access is the method by which client-side settings get pushed down to their respective computers.

- **Third-party VPN**: We have already discussed the downsides to dealing with software applications for third-party VPN vendors. If you can use something baked into the Windows operating system instead, that seems like a no-brainer.
- **Always On VPN**: The recommended way to roll out AOVPN settings to client computers is through the use of an MDM solution, namely either SCCM or Intune. If you have one of these systems, rolling out AOVPN settings to your workforce is a breeze. If you do not have one of those systems, it is still possible, but not a straightforward process.
- **DirectAccess**: I think DA's approach to client settings distribution is definitely the easiest to work with, and the most flexible. You have to keep in mind that DirectAccess is only for your domain-joined systems. Given that you can expect all clients to be domain-joined, you have access to rolling

> DirectAccess connectivity settings out via Group Policy, which exists inside any Microsoft-driven infrastructure.

I genuinely hope that we will see a Group Policy distribution option added in the future for Always On VPN configuration rollouts. If such a capability were introduced, I am completely confident that it would immediately become the most popular way to roll out AOVPN settings.

To summarize this whole topic, when comparing DirectAccess against traditional, manual-launch VPNs, DA cleanly takes first prize. There really is no comparison. Now that we have Always On VPN at our disposal, the benefits of one over the other (DA or AOVPN) are quite fuzzy. They both accomplish a lot of the same things, but in different ways. The primary deciding factors for most customers so far seem to be client-side rollout capabilities, whether or not they have access to an MDM solution, and how important Device Tunnel connectivity is to them. Microsoft's goal is for AOVPN to have feature parity with DirectAccess, and it's getting close. Always On VPN also has some advanced authentication features that DirectAccess does not have, such as integration with Windows Hello for Business or Azure MFA.

Web Application Proxy

DirectAccess and VPN are both great remote access technologies, and combining the two of them together can provide a complete remote access solution for your organization, without having to pay for or work with a third-party solution. Better still, in Windows Server 2019, there is yet another component of the Remote Access role available to use. This third piece of the remote access story is the **Web Application Proxy** (**WAP**). This is essentially a reverse-proxy mechanism, giving you the ability to take some HTTP and HTTPS applications that are hosted inside your corporate network, and publish them securely to the internet. Any of you who have been working with Microsoft technologies in the perimeter networking space over the last few years will probably recognize a product called Forefront **Unified Access Gateway** (**UAG**), which accomplished similar functionality. UAG was a comprehensive SSLVPN solution, also designed for publishing internal applications on the internet via SSL. It was considerably more powerful than a simple reverse-proxy, including components such as pre-authentication, SSTP VPN, and RDS gateway; DirectAccess itself could even be run through UAG.

If all of your mobile workers have access to launching either DirectAccess or VPN, then you probably don't have any use for WAP. However, with the growing cloud mentality, it is quite common for users to expect that they can open up a web browser from any computer, anywhere, and gain access to some of their applications. Document access is now often provided by web services such as SharePoint. Email access can be had remotely, from any computer, by tapping into Outlook Web Access.

So many applications and so much data can be accessed through only a web browser, and this enables employees to access this data without needing to establish a full-blown corporate tunnel such as a VPN. So what is the real-world use case for WAP? Home computers that you do not want to be VPN-connected. This way, you don't have to worry as much about the health and status of the home or user-owned computers, since the only interaction they have with your company is through the web browser. This limits the potential for sinister activity to flow into your network from these computers. As you can see, a technology such as WAP does certainly have its place in the remote access market.

I have hopes that, over time, WAP will continue to be improved, and that will enable it to be a true replacement for UAG. UAG ran on Windows Server 2008 R2, and has now been officially discontinued as a Microsoft product. The closest solution Microsoft now has to UAG is the WAP role. It is not yet nearly as comprehensive, but they are working on improving it. Currently, WAP is useful for publishing access to simple web applications. You can also publish access to rich clients that use basic HTTP authentication, such as Exchange ActiveSync. Also included is the ability to publish data to clients that use MSOFBA, such as when users try to pull down corporate data from their Word or Excel applications running on the local computer.

WAP can be used to reverse-proxy (publish) remote access to things such as Exchange and SharePoint environments. This is no small thing, as these are technologies that almost everyone uses, so it can certainly be beneficial to your company to implement WAP for publishing secure access to these resources; it's certainly better than NATing directly to your Exchange server.

WAP as AD FS Proxy

Another useful way in which you can utilize a WAP server is when setting up **Active Directory Federation Services** (**AD FS**) in your network (this is perhaps the most common use for WAP right now). AD FS is a technology designed to enable single sign-on for users and federation with other companies, and so it involves taking traffic coming in from the internet into your internal network. In the past, there was a Windows Server role component that accompanied AD FS, called the **AD FS Proxy**. In the latest versions of AD FS, this separate role no longer exists, and has been replaced by the Web Application Proxy component of the Remote Access role. This better unifies the remote access solution, bringing your inbound AD FS traffic through the official Remote Access Server, rather than needing a separate AD FS Proxy server. Anyone implementing outward-facing AD FS in their environments will likely find themselves required to deploy WAP at some point.

Requirements for WAP

Unfortunately, the ability to make use of the Web Application Proxy comes with a pretty awkward requirement: you must have AD FS installed in your environment to be able to use i—even to test it, because the WAP configuration is stored inside AD FS. None of the WAP configuration information is stored on the Remote Access Server itself, which makes for a lightweight server that can be easily moved, changed, or added to. The downside to this is that you must have AD FS running in your environment so that WAP can have a place to store that configuration information.

While a tight integration with AD FS does mean that we have better authentication options, and users can take advantage of AD FS single-sign-on to their applications that are published through WAP, so far this has proven to be a roadblock to implementation for smaller businesses. Many folks are not yet running AD FS, and if the only reason they are looking into implementing AD FS is so that they can use WAP to publish a few web applications to the internet, they may not choose to invest the time and effort just to make that happen.

One thing to keep in mind if you are interested in using WAP, and are therefore looking at the requirement for AD FS, is that AD FS can certainly be used for other functions. In fact, one of its most common uses at present is integration with Office 365. If you are planning to incorporate, or thinking of incorporating, Office 365 into your environment, AD FS is a great tool that can enhance authentication capabilities for that traffic.

Latest improvements to WAP

Web Application Proxy was introduced in Server 2012 R2, and had many improvements when Windows Server 2016 was released. There have been no major modifications since that time, but it is still important to point out the latest benefits that have been rolled into this feature, to show that it is still learning to do new things. The following are some of the improvements that have been made if you haven't taken a look at WAP since its first iteration.

Preauthentication for HTTP Basic

There are two different ways that users can authenticate to applications that are being published by Web Application Proxy—preauthentication or pass-thru authentication. When publishing an application with preauthentication, this means that users will have to stop by the AD FS interface to authenticate themselves before they are allowed through to the web application itself. In my eyes, preauthentication is a critical component to any reverse-proxy, and I would have to be stuck between a rock and a hard place in order to externally publish an application that did not require preauthentication. However, the second option is to do pass-thru authentication, and it does exactly that. When you publish access to an application and choose pass-thru authentication, all WAP is doing is shuttling the packets from the internet to the application server. Users are able to get to the web application without authentication, so in theory, anyone can hit the front website of your application. From there, the application itself will likely require the user to authenticate, but there is no man-in-the-middle protection happening; that web application is available for the public to view. As you can tell, I do not recommend taking this route.

We already know that WAP can preauthenticate web applications, but the original version could not do any form of preauthentication on HTTP Basic applications, such as when a company wanted to publish access to Exchange ActiveSync. This inability leaves ActiveSync a little bit too exposed to the outside world, and is a security risk. Thankfully, this changed in Windows Server 2016—you can now preauthenticate traffic streams that are using HTTP Basic.

HTTP to HTTPS redirection

Users don't like going out of their way or wasting time by having to remember that they need to enter `HTTPS://` in front of the URL when they access applications. They would rather just remember `email.contoso.com`. The inability of WAP to do HTTP to HTTPS redirection had been an annoyance and a hindrance to adoption, but that has since changed. Web Application Proxy now includes the capability for WAP itself to handle HTTP to HTTPS redirection, meaning that users do not need to type `HTTPS` into their browser address bar any longer; they can simply type in the DNS name of the site and let WAP handle the translations.

Client IP addresses forwarded to applications

In the reverse-proxy and SSLVPN world, we occasionally run across applications that require knowing what the client's local IP address is. While this requirement doesn't happen very often, and is typically segregated to what we would call legacy applications, it does still happen. When the backend application needs to know what the client's IP address is, this presents a big challenge with reverse-proxy solutions. When the user's traffic flows through WAP or any other reverse-proxy, it is similar to a NAT, where the source IP address information in these packets changes. The backend application server is unable to determine the client's own IP address, and trouble ensues. Web Application Proxy now has the ability to propagate the client-side IP address through to the backend application server, alleviating this problem.

Publishing Remote Desktop Gateway

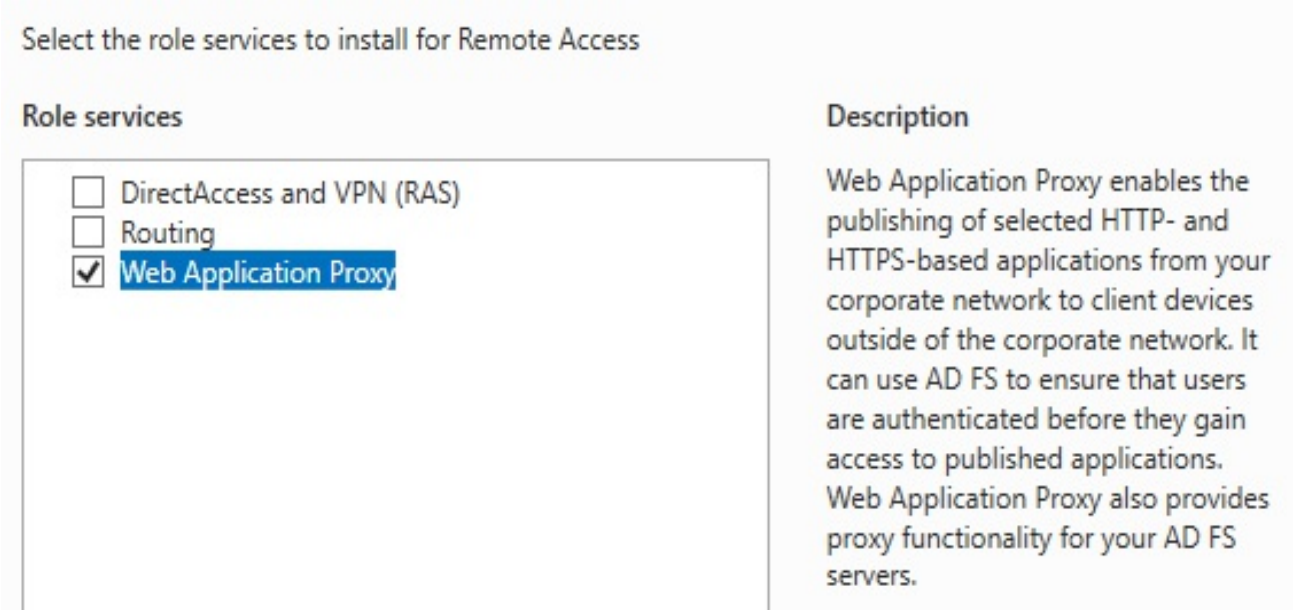
One of the items that UAG was commonly used for was publishing access to Remote Desktop Services. UAG was essentially its own Remote Desktop Gateway, which gave you the ability to publish access to RDSH servers, individual RDP connections to desktop computers, such as in a VDI implementation, and even access to RemoteApp applications. Unfortunately, WAP cannot do any of this, even in the new version, but the fact that they have added a little bit of functionality here means movement in the right direction is happening.

What has been improved regarding WAP and Remote Desktop is that you can now use WAP to publish access to the Remote Desktop Gateway server itself. Traditionally, a Remote Desktop Gateway sits on the edge of the network and connects external users through to internal Remote Desktop servers. Placing WAP in front of the RD Gateway allows stronger preauthentication for Remote Desktop services, and creates a bigger separation between the internal and external networks.

All of my fingers are crossed that we will continue to see improvements in this area, and that WAP can be expanded to handle traffic like Remote Desktop natively, without even needing a Remote Desktop Gateway in the mix.

Improved administrative console

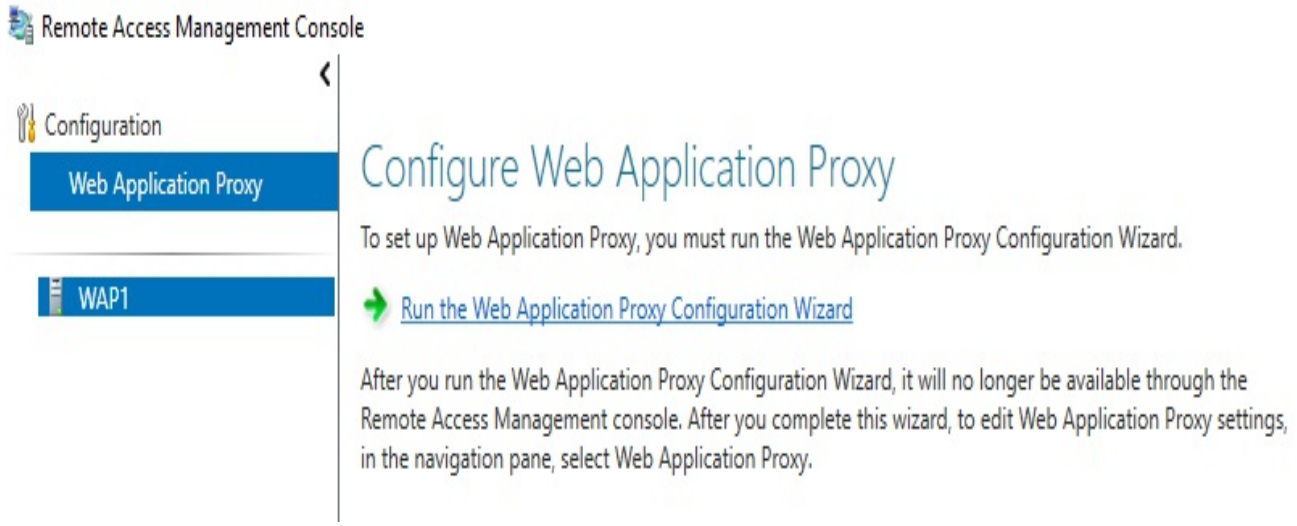
The original version of WAP inside Windows Server 2012 R2 was best served using PowerShell to implement it. You can certainly still use PowerShell to create your publishing rules if you so choose, but the Remote Access Management Console has now been improved in terms of how it relates to the Web Application Proxy. Before you see it in the console, you need to make sure that the appropriate box was checked during the Remote Access role installation. If you did not select Web Application Proxy when you first installed that role, revisit the add/remove Roles function inside Server Manager in order to add WAP to this server:



Note that, while Web Application Proxy is a component of the same Remote Access role that houses DirectAccess and VPN, it is not recommended to run WAP alongside DA and VPN on the same server. As you already know, you can certainly co-host DA and VPN connections together, simultaneously on a single Remote

Access Server. But once you make the foray into WAP, this should be a standalone component. Do not run WAP on a DA/VPN server, and do not run DA/VPN on a WAP server.

Now that we have added Web Application Proxy to our server, you can open up the Remote Access Management Console and see it listed inside the Configuration section. From here, you launch the **Web Application Proxy Configuration Wizard** and start walking through the steps to define your AD FS server, the certificates you are planning to use, and other criteria needed for the role:



HivaNetwork.Com

Summary

The technology of today demands that most companies enable their employees to work from wherever they are. More and more organizations are hiring a *work from home* workforce, and need a secure, stable, and efficient way to provide access to corporate data and applications for these mobile workers. The Remote Access role in Windows Server 2019 is designed to do exactly that. With three different ways of providing remote access to corporate resources, IT departments have never had so much remote access technology available at their fingertips, built right into the Windows operating system that they already own. If you are still supporting a third-party or legacy VPN system, you should definitely explore the new capabilities provided here and discover how much they could save your business.

DirectAccess and Always On VPN are particularly impressive and compelling connectivity options—a fresh way of looking at remote access. Automatic connectivity includes always-on machines that are constantly being patched and updated because they are always connected to your management servers. You can improve user productivity and network security at the same time. These two things are usually oxymorons in the IT world, but with the Microsoft remote access stack, they hold hands and sing songs together.

Next, we are going to take a look at some of the security functions built into your Windows Server 2019 operating systems, and at some of the ways that your servers can be hardened to provide even better security than what comes out of the box.

Questions

1. What does AOVPN stand for?
2. What are the two primary protocols used for connecting AOVPN clients?
3. In which version of Windows 10 was AOVPN released?
4. In what special instance would an AOVPN client be required to be joined to your domain?
5. Does DirectAccess require your corporate internal network to be running IPv6?
6. What is the name of the internal website that DirectAccess clients check-in with in order to determine when they are inside the corporate network?
7. What role does a Web Application Proxy server hold in a federation environment?

Hardening and Security

$3.8 *million* dollars. For anyone who read that in the voice of Dr. Evil, my hat goes off to you. For anyone who has no idea what I'm talking about, you may have had a sheltered childhood. Joking aside, that number is significant to IT security. Why? Because $3.8 million dollars is the average cost to a business when they are the victim of a data breach. I originally heard this and other scary statistics at a Microsoft conference in Redmond a couple of years ago, and the numbers have continued to climb year by year. How about looking at another statistic that can be used in order to get approval for an increase in your security budget? Depending on which study you read, the average number of days an attacker has **dwell time** in your network (the time they spend hanging around inside your files and infrastructure before they are detected and eradicated) is around 200. Think about that—200 days! That is the better part of a year that they're camping out for before you discover them! What are they typically doing during those 200 days? Siphoning all of your data, bit by bit out the back door. Another number is 76%—as in the percentage of network intrusions that happen as a result of compromised user credentials. Furthermore, it is becoming more and more difficult to identify these attacks in the first place, because attackers are using legitimate IT tools in order to grab what they want, such as socially engineering their way into the trust of a single employee, and leveraging that trust in order to get a remote access connectivity tool installed onto the user's work computer. Why use malware when you can use something that is *trusted* and is going to fly under the radar of intrusion detection systems? Makes sense to me.

Data security, network security, credential security—these things are all

becoming harder to accomplish, but there are always new tools and technology coming out that can help you fight off the bad guys. Windows Server 2019 is the most secure OS that Microsoft has produced; in this chapter, let's discuss some of the functionality included that makes that statement true:

- Windows Defender Advanced Threat Protection
- Windows Defender Firewall – no laughing matter
- Encryption technologies
- Banned passwords
- Advanced Threat Analytics
- General security best practices

Windows Defender Advanced Threat Protection

Windows Defender has been a thing for a number of years, but its terminology and capabilities have really developed over the last couple of OS releases. Originally, it started in the Windows 8 days as a free, built-in antivirus product, and it wasn't taken too seriously at the time. Fast forward to today, however, and I rarely run across a Windows 10 computer that has the Defender **Antivirus** (**AV**) or firewall capabilities disabled. These tools exist in the OS and are enabled by default, and as a result have a level of integration and responsiveness that is hard for third-party vendors to match. I can't tell you how many times I have tracked memory leaks and random server reboots back to a poorly functioning third-party antivirus software, which is unacceptable in today's server world. Some still consider the antivirus capabilities provided by Defender to be lackluster, probably only because they are free, but I find it to be robust and well integrated with Windows itself. I have yet to see a Windows Defender product tank a client or server.

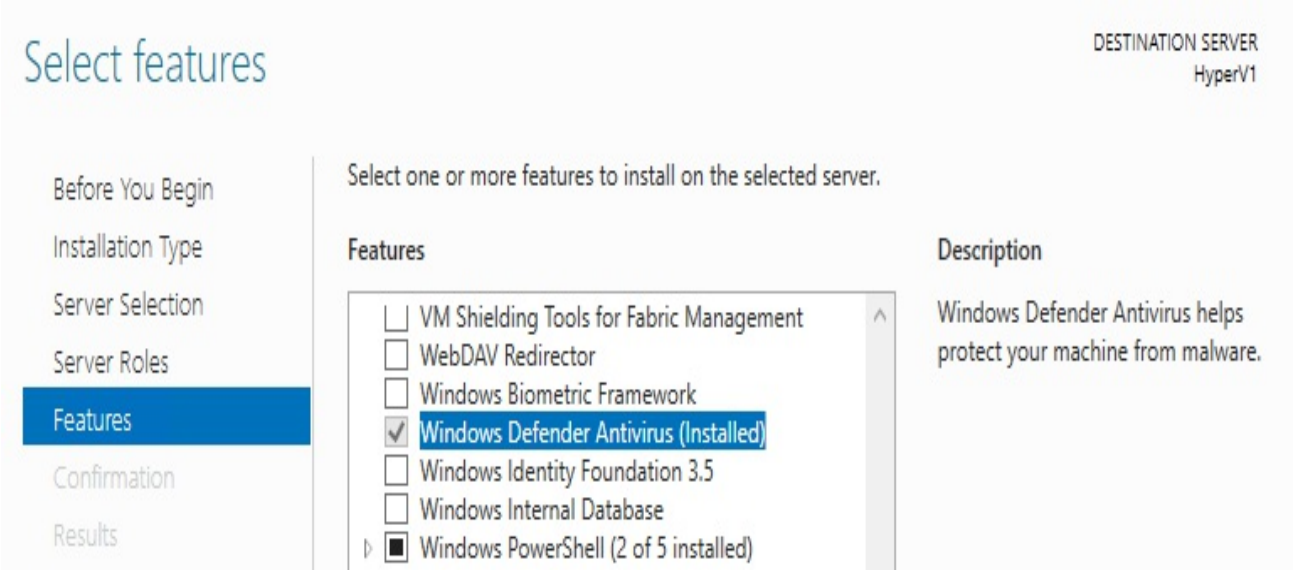
Even the newer, more specific-sounding Windows Defender **Advanced Threat Protection** (**ATP**) is really a family of products and systems that work together in order to protect your Windows machines. Antivirus/anti-malware is only one of those capabilities, and built-in antivirus is actually still quite a new idea when talking about the Windows Server family of OSes. The first server OS that we found with built-in Defender for antivirus was Server 2016. I suspect that the majority of servers running in production for companies around the world are still Server 2012 R2 at this point, and so the improved

existence of the Defender toolset in Server 2019 is yet another reason to start planning your migration today.

We simply do not have enough page space to dive into every aspect of Windows Defender ATP, and it is being continually improved upon. What we will do is explore some of the interfaces, make sure you know how to use the most common components that don't require policy-level manipulation, and to expand your knowledge on some of the more advanced features that are available for further learning and digging.

Installing Windows Defender AV

You're done! Windows Defender is installed by default in Windows Server 2019. In fact, unless you have somehow changed it, not only is Defender AV installed, it is automatically protecting your system as soon as the OS is installed. But don't take my word for it, if you open up **Server Manager** and choose Add roles and features, click ahead to the Select features page and you should find a checkbox next to Windows Defender Antivirus:

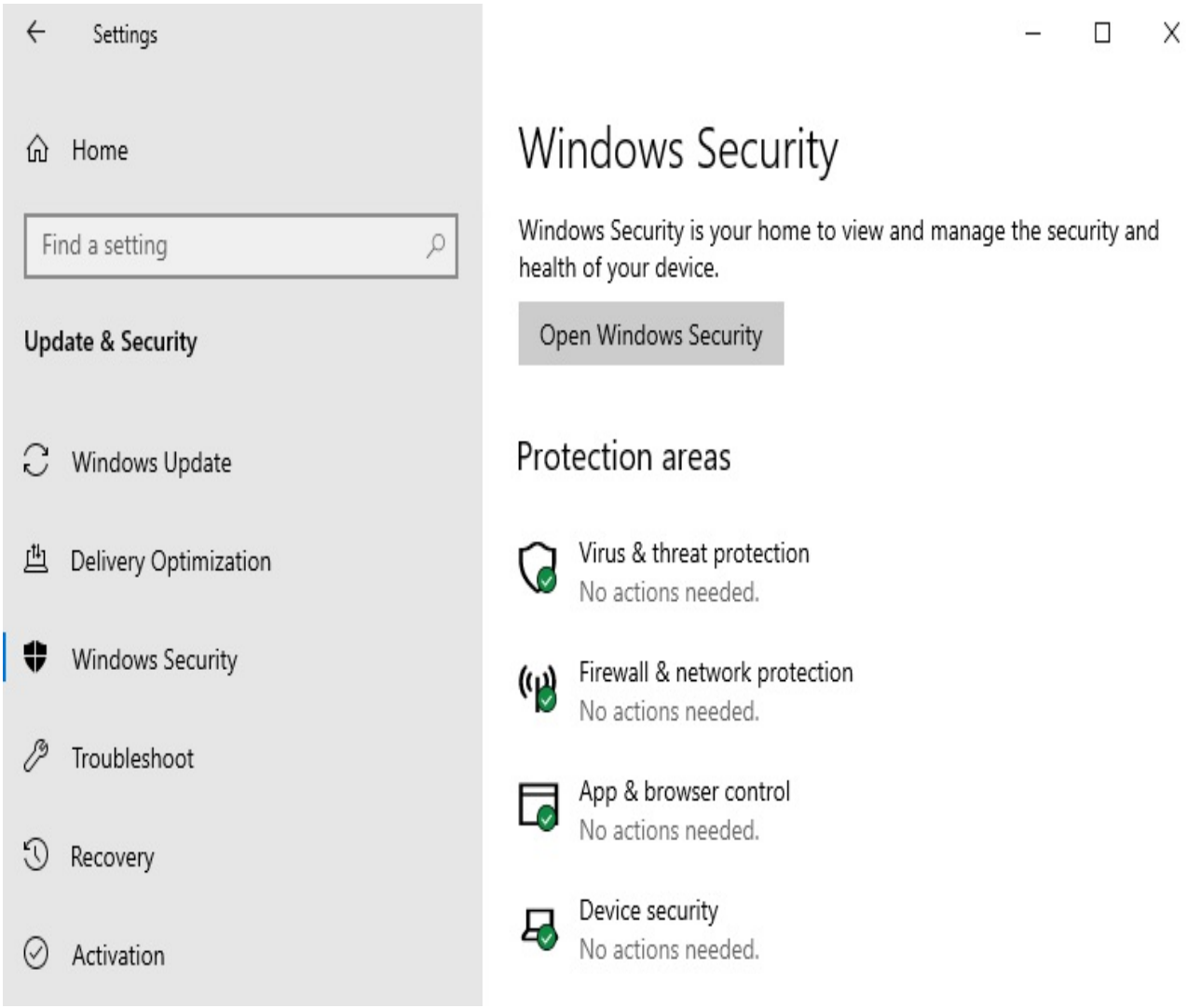


If it's not already checked for some reason, then this is exactly the place to visit in order to get it installed and working.

Exploring the user interface

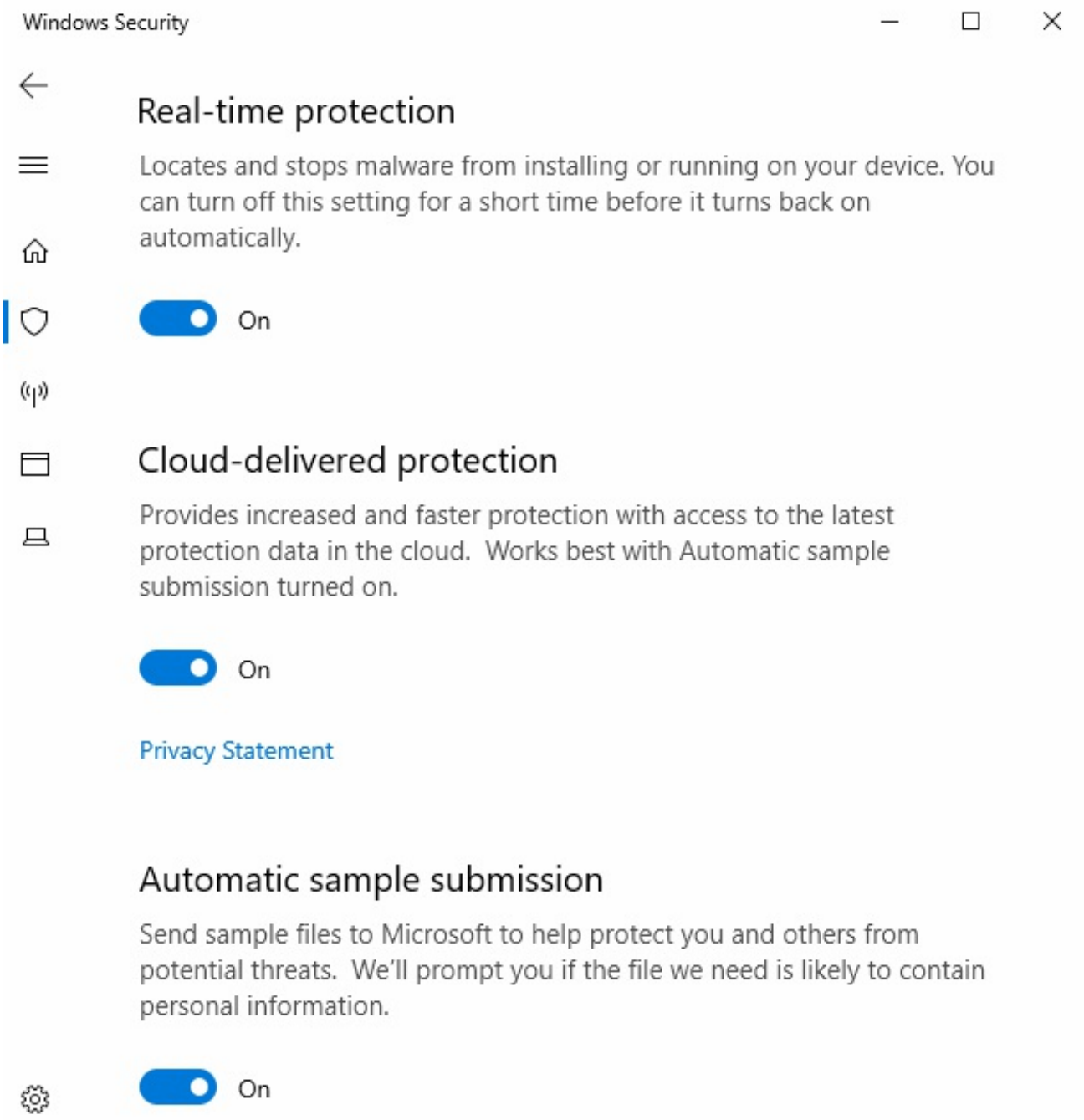
The interface for the Windows Defender toolset is the same as within the latest versions of Windows 10, but if you haven't explored that yet, we will take a quick look at it here. Go ahead and launch Settings from inside the Start menu, then click on Update & Security. Once inside that section, you will see Windows Security listed on the left. Here you get an overhead view of the different Defender components that are working together in order to protect your system.

Remember, you have done nothing to enable any of this functionality; these are all out-of-the-box capabilities:



Clicking further into any of these Protection areas will bring you more detailed descriptions of each capability, as well as many options for enabling or disabling particular protections that exist. For example, if you were to click on Virus & threat protection, you would see summary information about Defender AV, when its definition files were updated, what it's scanning, and so on. Then clicking further into a link called Manage settings will give you options for disabling Defender AV if you ever have the need, as well as numerous other options that can be selected or deselected. Here is a screenshot of just a few of the settings available inside Defender AV. I chose to display these three because

they are important to another topic we will cover shortly, when we discuss the ATP portion of Defender ATP:



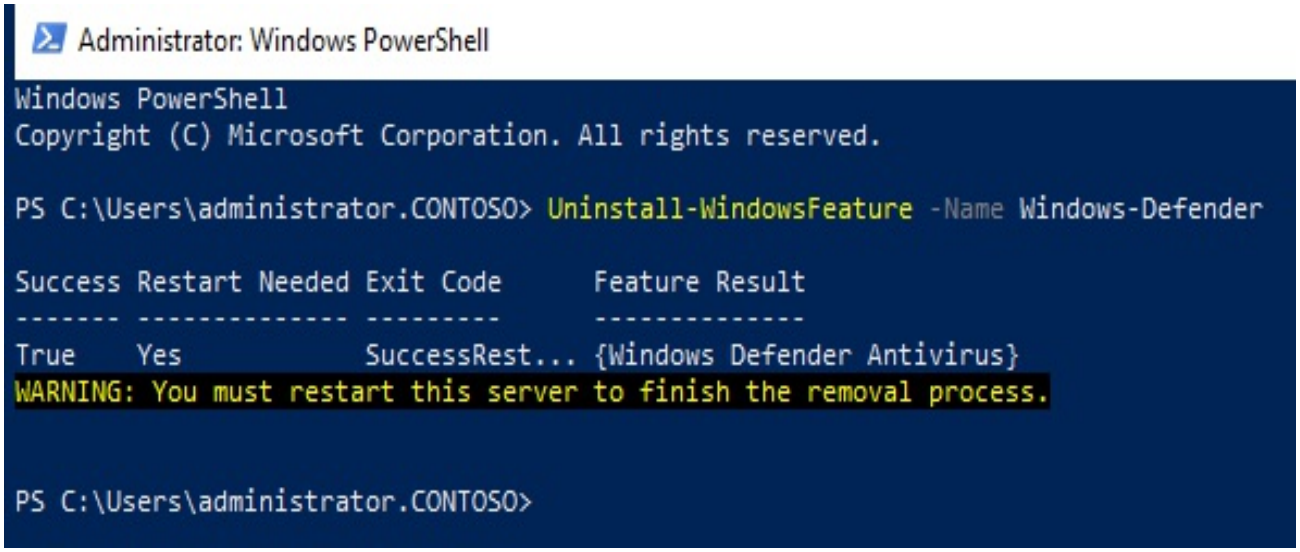
Disabling Windows Defender

You already know that Defender AV is enabled by default, as are many other components that make up the Windows Defender family of products. By flipping the radio option shown in the previous screenshot, you are able to temporarily disable AV. Taking it a step further, if you are absolutely sure that you do not want to use Defender AV because you have your own AV software that you have already paid for, you have two different avenues that could be taken.

First, Defender AV is designed to automatically step down in the event that another AV is installed. More than likely, all you need to do is install your other third-party antivirus tool, and after the server finishes restarting, Defender AV will stand down and allow the third-party product to run, so that they don't conflict with each other. This is important, because a fact that even many computer technicians don't realize is that multiple AV programs running on a single system is generally a terrible idea. They often cause conflicts with each other, have memory allocation errors, and cause otherwise slow and strange behavior on the system.

If you are planning to utilize your own AV and want to make sure Defender is *completely* removed, it is possible to uninstall the Defender feature completely from your server. This is most easily done via PowerShell, with the following command:

```
Uninstall-WindowsFeature -Name Windows-Defender
```


Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\Users\administrator.CONTOSO> Uninstall-WindowsFeature -Name Windows-Defender
Success Restart Needed Exit Code Feature Result
True Yes SuccessRest... {Windows Defender Antivirus}
WARNING: You must restart this server to finish the removal process.
PS C:\Users\administrator.CONTOSO>

What is ATP, anyway?

It's hard to define what exactly ATP means, because it is a culmination of Windows Defender parts, pieces, and security mechanisms working together in order to protect clients and servers from bad stuff: AV, firewalling capabilities, hardware protections, and even specific resistance against ransomware. The combination of capabilities inside the Windows Security section of Server 2019 work together to become ATP.

Something that should be incredibly intriguing to all of us is the smart way that Microsoft is now utilizing cloud connectivity and computing in order to improve Defender AV on a daily basis. Whether we realize it or not, most of the internet-connected Windows machines in the world are now continuously helping each other out by reporting newly discovered vulnerabilities and malicious activity up to Microsoft. This information is then parsed and investigated via machine learning, and the resulting information is able to be immediately used by the rest of the Windows machines around the globe.

While this sounds a little *Big Brother* and full of privacy concerns, I believe we as a community will soon get over that fear and realize that the benefits outweigh the potential fears. Millions of users now flow their email through Office 365; you may not even realize it, but Office 365 does this kind of data handling as well in order to identify and block exploits. For example, if an email address within a company is suddenly sending emails to a large group of people, and that email contains a macro-enabled Word document, which is something that user does not typically do, Office 365 can very quickly take that document offline into

a secure zone, open it (or launch it if the attachment happened to be an executable), and discover whether or not this file is actually malware of some kind. If it is, Office 365 will immediately start blocking that file, thereby stopping the spread of this potentially disastrous behavior. All of this happens without input of the user or of the company's IT staff. This is not even inner company-specific. If one of my users' emails is the first to receive a new virus and it is identified by Microsoft, that discovery will help to block the new virus for any other customers who also host their email in Microsoft's cloud. This is pretty incredible stuff!

This same idea holds true for Defender AV, when you choose to allow it to communicate with and submit information to Microsoft's cloud resources. Earlier, I pasted in a screenshot of some Defender AV capabilities called **cloud-delivered protection and automatic sample submission**—it is these pieces of Defender AV that allow this cloud-based magic to happen and benefit the entire computer population.

Windows Defender ATP Exploit Guard

Once again, we are taking a look at what seems to be a long title for a technology that must have a very specific purpose, right? Nope. The new Exploit Guard is not *a* new capability, but rather a whole *set* of new capabilities baked into the Windows Defender family. Specifically, these new protections are designed to help detect and prevent some of the common behaviors that are used in current malware attacks. Here are the four primary components of the Defender ATP Exploit Guard:

- **Attack Surface Reduction** (**ASR**): ASR is a series of controls that can be enabled that block certain types of files from being run. This can help mitigate malware installed by users clicking on email attachments, or from opening certain kinds of Office files. We are quickly learning as a computer society that we should never click on files in an email that appear to be executables, but oftentimes a traditional computer user won't know the difference between an executable and a legitimate file. ASR can help to block the running of any executable or scripting file from inside an email.

- **Network protection**: This enables Windows Defender SmartScreen, which can block potential malware from phoning

home, communicating back to the attacker's servers in order to siphon or transfer company data outside of your company. Websites on the internet have reputation ratings, deeming those sites or IP addresses to be trusted, or not trusted, depending on the types of traffic that have headed to that IP address in the past. SmartScreen taps into those reputation databases in order to block outbound traffic from reaching bad destinations.

- **Controlled folder access**: Ransomware protection! This one is intriguing because ransomware is a top concern for any IT security professional. If you're not familiar with the concept, ransomware is a type of malware that installs an application onto your computer, which then encrypts files on your computer. Once encrypted, you have no capability of opening or repairing those files without the encryption key, which the attackers will (most of the time) happily hand over to you for lots of money. Every year, many companies end up paying that ransom (and therefore engaging in passive criminal behavior themselves) because they do not have good protections or good backups from which to restore their information. Controlled folder access helps to protect against ransomware by blocking untrusted processes from grabbing onto areas of your hard drive that have been deemed as protected.
- **Exploit protection**: Generalized protection against many kinds of exploits that might take place on a computer. The exploit protection function of Defender ATP is a rollup of capabilities

from something called the **Enhanced Mitigation Experience Toolkit** (**EMET**) that was previously available, but reached end of life in mid-2018. Exploit protection watches and protects system processes as well as application executables.

Windows Defender Firewall – no laughing matter

Let's play a word association game. I will say something, and you say the first word that comes to mind.

Network security.

Did you say *firewall*? I think I would have. When we think of securing our devices at the network level, we think of perimeters. Those perimeters are defined and protected by firewalls, mostly at a hardware level, with specialized networking devices made to handle that particular task in our networks. Today, we are here to talk about another layer of firewalling that you can and should be utilizing in your environments. Yes, we are talking about the Windows Firewall. Stop laughing, it's rude!

It is easy to poke fun at the Windows Firewall based on its history. In the days of Windows XP and Server 2003, it was pretty useless, and caused way more headaches than it solved. In fact, these feelings were so common that I still today find many companies who completely disable the Windows Firewall on all of their domain-joined systems as a matter of default policy. If you ask them, there is usually no specific reason they are doing this—*it's always been this way* or *it's in our written security policy* are standard replies. This is a problem, because the **Windows Defender Firewall with Advanced Security** (**WFAS**) that exists in the Windows OSes of today is much more robust and advanced than ever before, and can absolutely be used to enhance

your security architecture. I would go as far as to say that it is entirely silly to disable WFAS on a current OS, unless you have a very good, very specific reason to do so.

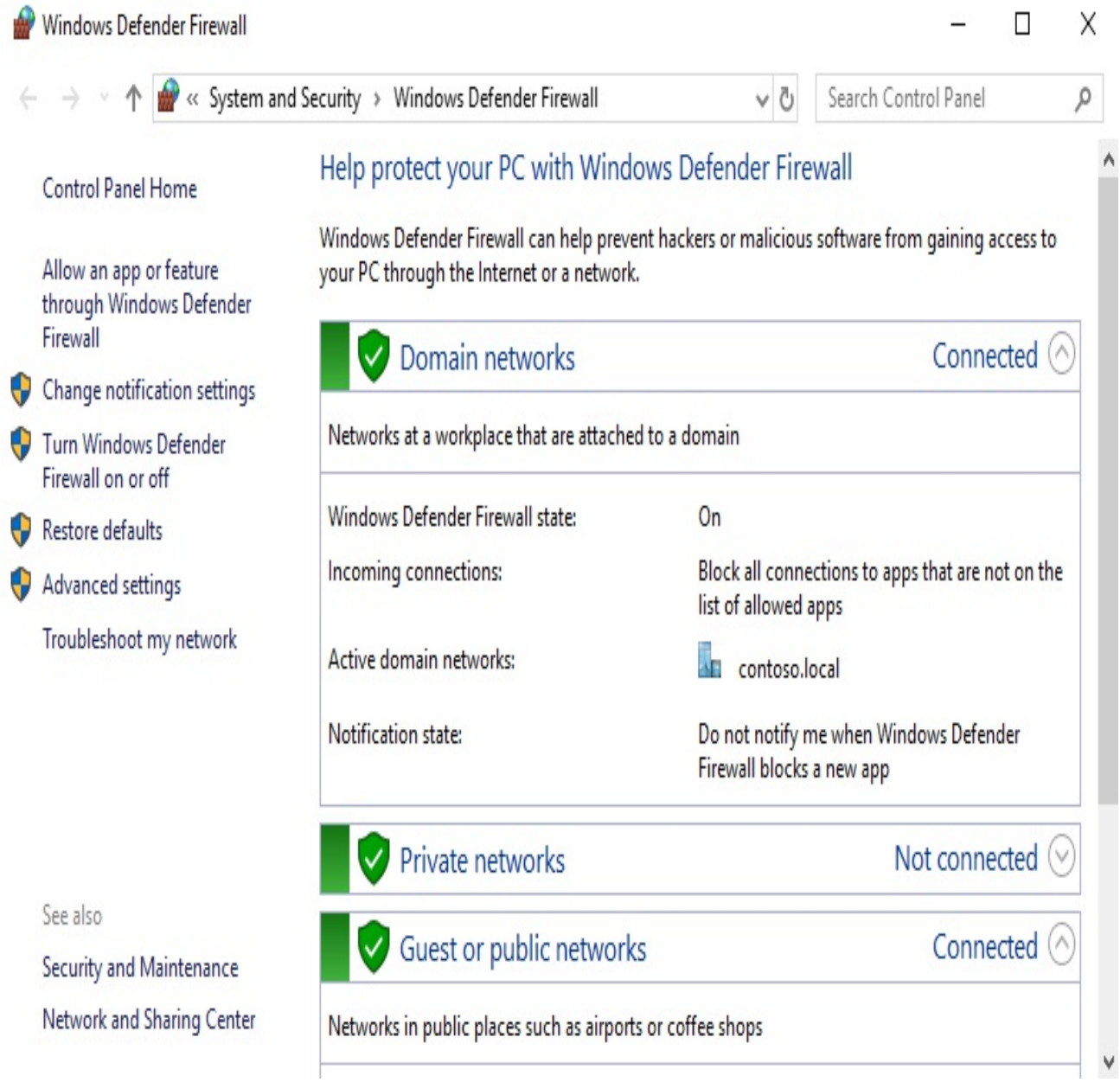
Three Windows Firewall administrative consoles

First, it is important to know that there are three different consoles from which you can configure Windows Firewall settings. Two of these consoles are redundant of each other, and the third is much more capable than the others. Let's take a quick look at each one.

Windows Defender Firewall (Control Panel)

When trying to launch any application or setting in Windows Server 2019, it is usually most efficient to simply click on the Start button, and then type a word relating to the task you are trying to accomplish. In my case, I clicked on Start and typed the word `firewall`. The best match option that was provided first in my search results was Windows Defender Firewall, so I went ahead and clicked on that.

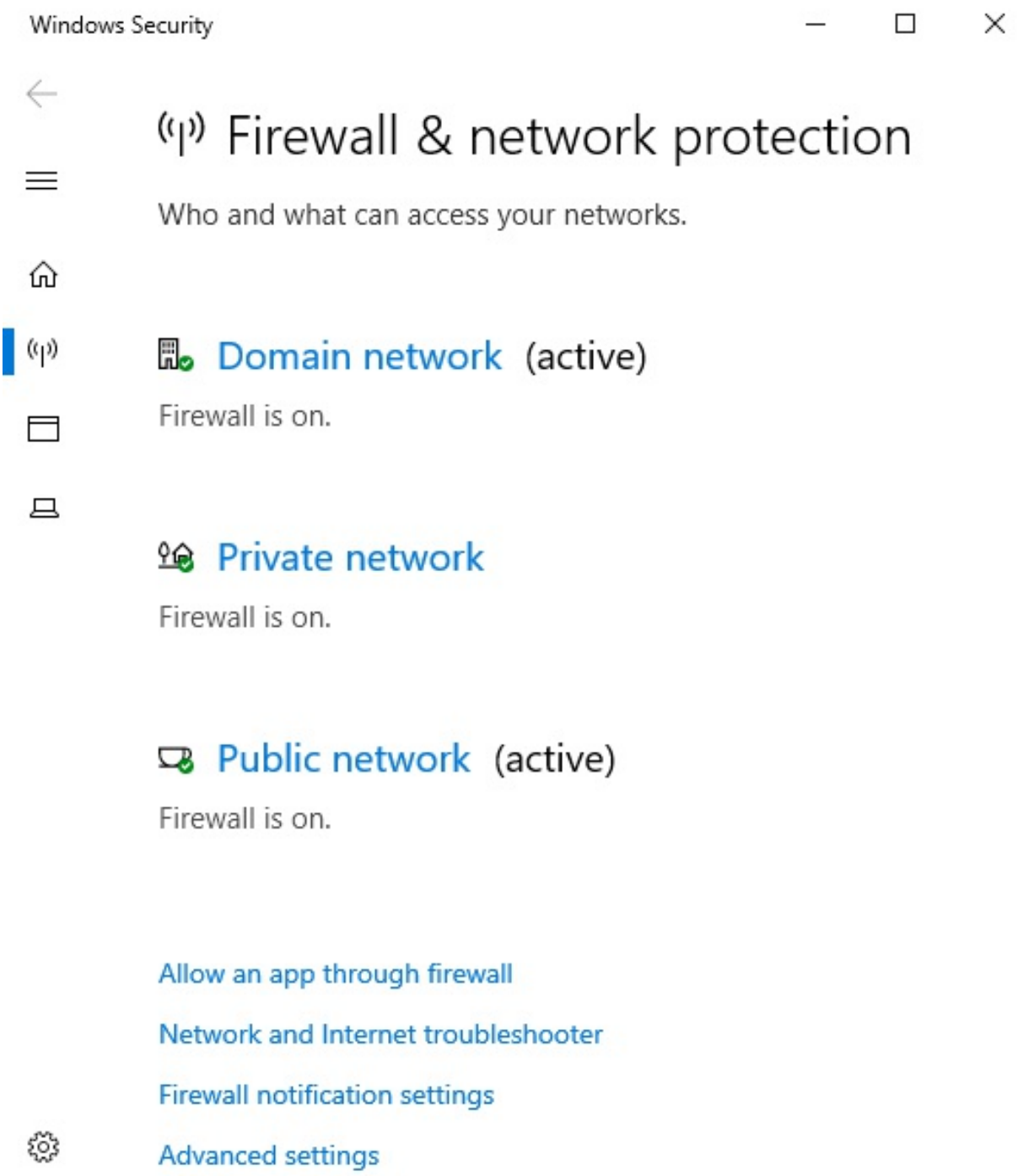
Interestingly, this link opens the Windows Firewall configuration console from inside **Control Panel**, the old-school way of doing system settings. This console is still online and fully capable of manipulating basic firewalling functions, such as enabling or disabling the Windows Firewall, but since this tool resides inside **Control Panel**, we have to assume that this is actually not the tool which Microsoft intends for us to utilize. Remember, all new configuration capabilities have been migrated to the **Windows Settings** screens, rather than the old **Control Panel**:


Windows Defender Firewall
« System and Security › Windows Defender Firewall
Search Control Panel
Control Panel Home
Allow an app or feature through Windows Defender Firewall
Change notification settings
Turn Windows Defender Firewall on or off
Restore defaults
Advanced settings
Troubleshoot my network
See also
Security and Maintenance
Network and Sharing Center
Help protect your PC with Windows Defender Firewall
Windows Defender Firewall can help prevent hackers or malicious software from gaining access to your PC through the Internet or a network.
Domain networks
Connected
Networks at a workplace that are attached to a domain
Windows Defender Firewall state:
On
Incoming connections:
Block all connections to apps that are not on the list of allowed apps
Active domain networks:
contoso.local
Notification state:
Do not notify me when Windows Defender Firewall blocks a new app
Private networks
Not connected
Guest or public networks
Connected
Networks in public places such as airports or coffee shops

Firewall & network protection (Windows Security Settings)

While the **Control Panel**-based tools were always the proper place to make these changes in past versions of the OS, we already know that there are many Windows Defender options stored inside **Windows Settings**. Could it be the case that there are also Windows Defender Firewall configuration settings stored inside the **Windows Security** section of **Settings**?

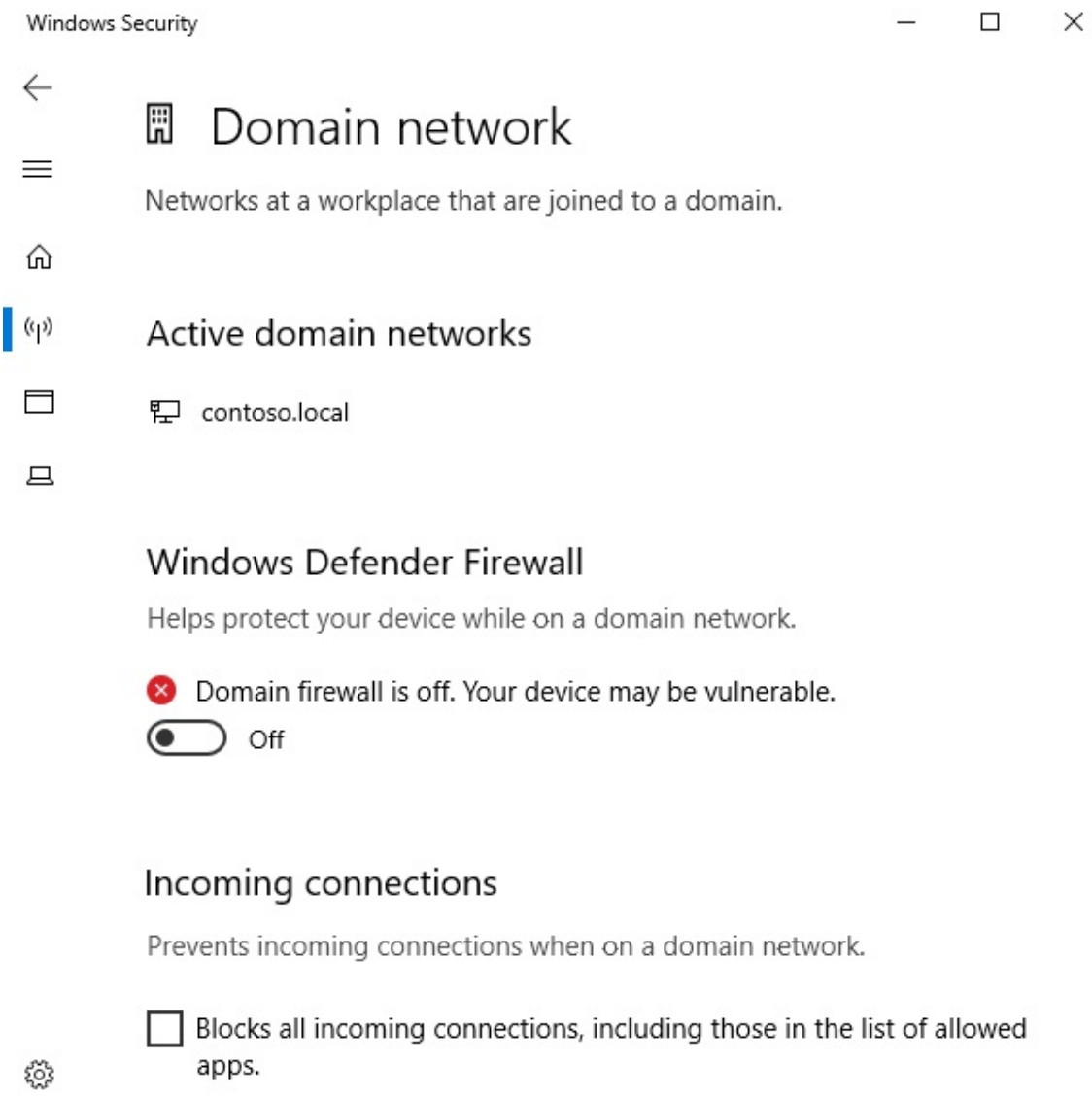
Yes, there definitely are. Open up **Windows Settings** and click on Update & Security, then on Windows Security. You've been here before—this is the screen that gives a quick summary of the Windows Defender components. Sure enough, there is one here called Firewall & network protection. Click on that button, and you will be taken into a new configuration platform for the Windows Firewall functions that did not exist in earlier versions of Windows Server:



Clicking on any of the links provided here will open additional configuration options. For example, if you wanted to quickly enable or disable particular firewall profiles (we will learn about those shortly),

you could click on the profile you want to configure, such as the Domain network profile, and from there easily turn off the firewall for this networking profile. Many companies disable the Domain network profile on their machines, so that the firewall is not protecting traffic that happens inside a corporate LAN network.

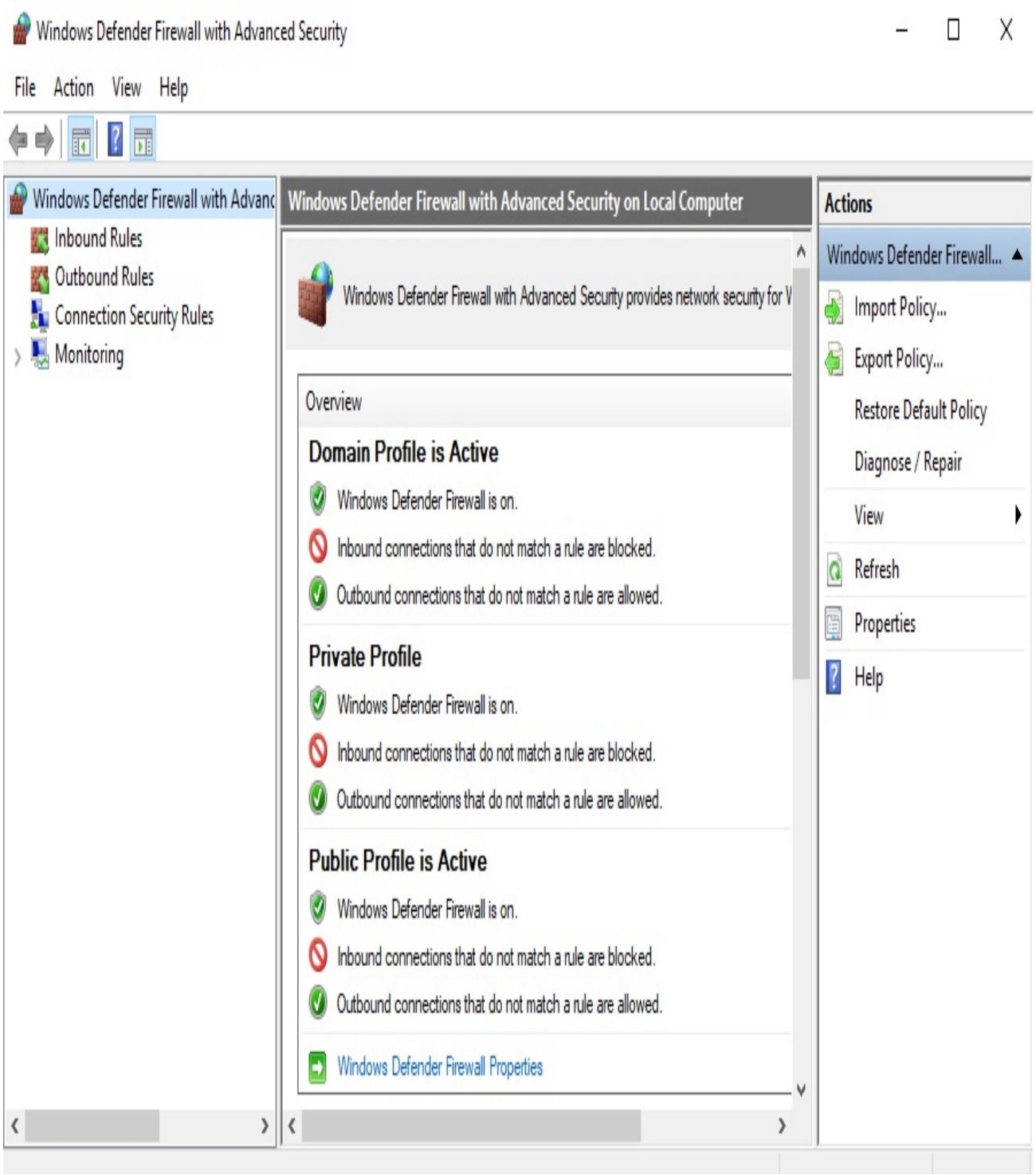
While disabling the firewall is generally a bad idea, sometimes it is required to fit your business model:



The firewall configuration screen available inside Windows Settings is a good place to make simple, overhead decisions about the Windows Defender Firewall, but this interface is limited in capabilities. For any real utilization of firewall functionality or configuration....

Windows Defender Firewall with Advanced Security (WFAS)

If you are anything like me, you won't be satisfied with this information and will want to see what is going on under the hood, and so you will want a little more information than the basic Windows Firewall tools alone can give you. You can either click on one of the Advanced settings links shown in previous screenshots, or simply open Command Prompt or a Start | Run prompt and type `wf.msc`. Either of these functions will launch the full WFAS administration console:



Here you can see much more in-depth information about the activity and rules that are in play with the Windows Firewall, and make more

acute adjustments in your allowances and blocks. There is also a Monitoring section where you can view actively engaged rules, including Connection Security Rules. This is an important section because it highlights the fact that WFAS does much more than block network traffic. It is not only a firewall, it is also a connectivity platform. If you plan to utilize IPsec for encryption of network traffic, whether it be native IPsec inside your network or through the remote access technology DirectAccess, you will see rules populated in this section that are the definitions of those IPsec tunnels. Windows Firewall is actually responsible for making those encrypted connections and tunnels happen. This is way more advanced than the Windows Firewall of yesteryear.

Three different firewall profiles

When any NIC on a computer or server is connected to a network, the Windows Firewall will assign that connection one of the three different profiles. You have probably interfaced with this decision-making process before without even realizing it. When you connect your laptop to the Wi-Fi at your local coffee shop, did Windows ask you whether you were connecting to a home, work, or public network? This is your Windows Firewall asking you which profile you would like to assign to the new network connection. The reason that you can assign NICs and network connections to different firewall profiles is that you can assign different access rules and criteria for what is or is not allowed over those different profiles. Effectively, it is asking you *how much do you trust this network?* For example, when your laptop is connected to the corporate network you can probably be a little bit more lax than when that same laptop is connected at a hotel across the country. By assigning more intense firewall rules to the profile that is active when you are in the hotel, you build bigger walls for attackers to face when you are out working on that public internet. Let's take a look at the three different types of profiles that are available, with a quick description of each:

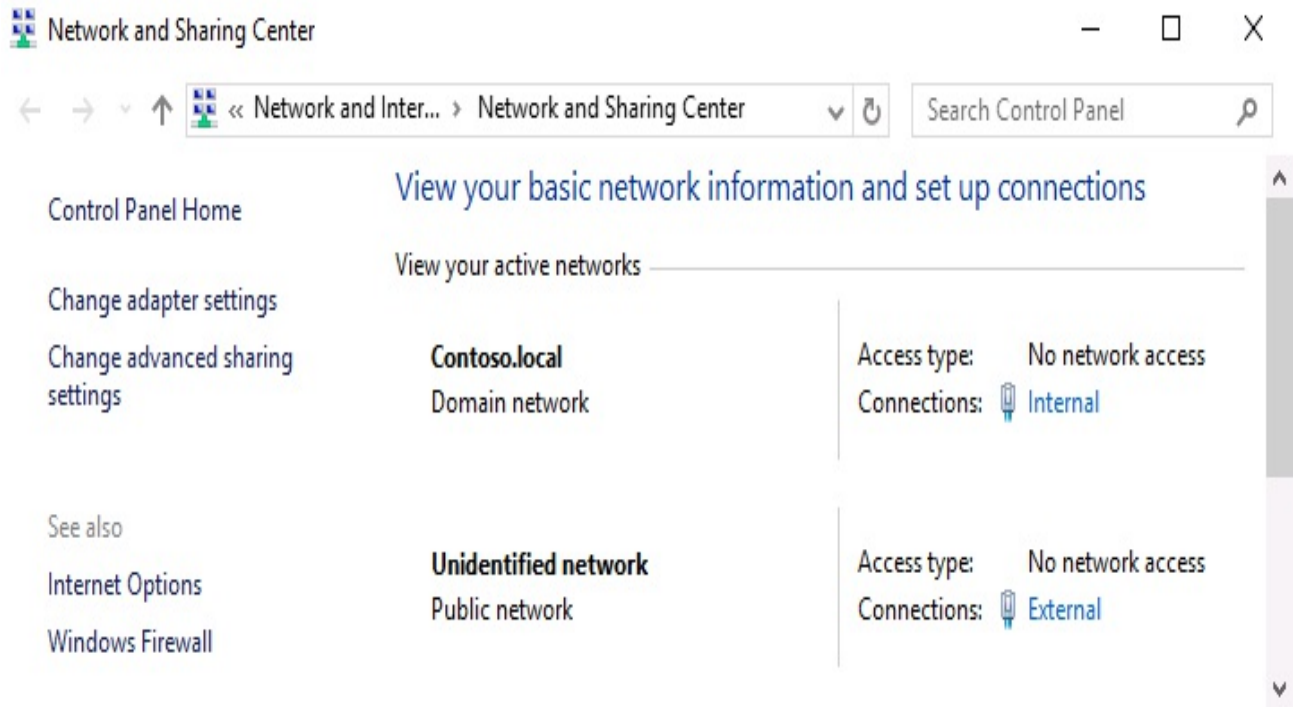
- **Domain Profile**: This is the only one that you cannot choose to assign. The Domain Profile is only active when you are on a domain-joined computer that is currently connected to a network where a domain controller for your domain is accessible. So, for any corporate machine inside the corporate network, you can expect that the Domain Profile would be active.

- **Private Profile**: When connecting to a new network and you are prompted to choose where you are connected, if you choose either **Home** or **Work**, that connection will be assigned the Private Profile.
- **Public Profile**: When prompted, if you choose **Public**, then of course you are assigned the public firewall profile. Also, if you are not prompted for some reason, or if you do not choose an option at all and simply close the window that is asking you what to assign to your new connection, this Public Profile will be the default profile that is given to any connections that do not have a different profile already assigned. In the more recent versions of Windows (particularly in Win10), you don't usually get the prompt asking what kind of a network it is; instead you get a prompt asking whether or not you want to allow your computer to communicate with other devices on the new network. Effectively, this is still the same prompt, and the decision you make at that prompt will assign your connection to either the public or private firewall profile.

Each network connection gets assigned its own profile definition, you could certainly have more than one firewall profile active at the same time on the same system. For example, my RA1 server is connected to both the corporate network as well as the public internet. Inside WFAS, you can see that both the Domain Profile and Public Profile are active:



Alternatively, if you open up Network and Sharing Center on this server, we can also see the profiles listed here, and you can easily tell which NIC is using which profile:


Network and Sharing Center
« Network and Inter... › Network and Sharing Center
Search Control Panel
Control Panel Home
Change adapter settings
Change advanced sharing settings
See also
Internet Options
Windows Firewall
View your basic network information and set up connections
View your active networks
Contoso.local
Domain network
Access type: No network access
Connections: Internal
Unidentified network
Public network
Access type: No network access
Connections: External

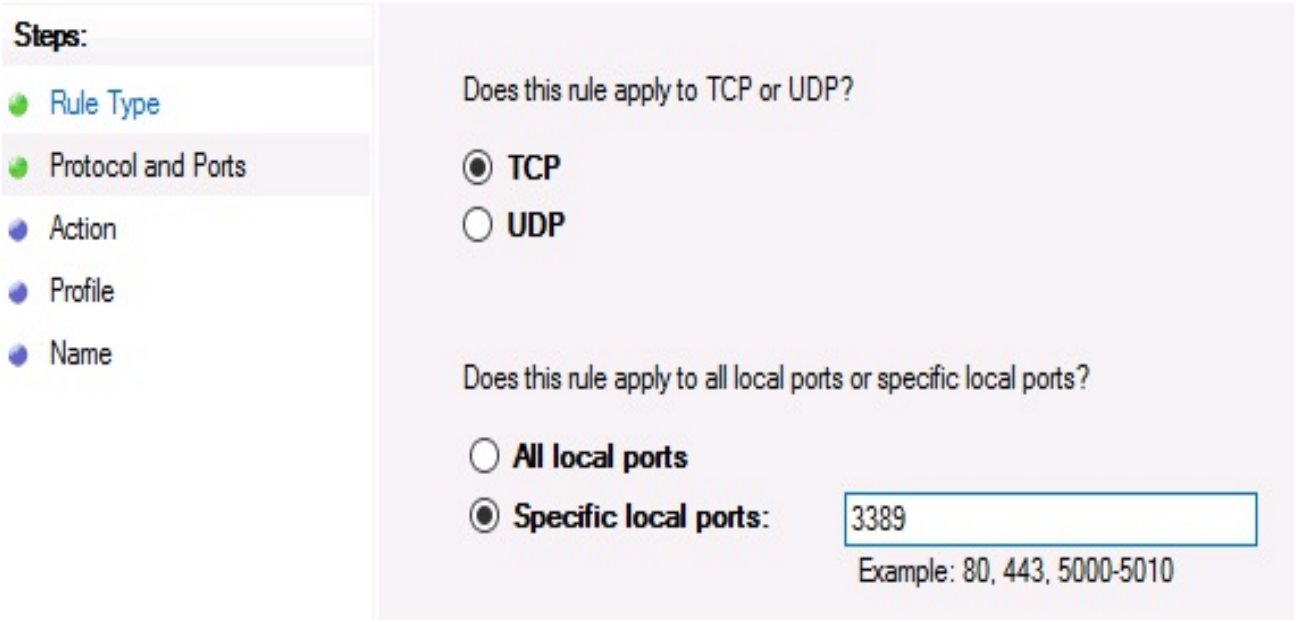
Building a new inbound firewall rule

Now we know that the real meat and potatoes of the Windows Firewall is inside the WFAS console, so let's use WFAS to build ourselves a new rule. On this RA1 server, I have enabled RDP access so that I can more easily manage this server from my desk. However, by turning on RDP I have now allowed RDP access from all of the networks on this server. That means I can RDP into RA1 from inside the network, but I can also RDP into RA1 from the internet, since this is a remote access server and happens to be connected straight to the internet. This is a big problem, because now any yahoo on the internet could potentially find my server, find the RDP login prompt, and try to brute force their way into RA1.

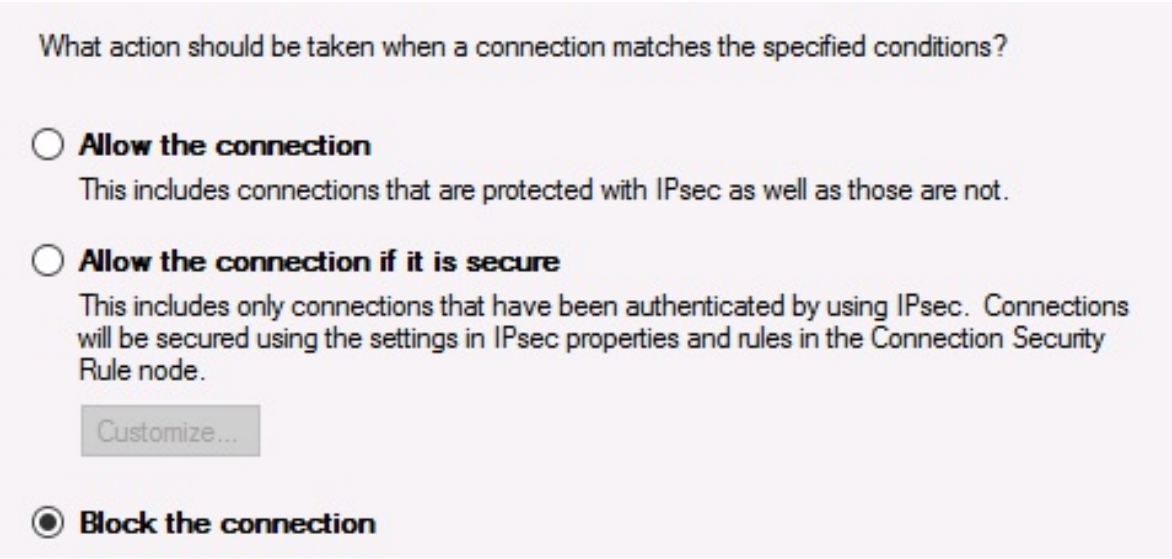
To alleviate this problem, I want to squash RDP only on my external NIC. I want it to remain active on the inside so that I can continue to access the server from my desk, but is there an easy way inside WFAS to create a firewall rule that blocks RDP access only from the outside? Yes, there certainly is.

Open up `wf.msc` in order to launch the Windows Defender Firewall with Advanced Security, and navigate to the Inbound Rules section and you will see all of the existing inbound firewall rules that exist on this server (there are many rules listed here even if you have never visited this console before, these rules are installed with the OS). Right-click on Inbound Rules, and choose New Rule.... This launches a wizard from which we will create our new firewall rule. The first screen is where we identify what kind of a rule we want to create. You can create a rule that

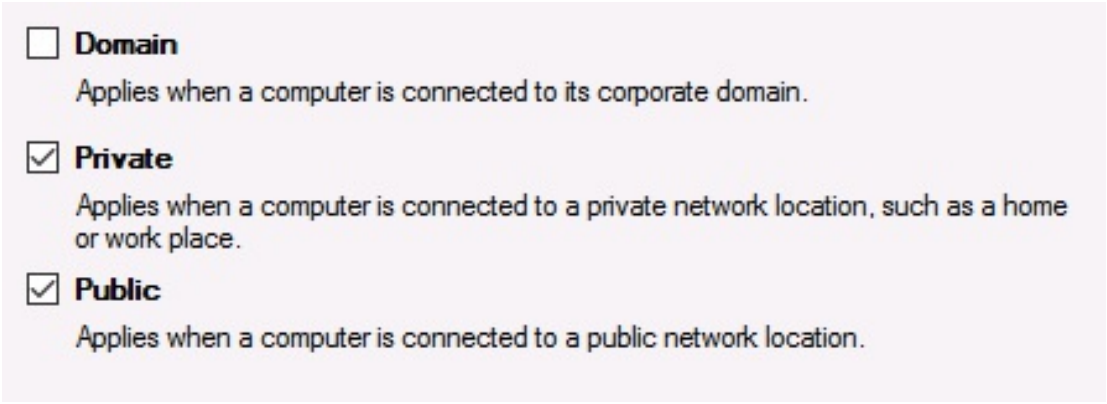
modifies traffic for a particular program, or you can look through a list of Predefined protocols. However, I like knowing exactly what my rule is doing because of the way that *I* defined it, not because of a pre-existing protocol definition, and I know that RDP runs over TCP port `3389`. So, I am going to choose port on this screen, and after I click on Next, I will define `3389` as the specific port that I want to modify:



Our third step is to decide whether we want to allow or block this particular port. There is a third option listed about only allowing the connection if it is authenticated by IPsec, which is a powerful option, but necessitates having IPsec established in our network already. Because of that requirement, this option doesn't apply to most people. For our example, we already have RDP working, but we want to block it on one of the NICs, so I am going to choose Block the connection:



We don't want to block RDP for *all* of the NICs, though, so this next screen is very important. Here we need to reference back to our knowledge about those firewall profiles we talked about. Remember that internal NICs connected to our domain network will have the Domain Profile assigned to them. But any NICs that are not connected to an internal network where a domain controller resides will have either Public or Private profiles active. That is the knowledge we need to employ on this screen. If we want to disable RDP only on the external NIC, we need this rule to be active for only the Private Profile and Public Profile. In fact, in looking back at the screenshots we already took, we can see that the external NIC is assigned the Public Profile specifically, and so we could check only the **Public** checkbox here and RDP would then be blocked on the external NIC. But in case we add more NICs to this server in the future over which we want to make sure RDP access is not possible, we will leave both Public and Private checked, to ensure better security for the future. Make sure that you **uncheck** the **Domain** profile! Otherwise you will block RDP access completely, and if you are currently using RDP in order to connect to this server, you will kick yourself out of it and be unable to reconnect:



And now we simply create a name for our new rule, and we are done! Our ability to RDP into this server from the internet has immediately been disabled, and we can rest much easier tonight.

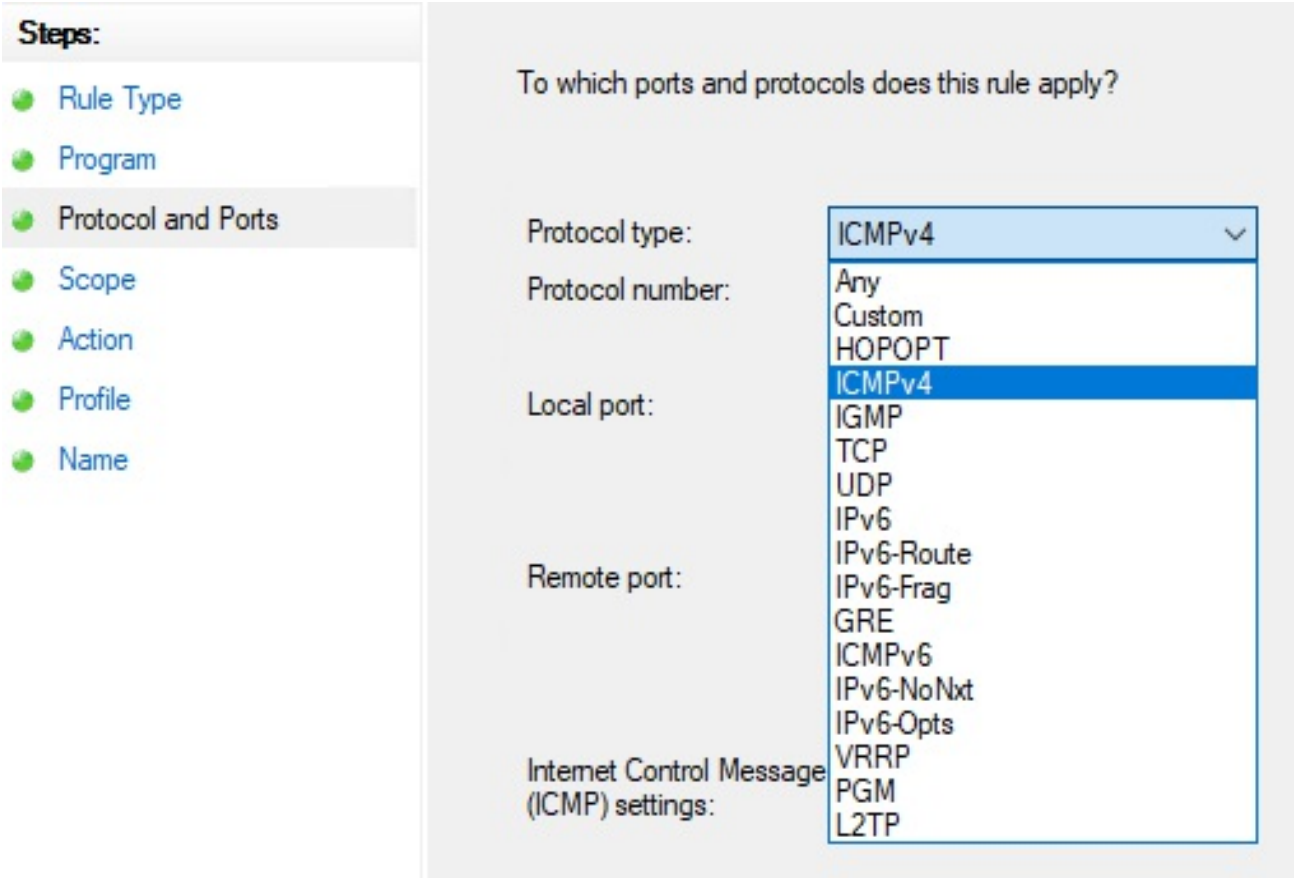
Creating a rule to allow pings (ICMP)

Very often I find myself needing to create either an allow or a block rule for ICMP. In other words, I often find myself needing to adjust the firewall on servers in order to enable or disable their ability to reply to ping requests. You probably noticed with newer server OSes that it is pretty normal for the firewall to automatically block pings (ICMP) out of the box. This is a problem for environments where ping is the standard method for testing whether an IP address is consumed or available. You may be laughing, but, trust me, there are still plenty of IT administrators out there that don't keep track of which IP addresses they have used inside their networks, and when faced with the need to set up a new server and decide what IP address to give it, they simply start pinging IP addresses in their network until they find one that times out! I have seen this so many times. While this is obviously not a good way to manage IP addresses, it happens. Unfortunately, this method encounters big problems, because most new Windows installations are designed to block ICMP responses out of the box, which means that you may ping an IP address and receive a timeout, but there could actually be a server running on that IP address.

So, getting back to the point. You may have a need to enable ICMP on your new server, so that it responds when someone tries to ping it. When we need to create a new rule that allows pings to happen, we set up a rule just like we did for RDP, but there is one big catch. On that very first Rule Type screen when creating the new rule where you have to identify what kind of rule you are creating, there are no options or

predefinitions for ICMP. I find this strange because this is a very common type of rule to put into place, but alas choosing ICMP from the drop-down list would just be too easy. Instead, what you need to do is create a new inbound rule just like we did for RDP, but at the very first screen for Rule Type, make sure you select the option that says Custom.

Next, leave the option selected to define this rule for All programs. Click next again, and now you have a drop-down box called Protocol type. This is the menu where you can choose for your new rule to manipulate ICMP traffic. As you can see in the following screenshot, you can choose **ICMPv4** or **ICMPv6**, depending on what your network traffic looks like. My test lab is IPv4-only, so I am going to choose ICMPv4:



For the rest of the ICMP rule creation, follow the same procedures

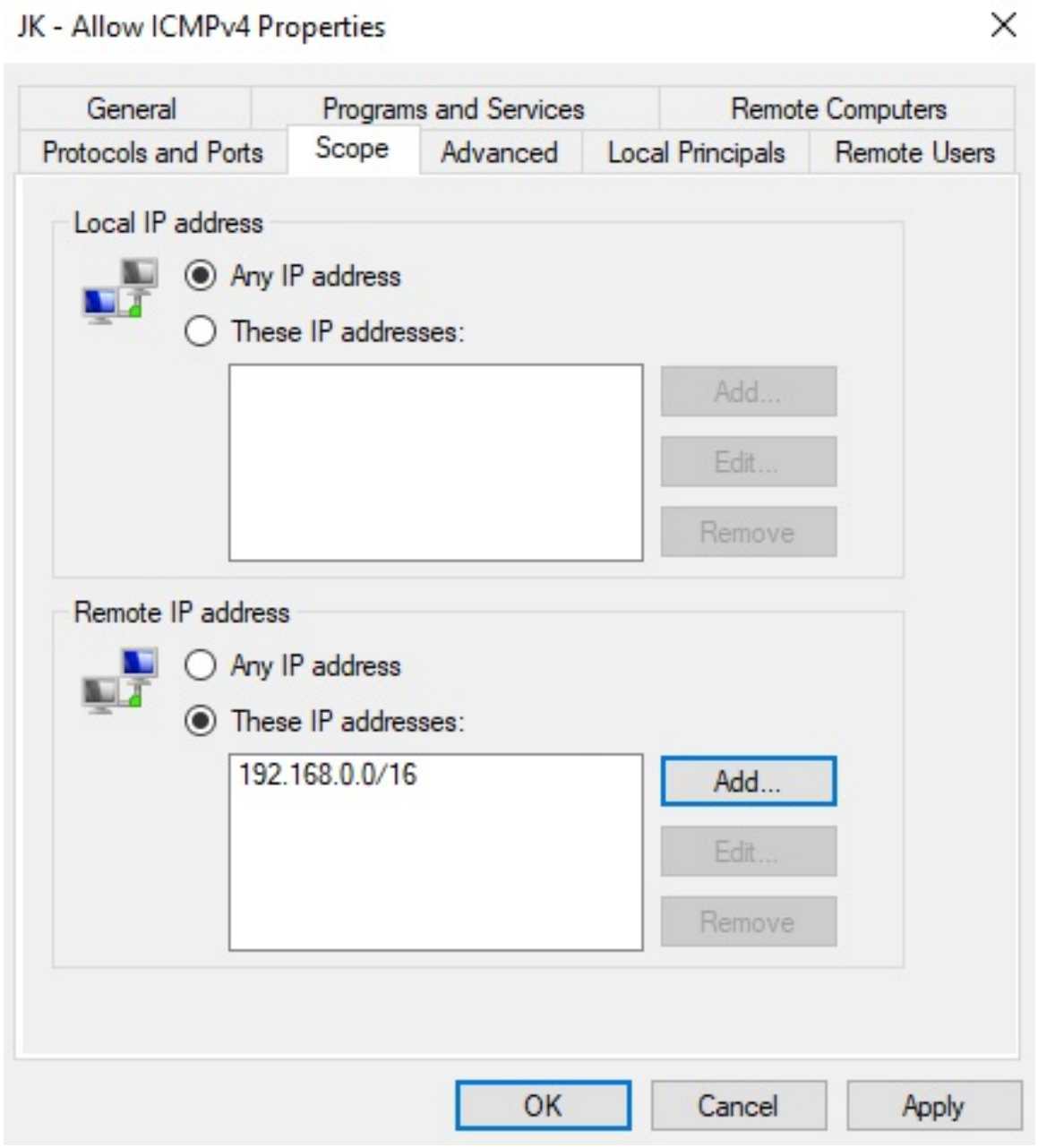
outlined when we created the RDP rule, choosing to either allow or block this traffic, and for which firewall profiles. Once finished, your new ICMPv4 rule is immediately enacted, and if you have configured an Allow rule, your new server will now successfully respond to ping requests:

```
PS C:\Users\administrator.CONTOSO> ping ra1

Pinging ra1.contoso.local [10.10.10.13] with 32 bytes of data:
Reply from 10.10.10.13: bytes=32 time<1ms TTL=128
Reply from 10.10.10.13: bytes=32 time<1ms TTL=128
Reply from 10.10.10.13: bytes=32 time<1ms TTL=128
Reply from 10.10.10.13: bytes=32 time<1ms TTL=128
```

If ever you need to modify a rule or dig into more advanced properties of a firewall rule, back at the Inbound Rules screen you can right-click on any individual firewall rule and head into Properties. Inside these tabs, you have the opportunity to modify any criteria about the rule. For example, you could accommodate additional ports, you could modify which firewall profiles it applies to, or you could even restrict which specific IP addresses this rule applies to by use of the Scope tab.

This enables you to apply your firewall rule only to traffic coming or going from a specific portion of your network, or a certain subset of machines. For example, here I have modified my **Scope** tab to reflect that I only want this firewall rule to apply to traffic that is coming in from the `192.168.0.0/16` subnet:


JK - Allow ICMPv4 Properties
General
Programs and Services
Remote Computers
Protocols and Ports
Scope
Advanced
Local Principals
Remote Users
Local IP address
Any IP address
These IP addresses:
Add...
Edit...
Remove
Remote IP address
Any IP address
These IP addresses:
192.168.0.0/16
Add...
Edit...
Remove
OK
Cancel
Apply

Managing WFAS with Group Policy

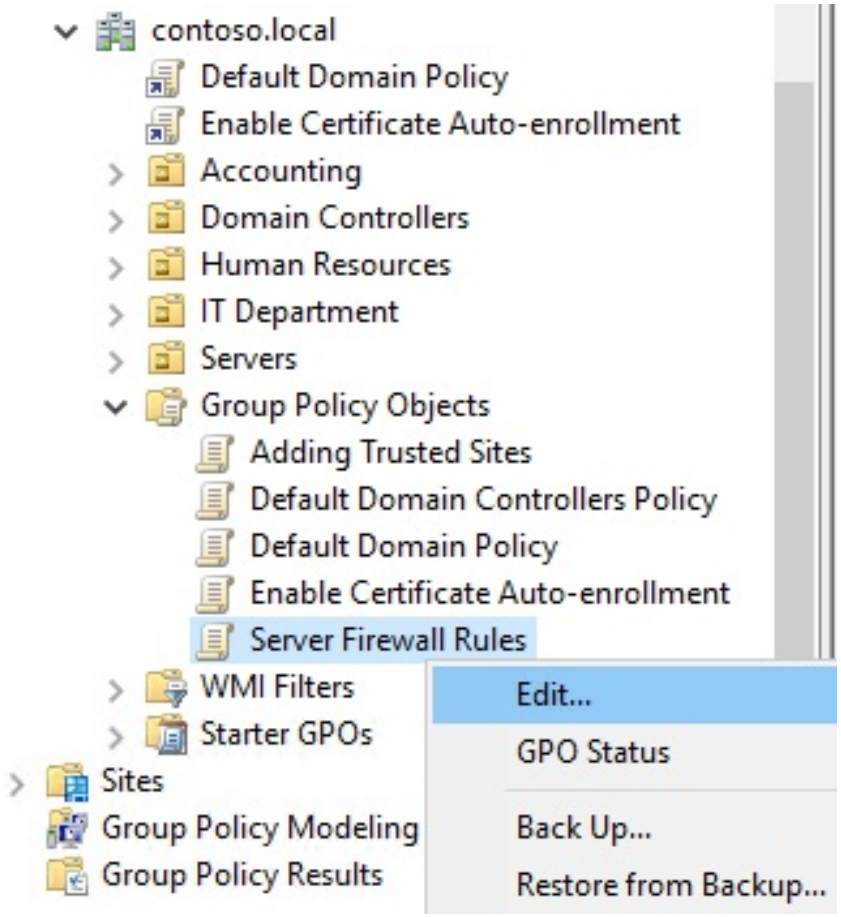
Managing firewall rules on your servers and clients can be a huge step toward a more secure environment for your company. The best part? This technology is enterprise class, and free to use since it's already built into the OSes that you use. The only cost you have associated with firewalling at this level is the time it takes to put all of these rules into place, which would be an administrative nightmare if you had to implement your entire list of allows and blocks on every machine individually.

Thank goodness for **Group Policy Object** (**GPO**). As with most settings and functions inside the Microsoft Windows platform, setting up a firewall policy that applies to everyone is a breeze for your domain-joined machines. You can even break it up into multiple sets of policies, creating one GPO that applies firewall rules to your clients, and a separate GPO that applies firewall rules to your servers, however you see fit. The point is that you can group many machines together into categories, create a GPO ruleset for each category, and automatically apply it to every machine by making use of the GPO's powerful distribution capabilities.

You are already familiar with creating GPOs, so go ahead and make one now that will contain some firewall settings for us to play with. Link and filter the GPO accordingly so that only the machines you want to have the settings will actually get them. Perhaps a good place to start is a testing OU, so that you can make sure all the rules you are about to

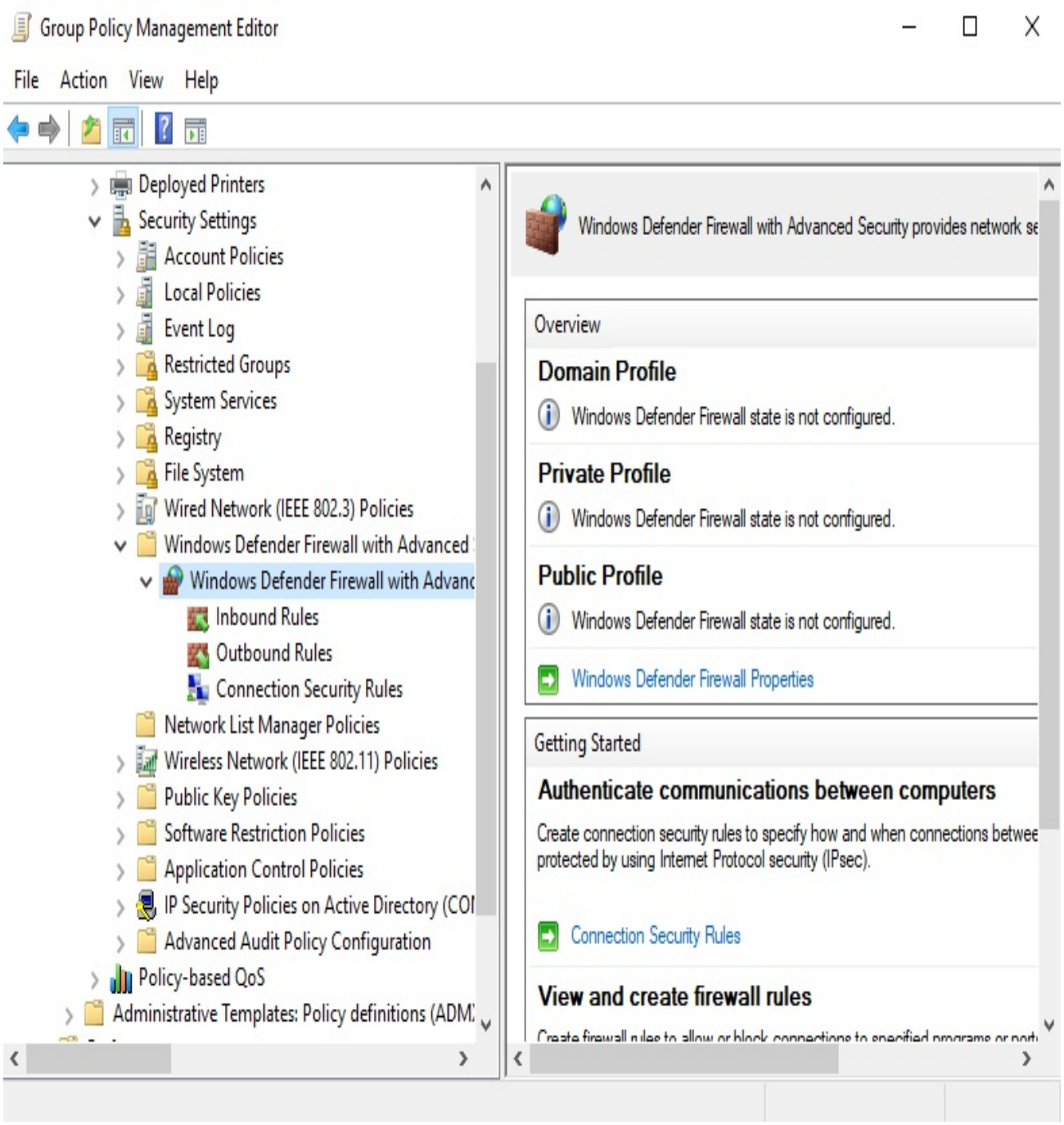
place inside the GPO work well together and with all of your other existing policies, before rolling the new policy out to your production workforce.

Once your new GPO is created, right-click on it from inside the **Group Policy Management Console** and click on Edit...:



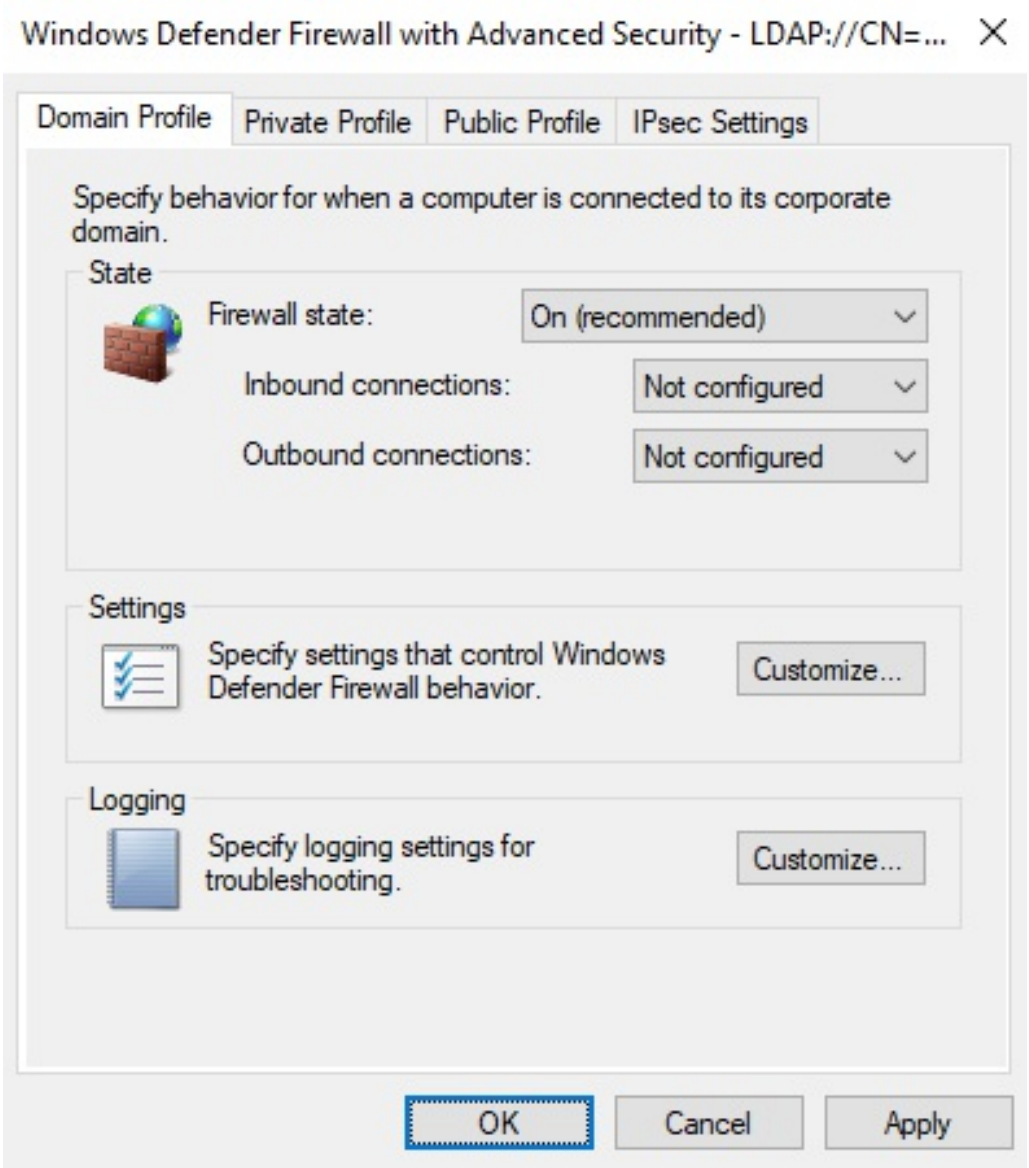
Now that we are looking at the insides of this new GPO, we just need to figure out where the correct location is in order for us to create some new firewall rules. When you are looking inside the rules on the local machine itself, everything is listed under a Windows Defender Firewall with Advanced Security heading, and that is located at Computer Configuration | Policies | Windows Settings | Security Settings |

Windows Defender Firewall with Advanced Security | Windows Defender Firewall with Advanced Security:



As you can see, this is also the place to go when you want to make sure

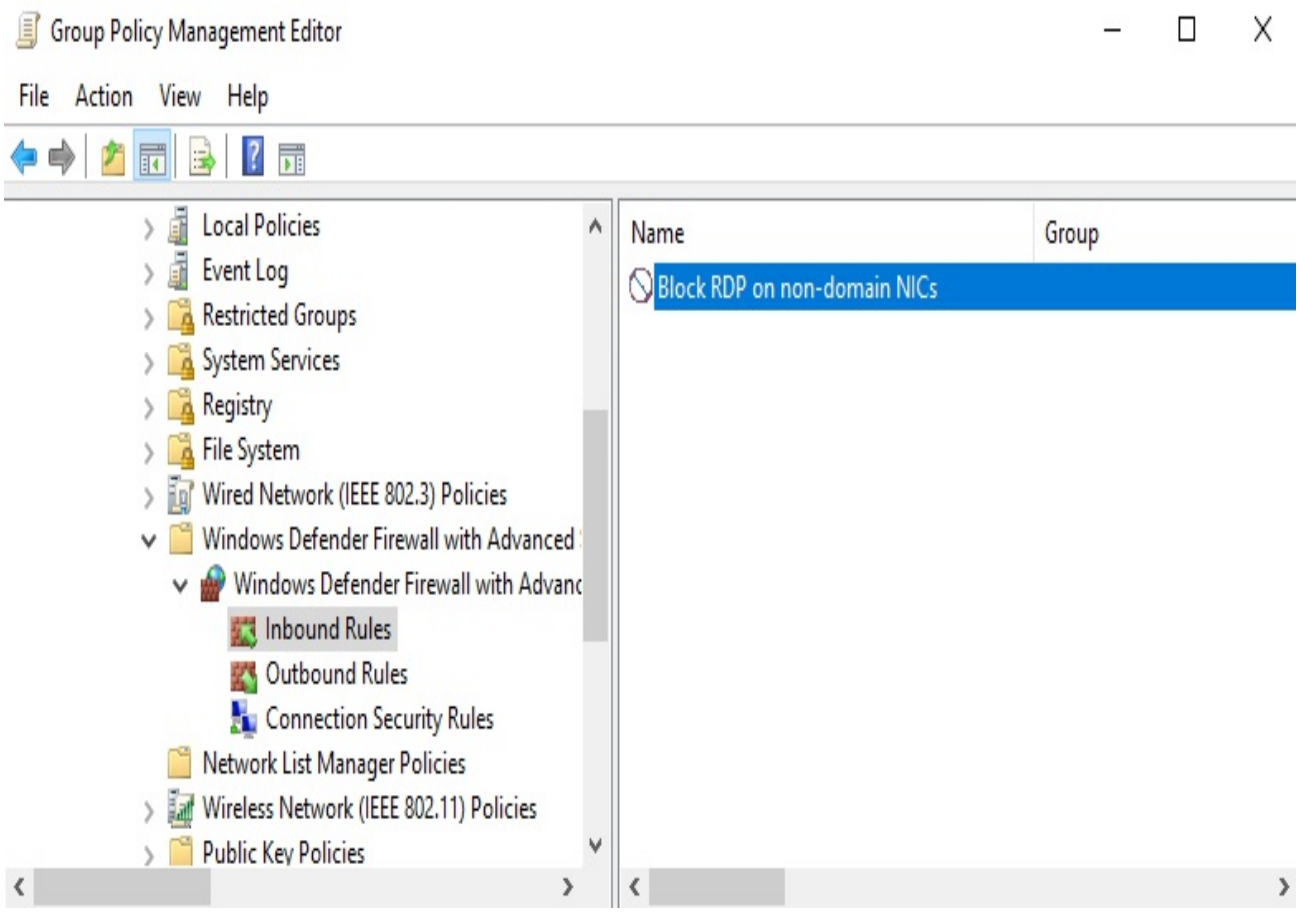
that particular firewall profiles, or the Windows Firewall as a whole, are specifically turned on or off. So, this is the same place that you would go if you wanted to disable the Windows Firewall for everyone. By clicking on the Windows Defender Firewall properties, link shown earlier, you can determine the status of each firewall profile individually:



Once you are finished setting your profiles according to your needs, click on OK, and you find yourself back at the WFAS part of the GPO.

Just like inside the local WFAS console, you have categories for Inbound Rules and Outbound Rules. Simply right-click on Inbound Rules and click on New Rule... in order to get started with building a rule right into this GPO. Walk through the same wizard that you are already familiar with from creating a rule in the local WFAS console, and when you are finished, your new inbound firewall rule is shown inside the GPO.

This firewall rule is already making its way around Active Directory, and installing itself onto those computers and servers that you defined in the policy's links and filtering criteria:



Encryption technologies

An idea that has taken a fast step from *something the big organizations are playing around with* to *everybody needs it* is the use of encryption. Most of us have been encrypting our website traffic for many years by using HTTPS websites, but even in that realm there are surprising exceptions, with a lot of the cheap web-hosting companies still providing login pages that transmit traffic in clear text. This is terrible, because with anything that you submit over the internet now using regular HTTP or an unencrypted email you *have* to assume that it is being read by someone else. Chances are you are being paranoid and nobody is actually intercepting and reading your traffic, but you need to know that if you are accessing a website that says HTTP in the address bar, or if you are sending an email from any of the free email services, any data that is being entered on that web page or in that email can easily be stolen by someone halfway around the world. Data encryption is an absolute requirement these days for corporate information that needs to traverse the internet; though at the same time I say that, the back of my mind is telling me that the vast majority of companies are still not using any kind of encryption technology on their email system, and so that is still a potential disaster waiting to happen for most.

While we are getting better and better at protecting internet browser traffic, we traditionally still are not paying a lot of attention to data that is *safe* within the walls of our organization. The bad guys aren't dumb, though, and they have a very large toolbox of tricks to socially engineer their way into our networks. Once inside, what do they find? In most cases, it's a big free-for-all. Get a hold of one user account or one computer and you've got keys to a large part of the kingdom.

Fortunately, there are several technologies built into Windows Server 2019 that are designed to combat these intrusions and protect your data even while it sits within the four walls of your data center. Let's look at some information on them so that you can explore the possibility of using these encryption technologies to further protect your data.

BitLocker and the virtual TPM

BitLocker is a technology that has become pretty familiar to see on our client systems within corporate networks. It is a full-drive encryption technology, giving us the advantage of making sure our data is fully protected on laptops or computers that might be stolen. If a thief gets their hands on a company laptop, claws out the hard drive, and plugs it into their computer...sorry, Charlie, no access. The entire volume is encrypted. This makes all kinds of sense for mobile hardware that could be easily lost or stolen, but in the beginning stages of this technology there was never real consideration for using BitLocker to protect our servers.

With the escalated adoption of cloud computing resources, suddenly it makes much more sense to want BitLocker on our servers. More particularly when talking about the cloud, what we really want is BitLocker on our virtual machines, whether they be client or server OSes. Whether you are storing your **Virtual Machines** (**VMs**) in a true cloud environment provided by a public cloud service provider or are hosting your own private cloud where tenants reach in to create and manage their own VMs, without the possibility of encrypting those virtual hard drives—the VHD and VHDX files—your data is absolutely **not** secure. Why not? Because anyone with administrative rights to the virtualization host platform can easily gain access to any data sitting on your server's hard drives, even without any kind of access to your network or user account on your domain. All they have to do is take a copy of your VHDX file (the entire hard drive contents of your server), copy it to a USB stick, bring it home, mount this virtual hard disk on their own system, and bingo—they have access to your server hard drive

and your data. This is a big problem for data security compliance.

Why has it historically not been feasible to encrypt VMs? Because BitLocker comes with an interesting requirement. The hard drive is encrypted, which means that it can't boot without the encryption being unlocked. How do we unlock the hard drive so that our machine can boot? One of two ways. The best method is to store the **unlock keys** inside a **Trusted Platform Module** (**TPM**). This is a physical microchip that is built right into most computers that you purchase today. Storing the BitLocker unlock key on this chip means that you do not have to connect anything physically to your computer in order to make it boot, you simply enter a pin to gain access to the TPM, and then the TPM unlocks BitLocker. On the other hand, if you choose to deploy BitLocker without the presence of a TPM, to unlock a BitLocker volume and make it bootable, you need to plug in a physical USB stick that contains the BitLocker unlock keys. Do you see the problem with either of these installation paths in a virtual machine scenario? VMs cannot not have a physical TPM chip, and you also have no easy way of plugging in a USB stick! So, how do we encrypt those VMs so that prying eyes at the cloud hosting company can't see all my stuff?

Enter the **virtual TPM**. This capability came to us brand new in Windows Server 2016; we now have the capability of giving our virtual servers a virtual TPM that can be used for storing these keys! This is incredible news, and means that we can finally encrypt our servers, whether they are hosted on physical Hyper-V servers in our data center or sitting in the Azure Cloud.

Shielded VMs

Using BitLocker and virtual TPMs in order to encrypt and protect virtual hard drive files produces something called **Shielded VMs**. Shielded virtual machines were a capability first introduced in Windows Server 2016, and have been improved in Server 2019. I know this is just a tiny taste and preview of this amazing new technology, but I wanted to mention it here because it definitely relates to the overall security posture of our server environments.

We will cover much more detail on Shielded VMs in `Chapter 12`, *Virtualizing Your Data Center with Hyper-V*.

Encrypted virtual networks

Wouldn't it be great if we could configure, control, and govern our networks from a graphical administrative interface, rather than looking at router CLIs all day? Would we not benefit from networking flexibility to move servers and workloads from one subnet to another, without having to change IP addressing or routing on those servers? Couldn't we find some way to automatically encrypt all of the traffic that is flowing between our servers, without having to configure that encryption on the servers themselves?

Yes, yes, yes! Through the use of **Software Defined Networking** (**SDN**) and a new capability called **encrypted virtual networks**, we can accomplish all of these things. This section of text is really just a reference point, a place to steer you back toward `Chapter 5`, *Networking with Windows Server 2019*, if you skipped over it and landed here instead. We have already discussed SDN and its new capability to create and automatically encrypt virtual networks that flow between Hyper-V VMs and Hyper-V host servers, so if this idea intrigues you, make sure to head back and revisit that chapter.

Encrypting File System

The **Encrypting File System** (**EFS**) is a component of Microsoft Windows that has existed on both client and server OSes for many years. Whereas BitLocker is responsible for securing an entire volume or disk, EFS is a little more particular. When you want to encrypt only particular documents or folders, this is the place you turn to. When you choose to encrypt files using EFS, it is important to understand that Windows needs to utilize a user certificate as part of the encrypt/decrypt process, and so the availability of an internal PKI is key to a successful deployment. Also important to note is that authentication keys are tied to the user's password, so a fully compromised user account could negate the benefits provided by EFS.

I think that many companies don't employ EFS because you leave the decision on what documents to encrypt up to the user. This also means that you depend on them to remember to do the encryption in the first place, which means they will have to understand the importance of it in order to make it worthy of their time. I wanted to mention EFS because it is still alive and is still a valid platform for which you can encrypt data, but most administrators are landing on BitLocker as a better solution. Lack of responsibility on the user's part and a good centralized management platform do put BitLocker a solid step ahead of EFS. Both the technologies could certainly co-exist, though, keeping data safe at two different tiers instead of relying on only one of the data encryption technologies available to you.

IPsec

A lot of the encryption technology built into OSes revolves around data at rest. But what about our data on the move? We talked about using SSL on HTTPS websites as a way of encrypting web browser data that is on the move across the internet, but what about data that is not flowing through a web browser?

And what if I'm not even concerned about the internet; what if I am interested in protecting traffic that could even be flowing from point to point *inside* my corporate network? Is there anything that can help with these kinds of requirements? Certainly.

IPsec is a protocol suite that can be used for authenticating and encrypting the packets that happen during a network communication. IPsec is not a technology that is particular to the Microsoft world, but there are various ways in Windows Server 2019 that IPsec can be utilized in order to secure data that you are shuttling back and forth between machines.

The most common place that IPsec interaction shows up on a Windows Server is when using the Remote Access role. When configuring VPN on your RA server, you will have a number of different connection protocols that the VPN clients can use to connect to the VPN server. Included in this list of possible connection platforms are IPsec (IKEv2) tunnels. The second remote access technology that uses IPsec is DirectAccess. When you establish DirectAccess in your network, every time that a client computer creates a DirectAccess tunnel over the internet to the DirectAccess server, that tunnel is protected by IPsec.

Thankfully the Remote Access Management Console that you use to deploy both VPN and DirectAccess is smart enough to know everything that is needed to make IPsec authentication and encryption work, and you don't need to know a single thing about IPsec in order to make these remote access technologies work for you!

The big missing factor with IPsec provided by the Remote Access role is traffic *inside* your network. When you are talking about VPN or DirectAccess, you are talking about traffic that moves over the internet. But what if you simply want to encrypt traffic that moves between two different servers inside the same network? Or the traffic that is flowing from your client computers inside the office to their local servers, also located in the office? This is where some knowledge of the IPsec policy settings comes in handy, because we can specify that we want traffic moving around inside our corporate networks to be encrypted using IPsec. Making that happen is a matter of putting the right policies into place.

Configuring IPsec

There are two different places that IPsec settings can be configured in a Microsoft Windows environment. Both old and new systems can be supplied with IPsec configurations through the traditional **IPsec Security Policy snap-in**. If you are running all systems that are newer, such as Windows 7 and Server 2008 and above, then you can alternatively employ the **Windows Defender Firewall with Advanced Security** for setting up your IPsec policies. WFAS is the most flexible solution, but isn't always an option depending on the status of legacy systems in your environment.

First, let's take a glance at the older IPsec policy console. We will start here because the different options available will help to build a baseline for us to start wrapping our minds around the way that IPsec interaction works between two endpoints. There are three different classifications of IPsec policy that can be assigned to your machines that we will encounter in this console. Let's take a minute to explain each one, because the policy names can be a little bit misleading. Understanding these options will put you a step ahead for understanding how the settings inside WFAS work as well.

Server policy

The server policy should probably be renamed to *Requestor* policy, because that is really what this one does. When a computer or server makes a network request outbound to another computer or server, it is requesting to establish a network connection. On these requesting computers—the ones initiating the traffic—this is where we tell the IPsec Server policy to apply. Once applied, the server policy tells that computer or server to request IPsec encryption for the communication session between the initiating machine and the remote computer. If the remote system supports IPsec, then the IPsec tunnel is created in order to protect the traffic flowing between the two machines. The Server policy is pretty lenient though, and if the remote computer does not support IPsec, then the network connection is still successful, but remains unencrypted.

Secure Server policy

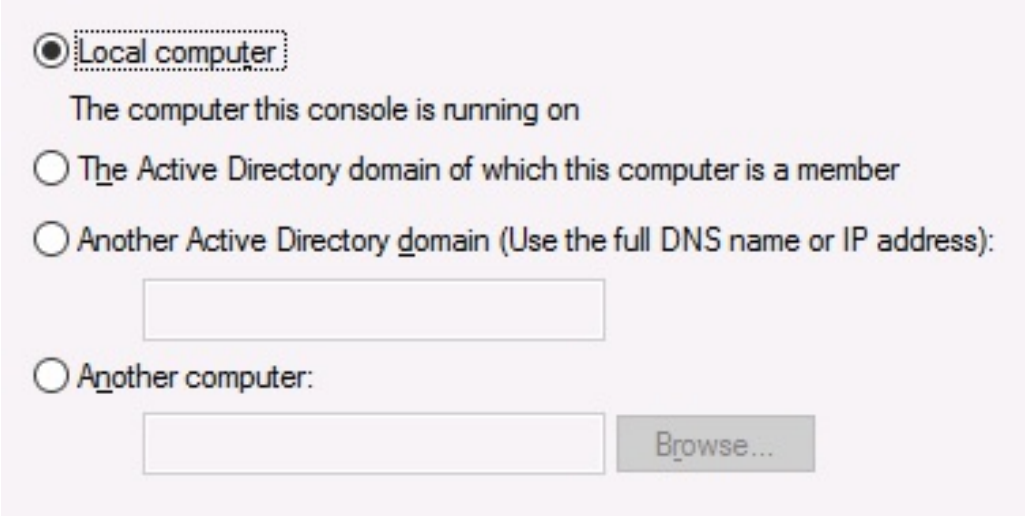
The difference here is that the Secure Server policy requires IPsec encryption in order to allow the network communication to happen. The regular server policy that we talked about earlier will encrypt with IPsec when possible, but if not possible it will continue to flow the traffic unencrypted. The Secure Server policy, on the other hand, will fail to establish the connection at all if IPsec cannot be negotiated between the two machines.

Client policy

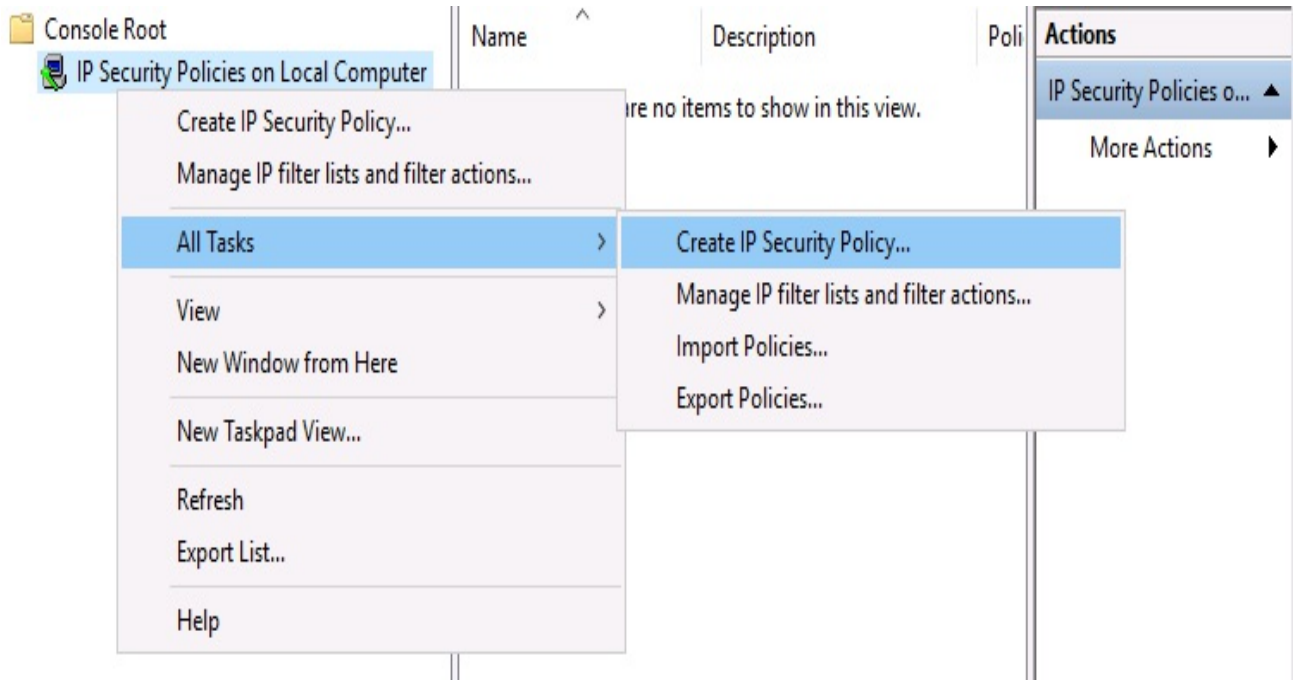
The Client policy needs to be renamed to *Response* policy, because this one is on the other end of the connection. The Client policy does not care about requesting an IPsec session, it only cares about **receiving** one. When a computer makes a network request to a server, and that computer has the Server or Secure Server policy so it is requesting IPsec, then the server would need to have the Client policy assigned to it in order to accept and build that IPsec tunnel. The Client policy responds by allowing the encryption to happen on that session.

IPsec Security Policy snap-in

The original console for manipulating IPsec settings is accessed via MMC. Open that up, and add the IP Security Policy Management snap-in. Interestingly, when adding this snap-in you will notice that you can view either the local IPsec policy of the machine, which you are currently logged in to, or you can open the IPsec policy for the domain itself. If you are interested in configuring a domain-wide IPsec implementation, this would be your landing zone for working on those settings. But for the purposes of just sticking our head in here to poke around a little, you can choose the Local computer in order to take a look at the console:

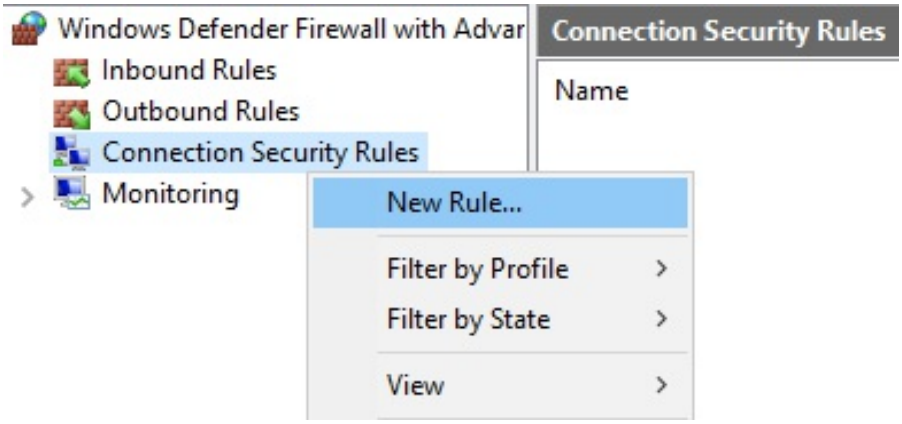


Once inside, you can see any existing IPsec policies that might be in place, or you can start creating your own by using the Create IP Security Policy... action available when right-clicking on IP Security Policies. Doing this will invoke a wizard that will walk through the configuration of your particular IPsec policy:

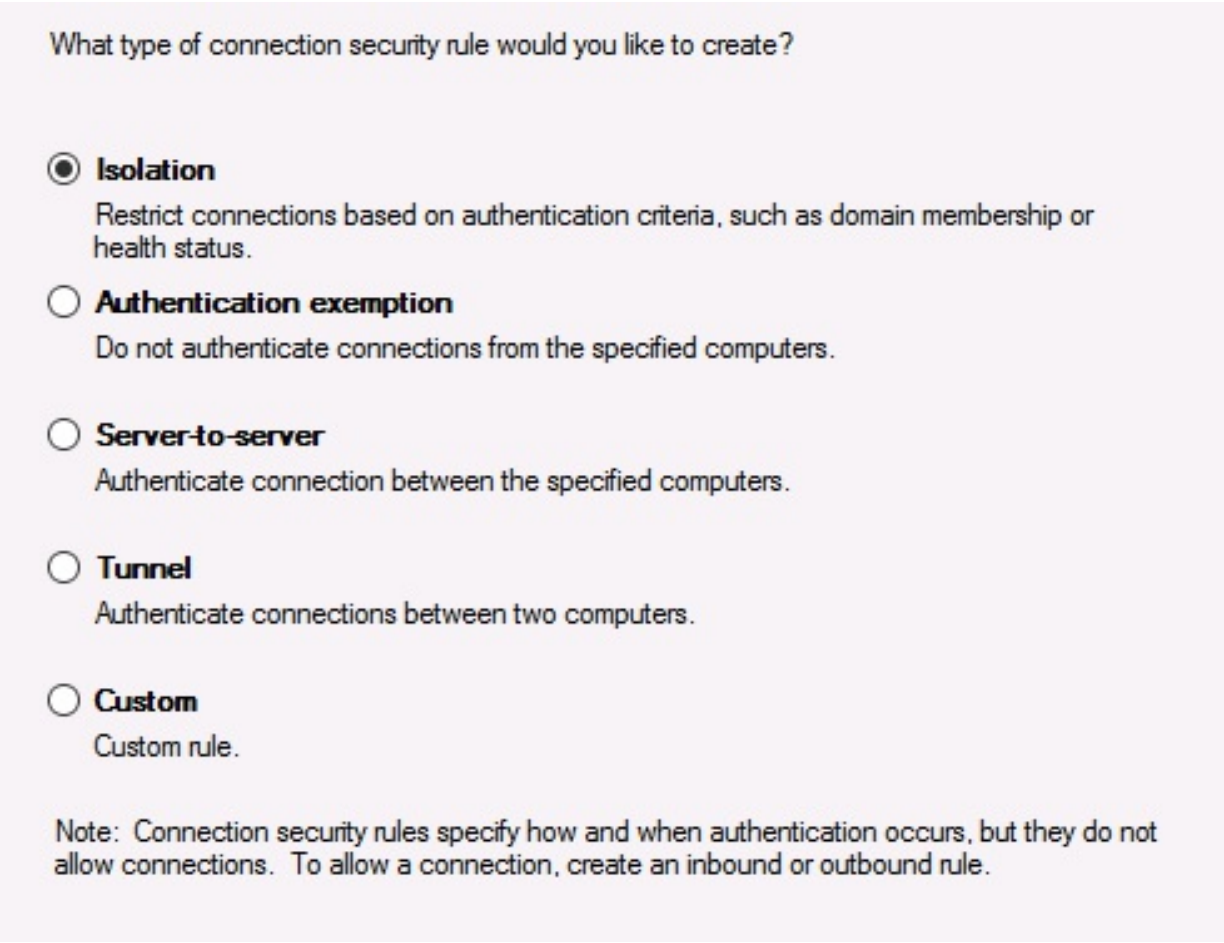

Console Root
IP Security Policies on Local Computer
Name
Description
Poli
re no items to show in this view.
Actions
IP Security Policies o...
More Actions
Create IP Security Policy...
Manage IP filter lists and filter actions...
All Tasks
View
New Window from Here
New Taskpad View...
Refresh
Export List...
Help
Create IP Security Policy...
Manage IP filter lists and filter actions...
Import Policies...
Export Policies...

Using WFAS instead

The newer platform used for establishing IPsec connection rules is the Windows Defender Firewall with Advanced Security. Go ahead and open that up, as you are already familiar with doing. Once inside, navigate to the Connection Security Rules section, which is listed immediately below Inbound Rules and Outbound Rules. **Connection Security Rules**, is where you define IPsec connection rules. If you right-click on Connection Security Rules and choose New Rule... you will then walk through a wizard that is similar to the one for creating a firewall rule:



Once inside the wizard to create your new rule, you start to see that the options available to you are quite different from the ones shown when creating a new firewall rule. This is the platform from which you will establish IPsec connection security rules that define what the IPsec tunnels look like, and on which machines or IP addresses they need to be active:



We do not have space here to cover all of the available options in this wizard, but I definitely recommend picking up from here and taking it a step further with some added knowledge on TechNet, as given here: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/hh831807(v=ws.11).

Banned passwords

If you are an Azure Active Directory customer, you already have access to this new function called **banned passwords**. The idea is this: Microsoft maintains a global ongoing list of commonly bad passwords (such as the word *password*), and automatically blocks all variants of password such as *P@ssword*, *Password123*, and so on. Any of these potential passwords would be blocked altogether if a user ever tried to create one as their own password. You also have the ability to add your own custom banned passwords inside the Azure Active Directory interface. Once you have banned passwords up and running in Azure, this capability can then be ported to your on-premises Active Directory environment as well, by implementing the Azure Active Directory password protection proxy service (whew, that's a mouthful). This proxy interfaces between your on-premises Domain Controllers and your Azure Active Directory, ensuring that passwords that users attempt to put into place on your local Domain Controllers are fitting within the rules defined by Azure's banned password algorithms.

In order to use this technology, you must of course be utilizing Azure Active Directory, so this isn't for everyone. However, if you do have and sync to Azure Active Directory, then this capability is even backported to older versions of on-premises Domain Controllers. These servers can be as old as Windows Server 2012.

Here is a link to further information on banned passwords: https://docs.microsoft.com/en-us/azure/active-directory/authentication/concept-password-ban-bad-on-premises.

Advanced Threat Analytics

In my opinion, one of the coolest security features to come out of Microsoft over the past few years is **Advanced Threat Analytics** (**ATA**), and yet I hardly hear anyone talking about it. It's not a feature or function built into the Windows Server OS, not yet anyway, but is an on-premises software that rides on top of Windows to produce some amazing functionality. Essentially, what ATA does is monitor all of your Active Directory traffic, and warns you of dangerous or unusual behavior in real time, immediately as it is happening.

The idea of ATA is pretty simple to understand and makes so much common sense that it's something we are all going to wonder why it took so long to put into place. The reason for that, though, is because under the hood the processing and learning that ATA is doing is very advanced. Yes, I said learning. This is the coolest part of ATA. You configure your network so that all of the traffic flowing in or out of your Domain Controllers also lands onto the ATA system. The most secure way to accomplish this is at the networking level, establishing port mirroring so that all of the Domain Controller packets also make their way to ATA, but at a level that an attacker would not be able to see. This way, even if someone nefarious is inside your network and is on the lookout for some kind of protections working against them, ATA remains invisible to their prying eyes. However, port mirroring that traffic is something that smaller companies may not be able to do, or may be too complex for an initial setup, and so a second option exists to install an ATA lightweight agent right onto the Domain Controllers themselves. This agent then sends the necessary information over to the ATA processing servers.

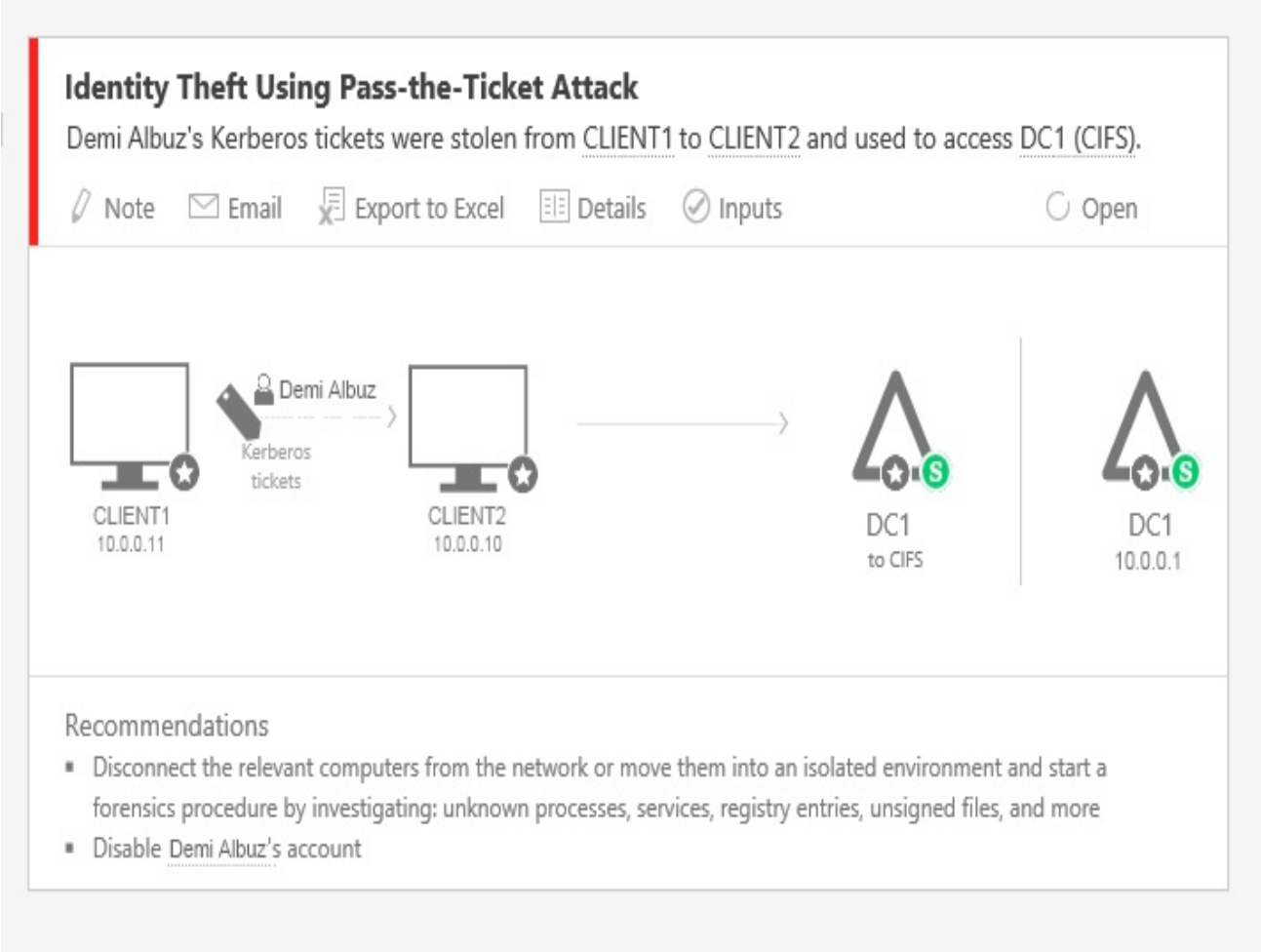
In either case, those ATA processing servers receive all of this data, and start finding patterns. If Betty uses a desktop computer called **BETTY-PC** and a tablet called **BETTY-TABLET**, ATA will see that pattern and associate her user account with those devices. It also watches for her normal traffic patterns. Betty usually logs in around 8 a.m. and her traffic usually stops somewhere around 5 p.m. She typically accesses a few file servers and a SharePoint server. After a week or so of collecting and monitoring data, ATA has a pretty good idea of Betty's standard MO.

Now, one night, something happens. ATA sees a bunch of password failures against Betty's account. That in itself might not be something to get too excited about, but then all of a sudden Betty logs into a terminal server that she doesn't typically access. From there, her credentials are used to access a Domain Controller. Uh oh, this clearly sounds like an attack to me. With the tools built into Active Directory that we currently have at our disposal, what do we know? Nothing, really. We might see the password failures if we dig into the event logs, and based on that we could try poking around other servers' event logs in order to find out what that account was accessing, but we really wouldn't have any reason to suspect anything. This could be the beginning of a very large breach, and we would never see it. Thankfully, ATA knows better.

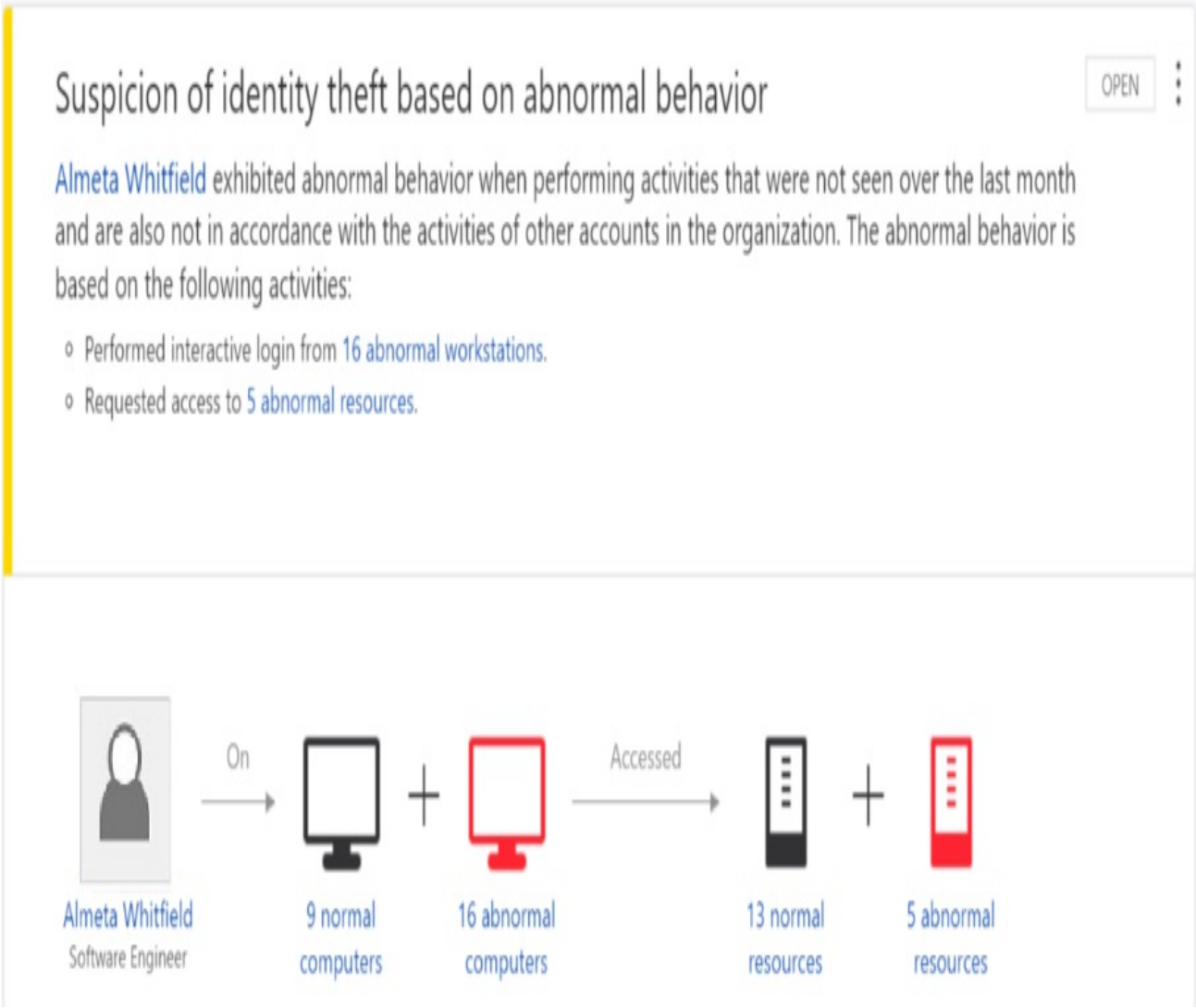
The management interface for ATA is like a social media feed, updated almost in real time. During the events I have just laid out, if we had been looking at the ATA media feed, we would have seen all of these items, which I pointed out happen, as they happened, and it would be immediately obvious that someone compromised Betty's account and used it to gain access to a Domain Controller. There has never been a technology that watches Active Directory traffic so intensely, and there has certainly never been anything that learns patterns and behavioral diversions like this. It is truly an amazing technology, and I don't say that only because I happen to know the guys who built it. But since I do, I can tell you that they are brilliant, which is already pretty obvious

since Microsoft scooped them up.

At this point, ATA is still new enough that most of the IT community hasn't had any interaction with it, and I strongly encourage you to change that. It may save your bacon one day. The following is a screenshot of the ATA web interface so you can get a visual on that social media-style feed. This screenshot was taken from a Microsoft demo where they purposefully stole the Kerberos ticket from a user, and then utilized it on another computer in order to access some confidential files that only Demi Albuz should have been able to access. While ATA did not stop this activity, it immediately—and I mean within seconds—alerted inside this feed to show the **Pass-the-Ticket Attack**:



Here's another example where a user named Almeta Whitfield suddenly accessed 16 computers that she does not usually access, another big red flag that something is going on with her user account:



For more information or to get started using ATA, make sure to check out the following link: https://docs.microsoft.com/en-us/advanced-threat-analytics/what-is-ata.

General security best practices

Sometimes we need only to rely on ourselves, and not necessarily functionality provided by the OS, to secure our systems. There are many common-sense approaches to administratorship (if that is a word) that are easy to accomplish but are rarely used in the field. The following are a few tips and tricks that I have learned over the years and have helped companies implement. Hopefully, you as the reader have even more to add to this list as to what works well for you, but if nothing else this section is intended to jog your thinking into finding creative ways with which you can limit administrative capability and vulnerability within your network.

Getting rid of perpetual administrators

Do all of your IT staff have domain admin rights the day they are hired? Do any of your IT staff have access to the built-in domain administrator account password? Do you have regular users whose logins have administrative privileges on their own computers? You know where I'm going with this—these are all terrible ideas!

Unfortunately, that was all the status quo for many years in almost every network, and the trend continues today. I still regularly watch engineers use the **administrator** domain account for many tasks when we set up new servers. This means they not only have access to potentially the most important account in your network and are using it for daily tasks, but it also means that anything being set up with this user account is not accountable. What do I mean by that? When I set up a new server or make changes to an existing server using the general administrator account, and I end up causing some kind of big problem, nobody can prove that I did it. Using generalized user accounts is a sure way to thwart responsibility in the event that something goes wrong. I'm not trying to imply that you are always on the lookout for *who did that?*, but if I mess something up on an application server that I don't normally administer, it would be nice if the guys trying to fix it could easily figure out that it was me and come ask me what I did so that they can reverse it. There are many reasons that using the built in administrator account should be off limits for all of us.

To address the client side, do your users really need administrative

rights on their computers? *Really*? I think you could probably find ways around it. Bringing regular users down to user or even power user rights on their systems can make a huge impact on the security of those computers. It gives viruses a much harder time installing themselves if the user needs to walk through a prompt asking for admin privileges before they can proceed with the install. It also keeps all of your machines in a much more consistent behavioral pattern, without new and unknown applications and settings being introduced by the user.

Using distinct accounts for administrative access

This idea piggy backs off the last one, and is something that I have started employing even on all of the home computers that I install for friends and family members. It really boils down to this: utilize two different user accounts. One with administrative access, and one without. When you are logged in for daily tasks and chores, make sure that you are logged in with your regular user account that does not have administrative privileges, either on the local computer or on the domain. That way, if you attempt to install anything, or if something attempts to install itself, you will be prompted by the **User Account Control** (**UAC**) box, asking you to enter an administrative username and password before the installer is allowed to do anything. I can tell you that this works, as I have stopped a number of viruses on my own computer from installing themselves as I'm browsing around the internet trying to do research for one project or another. If I get a UAC prompt asking me for an admin password and I haven't clicked on an installer file, I know it's something I don't want. All I have to do is click on **No**, and that installer will not get a hold of my computer. On the other hand, if it is something that I am intending to install, then it is a minor inconvenience to simply enter the password of my administrative account, and allow the installer to continue.

Maintaining two separate accounts allows you to work your way through most daily tasks while putting your mind at ease that you do not have rights to inadvertently do something bad to your system. This mindset also limits the amount of activity that any administrative

account needs to take on a computer or on the network, and makes those administrative accounts easier to track when admins are making changes in the environment.

Using a different computer to accomplish administrative tasks

If you want to progress even further on the idea of separate user accounts, you could make your computing experience even more secure by utilizing a separate computer altogether when accomplishing administrative-level tasks. One computer for regular knowledge worker tasks, and another computer for administration. This would certainly help to keep your administrative system secure, as well as the remote systems that it has access to. And while it does seem cumbersome to have two physical computers at your desk, remember that with most SKUs in Windows 10 we have the ability to run Hyper-V right on our desktop computers. I actually do exactly this with my own computer. I have my computer that is running Windows 10, and then inside that computer I am running a virtual machine via Hyper-V from which I do all administrative tasks on the sensitive servers. This way a compromise of my day-to-day OS doesn't necessitate a compromise of the entire environment.

Whether you choose to split up administrative access at the user account level or the computer level, remember this simple rule: **never administer Active Directory from the same place that you browse Facebook**. I think that pretty well sums this one up.

Never browse the internet from servers

Seems like a no brainer, but everyone does it. We spend all day working on servers, and very often have to reach out and check something from a web browser. Since Internet Explorer exists on Windows Servers, sometimes it is just quicker and easier to check whatever it is that we need to check from the server console where we are working, rather than walk back over to our desks. Resist the temptation! It is so easy to pick up bad things from the internet, especially on servers because if any machines in our network are running without antivirus protection, it is probably on the server side. The same is true for internet filters. We always make sure that the client traffic is flowing through our corporate proxy (if we have one), but we don't always care whether or not the server traffic is moving outward the same way.

Don't even do it for websites that you trust. A man-in-the-middle attack or a compromise of the website itself can easily corrupt your server. It's much easier to rebuild a client computer than it is a server.

Role-Based Access Control (RBAC)

The phrase **Role-Based Access Control** (**RBAC**) is not one that is limited to Microsoft environments. It also is not a particular technology that can be utilized inside Windows Server 2019, but rather it is an ideology that is all about separating job roles and duties. When we think about separating our employees' job roles from an IT perspective, we traditionally think in terms of Active Directory groups. While adding user accounts to groups does solve many problems about splitting up levels of permissions and access, it can be complicated to grow in this mentality, and ultimately AD groups still empower administrators with full access to the groups themselves. RBAC technologies divide up roles at a different level, caring about more than permissions. RBAC focuses more on employee job descriptions than access restrictions. There are a number of different technologies that take advantage of RBAC tools integrated into them, and there are even third-party RBAC solutions that ride on top of all your existing infrastructure, making it widely accessible across your entire organization, and not restricted to working in the confines of a single domain or forest.

Just Enough Administration (JEA)

A great example of an RBAC technology that is included in Windows Server 2019 is **Just Enough Administration** (**JEA**), which is part of PowerShell. JEA provides you with a way to grant special privileged access for people, without needing to give them administrative rights, which would have been required to accomplish the same duties in the past. The necessity to add someone to the administrators group on a server so that they can do their job is quite common, but JEA is a first step away from that necessity.

In our old way of thinking, it might be easy to think of JEA as doing something like allowing users to have administrative access within PowerShell even when they don't have administrative access to the OS itself, but it's even more powerful than that. The design of JEA is such that you can permit users to have access only to run particular PowerShell commands and cmdlets at an administrative level, leaving other commands that they do not need access to in the dark.

In fact, if a user is working within a JEA context of PowerShell and they try to invoke a cmdlet that is not part of their **allowed** cmdlets, PowerShell pretends as though it doesn't even recognize that cmdlet. It doesn't say, *sorry, you can't do this*—it just ignores the command! This definitely helps to keep prying fingers out of the cookie jar, unless you want to let them in.

Let's take it a step further. Maybe you are a DNS administrator, and you might occasionally need to restart the DNS services. Since we are adopting the JEA/RBAC mentality, you are not going to have

administrative rights on the OS of that DNS server, but you will have JEA-based rights within PowerShell so that you can run the tools that you need in order to do your work. Restarting the DNS service requires access to use the `Restart-Service` cmdlet, right? But doesn't that mean I would be able to restart any service on that server, and could potentially do all sorts of things that I don't need to do? JEA is even powerful enough to deal with this scenario. When setting up the level of access that the user needs to get, you can even dive into particular cmdlets and divide up permissions. In our example, you could provide the user with access to the `Restart-Service` cmdlet, but only give permissions to restart particular services, such as those pertaining to DNS. If the user tried to `Restart-Service` on WINrm, they would be denied.

Summary

The number-one agenda item for many CIOs this year is security. Security for your client machines, security for your networks, security for your cloud resources, and most importantly security for your data. There is no single solution to secure your infrastructure, it requires many moving parts and many different technologies all working together in order to provide safety for your resources. The purpose of this chapter was to provide examples of security measures and technologies that can be utilized in anyone's environments, as well as to reprioritize the importance that security has in today's IT world. Concerns about privacy and security need to be discussed for any and every technology solution that we put into place. Too many times do I find new applications being implemented inside organizations without any regard to how secure that application platform is. Applications that transmit or store data unencrypted need to be modified or dumped. Protection of information is essential to the longevity of our businesses.

We cannot complete a discussion about security in Windows Server 2019 without discussing the default OS installation option that we have thus far ignored throughout this book. Turn the page and let's dive into Server Core, our headless and less vulnerable version of Windows Server.

Questions

1. What is the name of the anti-malware product built into Windows Server 2019?
2. When a domain-joined computer is sitting inside the corporate LAN, which Windows Defender Firewall profile should be active?
3. Other than the Domain profile, what are the other two possible firewall profiles inside Windows Defender Firewall?
4. When creating a firewall rule to allow IPv4 ping replies, what protocol type must you specify inside your inbound rule?
5. What is the easiest way to push standardized Windows Defender Firewall rules to your entire workforce?
6. A virtual machine whose virtual hard disk file is encrypted is called a...?
7. What is the name of the Microsoft technology that parses domain controller information in order to identify pass-the-hash and pass-the-ticket attacks?

Server Core

Honey, I shrunk the server! Another chapter, another outdated movie reference. Over the past 20 years or so, we have seen nothing but growth out of the Microsoft operating systems. Growth can be good; new features and enhancements make our lives easier. Growth can also be bad, such as bloated file structures and memory-hogging graphical interfaces. If you were to chronologically graph the Windows and Windows Server operating systems in terms of their footprints, based on factors such as disk-space consumption and memory requirements, it would show a steady upward slope. Every new release requires just a little more processing power, and just a little more hard drive space than the previous version. That was the case until, I'm guesstimating a little bit here, maybe Windows 8 and Server 2012. We saw some surprising steps taken with lowering these threshold numbers, a welcome change. But the change wasn't too dramatic. I mean, what can you glean from the fact that a new Windows Server 2019 box contains all kinds of core items still running inside `C:\Windows\System32`? We're not even going to talk about what's in the registry. Clearly, there are still cutbacks that could be made, and at some level, new operating systems are still just being built and patched on top of the old ones.

Until now, perhaps. Here, we are going to talk about an alternate way to use Windows Server 2019 on a much, much smaller scale. Server Core has been around for quite some time now, but I'm hard-pressed to find people that actually use it. This miniaturized version of Server 2019 has been built to provide you a smaller, more efficient, and more secure server platform.

We will cover the following topics in this chapter:

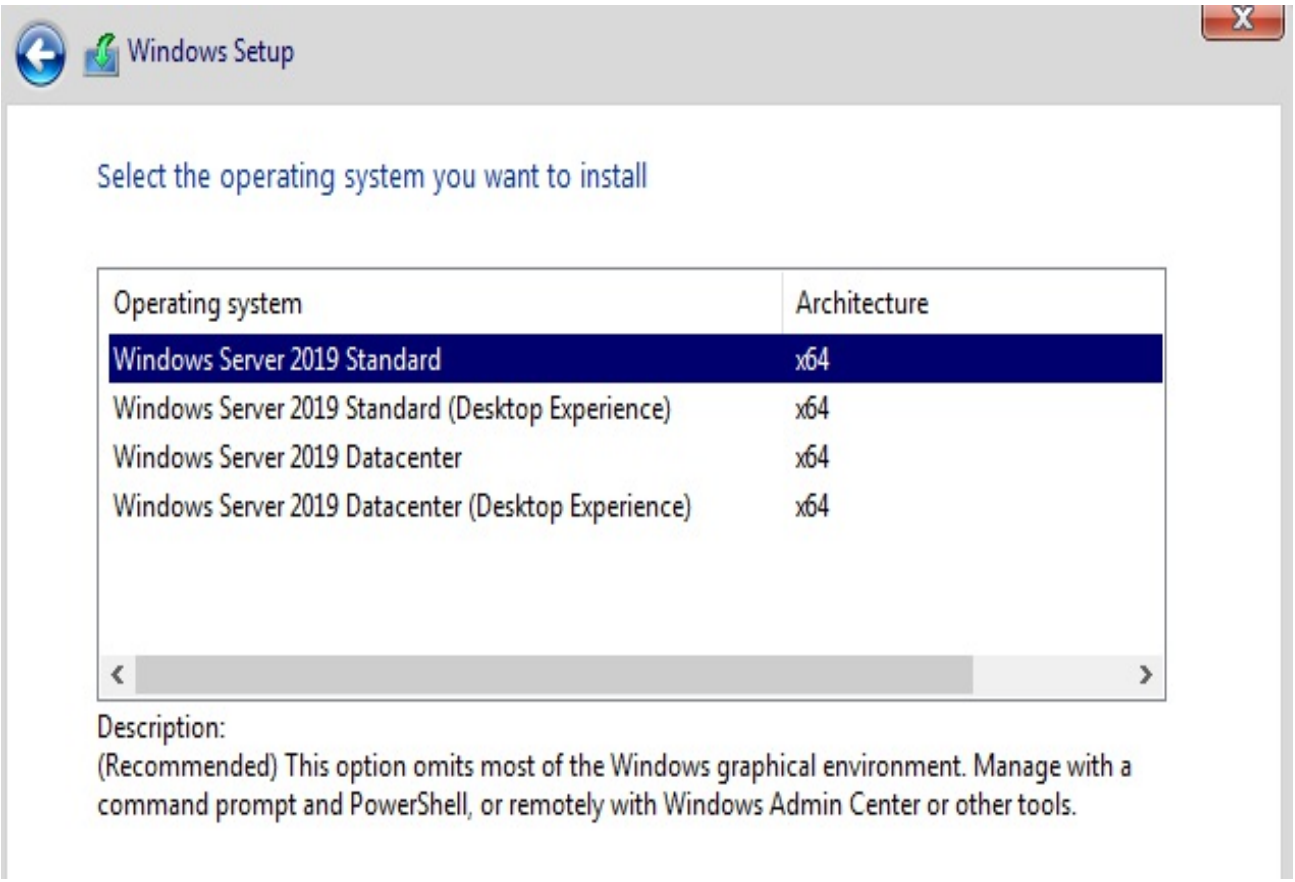
- Why use Server Core?
- Interfacing with Server Core
- Windows Admin Center for managing Server Core
- The Sconfig utility
- Roles available in Server Core
- What happened to Nano Server?

Why use Server Core?

Why am I even talking about Server Core? Hasn't it been around since 2008? Yes, that is exactly why I *am* talking about it. The Server Core variant of the Windows Server operating system has been around for quite some time, but it seems like many admins are scared to trust it. I work with many different companies from many different industries. They all have one big thing in common: they use a lot of Windows Servers, and all of these Windows Servers are running the full GUI (Desktop Experience). Have they heard of Server Core? Sure. Have they tested it out in a lab? Sometimes. Everyone seems to have a slightly different experience level with Core, but it's quite rare to find one in production. Maybe I'm just talking to the wrong people, but I have to assume that there is a majority of us out there, myself included, that need to start using Server Core on a more regular basis.

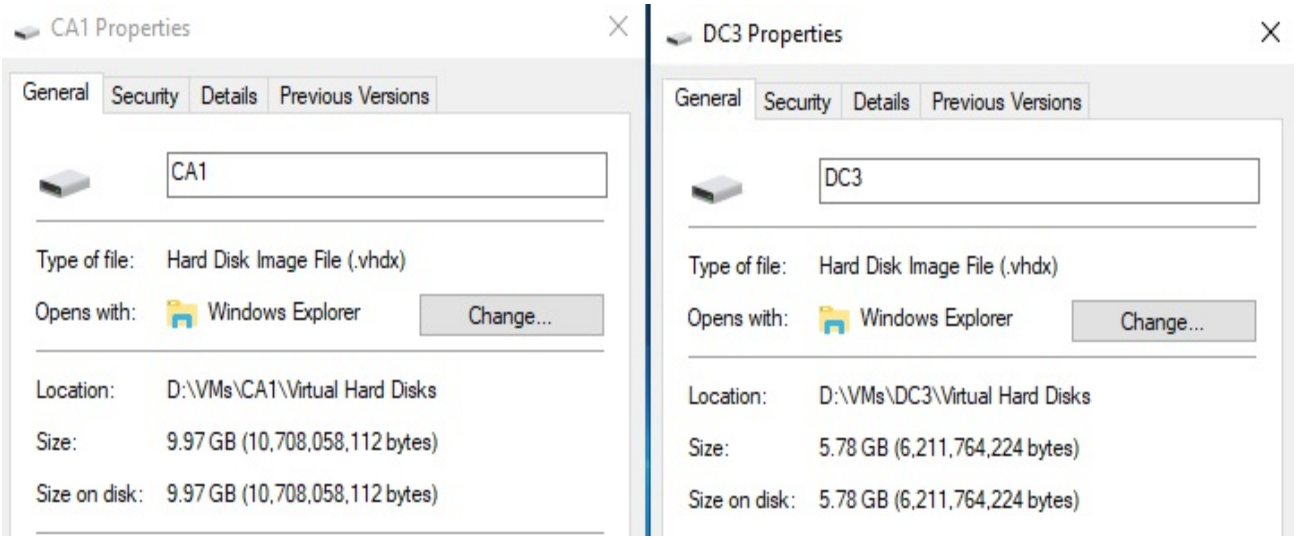
Why do we need to start using Server Core? Because GUI-less servers are the future, says Microsoft. Would you believe that early in the previews for Windows Server 2016, the Desktop Experience option didn't even exist? You couldn't run a full GUI desktop shell on a Server 2016 if you wanted to, save for a quasi-, mini shell that could be plopped on top of Server Core. Microsoft received so much flak about this that the full Desktop Experience was added back during one of the Technical Preview rollouts. Even so, since that time, you have probably noticed that Server Core is the default option when installing any Windows Server operating system. Remember, back at the beginning of our book, where we did a quick review of the actual Server 2019 installation? The default option for installation is not Desktop Experience; rather, that top option in the following screenshot is the option for installing the

command-line-driven Server Core:



One of the reasons for a move away from the graphical interface is increased capabilities for automation and scalability. When all of our servers are built similarly, it means that we can do more cloud-like functions with them. Automatic spinning up and down of resources as they are needed, rolling out dozens of servers at the click of a switch—this kind of automation and sizing is possible in the cloud, but is only possible because the infrastructure is set up in a way that it is so standardized. Cloud hardware resources need to be so streamlined that the operations and automation tools can make them do what is needed, without worrying about all of the variables that would be present in a user-tweaked graphical interface.

There are other obvious advantages to running all of your servers as this limited, restricted version. Server Core boasts reduced hard drive space, reduced memory consumption, and a reduced attack surface when compared to a traditional, full-blown server experience. Now you can see why I made the hefty statements a minute ago about how we all need to start becoming more comfortable with Server Core! In fact, let's take a look at that reduced footprint. A base Server 2019 Standard running Desktop Experience consumes around 10 GB of hard drive space; I just verified this by taking a look at the properties of my virtual hard disk file being used by my CA1 server. CA1 is a standard Windows Server 2019 running the full Desktop Experience. Now, I have just finished running through the installation for my first Server Core operating system, and we can see in the following screenshot that the VHDX file being used by this new virtual machine is only 5.8 GB, a 40% reduction in space:



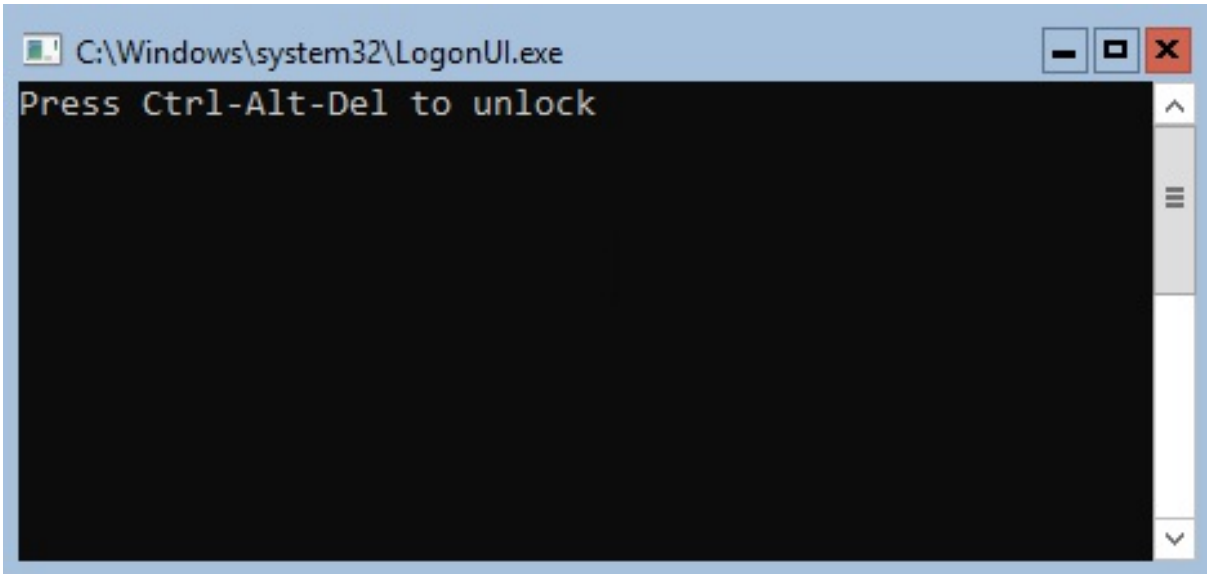
No more switching back and forth

There is a very important note that I wanted to make here: those of you who have worked with Server Core in Windows Server 2012 R2 know that we had the option of changing a server *on the fly*. What I mean is that if you created a new server as the full Desktop Experience, you could later change it to be Server Core. The opposite approach was equally possible; you could take a Server Core and flip it over to a full Desktop Experience.

Not anymore! This capability to move servers back and forth between platforms has been removed. I repeat, this is no longer possible. So plan carefully from here on out when installing these operating systems. If you implement a server as Server Core, that guy is going to remain a Server Core for its lifetime.

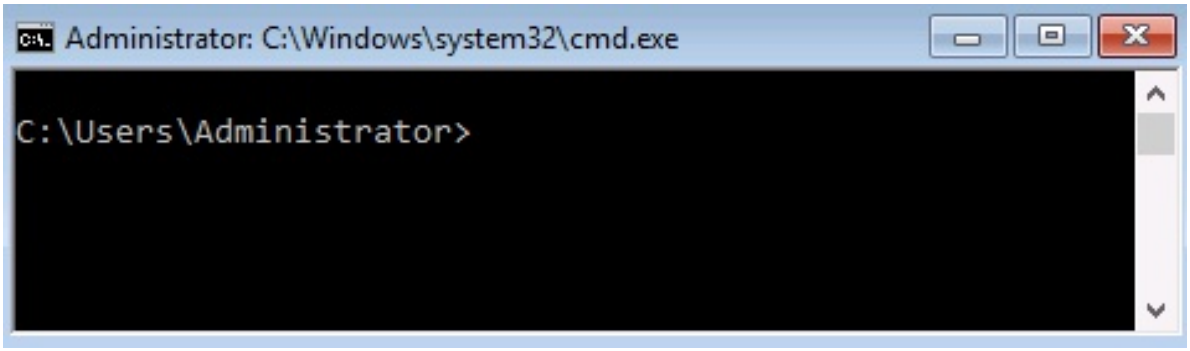
Interfacing with Server Core

After running through your first installation of Server Core, you will be presented with the following lock screen:



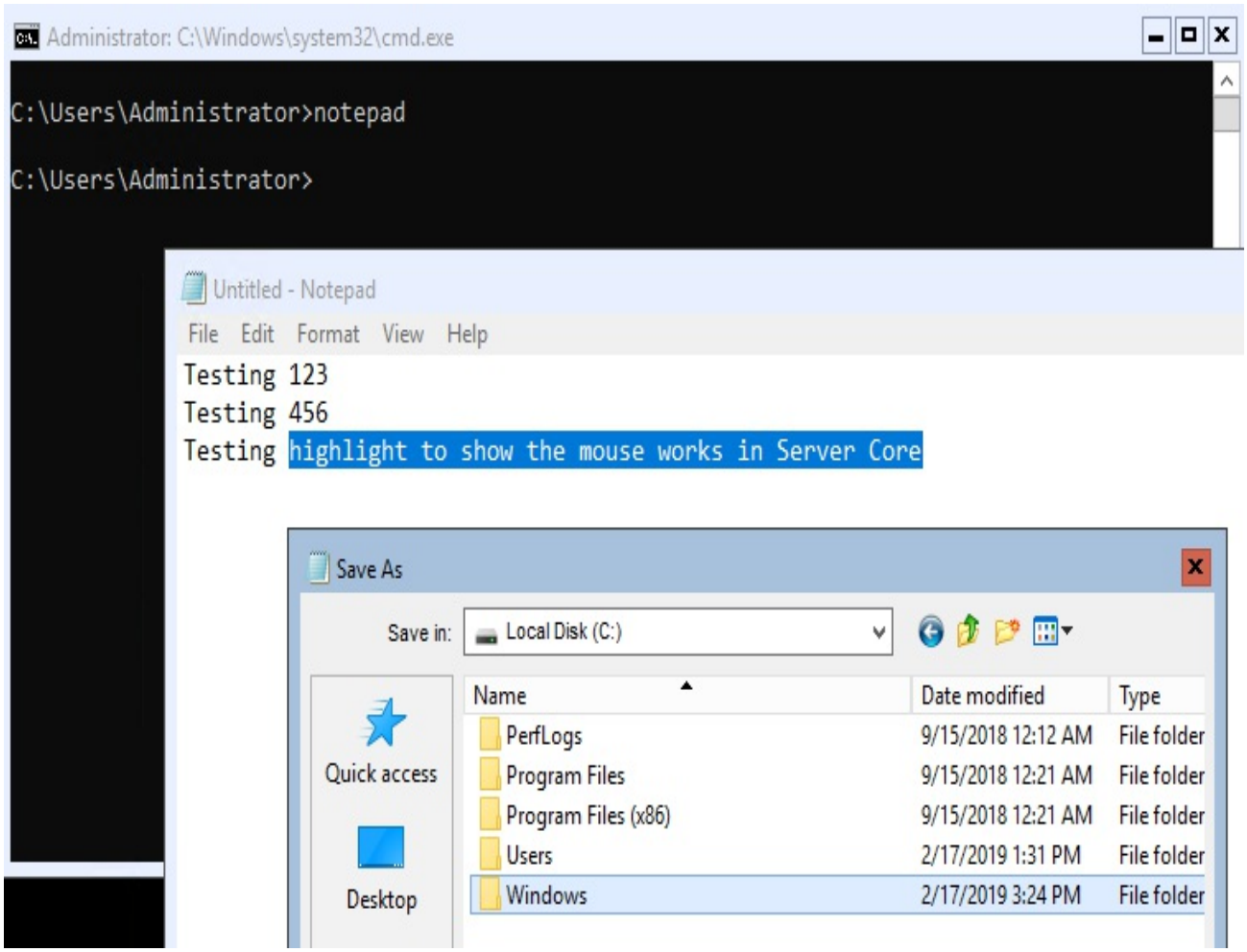
Is that really a Command Prompt window that says `Press Ctrl-Alt-Del to unlock`? Yes, yes it is. This usually gets a few chuckles when an admin sees it for the first time. I know it did for me, anyway. It reminded me a little of when we used to code if/then games on our TI-83 calculators during high school math class. Press *Ctrl + Alt + Del*, and you will be prompted to change your administrator password for the first time, which is the same task that must always be performed first inside GUI versions of Windows Server. Except, of course, that you do it all from within the Command Prompt window using only your keyboard. Once you are officially logged into the server, you will find yourself sitting at a traditional `C:\Windows\system32\cmd.exe` prompt, with a flashing cursor awaiting

instructions:



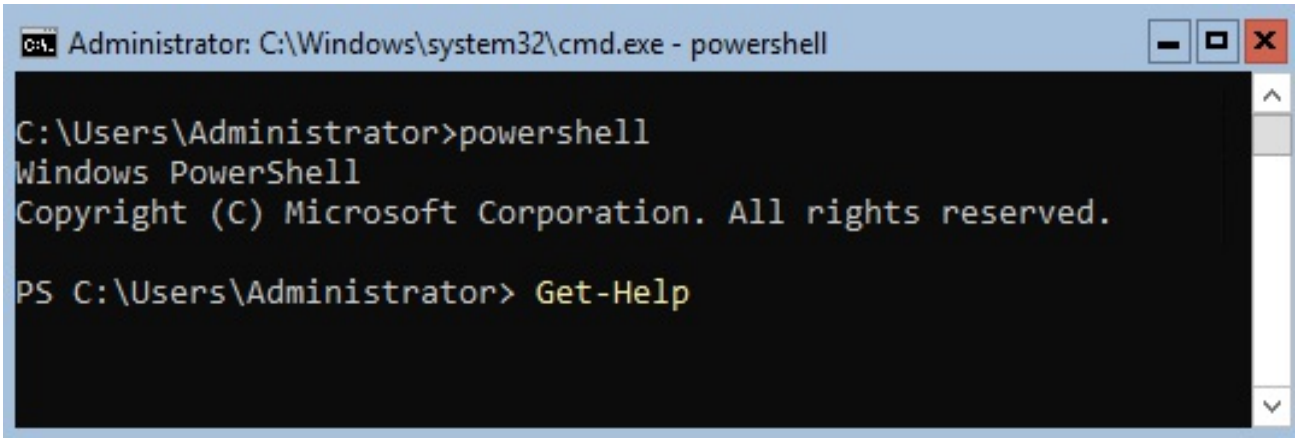
Interestingly, the Command Prompt window isn't consuming the full screen; it is clear that there is a black background that `cmd.exe` is riding on top of. I only find this interesting because you can tell that the Core operating system itself is something other than Command Prompt, and that `cmd.exe` is just an application that autolaunches upon login. You can even utilize the mouse here and resize or move that Command Prompt window around. I do wonder if and when this will be replaced with a PowerShell prompt as the default interface.

Even more interesting and good to know is that you can launch some actual GUI-like applications from this prompt. For example, you can open up Notepad and utilize it with both keyboard and mouse, just like you would from any version of Windows. If you have Notepad open, create a note and then save it; you can see that there is in fact a real file structure and a set of relatively normal-looking system folders. So rather than some form of black magic, Server Core is actually the real Windows Server operating system, wrapped up in a smaller and more secure package:


Administrator: C:\Windows\system32\cmd.exe
C:\Users\Administrator>notepad
C:\Users\Administrator>
Untitled - Notepad
File Edit Format View Help
Testing 123
Testing 456
Testing highlight to show the mouse works in Server Core
Save As
Save in: Local Disk (C:)
Quick access
Desktop
Name
Date modified
Type
PerfLogs 9/15/2018 12:12 AM File folder
Program Files 9/15/2018 12:21 AM File folder
Program Files (x86) 9/15/2018 12:21 AM File folder
Users 2/17/2019 1:31 PM File folder
Windows 2/17/2019 3:24 PM File folder

PowerShell

So, as far as managing a Server Core, you can obviously work straight from the console and use Command Prompt, which appears to be the default interface presented by the OS. In reality though, the commands and functions available from inside Command Prompt are going to be limited. If you are working from the console of a Windows Server Core box, it makes much more sense to use Command Prompt for just one purpose: to invoke PowerShell, and then use it to accomplish whatever tasks you need to do on that server. The quickest way I know to move into PowerShell from the basic Command Prompt is to simply type the `powershell` and press *Enter*. This will bring the PowerShell capabilities right into your existing Command Prompt window, so that you can start interfacing with the PowerShell commands and cmdlets that you need in order to really manipulate this server:



What is the first thing we typically do on new servers? Give them IP addresses, of course. Without network connectivity, there isn't much that we can do on this server. You can assign IP address information to

NICs using PowerShell on any newer Windows Server, but most of us are not in the habit of doing so. Since we can't just open up Control Panel and get into the Network and Sharing Center like we can from inside the Desktop Experience GUI of Windows Server, where do we begin with getting network connectivity on this new Server Core?

Using cmdlets to manage IP addresses

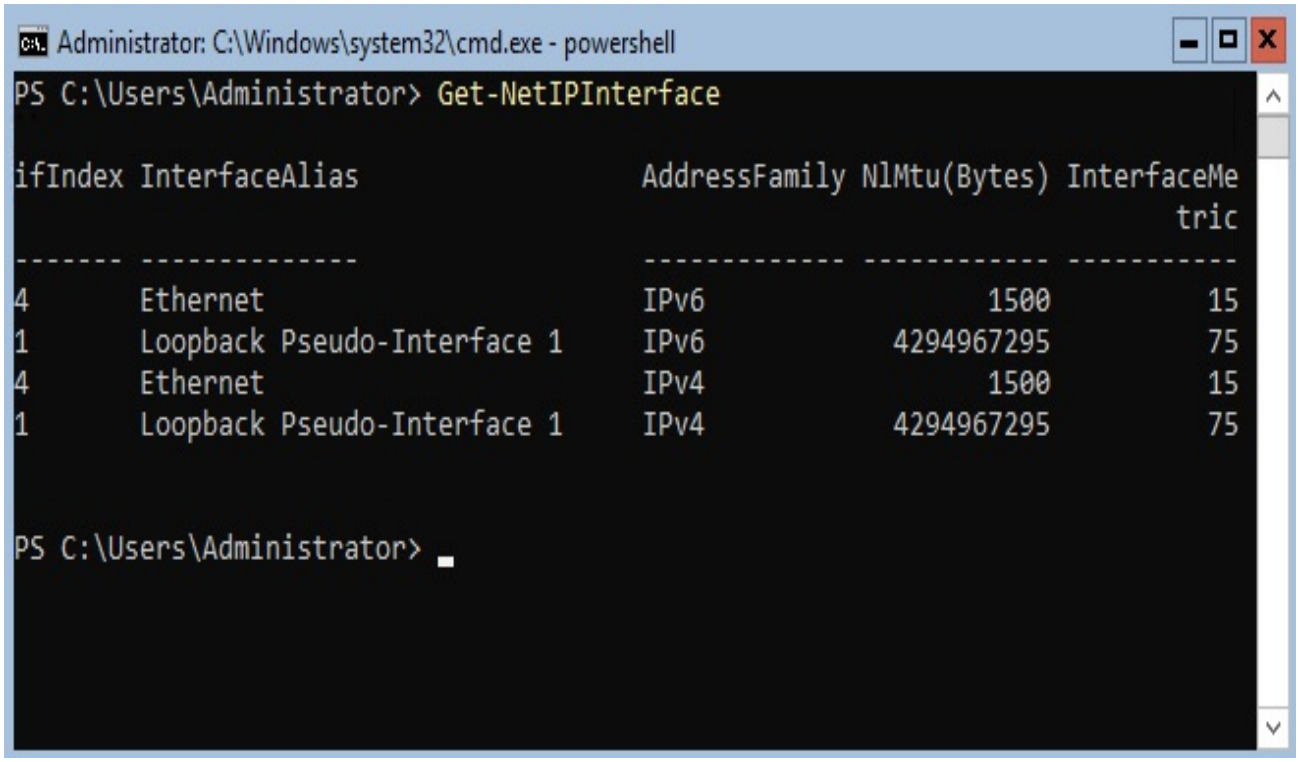
Here are cmdlets that you can use to view and manipulate IP address settings from within PowerShell. Again, these same cmdlets can be used in the full GUI version of Windows Server or from within Server Core.

Currently, working from Server Core where we only have command-line interfacing available to us, these cmdlets are essential to getting network connectivity flowing on our new server:

- `Get-NetIPConfiguration`: This displays the current networking configuration.
- `Get-NetIPAddress`: This displays the current IP addresses.
- `Get-NetIPInterface`: This shows a list of NICs and their interface ID numbers. This number is going to be important when setting an IP address, because we want to make sure we tell PowerShell to configure the right IP onto the right NIC.
- `New-NetIPAddress`: This is used to configure a new IP address.
- `Set-DNSClientServerAddress`: This is used to configure DNS Server settings in the NIC properties.

Let's quickly walk through the setup of a static IP address on a new

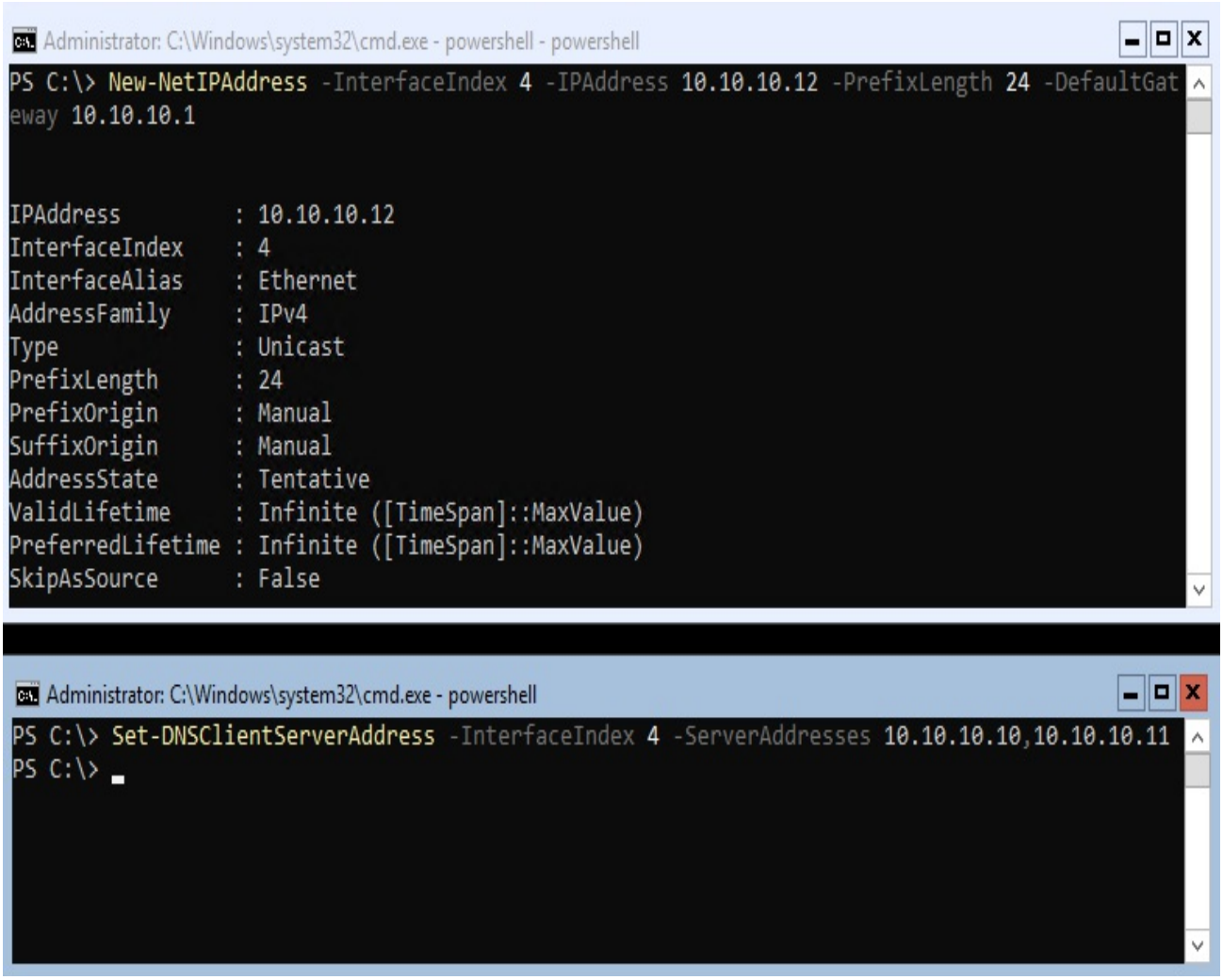
Server Core instance to make sure this all makes sense. I want to assign the `10.10.10.12` IP address to this new server, but first we need to find out which NIC interface ID number it needs to be assigned to. The output of `Get-NetIPInterface` tells us that the `ifIndex` I am interested in is number `4`:



Now that we know the interface number, let's build the commands that are going to assign the new IP address settings to the NIC. I am going to use one command to assign the IP address, subnet mask prefix, and default gateway. I will use a second command to assign DNS server addresses:

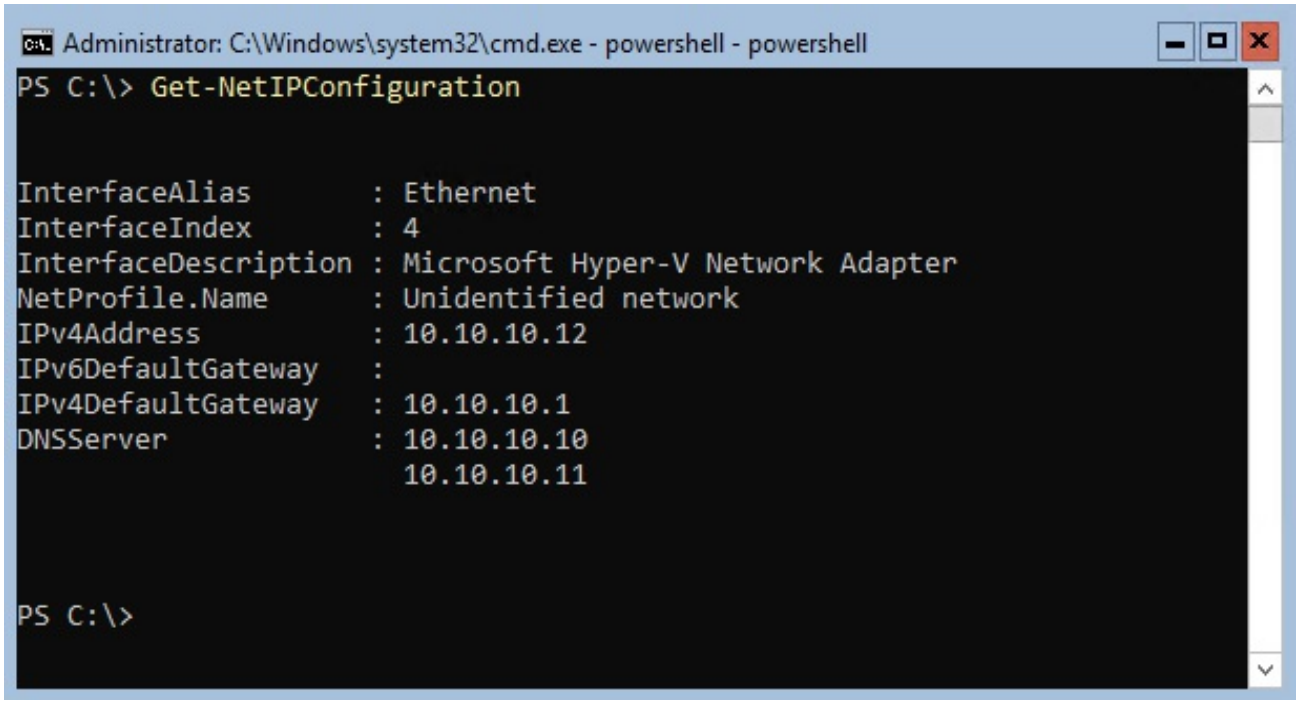
```
New-NetIPAddress -InterfaceIndex 4 -IPAddress 10.10.10.12 -PrefixLength 24 -
DefaultGateway 10.10.10.1

Set-DNSClientServerAddress -InterfaceIndex 4 -ServerAddresses 10.10.10.10,10.10.10.11
```



Hold the phone! How did I get two PowerShell prompts open at the same time within the Server Core interface? Make sure to read the Accidentally closing Command Prompt section later in this chapter to discover how you can launch multiple windows and tools inside the Server Core console.

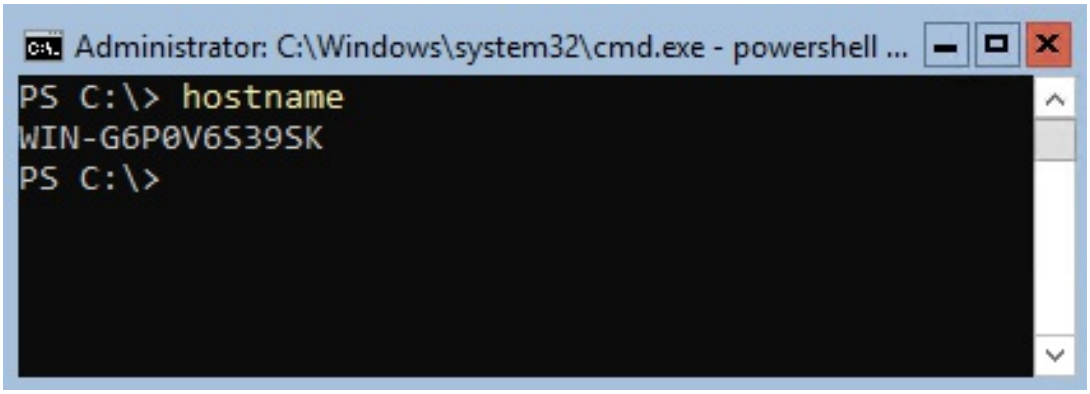
Now all of these IP settings should be in place on the NIC. Let's double-check that with a `Get-NetIPConfiguration` command, seen in the following screenshot. Alternatively, you could use good old `ipconfig` to check these settings, but where's the fun in that?



Remember, you can always utilize DHCP Reservations to make this a little bit easier. If you were to run a simple `ipconfig /all` *from your Server Core and jot down the MAC address of your NIC, you could use this address to create a reservation in DHCP and assign a specific IP address to the new server that way.*

Setting the server hostname

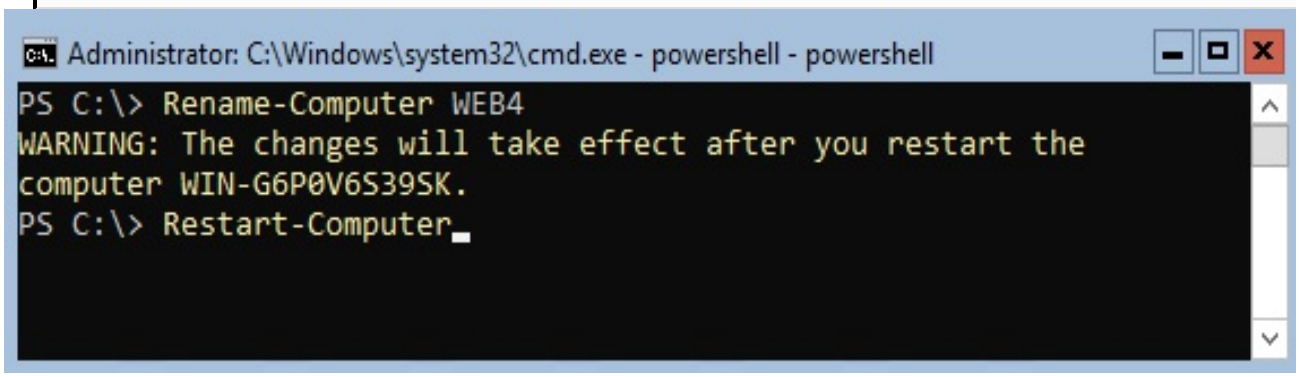
Now that we have network connectivity, a good next step is setting the hostname of our server and joining it to the domain. First things first, let's see what the current name of the server is, and change it to something that fits our standards. When you freshly install Windows, it self-assigns a random hostname to the server. You can view the current hostname by simply typing `hostname` and pressing *Enter*:



To change the hostname of your server, we need to use PowerShell. Bring yourself into a PowerShell prompt if not already there, and all we need to do is use the `Rename-Computer` cmdlet to set our new hostname. I have decided to name my new server `WEB4`, because later we will install the Web Services role onto it and host a website. Remember, after renaming your computer just like in the GUI version of Windows Server, a system restart is necessary to put that change into action. So following your `Rename-Computer` command, you can issue a `Restart-Computer` to reboot the box:

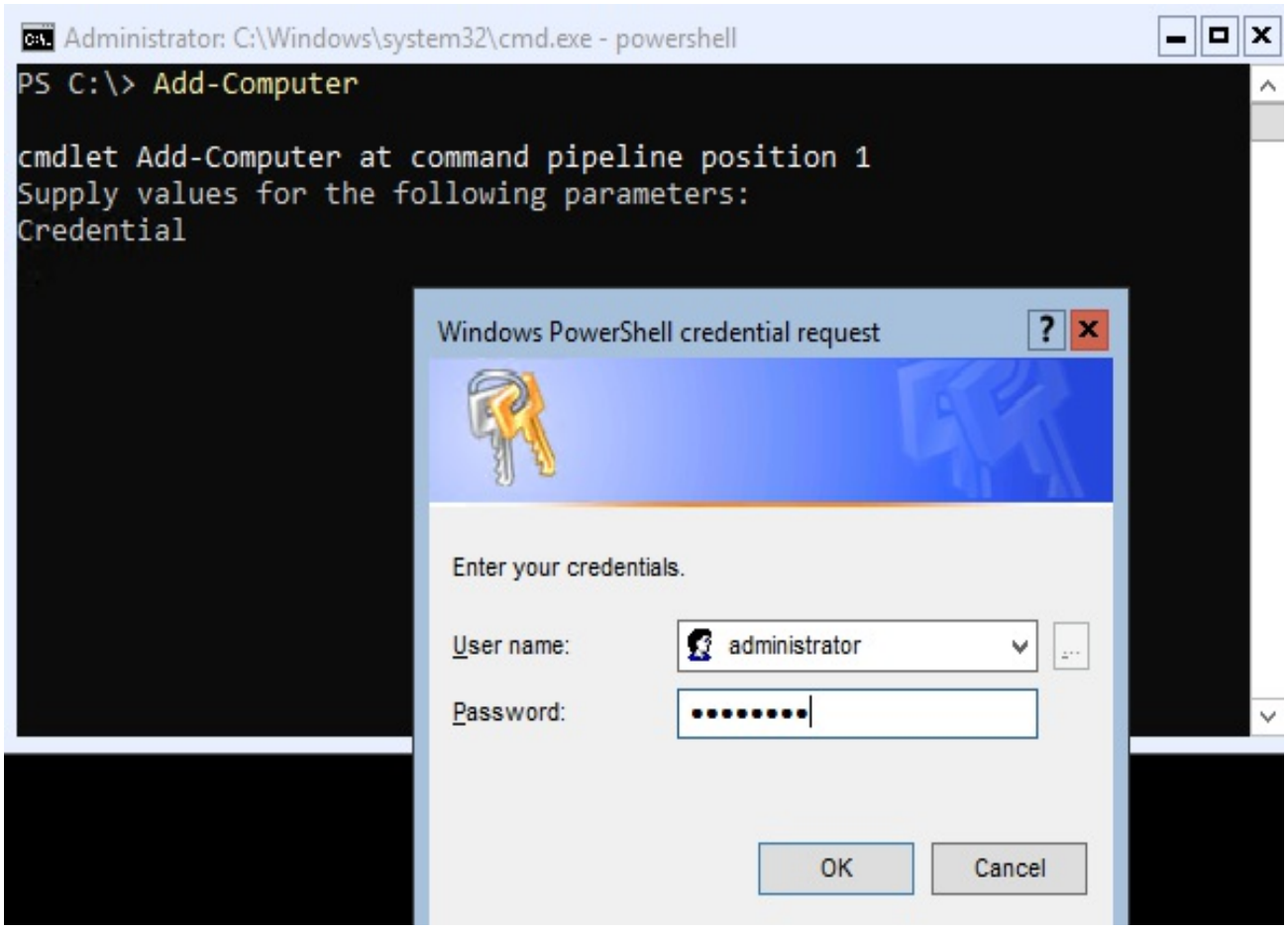
```
Rename-Computer WEB4
```

```
Restart-Computer
```

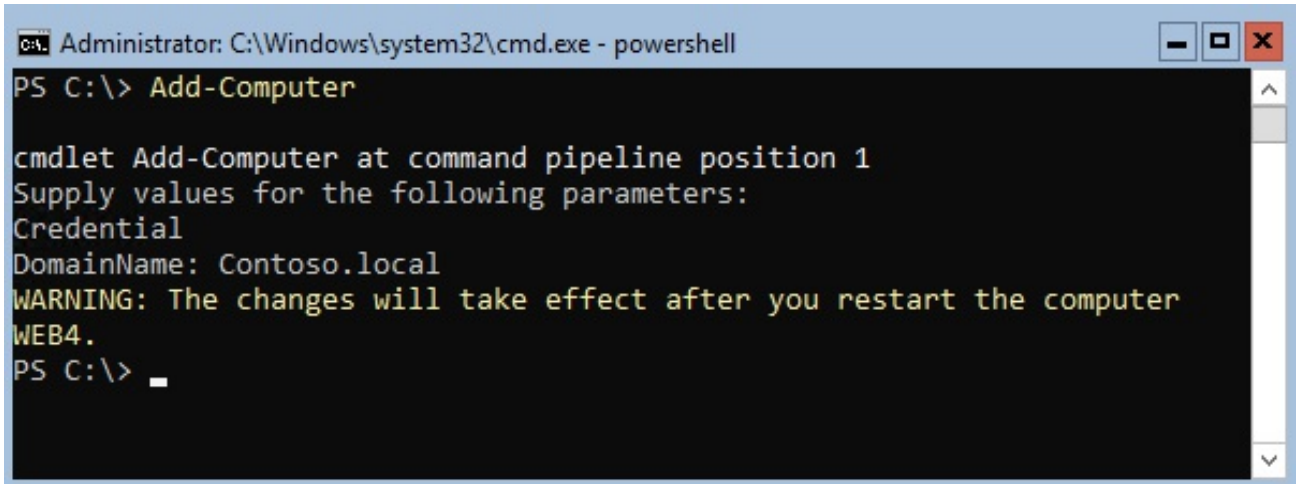


Joining your domain

The next logical step is, of course, joining your domain. These are the standard functions that we would perform on any new server in our environment, but done in a way that you may have never encountered before, since we are doing all of this strictly from the Command Prompt and PowerShell interfaces. To join a Server Core to your domain, head into PowerShell and then use the `Add-Computer` cmdlet. You will be asked to specify both the domain name and your credentials for joining the domain—the same information you would have to specify if you were joining a Windows Server 2019 in Desktop Experience mode to a domain. First, you must specify the credentials needed in order to do this domain join:



Then you tell it what domain you would like to join:



Alternatively, you could utilize the `-DomainName` parameter in combination with the original `Add-Computer` cmdlet in order to specify the name of the domain as part of the original command. And of course, after joining the domain, you need to `Restart-Computer` once again to finalize this change.

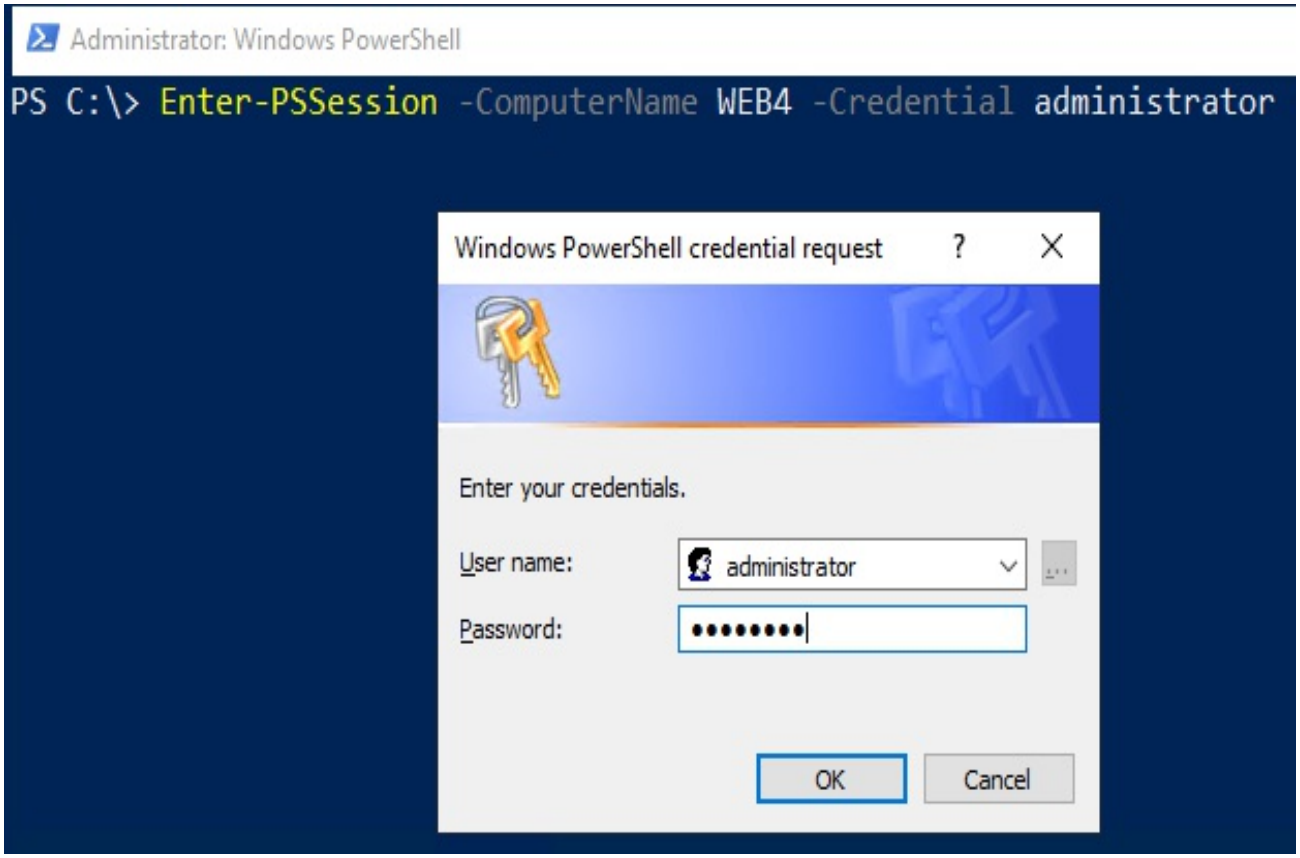
Remote PowerShell

Once the new server is IP-addressed, named, and domain-joined, we can start doing some real administration on this new Server Core instance. You could certainly continue to log in and interface directly with the console, but as with managing any other server in your environment, there must be ways to handle this remotely, right? One of the ways that you can manipulate Server Core without having to sit in front of it is by using a remote PowerShell connection.

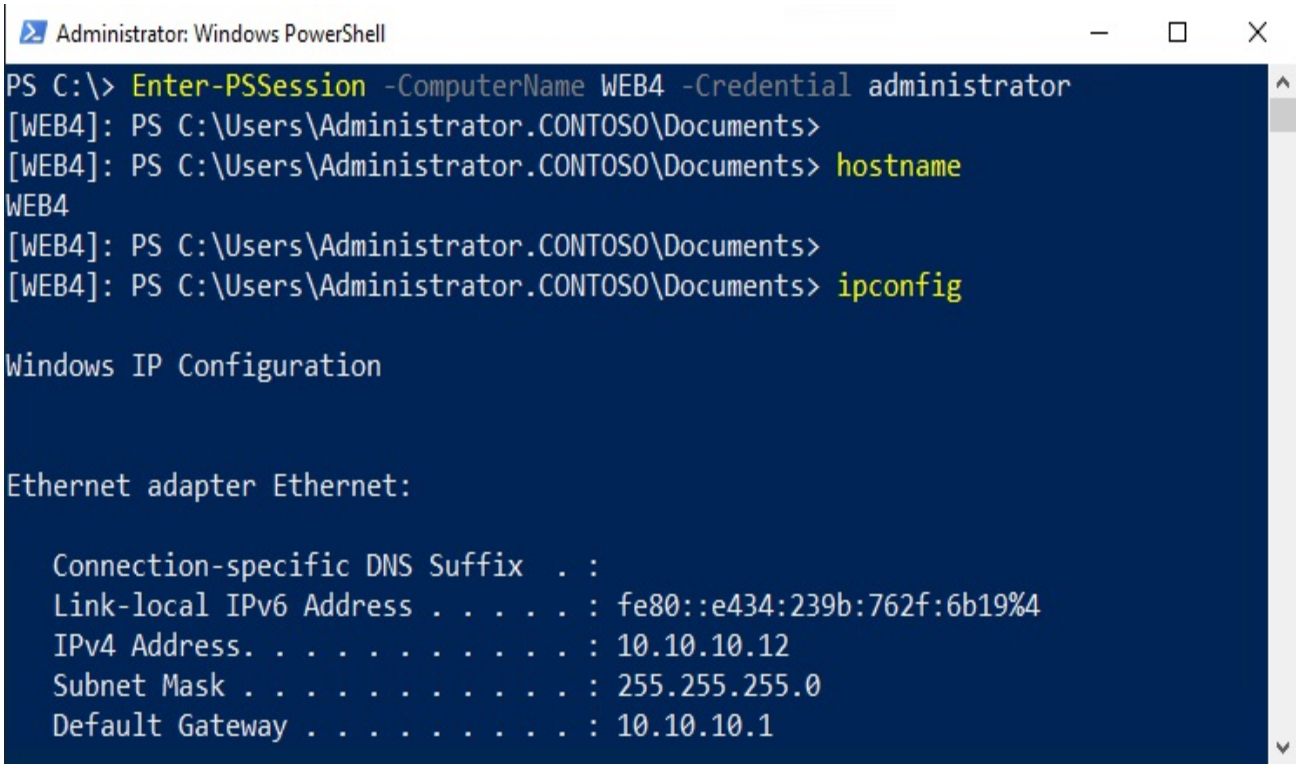
We will cover the process for using remote PowerShell to manipulate servers (both GUI and headless) in more detail in `Chapter 10`, *PowerShell*, but here is a glimpse of the commands necessary and the capabilities present when you are able to achieve a remote session from a PowerShell prompt on a workstation inside a domain-joined environment.

Open up PowerShell from another system—it can be a server or even a client operating system. This PowerShell window is obviously open within the context of whatever machine you are currently logged into, and any commands you issue via PowerShell will illicit a response from the local system. In order to tap PowerShell into the `WEB4` Server Core instance, I will issue the following command. After running this, I am prompted for a password corresponding to the administrator account, and will then be able to issue remote PowerShell commands against our Server Core:

```
Enter-PSSession -ComputerName WEB4 -Credential administrator
```



Now we are sitting at a PowerShell prompt, remotely connected to the `WEB4` Server Core box. You can see this by (`WEB4`) being listed to the left of our prompt. Perhaps you don't trust that little identifier, and want to make sure that this PowerShell window is now accessing and manipulating the remote `WEB4` server? Let's issue a couple of quick commands, such as `hostname` and `ipconfig`, to prove that the information being given to us in this PowerShell session is really coming from the new WEB4 server:

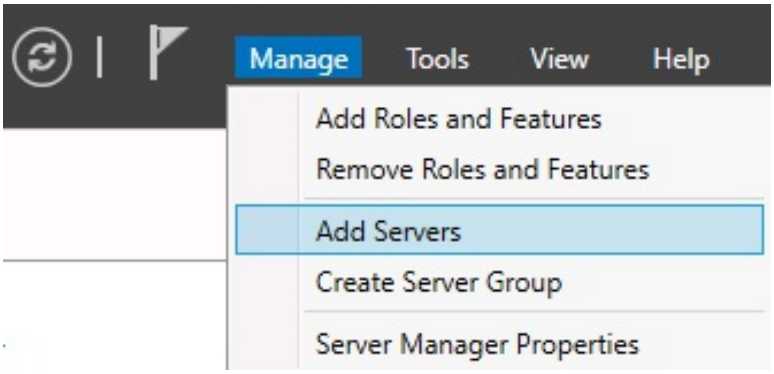


Now that we have a remote PowerShell connection to this new Server Core, we can do pretty much whatever we want to that server, right from this console.

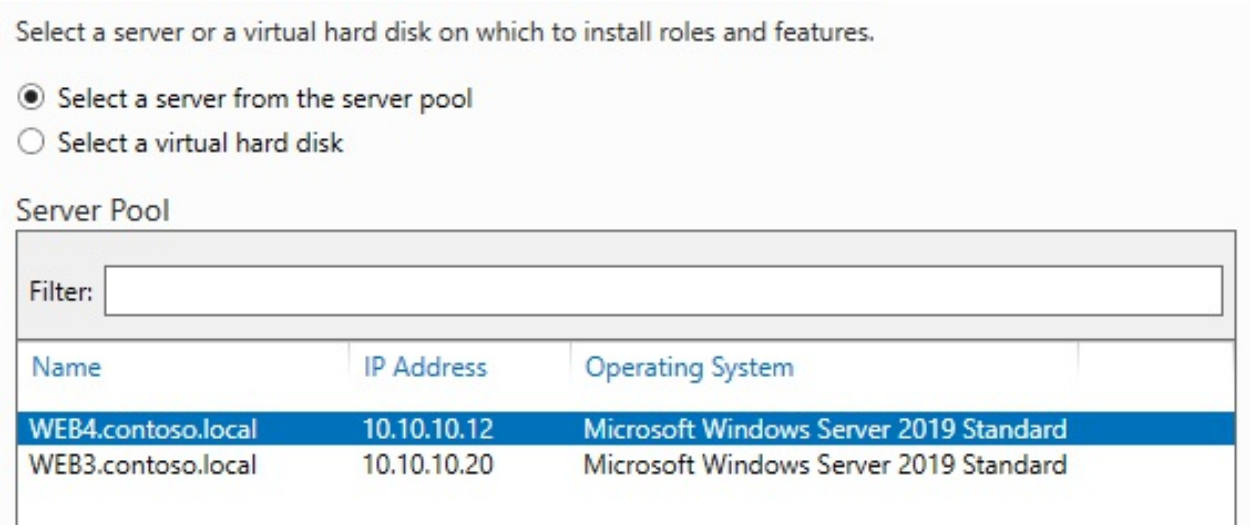
Server Manager

While the initial configuration of your server will be somewhat handled from the command-line interfaces available at the console, once your server has been established on the network, it will likely be more advantageous for you to expand your horizons a little. You could probably find PowerShell cmdlets that allow you do manage and manipulate anything in your new server, but that is still a pretty new mentality for most of us—we are generally more accustomed to using graphical tools such as Server Manager. You already know that Server Manager can be used to manage multiple servers, local and remote, and is a piece of the Microsoft *centralized management* puzzle. This remote management capability in Server Manager that we explored earlier in the book allows you to tap into not only GUI-based Windows Servers, but Server Core instances as well.

I want to install a role onto my new `WEB4` server. I could do that with PowerShell right on the server console, but instead let's try adding `WEB4` into Server Manager that is running on another one of my servers. I am going to log into `WEB3`, and use Server Manager from there. Just like we have already seen, I can add a new server into Server Manager using the Manage menu and choosing Add Servers:



Add the new WEB4 server into our list of managed machines, and it is now manageable from inside this instance of Server Manager. Getting back to what my original intentions were, I want to install the Web Server (IIS) role onto WEB4. If I use the Add roles and features function inside **Server Manager**, I can now choose to manipulate the WEB4 server:

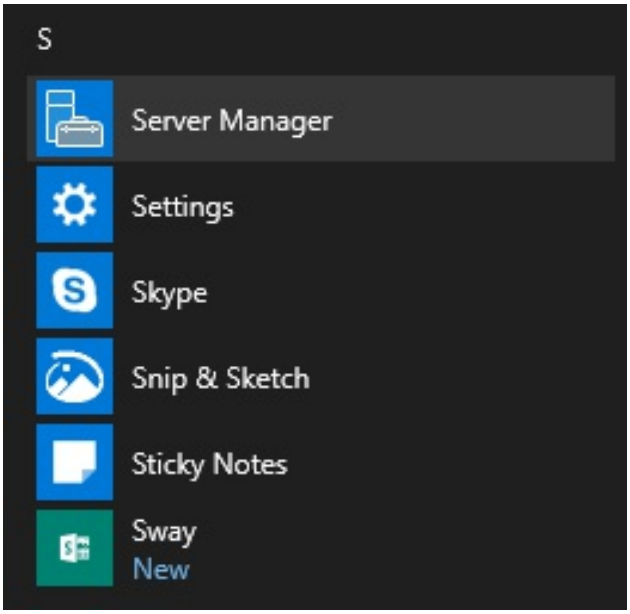


Just like with any server running the full Desktop Experience version of Windows Server, we can now finish walking through the role installation wizard, and the Web Server role will be installed on WEB4.

Remote Server Administration Tools

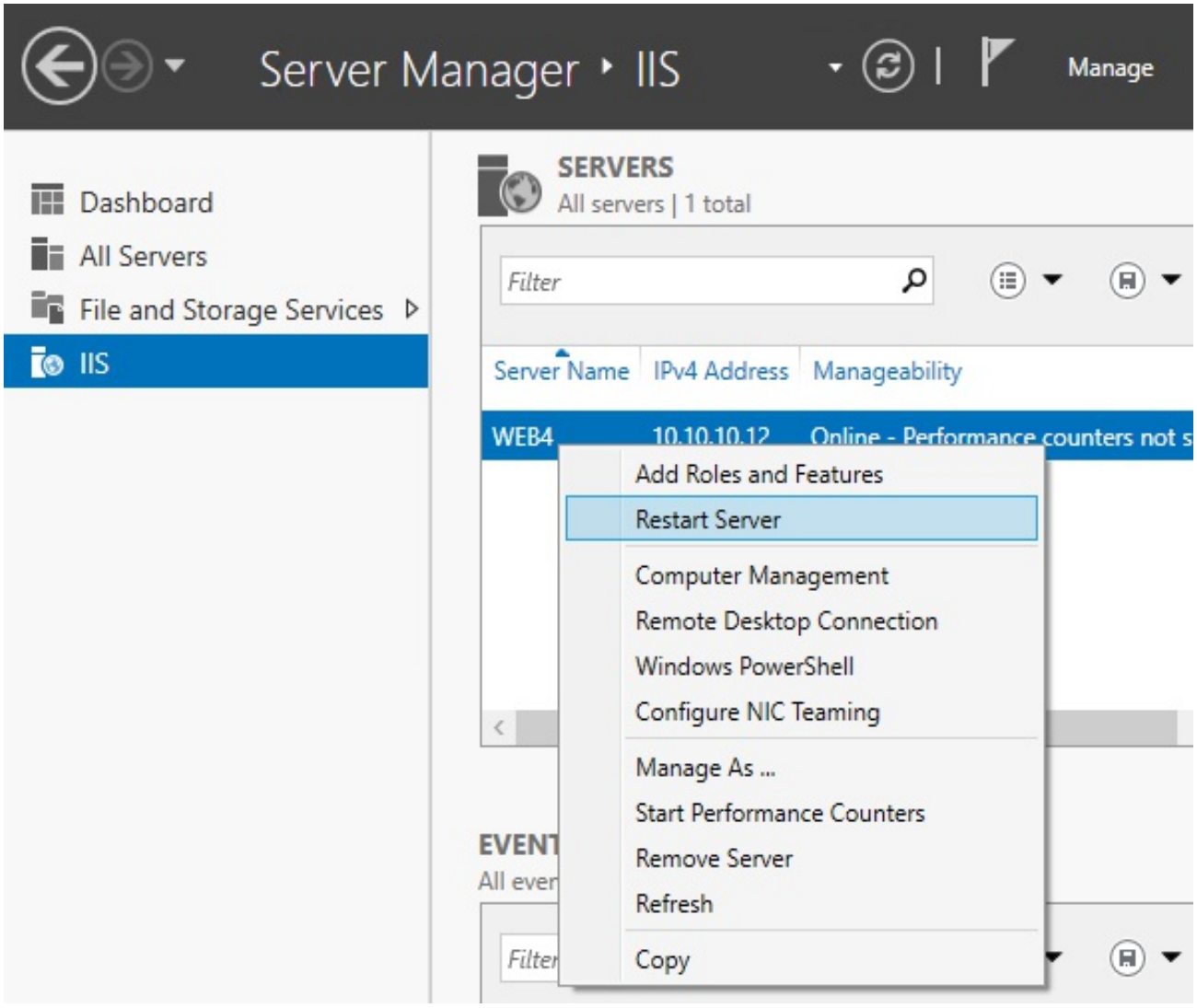
Also true is the fact that you can manage Server Core instances with the **Remote Server Administration Tools** (**RSAT**) in Windows 10. RSAT is essentially just a copy of Server Manager that is designed to run on the client operating system. In our case, I already have a Windows 10 machine on which I installed RSAT earlier in the book, so I will test by logging into that guy and adding `WEB4` into the interface. I just finished installing the IIS role on `WEB4` in our previous task, so I should be able to see that listed inside RSAT when I connect it to `WEB4`.

If you haven't used RSAT before and haven't read over that section of our text, it is important to know that there is no application called **Remote Server Administration Tools**. Instead, after the RSAT installation has completed, take a look inside your Start Menu for the application called Server Manager. This is how you utilize a Windows 10 client to remotely manage Windows Server 2019 instances:



Exactly like you would do from a Server Manager interface of Windows Server 2019, go ahead and walk through the wizard to add other servers to manage. Once I have added WEB4 as a Managed Server in my Win10's Server Manager, I can see IIS listed inside my Dashboard. This indicates that my IIS service running on WEB4 is visible, accessible, and configurable right from my Windows 10 desktop computer. For the majority of the tasks that I need to accomplish on WEB4, I will never have to worry about logging into the console of that server.

If I right-click on the WEB4 server name from within this RSAT console, you can see that I have many features available to me that I can use to manage this remote Server Core instance:



So you can see that there are ways to use the GUI tools in order to manage our GUI-less instances of Windows Server. It's just a matter of putting your mind into a place where you are thinking of servers as headless, and that tools such as PowerShell or Server Manager really don't care at all whether the server they are changing is local or remote. The processes and tools are the same either way. You can see in the previous screenshot that I could even click from here in order to launch a remote PowerShell connection to `WEB4`. Clicking on this button immediately launches a PowerShell prompt that is remotely tied to the `WEB4` server, even though I am currently only logged into my Windows 10

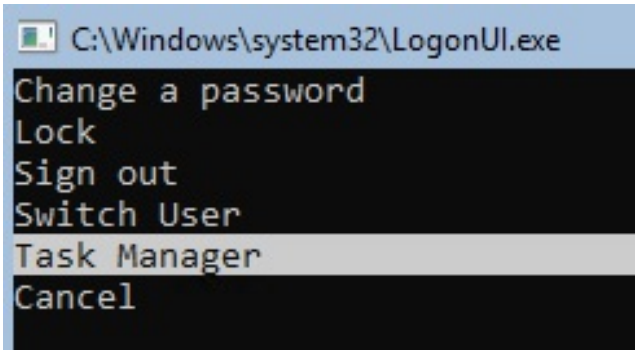
workstation. This is even easier than issuing the `Enter-PSSession` cmdlet from inside PowerShell.

Accidentally closing Command Prompt

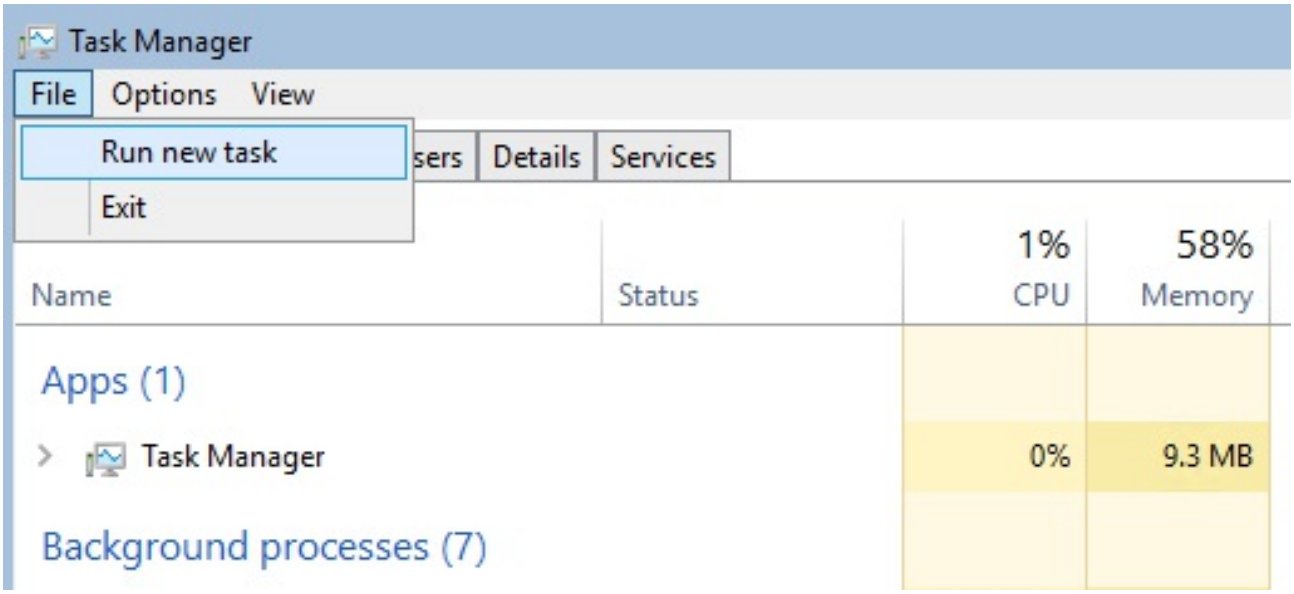
Let's take a look at one more thing directly from the Server Core console; this is a common hurdle to overcome if you haven't utilized Server Core much. It is our tendency to close windows and applications that are no longer being used, and so you might unconsciously close the Command Prompt window that is serving your entire administrative existence within a Server Core console session. Now you're sitting at a large blank screen, with seemingly no interface and nowhere to go from here.

How do you get back to work on this server? Do we have to turn the server off and back on in order to reset it? That would interrupt any roles or traffic that this server might be serving up to users, so obviously it isn't the ideal approach.

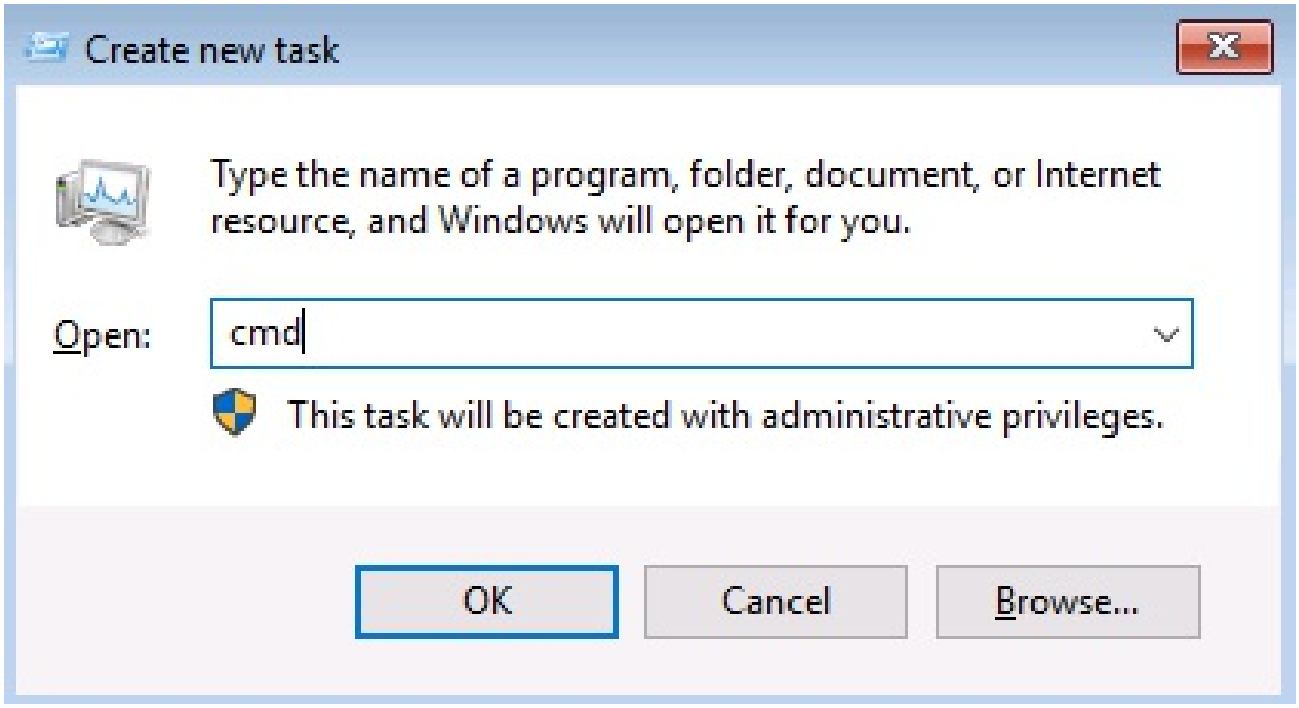
There is a simple way to get Command Prompt back, by using **Task Manager** to launch a new instance of Command Prompt. After mistakenly closing your current Command Prompt window, when sitting at the empty black screen of a Server Core console, you can press *Ctrl + Alt + Del* and you will be presented with the following options:



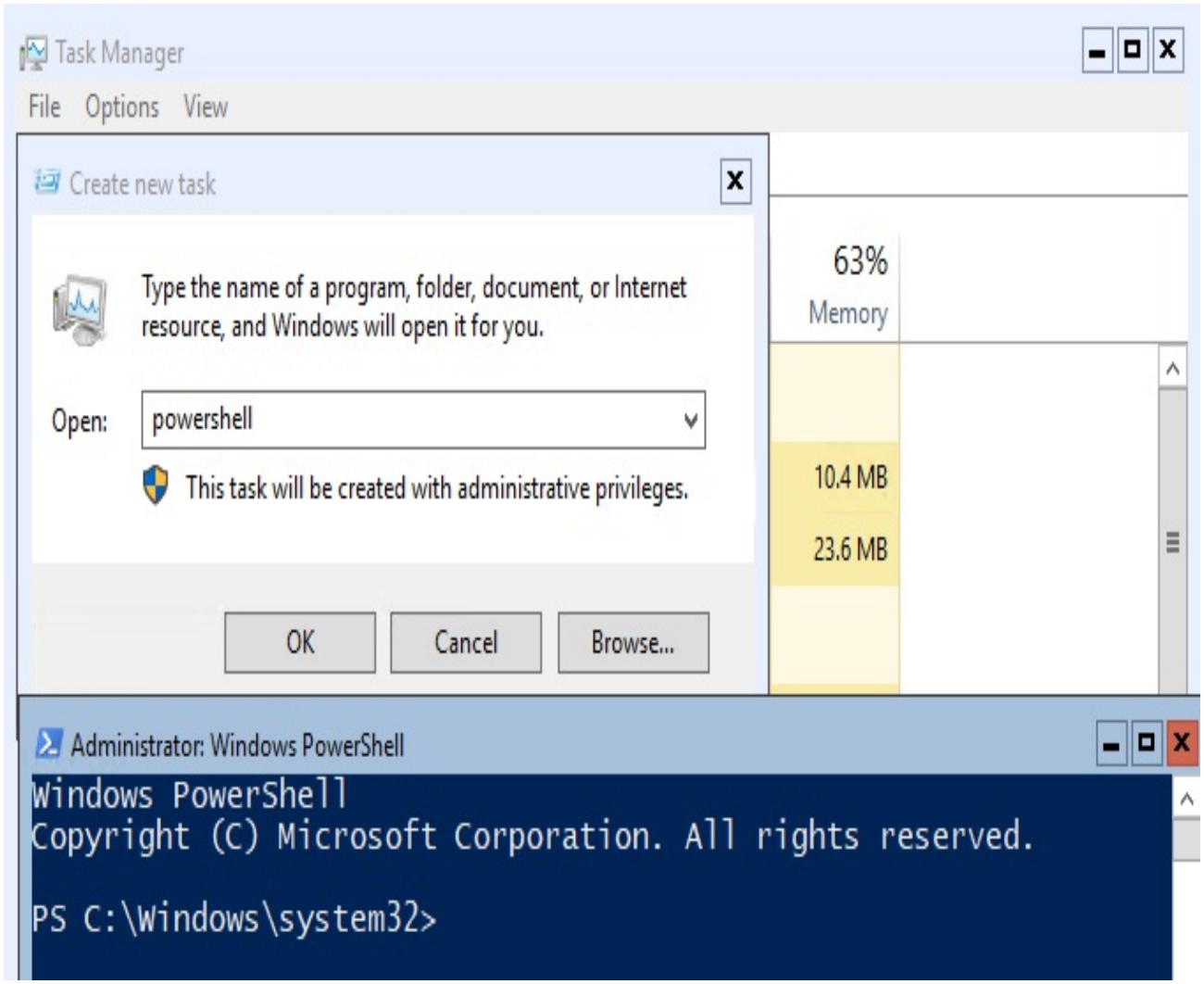
There are actually a few different functions you can perform here, which is pretty neat. But to get our Command Prompt window back, arrow-down to `Task Manager` and press *Enter*. This will launch the **Task Manager** application that we are all familiar with. Now click on More details in order to expand Task Manager's screens. Drop down the File menu, and click on Run new task:



In the Create new task box, type `cmd` and then click on OK:



Alternatively, you could specify to launch any application directly from this Create new task prompt. If you were interested in moving straight into PowerShell, instead of typing `cmd`, you could instead simply type `powershell` into that prompt, and it would open directly:


Task Manager
File Options View
Create new task
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
powershell
This task will be created with administrative privileges.
OK
Cancel
Browse...
63%
Memory
10.4 MB
23.6 MB
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\Windows\system32>

Windows Admin Center for managing Server Core

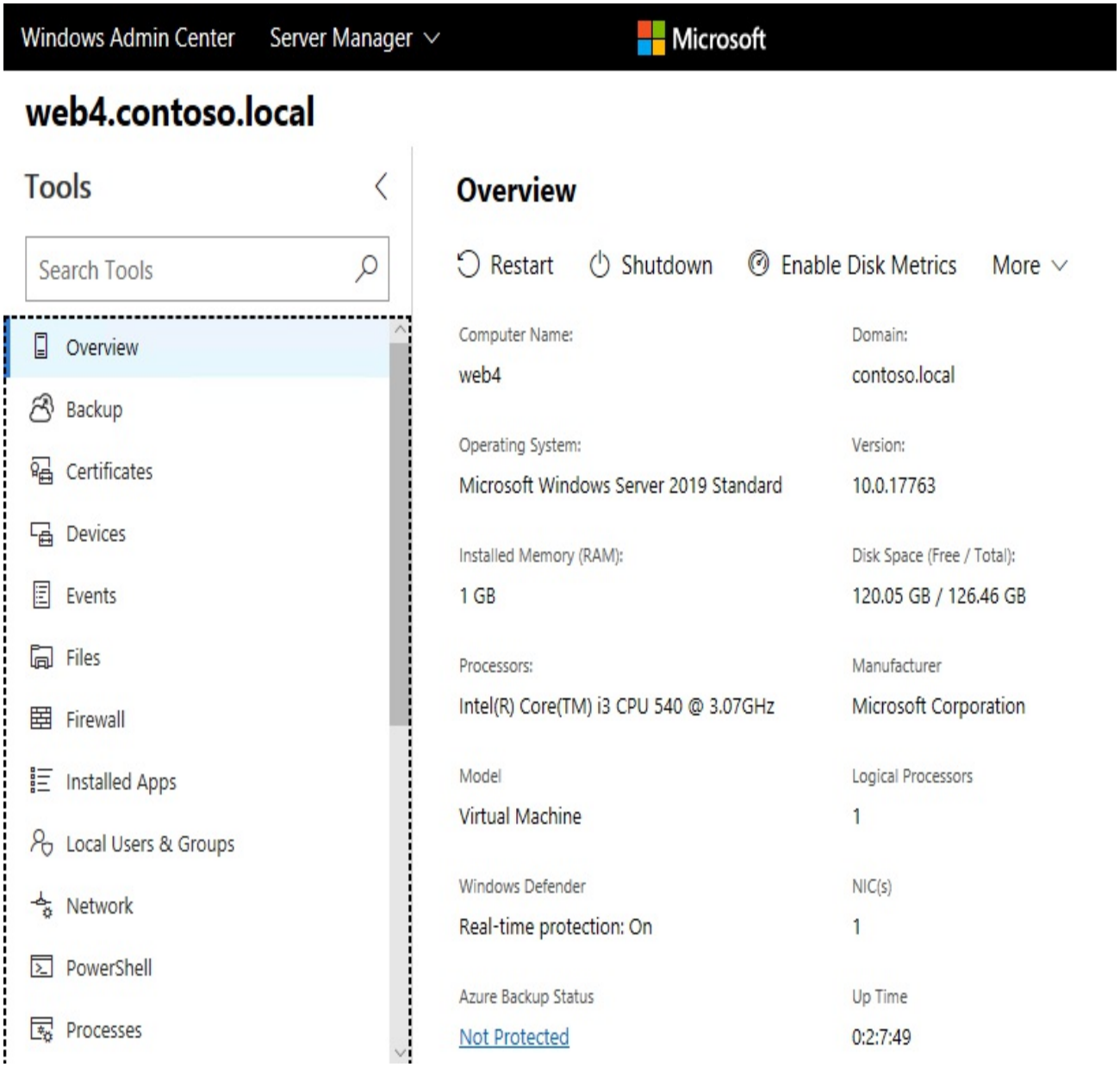
While Command Prompt from the console, remote PowerShell connections, remote Server Manager administration, and even the RSAT tools running on a Windows 10 workstation are all valid and powerful tools for administering our Server Core instances, they have all now been upstaged by the release of Windows Admin Center. You have already learned what Windows Admin Center can do for centrally managing your entire server infrastructure, but what we need to point out here is that WAC can be used for servers both with and without graphical interfaces.

I have spoken with many Windows Server administrators about the topic of Server Core, and one of the biggest blocks to implementing these more efficient and secure server platforms is an apprehension that, once configured, ongoing administration and maintenance of these servers will be more difficult to handle. Admins who are familiar and comfortable working within the Windows Server Desktop Experience know exactly what needs to be done in order to accomplish their daily tasks, but remove that point-and-click interface, and suddenly the workday gets a lot more complicated.

Thankfully, you don't have to memorize the PowerShell handbook in order to use Server Core! Windows Admin Center treats Server Core instances in the same way that it does a server running Desktop Experience. It just works!

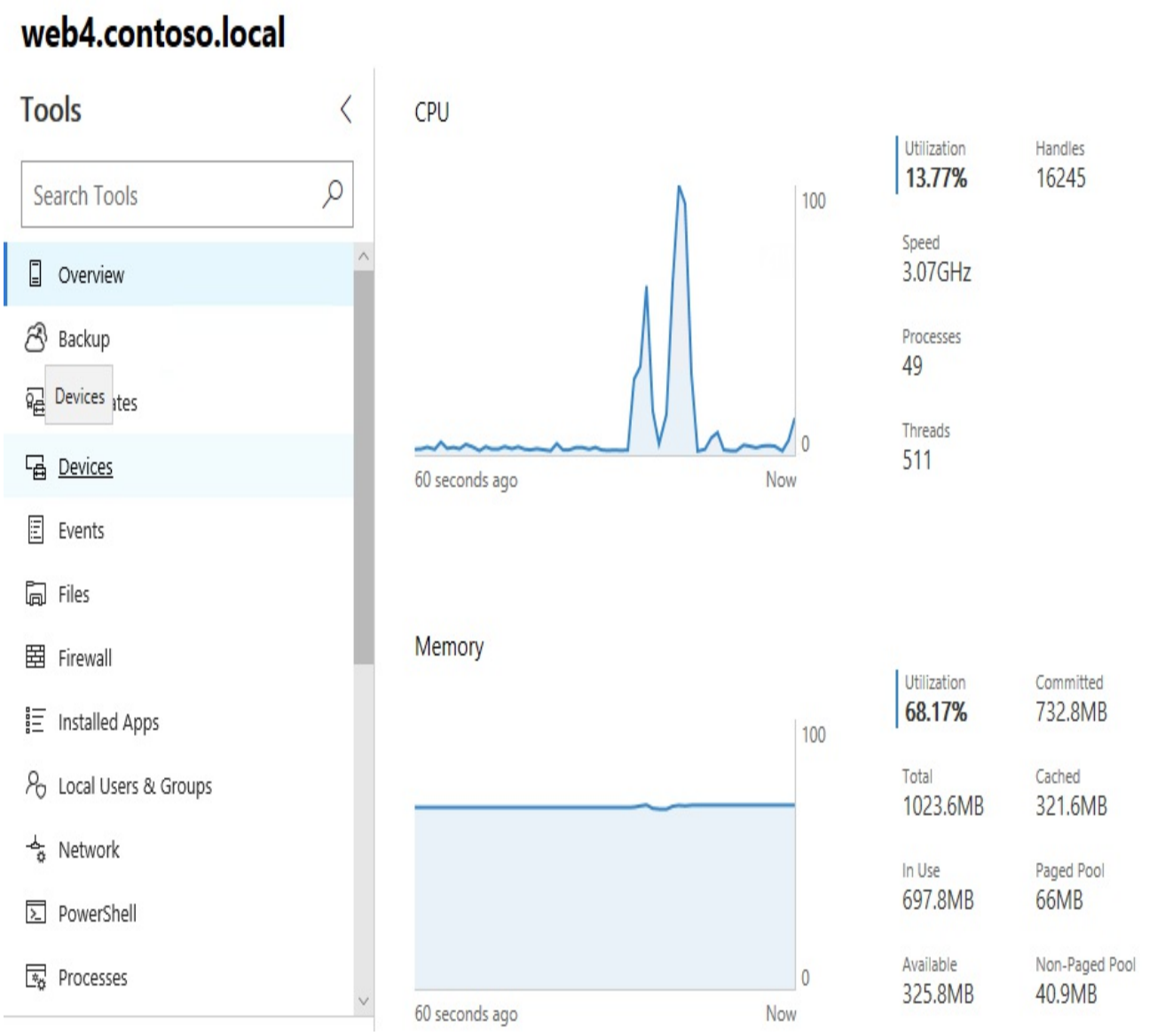
We already have WAC installed onto a server in our test lab, so let's open it up and add in my new WEB4 server to be administered, and take a look at what options are available for ongoing maintenance of this server.

When we first connect to WEB4 via the WAC console, there is actually nothing in here that even indicates this is a Server Core instance, we have all of the WAC tools and utilities available to click on:



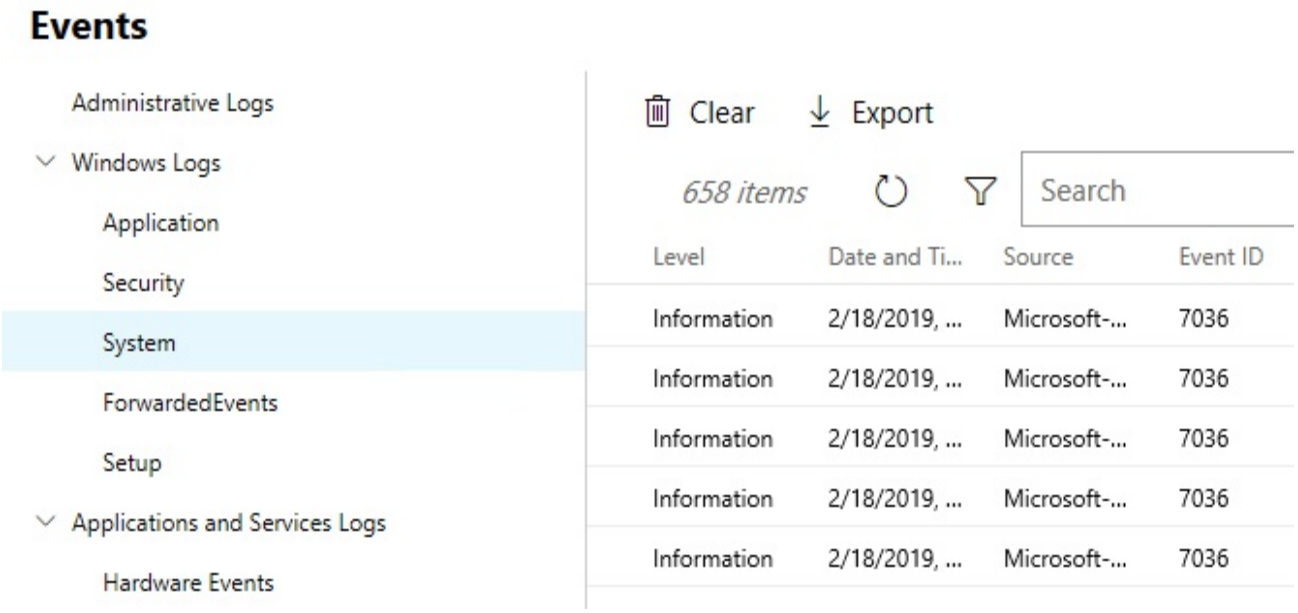
Let's try a couple of things from Windows Admin Center. You, of course, have power controls right near the top of the screen, from which you could easily shut down or restart the server. That is much easier and faster than having to establish a remote PowerShell connection in order to issue commands to accomplish the same actions. There are also performance metrics on the home screen (if you scroll down), showing your consumed CPU, Memory, and Networking resources. Without

WAC, you would need to log into WEB4 and launch **Task Manager** in order to see these statistics:



Moving away from the home screen, useful as it is, try clicking on one of the Tools listed along the left side of the screen, such as Events. Without WAC, if you wanted to troubleshoot an issue on a Server Core, it would make sense to look into the Windows Event Logs on that server, but how would you go about doing that from a command-line

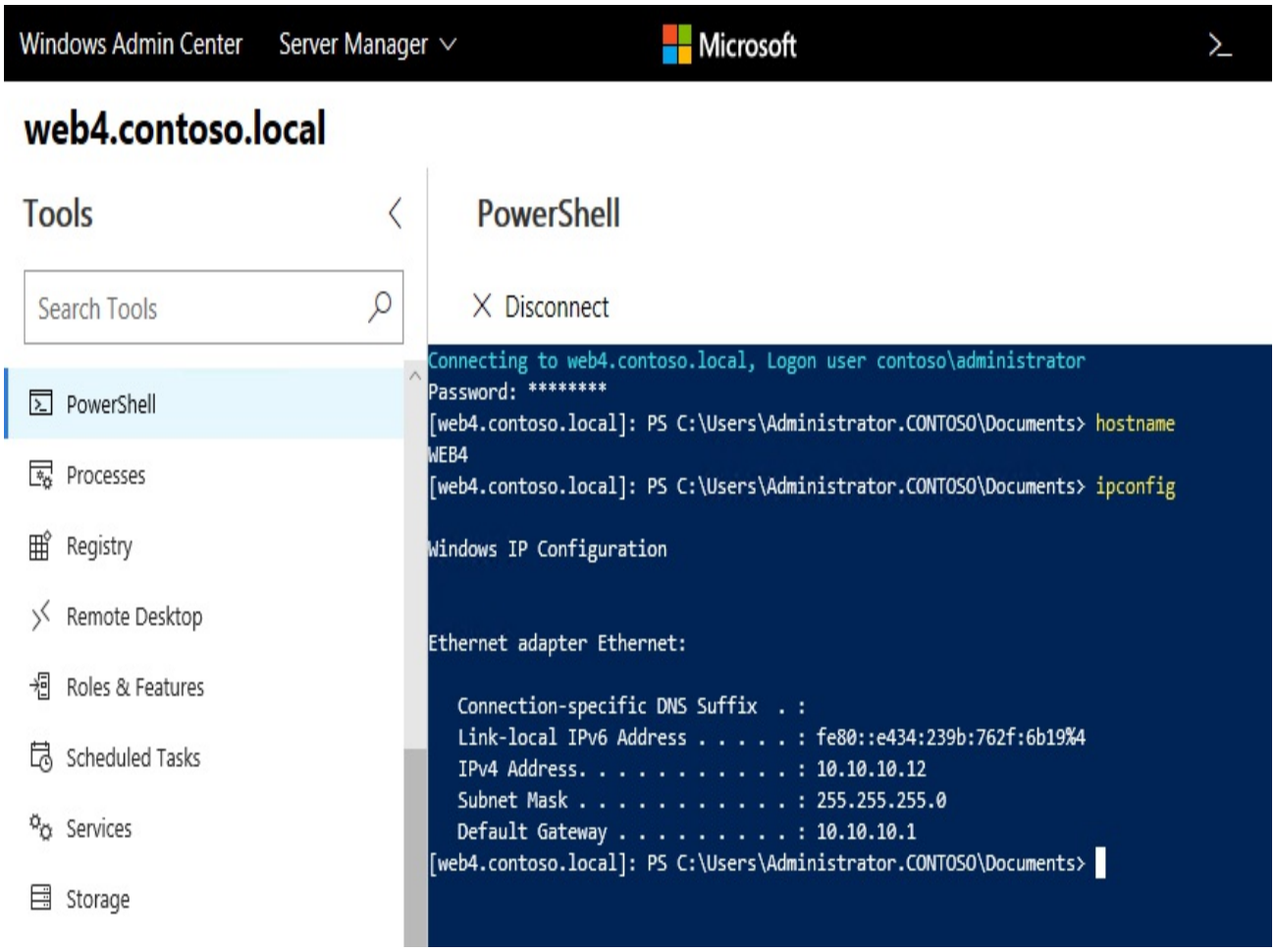
interface? I suppose you could have logged onto the Server Core console, and used Task Manager to launch EventVwr, but opening up WAC and simply clicking on Events is much easier:



Other examples of useful functions inside WAC, particularly when working with a Server Core instance, would be using Files to navigate the file and folder structure of WEB4's hard drive, or using the Firewall function here in order to create or remove Windows Firewall rules on WEB4. There is also a Network tool, from which you can manipulate IP addressing configurations.

While many more tools exist inside the Windows Admin Center, the last one I want to point out is that, once again, we have a PowerShell option (similar to what we can launch from inside Server Manager). This **PowerShell** button will invoke and display for us a remote PowerShell connection to the WEB4 Server Core instance, if ever we can't find a function that is needed inside WAC and need to dive a little further under the hood in order to accomplish something from a command interface. And the best part is that you never actually had to launch PowerShell! This is still all happening from within your internet

browser window:



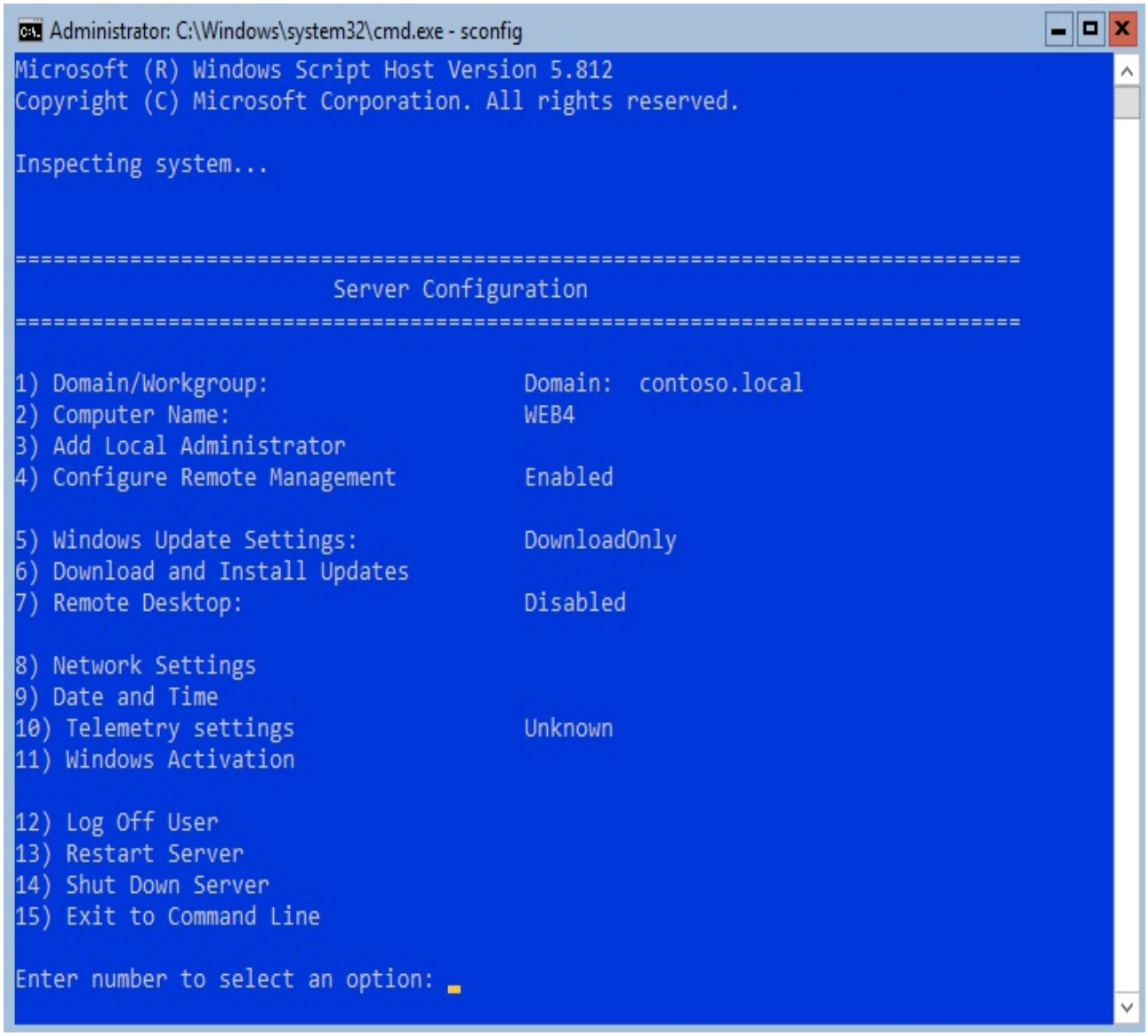
There is so much more that can be accomplished from inside the Windows Admin Center. Editing Registry, adding Roles & Features, checking the status of your Services, even interfacing with Windows Update. If you aren't already using WAC, you're missing the boat!

The Sconfig utility

Now we are going to take a step backward and check out a tool that is available inside Server Core, but one that is generally only useful when working on the console of your server. As you have seen, any time that you boot a Server Core, you land inside a Command Prompt window from which you can flip over into PowerShell and then use traditional Windows cmdlets in order to configure your new Server Core instance.

Alternatively, you may employ the **Sconfig** utility. This is a set of tools, kind of like command-line shortcuts, for implementing the basic items needed in order to bring your new server online and get it connected to the network. The purpose of Sconfig is to be step 1 after installing the operating system, taking care of the initial configurations on the new server so that you can then jump over to start using one of the more robust administrative interfaces, such as Server Manager or Windows Admin Center.

Immediately after spinning up a new Server Core instance, you find yourself at the Command Prompt, which is awaiting input. Inside this screen, simply type `Sconfig` and press *Enter*; you should experience a quick change from black to blue, and see the following screen:



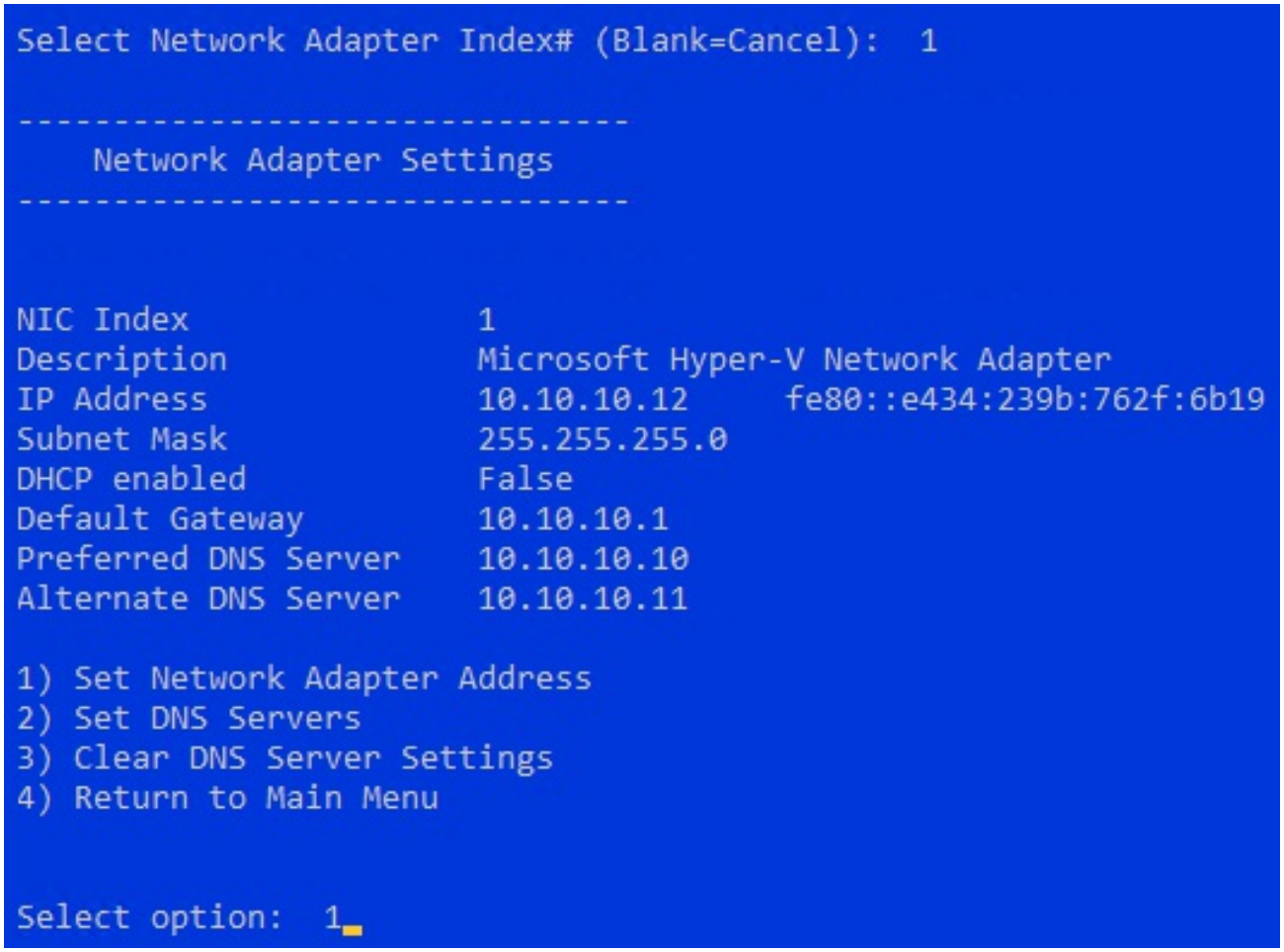
The options available inside Sconfig are fairly self-explanatory, but we'll cover the common tasks performed here. Again, these are all things that you could instead accomplish via PowerShell cmdlets, but I find it easier to take the Sconfig approach. The most common uses of this interface are to configure initial networking settings by pressing *8*, or to configure the server hostname and domain membership by using options *2* and *1*.

I will go ahead and press *2* on my keyboard, and then press *Enter*, and I

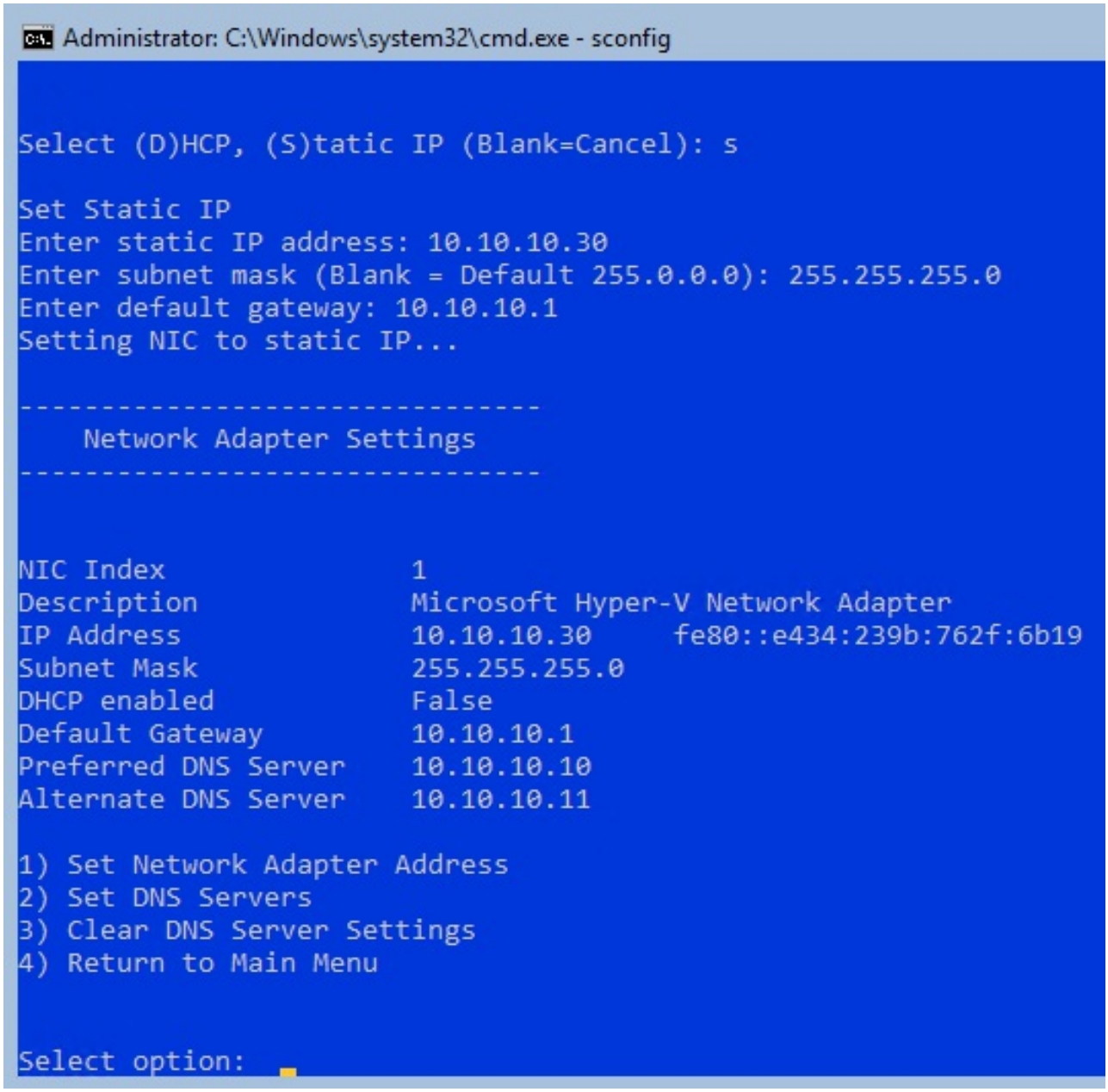
am immediately presented with a prompt asking me to specify a new computer name. This is an extremely fast way to configure the hostname of new Server Core servers. Since I don't actually want to rename WEB4, I will leave the selection blank and press *Enter* to return to the main screen.

Now I want to check out networking settings. Pressing *8* and then *Enter* brings me into `Network Settings`, where I can see that my current IP address on WEB4's NIC is `10.10.10.12`. This is correct, but let's change that address for the sake of walking through a real Sconfig setting change.

I first selected my network adapter index, which was number one. I am now shown additional information about what is already configured on this NIC, and have options to change this information. Selecting option one again will allow me to `Set Network Adapter Address`:



Input the letter `s`, which tells Sconfig that you want to enter a Static IP address, and then input the new IP address that you want configured on this NIC. I will change `WEB4` to be `10.10.10.30`, just to prove that this works. After inputting the IP address, I must also define a new subnet mask and gateway address:



WEB4's NIC has been immediately updated to a new IP address of `10.10.10.30`, as seen in the resulting output. While it may not be a common occurrence to visit the Sconfig tool after the initial configuration of a Server Core instance, this tool can be a real time-saver when used for the initial configuration of network and naming settings of any new Server Core.

Roles available in Server Core

Server Core is obviously a restricted form of the operating system, and some of the roles inside Windows Server are just not designed to work properly within that limited context. Fortunately for us, most of them are, which enables Server 2019 administrators to deploy most of their critical infrastructure via the more secure Server Core platform. Here is a list of the roles that are currently supported to run on a Windows Server 2019 Server Core instance:

- Active Directory Certificate Services
- Active Directory Domain Services
- Active Directory Federation Services
- Active Directory Lightweight Directory Services
- Active Directory Rights Management Services
- Device Health Attestation
- DHCP Server
- DNS Server
- File Services
- Host Guardian Service
- Hyper-V

- Print and Document Services
- Remote Access
- Volume Activation Services
- Web Server (IIS)
- Windows Server Update Services

What happened to Nano Server?

This story about small-footprint Windows Server platforms doesn't used to end with Server Core. Anyone who kept tabs on the new features coming out with Server 2016 is aware that there was another installation option for the Server 2016 operating system called Nano Server. The premise of Nano Server was an even smaller, more secure, more efficient, super-tiny operating system that could run a limited set of roles. Though limited, it was still capable of being installed onto a physical or virtual server platform, run as a true server operating system, and could still host traditional workloads on it.

Unfortunately for Nano Server enthusiasts, and especially for anyone who has already done the work of installing and using it, the story behind Nano Server has flipped around completely over the last couple of years. To make a long story short: you can no longer use Nano Server for anything that a traditional server can do. You cannot install it onto physical hardware; you cannot even install Nano onto a VM. Additionally, management functionalities, such as PowerShell and WinRM, have been removed from Nano Server, and you cannot install any of the Microsoft infrastructural roles onto it.

With all of this functionality having been ripped out of Nano Server's scope, what is left? Is it dead? Can Nano Server do *ANYTHING*?

The answer is containers. If you are interested in utilizing containers in order to build and host cloud-ready and scalable applications, this is where Nano is now focused. We will cover more information about containers and the fact that Nano Server is completely married to them

in Chapter 11, *Containers and Nano Server*, but suffice it to say that downloading container images from Microsoft will now be the *only* place that you will find Nano Server.

Summary

I have to be honest with you—writing this chapter has been exactly the kick in the pants that I needed to start thinking about shrinking my own servers. I am in the same boat as many of you: I know what Server Core is and have played around with it, but have never taken the steps to really use it in the production environment that I support. Now that tools such as Sconfig and the new Windows Admin Center are available to us, I have officially run out of excuses as to why I shouldn't be deploying new roles onto Server Core boxes. While it never hurts to learn something new, using Server Core no longer comes with the requirement that you must be fluent in PowerShell. The early days of Server Core came with a requirement to be really good with PowerShell, as this was the only reliable way to configure and interface with your tiny servers, but these new tools allow us to utilize the smaller platform and administer it without memorizing a bunch of new cmdlets.

Security is the primary reason that we should all be considering Server Core as our new standard. The Windows graphical interface adds a lot of code, and grants a lot of ability to those logged into the servers, such as the ability to browse the internet. This opens up all kinds of doors to vulnerabilities that simply don't exist in Server Core. The next chapter deals with the redundancies in Windows Server 2019.

Questions

1. True or False—Server Core is the default installation option for Windows Server 2019.
2. True or False—You can utilize PowerShell to change a Server 2019 from *Server Core* mode to *Desktop Experience* mode.
3. When sitting at the console of a freshly booted Windows Server 2019 Server Core instance, what application do you see on the screen?
4. What cmdlet can be used to view the current networking configuration on a Server Core?
5. Which PowerShell cmdlet can be used to configure the hostname of a Server Core?
6. Name some of the management tools that can be used to remotely interface with a Server Core.
7. What is the name of the utility built into Server Core that can be launched to provide quick task links for configuring IP addresses, hostname, and domain membership?

Redundancy in Windows Server 2019

Multiply that by two. This is a phrase I hear all the time when planning rollouts for work. I'm sure you have as well. Any time you are rolling out a new technology, you want to plan that rollout very carefully. Figure out what servers you need, where they need to be placed, and how the networking needs to be configured for those guys. Once the planning is done, order two of everything, in case one breaks. We live in a world of always-on technology. Services going down is unacceptable, particularly if we are hosting cloud or private cloud services. Really, any application or service that our users depend on to get their work done is mission-critical, and needs 100% uptime, or darn close to it. The problem with redundancy is that it's much easier to *talk the talk* than to *walk the walk*. Maybe one day we will be blessed with a magic *Press here to make this server redundant* button – but today is not that day. We need to understand the technologies that are available to us that enable us to provide redundancy on our systems. This chapter will introduce us to some of those technologies. This book is focused on Server 2019 used on-premise, so the technologies we will discuss today are the ones that you can utilize in your own local data centers, on the real (physical or virtual) servers that you are responsible for building, configuring, and maintaining yourself. Yes, the cloud can provide us with some magical scalability and redundancy options, but those are easy, and often we don't even need to understand how they work. When we are using our own servers within our own walls, *how can we add some increased reliability to our systems?*

We will cover the following topics cover in this chapter:

- **Network Load Balancing** (**NLB**)
- Configuring a load-balanced website
- Failover clustering
- Clustering tiers
- Setting up a failover cluster
- Recent clustering improvements in Windows Server
- **Storage Spaces Direct** (**S2D**)

Network Load Balancing (NLB)

Often, when I hear people discussing redundancy on their servers, the conversation includes many instances of the word *cluster,* such as, "*If we set up a cluster to provide redundancy for those servers..." or "Our main website is running on a cluster...*" While it is great that there is some form of resiliency being used on the systems to which these conversations pertain, it is often the case that clustering is not actually involved anywhere. When we boil down the particulars of how their systems are configured, we discover that it is NLB doing this work for them. We will discuss real clustering further along in this chapter, but first I wanted to start with the more common approach to making many services redundant. NLB distributes traffic at the TCP/IP level, meaning that the server operating systems themselves are not completely aware of or relying on each other, with redundancy instead being provided at the network layer. This can be particularly confusing—NLB versus clustering—because sometimes Microsoft refers to something as a cluster, when in fact it is using NLB to make those connections happen. A prime example is **DirectAccess**. When you have two or more DA servers together in an array, there are TechNet documents and even places inside the console where it is referred to as a cluster. But there is no failover clustering going on here; the technology under the hood that is making connections flow to both nodes is actually Windows NLB.

You've probably heard some of the names in the hardware load balancer market—**F5**, **Cisco**, **Kemp**, **Barracuda**. These companies provide dedicated hardware boxes that can take traffic headed toward a particular name or destination, and split that traffic between two or more application servers. While this is generally the most robust way

that you can establish NLB, it is also the most expensive and makes the overall environment more complex. One feature these guys offer that the built-in Windows NLB cannot provide is SSL termination, or SSL offloading, as we often call it. These specialized appliances are capable of receiving website traffic from user computers, that is SSL, and decrypting the packets before sending them on their way to the appropriate web server. This way, the web server itself is doing less work, since it doesn't have to spend CPU cycles encrypting and decrypting packets. However, today we are not going to talk about hardware load balancers at all, but rather the NLB capabilities that are provided right inside Windows Server 2019.

Not the same as round-robin DNS

I have discovered, over the years, that some people's idea of NLB is really round-robin DNS. Let me give an example of that: say you have an intranet website that all of your users access daily. It makes sense that you would want to provide some redundancy to this system, and so you set up two web servers, in case one goes down. However, in the case that one does go down, you don't want to require manual cutover steps to fail over to the extra server, you want it to happen automatically. In DNS, it is possible to create two host A records that have the same name, but point to different IP addresses. If **Server01** is running on `10.10.10.5` and **Server02** is running on `10.10.10.6`, you could create two DNS records both called INTRANET, pointing one host record at `10.10.10.5`, and the other host record at `10.10.10.6`. This would provide round-robin DNS, but not any real load balancing. Essentially what happens here is that when the client computers reach out to INTRANET, DNS will hand them one or the other IP address to connect. DNS doesn't care whether that website is actually running, it simply responds with an IP address. So even though you might set this up and it appears to be working flawlessly because you can see that clients are connecting to both Server01 and Server02, be forewarned. In the event of a server failure, you will have many clients who still work, and many clients who are suddenly getting Page cannot be displayed when DNS decides to send them to the IP address of the server that is now offline.

NLB is much more intelligent than this. When a node in an NLB array goes down, traffic moving to the shared IP address will only be directed to the node that is still online. We'll get to see this for ourselves shortly, when we set up NLB on an intranet website of our own.

What roles can use NLB?

NLB is primarily designed for *stateless* applications, in other words, applications that do not require a long-term memory state or connection status. In a stateless application, each request made from the application could be picked up by Server01 for a while, then swing over to Server02 without interrupting the application. Some applications handle this very well (such as websites), and some do not.

Web services (IIS) definitely benefits the most from the redundancy provided by NLB. NLB is pretty easy to configure, and provides full redundancy for websites that you have running on your Windows Servers, without incurring any additional cost. NLB can additionally be used to enhance FTP, firewall, and proxy servers.

Another role that commonly interacts with NLB is the remote access role. Specifically, DirectAccess can use the built-in Windows NLB to provide your remote access environment with redundant entry-point servers. When setting up DirectAccess to make use of load balancing, it is not immediately obvious that you are using the NLB feature built into the operating system because you configure the load-balancing settings from inside the Remote Access Management console, rather than the NLB console. When you walk through the Remote Access Management wizards in order to establish load balancing, that Remote Access console is actually reaching out into the NLB mechanism within the operating system and configuring it, so that its algorithms and transport mechanisms are the pieces being used by DirectAccess in order to split traffic between multiple servers.

One of the best parts about using NLB is that you can make changes to the environment without affecting the existing nodes. *Want to add a new server into an existing NLB array?* No problem. Slide it in without any downtime. *Need to remove a server for maintenance?* No issues here either. NLB can be stopped on a particular node, allowing another node in the array to pick up the slack. In fact, NLB is actually NIC-particular, so you can run different NLB modes on different NICs within the same server. You can tell NLB to stop on a particular NIC, removing that server from the array for the time being. Even better, if you have a little bit of time before you need to take the server offline, you can issue a `drainstop` command instead of an immediate stop. This allows the existing network sessions that are currently live on that server to finish cleanly. No new sessions will flow to the NIC that you have drain-stopped, and old sessions will evaporate naturally over time. Once all sessions have been dropped from that server, you can then yank it and bring it down for maintenance.

Virtual and dedicated IP addresses

The way that NLB uses IP addresses is an important concept to understand. First of all, any NIC on a server that is going to be part of a load-balanced array must have a static IP address assigned to it. NLB does not work with DHCP addressing. In the NLB world, a static IP address on an NIC is referred to as a **Dedicated IP Address** (**DIP**). These DIPs are unique per NIC, obviously meaning that each server has its own DIP. For example, in my environment, `WEB1` is running a DIP address of `10.10.10.40`, and my `WEB2` server is running a DIP of `10.10.10.41`.

Each server is hosting the same website on their own respective DIP addresses. It's important to understand that when establishing NLB between these two servers, I need to retain the individual DIPs on the boxes, but I will also be creating a new IP address that will be shared between the two servers. This shared IP is called the **Virtual IP Address** (**VIP**). When we walk through the NLB setup shortly, I will be using the IP address of `10.10.10.42` as my VIP, which is so far unused in my network. Here is a quick layout of the IP addresses that are going to be used when setting up my network load-balanced website:

```
WEB1 DIP = 10.10.10.40
WEB2 DIP = 10.10.10.41
Shared VIP = 10.10.10.42
```

When establishing my DNS record for `intranet.contoso.local`, which is the name of my website. I will be creating just a single host A record, and it

will point at my `10.10.10.42` VIP.

NLB modes

Shortly, we will find ourselves in the actual configuration of our load balancing, and will have a few decisions to make inside that interface. One of the big decisions is what NLB mode we want to use. Unicast is chosen by default, and is the way that I see most companies set up their NLB, perhaps because it is the default option and they've never thought about changing it. Let's take a minute to discuss each of the available options, to make sure you can choose the one that is most appropriate for your networking needs.

Unicast

Here, we start to get into the heart of how NLB distributes packets among the different hosts. Since we don't have a physical load balancer that is receiving the traffic first and then deciding where to send it, *how do the load-balanced servers decide who gets to take which packet streams?*

To answer that question, we need to back up a little bit and discuss how traffic flows inside your network. When you open up a web browser on your computer and visit `HTTP://WEB1`, DNS resolves that IP address to `10.10.10.40`, for example. When the traffic hits your switches and needs to be directed somewhere, the switches need to decide where the `10.10.10.40` traffic needs to go. You might be familiar with the idea of MAC addresses.

Each NIC has a MAC address, and when you assign an IP address to a NIC, it registers its own MAC address and IP with the networking equipment. These MAC addresses are stored inside an ARP table, which is a table that resides inside most switches, routers, and firewalls. When my WEB1 server was assigned the `10.10.10.40` IP address, it registered its MAC address corresponding to `10.10.10.40`. When traffic needs to flow to WEB1, the switches realize that traffic destined for `10.10.10.40` needs to go to that specific NIC's MAC address, and shoots it off accordingly.

So in the NLB world, when you are sending traffic to a single IP address that is split between multiple NICs, *how does that get processed at the MAC level?* The answer with unicast NLB is that the physical NIC's MAC address gets replaced with a virtual MAC address, and this MAC is

assigned to all of the NICs within the NLB array. This causes packets flowing to that MAC address to be delivered to all of the NICs, therefore all of the servers, in that array. If you think that sounds like a lot of unnecessary network traffic is moving around the switches, you would be correct. Unicast NLB means that when packets are destined for the virtual MAC address of an array, that traffic is basically bounced through all ports on the switch before finding and landing on their destinations.

The best part about unicast is that it works without having to make any special configurations on the switches or networking equipment in most cases. You set up the NLB configuration from inside the Windows Server tools, and it handles the rest. A downside to unicast is that, because the same MAC address exists on all the nodes, it causes some intra-node communication problems. In other words, the servers that are enabled for NLB will have trouble communicating with each other's IP addresses. Often, this doesn't really matter, because WEB1 would rarely have reason to communicate directly with WEB2. But if you really need those web servers to be able to talk with each other consistently and reliably, the easiest solution is to install a separate NIC on each of those servers, and use that NIC for those intra-array communications, while leaving the primary NICs configured for NLB traffic.

The other downside to unicast is that it can create some switch flooding. The switches are unable to learn a permanent route for the virtual MAC address, because we need it to be delivered to all of the nodes in our array. Since every packet moving to the virtual MAC is being sent down all avenues of a switch so that it can hit all of the NICs where it needs to be delivered, it has the potential to overwhelm the switches with this flood of network packets. If you are concerned about that or are getting complaints from your networking people about switch flooding, you might want to check out one of the multicast modes for your NLB cluster.

An alternative method for controlling unicast switch flooding is to get creative with VLANs on your switches. If you plan an NLB server array and want to ensure that the switch traffic being generated by this array will not affect other systems in your network, you could certainly create a small VLAN on your switches and plug only your NLB-enabled NICs into that VLAN. This way, when the planned flood happens, it only hits that small number of ports inside your VLAN, rather than segmenting its way across the entire switch.

Multicast

Choosing multicast as your NLB mode comes with some upsides, and some headaches. The positive is that it adds an extra MAC address to each NIC. Every NLB member then has two MAC addresses: the original and the one created by the NLB mechanism. This gives the switches and networking equipment an easier job of learning the routes and sending traffic to its correct destinations, without an overwhelming packet flood. In order to do this, you need to tell the switches which MAC addresses need to receive this NLB traffic, otherwise, you will cause switch flooding, just like with unicast. Telling the switches which MACs need to be contacted is done by logging into your switches and creating some static ARP entries to accommodate this. For any company with a dedicated networking professional, usually proficient in Cisco equipment, this will be no sweat. If you are not familiar with modifying ARP tables and adding static routes, it can be a bit of a nuisance to get it right. In the end, multicast is generally better than unicast, but it can be more of an administrative headache. My personal preference still tends to be unicast, especially in smaller businesses. I have seen it used in many different networks without any issues, and going with unicast means we can leave the switch programming alone.

Multicast IGMP

Better yet, but not always an option, is multicast with **Internet Group Management Protocol** (**IGMP**). Multicast IGMP really helps to mitigate switch flooding, but it only works if your switches support IGMP snooping. This means that the switch has the capability to look inside multicast packets in order to determine where exactly they should go. So where unicast creates some amount of switch flooding by design, multicast can help to lower that amount, and IGMP can get rid of it completely.

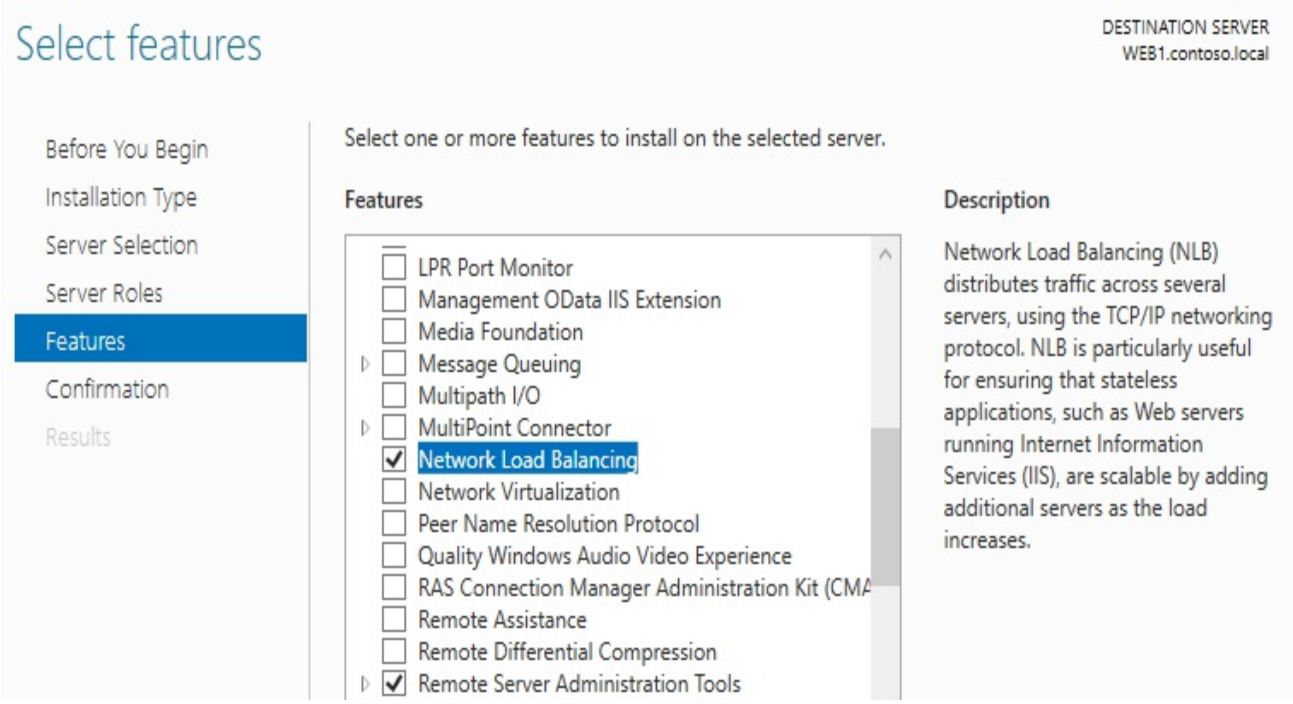
The NLB mode that you choose will depend quite a bit upon the capabilities of your networking equipment. If your servers have only a single NIC, try to use multicast or you will have intra-array problems. On the other hand, if your switches and routers don't support multicast, you don't have a choice—unicast will be your only option for configuring Windows NLB.

Configuring a load-balanced website

Enough talk; it's time to set this up for ourselves and give it a try. I have two web servers running on my lab network, WEB1 and WEB2. They both use IIS to host an intranet website. My goal is to provide my users with a single DNS record for them to communicate with, but have all of that traffic be split between the two servers with some real load balancing. Follow along for the steps on making this possible.

Enabling NLB

First things first, we need to make sure that WEB1 and WEB2 are prepared to do NLB, because it is not installed by default. NLB is a feature available in Windows Server 2019, and you add it just like any other role or feature, by running through the Add roles and features wizard. Add this feature on all of the servers that you want to be part of the NLB array:



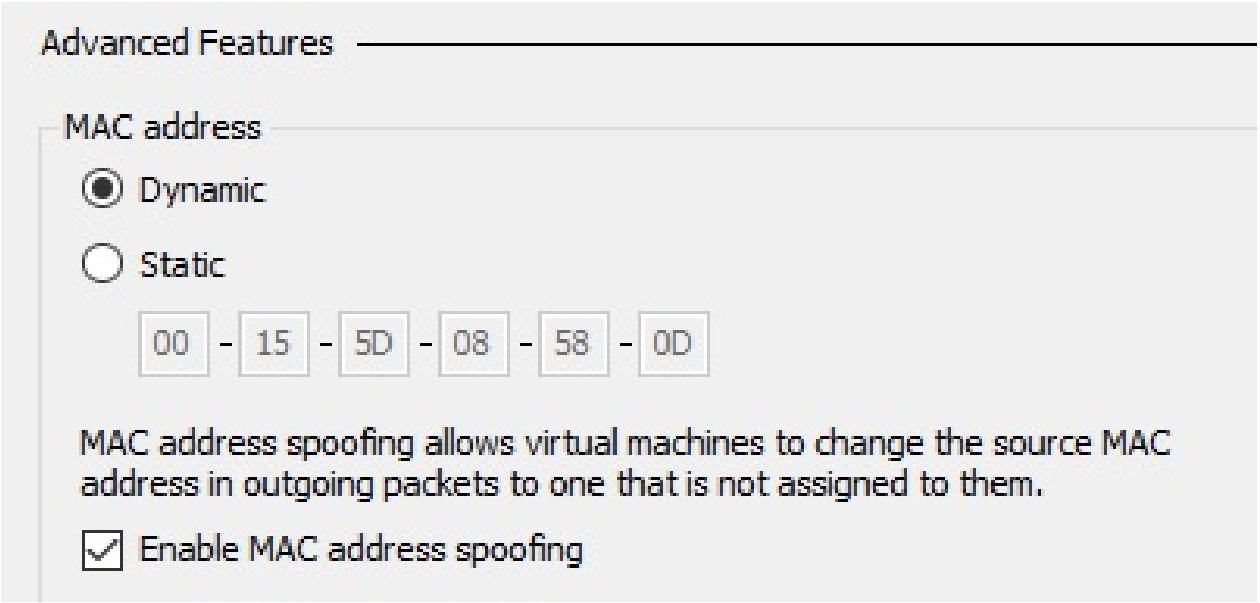
Enabling MAC address spoofing on VMs

Remember *when we talked about unicast NLB and how the physical MAC address of the NIC gets replaced with a virtual MAC address that is used for NLB array communications?* Yeah, virtual machines don't like that. If you are load balancing physical servers with physical NICs, you can skip this section. But many of you will be running web servers that are VMs. Whether they are hosted with Hyper-V, VMware, or some other virtualization technology, there is an extra option in the configuration of the virtual machine itself that you will have to make, so that your VM will happily comply with this MAC addressing change.

The name of this setting will be something along the lines of Enable MAC address spoofing, though the specific name of the function could be different depending on what virtualization technology you use. The setting should be a simple checkbox that you have to enable in order to make MAC spoofing work properly. Make sure to do this for all of your virtual NICs upon which you plan to utilize NLB. Keep in mind, this is a per-NIC setting, not a per-VM setting. If you have multiple NICs on a VM, you may have to check the box for each NIC, if you plan to use them all with load balancing.

The VM needs to be shut down in order to make this change, so I have shut down my WEB1 and WEB2 servers. Now find the checkbox and enable it. Since everything that I use is based on Microsoft technology, I am of course using Hyper-V as the platform for my virtual machines here in the lab. Within Hyper-V, if I right-click on my WEB1 server and

head into the VM's settings, I can then click on my network adapter to see the various pieces that are changeable on WEB1's virtual NIC. In the latest versions of Hyper-V, this setting is listed underneath the NIC properties, inside the section titled Advanced Features. And there it is, my Enable MAC address spoofing checkbox. Simply click on that to enable, and you're all set:

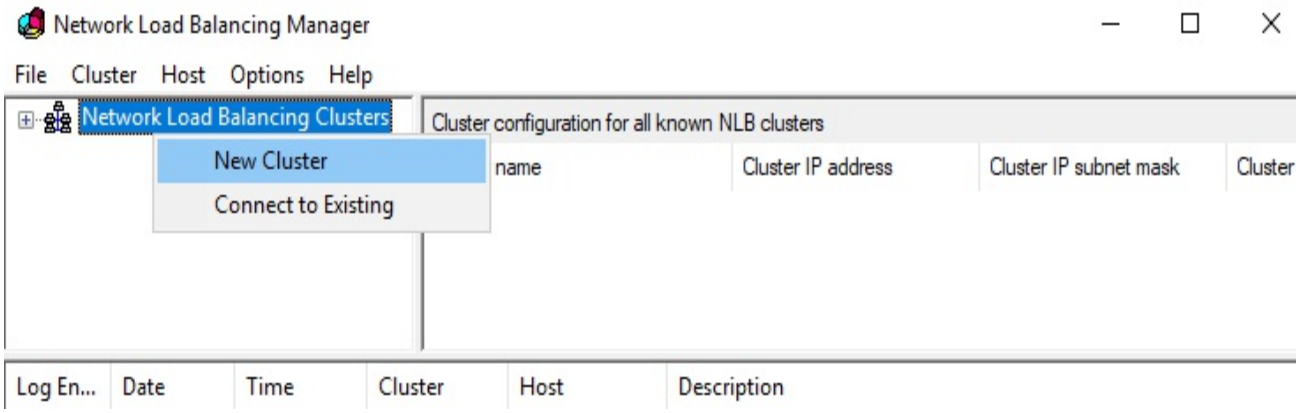


If Enable MAC address spoofing is grayed out, remember that the virtual machine must be completely shut down before the option appears. Shut it down, then open up Settings and take another look. The option should now be available to select.

Configuring NLB

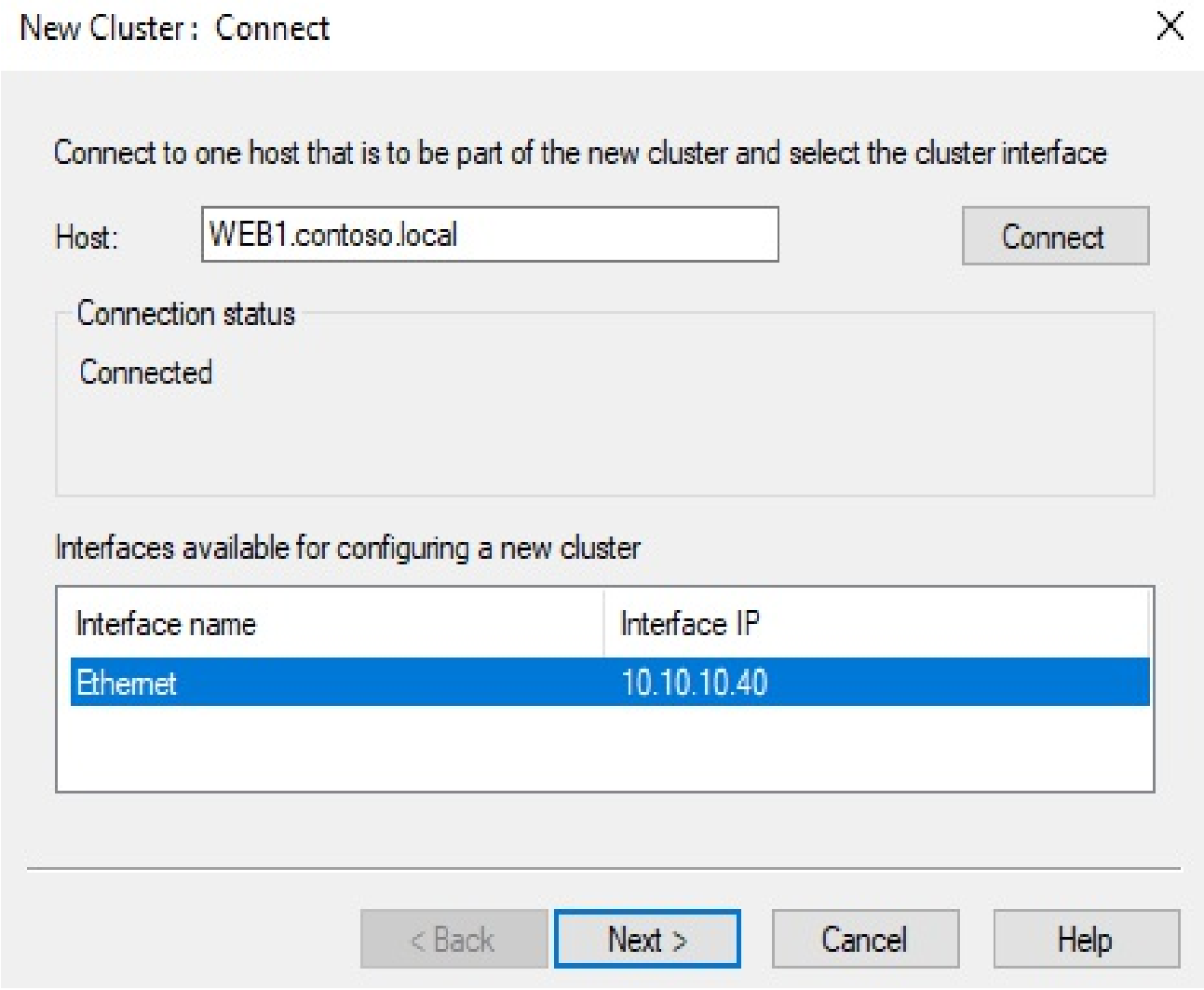
Let's summarize where we are at this point. I have two web servers, WEB1 and WEB2, and they each currently have a single IP address. Each server has IIS installed, which is hosting a single website. I have enabled MAC address spoofing on each (because these servers are virtual machines), and I just finished installing the NLB feature onto each web server. We now have all of the parts and pieces in place to be able to configure NLB and get that web traffic split between both servers.

I will be working from WEB1 for the initial configuration of NLB. Log into this, and you will see that we have a new tool in the list of tools that are available inside Server Manager, called Network Load Balancing Manager. Go ahead and open up that console. Once you have NLB Manager open, right-click on Network Load Balancing Clusters, and choose New Cluster, as shown in the following screenshot:



When you create a new cluster, it is important to note that currently, there are zero machines in this cluster. Even the server where we are

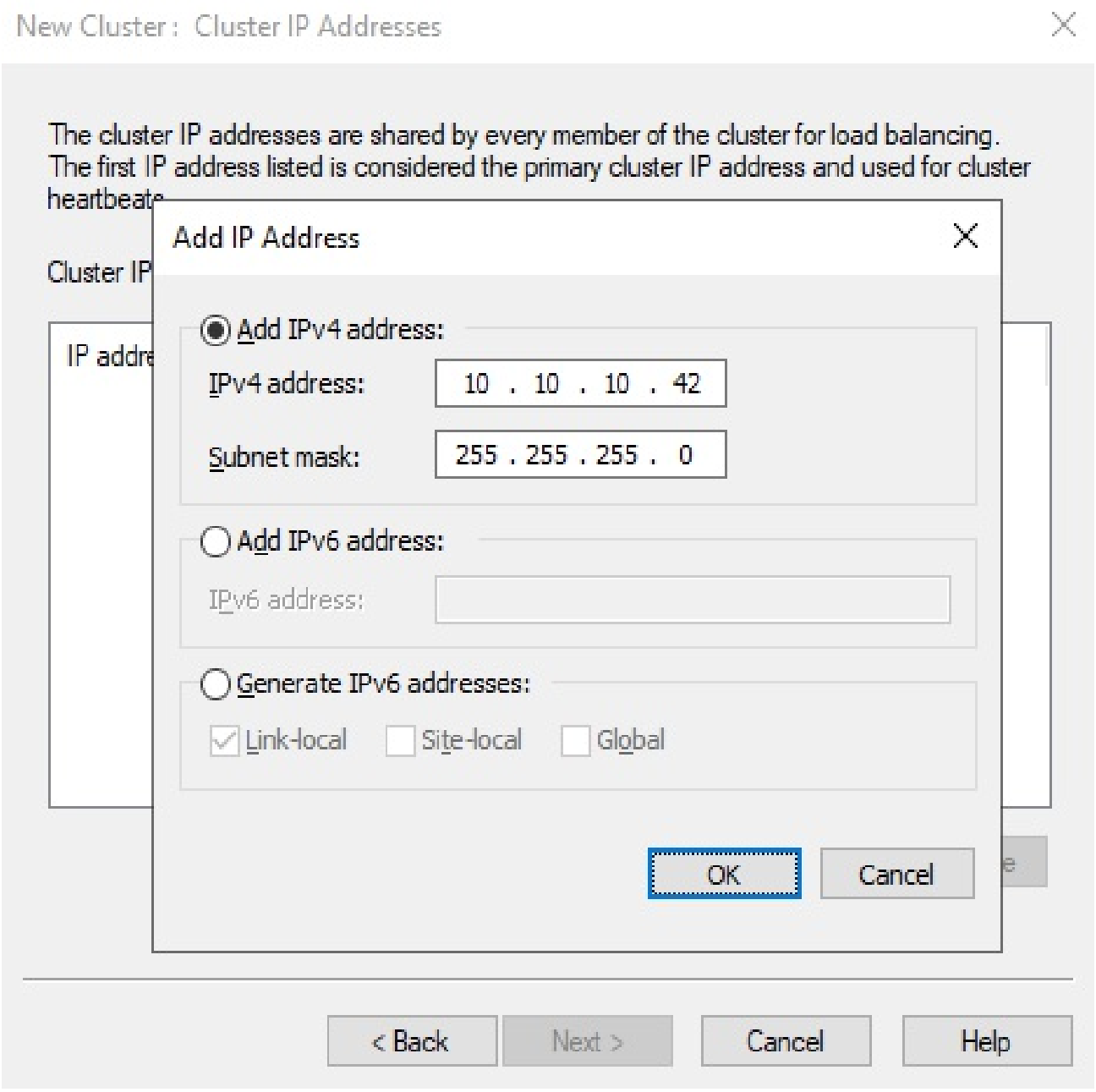
running this console is not automatically added to the cluster, and we must remember to manually place it into this screen. So first, I am going to type in the name of my WEB1 server and click on Connect. After doing that, the NLB Manager will query WEB1 for NICs and will give me a list of available NICs upon which I could potentially set up NLB:



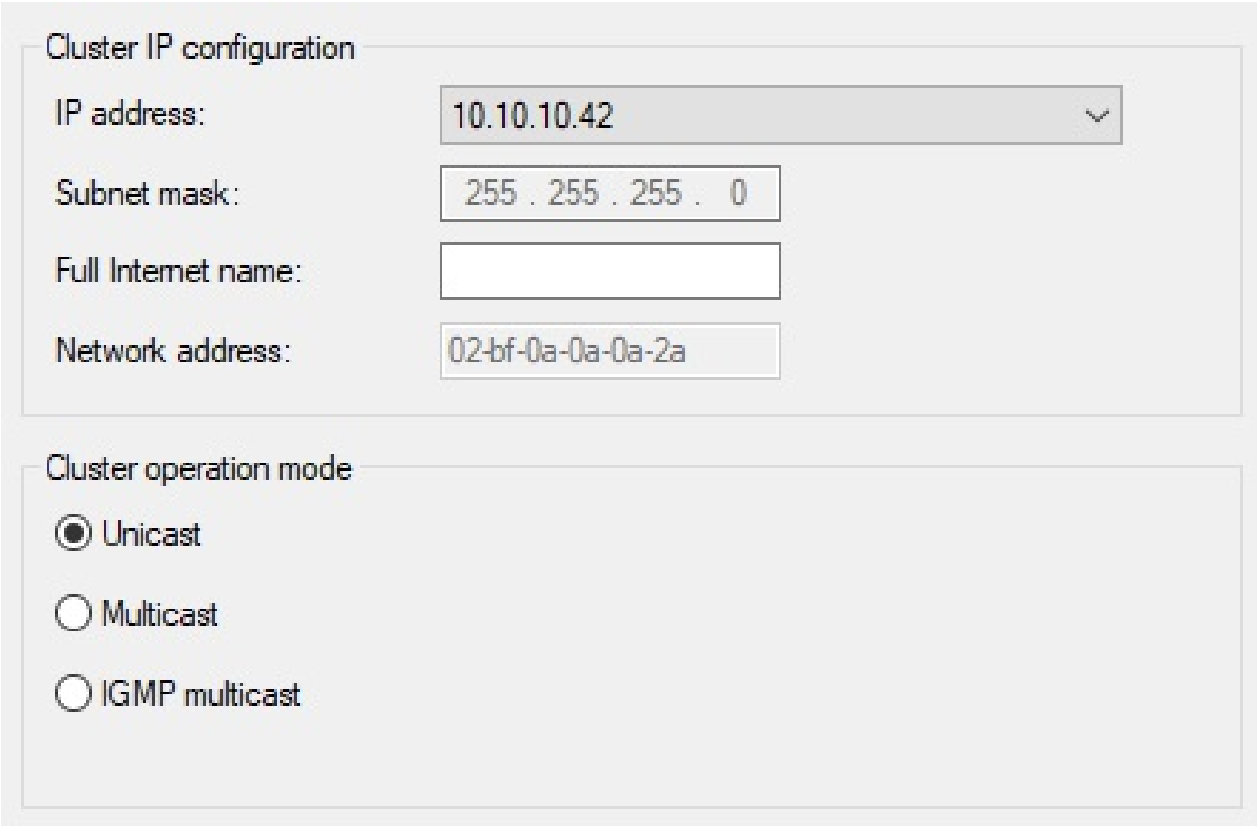
Since I only have one NIC on this server, I simply leave it selected and click on Next. The following screenshot gives you the opportunity to input additional IP addresses on WEB1, but since we are only running

one IP address, I will leave this screen as is, and click on Next again.

Now we have moved on to a window asking us to input cluster IP addresses. These are the VIPs that we intend to use to communicate with this NLB cluster. As stated earlier, my VIP for this website is going to be `10.10.10.42`, so I click on the Add... button and input that IPv4 address along with its corresponding subnet mask:

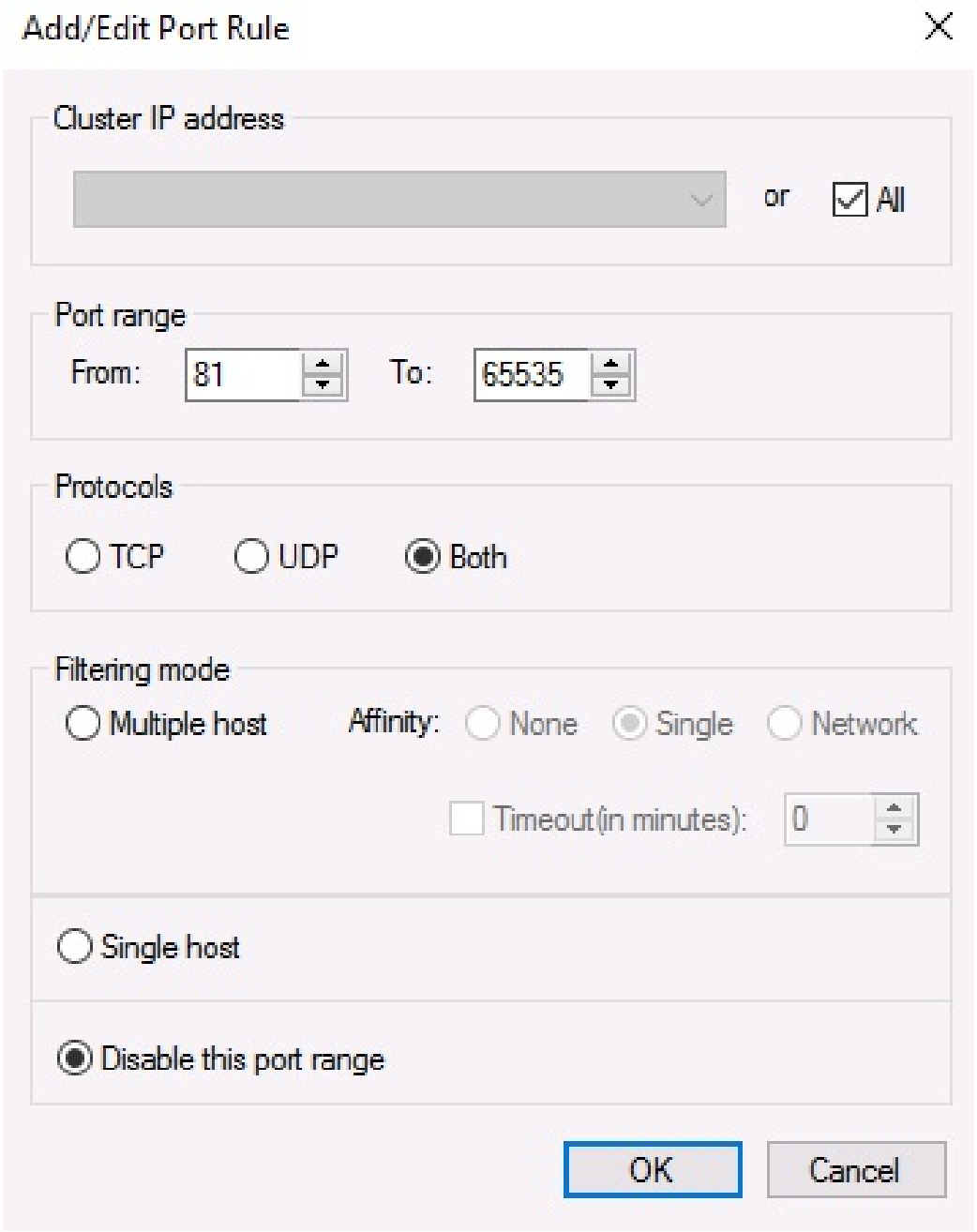


One more click of the Next button, and we can now see our option for which Cluster operation mode we want to run. Depending on your network configuration, choose between Unicast, Multicast, and IGMP multicast:



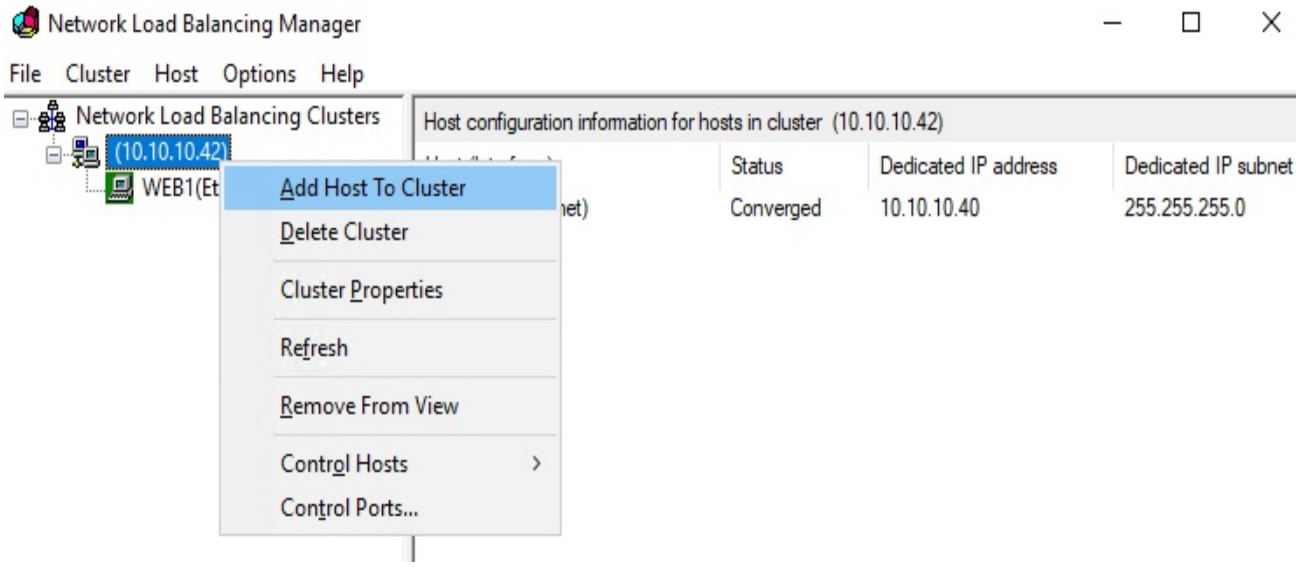
The following screenshot of our NLB wizard allows you to configure port rules. By default, there is a single rule that tells NLB to load balance any traffic coming in on any port, but you can change this if you want. I don't see a lot of people in the field specifying rules here to distribute specific ports to specific destinations, but one neat feature in this screenshot is the ability to disable certain ranges of ports.

That function could be very useful if you want to block unnecessary traffic at the NLB layer. For example, the following screenshot shows a configuration that would block ports `81` and higher from being passed through the NLB mechanism:



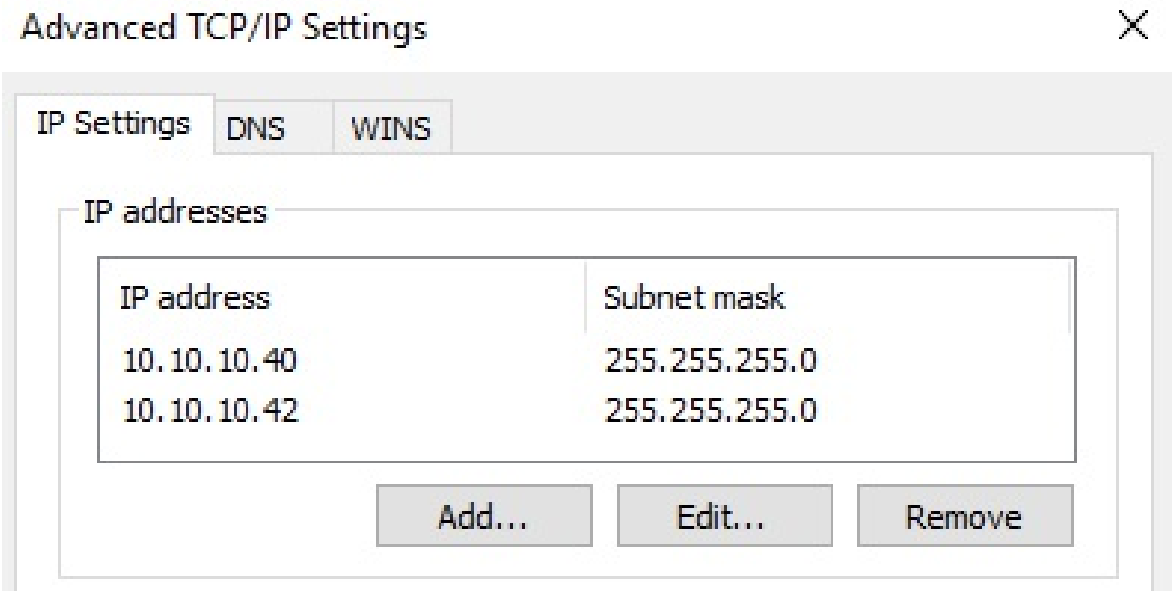
Finish that wizard, and you have now created an NLB cluster! However, at this point we have only specified information about the VIP, and about the WEB1 server. We have not established anything about WEB2. We are running an NLB array, but currently that array has just a single node inside of it, so traffic to the array is all landing on WEB1. Right-

click on the new cluster and select Add Host To Cluster:



Input the name of our WEB2 server, click on Connect, and walk through the wizard in order to add the secondary NLB node of WEB2 into the cluster. Once both nodes are added to the cluster, our NLB array, or cluster, is online and ready to use. (See, I told you that the word *cluster* is used in a lot of places, even though this is not talking about a failover cluster at all!)

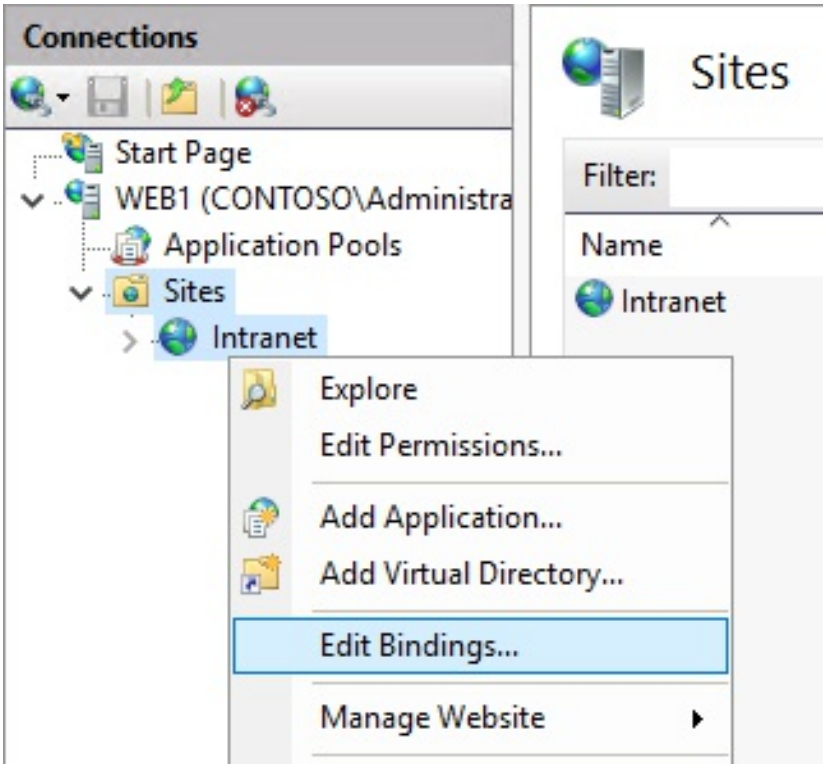
If you take a look inside the NIC properties of our web servers, and click on the Advanced button inside TCP/IPv4 properties, you can see that our new cluster IP address of `10.0.0.42` has been added to the NICs. Each NIC will now contain both the DIP address assigned to it, as well as the VIP address shared in the array:



The traffic that is destined for the `10.10.10.42` IP address is now starting to be split between the two nodes, but right now the websites that are running on the WEB1 and WEB2 servers are configured to only be running on the dedicated `10.10.10.40` and `10.10.10.41` IP addresses, so we need to make sure to adjust that next.

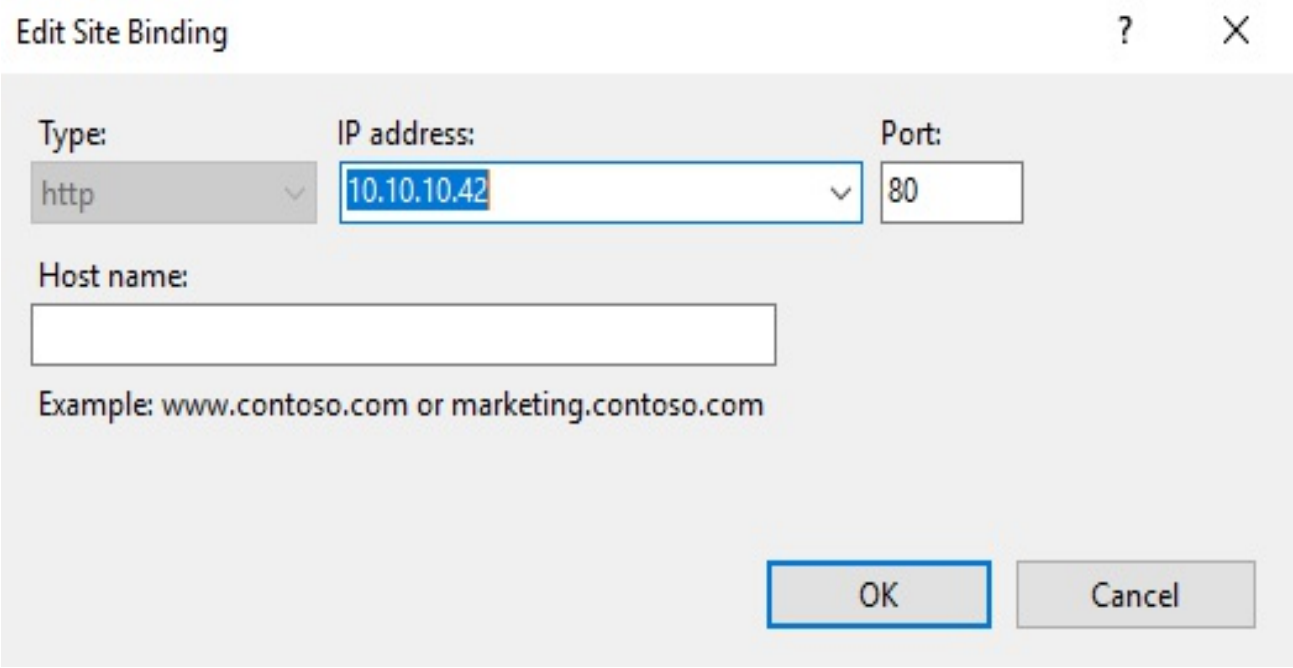
Configuring IIS and DNS

Just a quick step within IIS on each of our web servers should get the website responding on the appropriate IP address. Now that the NLB configuration has been established and we confirmed that the new `10.10.10.42` VIP address has been added to the NICs, we can use that IP address as a website binding. Open up the IIS management console, and expand the Sites folder so that you can see the properties of your website. Right-click on the site name, and choose Edit Bindings...:

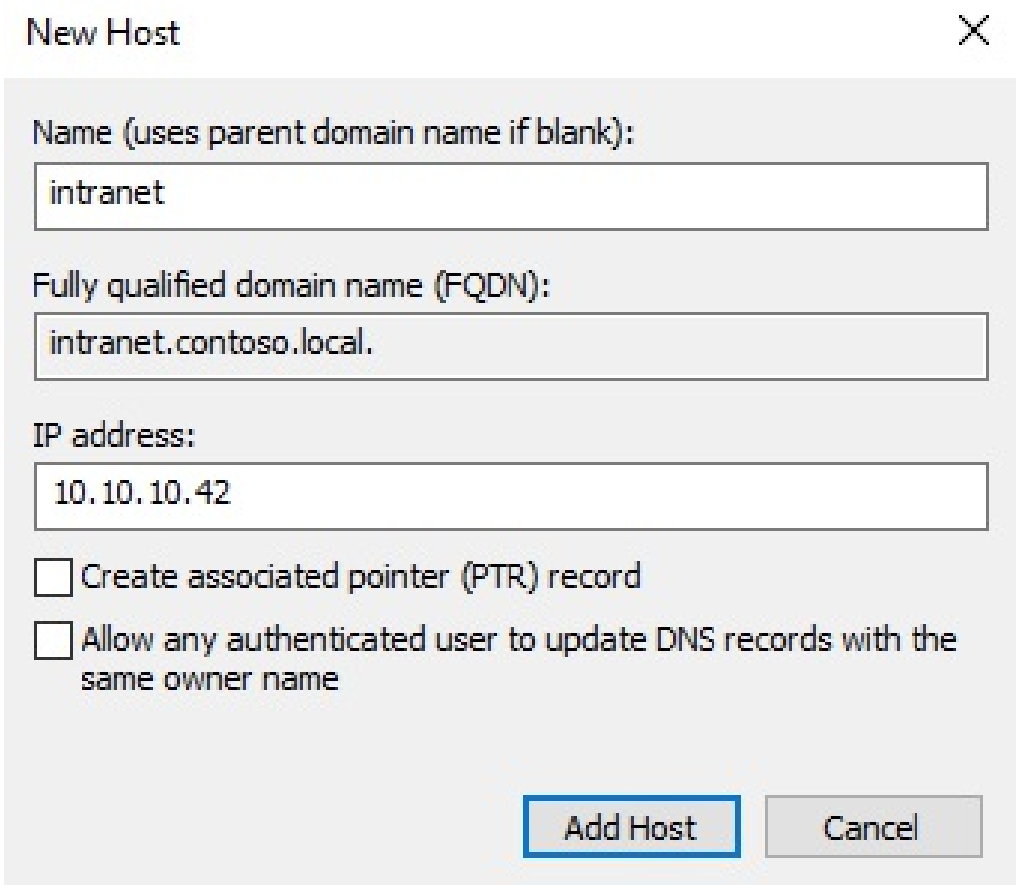


Once inside Site Bindings, choose the binding that you want to manipulate, and click on the Edit... button. This intranet website is just a simple HTTP site, so I am going to choose my HTTP binding for this

change. The binding is currently set to `10.10.10.40` on WEB1, and `10.10.10.41` on WEB2. This means that the website is only responding to traffic that comes in on these IP addresses. All I have to do is change that IP address drop-down menu to the new VIP, which is `10.10.10.42`. After making this change (on both servers) and clicking on OK, the website is immediately responding to traffic coming in through the `10.10.10.42` IP address:



Now we come to the last piece of the puzzle: DNS. Remember, we want the users to have the ability to simply enter `http://intranet` into their web browsers in order to browse this new NLB website, so we need to configure a DNS host A record accordingly. That process is exactly the same as any other DNS host record; simply create one and point `intranet.contoso.local` to `10.10.10.42`:


New Host
Name (uses parent domain name if blank):
intranet
Fully qualified domain name (FQDN):
intranet.contoso.local.
IP address:
10.10.10.42
Create associated pointer (PTR) record
Allow any authenticated user to update DNS records with the same owner name
Add Host
Cancel

Testing it out

Is NLB configured? Check.

Are the IIS bindings updated? Check.

Has the DNS record been created? Check.

We are ready to test this thing out. If I open up an internet browser on a client computer and browse to `http://intranet`, I can see the website:



But *how can we determine that load balancing is really working?* If I continue refreshing the page, or browse from another client, I continue accessing `http://intranet`, and eventually the NLB mechanism will decide that a new request should be sent over to WEB2, instead of WEB1. When this happens, I am presented with this page instead:



As you can see, I modified the content between WEB1 and WEB2 so that I could distinguish between the different nodes, just for the purposes of this test. If this were a real production intranet website, I would want to make sure that the content of both sites was exactly the same, so that users were completely unaware of the NLB even happening. All they need to know is that the website is going to be available and working, all of the time.

Flushing the ARP cache

Earlier, we had a little discussion about how switches keep a cache of ARP information, which lessens the time those switches need to take when deciding where packets should flow. When you assign an NIC an IP address, the MAC address of that NIC gets associated with the IP address inside the ARP table of certain pieces of networking equipment. Switches, routers, firewalls – these tools commonly have what we refer to as an ARP table, and therefore they have a set of data in that table that is known as the ARP cache.

When configuring NLB, particularly unicast, the NIC's MAC address gets replaced with a new, virtual MAC address. Sometimes the switches and networking equipment are very quick to catch on to this change, and they associate the new MAC address with the new IP address, and everything works just fine. However, I find that when configuring NLB, the following is generally true: *The smarter and more expensive your networking equipment is, the dumber it gets when configuring NLB*. What I mean is that your networking equipment might continue to hold onto the old MAC address information that is stored in its ARP table, and doesn't get updated to reflect the new MAC addressing.

What does this look like in real life? Network traffic will stop flowing to or from those NICs. Sometimes when you establish NLB and it turns itself on, all network traffic will suddenly stop cold to or from those network interfaces. *What do you need to do to fix this situation?* Sometimes you can wait it out, and within a few minutes, hours, or even a few days the switches will drop the old ARP info and allow the new virtual MACs to register themselves in that table. *What can you do to*

speed up this process? Flush the ARP cache.

The procedure for doing this will be different depending on what kind of networking equipment you are working on – whether it is a switch or router, what brand it is, what model it is, and so on. But each of these guys should have this capability, and it should be named something along the lines of *flushing the ARP cache*. When you run this function on your equipment, it cleans out that ARP table, getting rid of the old information that is causing you problems and allowing the new MAC addresses to register themselves appropriately into the fresh table.

I only wanted to point this out in the event that you configure NLB, only to see traffic flow cease on your server. More than likely, you are dealing with the ARP cache being stuck on one or more pieces of network equipment that is trying to shuttle the traffic to and from your server.

Failover clustering

We have established that NLB is a great solution for stateless applications, with a prime example being websites that you want to make highly available. *What about other server roles or functions that you want to make redundant?* Well, the opposite of stateless is stateful, so *how about giving high availability to stateful pieces of technology?*

Failover clustering provides this level of capability, and can be used in cases where the nodes within the cluster are accessing shared data. This is a key factor in the way failover clustering is designed, the storage used by the cluster nodes must be shared and accessible by each node that needs it. There are many different roles and services that can take advantage of failover clustering, but there are four specific technologies that seem to make up the majority of clusters running in data centers today: Hyper-V, file services, Exchange, and SQL. If you are working with any of these technologies – and chances are that you work with all of them – you need to look into the high-availability capabilities that can be provided for your infrastructure by use of failover clustering.

While failover clustering provided by Windows Server is Microsoft-built and has the capacity to work very well out of the box with many Microsoft roles and services, it is important to note that you can establish failover clustering for non-Microsoft applications as well. Third-party applications that run on Windows Servers in your environment, or even homegrown applications that have been built in-house, can also take advantage of failover clustering. As long as that application uses shared storage and you can specify the tasks that it needs to be able to perform against those applications for the clustering

administration tools – how to start the service, how to stop the service, how to monitor the service health, and so on – you can interface these custom services and applications with failover clustering and provide some major redundancy for just about any type of application.

Clustering Hyper-V hosts

One of the most powerful examples of failover clustering is displayed when combining clustering with Hyper-V. It is possible to build out two or more Hyper-V servers, cluster them together, and give them the capability to each host all of the virtual machines that are stored in that virtual environment. By giving all of the Hyper-V host servers access to the same shared storage where the virtual hard disks are stored, and configuring failover clustering between the nodes, you can create an incredibly powerful and redundant virtualization solution for your company. When a Hyper-V server goes down, the VMs that were running on that Hyper-V host will fail over to another Hyper-V host server and spin themselves up there instead.

After minimal service interruption while the VMs spin up, everything is back online automatically, without any administrative input. Even better, *how about when you need to patch or otherwise take a Hyper-V host server offline for maintenance?* You can easily force the VMs to run on a different member server in the cluster; they are live-migrated over to that server so there is zero downtime, and then you are free to remove the node for maintenance and finish working on it before reintroducing it to the cluster. We use virtual machines and servers for all kinds of workloads, so *wouldn't it be great if you could get rid of any single points of failure within that virtualization environment?* That is exactly what failover clustering can provide.

Virtual machine load balancing

In fact, not only does a Hyper-V cluster have the ability to quickly self-recover in the event of a Hyper-V server node going offline, but we now have some smart load-balancing logic working along with these clustered services. If your Hyper-V cluster is becoming overloaded with virtual machines, it makes sense that you would add another node to that cluster, giving the cluster more capability and computing power. But once the node is added, *how much work is involved in sliding some of the VMs over to this new cluster node?*

None! As long as you have VM load-balancing enabled, the cluster's weights will be evaluated automatically, and VM workloads will be live-migrated, without downtime, on the fly, in order to better distribute the work among all cluster nodes, including the new server. VM load-balancing can be run and evaluated on demand, whenever you deem fit, or can be configured to run automatically, taking a look at the environment every 30 minutes, automatically deciding whether any workloads should be moved around.

Clustering for file services

Clustering for file servers has been available for quite a while; this was one of the original intentions behind the release of clustering. Originally, file-server clustering was only useful for document and traditional file utilization, in other words, when knowledge-worker types of users need to access files and folders on a daily basis, and you want those files to be highly available. To this day, this general-purpose file-server clustering works in an active-passive scenario. When multiple file servers are clustered together for general-purpose file access, only one of those file-server nodes is active and presented to the users at a time. Only in the event of downtime on that node does the role gets flipped over to one of the other cluster members.

Scale-out file server

While general file-server clustering is great for ad hoc access of files and folders, it wasn't comprehensive enough to handle files that were continuously open or being changed. A prime example of these files are virtual hard disk files used by Hyper-V virtual machines.

Obviously, there was a need for virtual hard disk files to be redundant; losing these files would be detrimental to our businesses. Thankfully, hosting application-data workloads such as this is exactly what **Scale-Out File Server** (**SOFS**) was designed to do. If you plan to host virtual machines using Hyper-V, you will definitely want to check out the failover clustering capabilities that are available to use with Hyper-V services. Furthermore, if you intend to use clustered Hyper-V hosts, you should check out SOFS as an infrastructure technology to support that highly-available Hyper-V environment. SOFS helps support failover clustering by providing file servers with the capability to have multiple nodes online (active-active) that remain persistent between each other constantly. This way, if one storage server goes down, the others are immediately available to pick up the slack, without a cutover process that involves downtime. This is important when looking at the difference between storing static data, such as documents, and storing virtual hard disk files being accessed by VMs. The VMs are able to stay online during a file-server outage with SOFS, which is pretty incredible!

Clustering tiers

An overhead concept to failover clustering that is important to understand is the different tiers at which clustering can benefit you. There are two levels upon which you can use clustering: you can take an either/or approach and use just one of these levels of failover clustering, or you can combine both to really impress your high availability friends.

Application-layer clustering

Clustering at the application level typically involves installing failover clustering onto VMs. Using VMs is not a firm requirement, but is the most common installation path. You can mix and match VMs with physical servers in a clustering environment, as long as each server meets the installation criteria. This application mode of clustering is useful when you have a particular service or role running within the operating system that you want to make redundant. Think of this as more of a microclustering capability, where you are really digging in and making one specific component of the operating system redundant with another server node that is capable of picking up the slack in the event that your primary server goes down.

Host-layer clustering

If application clustering is micro, clustering at the host layer is more macro. The best example I can give of this is the one that gets most admins started with failover clustering in the first place: Hyper-V. Let's say you have two physical servers that are both hosting virtual machines in your environment. You want to cluster these servers together, so that all of the VMs being hosted on these Hyper-V servers are able to be redundant between the two physical servers. If a whole Hyper-V server goes down, the second one is able to spin up the VMs that had been running on the primary node, and after a minimal interruption of service, your VMs that are hosting the actual workloads in your environment are back up and running, available for users and their applications to tap into.

A combination of both

These two modes of using failover clustering mentioned earlier can certainly be combined together for an even better and more comprehensive high availability story. Let's let this example speak for itself: you have two Hyper-V servers, each one prepared to run a series of virtual machines. You are using host clustering between these servers, so if one physical box goes down, the other picks up the slack. That in itself is great, but you use SQL a lot, and you want to make sure that SQL is also highly available. You can run two virtual machines, each one a SQL server, and configure application-layer failover clustering between those two VMs for the SQL services specifically. This way, if something happens to a single virtual machine, you don't have to fail over to the backup Hyper-V server, rather your issue can be resolved by the second SQL node taking over. There was no need for a full-scale Hyper-V takeover by the second physical server, yet you utilized failover clustering in order to make sure that SQL was always online. This is a prime example of clustering on top of clustering, and by thinking along those lines, you can start to get pretty creative with all of the different ways that you can make use of clustering in your network.

How does failover work?

Once you have configured failover clustering, the multiple nodes remain in constant communication with each other. This way, when one goes down, they are immediately aware and can flip services over to another node to bring them back online. Failover clustering uses the registry to keep track of many per-node settings. These identifiers are kept synced across the nodes, and then when one goes down, those necessary settings are blasted around to the other servers and the next node in the cluster is told to spin up whatever applications, VMs, or workloads were being hosted on the primary box that went offline. There can be a slight delay in services as the components spin up on the new node, but this process is all automated and hands-off, keeping downtime to an absolute minimum.

When you need to cut services from one node to another as a planned event, such as for patching or maintenance, there is an even better story here. Through a process known as **live migration**, you are able to flip responsibilities over to a secondary node with zero downtime. This way, you can take nodes out of the cluster for maintenance or security patching, or whatever reason, without affecting the users or system uptime in any way. Live migration is particularly useful for Hyper-V clusters, where you will often have the need to manually decide which node your VMs are being hosted on, in order to accomplish work on the other node or nodes.

In many clusters, there is an idea of **quorum**. This means that if a cluster is split, for example, if a node goes offline or if there are multiple nodes that are suddenly unavailable through a network disconnect of

some kind, then quorum logic takes over in order to determine which segment of the cluster is the one that is still online. If you have a large cluster that spans multiple subnets inside a network, and something happens at the network layer that breaks cluster nodes away from each other, all the two sides of the cluster know is that they can no longer communicate with the other cluster members, and so both sides of the cluster would automatically assume that they should now take responsibility for the cluster workloads.

Quorum settings tell the cluster how many node failures can happen before action is necessary. By the entire cluster knowing the quorum configuration, it can help provide answers to those questions about which section of the cluster is to be primary in the event that the cluster is split. In many cases, clusters provide quorum by relying on a third party, known as a **witness**. As the name implies, this witness watches the status of the cluster and helps to make decisions about when and where failover becomes necessary. I mention this here as a precursor to our discussion on new clustering capabilities baked into Server 2019, one of which is an improvement in the way that witnesses work in small environments.

There is a lot more information to be gained and understood if you intend to create clusters large enough for quorum and witness settings. If you're interested in learning more, check out https://docs.microsoft.com/en-us/windows-server/storage/storage-spaces/understand-quorum.

Setting up a failover cluster

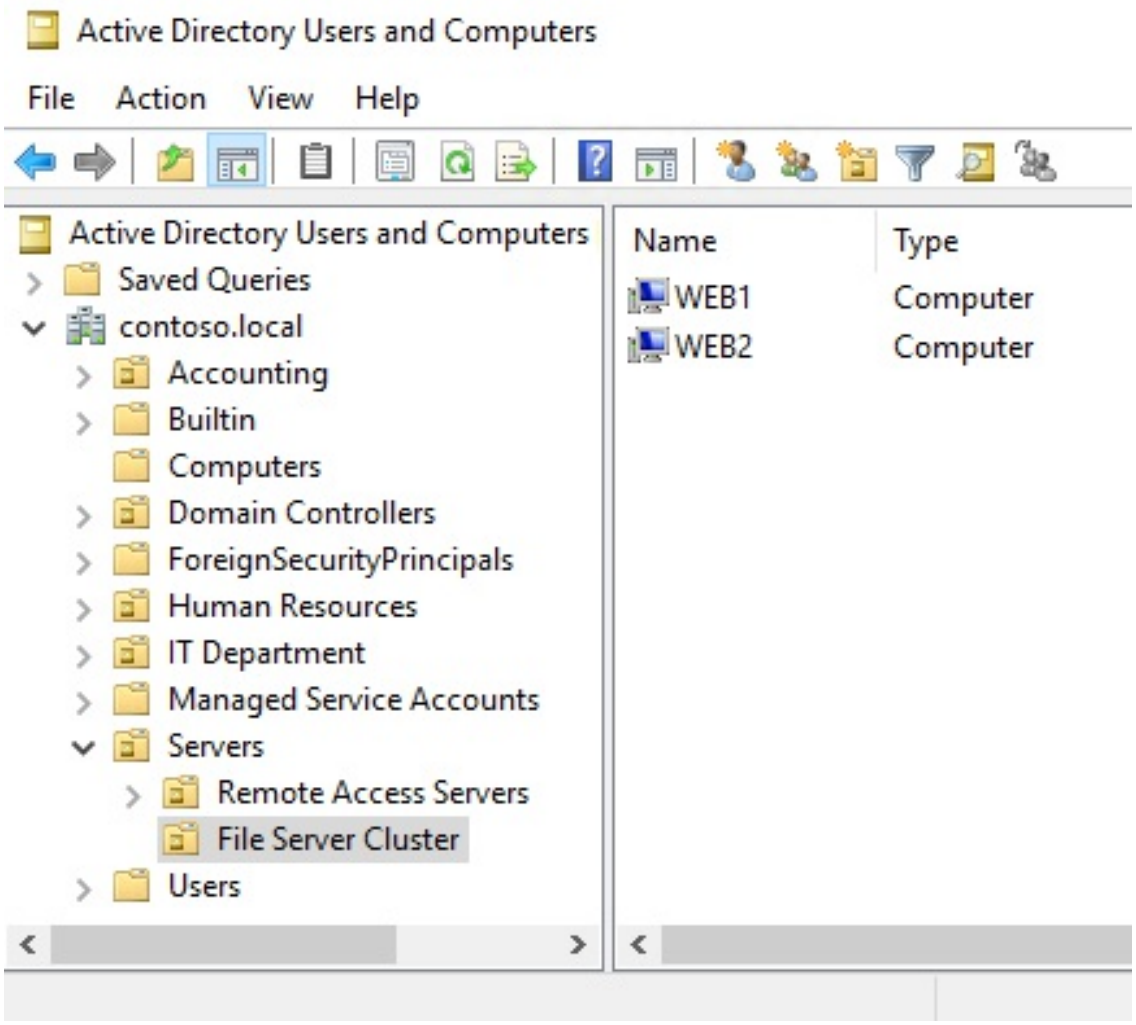
We are going to take a few minutes to set up a small cluster of servers, so that you can see the management tools and the places that have to be touched in order to accomplish this. I have now backed out all of the NLB config on my WEB1 and WEB2 servers that we set up earlier, so that they are just simple web servers at the moment, once again with no redundancy between them. Let's set up our first failover cluster and add both of these servers into that cluster.

Building the servers

We have two servers already running with Windows Server 2019 installed. Nothing special has been configured on these servers, but I have added the File Server role to both of them, because eventually I will utilize these as a cluster of file servers. The key point here is that you should have the servers as identical as possible, with the roles already installed that you intend to make use of within the cluster.

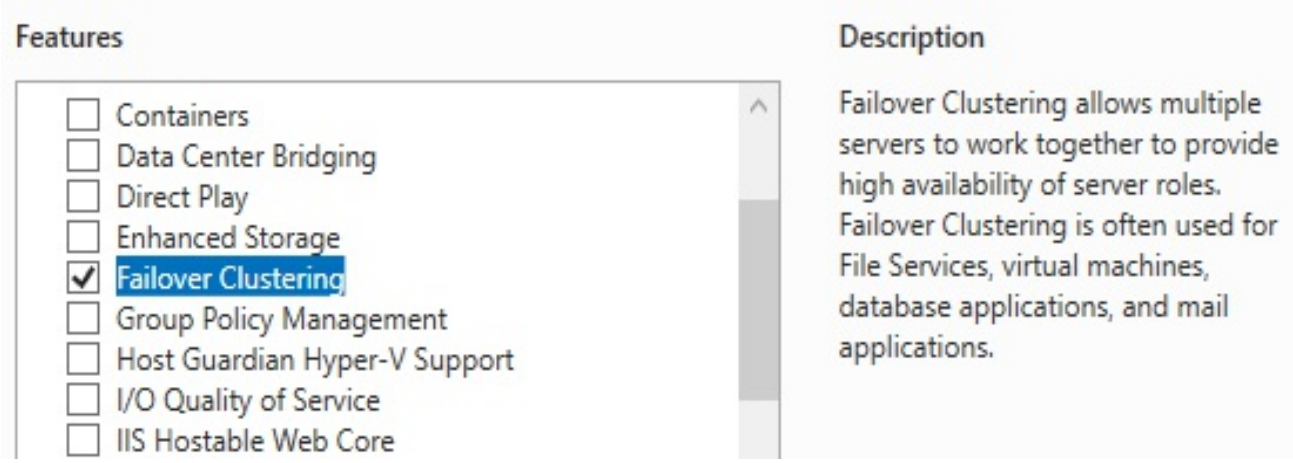
One other note during the building phase: if possible, it is a best practice with clustering that member servers belonging to the same cluster reside within the same **organizational unit** (**OU**) in **Active Directory** (**AD**). The reason for this is twofold: first, it ensures that the same GPOs are being applied to the set of servers, in an effort to make their configurations as identical as possible.

Second, during cluster creation, some new objects will be auto-generated and created in AD, and when the member servers reside in the same OU, these new objects will be created in that OU as well. It is very common with a running cluster to see all of the relevant objects in AD be part of the same OU, and for that OU to be dedicated to this cluster:


Active Directory Users and Computers
File
Action
View
Help
Active Directory Users and Computers
Saved Queries
contoso.local
Accounting
Builtin
Computers
Domain Controllers
ForeignSecurityPrincipals
Human Resources
IT Department
Managed Service Accounts
Servers
Remote Access Servers
File Server Cluster
Users
Name
Type
WEB1
Computer
WEB2
Computer

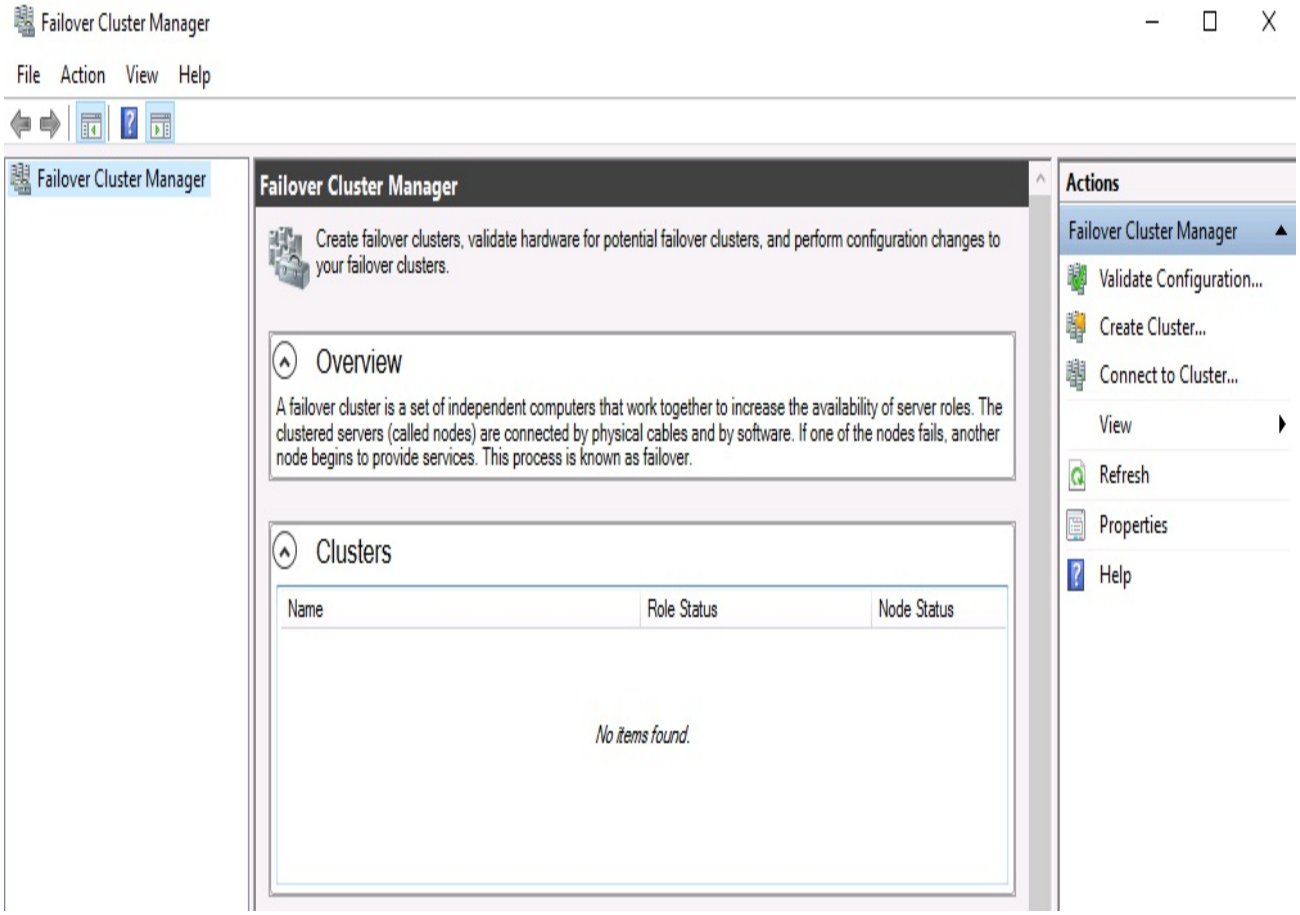
Installing the feature

Now that our servers are online and running, we want to install the clustering capabilities on each of them. Failover Clustering is a feature inside Windows Server, so open up the Add roles and features wizard and add it to all of your cluster nodes:



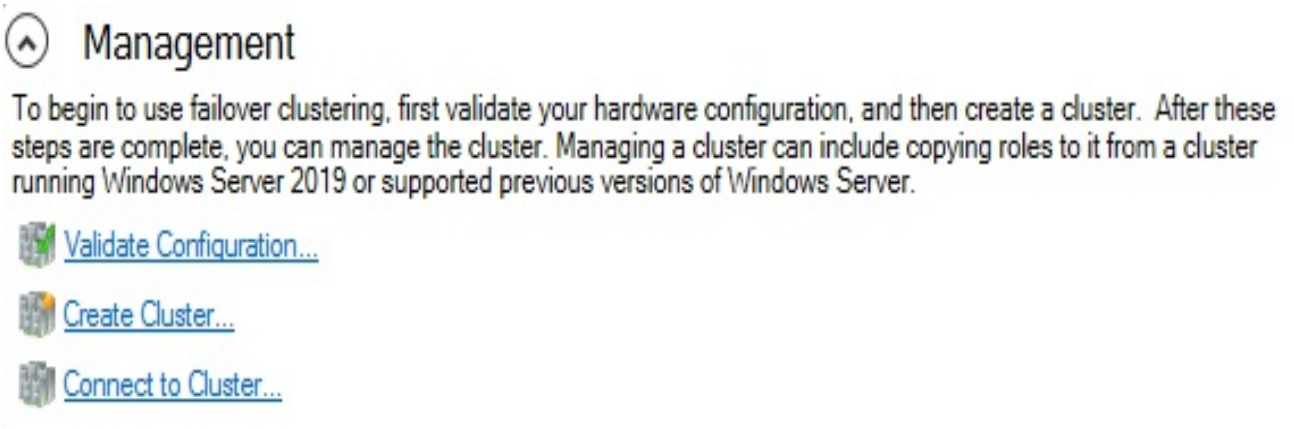
Running the failover cluster manager

As is the case with most roles or features that can be installed on Windows Server 2019, once implemented, you will find a management console for it inside the Tools menu of Server Manager. If I look inside there on WEB1 now, I can see that a new listing for Failover Cluster Manager is available for me to click on. I am going to open that tool, and start working on the configuration of my first cluster from this management interface:


Failover Cluster Manager
File Action View Help
Failover Cluster Manager
Failover Cluster Manager
Create failover clusters, validate hardware for potential failover clusters, and perform configuration changes to your failover clusters.
Overview
A failover cluster is a set of independent computers that work together to increase the availability of server roles. The clustered servers (called nodes) are connected by physical cables and by software. If one of the nodes fails, another node begins to provide services. This process is known as failover.
Clusters
Name
Role Status
Node Status
No items found.
Actions
Failover Cluster Manager
Validate Configuration...
Create Cluster...
Connect to Cluster...
View
Refresh
Properties
Help

Running cluster validation

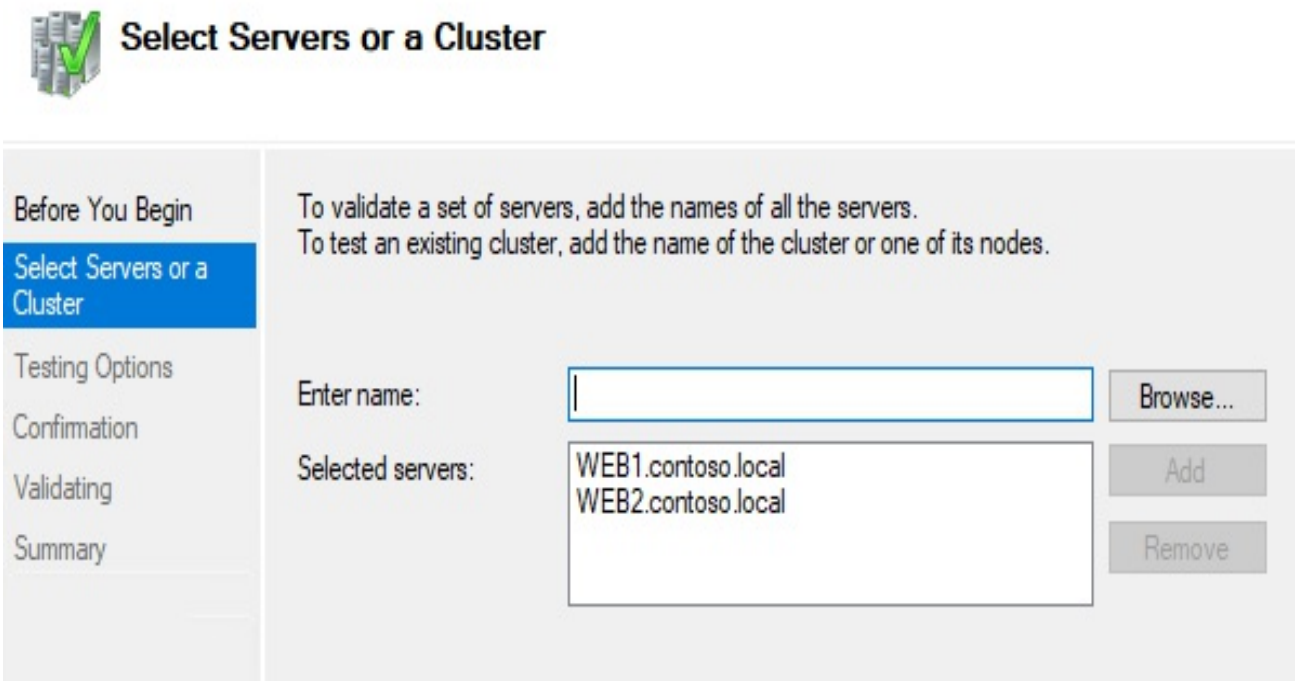
Now that we are inside Failover Cluster Manager, you will notice a list of tasks available to launch under the Management section of the console, near the middle of your screen:



Before we can configure the cluster itself or add any server nodes to it, we must first validate our hardware configuration. Failover clustering is a pretty complex set of technologies, and there are many places where misconfigurations or inconsistencies could set the whole cluster askew. Your intentions behind setting up a cluster are obviously for reliable redundancy, but even a simple mistake in the configuration of your member servers could cause problems large enough that a node failure would not result in automated recovery, which defeats the purpose of the cluster in the first place. In order to make sure that all of our *T's* are crossed and *I's* dotted, there are some comprehensive validation checks built into Failover Cluster Manager, sort of like a built-in best practices analyzer. These checks can be run at any time – before the cluster is built or after it has been running in production for years. In fact, if you

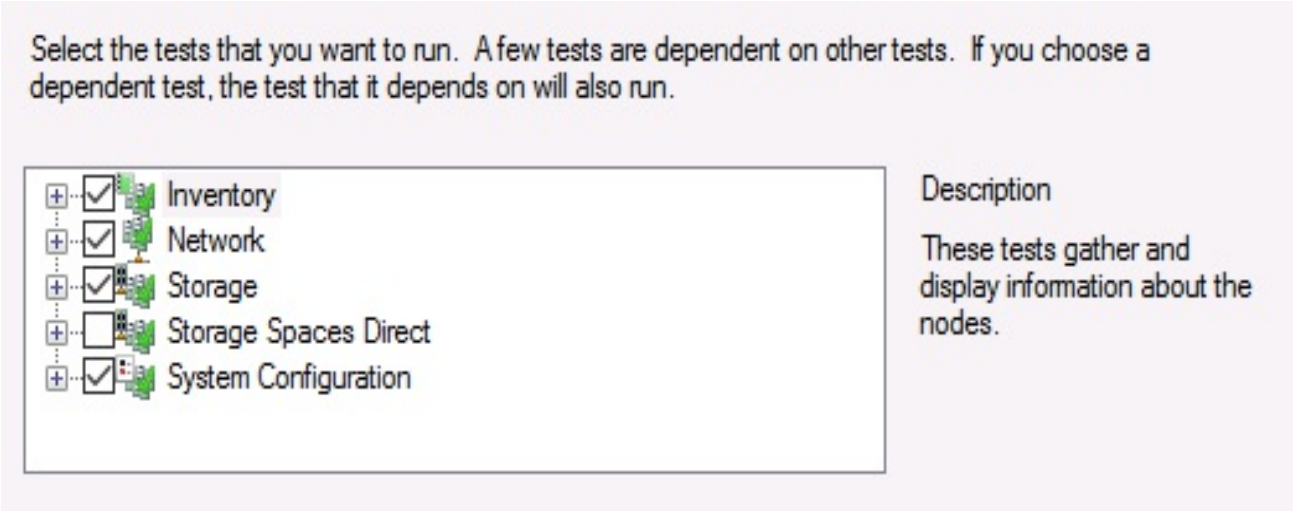
ever have to open a support case with Microsoft, it is likely that the first thing they will ask you to do is run the Validate Configuration tools and allow them to look over the output.

In order to start the validation process, click on the Validate Configuration... link. We are now launched into a wizard that allows us to select which pieces of the cluster technology we would like to validate. Once again, we must put on our Microsoft *centralized management theology* thinking caps, and realize that this wizard doesn't know or care that it is running on one of the member servers that I intend to be part of the cluster. We must identify each of the server nodes that we want to scan for validation checks, so in my case I am going to tell it that I want to validate the WEB1 and WEB2 servers:

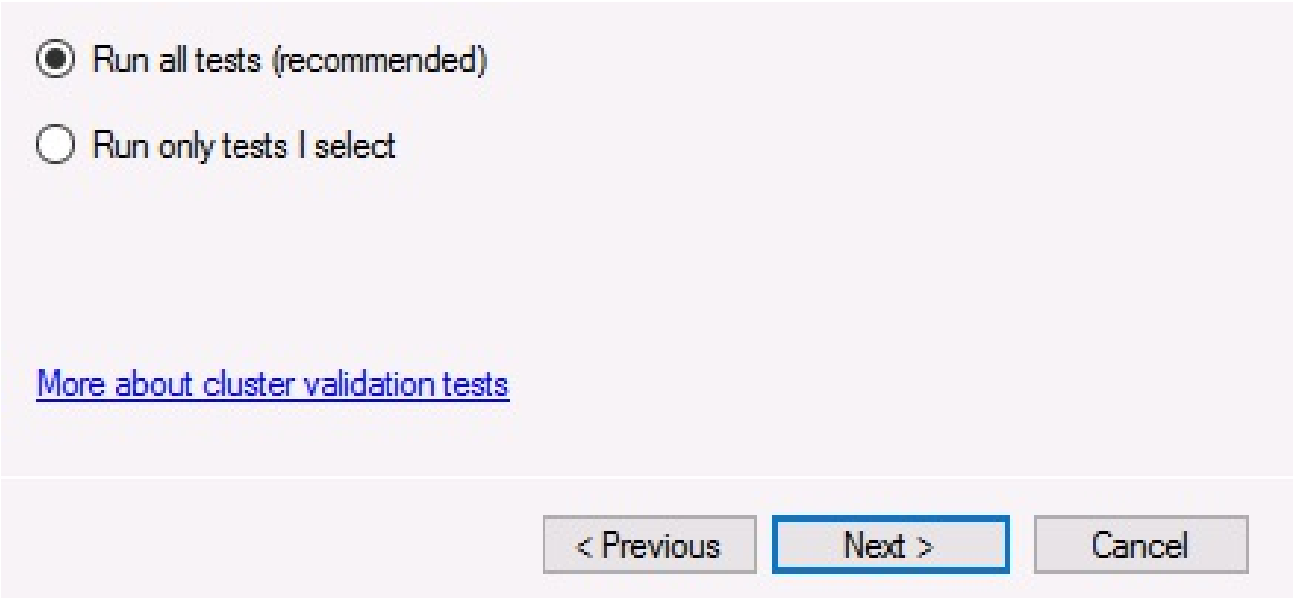


The Testing Options screen allows you to choose the Run only tests I select radio button and you will then be able to run only particular validation tests. Generally, when setting up a new cluster, you want to

run *all* of the tests so that you can ensure everything measures up correctly. On a production system, however, you may choose to limit the number of tests that run. This is particularly true with respect to tests against Storage, as those can actually take the cluster offline temporarily while the tests are being run, and you wouldn't want to interfere with your online production services if you are not working within a planned maintenance window:

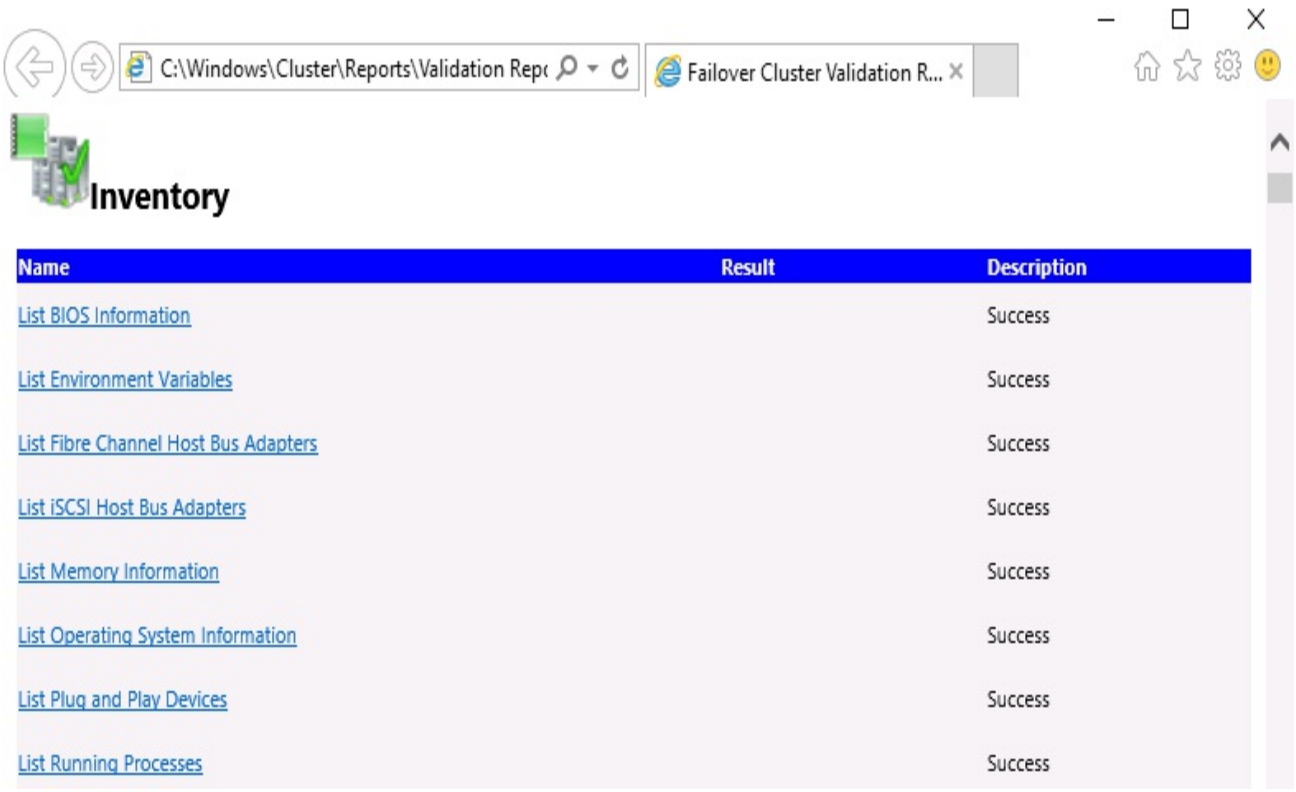


Since I am setting up a brand new cluster, I am going to let all of the tests run. So I will leave the recommended option selected for Run all tests (recommended), and continue:



Once the tests have completed, you will see a summary output of their results. You can click on the View Report... button in order to see a lot of detail on everything that was run. Keep in mind that there are three tiers of pass/fail. Green is *good* and red is *bad*, but yellow is more like *it'll work, but you're not running best practices*. For example, I only have one NIC in each of my servers; the wizard recognizes that, for my setup to be truly redundant in all aspects, I should have at least two. It'll let this slide and continue to work, but it is warning me that I could make this cluster even better by adding a second NIC to each of my nodes.

If you ever need to re-open this report, or grab a copy of it off the server for safekeeping, it is located on the server where you ran the tests, in `C:\Windows\Cluster\Reports`:

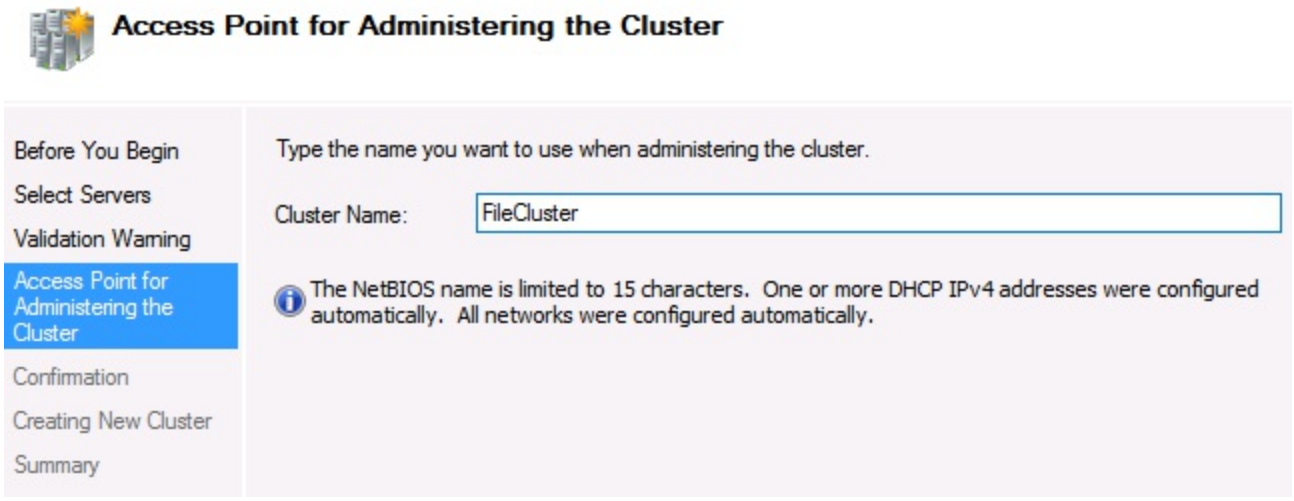


Remember, you can rerun the validation processes at any time to test your configuration using the Validate Configuration... task inside Failover Cluster Manager.

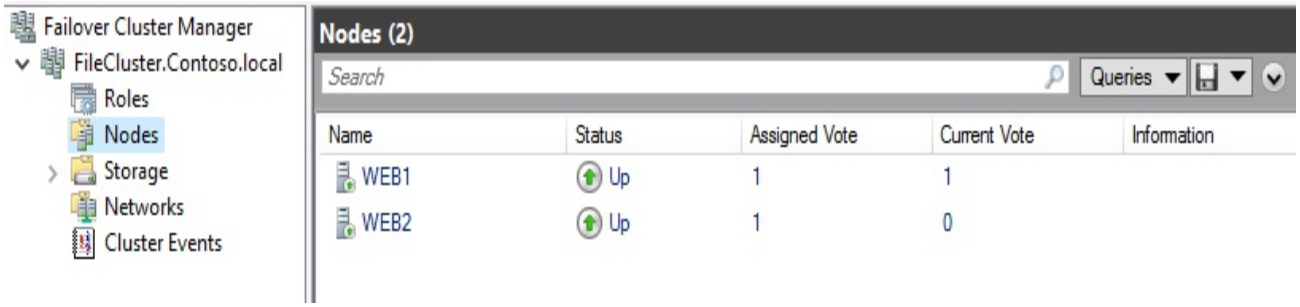
Running the Create Cluster wizard

The validation phase might take a while if you have multiple results that need fixing before you can proceed. But once your validation check comes back clean, you are ready to build out the cluster. For this, click on the next action that we have available in our Failover Cluster Manager console: Create Cluster....

Once again, we must first specify which servers we want to be part of this new cluster, so I am going to input my WEB1 and WEB2 servers. After this, we don't have a whole lot of information to input about setting up the cluster, but one very key piece of information comes in the Access Point for Administering the Cluster screen. This is where you identify the unique name that will be used by the cluster, and shared among the member servers. This is known as a **Cluster Name Object** (**CNO**), and after completing your cluster configuration, you will see this name show up as an object inside AD:



After finishing the wizard, you can see the new cluster inside the Failover Cluster Manager interface, and are able to drill down into more particular functions within that cluster. There are additional actions for things, such as Configure Role..., which will be important for setting up the actual function that this cluster is going to perform, and Add Node..., which is your spot to include even more member servers into this cluster down the road:



Recent clustering improvements in Windows Server

The clustering feature has been around for a while, but is continually being improved. There have been some big changes and additions to failover clustering in the two latest LTSC releases, Server 2016 and Server 2019. Some of the changes that we will discuss were originally introduced in 2016, so they are not brand new, but are still relevant to the way that we handle clusters in Server 2019 so they are worth mentioning here.

True two-node clusters with USB witnesses

When configuring quorum for a failover cluster, prior to Server 2019, a two-node cluster required three servers, because the witness for quorum needed to reside on a witness share of some kind, usually a separate file server.

Starting in 2019, that witness can now be a simple USB drive, and it doesn't even have to be plugged into a Windows Server! There are many pieces of networking equipment (switches, routers, and so on) that can accept a USB-based file storage media, and a USB stick plugged into such a networking device is now sufficient to meet the requirements for cluster witness. This is a win for enhanced clustering in small environments.

Higher security for clusters

A number of security improvements have been made to failover clustering in Windows Server 2019. Previous versions relied on **New Technology LAN Manager** (**NTLM**) for authentication of intra-cluster traffic, but many companies are taking proactive steps to disable the use of NTLM (at least the early versions) within their networks. Failover clustering can now do intra-cluster communication using Kerberos and certificates for validation of that networking traffic, removing the requirement for NTLM.

Another security/stability check that has been implemented when establishing a failover cluster file share witness is the blocking of witnesses stored inside DFS. Creating a witness inside a DFS share has never been supported, but the console previously allowed you to do so, which means that some companies did exactly this, and paid the price for it as this can cause cluster stability issues. The cluster-management tools have been updated to check for the existence of DFS namespace when creating a witness, and will no longer allow it to happen.

Multi-site clustering

Can I configure failover clustering across subnets? In other words, if I have a primary data center, and I also rent space from a CoLo down the road, or I have another data center across the country, *are there options for me to set up clustering between nodes that are physically separate?* There's a quick, easy answer here: yes, failover clustering doesn't care! Just as easily as if those server nodes were sitting right next to each other, clustering can take advantage of multiple sites that each host their own clustered nodes, and move services back and forth across these sites.

Cross-domain or workgroup clustering

Historically, we have only been able to establish failover clustering between nodes that were joined to the same domain. Windows Server 2016 brings the ability to move outside of this limitation, and we can even build a cluster without Active Directory being in the mix at all. In Server 2016 and 2019 you can, of course, still create clusters where all nodes are joined to the same domain, and we expect this will be the majority of installations out there. However, if you have servers that are joined to different domains, you can now establish clustering between those nodes. Furthermore, member servers in a cluster can now be members of a workgroup, and don't need to be joined to a domain at all.

While this expands the available capabilities of failover clustering, it also comes with a couple of limitations. When using multi-domain or workgroup clusters, you will be limited to only PowerShell as your cluster-management interface. If you are used to interacting with your clusters from one of the GUI tools, you will need to adjust your thinking cap on this. You will also need to create a local user account that can be used by clustering and provision it to each of the cluster nodes, and this user account needs to have administrative rights on those servers.

Migrating cross-domain clusters

Although establishing clusters across multiple domains has been possible for a few years, migrating clusters from one AD domain to another was not an option. Starting with Server 2019, this has changed. We have more flexibility in multi-domain clustering, including the ability to migrate clusters between those domains. This capability will help administrators navigate company acquisitions and domain-consolidation projects.

Cluster operating-system rolling upgrades

This new capability given to us in 2016 has a strange name, but is a really cool feature. It's something designed to help those who have been using failover clustering for a while be able to improve their environment. If you are running a cluster currently, and that cluster is Windows Server 2012 R2, this is definitely something to look into. Cluster Operating System Rolling Upgrade enables you to upgrade the operating systems of your cluster nodes from Server 2012 R2 to Server 2016, and then to Server 2019, *without downtime*. There's no need to stop any of the services on your Hyper-V or SOFS workloads that are using clustering, you simply utilize this rolling upgrade process and all of your cluster nodes will run the newer version of Windows Server. The cluster is still online and active, and nobody knows that it even happened. Except you, of course.

This is vastly different from the previous upgrade process, where in order to bring your cluster up to Server 2012 R2, you needed to take the cluster offline, introduce new server nodes running 2012 R2, and then re-establish the cluster. There was plenty of downtime, and plenty of headaches in making sure that it went as smoothly as possible.

The trick that makes this seamless upgrade possible is that the cluster itself remains running at the 2012 R2 functional level, until you issue a command to flip it over to the Server 2016 functional level. Until you issue that command, clustering runs on the older functional level, even on the new nodes that you introduce, which are running the Server 2016

operating system. As you upgrade your nodes one at a time, the other nodes that are still active in the cluster remain online and continue servicing the users and applications, so all systems are running like normal from a workload perspective. As you introduce new Server 2016 boxes into the cluster, they start servicing workloads like the 2012 R2 servers, but doing so at a 2012 R2 functional level. This is referred to as **mixed mode**. This enables you to take down even that very last 2012 R2 box, change it over to 2016, and reintroduce it, all without anybody knowing. Then, once all of the OS upgrades are complete, issue the `Update-ClusterFunctionalLevel` PowerShell command to flip over the functional level, and you have a Windows Server 2016 cluster that has been seamlessly upgraded with zero downtime.

Virtual machine resiliency

As you can successfully infer from the name, **virtual machine resiliency** is an improvement in clustering that specifically benefits Hyper-V server clusters. In the clustering days of Server 2012 R2, it wasn't uncommon to have some intra-array, or intra-cluster, communication problems. This sometimes represented itself in a transient failure, meaning that the cluster thought a node was going offline when it actually wasn't, and would set into motion a failover that sometimes caused more downtime than if the recognition patterns for a real failure would have simply been a little bit better in the first place. For the most part, clustering and the failover of cluster nodes worked successfully, but there is always room for improvement. That is what virtual machine resiliency is all about. You can now configure options for resiliency, giving you the ability to more specifically define what behavior your nodes will take during cluster-node failures. You can define things such as **resiliency level**, which tells the cluster how to handle failures. You also set your own **resiliency period**, which is the amount of time that VMs are allowed to run in an isolated state.

Another change is that unhealthy cluster nodes are now placed into a quarantine for an admin-defined amount of time. They are not allowed to rejoin the cluster until they have been identified as healthy and have waited out their time period, preventing situations such as a node that was stuck in a reboot cycle inadvertently rejoining the cluster and causing continuous problems as it cycles up and down.

Storage Replica (SR)

SR is a new way to synchronize data between servers. It is a data-replication technology that provides the ability for block-level data replication between servers, even across different physical sites. SR is a type of redundancy that we hadn't seen in a Microsoft platform prior to Windows Server 2016; in the past, we had to rely on third-party tools for this kind of capability. SR is also important to discuss on the heels of failover clustering, because SR is the secret sauce that enables multi-site failover clustering to happen. When you want to host cluster nodes in multiple physical locations, you need a way to make sure that the data used by those cluster nodes is synced continuously, so that a failover is actually possible. This data flow is provided by SR.

One of the neat data points about SR is that it finally allows a single-vendor solution, that vendor being Microsoft of course, to provide the end-to-end technology and software for storage and clustering. It is also hardware-agnostic, giving you the ability to utilize your own preference for the storage media.

SR is meant to be tightly integrated and one of the supporting technologies of a solid failover clustering environment. In fact, the graphical management interface for SR is located inside the Failover Cluster Manager software – but is of course also configurable through PowerShell – so make sure that you take a look into Failover Clustering and SR as a *better together* story for your environment.

Updated with Windows Server 2019 is the fact that SR is now available inside Server 2019 Standard Edition! (Previously, it required Datacenter,

which was prohibitive to some implementations.) Administration of SR is also now available inside the new **Windows Admin Center** (**WAC**).

Storage Spaces Direct (S2D)

S2D is a clustering technology, but I list it here separate from general failover clustering because S2D is a core component of the **software-defined data center** (**SDDC**) and has had so much focus on improvements over the past few years that it really is in a category of its own.

In a nutshell, S2D is a way to build an extremely efficient and redundant centralized, network-based storage platform, entirely from Windows Servers. While serving the same general purpose (file storage) as a traditional NAS or SAN device, S2D takes an entirely different approach in that it does not require specialized hardware, nor special cables or connectivity between the nodes of the S2D cluster.

In order to build S2D, all you need is Windows Servers; the faster the better, but they could be normal, everyday servers. These servers must be connected through networking, but there are no special requirements here; they simply all get connected to a network, just like any other server in your environment. Once you have these servers running, you can utilize clustering technologies or the new WAC to bind these servers together into S2D arrays.

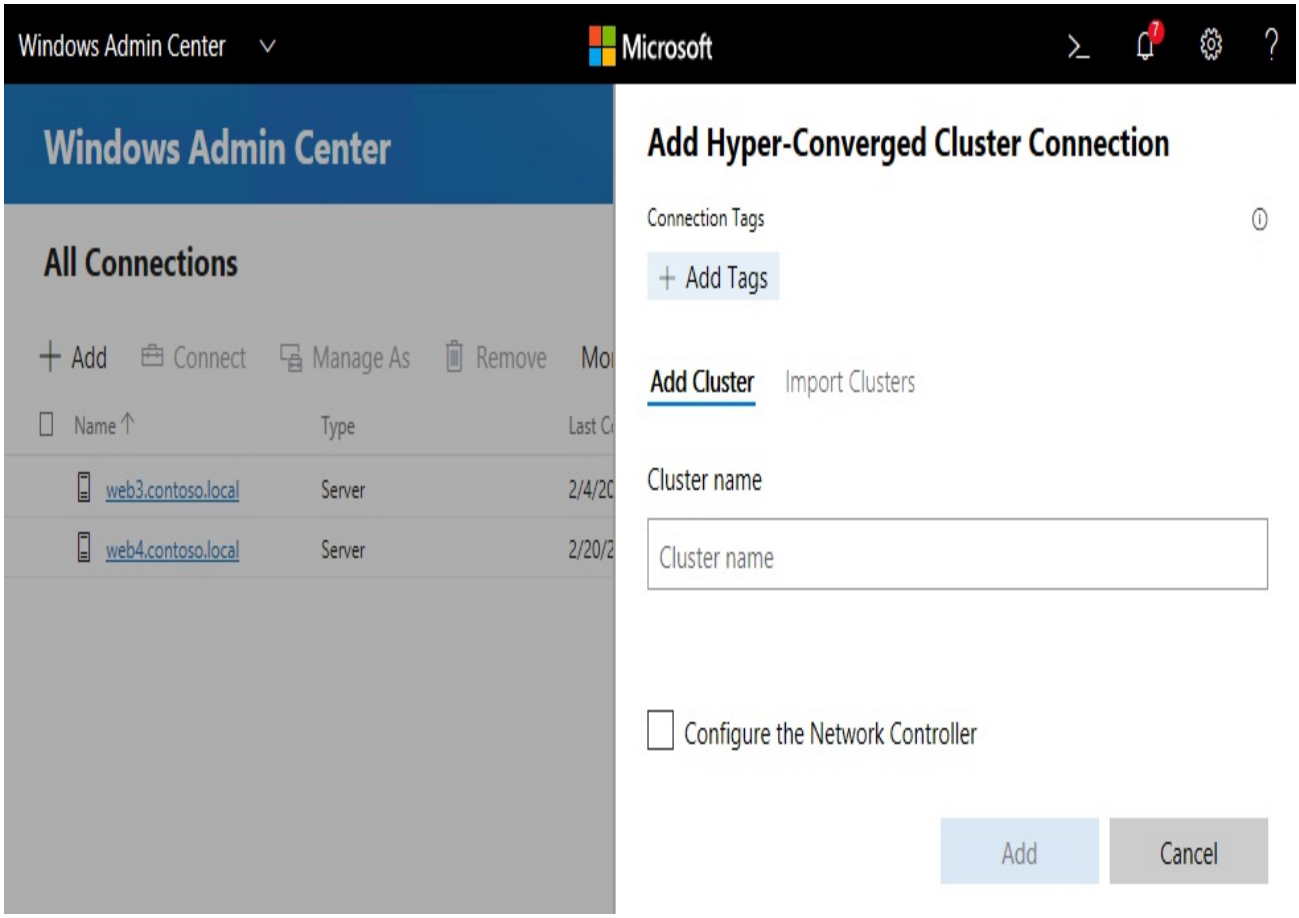
S2D is part of the overall **Hyper-Converged Infrastructure** (**HCI**) story, and is a wonderful way to provide extremely fast and protected storage for anything, but especially for workloads such as clusters of Hyper-V servers. As you already know, when building a Hyper-V server cluster, the nodes of that cluster must have access to shared storage upon which the virtual machine hard disk files will reside. S2D is the

best way to provide that centralized storage.

S2D will take the hard drives inside your S2D cluster node servers, and combine all of their space together into software-defined pools of storage. These storage pools are configured with caching capabilities, and even built-in fault-tolerance. You obviously wouldn't want a single S2D node, or even a single hard drive going offline, to cause a hiccup to your S2D solution, and of course Microsoft doesn't want that to happen either. So when you group servers and all of their hard drives together into these large pools of S2D storage, they are automatically configured with parity among those drives so that particular components going offline does not result in lost data, or even slow down the system.

S2D is the best storage platform for both SOFS and Hyper-V clusters.

While Server 2016-based S2D was configured mostly through PowerShell (which unfortunately means that a lot of administrators haven't tried it yet), Windows Server 2019 brings us the new WAC toolset, and WAC now includes built-in options for configuring an S2D environment:



S2D is one of those technologies that warrants its own book, but anyone looking to try out or get started with this amazing storage technology should start at https://docs.microsoft.com/en-us/windows-server/storage/storage-spaces/storage-spaces-direct-overview.

New in Server 2019

For those of you already familiar with the concept of S2D who want to know what is new or different in the Server 2019 flavor, here are some of the improvements that have come with this latest version of the operating system:

- **Improved use of Resilient File System (ReFS) volumes**: We now have deduplication and compression functions on ReFS volumes hosted by S2D.
- **USB witness**: We already discussed this one briefly, when using a witness to oversee an S2D cluster that is only two nodes, you can now utilize a USB key plugged into a piece of networking equipment, rather than running a third server for this witnessing purpose.
- **WAC**: WAC now includes tools and functionality for defining and managing S2D clusters. This will make adoption much easier for folks who are not overly familiar with PowerShell.
- **Improved capability**: We can now host four petabytes per cluster.
- **Improved speed**: While S2D has been fast since the very first version, we have some efficiency improvements in Server 2019. At last year's Ignite conference, Microsoft showcased an 8-node

S2D cluster that was capable of achieving 13,000,000 IOPs. Holy moly!

Summary

Redundancy is a critical component in the way that we plan infrastructure and build servers in today's world. Windows Server 2019 has some powerful capabilities built right into it that you can utilize in your own environments, starting today! I hope that by gleaning a little more information about both NLB and failover clustering, that you will be able to expand the capabilities of your organization by employing these techniques and stretching the limits of your service uptime. In my opinion, if there is any one avenue from this chapter for you to pursue, it is to start building your own HCI by utilizing S2D and failover clustering in order to create resiliency in your Hyper-V infrastructure. HCI will literally change the way that you work and give you some peace of mind that you didn't think was possible in a world aiming for 99.999% uptime. In the next chapter, we will look into PowerShell.

Questions

1. Which technology is more appropriate for making web server traffic redundant - Network Load Balancing or Failover Clustering?
2. In Network Load Balancing, what do the acronyms DIP and VIP stand for?
3. What are the three NLB modes?
4. In Windows Server 2019, is "Network Load Balancing" a role or a feature?
5. What roles are most often used with failover clustering?
6. What type of small device can now be used as a cluster quorum witness (this is brand new as of Server 2019)?
7. True or False—Storage Spaces Direct requires the use of SSD hard drives.

PowerShell

Let's be honest, many of us are still using Command Prompt on a daily basis. If you have cut over and are using the newer PowerShell prompt as a total replacement for Command Prompt, I applaud you! I, however, still tend to open up `cmd.exe` as a matter of habit, though with the most recent releases of Windows 10 and Windows Server 2019, I am definitely making a more conscious effort to use the newer, bluer, prettier, and more powerful interface that is PowerShell. In this chapter, we are going to explore some of the reasons that you should do so too. Other than the fact that Microsoft seems to have shrunk the default text size in Command Prompt to deter us from using it, which I find pretty funny, we are going to take a look at some of the technical reasons that PowerShell is far and away more useful and powerful than Command Prompt could ever dream to be.

In this chapter, we will cover the following topics:

- Why move to PowerShell?
- Working within PowerShell
- PowerShell Integrated Scripting Environment
- Remotely managing a server
- Desired State Configuration

Why move to PowerShell?

I don't think there is any question in people's minds that PowerShell is indeed the evolution of Command Prompt, but the reason that many of us still default to the old interface is that it still has all of the capabilities to accomplish what we need to do on our servers. What Command Prompt really contains is the ability to do the same things that we have always done from Command Prompt, and nothing else. Without realizing it, there are a lot of functions that you use the GUI to accomplish that cannot be done well from within a Command Prompt window.

The limitations within Command Prompt that force you into using your mouse to interface with the GUI do not exist with PowerShell. It is fully comprehensive and capable of modifying almost any aspect of the Windows operating system. How did PowerShell become so much more powerful than Command Prompt? It differs from any classic I/O shell in that it is built on top of .NET, and runs much more like a programming language than simple input and output commands.

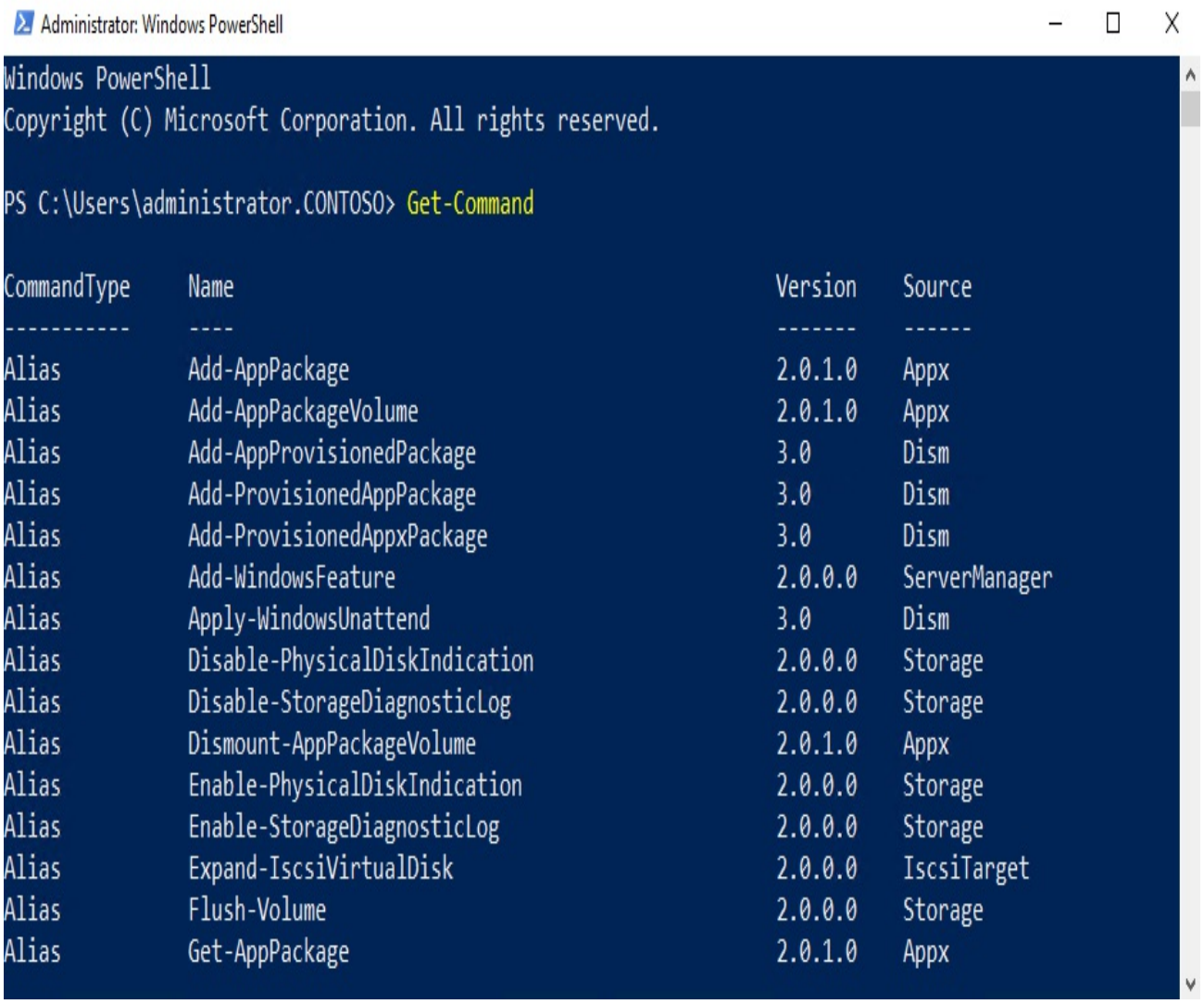
Cmdlets

Most of the functionality that a traditional server admin will use comes in the form of cmdlets (pronounced *command-lets*). These are commands that you run from within the PowerShell prompt, but you can think of them as tools rather than simple commands. Cmdlets can be used to both get information from a server and to set information and parameters on a server. Many cmdlets have intuitive names that begin with `get` or `set`, and similar to the way that most command-line interfaces work, each cmdlet has various switches or variables that can be configured and flagged at the end of the cmdlet, in order to make it do special things. It is helpful to understand that cmdlets are always built in a verb-noun syntax. You specify the action you want to accomplish, such as `get` or `set`, and then your noun is the piece inside Windows that you are trying to manipulate. Here are a few simple examples of cmdlets in PowerShell to give you an idea of what they look like, and how they are named in a fairly simple way:

- `Get-NetIPAddress`: With this cmdlet, we can see the IP addresses on our system.
- `Set-NetIPAddress`: We can use this guy to modify an existing IP address.
- `New-NetIPAddress`: This cmdlet allows us to create a new IP address on the computer.
- `Rename-Computer`: As we saw earlier in the book, `Rename-Computer` is a quick

and easy way to set the computer hostname of a system.

If you're ever struggling to come up with the name or syntax of a particular command, Microsoft's online Docs website (formerly and sometimes still called TechNet) has a full page of information dedicated to each cmdlet inside PowerShell. That can be incredibly useful, but sometimes you don't want to take the time to pop onto the internet just to find the name of a command that you are simply failing to remember at the moment. One of the most useful cmdlets in PowerShell shows you a list of all the available cmdlets. Make sure to check out `Get-Command`:

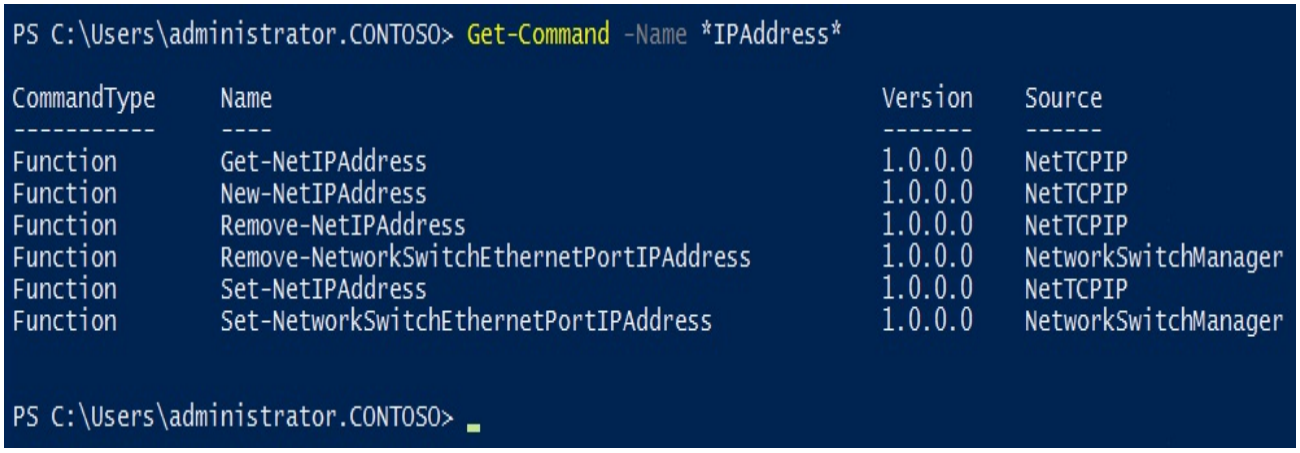


Whoa, there are pages and pages and pages of cmdlets! Rather than scrolling through the entire list in order to find the one you are looking for, it is easy to filter this list down based on any criteria that you would like. If we were interested in seeing only the commands that deal with IP addressing, we could give this a try:

```
Get-Command -Name *IPAddress*
```

The `Get-Command` cmdlet combined with the `-Name` parameter allows you to

selectively search for useful items in PowerShell that relate to any name or portion of a name:



PowerShell is the backbone

As you will discover in this chapter, interfacing with PowerShell puts all kinds of power at your fingertips. What we sometimes find, though, is that admins don't fully trust PowerShell, because they are used to taking these actions and making these changes from a graphical interface. After running a single PowerShell cmdlet to set a configuration that would have taken you a dozen different mouse clicks to accomplish the same thing, it is easy to think that it must not have actually done anything. That was too easy, and it processed my command way too quickly, right? I'd better go into that graphical interface anyway, just to double-check that PowerShell actually did the job.

When I started using PowerShell, I was tempted to do exactly that, all the time. But the more I used it and the more I started digging into those graphical interfaces themselves, the more I realized that I'm not the only one using PowerShell. A lot of the Administrative Tool GUIs use PowerShell too! Without even realizing it, you use PowerShell for quite a few tasks inside the Windows Server operating system. When you open up that management console for whatever you happen to be changing on the server, make your configurations, and then click on the Go or Finish button, how does that graphical console put your configuration into place? PowerShell. Under the hood, in the background, the console is taking the information that you input, plugging that information into PowerShell cmdlets, and running them in order to do the actual configuration work. Many of the Administrative Tools that we run from inside Server Manager take this approach, accepting changes and configurations from you and then formulating those settings into PowerShell commands that run in the background in

order to push the changes into action.

So, if you're hesitant to start using PowerShell because it just feels different, or you don't trust the process to be uniform to the way that it would have worked in the GUI, forget all of that. Because often when you are using mouse clicks to change settings on your server, you are actually invoking PowerShell cmdlets.

Scripting

The more you use PowerShell, the more powerful it becomes. In addition to running ad hoc, single commands and cmdlets, you have the ability to build extensive scripts that can accomplish all sorts of different things. I mentioned that PowerShell has similarities to a regular programming language, and scripting is where we start to navigate into that territory. PowerShell provides the ability to create script files, which we will do for ourselves coming up shortly, saving scripts for easy running of those same scripts time and time again. Variables can also be used, like in other forms of coding, so that you can provide variable input and objects that can be used by the scripts, in order to make them more flexible and squeeze even more functionality out of them.

Server Core

If there were any one area where I think we as server admins could do a better job of using the technology at our disposal, it is using PowerShell to fulfill the Microsoft model of centralized management. When we have a task that needs to be accomplished on a server, it is our default tendency to log into that server (usually via RDP), then use our mouse to start clicking around and doing the work. Logging into the server is becoming more and more unnecessary, and we could save a lot of time by using the central management tools that are available to us. PowerShell is one of these tools. Rather than RDPing into that server, simply use the PowerShell prompt on your local machine in order to reach out and change that setting on the remote server.

This kind of remote management becomes not only efficient but necessary, as we start dealing more with headless servers. I hope to see increased utilization of Server Core in our organizations over the next few years, and interacting with these servers is going to require a shift in your administrative mindset. By becoming familiar with accomplishing daily tasks from inside PowerShell now, you will better equip yourself for future administration of these headless machines, which are going to require you to interface with them differently than you are comfortable doing today.

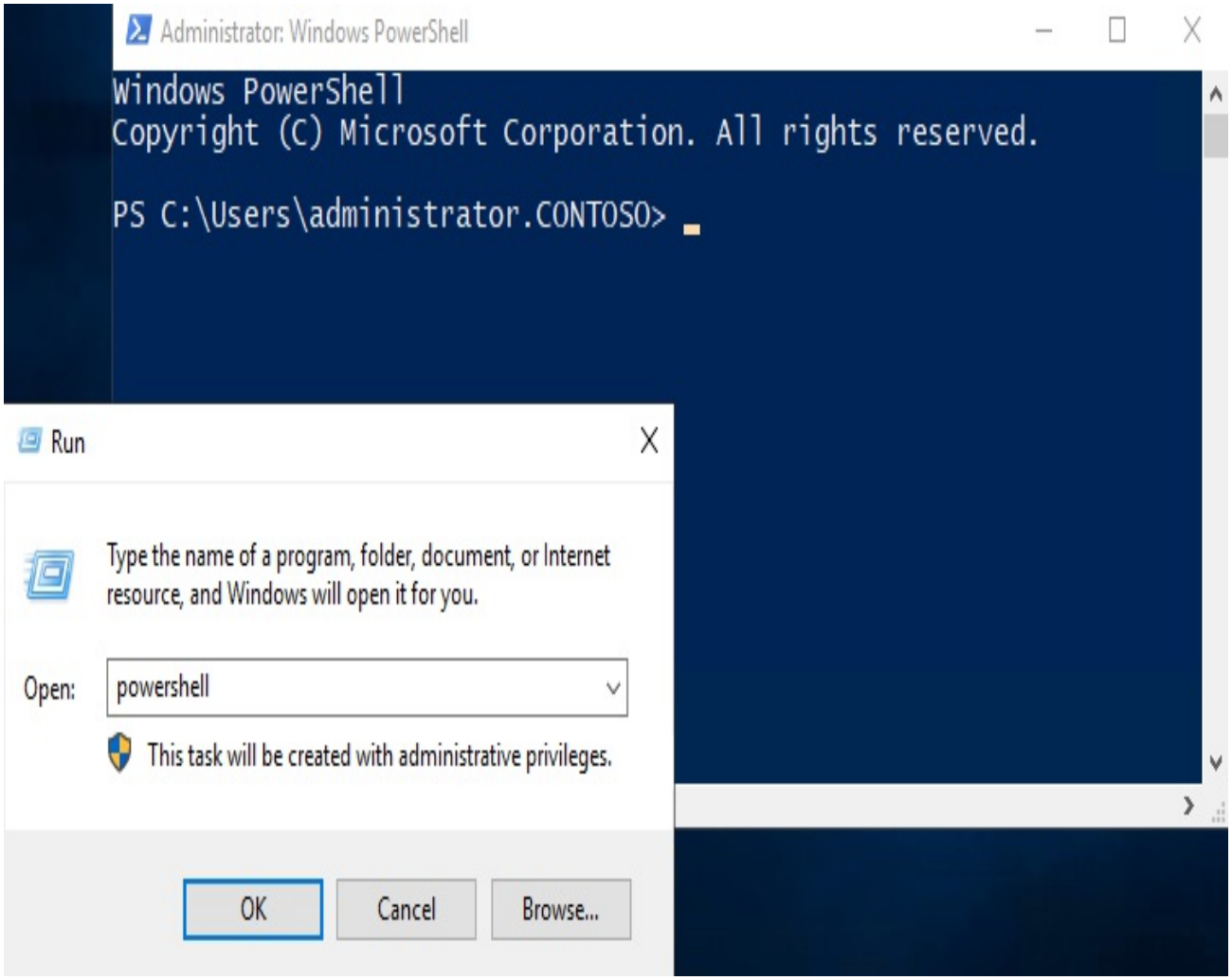
Working within PowerShell

The first step to doing real work with PowerShell is getting comfortable interfacing with the platform, and becoming familiar with the daily routines of working from this command line, rather than relying on your mouse pointer. Here, we will explore some of the most common ways that I have seen server administrators make use of PowerShell in order to enhance their daily workload.

Launching PowerShell

Pretty simple—the first thing we need to do is get PowerShell opened up to start using it. The PowerShell console is installed by default in all recent versions of Windows, so you can run it from the Start menu, pin it to the Desktop, or access it in any way that you normally open any application.

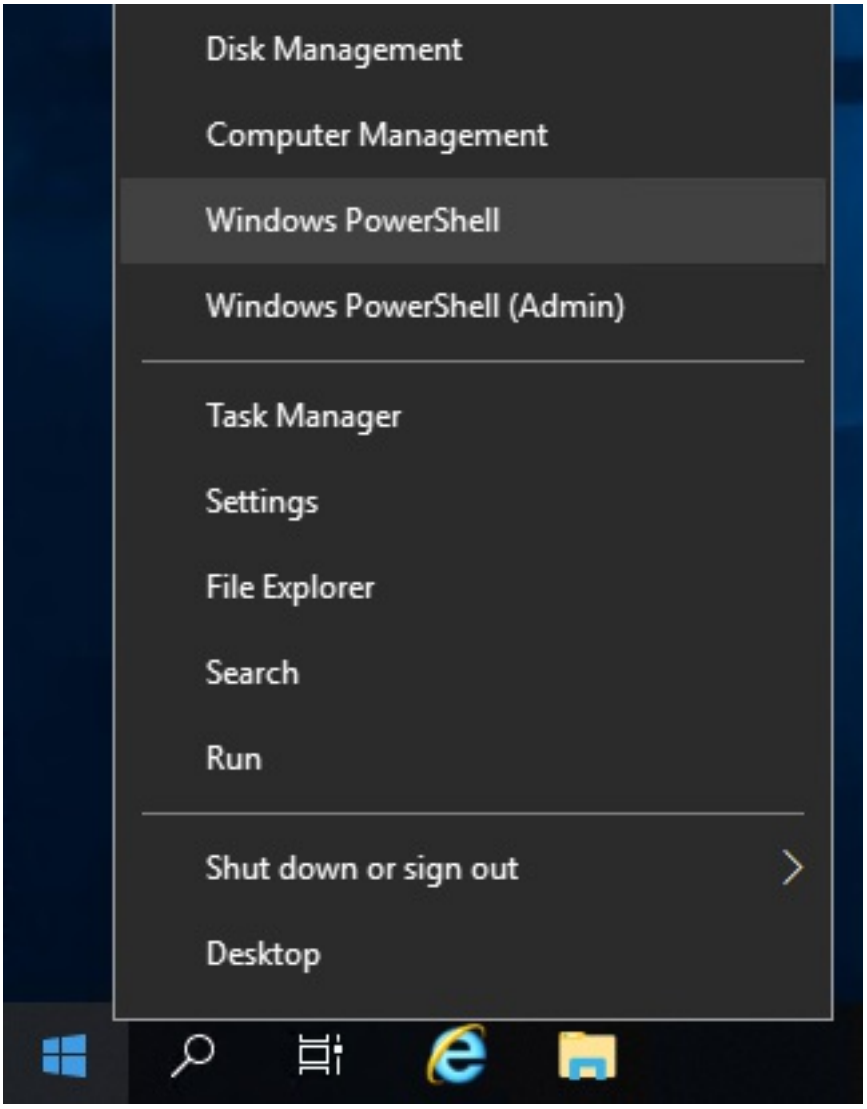
Since I tend to prefer using my keyboard for everything, the way that I normally open PowerShell is to hold down the *WinKey* and press *R* in order to open a Run prompt, type the word `powershell`, and press *Enter*:



As you can see, since I am logged into an administrative account on my server, my PowerShell prompt has been opened with elevated permissions. This is visible in the fact that the word Administrator is listed in the top toolbar of the PowerShell window. It is important to note that, just like Command Prompt, you can open a PowerShell prompt with either regular user permissions, or elevated Administrator privileges. It is generally safer to work from within a regular PowerShell session that does not have elevated rights, unless the task that you are trying to accomplish requires those extra permissions.

Another quick and easy way to open PowerShell on any newer Windows

platform is by right-clicking on the Start button and selecting it right from the quick-tasks list that is presented. As you can see in the following screenshot, I have right-clicked on the Start button of one of my new Server 2019 boxes and can choose from here to open PowerShell, or even an elevated (administrative) PowerShell prompt:

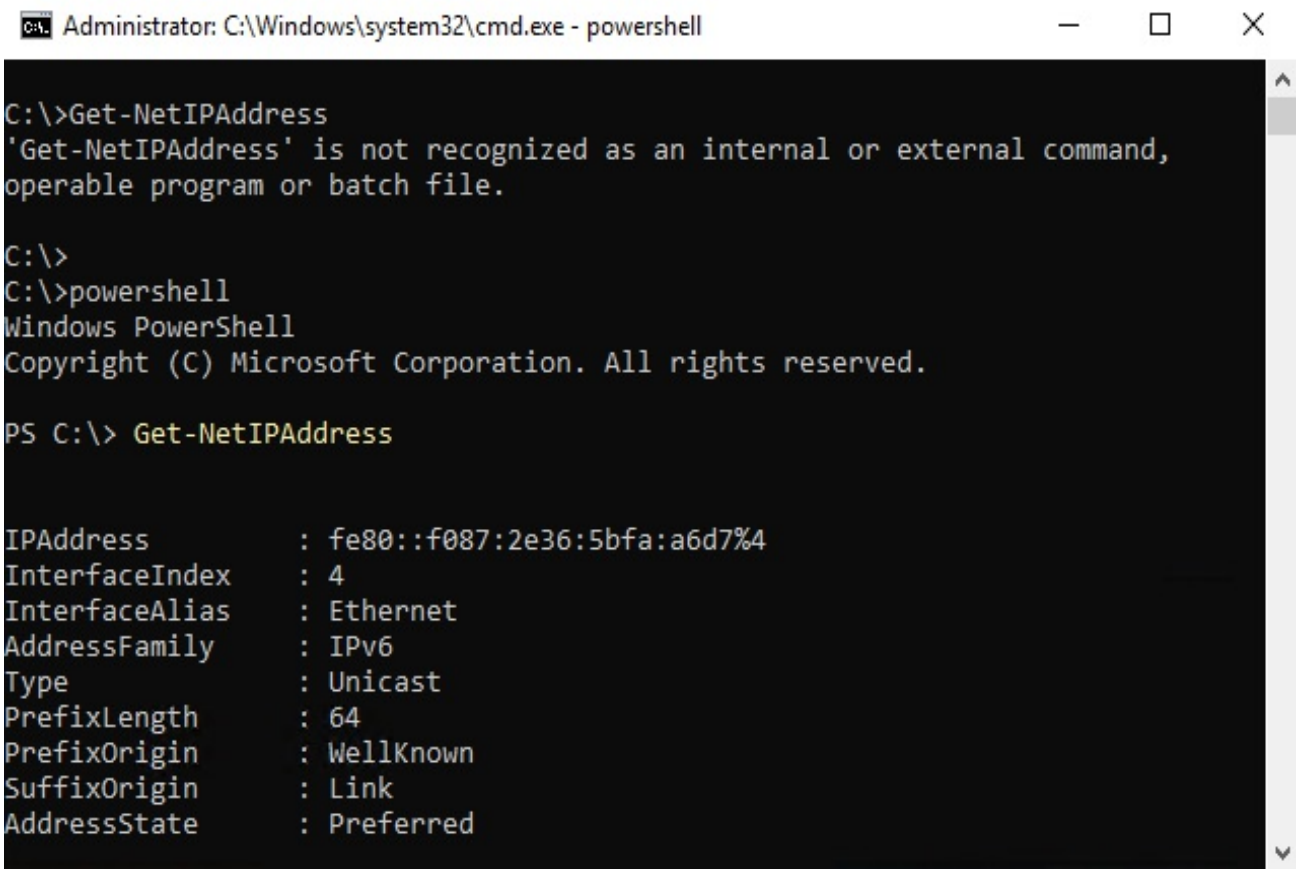


If you right-click on your Start button and do not find options for PowerShell, but rather for opening Command Prompt, do not be dismayed. This is a configurable option; you are able to show either

Command Prompt or PowerShell options in the quick admin tasks menu. If you right-click on your Taskbar and select Taskbar settings, you will find an option that is called Replace Command Prompt with Windows PowerShell in the menu when I right-click the start button or press Windows key+X. Toggling this option will swing your quick admin menu back and forth between the two command-line interfaces.

You also have the option of entering into a PowerShell prompt from inside an existing Command Prompt window. Normally, when you are working from Command Prompt, you cannot make use of any PowerShell cmdlets. Let's go ahead and give this a shot. Open an administrative Command Prompt window, and try to type in the name of one of the cmdlets we mentioned earlier. Perhaps type `Get-NetIPAddress` to show us what IP addresses reside on this system. Whoops—that failed because Command Prompt doesn't recognize the `Get-NetIPAddress` cmdlet.

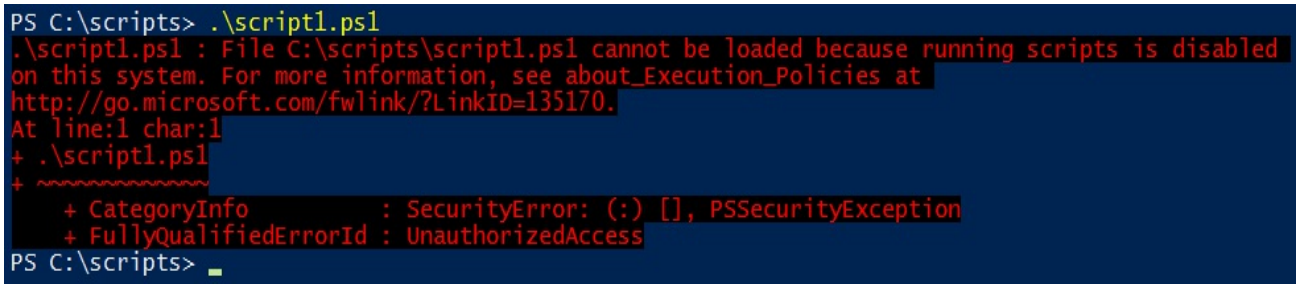
Now type `powershell` and press *Enter*. Instead of opening a separate PowerShell window, your prompt changes but the application window itself remains the same. You have now entered the PowerShell shell from inside the black Command Prompt window, and you can start utilizing cmdlets as you wish. Running `Get-NetIPAddress` again now produces some information:



You can move from PowerShell mode back to regular Command Prompt mode by typing `exit`.

Default Execution Policy

When you are working with the PowerShell command-line interface directly, you can simply open up PowerShell, start typing cmdlets, and start getting work done. However, one of the big advantages of using PowerShell comes when you start playing around with creating, saving, and running scripts. If you open up PowerShell, create a script, and then try to run it, you will sometimes find that it fails with a big messy error message, such as this one:



This shouldn't happen on a fresh instance of Windows Server 2019, but could if you have any GPOs being applied to your new server or if you are using a different operating system and are trying to run some PowerShell scripts; you might find yourself stuck at one of these error messages right out of the gate. While the nature of some versions of Windows to block the running of scripts by default is a security enhancement, it can be a nuisance to work around when you are trying to get something done. Thankfully, if you do encounter this problem, the resolution is easy: you simply need to adjust the **Default Execution Policy** (**DEP**) inside PowerShell, so that it allows the execution of scripts to happen properly.

This is not a simple ON/OFF switch. There are five different levels within the DEP, and it is important to understand each one so that you can set your DEP accordingly, based on the security that you want in place on your servers. Here are descriptions of each level, in order of most to least secure.

Restricted

The Restricted policy allows commands and cmdlets to be run, but stops the running of scripts altogether.

AllSigned

This requires that any script being run needs to be signed by a trusted publisher. When set to AllSigned, even scripts that you write yourself will have to be put through that validation process and signed before they will be allowed to run.

RemoteSigned

RemoteSigned is the default policy in Windows Server 2019. For scripts that have been downloaded from the internet, it requires that these scripts are signed with a digital signature from a publisher that you trust. However, if you choose to create your own scripts, it will allow these local scripts to run without requiring that digital signature.

Unrestricted

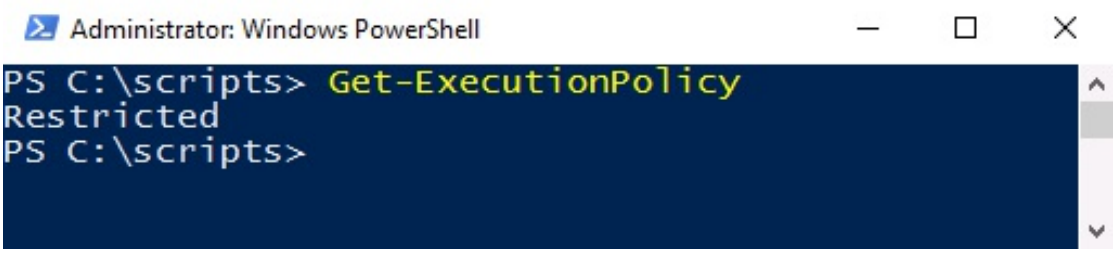
Scripts are allowed to run, signed or unsigned. You do still receive a warning prompt when running scripts that have been downloaded from the internet.

The Bypass mode

In Bypass mode, nothing is blocked and no warnings are given when you run scripts. In other words, you're on your own.

Sometimes a single execution policy doesn't meet all of your needs, depending on how you utilize PowerShell scripts. DEPs can be further enhanced by setting an **Execution Policy Scope** that allows you to set different execution policies to different aspects of the system. For example, the three scopes that you can manipulate are **Process**, **CurrentUser**, and **LocalMachine**. By default, the DEP affects LocalMachine so that any scripts running adhere to the DEP. But if you need to modify this behavior so that different DEPs are set for the CurrentUser or even an individual Process, you have the ability to do that.

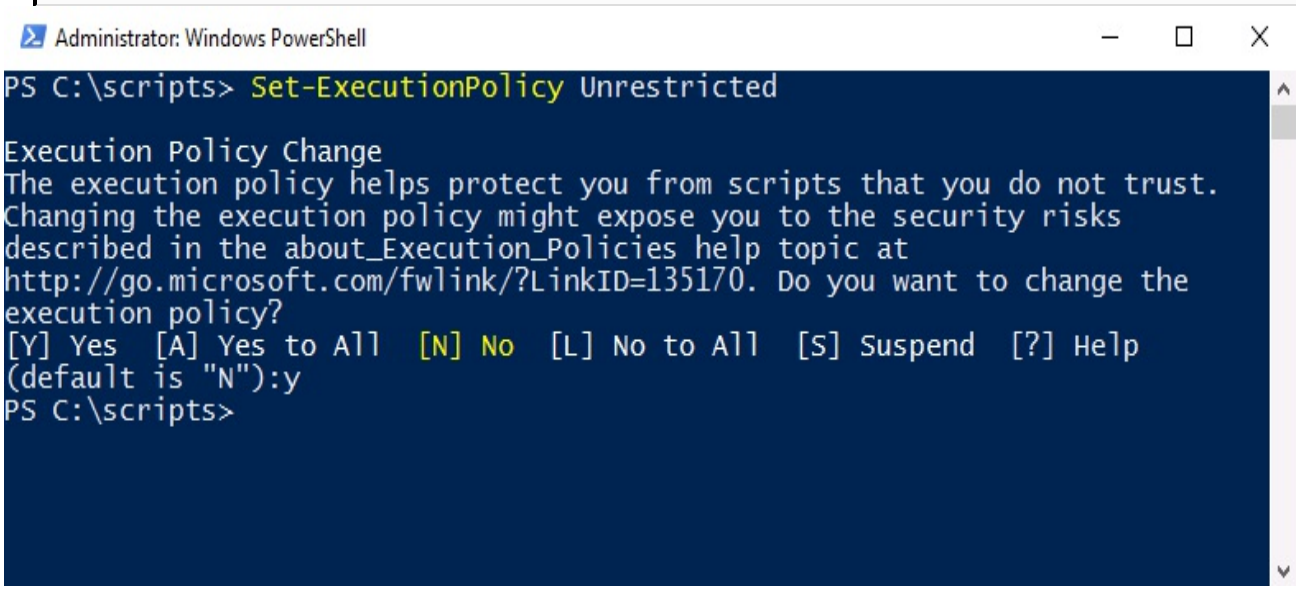
If you are unsure about the current status of your DEP or suspect that someone may have changed it, you can easily view the currently assigned execution policy with a simple cmdlet called `Get-ExecutionPolicy`. As you can see in the following image, mine is set to `Restricted`, which explains my earlier error message when I tried running a script:



Once you have decided on the level of DEP that you want on your server

or workstation, you can set it accordingly with a quick cmdlet. For example, since this is a test lab and I want scripts to be able to run, and I am not really concerned about security since I am isolated, I am going to change mine to `Unrestricted`. Here is my command for doing just that:

```
Set-ExecutionPolicy Unrestricted
```



Remember, right now we are running PowerShell on this local system (I happen to be logged into my WEB3 server), so the only execution policy I am setting is the local one for my WEB3 system. If I wanted to change this setting globally, or for a group of machines at the same time, I could utilize Group Policy for that change. The location inside Group Policy for configuring PowerShell script execution policy is Computer Configuration | Policies | Administrative Templates | Windows Components | Windows PowerShell | Turn on script execution.

Using the Tab key

Before we get started navigating inside PowerShell, there is one important thing I want to point out: get used to pressing the *Tab* key when you are inside the PowerShell prompt! If you type the first few letters of any command or cmdlet, and then press *Tab*, the remainder of the cmdlet name will be automatically populated on the screen.

If we type `get-co` and then press *Tab*, the prompt automatically populates the full `Get-Command` cmdlet. Since there are multiple cmdlets that started with `get-co`, if you press *Tab* numerous times you can see that it cycles through all of the available cmdlets that start with those letters.

Tab also works with file and folder names. For example, I downloaded a hotfix that needs to be installed onto a server. I want to launch this hotfix using the PowerShell prompt that I already have open, but I don't want to spend an entire minute or more trying to type out the huge filename of this hotfix. I have already navigated to the folder where my hotfix resides, and now if I simply type the first few letters of the filename and press the *Tab* key, PowerShell will populate the remainder of the filename. From there, all we need to do is press *Enter* to launch that installer:



Useful cmdlets for daily tasks

When I started incorporating PowerShell into my daily workflow, I found it useful to keep a list of commonly-used commands and cmdlets handy. Until you get to the point where they become memorized and second nature, if you don't have a quick and easy way to recall those commands, chances are you aren't going to use them and will revert to the old method of configuring your servers. Here is a list of some of the items I use regularly when I'm building servers. Some are traditional commands that would also work from a Command Prompt, and some are cmdlets, but they are all useful when working inside a PowerShell window:

- `Get-Command`: This is useful for finding additional commands or cmdlets that you may want to run or research.
- `Get-Command -Name *example*` : Enhances the usefulness of `Get-Command` by adding the `-Name` switch to the end of it, so that you can filter results to whatever types of cmdlets you are searching for.
- `GC`: This is simply a short alias for `Get-Command`. I only wanted to point this one out because some of the PowerShell cmdlets have aliases, such as `gcm`, that allow you to launch these commonly-used cmdlets with fewer keystrokes.
- `Get-Alias`: Since we just mentioned the GCM alias for `Get-Command`, you may be wondering what other aliases are available inside

PowerShell. To see a complete list, simply plug in the `Get-Alias` cmdlet.

- `Rename-Computer`: This allows you to set a new hostname for the server.
- `Add-Computer`: Use the `Add-Computer` cmdlet to join servers or computers to a domain.
- `Hostname`: This displays the name of the system you are currently working on. I use hostname all the time to make sure that I really am working on the server that I think I am. Have you ever rebooted the wrong server? I have. By running a quick `hostname` command, you can get peace of mind that the function you are about to perform is really happening on the correct system.
- `$env:computername`: This presents you with the hostname of the system you are working on, but I'm calling it out to show that PowerShell can easily tap into your environment variables in order to pull out information. The simpler `hostname` command is useful when you are logged into a local system and are simply trying to verify its name, but the ability to pull information from a variable, such as `$env:computername`, will be much more useful when creating scripts or trying to perform a function against a remote system.

- `Logoff`: The name is self-explanatory, `Logoff` just logs you out of the system. Rather than trying to find the Sign out function by

clicking around inside your server's Start menu, you can throw a quick `Logoff` command into either a Command Prompt or a PowerShell window, and it will immediately log you off that session. I use this one all the time when closing out RDP connections.

Both `shutdown` or `Restart-Computer` are useful for shutting down or restarting a server. On my own computer, these commands are most commonly preceded by the `hostname` command. When rebooting a server, you want to take special care that you restart the correct machine, so I find it best to open a PowerShell prompt, do a quick `hostname` check, and then run a restart command from that same prompt. This ensures that I am restarting the server that was returned in the hostname output.

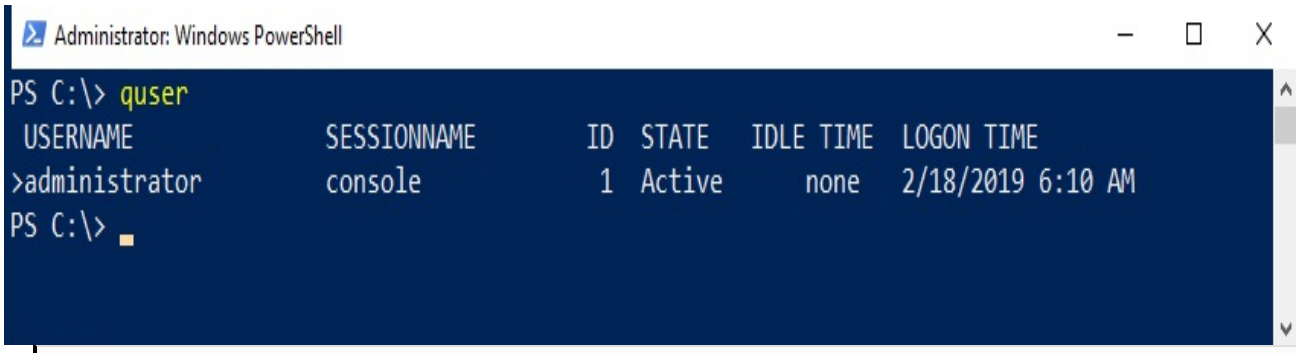
```
shutdown /r /t 0
```

If you run a simple `shutdown` command, the system will shut down in one minute. I'm not sure why this is the default, as I have never found any IT administrator who actually wanted to wait that extra minute before shutting down their system. Instead, it is more efficient to set a time limit before that shutdown commences. In the preceding command, I have told the `shutdown` command that I want to restart instead of shutting down, that is what `/r` does; I have also told it to wait zero seconds before performing this restart. This way, it happens immediately; I don't have to wait for that default 60 seconds.

`Query user` or `quser`

Often most useful in RDS environments, the `quser` command will display all of the users that are currently logged into a server, including statistics about whether they are logged in locally or remotely, and how

long their session has been active:




```
quser /computer:WEB1
```

Using `quser` in combination with the `/computer` switch allows you to see the currently-logged-in users on a remote system. This way, you can remain logged into a single server in your RDS farm, but check on the user sessions for all of your systems without having to log into them. You could even write a script that runs this command against each of your session host servers, and outputs the data to a file. This output could then be run on a schedule, and used as a reporting mechanism for keeping track of which users were logged into which RDS session host servers at any given time.

```
Install-WindowsFeature
```

Use PowerShell to simplify the installation of roles and features onto your servers.

```
New-NetIPAddress -InterfaceIndex 12 -IPAddress 10.10.10.40 -PrefixLength 24 -
DefaultGateway 10.10.10.1
```

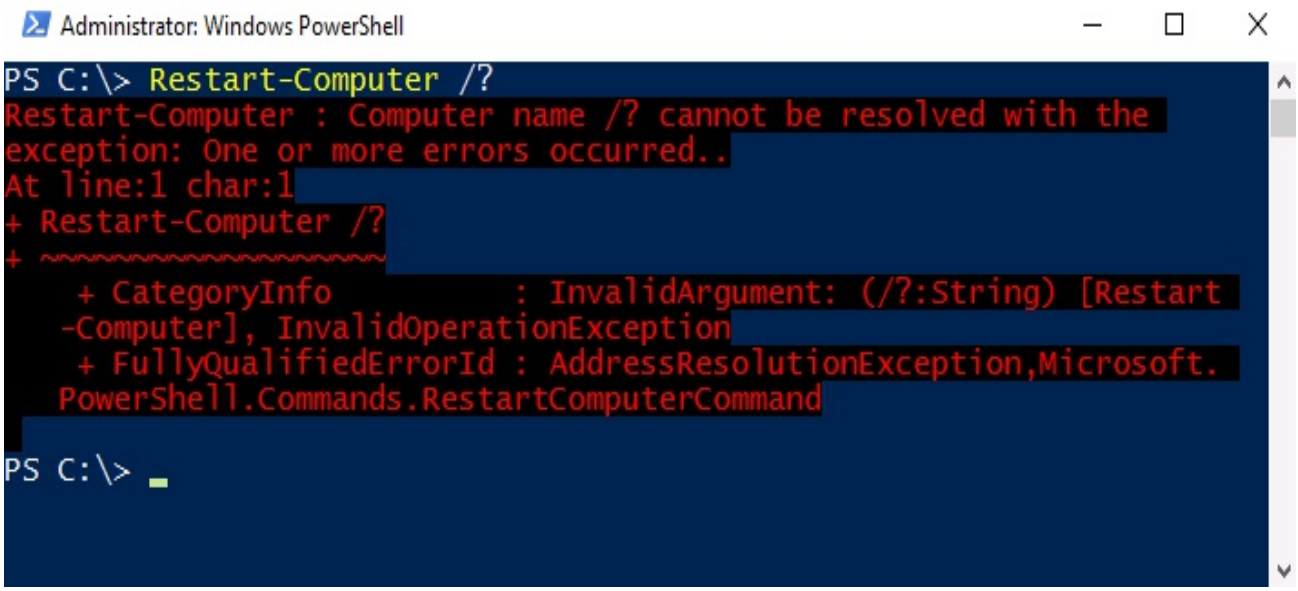
Use `New-NetIPAddress` to assign IP addresses to your NICs. Keep in mind that the information in the preceding cmdlet is clearly example data, and needs to be replaced with your own information.

```
Set-DnsClientServerAddress -InterfaceIndex 12 -ServerAddresses 10.10.10.2,10.10.10.3
```

Often used in combination with `New-NetIPAddress`, use this to set the DNS server addresses in your NIC properties.

Using Get-Help

How many hundreds of times have you used the `/?` switch in Command Prompt to pull some extra information about a command that you want to run? The extra information provided by this help function can sometimes mean the difference between a command being useful, or completely useless. PowerShell cmdlets have a similar function, but you cannot simply `/?` at the end of a PowerShell cmdlet because a space following a cmdlet in PowerShell indicates that you are about to specify a parameter to be used with that cmdlet. For example, if we try to use `/?` with the `Restart-Computer` cmdlet in order to find more information about how to use `Restart-Computer`, it will fail to recognize the question mark as a valid parameter, and our output is as follows:

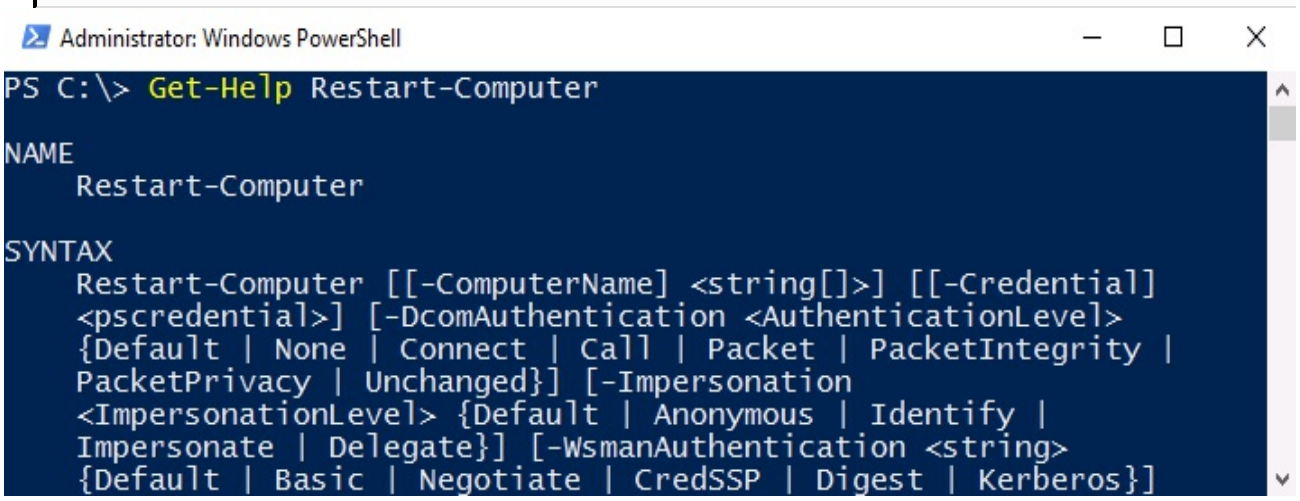


Instead, there is an even more powerful help function inside PowerShell. `Get-Help` is a cmdlet itself, and like any cmdlet, we need to use

information following the cmdlet in order to specify and pull the information that we are looking for. So instead of using `Get-Help` at the end of a command, like we used to do with the question mark, we use it as its own entity.

Running `Get-Help` by itself only gives us more information about the `Get-Help` command, which may be useful to look over, but right now we are more interested in finding out how we can use `Get-Help` to give us additional information for a cmdlet we want to run, such as the `Restart-Computer` function. What we need to do is use `Get-Help` as a cmdlet, and then specify the other cmdlet as a parameter to pass to `Get-Help`, by placing a space between them:

```
Get-Help Restart-Computer
```



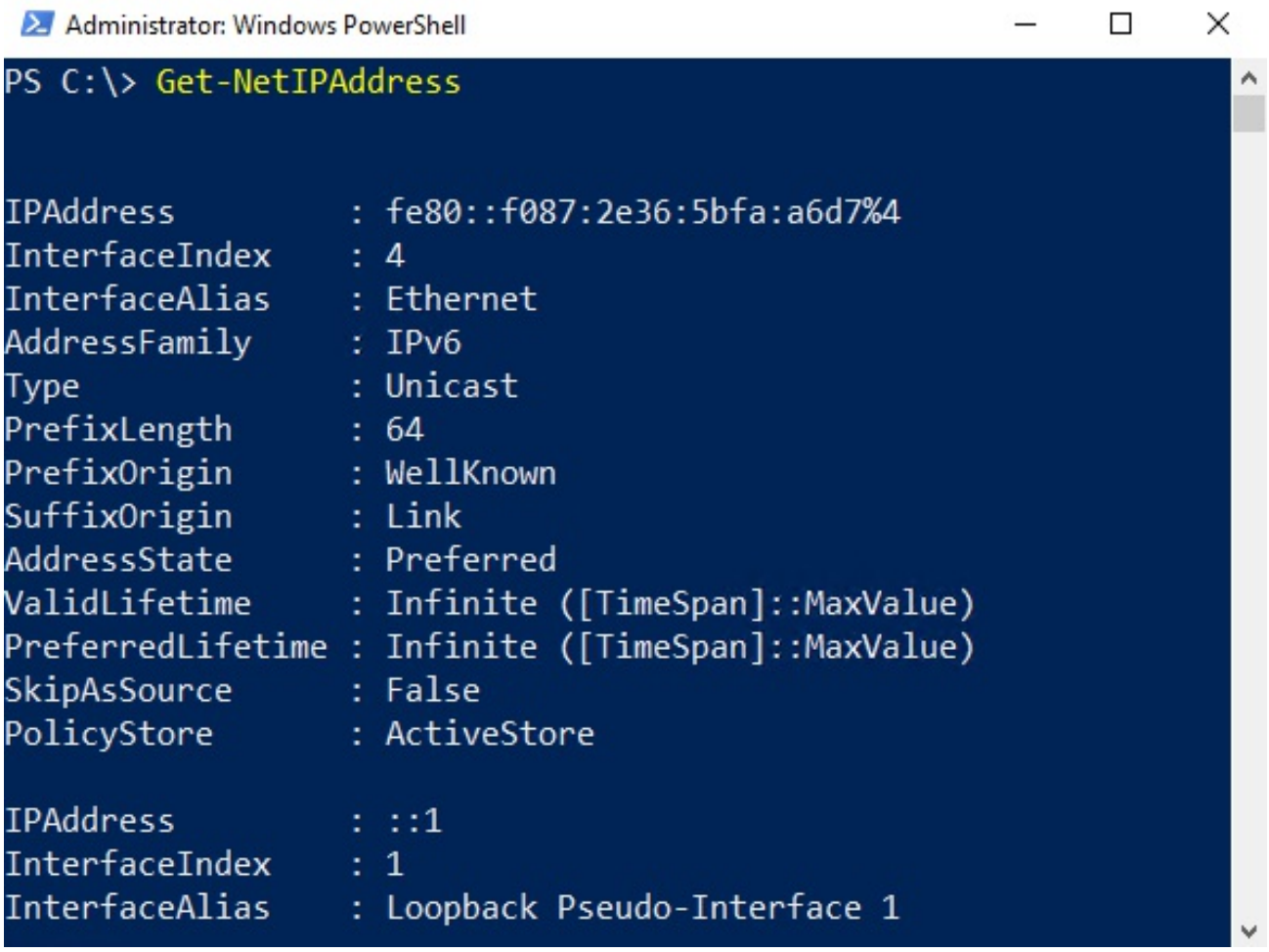
The information provided by `Get-Help` is very comprehensive; in some cases, it has all of the same information that you can find on TechNet. Make sure to start utilizing `Get-Help` to further your knowledge of any cmdlet in PowerShell!

Formatting the output

When searching for information in PowerShell, I often encounter the case where so much information is provided to me that it's difficult to sort through. Are you trying to find useful cmdlets from `Get-Command`, or maybe track down a particular alias with `Get-Alias`? The output from these cmdlets can be staggeringly long. While we have discussed some parameters you can use to whittle down this output, such as specifying particular `-Name` parameters, there are a couple of formatting parameters that can also be appended to cmdlets, in order to modify the data output.

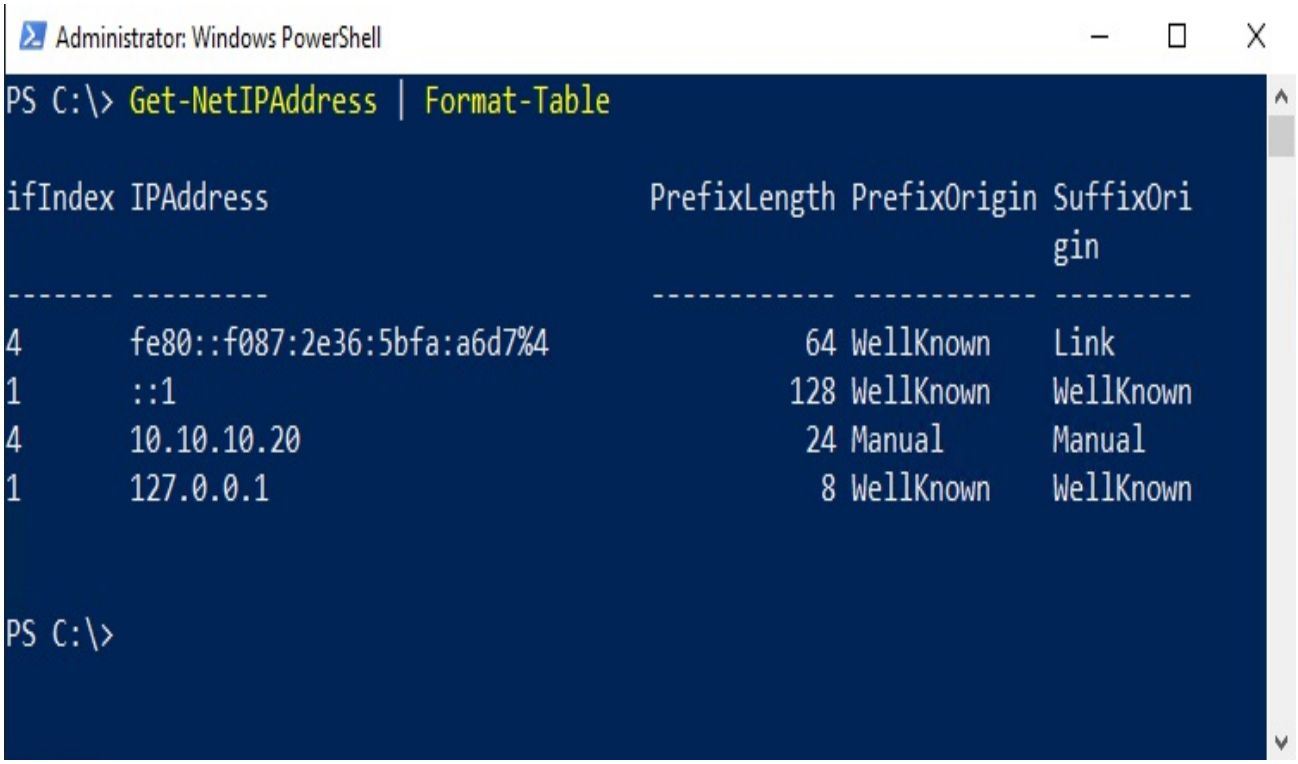
Format-Table

The purpose of `Format-Table` is pretty simple: it takes the data output from a command and puts it into a table format. This generally makes the information much easier to read and work with. Let's look at an example. We have used `Get-NetIPAddress` a couple of times, but, let's be honest, its output is a little messy. Running the cmdlet by itself on my virtual server, which only has a single NIC assigned to it, results in four pages of data inside my PowerShell window, with all kinds of informational fields that are either empty or not important to finding the IP addresses assigned to my server:



If we simply add `Format-Table` to the end of my `Get-NetIPAddress` cmdlet, the generated data is much easier on the eyes, while still giving me the important information that I am really looking for: the IP addresses being used on the system:

```
Get-NetIPAddress | Format-Table
```



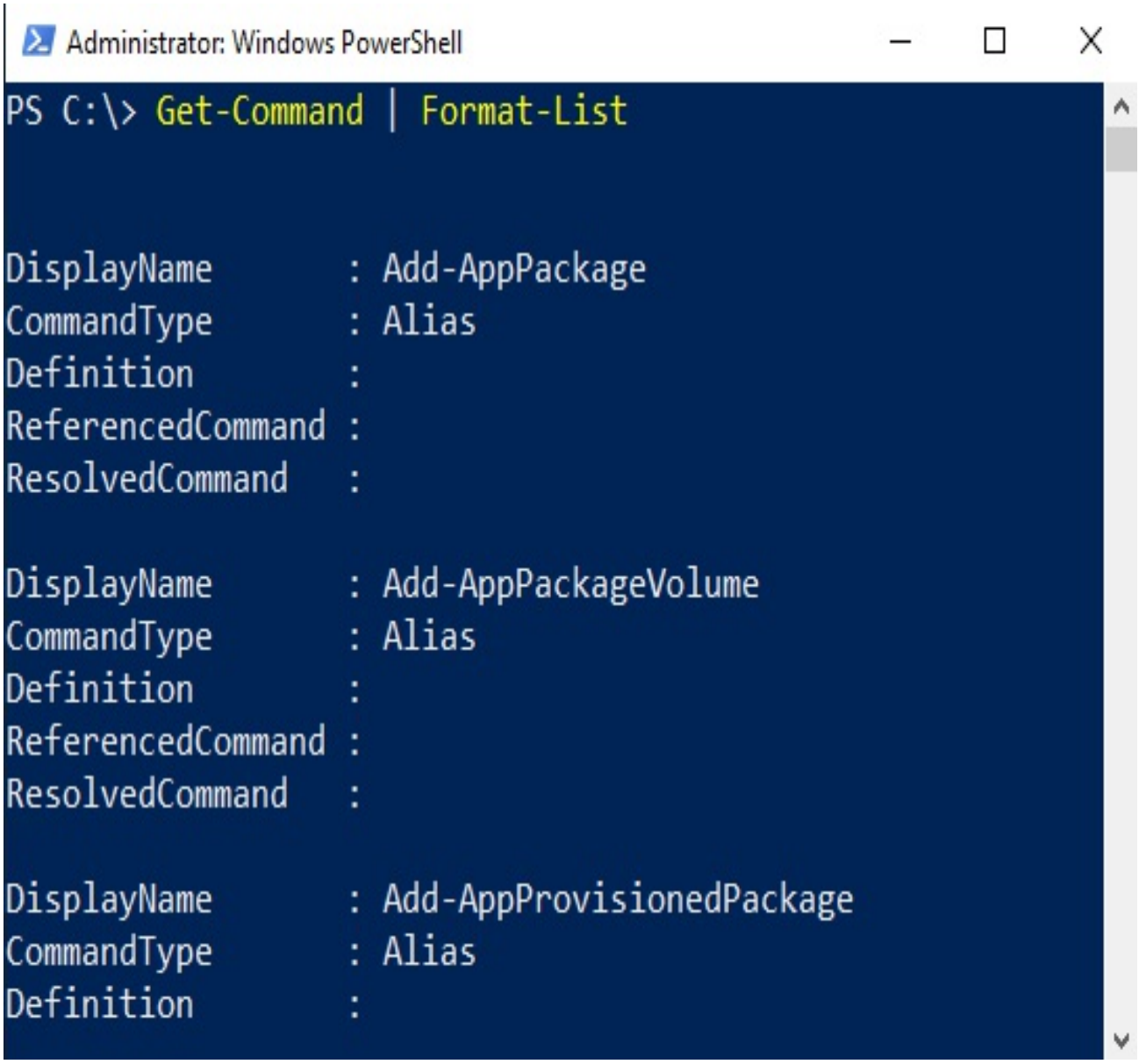
Some of you may be familiar with a cmdlet called `Select-Object`, which can perform the same functions as `Format-Table`. While `Select-Object` seems to be the more widely-known cmdlet, in my experience, it is actually less powerful than `Format-Table`, and so I suggest you spend some time playing around with the one we have discussed here.

Format-List

Similar to the way that `Format-Table` works, you can utilize `Format-List` to display command output as a list of properties. Let's give it a quick try. We already know that `Get-Command` gives us the available cmdlets within PowerShell, and by default, it gives them to us in a table format.

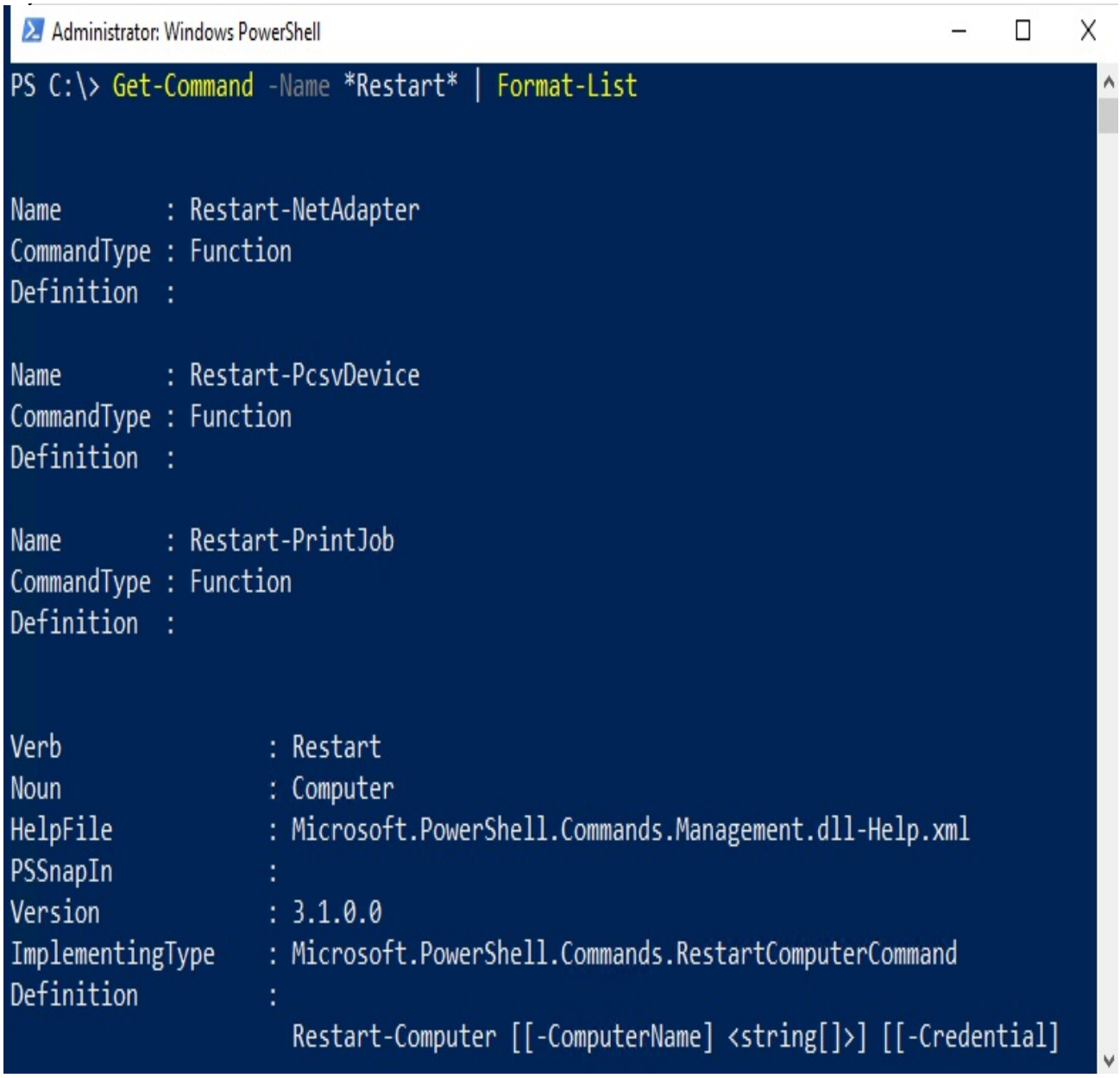
If we wanted to view that output in a list instead, with more information being provided about each cmdlet, we could tell `Get-Command` to output its data in list format, with the following command:

```
Get-Command | Format-List
```



This results in a tremendously long output of information, so long in fact that my PowerShell window had a problem displaying it all. Maybe we need to whittle that info down a little bit by narrowing our focus. Let's search for all of the cmdlets that include the word `Restart` while displaying them in list format:

```
Get-Command -Name *Restart* | Format-List
```


Administrator: Windows PowerShell
PS C:\> Get-Command -Name *Restart* | Format-List
Name : Restart-NetAdapter
CommandType : Function
Definition :
Name : Restart-PcsvDevice
CommandType : Function
Definition :
Name : Restart-PrintJob
CommandType : Function
Definition :
Verb : Restart
Noun : Computer
HelpFile : Microsoft.PowerShell.Commands.Management.dll-Help.xml
PSSnapIn :
Version : 3.1.0.0
ImplementingType : Microsoft.PowerShell.Commands.RestartComputerCommand
Definition :
Restart-Computer [[-ComputerName] <string[]>] [[-Credential]

PowerShell Integrated Scripting Environment

Most server administrators are familiar with the concept of creating batch files for use in the Command Prompt world. Have a series of commands that you want to run in sequence? Need to run this sequence of commands multiple times across different servers or over and over again in the future? Throwing multiple commands inside a text document and then saving it with the `.BAT` file extension will result in a batch file that can be run on any Windows computer, issuing those commands in sequence, which saves you the time and effort of having to plunk out these commands over and over inside the command-line interface.

Scripting in PowerShell is the same idea, but is much more powerful. Commands in Command Prompt are useful, but limited, while PowerShell cmdlets have the ability to manipulate anything within the operating system. With PowerShell, we also have the ability to reference items from inside environment variables or the registry; we can easily issue commands to remote systems, and we can even utilize variables inside a PowerShell script, just like you would do with any full programming language.

Let's explore a couple of different ways that can be used to start creating your first PowerShell scripts.

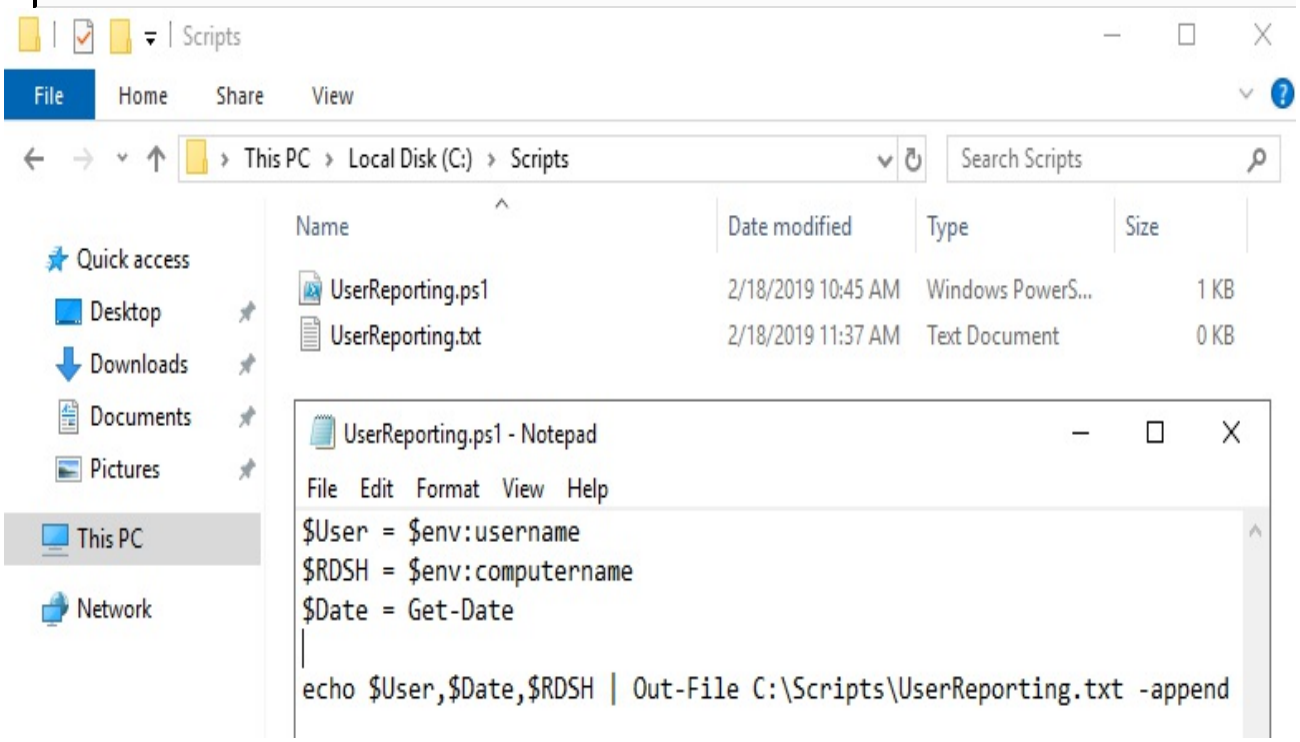
PS1 files

Creating a simple `.PS1` file (a PowerShell script file) is almost exactly the same idea as creating a `.BAT` file. All you have to do is open up a text document using your favorite editor, throw in a series of commands or cmdlets, and then save the file as `FILENAME.PS1`. As long as your PowerShell environment allows the running of scripts – see earlier in the chapter about the DEP – you now have the ability to double-click on that `.PS1` file, or launch it from any PowerShell prompt, to run the series of cmdlets inside that script. Let's give it a try and prove that we can get a simple script up and operational.

Since you are only going to create scripts that serve a purpose, let's think of a real-world example. I work with terminal servers quite a bit – pardon me, RDS servers – and a common request from customers is a log of what users logged in to which servers. A simple way to gather this information is to create a logon script that records information about the user session to a file as they are logging in. To do this, I need to create a script that I can configure to run during the logon process. To make the script a little bit more interesting and flexible down the road, I am going to utilize some variables for my username, the current date and time, and record the name of the RDS server being logged in to. That way, I can look at the collective set of logs down the road, and easily determine which users were on which servers. I am going to use Notepad to create this script. I have opened up a new instance of Notepad, entered in the following commands, and am now saving this as `C:\Scripts\UserReporting.ps1`:

```
$User = $env:username $RDSH = $env:computername $Date = Get-Date echo
```

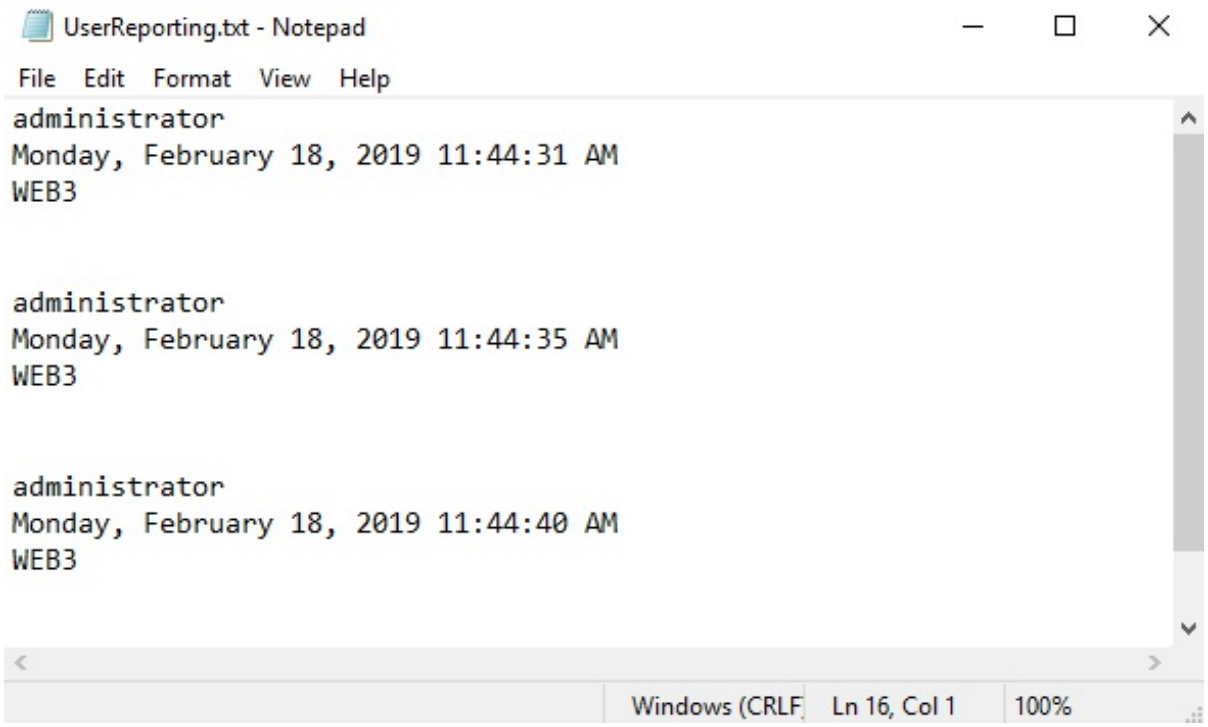
```
$User,$Date,$RDSH | Out-File C:\Scripts\UserReporting.txt -append
```



You can probably already tell what this script is doing, but let's walk through it anyway. First, we are defining three variables. I am telling the script that `$User` needs to equal, no matter what the system's username environment variable displays. `$RDSH` is going to be the name of the server where the user is logging in, also pulled by accessing the server's environment variables. The third variable defined is `$Date`, which simply pulls the current system date by calling a PowerShell cmdlet named `Get-Date`.

After pulling all of the information into the PowerShell variables, I am then outputting these three items into a text file that is sitting on my server's hard drive.

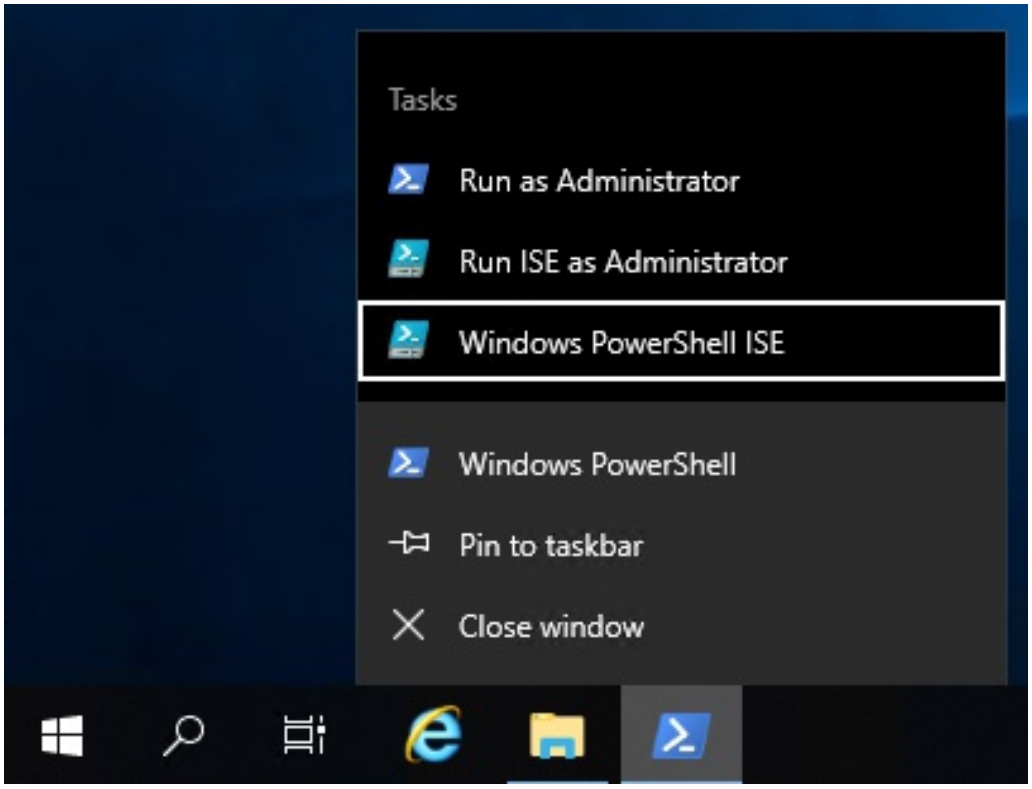
If I run this script a few times, I can open up my `UserReporting.txt` file and see that every time the script is run, it successfully logs my specified variables into this report file:


UserReporting.txt - Notepad
File Edit Format View Help
administrator
Monday, February 18, 2019 11:44:31 AM
WEB3
administrator
Monday, February 18, 2019 11:44:35 AM
WEB3
administrator
Monday, February 18, 2019 11:44:40 AM
WEB3
Windows (CRLF
Ln 16, Col 1
100%

PowerShell Integrated Scripting Environment

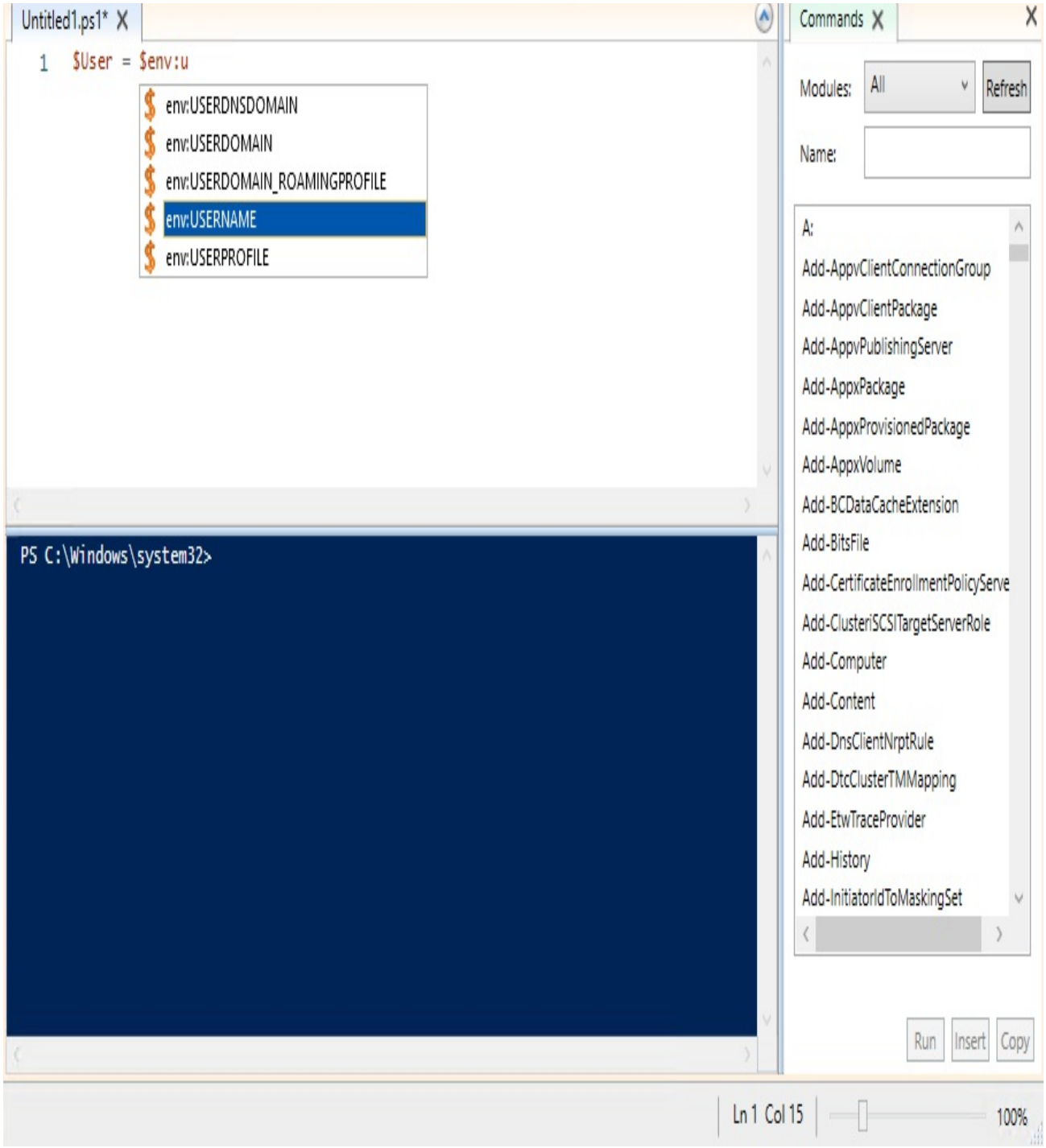
If I'm being honest, putting together that simple script we just ran took some trial and error. I didn't have a copy of it readily available to work from, and I needed to test a couple of the lines individually in PowerShell before I was confident they would work in my script. I also first tried to pull the username without using the environment variable, and it didn't work. Why did I have so much trouble putting together just a few simple lines of code? Because as I type those lines in Notepad, I have absolutely no idea whether they are going to work when I save and attempt to run that script. All of the text is just black with a white background, and I am fully trusting my own knowledge and scripting abilities in order to put together something that actually works.

Thankfully, we have access to the PowerShell **Integrated Scripting Environment** (**ISE**). This is a program that is installed by default in Windows Server 2019; it is a scripting shell that allows you to write PowerShell scripts, and provides help along the way. Let's go ahead and open it up. If you have any PS1 PowerShell script files, you can simply right-click on one of them and choose Edit. Otherwise, by right-clicking on the PowerShell application icon (from the Taskbar, for example), you will find an option to launch Windows PowerShell ISE right from that menu:



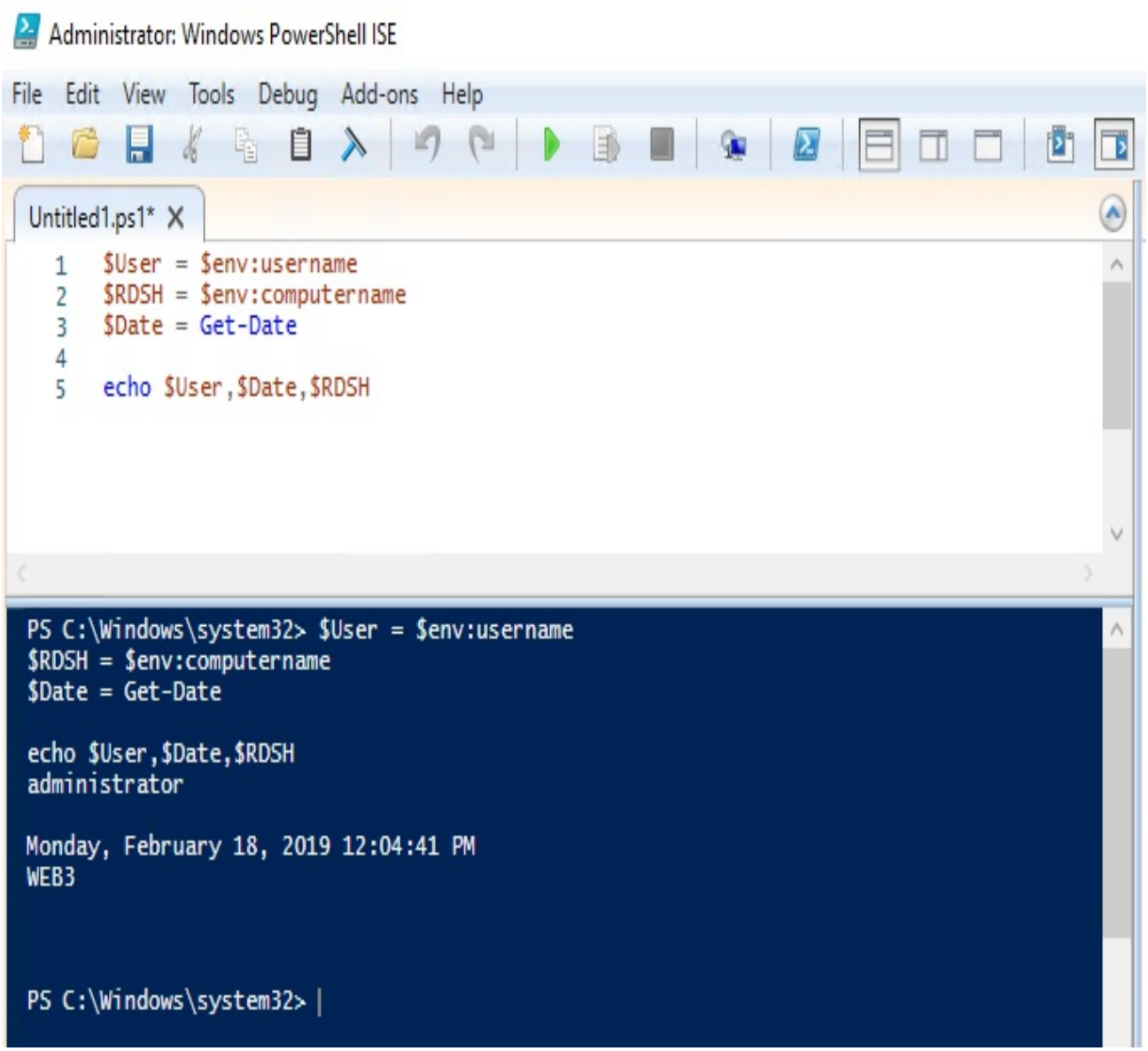
Now, if we start typing in the same script information that I used in Notepad a few minutes ago, you can see that even as we type, we get popups and prompts that help us decide which cmdlets or variables we want to utilize. Similar to the way that our autocomplete keyboards on our smart phones work, ISE will give suggestions about what you are starting to type, so that you don't necessarily have to remember what the cmdlets or parameters are called; you can take an educated guess on what letter it starts with and then choose one from the list that is presented. There is also a list off to the right of all the commands available, and it is searchable! That is a great feature that really helps to get these scripts rolling:





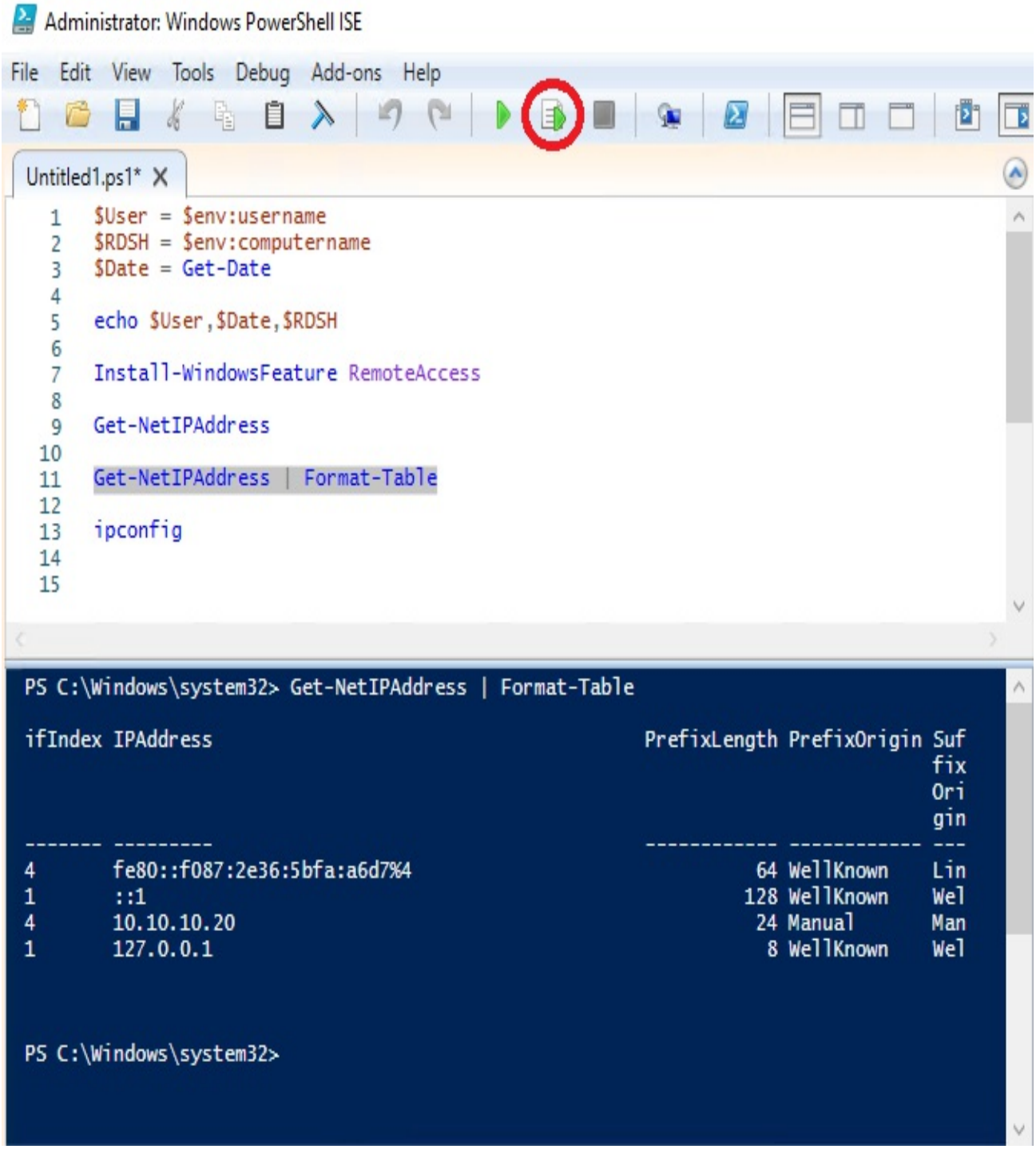
Also useful is the blue PowerShell mini screen that consumes the bottom half of the development window inside ISE. Basically, when you type in some commands, ISE helps to make sure they are all going to

work by color-coding the cmdlets and parameters for easy identification, and then you can click on the green arrow button in the taskbar that is labeled Run Script (*F5*). Even if you haven't saved your script anywhere yet, ISE launches through your commands and presents the output in the following PowerShell prompt window. This allows you to test your script, or test changes that you are making to an existing script, without having to save the file and then launch it separately from a traditional PowerShell window:



Even better is that you can highlight particular sections of your script, and choose to run only isolated pieces of the code. This allows you to test certain sections of a script, or do something creative, such as keep one big. The PS1 script file is full of common PowerShell commands that you might use on a daily basis, and when you have the need to run just one of them, you can simply highlight the text that you want to run, and click on the Run Selection (*F8*) button. By highlighting text before running the script from within ISE, only the selected cmdlet(s) will be

put into action. In the following screenshot, you can see that I have numerous cmdlets listed inside my script file, but only the one that is highlighted has been run:


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1*
1 $User = $env:username
2 $RDSH = $env:computername
3 $Date = Get-Date
4
5 echo $User,$Date,$RDSH
6
7 Install-WindowsFeature RemoteAccess
8
9 Get-NetIPAddress
10
11 Get-NetIPAddress | Format-Table
12
13 ipconfig
14
15
PS C:\Windows\system32> Get-NetIPAddress | Format-Table
ifIndex IPAddress PrefixLength PrefixOrigin Suffix Origin
4 fe80::f087:2e36:5bfa:a6d7%4 64 WellKnown Lin
1 ::1 128 WellKnown Wel
4 10.10.10.20 24 Manual Man
1 127.0.0.1 8 WellKnown Wel
PS C:\Windows\system32>

Remotely managing a server

Now that we have worked a little bit in the local instance of PowerShell, and have explored a couple of methods that can be used to start creating scripts, it is time to take a closer look at how PowerShell fits into your centralized administration needs. If you start using PowerShell for server administration, but are still RDPing into the servers and then opening PowerShell from there, you're doing it wrong. We already know that you can tap remote servers into Server Manager so that they can be managed centrally, and we also know that the tools inside Server Manager are, for the most part, just issuing a series of PowerShell cmdlets when you click on the buttons. Combine those two pieces of information, and you can surmise that PowerShell commands and cmdlets can be easily run against remote systems, including ones that you are not currently logged into.

Taking this idea and running with it, we are going to look over the criteria necessary to make this happen in our own environment. We are going to make sure that one of our servers is ready to accept remote PowerShell connections, and then use a PowerShell prompt on a different machine in order to pull information from and make changes to that remote server.

Preparing the remote server

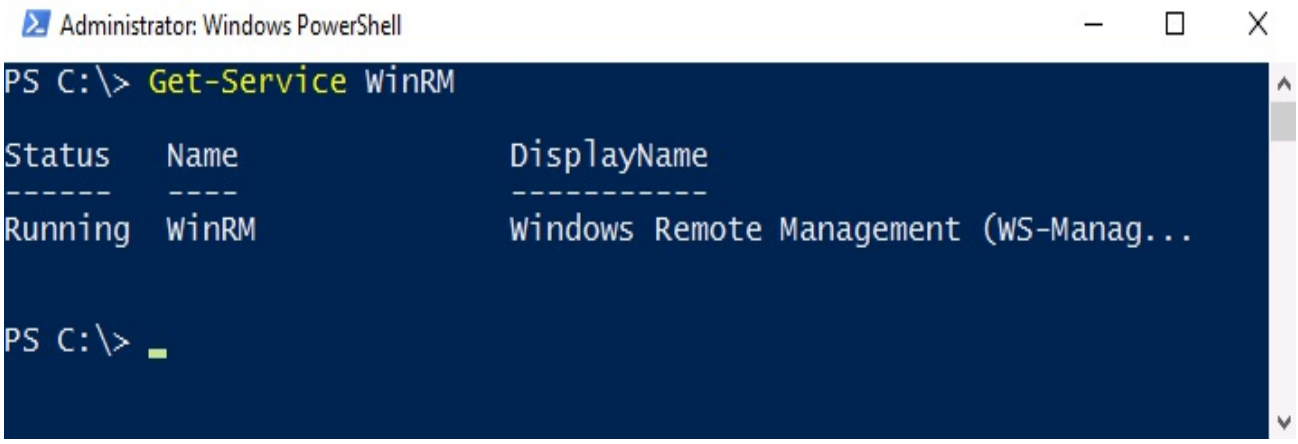
There are just a couple of items that need to be running and enabled on your remote servers in order for you to tap PowerShell into them from a different machine. If all of your servers are Windows Server 2019 (in fact, if they are all Windows Server 2012 or higher), then PowerShell remoting is enabled by default, and you may be able to skip the next couple of sections. However, if you try to use PowerShell remoting and it's not working for you, it is important that you understand how it works under the hood. This way, you can troubleshoot it and manually establish remote capabilities in the event that you run into problems, or are running some older operating systems where these steps may be necessary. It is also possible that you have pre-existing security policies that are disabling components used by the remote connection capabilities of PowerShell, so if you find your remote access to be blocked, these are the items to look into on those systems.

The WinRM service

One piece of the remote-management puzzle is the WinRM service. Simply make sure that this service is running. If you have stopped it as some sort of hardening or security benefit, you will need to reverse that change and get the service back up and running in order to use PowerShell remoting.

You can check the status of the WinRM service from `services.msc`, of course, or since we are using PowerShell in this chapter, you could check it with the following command:

```
Get-Service WinRM
```



Enable-PSRemoting

Typically, the only other thing that needs to be accomplished on your remote server is to run a single, simple cmdlet. Well, it needs to have network access, of course, or you won't be able to see it on the network at all. But other than making sure network connectivity and flow are working directly from the console of your new server, you are then ready to issue the PowerShell command that enables this server to be able to accept incoming, remote PowerShell connections:

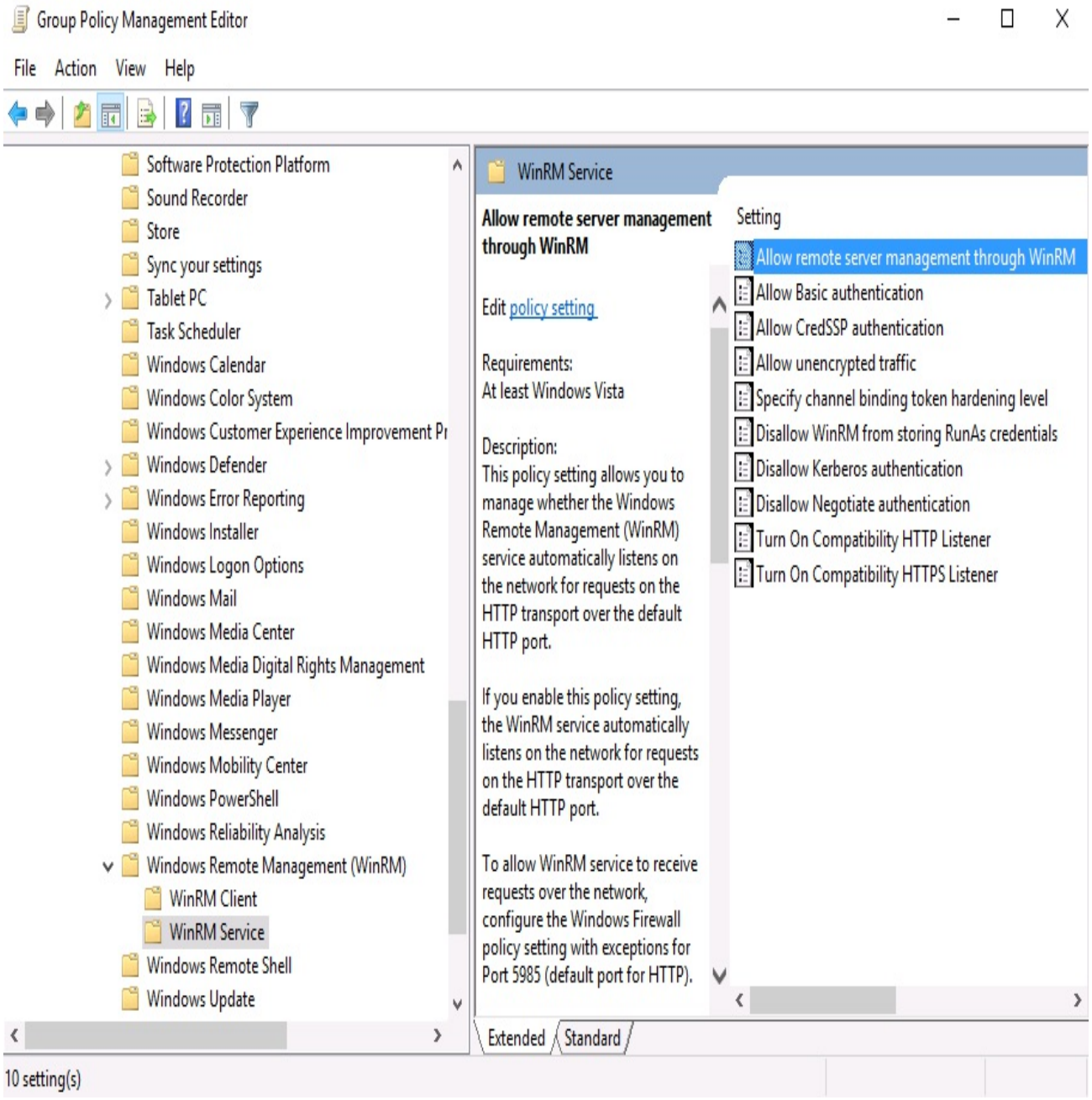
```
Enable-PSRemoting -Force
```

Using `-Force` at the end of the `Enable-PSRemoting` command causes the command to roll without asking you for confirmations. There are a few different things that `Enable-PSRemoting` is doing in the background here. First, it is attempting to start the WinRM service. Why did I already specify that you should check it manually? Because if you have it disabled as part of a lockdown strategy, you will interfere with this process. Checking WinRM before using `Enable-PSRemoting` increases your chances of success when running the `Enable-PSRemoting` cmdlet. There are two other things that this command is doing: starting the listener for remote connections and creating a firewall rule on the system to allow this traffic to pass successfully.

If you intend to use PowerShell remoting on a large scale, it is daunting to think about logging into every single server and running this command. Thankfully, you don't have to! As with most functions in the Windows world, we can use Group Policy to make this change for us

automatically. Create a new GPO, link and filter it appropriately so that it only applies to those servers that you want to be centrally managed, and then configure this setting: Computer Configuration | Policies | Administrative Templates | Windows Components | Windows Remote Management (WinRM) | WinRM Service.

Set Allow remote server management through WinRM to Enabled, as follows:


Group Policy Management Editor
File Action View Help
Software Protection Platform
Sound Recorder
Store
Sync your settings
Tablet PC
Task Scheduler
Windows Calendar
Windows Color System
Windows Customer Experience Improvement Pr
Windows Defender
Windows Error Reporting
Windows Installer
Windows Logon Options
Windows Mail
Windows Media Center
Windows Media Digital Rights Management
Windows Media Player
Windows Messenger
Windows Mobility Center
Windows PowerShell
Windows Reliability Analysis
Windows Remote Management (WinRM)
WinRM Client
WinRM Service
Windows Remote Shell
Windows Update
WinRM Service
Allow remote server management through WinRM
Edit policy setting
Requirements:
At least Windows Vista
Description:
This policy setting allows you to manage whether the Windows Remote Management (WinRM) service automatically listens on the network for requests on the HTTP transport over the default HTTP port.
If you enable this policy setting, the WinRM service automatically listens on the network for requests on the HTTP transport over the default HTTP port.
To allow WinRM service to receive requests over the network, configure the Windows Firewall policy setting with exceptions for Port 5985 (default port for HTTP).
Setting
Allow remote server management through WinRM
Allow Basic authentication
Allow CredSSP authentication
Allow unencrypted traffic
Specify channel binding token hardening level
Disallow WinRM from storing RunAs credentials
Disallow Kerberos authentication
Disallow Negotiate authentication
Turn On Compatibility HTTP Listener
Turn On Compatibility HTTPS Listener
Extended
Standard
10 setting(s)

Allowing machines from other domains or workgroups

If you are working with servers that are all part of the same corporate domain, which will most often be the case, then authentication between machines is easy to accomplish. They automatically trust each other at this level. However, on the server you are prepping to accept remote connections, if you expect those computers will be members of a different domain that is not trusted – or even members of a workgroup – then you will have to issue a command to manually trust the individual computers that are going to be connecting in. For example, if I am planning to manage all of my servers from a client computer called `Win10Client` that is not trusted by the servers, I would need to run the following command on these servers:

```
Set-Item wsman:\localhost\client\trustedhosts Win10Client
```

If you wanted to allow any machine to connect remotely, you could replace the individual computer name with a `*`, but in general, this wouldn't be a good practice, as you may be inviting trouble by allowing *any* machine to connect to your server in this way.

Connecting to the remote server

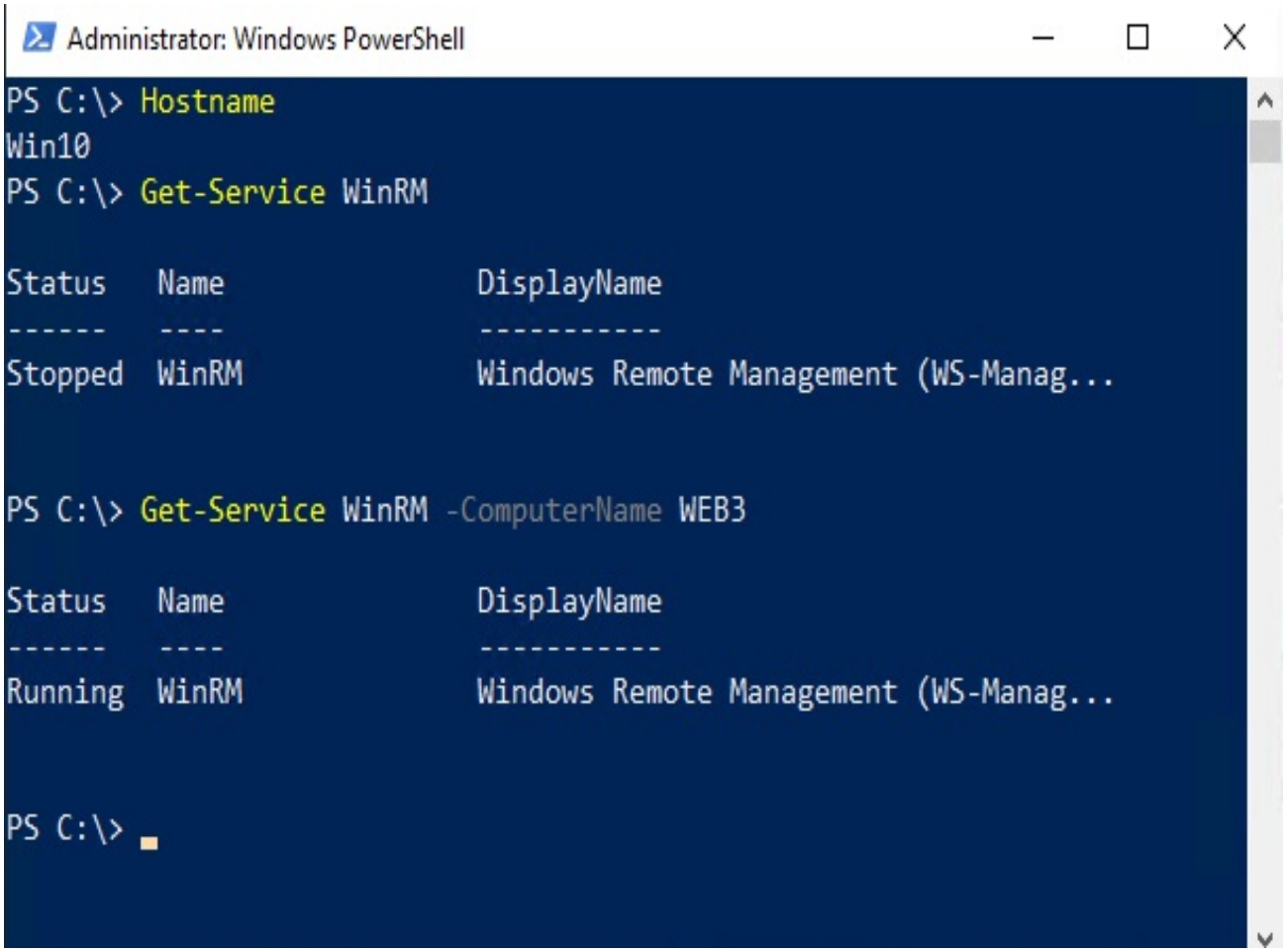
I typically see administrators utilize remote PowerShelling in two different ways. You can perform some commands against remote systems on an ad hoc basis while your PowerShell prompt is still local, or you can launch a full-blown remote PowerShell session in order to make your PowerShell prompt behave as if it is running directly on that remote system. Let's take a look at both options.

Using -ComputerName

Many of the cmdlets available in PowerShell, particularly ones that begin with `Get-`, are able to be used with the `-ComputerName` parameter. This specifies that the command you are about to run needs to execute against the remote system that you specify in the `-ComputerName` section. For our remote PowerShell examples, I will be using a PowerShell prompt on my Windows 10 client computer to access information on some of my servers in the network. I want to query the WinRM service, to make sure that it is up and running. For the sake of proving to you that I am remotely communicating with WEB3, you will see in the output that I have first queried my local WinRM service, which I happened to disable on my Win10 workstation.

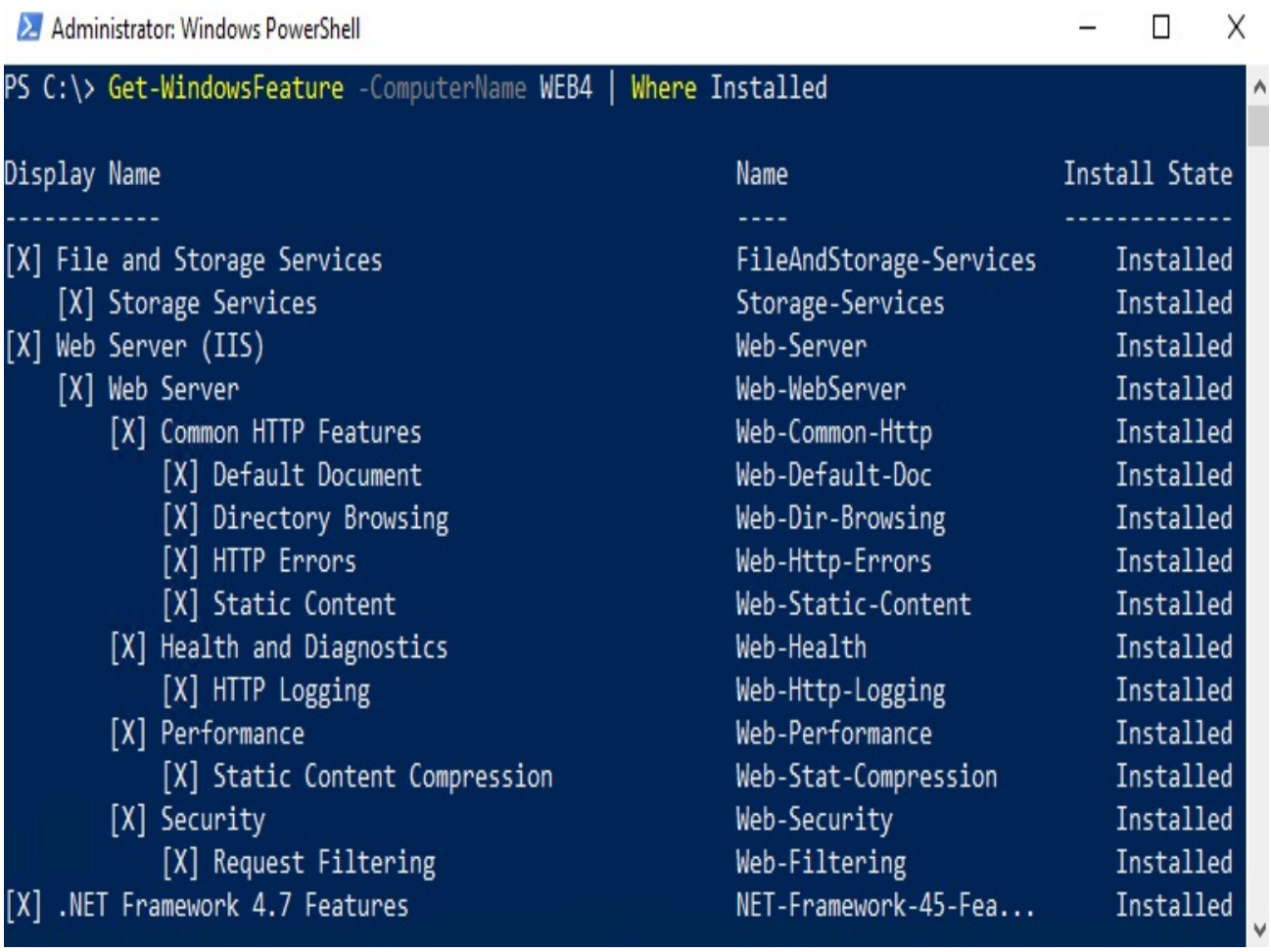
You see that my local WinRM service shows as `Stopped`, but when I issue the same command specifying to query ComputerName of WEB3, it reaches out and reports back to me that the WinRM service is indeed `Running` successfully on the WEB3 server:

```
Hostname
Get-Service WinRM
Get-Service WinRM -ComputerName WEB3
```



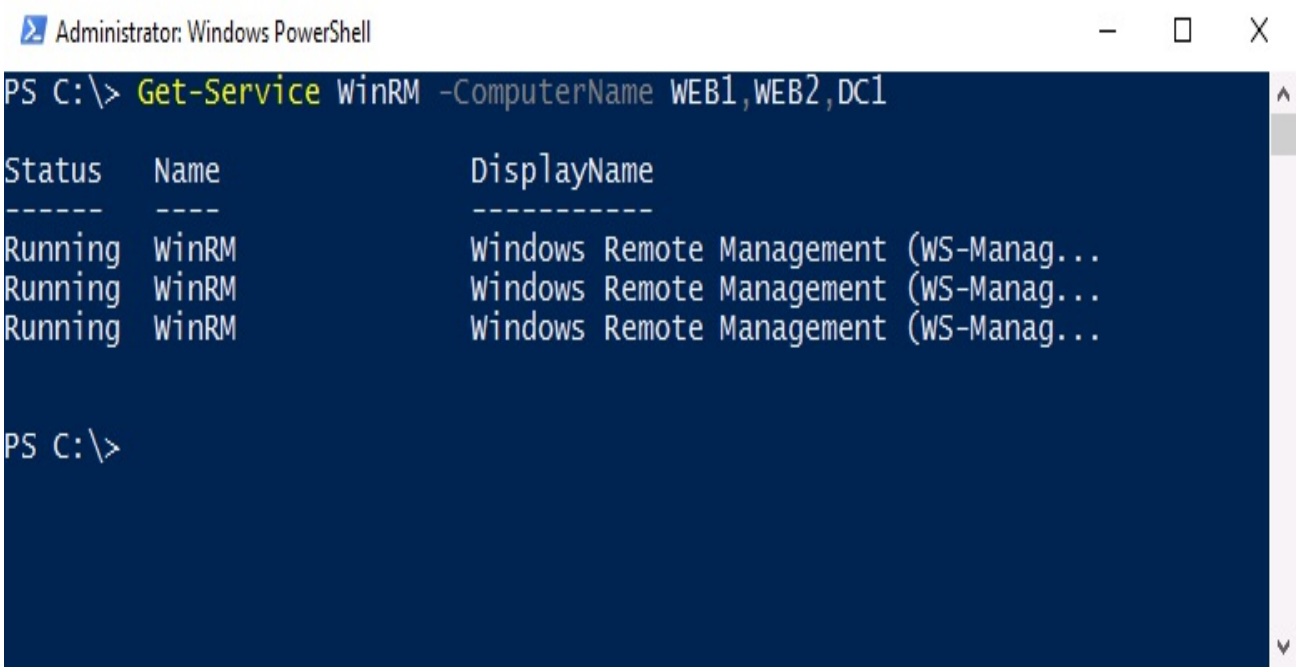
Alternatively, perhaps I want to query the new Server Core instance we set up a little while ago, and check which roles are currently installed on `WEB4`:

```
Get-WindowsFeature -ComputerName WEB4 | Where Installed
```



The `-ComputerName` parameter can even accept multiple server names at the same time. If I wanted to check the status of the WinRM service on a few of my servers, by using a single command, I could do something such as this:

```
Get-Service WinRM -ComputerName WEB1,WEB2,DC1
```


Administrator: Windows PowerShell
PS C:\> Get-Service WinRM -ComputerName WEB1,WEB2,DC1
Status Name DisplayName
Running WinRM Windows Remote Management (WS-Manag...
Running WinRM Windows Remote Management (WS-Manag...
Running WinRM Windows Remote Management (WS-Manag...
PS C:\>

Using Enter-PSSession

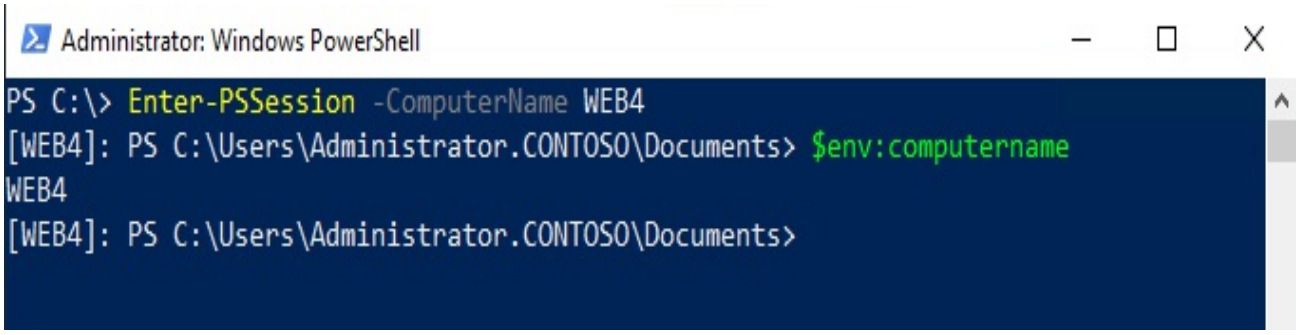
On the other hand, sometimes you have many different cmdlets that you want to run against a particular server. In this case, it makes more sense to invoke the fully-capable, fully-remote PowerShell instance to that remote server. If you open up PowerShell on your local system and utilize the `Enter-PSSession` cmdlet, your PowerShell prompt will be a full remote representation of PowerShell on that remote server. You are then able to issue commands in that prompt, and they will execute as if you were sitting at a PowerShell prompt from the console of that server. Once again, I am logged in to my Windows 10 client computer, and have opened up PowerShell. I then use the following command to remotely connect to my `WEB4` server:

```
Enter-PSSession -ComputerName WEB4
```

You will see the prompt change, indicating that I am now working in the context of the `WEB4` server.

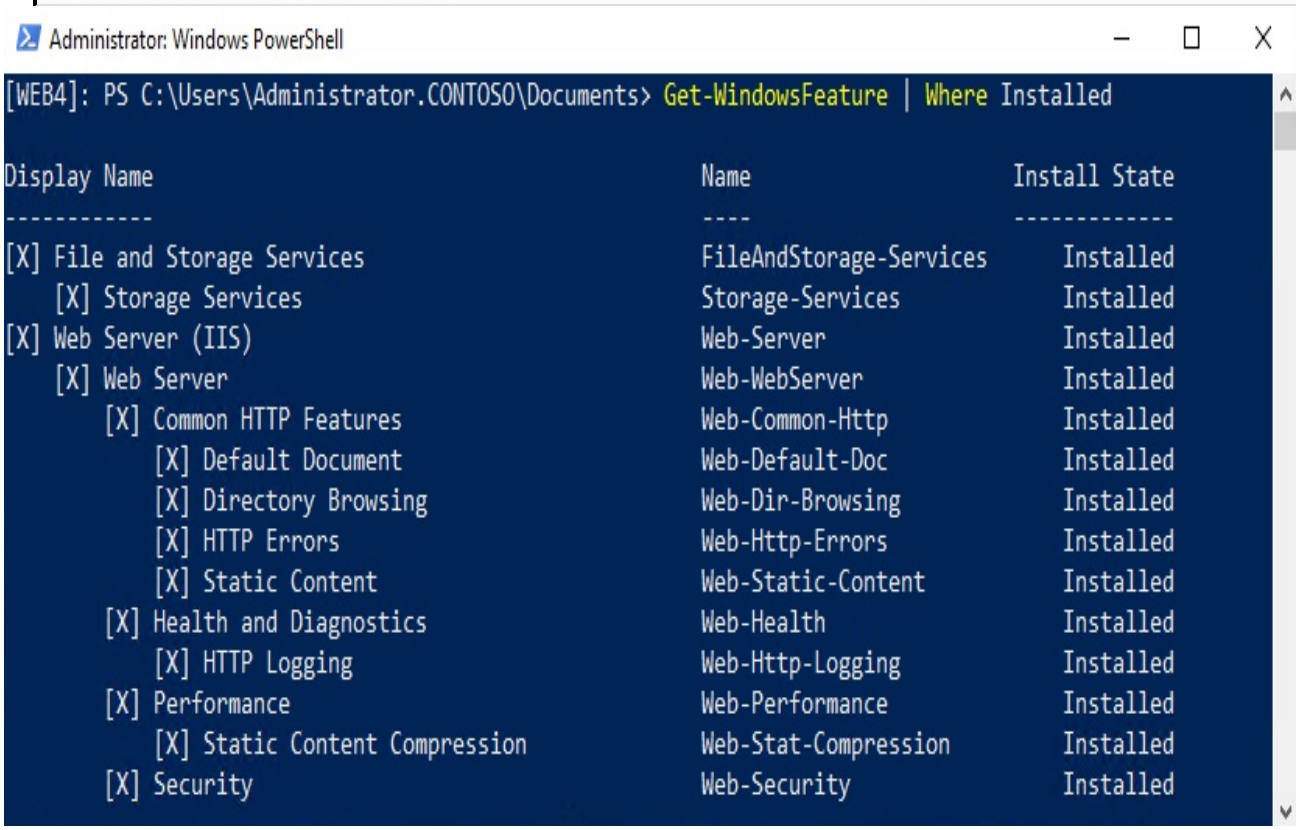
> *If your user account does not have access to the server, you can specify alternative credentials to be used when creating this remote connection. Simply append your* `Enter-PSSession` *cmdlet with* `-Credential USERNAME` *in order to specify a different user account.*

Commands that I issue from this point forward will be executed against `WEB4`. Let's verify this. If I check a simple `$env:computername`, you can see that it presents me with the `WEB4` hostname:



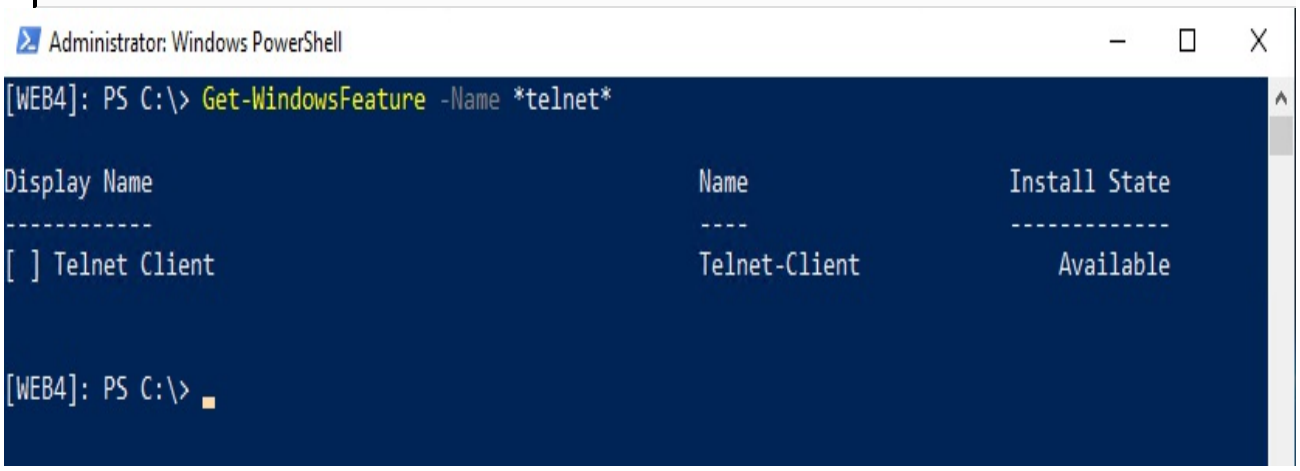
And to further verify this, if I check the installed Windows roles and features, you can see that I have the Web Server role installed, as we accomplished when we initially configured this Server Core to be a web server. Clearly, I do not have the Web Server role installed onto my Windows 10 workstation; PowerShell is pulling this data from the `WEB4` server.

```
Get-WindowsFeature | Where Installed
```



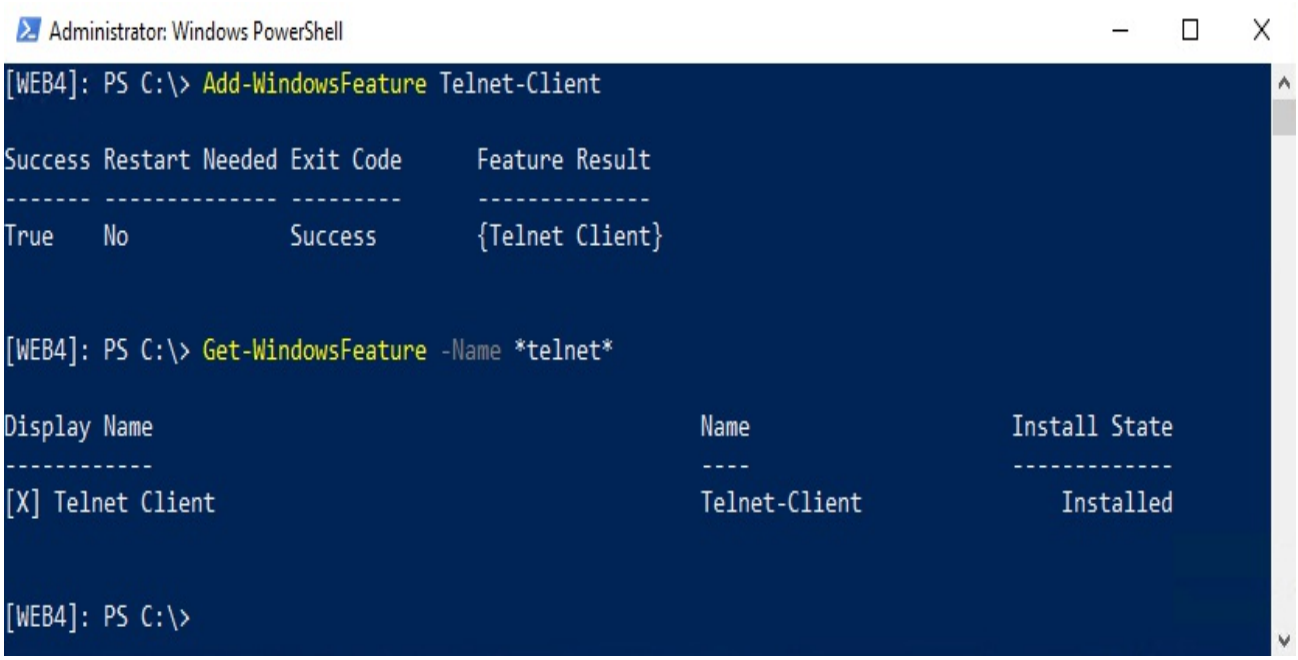
This is pretty powerful stuff. We are sitting at our local desktop computer, have a remote PowerShell session running to the `WEB4` server, and are now able to pull all kinds of information from `WEB4` because it is as if we are working from PowerShell right on that server. Let's take it one step further, and try to make a configuration change on `WEB4`, just to verify that we can. Maybe we can install a new feature onto this server. I use Telnet Client quite a bit for network connectivity testing, but can see that it is currently not installed on `WEB4`.

```
Get-WindowsFeature -Name *telnet*
```



By using the `Add-WindowsFeature` cmdlet, I should be able to make quick work of installing that feature:

```
Add-WindowsFeature Telnet-Client
```



This remote PowerShell capability is powerful stuff, not only for your servers that are running the full-blown Desktop Experience graphical interface, but also for interacting with your security-focused Server Core deployments. Becoming familiar with working in remote PowerShell sessions will be essential to a successful deployment of Server Core in your infrastructure.

Desired State Configuration

There is some new and powerful functionality in the more recent versions of PowerShell, provided by something called **Desired State Configuration** (**DSC**). DSC is a management platform plugged into PowerShell, which provides some new functions and cmdlets that you can take advantage of in your scripts to enable some really cool features. As the name implies, it allows you to build configurations inside PowerShell that will provided a *desired state*. What do I mean by that? Well, in a basic sense, DSC makes sure that the PowerShell scripts you build will always work the same way, across all of the servers where you apply them. It is quite easy to build a script in a way that it will work correctly on the server you are currently working from, but if you try to roll that same script out to a different server that might reside in a different **organizational unit** (**OU**), or have different items installed on it to begin with, then the script could produce different results than what it was originally intended to do. DSC was built to counteract these differences.

In building your DSC configuration, you identify particular roles, settings, functions, accounts, variables, and so on that you want to retain in your specific desired state. Once you identify and configure these variables, DSC will work to ensure they stay where you have them set, and that they remain uniform according to your DSC configuration policy, which means they are uniform against the other servers where you have run this script.

DSC also helps to prevent unwanted changes on servers. If your DSC-enabled script has identified that a particular service should be running

all the time on your servers, and that service stops for some reason, DSC can be there to help spin it back up so that you don't experience an outage. Alternatively, perhaps you have a script that configures a server to a particular set of standards, and another person in IT comes along and adjusts that configuration on the server itself—perhaps they log in and stop a service purposefully for some reason. Normally, this could result in an outage for that server, but DSC will get that service running again in order to maintain your originally-configured *desired state* for this server. DSC is your *scripting nanny*, so to speak. It helps to build configurations that will remain uniform across multiple platforms, and will then work to ensure these configurations are always true. You can then be confident that your servers are always running within the context of your specified desired state.

After building a configuration that identifies the items you want to be installed or monitored, something called the **Local Configuration Manager** (**LCM**) works to ensure the resources remain within the configuration specifications. LCM polls the system regularly, watching for irregularities and changes, and takes action when needed to bring your servers back into the DSC.

The ultimate goal of DSC is to keep everything constant and consistent across your servers and services. DSC's capabilities and access to reach into more and more places in the operating system grows constantly, as roles are rewritten to accept DSC parameters and monitoring. In the end, I believe that it will be Microsoft's goal that every server runs a DSC configuration script, ensuring that it is constantly working within your standards and helping to maintain your 99.999% uptime status.

There is a lot to learn about DSC, and I encourage you to explore this topic more once you are familiar with creating and using PowerShell scripts. Here are some great starting points for learning more about DSC:

- https://msdn.microsoft.com/en-us/powershell/dsc/overview
- https://mva.microsoft.com/en-US/training-courses/getting-started-with-powershell-desired-state-configuration-dsc--8672?l=ZwHuclG1_2504984382

Summary

In Windows Server 2019, we see in multiple places that administration via PowerShell is the recommended path for interacting with our servers. Because the management GUIs are now just shells running PowerShell scripts and the default installation option for Windows Server is Server Core, we can assume that headless, command-line oriented servers are going to be our servers of the future. Even though PowerShell has been at the core of our operating system functionality since Server 2012, until this point I believe that PowerShell has been viewed by most admins as simply an alternative way of managing servers. Yeah, I know it exists and that I should start using it, and the scripting looks pretty cool, but I can still do anything I want to with the old Command Prompt or my mouse button. That old mentality is quickly changing.

Now that we are experiencing the onset of new technologies, such as DSC, we can see that PowerShell is starting to develop functionality that simply does not exist anywhere else in the operating system. This, combined with the remote management accessibility provided by the standardized PowerShell platform that can be used across all of your current Windows devices (even against servers sitting inside Azure!), means that we will definitely be seeing more and more PowerShell in subsequent Microsoft operating systems and services. The next chapter deals with containers and nano server.

Questions

1. What is the fastest way to get from a Command Prompt into PowerShell?
2. What is the cmdlet that will display all available PowerShell cmdlets?
3. What PowerShell cmdlet can be used to connect your PowerShell prompt to a remote computer?
4. What file extension does a PowerShell scripting file have?
5. To which setting is the Default Execution Policy configured on a fresh Windows Server 2019 instance?
6. What key on your keyboard can be used to auto-populate the remainder of a cmdlet or filename when working in a PowerShell prompt?
7. Which service must be running on a system before it can be connected to by a remote PowerShell connection?

Containers and Nano Server

Many of the new technologies included in Windows Server 2019 are designed to reflect capabilities provided by cloud computing, bringing your private clouds to life and granting you the ability to produce the same solutions given to you by public cloud providers within your own physical infrastructure. The last few iterations of the Server operating system have also revolved around virtualization, and the idea of application containers is something that taps into both of these mindsets. Application containers are going to make the deployment of applications more streamlined, more secure, and more efficient. Containers are a relatively new idea in the Microsoft world, and I haven't heard many IT admins talking about it yet, but that will soon change. This is something that has been enhancing Linux computing for a while now, and this newest Windows Server operating system brings it a little bit closer to home for us Microsoft-centric shops.

Application developers will be very interested in the application containers provided by Windows Server 2019, and in truth, they probably understand the concepts behind containers much better than a traditional server administrator. While the premise of this book is not focused on development opportunities and is clearly not focused on Linux, we are going to discuss containers because the benefits provided are not only for developers. We, as system operations, will also benefit from using containers, and if nothing else it is going to be important for us to know and understand how to conceptualize, and how to spin up containers so that we can provide the infrastructure that our developers are going to require.

In this chapter, we will cover some topics dealing with application containers; specifically, the new capabilities that are available in Windows Server 2019 to bring this technology into our data centers:

- Understanding application containers
- Containers and Nano Server
- Windows Server Containers versus Hyper-V Containers
- Docker and Kubernetes
- Working with containers

Understanding application containers

What does it mean to *contain* an application? We have a pretty good concept these days of containing servers, by means of virtualization. Taking physical hardware, turning it into a virtualization host-like Hyper-V, and then running many virtual machines on top of it is a form of containment for those VMs. We are essentially tricking them into believing that they are their own entity, completely unaware that they are sharing resources and hardware with other VMs running on that host. At the same time that we are sharing hardware resources, we are able to provide strong layers of isolation between VMs, because we need to make sure that access and permissions cannot bleed across VMs – particularly in a cloud provider scenario, as that would spell disaster.

Application containers are the same idea, at a different level. Where VMs are all about virtualizing hardware, containers are more like virtualizing the operating system. Rather than creating VMs to host our applications, we can create containers, which are much smaller. We then run applications inside those containers, and the applications are tricked into thinking that they are running on top of a dedicated instance of the operating system.

A huge advantage to using containers is the unity that they bring between the development and operations teams. We hear the term DevOps all the time these days, which is a combination of development and operation processes in order to make the entire application-rollout

process more efficient. The utilization of containers is going to have a huge impact on the DevOps mentality, since developers can now do their job (develop applications) without needing to accommodate for the operations and infrastructure side of things. When the application is built, operations can take the container within which the application resides, and simply spin it up inside their container infrastructure, without any worries that the application is going to break servers or have compatibility problems.

I definitely foresee containers taking the place of many virtual machines, but this will only happen if admins jump in and try it out for themselves. Let's discuss a few particular benefits that containers bring to the table.

Sharing resources

Just like when we are talking about hardware being split up among VMs, application containers mean that we are taking physical chunks of hardware and dividing them up among containers. This allows us to run many containers from the same server – whether a physical or virtual server.

However, in that alone, there is no benefit over VMs, because they simply share hardware as well. Where we really start to see benefits in using containers rather than separate VMs for all of our applications is that all of our containers can share the same base operating system. Not only are they spun up from the same base set, which makes it extremely fast to bring new containers online, it also means that they are sharing the same kernel resources. Every instance of an operating system has its own set of user processes, and often it is tricky business to run multiple applications together on servers because those applications traditionally have access to the same set of processes, and have the potential to be negatively affected by those processes. In other words, it's the reason that we tend to spin up so many servers these days, keeping each application on its own server, so that they can't negatively impact each other. Sometimes apps just don't like to mix. The kernel in Windows Server 2019 has been enhanced so that it can handle multiple copies of the user mode processes. This means you not only have the ability to run instances of the same application over many different servers, but it also means that you can run many different applications, even if they don't typically like to coexist, on the same server.

Isolation

One of the huge benefits of application containers is that developers can build their applications within a container running on their own workstation! A host machine for hosting containers can be a Windows Server, or it can be a Windows 10 workstation. When built within this container sandbox, developers will know that their application contains all of the parts, pieces, and dependencies that it needs in order to run properly, and that it runs in a way that doesn't require extra components from the underlying operating system. This means the developer can build the application, make sure it works in their local environment, and then easily slide that application container over to the hosting servers where it will be spun up and ready for production use. That production server might even be a cloud-provided resource, but the application doesn't care. The isolation of the container from the operating system helps to keep the application standardized in a way that it is easily mobile and movable, and saves the developer time and headaches since they don't have to accommodate differences in underlying operating systems during the development process.

The other aspect of isolation is the security aspect. This is the same story as multiple virtual machines running on the same host, particularly in a cloud environment. You want security boundaries to exist between those machines, in fact most of the time you don't want them to be aware of each other in any way. You even want isolation and segregation between the virtual machines and the host operating system, because you sure don't want your public cloud service provider snooping around inside your VMs. The same idea applies with application containers.

The processes running inside a container are not visible to the hosting operating system, even though you are consuming resources from that operating system. Containers maintain two different forms of isolation. There is namespace isolation, which means the containers are confined to their own filesystem and registry. Then there is also resource isolation, meaning that we can define what specific hardware resources are available to the different containers, and they are not able to steal from each other. Shortly, we will discuss two different categories of containers, Windows Server Containers and Hyper-V Containers. These two types of containers handle isolation in different ways, so stay tuned for more info on that topic.

We know that containers share resources and are spun up from the same base image, while still keeping their processes separated so that the underlying operating system can't negatively affect the application and also so that the application can't tank the host operating system. But how is the isolation handled from a networking aspect? Well, application containers utilize technology from the Hyper-V virtual switch in order to keep everything straight on the networking side. In fact, as you start to use containers, you will quickly see that each container has a unique IP address assigned to it in order to maintain isolation at this level.

Scalability

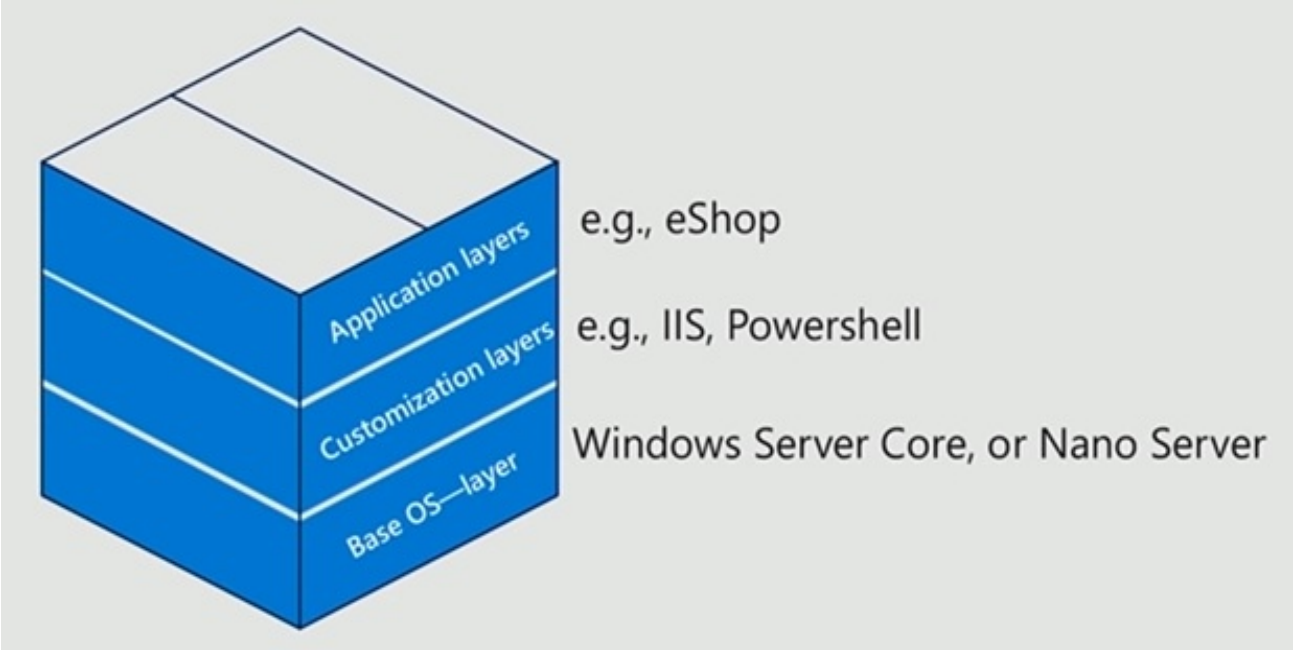
The combination of spinning up from the same base image and the isolation of the container makes a very compelling scalability and growth story. Think about a web application that you host whose use might fluctuate greatly from day to day. Providing enough resources to sustain this application during the busy times has traditionally meant that we are overpaying for compute resources when that application is not being heavily used. Cloud technologies are providing dynamic scaling for these modern kinds of applications, but they are doing so often by spinning up or down entire virtual machines. There are three common struggles with dynamically scaling applications like this. First is the time that it takes to produce additional virtual machines; even if that process is automated, your application may be overwhelmed for a period of time while additional resources are brought online. Our second challenge is the struggle that the developer needs to go through in order to make that application so agnostic that it doesn't care if there are inconsistencies between the different machines upon which their application might be running. Third is cost. Not only the hardware cost, as new VMs coming online will each be consuming an entire set of kernel resources, but monetary costs as well. Spinning virtual machines up and down in your cloud environment can quickly get expensive. These are all hurdles that do not exist when you utilize containers as your method for deploying applications.

Since application containers are using the same underlying kernel, and the same base image, their time to live is extremely fast. New containers can be spun up or down very quickly, and in batches, without having to wait for the boot and kernel mode processes to start. Also, since we have

provided the developer this isolated container structure within which to build the application, we know that our application is going to be able to run successfully anywhere that we spin up one of these containers. No more worries about whether or not the new VM that is coming online is going to be standardized correctly, because containers for a particular application are always the same, and contain all of the important dependencies that the application needs, right inside that container.

Containers and Nano Server

This topic wraps us back around to our discussion about Nano Server and why it has partially disappeared as a Windows Server installation option. Before discussing the purpose that Nano Server now serves, let's take a quick look at the structure of a Windows-based container. Here is a graphic borrowed from a public slide deck that was part of a Microsoft Ignite presentation:



The lowest layer of a container is the base operating system. When spinning up a container, you need a base set of code and kernel from which to build upon. This base operating system can be either Server Core or Nano Server.

The next layer of a container is the customization layer. This is where

the technologies that will ultimately be used by your application reside. For example, our containers may include IIS for hosting a website, PowerShell, or even something such as .NET. Any of these toolsets reside in this layer.

Finally, the top slice of the container cake is the application layer. This, of course, is the specific app that you plan to host inside this container, which your users are accessing.

While Server Core is a great operating system for building small and efficient servers, it is still a heavyweight compared to Nano Server. Nano is so incredibly different, and so incredibly tiny, that it really isn't a comparison. You probably remember earlier where we installed Server Core and came out with a hard drive size of around 6 GB. While that is much smaller than a Desktop Experience version of Windows Server, think about this. A Nano Server base image can be less than 500 MB!

That is amazingly small. Additionally, updates to Nano Server are expected to be few and far between. This means you won't have to deal with monthly patching and updates on your application containers. In fact, since containers include all that they need in order to run the applications hosted on them, it is generally expected that when you need to update something about a container, you'll just go ahead and build out a new container image, rather than update existing ones. If Nano Server gets an update, just build out a new container, install and test the application on it, and roll it out. Need to make some changes to the application itself? Rather than figuring out how to update the existing container image, it's quick and easy to build out a new one, test it outside of your production environment, and once it is ready, simply start deploying and spinning up the new container image into production, letting the old version fall away.

Nano Server is now only used as a base operating system for containers.

This is a major change since the release of Server 2016, when the scope that Nano was expected to provide was much larger. If you are utilizing Nano Server for workloads outside of container images, you need to start working on moving those workloads into more traditional servers, such as Server Core.

You may be wondering, "Why would anybody use Server Core as the base for a container image, if Nano Server is available?" The easiest answer to that question is application compatibility. Nano Server is incredibly small, and as such it is obviously lacking much of the code that exists inside Server Core. When you start looking into utilizing containers to host your applications, it's a great idea to utilize the smaller Nano Server as a base *if possible*, but often your applications simply won't be able to run on that platform and in these cases, you will be using Server Core as the base operating system.

Windows Server containers versus Hyper-V containers

When spinning up your containers, it is important to know that there are two categories of containers that you can run in Windows Server 2019. All aspects of application containers that we have been talking about so far apply to either Windows Server containers or to Hyper-V containers. Like Windows Server Containers, Hyper-V Containers can run the same code or images inside of them, while keeping their strong isolation guarantees to make sure the important stuff stays separated. The decision between using Windows Server Containers or Hyper-V Containers will likely boil down to what level of security you need your containers to maintain. Let's discuss the differences between the two so that you can better understand the choice you are facing.

Windows Server Containers

In the same way that Linux containers share the host operating system kernel files, Windows Server Containers make use of this sharing in order to make the containers efficient. What this means, however, is that while namespace, filesystem, and network isolation is in place to keep the containers separated from each other, there is some potential for vulnerability between the different Windows Server Containers running on a host server. For example, if you were to log into the host operating system on your container server, you would be able to see the running processes of each container. The container is not able to see the host or other containers, and is still isolated from the host in various ways, but knowing that the host is able to view the processes within the container shows us that some interaction does exist with this level of sharing. Windows Server Containers are going to be most useful in circumstances where your container host server and the containers themselves are within the same *trust boundary*. In most cases, this means that Windows Server Containers are going to be most useful for servers that are company-owned, and only run containers that are owned and trusted by the company. If you trust both your host server and your containers, and are okay with those entities trusting each other, deploying regular Windows Server Containers is the most efficient use of your hardware resources.

Hyper-V Containers

If you're looking for an increased amount of isolation and stronger boundaries, that is where you will foray into Hyper-V Containers. Hyper-V Containers are more like a super-optimized version of a virtual machine. While kernel resources are still shared by Hyper-V Containers, so they are still much more performant than full virtual machines, each Hyper-V Container gets its own dedicated Windows shell within which a single container can run. This means you have isolation between Hyper-V Containers that is more on par with isolation between VMs, and yet are still able to spin up new containers at will and very quickly because the container infrastructure is still in place underneath. Hyper-V Containers are going to be more useful in multi-tenant infrastructures, where you want to make sure no code or activity is able to be leaked between the container and host, or between two different containers that might be owned by different entities. Earlier, we discussed how the host operating system can see into the processes running within a Windows Server Container, but this is not the case with Hyper-V Containers. The host operating system is completely unaware of, and unable to tap into, those services that are running within the Hyper-V Containers themselves. These processes are now invisible.

The availability of Hyper-V Containers means that even if you have an application that must be strongly isolated, you no longer need to dedicate a full Hyper-V VM to this application. You can now spin up a Hyper-V Container, run the application in that container, and have full isolation for the application, while continuing to share resources and provide a better, more scalable experience for that application.

Docker and Kubernetes

Docker is an open source project – a toolset, really – that was originally designed to assist with the running of containers on Linux operating systems. Wait a minute, what? The words **Linux** and **open source** written once again inside a Microsoft book! What is this world coming to? You see, containers are quickly becoming a big deal, and rightfully so. In Server 2016, Microsoft took some steps to start reinventing the container wheel, with the inclusion of PowerShell cmdlets that could be used to spin up and control containers running on your Windows Server, but the Docker platform has been growing at such a fast rate that Microsoft really now expects that anyone who wants to run containers on their Windows machines is going to do so via the Docker toolset. If you want to utilize or even test containers in your environment, you'll need to get Docker for Windows in order to get started.

Docker is a container *platform*. This means that it provides the commands and tools needed to download, create, package, distribute, and run containers. Docker for Windows is fully supported to run on both Windows 10 and on Windows Server 2019. By installing Docker for Windows, you acquire all of the tools needed to begin using containers to enhance your application isolation and scalability.

Developers have the ability to use Docker to create an environment on their local workstation that mirrors a live server environment, so that they can develop applications within containers and be assured that they will actually run once those applications are moved to the server. Docker is the platform that provides pack, ship, and run capabilities for

your developers. Once finished with development, the container package can be handed over to the system administrator, who spins up the container(s) that will be running the application, and deploys it accordingly. The developer doesn't know or care about the container host infrastructure, and the admin doesn't know or care about the development process or compatibility with their servers.

Linux containers

There is a major update that is in ongoing development when discussing the capabilities that Windows Server 2019 possesses to interact with different kinds of containers. Earlier, in Server 2016, a Windows container host server could only run Windows-based containers, because Windows Server Containers share the kernel with the host operating system, so there was no way that you could spin up a Linux container on a Windows host.

Times are a-changing, and we now have some creative new capabilities in Server 2019 to handle scenarios such as Linux containers. While these features are still being polished, there are some new options, called **Moby VM** and LCOW, that are going to enable Linux containers to run on a Windows Server container host, even running side by side with Windows containers!

This is all new enough and still being built that more details are forthcoming, but visit this link in order to check the current status on these new capabilities if you're interested in running Linux containers: https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/linux-containers.

Docker Hub

When you work with containers, you are building container images that are usable on any server instance running the same host operating system – that is the essence of what containers enable you to do. When you spin up new instances of containers, you are just pulling new copies of that exact image, which is all-inclusive. This kind of standardized imaging mentality lends well to a shared community of images, a repository, so to speak, of images that people have built that might benefit others. Docker is open source, after all. Does such a sharing resource exist, one you can visit in order to grab container image files for testing, or even to upload images that you have created and share them with the world? Absolutely! It is called **Docker Hub**, and is available at `https://hub.docker.com`.

Visit this site and create a login, and you immediately have access to thousands of container base images that the community has created and uploaded. This can be a quick way to get a lab up and running with containers, and many of these container images could even be used for production systems, running the applications that the folks here have pre-installed for you inside these container images. Or you can use Docker Hub to upload and store your own container images:

Welcome to Docker Hub

Here are a few things to get you started.



Create a Repository

Push container images to Docker Hub



Create an Organization

Manage Docker Hub repositories with your team

In fact, you really should go ahead and create an account for Docker Hub now, because if you want to follow along later in the chapter and test implementing a container with Docker, you'll need a login to do it.

Docker Trusted Registry

If you're anything like me, you think the idea of Docker Hub is a great one: a neat place to store images and even to share them among the community. However, my next inclination is to look at this through an Enterprise spyglass, which quickly turns my perspective from *neat* to *unsecured*. In other words, you may not be comfortable placing images in this public repository. Certainly not images that contain anything sensitive to your organization, anyway.



Here is where **Docker Trusted Registry** may be something to look into. Docker Trusted Registry is a container image repository system, similar to Docker Hub, but it's something that you can contain within your network, behind your own firewalls and security systems. This gives you a container image-repository system without the risk of sharing sensitive information with the rest of the world.

Kubernetes

While Docker is our primary interface for building and hosting containers, allowing us to create platforms within which we can host applications in this new and exciting way, the real magic comes after we are finished with the container's setup. Let's peek into the future a little, and assume that you have an application now successfully hosted inside a container. This container is able to be spun up on a container host server in your own environment, or even slid over to an Azure container host very easily. This provides easy interaction with the infrastructure needed to be able to seamlessly scale this application up or down, but there is a missing piece to this scalability: **orchestration**.

Kubernetes is a container-orchestration solution. This means that Kubernetes orchestrates, or facilitates, how the containers run. It is the tool that enables many containers to run together, in harmony, as if they were one big application. If you intend to use containers to create scaling applications that have the ability to spin up new containers whenever additional resources are needed, you will absolutely need to have a container orchestrator, and Kubernetes is currently the best and most popular.

Microsoft recognized this popularity, and has taken steps to ensure that Kubernetes is fully supported on Windows Server 2019.

As with any software, Kubernetes is not the only name in the game. In fact, Docker has its own orchestration platform, called Docker Swarm. While it might make sense that Docker and Docker Swarm would work together better than Docker and any other orchestrator, the numbers

don't lie. A recent report shows that 82% of companies using scaling applications in the cloud are utilizing Kubernetes for their container orchestration.

As I mentioned earlier, tools such as containers, Docker, and Kubernetes are part of a cloud-first vision. While the use of containers for most companies is going to start onsite, by using their own servers and infrastructure for the hosting of containers, this is a technology that is already capable of extending to the cloud. Because the containers themselves are so standardized and fluid – making them easy to expand and move around – sliding them into a cloud environment is easily done.

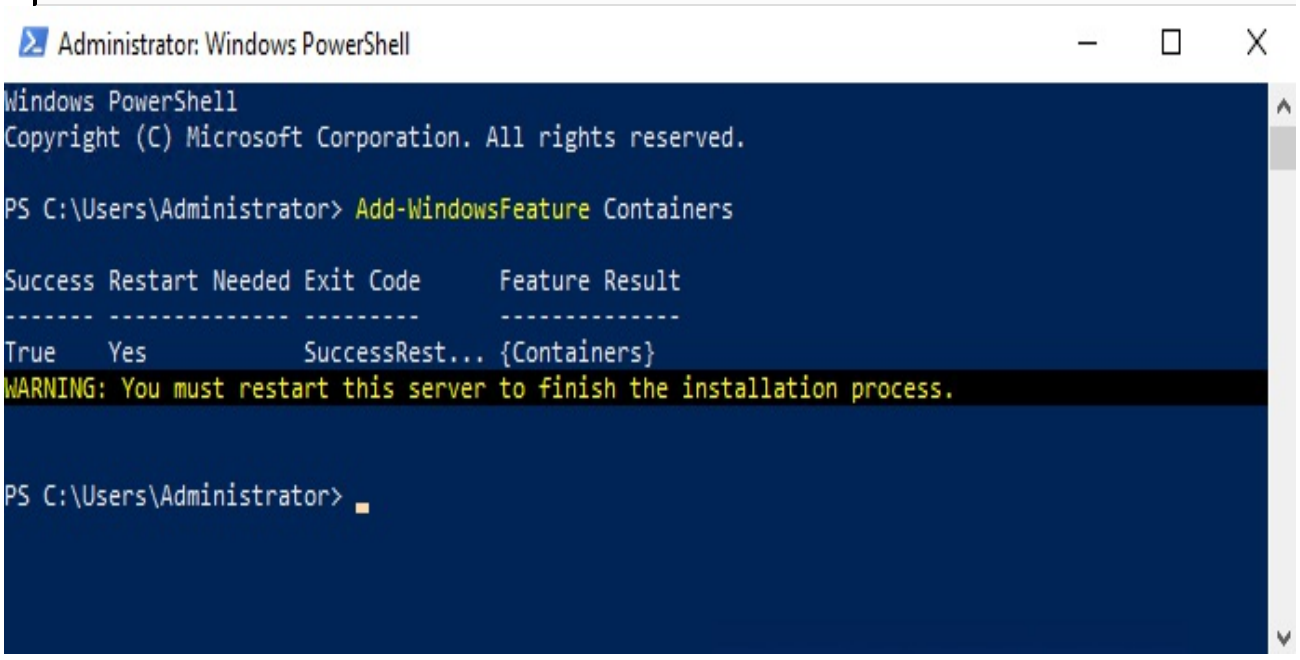
Working with containers

There are a lot of moving pieces that work together to make containers a reality in your environment, but it's really not too difficult to get started. Let's walk through the initial setup of turning Windows Server 2019 into a container-running mega machine.

Installing the role and feature

The amount of work that you need to accomplish here depends on whether you want to run Windows Server Containers, Hyper-V Containers, or both. The primary feature that you need to make sure that you install is **Containers**, which can be installed by using either the Add roles and features link from inside Server Manager, or by issuing the following PowerShell command:

```
Add-WindowsFeature Containers
```



Additionally, if you intend to run Hyper-V Containers, you need to ensure that the underlying Hyper-V components are also installed onto your container host server. To do that, install the **Hyper-V role** and accompanying management tools onto this same server.

As indicated following the role and feature installation, make sure to restart your server after these changes.

At this point, you may be wondering, "If my container host server needs to have the Hyper-V role installed, doesn't that mean it must be a physical server? You can't install the Hyper-V role onto a virtual machine, right?" *Wrong*. Windows Server 2019 supports something called nested virtualization, which was added for the purpose of containers. You see, requiring physical hardware is becoming a limiting factor for IT departments these days, as almost everything is done from virtual machines. It makes sense that companies would want to deploy containers, but they may also want their container host servers to be VMs, with multiple containers being run within that VM. Therefore, nested virtualization was required to make this possible. If you are running a Windows Server 2019 physical hypervisor server, and a Windows Server 2019 virtual machine inside that server, you will now find that you are able to successfully install the Hyper-V role right onto that VM. I told you virtual machines were popular, so much so that they are now being used to run other virtual machines!

> *Remember that we can also host and run containers on our Windows 10 machines! In order to prep a Win10 client for this purpose, simply add the Windows feature called Containers, just like on the server operating system.*

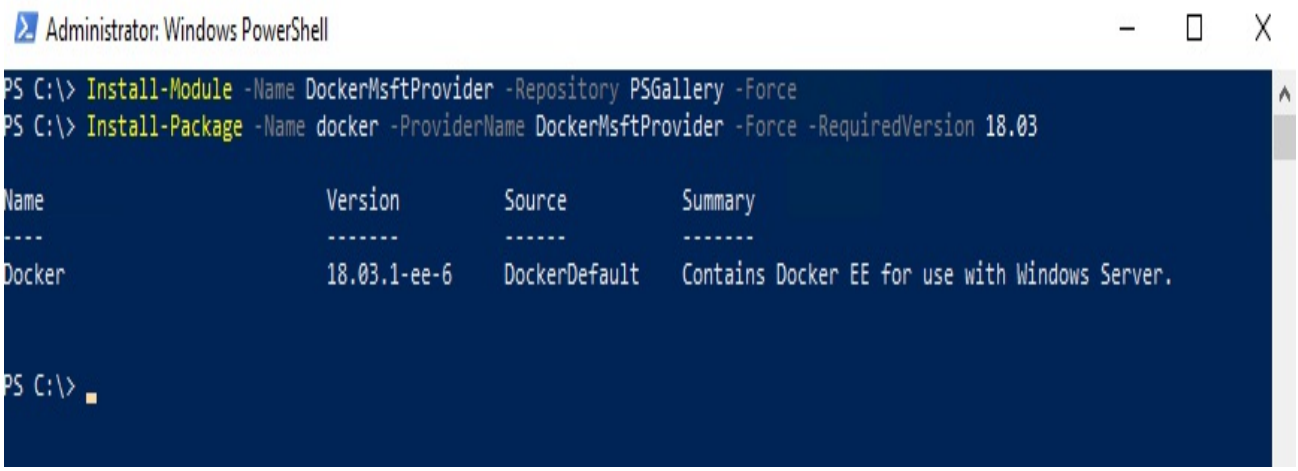
Installing Docker for Windows

Now that our container host server is prepped with the necessary Windows components, we need to grab Docker for Windows from the internet. The Docker interface is going to provide us with all of the commands that are needed in order to start building and interacting with our containers.

This is the point where that Docker Hub login becomes important. If you are working through this in order to test containers on your own workstation and need to install Docker Desktop for Windows on your Win10 client, the easiest way is to visit Docker Hub, log in, and search for the Docker client software. Here is a link to that software (this is the tool you need to use if you are installing on Windows 10): https://hub.docker.com/editions/community/docker-ce-desktop-windows.

However, since I am sitting on a Windows Server 2019, my license for the server also includes licensing for Docker Enterprise, which can be pulled down without having to visit Docker Hub. If I open up an elevated PowerShell prompt and run the following two commands, my server will reach out and grab Docker Enterprise, and install it onto my server:

```
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
Install-Package -Name docker -ProviderName DockerMsftProvider -Force -RequiredVersion
18.03
```



After package installation has finished, Docker is now configured on your server as a service, but that service needs to be started with the following command:

```
Start-Service docker
```

Docker commands

Once Docker is installed on your system, whether you are working with a Windows Server 2019 or a Windows 10 machine, you now have the Docker Engine running on your machine and it is ready to accept some commands in order to begin working with containers. If there is a single word to remember when it comes to working with containers, it is **Docker**. That is because every command that you issue to interact with containers will begin with the word *docker*. Let's have a look are some of the common commands that you will be working with.

docker --help

This is sort of like issuing *docker /?*, if that were a real command. The `help` function for Docker will generate a list of the possible docker commands that are available to run. This is a good reference point as you get started.

docker images

After downloading some container images from a repository (we will do this for ourselves in the next section of this chapter), you can use the `docker images` command to view all of the images that are available on your local system.

docker search

Utilizing the `search` function allows you to search the container repositories (such as Docker Hub) for base container images that you might want to utilize in your environment. For example, in order to search and find images provided from inside Microsoft's Docker Hub repository, issue the following:

```
docker search microsoft
```

docker pull

We can use `docker pull` to pull down container images from online repositories. There are multiple repositories from which you can get container images. Most often, you will be working with images from Docker Hub, which is where we will pull a container image from shortly. However, there are other online repositories from which you can get container images, such as Microsoft's public container registry, known as MCR.

Here are some sample `docker pull` commands showing how to pull container images from Docker Hub, as well as MCR:

```
docker pull Microsoft\nanoserver
docker pull Microsoft\windowsservercore
docker image pull mcr.microsoft.com/windows/servercore:1809
docker image pull mcr.microsoft.com/windows/nanoserver:1809
```

docker run

This is the command for starting a new container from a base image. You will find that you can retain multiple container images in your local repository that are all based off the same container image. For example, as you add new things into your containers or update the application inside your containers, you may be building new container images that are now a subset of an existing container image. You may have numerous container images that are all named `windowsservercore`, for example. In this case, container tags become very important, as tags help you to distinguish between different versions of those container images. As an example, here is a command that would start a container based on a `windowsservercore` image for which I had associated the `ltsc2019` tag:

```
docker run -it --rm Microsoft\windowsservercore:ltsc2019
```

In the preceding command, the `-it` switch creates a shell from which we can interact with a container, which is useful for building and testing containers, but you generally wouldn't need that switch for launching production containers that were 100% ready to serve up applications. `--rm` is a cleanup switch, meaning that once this particular container exits, the container and its filesystem will be automatically deleted.

docker ps -a

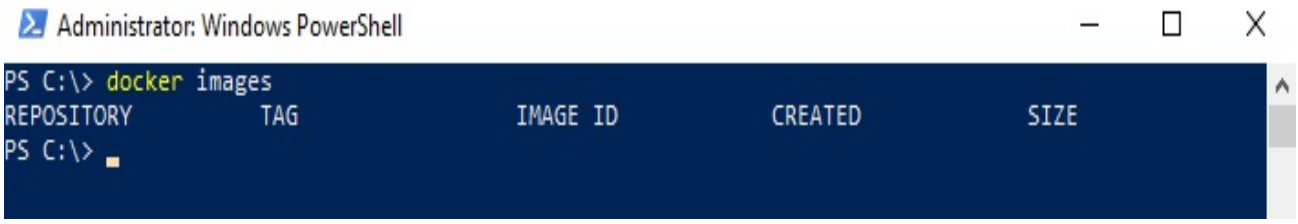
You utilize `docker ps` when you want to view the containers that are currently running on your system.

docker info

This will summarize your Docker environment, including the number of containers that you have running and additional information about the host platform itself.

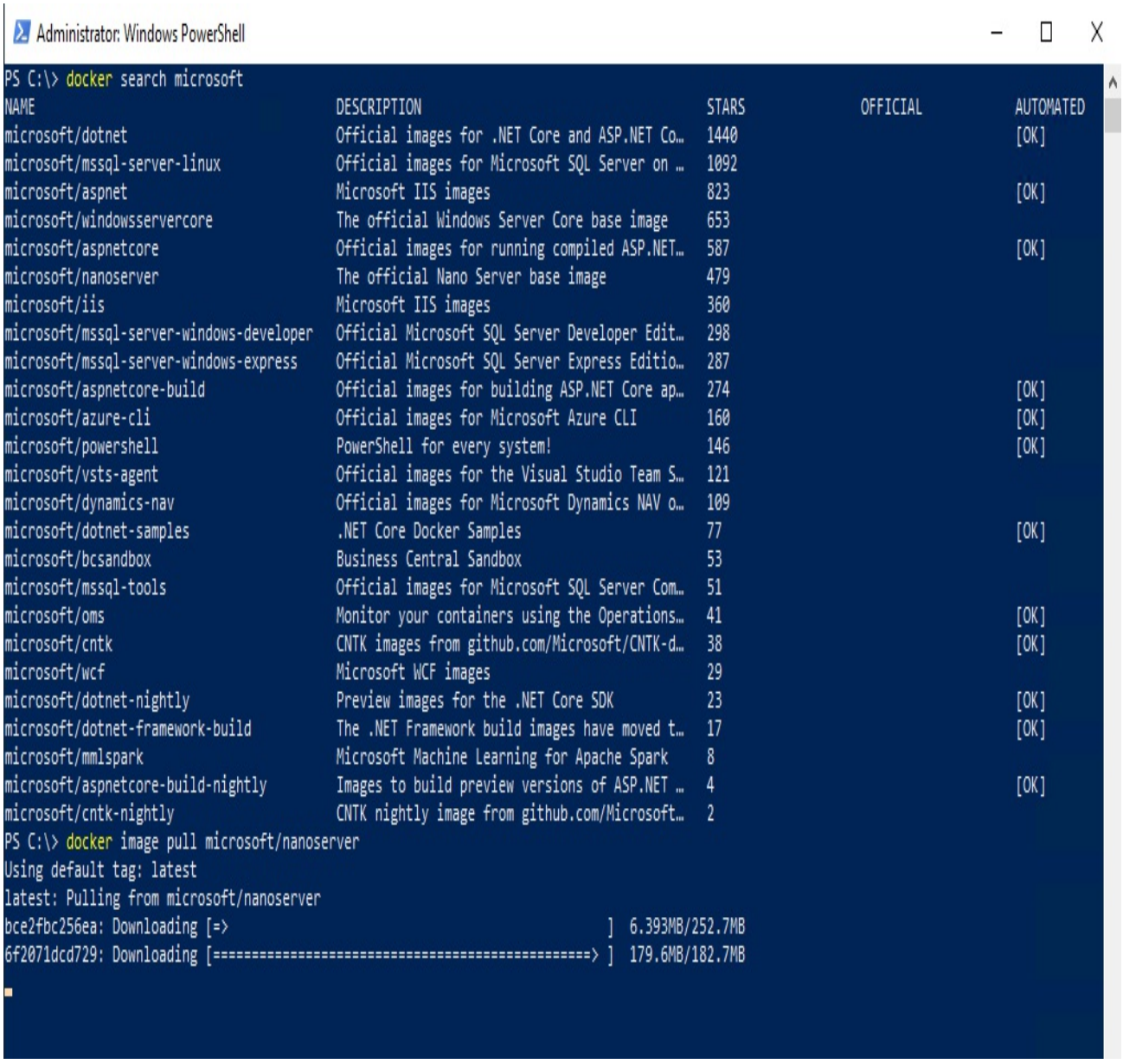
Downloading a container image

The first command we will run on our newly-created container host is `docker images`, which shows us all of the container images that currently reside on our system; there are none:



Of course there are no container images yet, as we haven't downloaded any. Let's grab a couple so that we can test this out. There is a sample container image file provided by the .NET team that showcases running a .NET application inside a Nano Server container—that one sounds like a fun way to get started with verifying that I can successfully run containers on this new host server. First, we can use `docker search` to check the current container images that reside inside Microsoft's Docker Hub repository. Once we find the image that we want to download, we use `docker pull` to download it onto our server:

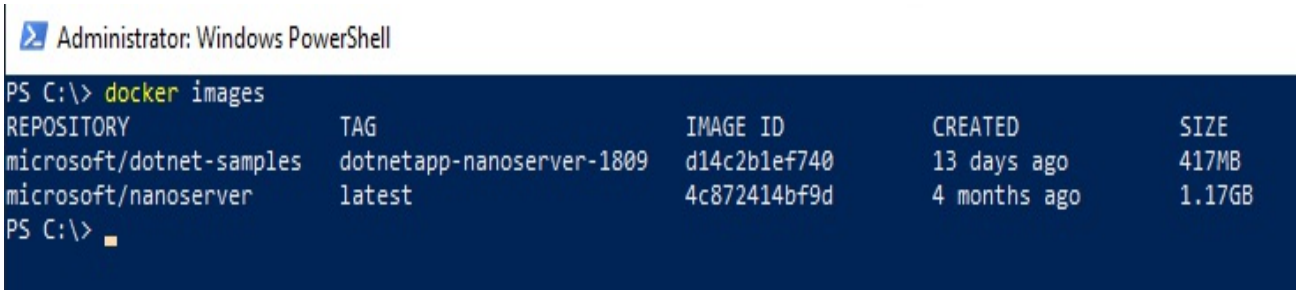
```
docker search microsoft
docker image pull microsoft/nanoserver
```



The preceding command downloaded a copy of the standard Nano Server base image, but we want to make our container do something in the end, so here is a command that will download that .NET sample image as well:

```
docker image pull microsoft/dotnet-samples:dotnetapp-nanoserver-1809
```

After the downloads are finished, running `docker images` once again now shows us the newly-available Nano Server container image, as well as the .NET sample image:



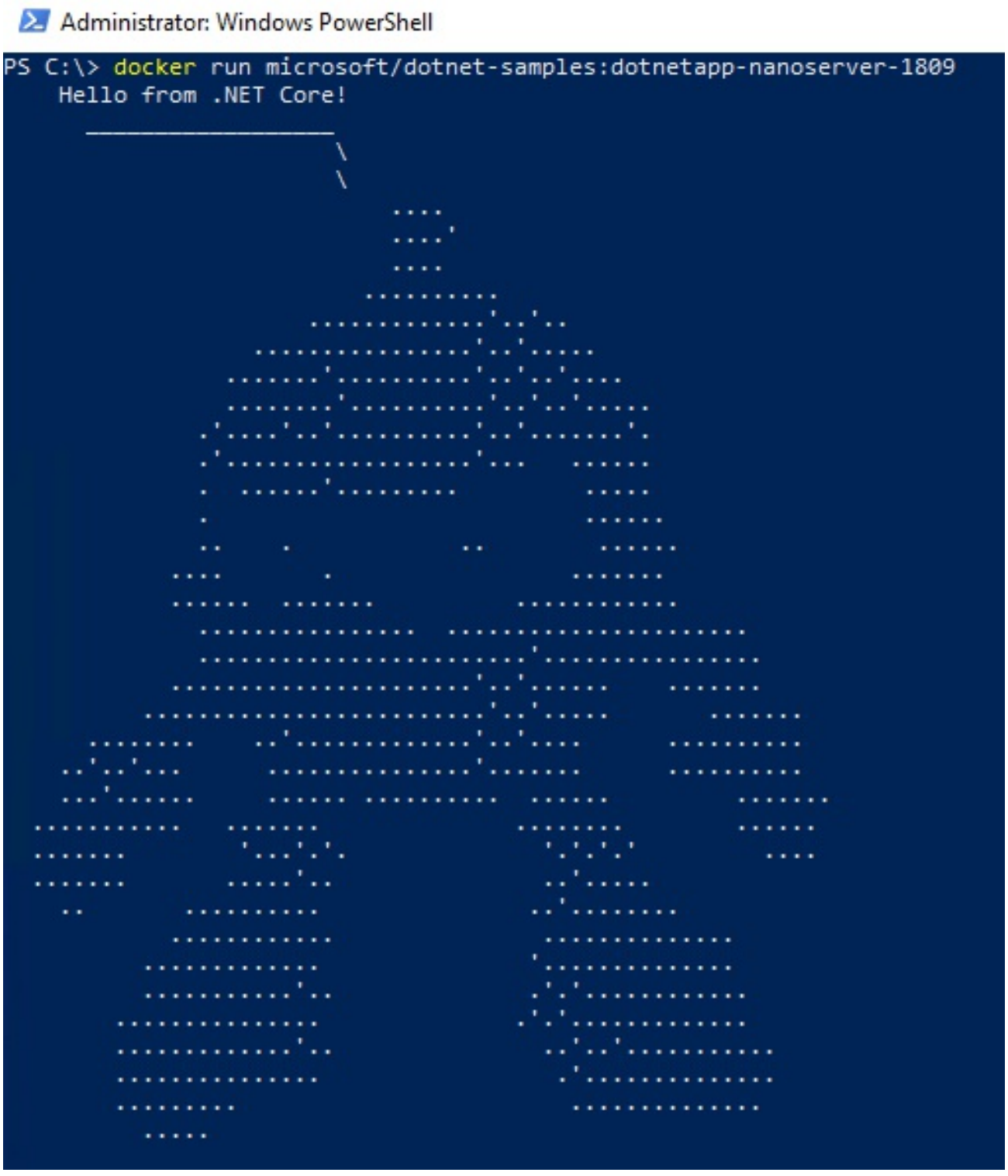
From these base images we are now able to launch and run a real container.

Running a container

We are so close to having a container running on our host! Now that we have installed the service, implemented Docker, imported the Docker module into our PowerShell prompt, and downloaded a base container image, we can finally issue a command to launch a container from that image. Let's run the .NET container that we downloaded before:

```
docker run microsoft/dotnet-samples:dotnetapp-nanoserver-1809
```

The container starts and runs through its included code, and we see some fun output:



This container showcases that *all* components necessary for this .NET

application to run are included *inside* the container. This container is based on Nano Server, which means it has an incredibly small footprint. In fact, looking back a few pages at the last Docker images command that we ran, I can see that this container image is only 417 MB! What a resource saving, when compared with running this application on a traditional IIS web server.

The main resource for Microsoft documentation on containers is `https://aka.ms/windowscontainers`. The tools used to interact with containers are constantly changing, including changes to Docker and Kubernetes. Make sure to check over the Microsoft Docs site in order to find the latest best practices and approved installation path for preparing your container host servers.

Summary

Containers are going to revolutionize the way that we build and host modern applications. By containerizing apps, we are going to be able to run many more applications on each physical server, because they are capable of being fully isolated away from each other. Additionally, the container mentality allows the development of applications to happen in a much more fluid fashion. App developers can build their applications inside containers running on their own laptops, and once finished, simply hand them over to the infrastructure team to slide that container image onto a production container host server. That host server could be on-premise, or even in the cloud. Orchestration tools such as Kubernetes can then be leveraged in order to scale that application, increasing or decreasing resource capacity and the number of necessary containers based on load or other factors. Usability of containers in the real world has been expanded greatly by the Docker project. The folks at Docker are clearly the front-runners in this space, enough that Microsoft has decided to incorporate the use of Docker – an open source project developed by Linux! – straight into Windows Server 2019. We can now utilize both the Docker engine to run containers on our Windows servers, as well as the Docker client toolset to manage and manipulate containers inside Windows in the same way we can work with containers in the Linux world.

Linux containers and Windows Server Containers have a lot in common, and function in basically the same way. Microsoft's ingenious idea to create an additional scope of container, the Hyper-V Container, brings a solid answer to a lot of common security questions that present themselves when approaching the idea of containers in general.

Everyone uses virtual machines heavily these days; I don't think anybody can disagree with that. Assuming the use of containers evolves into something that is easy to implement and administer, I foresee Hyper-V Containers replacing many of our existing Hyper-V virtual machines in the coming years. This will save time, money, and server space.

Speaking of Hyper-V, it has become such an integral part of so many of our corporate networks today. In the next, and final, chapter, we will learn more about this amazing virtualization technology.

Questions

1. A Windows Server Container can run a base OS that is one of two different types, what are they?
2. Compared to a Windows Server Container, what type of container provides even greater levels of isolation?
3. True or False—In Windows Server 2016, you could run both Windows and Linux containers on the same Windows Server host platform.
4. What is the Docker command to see a list of container images on your local system?
5. What is currently the most popular container orchestration software that integrates with Windows Server 2019?
6. True or False—Developers can install Docker onto their Windows 10 workstations in order to start building applications inside containers.

Virtualizing Your Data Center with Hyper-V

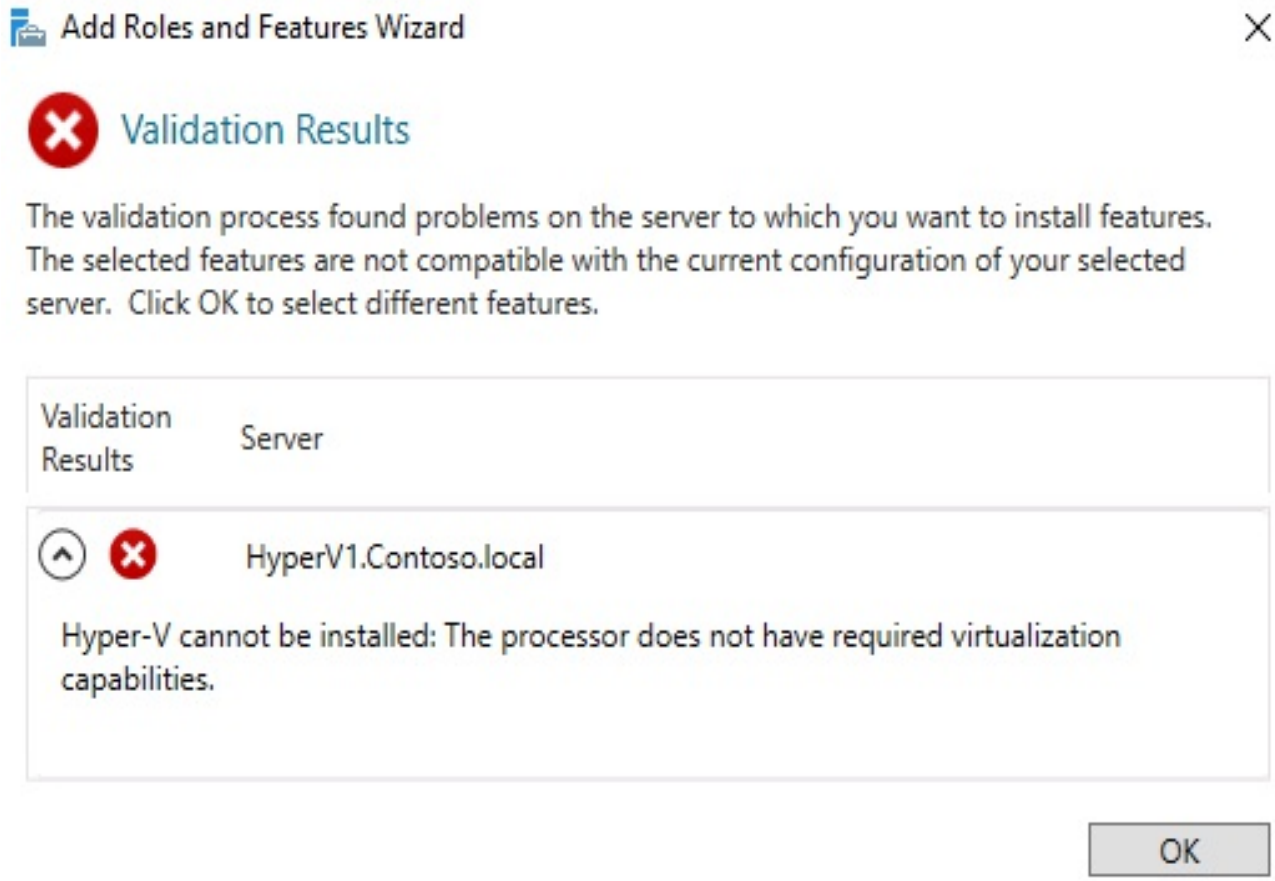
I've always been a country boy. Driving dirt roads, working on cars, and hunting tend to fill my free time. Traveling to cities, and particularly a recent trip to Hong Kong, always hits me with a bit of culture shock. All those skyscrapers and tall apartment buildings serve an important purpose though, and serve to fulfill my metaphor: if there isn't enough land to grow outward, you have to build up. The vertical ascension of large cities is similar to what we have seen happening in our data centers over the past decade. Cities need more and more places for people and businesses, just like we need to house more and more servers every year. Rather than horizontal expansion, with enormous server rooms filled with racks and racks of hardware, we are embracing the skyscraper mentality, and virtualizing everything. We build considerably fewer servers, but make them incredibly powerful. Then on top of these super computers, we can run dozens, if not hundreds, of virtual servers. The technology that provides this hypervisor layer, the ability to run **Virtual Machines** (**VMs**) in Microsoft-centric shops, is the **Hyper-V** role in Windows Server. This is one of the most critical roles to understand as a server administrator, because if your organization is not yet making use of server virtualization, trust me when I say that it will be soon. Virtualization is the way of the future. The following are some topics we are going to explore so that you can become familiar with the virtualization capabilities provided by Microsoft in Windows Server 2019:

- Designing and implementing your Hyper-V Server

- Using virtual switches
- Implementing a new virtual server
- Managing a virtual server
- Shielded VMs
- Integrating with Linux
- **Resilient Filesystem** (**ReFS**) deduplication
- Hyper-V Server 2019

Designing and implementing your Hyper-V Server

Creating your own Hyper-V Server is usually pretty simple: build a server, install the Hyper-V role, and you're ready to get started. In fact, you can even install the Hyper-V role onto a Windows 10 Pro or Enterprise computer, if you need to run some virtual machines from your own desktop. While most hardware that is being created these days fully supports the idea of being a hypervisor provider, some of you may try installing the Hyper-V role only to end up with the following error message:



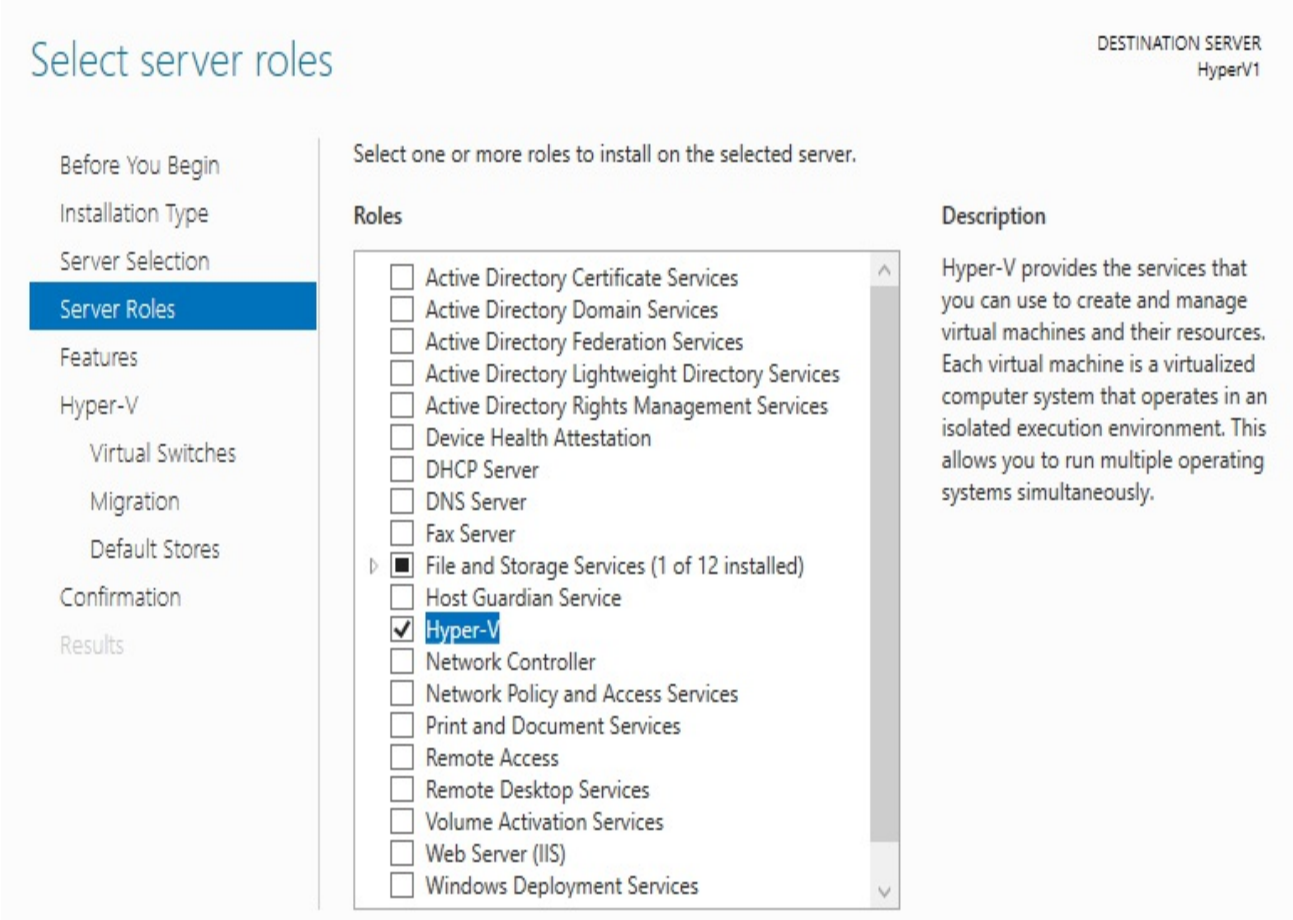
Uh oh, that's not good. This means one of two things: either my CPU really doesn't support virtualization, or I simply have some settings turned off inside the BIOS on my server that are preventing this from working. There are three considerations you should check into on your server to make sure it is ready to run Hyper-V. First, you need to be running an x64-based processor. This is kind of a given, since Windows Server 2019 only comes in 64 bit anyway. If you don't have an x64 processor, you're not going to be able to install the operating system in the first place. Second, your CPUs need to be capable of hardware-assisted virtualization. This is typically called either **Intel Virtualization Technology** (**Intel VT**) or **AMD Virtualization** (**AMD-V**). And last but not least, you must have **Data Execution Prevention** (**DEP**) available and enabled on your system. If you have investigated the hardware itself and it seems to be virtualization-

capable, but it's still not working, it is likely that you have DEP currently disabled inside the BIOS of that system. Boot into the BIOS settings and enable DEP, along with any other more-user-friendly-named settings that might indicate they are currently blocking your ability to run virtual machines.

As long as your processors are happy to run virtual machines, you can turn just about any size of hardware into a hypervisor by installing the Hyper-V role. It is not important to think about *minimum* system requirements because you want your system hardware to be as large as possible in a Hyper-V Server. The more CPU cores, RAM, and hard drive space you can provide, the more VMs you will be able to run. Even the smallest Hyper-V Servers I have seen in production environments are running hardware such as dual Xeon processors, 96 GB of RAM, and many terabytes of storage space. While 96 GB of RAM may seem like a lot for a single system, if your standard workload server build includes 8 GB of RAM, which is a pretty low number, and you want to run 12 servers on your Hyper-V Server, you are already beyond the capabilities of a Hyper-V Server with only 96 GB of RAM. 8 times 12 is 96, and you haven't left any memory for the host operating system to use! So *the moral of the story?* Go big or go home!

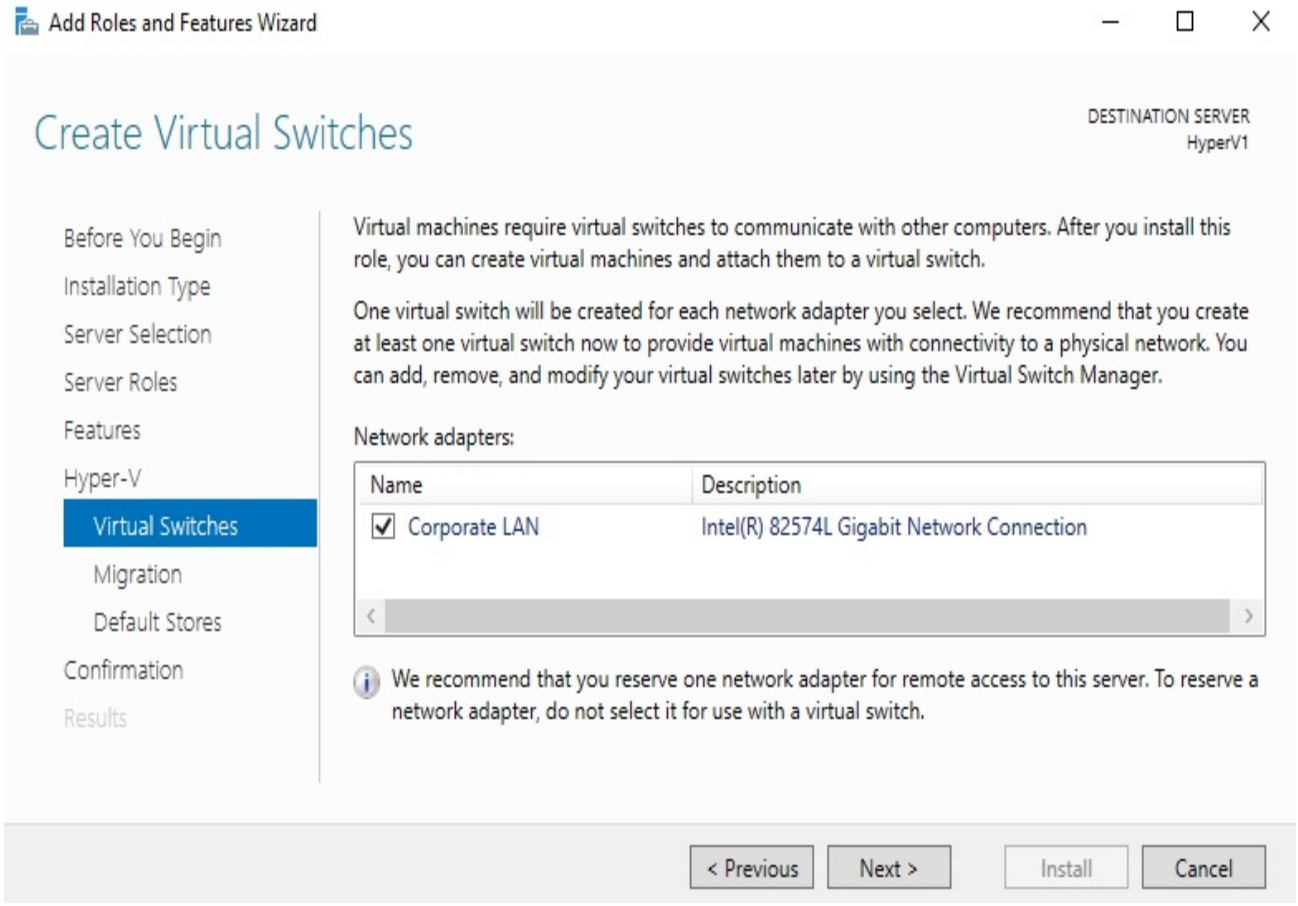
Installing the Hyper-V role

Hyper-V is just another role in Windows Server 2019, but during the installation of that role, you will be asked a few questions and it is important to understand what they are asking, so that you can be sure your new Hyper-V Server is built to last and to work in an efficient manner. First of all, you will need to have Windows Server 2019 already installed, and use the Add roles and features function in order to install the role called Hyper-V:

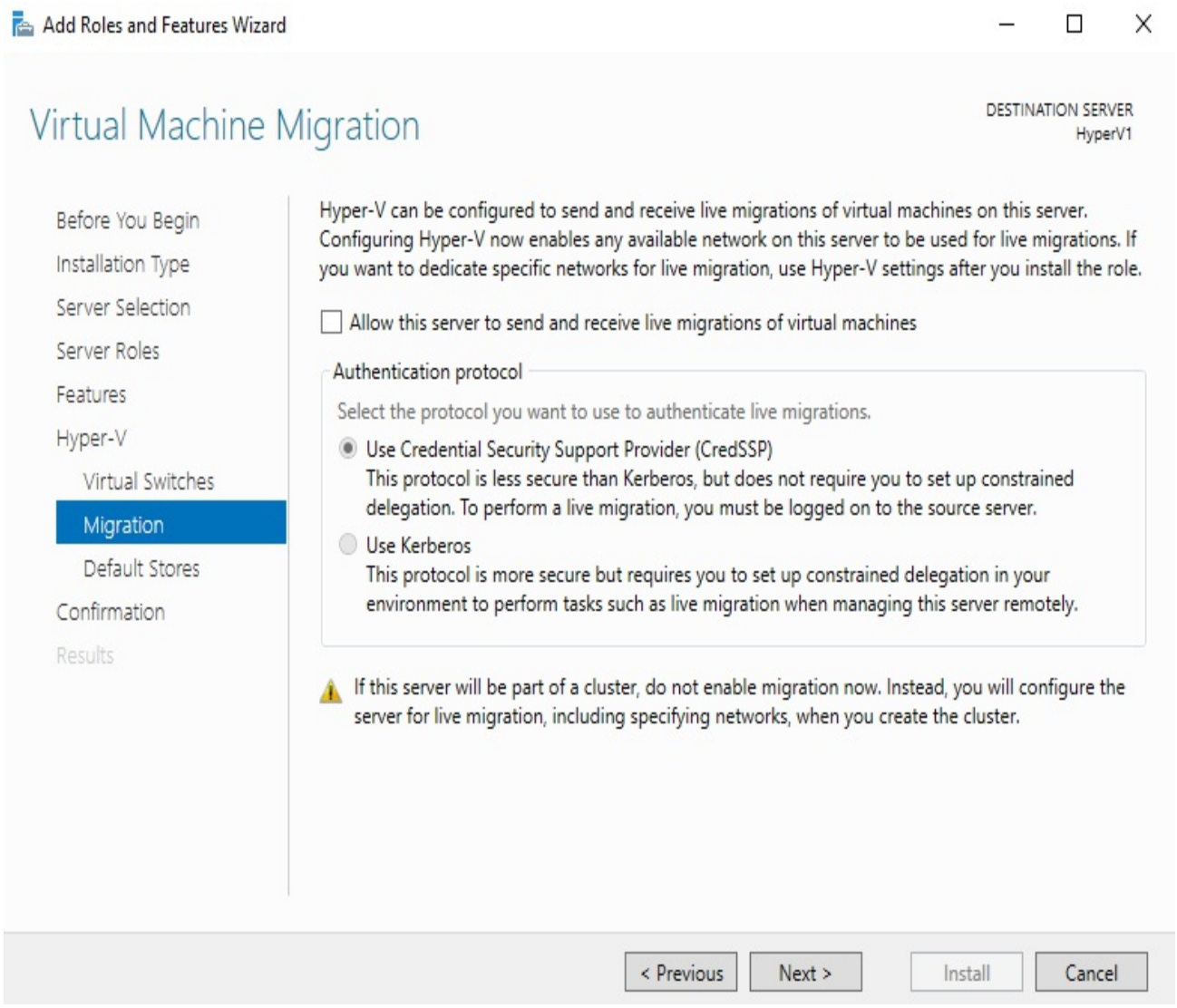


As you continue working through the wizard to install the role, you come across a screen labeled Create Virtual Switches. We will discuss networking in Hyper-V a little bit more in the next section of this chapter, but what is important here is that you get to define which of your server's physical NICs will be tied into Hyper-V, and available for your virtual machines to use. It is a good idea for each Hyper-V Server to have multiple NICs. You want one NIC dedicated to the host itself, which you would not select in this screen. Leave that one alone for the hypervisor's own communications. In addition to that NIC, you will want at least one network card that can bridge the VMs into the corporate network. This one you would select, as you can see in the upcoming screenshot. If you will be hosting many different VMs on this server, and they need to be connected to different physical networks, you might have to install many different NICs onto your Hyper-V Server:

HivaNetwork.Com



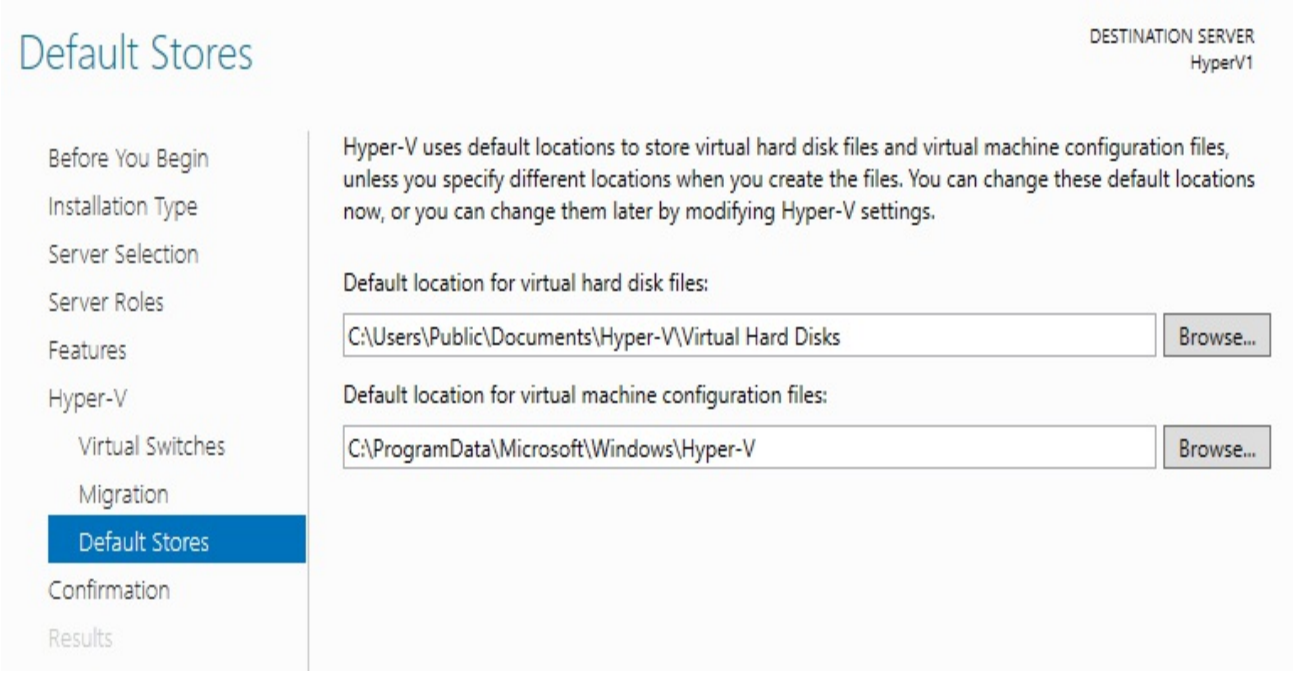
After defining NICs, we get to decide whether this Hyper-V Server will be able to handle the live migration of VMs. Live VM migration is the ability to move a VM from one Hyper-V host to another, without any interruption of service on that VM. As you can see in the following screenshot, there are a couple of different ways you can set up the server to prepare it for handling live migrations, and take note of the text at the bottom that is telling you to leave this option alone for now if you plan to make this Hyper-V Server part of a cluster. In clustered environments, these settings are handled at a different layer:



The last screen that I wanted to point out is the definition of storage locations for your VM data. After creating VMs and digging into what they look like at the hard-disk level (looking at the actual files that are created per-VM), you will see that there are two key aspects to a VM: the virtual hard disk file, **Virtual Hard Disk** (**VHD**) or VHDX, and a folder that contains the configuration files for that VM.

As you can see in the upcoming screenshot, the default locations for storing these items are something you would expect out of a client

application you were installing onto a laptop, but you wouldn't expect something as heavy as Hyper-V to be storing its core files into a shared-user `Documents` folder. I suppose since Microsoft doesn't know the configuration of your server, it can't make any real guesses as to where you want to really store that data, and so it sets the default to be something that would work technically, but should probably be changed as a matter of best practice. Many Hyper-V Servers will have dedicated storage, even if only a separate hard disk, on which these files are planned to be stored. Make sure you take a minute on this screen and change the default storage locations of your VM files:

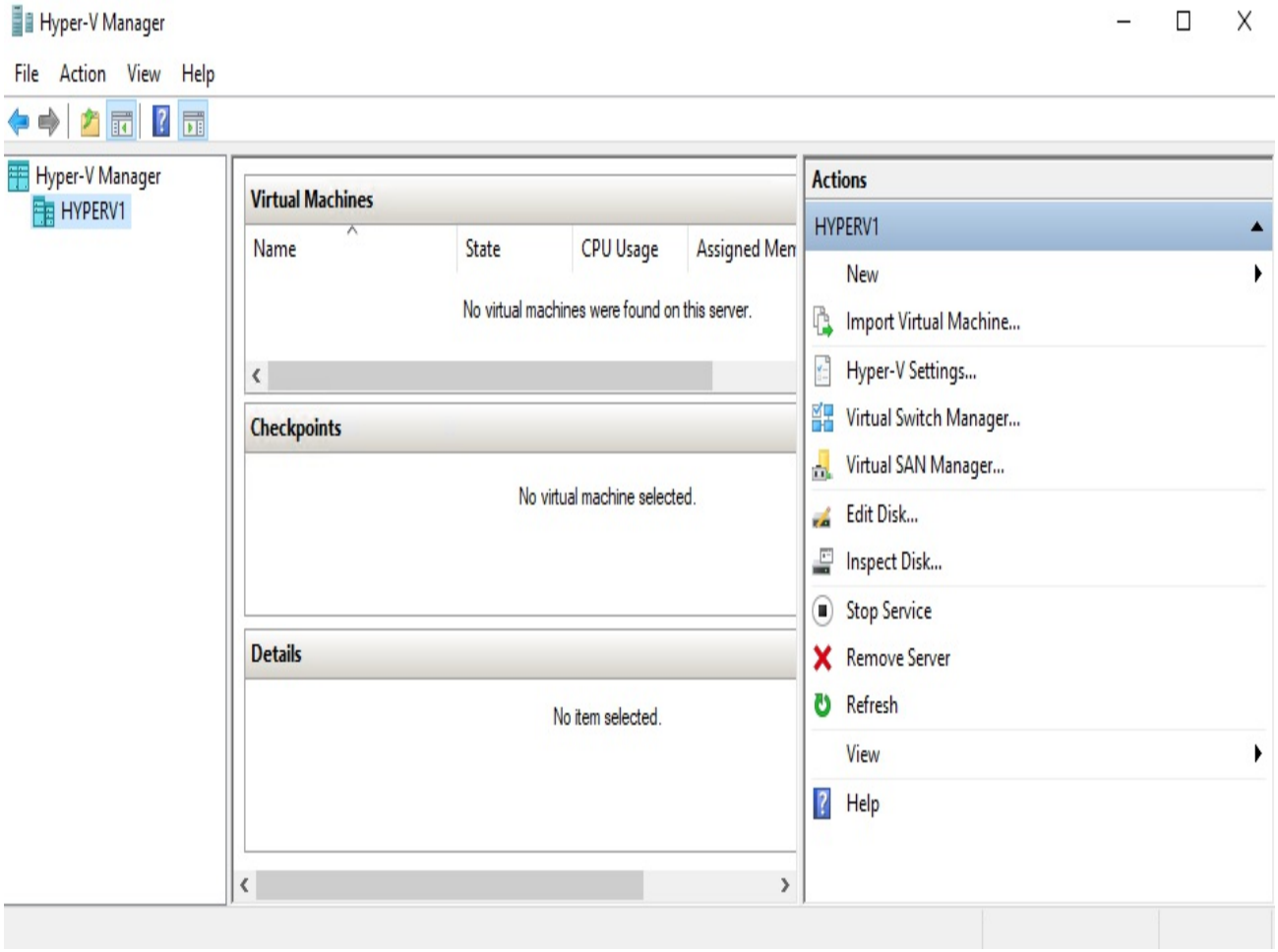


Remember that the version of Windows Server 2019 you are running determines how many VMs you will be able to run on top of this host. Server 2019 Standard limits you to running two VMs, while Datacenter edition gives you access to launch as many as you can fit on the hardware.

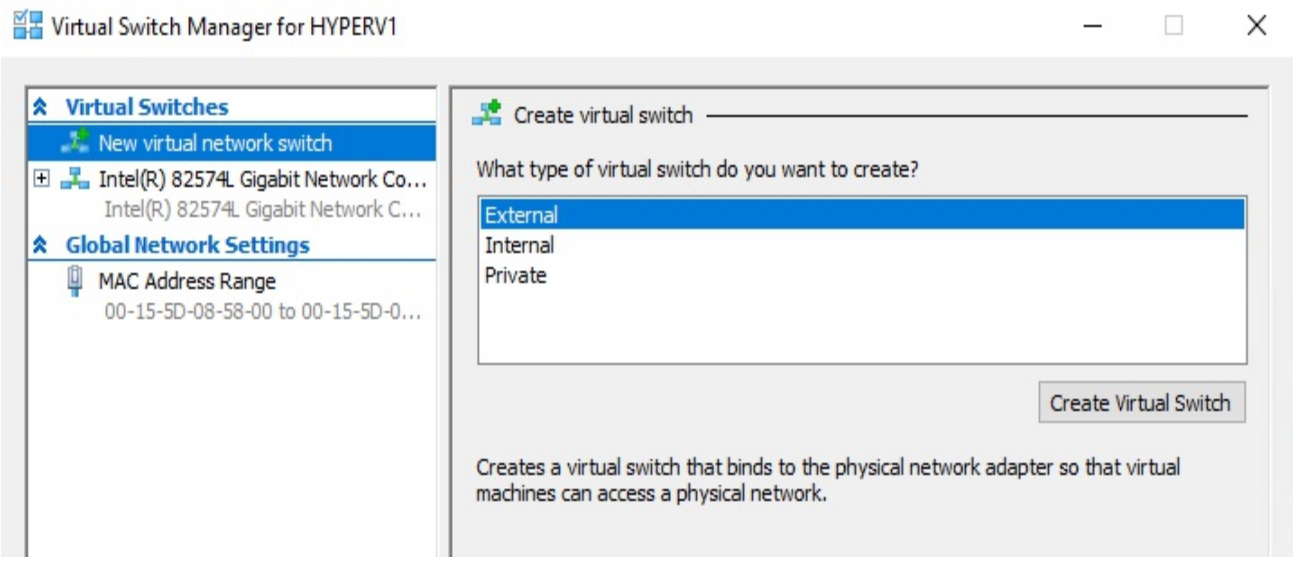
Using virtual switches

Upon completion of the Hyper-V role installation, your first inclination may be to jump right in and start creating VMs, but you should really take a minute to make sure that the networking capabilities of your Hyper-V Server are adequate to meet your needs. During the role-installation process, we selected the physical NICs that are to be passed through into Hyper-V, and that screen told us it was going to establish a virtual switch for each of these NICs. But *what does that look like inside the console?* And *what options do we have for establishing networking between our virtual machines?*

In order to answer these questions, we need to open up the management interface for Hyper-V. As with any administrative tool of a Windows role, check inside the Tools menu of Server Manager, and now that the role has been installed, you will see a new listing for Hyper-V Manager. Launch that and we are now looking at the primary platform from which you will be managing and manipulating every aspect of your Hyper-V environment:



We currently have a lot of blank space in this console, because we don't have any VMs running yet. Over on the right side of Hyper-V Manager, you can see a link that says Virtual Switch Manager.... Go ahead and click on that link to be taken into the settings for our virtual switches and networking:

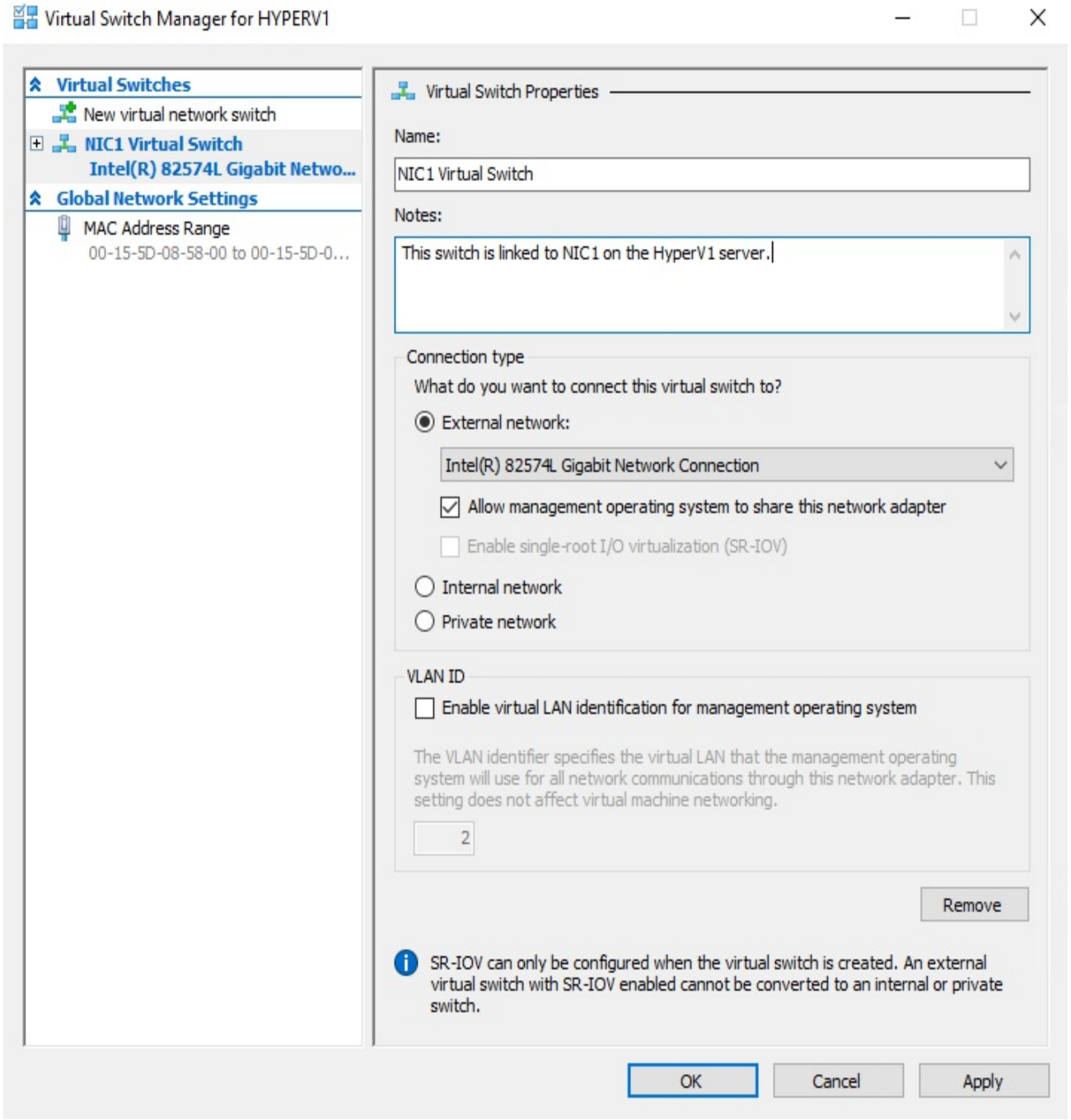


Toward the left, you see a list of current Virtual Switches. On my server, there is only one switch listed there at the moment, which is named after the physical NIC to which it is connected. This is the virtual switch that the role-installation process created for us when we selected the NIC to be included with Hyper-V. If you selected multiple NICs during the role installation, you will have multiple virtual switches available here, each corresponding to a single physical NIC. Every VM that you create will have one or more virtual NICs, and you will see shortly that you have the ability to choose where each of those virtual NICs gets connected. If there are five different physical networks that your VMs might need to contact, you can use five physical NICs in the Hyper-V Server, plug each one into a different network, and then have five virtual switches here in the console that your VM NICs can be *plugged* into.

As you can see in the previous screenshot, we have a button named Create Virtual Switch, which is self-explanatory. Obviously, this is where we go to create new switches, but there are three different types of switches that you can create. Let's take just a minute to discuss the differences between them.

The external virtual switch

The external virtual switch is the most common type to use for any VMs that need to contact a production network. Each external virtual switch binds to a physical NIC that is installed onto the Hyper-V Server. If you click on an external virtual switch, you can see that you have some options for configuring this switch, and that you can even change a switch type. In the following screenshot, I have renamed my external virtual switch so that it is easier to identify when I decide to add additional NICs to this server in the future:


Virtual Switch Manager for HYPERV1
Virtual Switches
New virtual network switch
NIC1 Virtual Switch
Intel(R) 82574L Gigabit Netwo...
Global Network Settings
MAC Address Range
00-15-5D-08-58-00 to 00-15-5D-0...
Virtual Switch Properties
Name:
NIC1 Virtual Switch
Notes:
This switch is linked to NIC1 on the HyperV1 server.
Connection type
What do you want to connect this virtual switch to?
External network:
Intel(R) 82574L Gigabit Network Connection
Allow management operating system to share this network adapter
Enable single-root I/O virtualization (SR-IOV)
Internal network
Private network
VLAN ID
Enable virtual LAN identification for management operating system
The VLAN identifier specifies the virtual LAN that the management operating system will use for all network communications through this network adapter. This setting does not affect virtual machine networking.
2
Remove
SR-IOV can only be configured when the virtual switch is created. An external virtual switch with SR-IOV enabled cannot be converted to an internal or private switch.
OK
Cancel
Apply

The internal virtual switch

Internal virtual switches are not bound to a physical NIC, and so if you create an internal virtual switch and connect a VM to it, that virtual machine will not be able to contact a physical network outside the Hyper-V Server itself. It's sort of a middleman between the other two types of switch; using an internal virtual switch is useful when you want the VM traffic to remain within the Hyper-V environment, but still provide network connectivity between the VMs and the Hyper-V host itself. In other words, VMs connected to an internal virtual switch will be able to talk to each other, and talk to the Hyper-V Server, but not beyond.

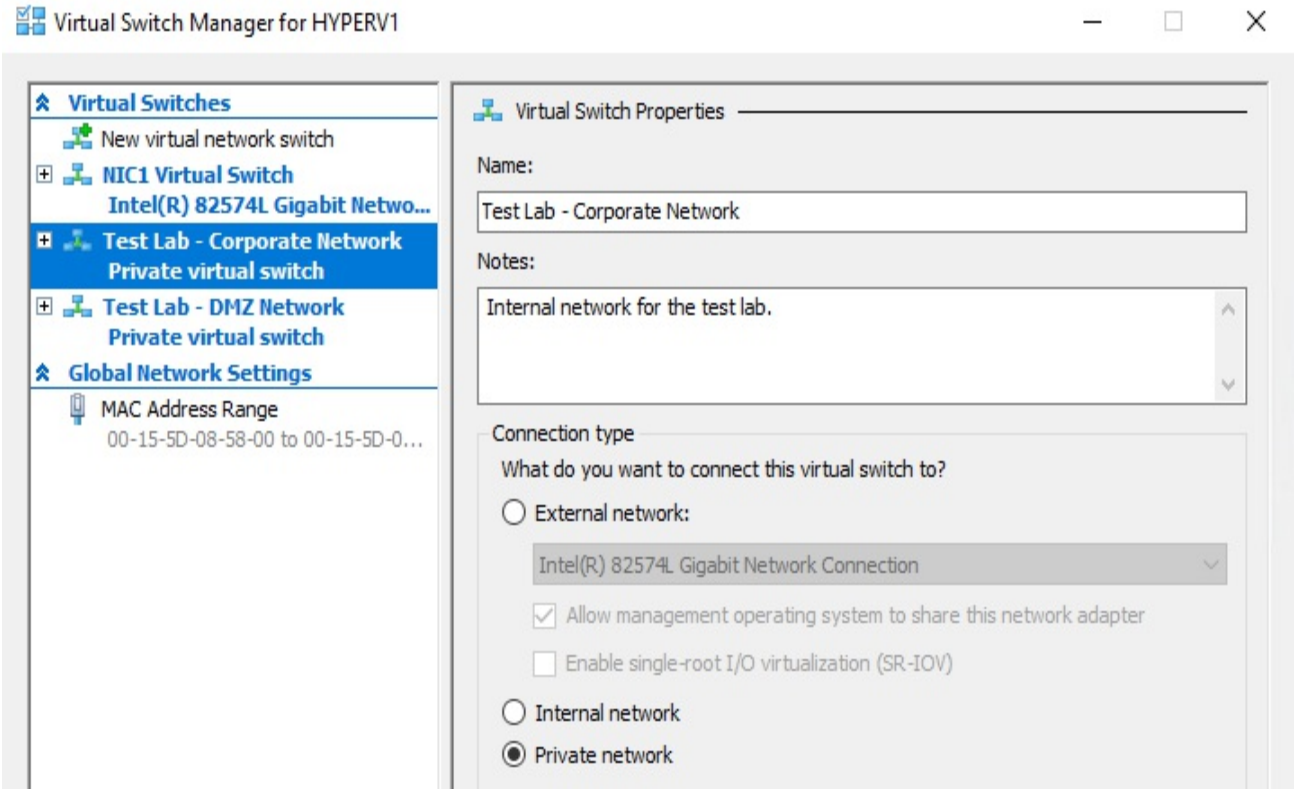
The private virtual switch

The private virtual switch is just what the name implies: private. VMs plugged into the same private virtual switch can communicate with each other, but not beyond. Even the Hyper-V host server does not have network connectivity to a private virtual switch. Test labs are a great example of a use case for private virtual switches, which we will discuss immediately following this text, when we create a new virtual switch of our own.

Creating a new virtual switch

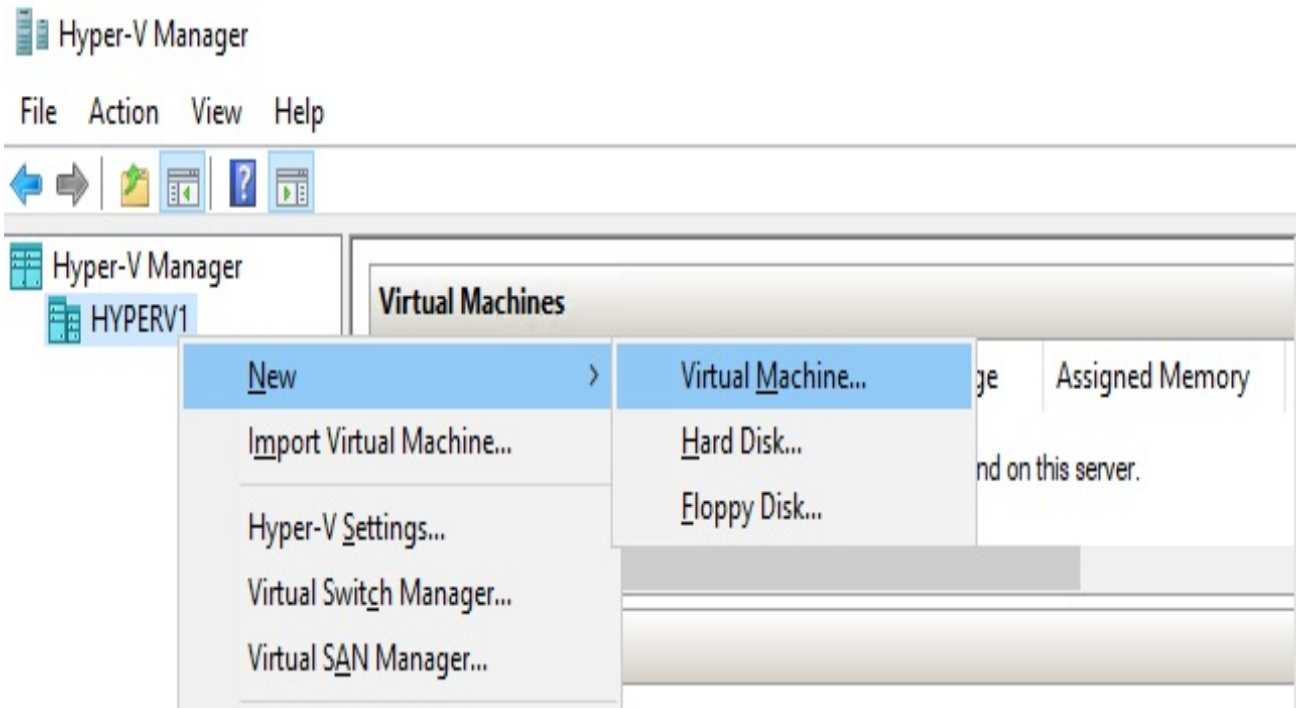
The following is an example I use often. I am running a new Hyper-V Server, which is connected physically to my corporate network, and so I can spin up new VMs, connect them to my external virtual switch, and have them communicate directly to the corp network. This allows me to domain-join them, and interact with them like I would any server on my network. Maybe I need to create some VMs that I want to be able to talk with each other, but I do not want them to be able to communicate with my production network. A good example of this scenario in the real world is when building a test lab. In fact, I am taking this exact approach for all of the servers that we have used throughout this book. My physical Hyper-V Server is on my production network, yet my entire *Contoso* network and all of the VMs running within it are on their own separate network, which is completely segregated from my real network. I did this by creating a new private virtual switch. Remember from the description that when you plug VMs into this kind of switch, they can communicate with other VMs that are plugged into that same virtual switch, but they cannot communicate beyond that switch.

Inside the Virtual Switch Manager, all I have to do is choose the kind of virtual switch that I want to create, Private in this case, and click on that Create Virtual Switch button. I can then provide a name for my new switch, and I am immediately able to connect VMs to this switch. You can see in the following screenshot that I have created two new private virtual switches: one to plug my test lab VM's internal NICs into, and another switch that will act as my test lab's DMZ network:


Virtual Switch Manager for HYPERV1
Virtual Switches
New virtual network switch
NIC1 Virtual Switch
Intel(R) 82574L Gigabit Netwo...
Test Lab - Corporate Network
Private virtual switch
Test Lab - DMZ Network
Private virtual switch
Global Network Settings
MAC Address Range
00-15-5D-08-58-00 to 00-15-5D-0...
Virtual Switch Properties
Name:
Test Lab - Corporate Network
Notes:
Internal network for the test lab.
Connection type
What do you want to connect this virtual switch to?
External network:
Intel(R) 82574L Gigabit Network Connection
Allow management operating system to share this network adapter
Enable single-root I/O virtualization (SR-IOV)
Internal network
Private network

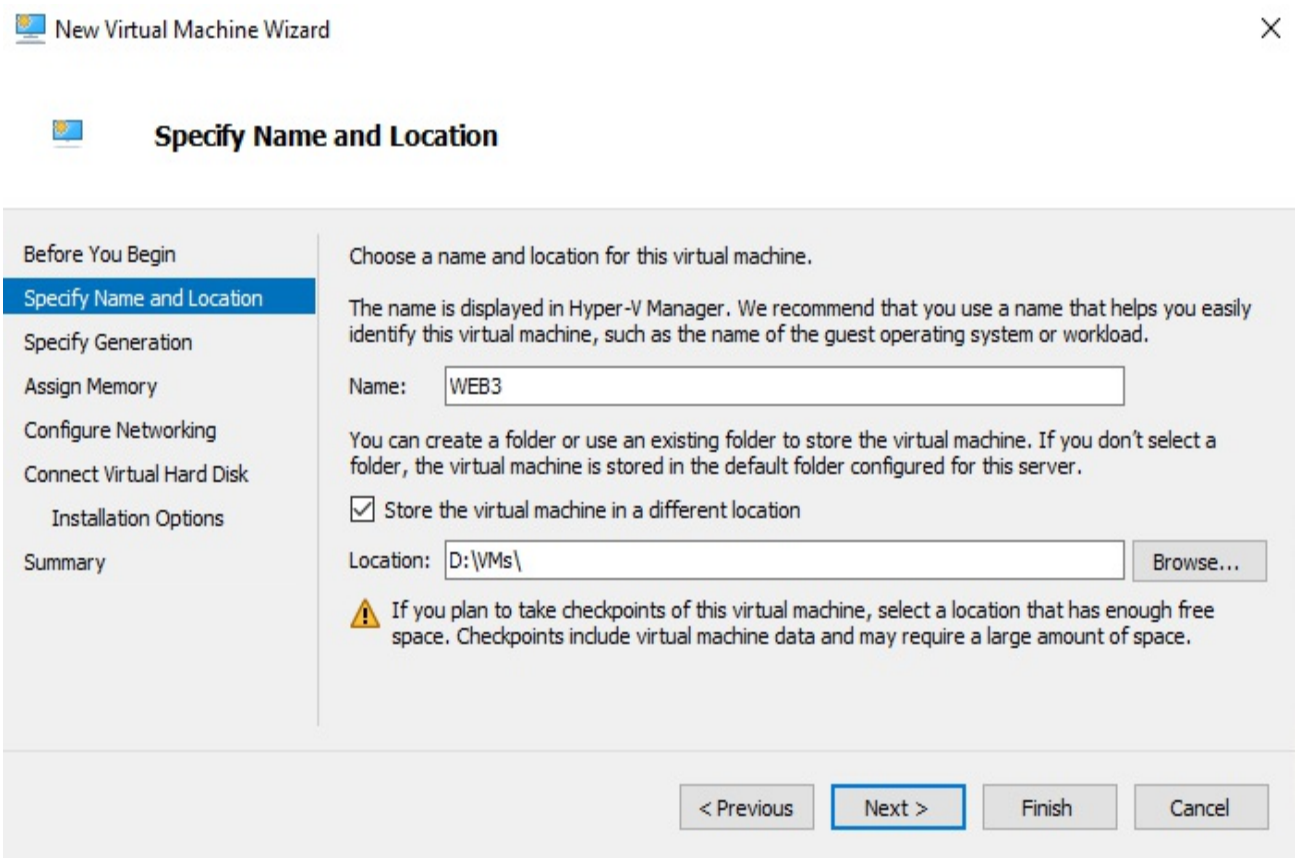
Implementing a new virtual server

Now we are ready to spin up our first virtual server! Similar to creating new virtual switches, the process for creating a new VM is fairly straightforward, but there are some steps along the way that might need some explanation if you haven't been through this process before. We start in the same management interface from which we do everything in the Hyper-V world. Open up Hyper-V Manager and right-click on the name of your Hyper-V Server. Navigate to New | Virtual Machine... to launch the wizard:



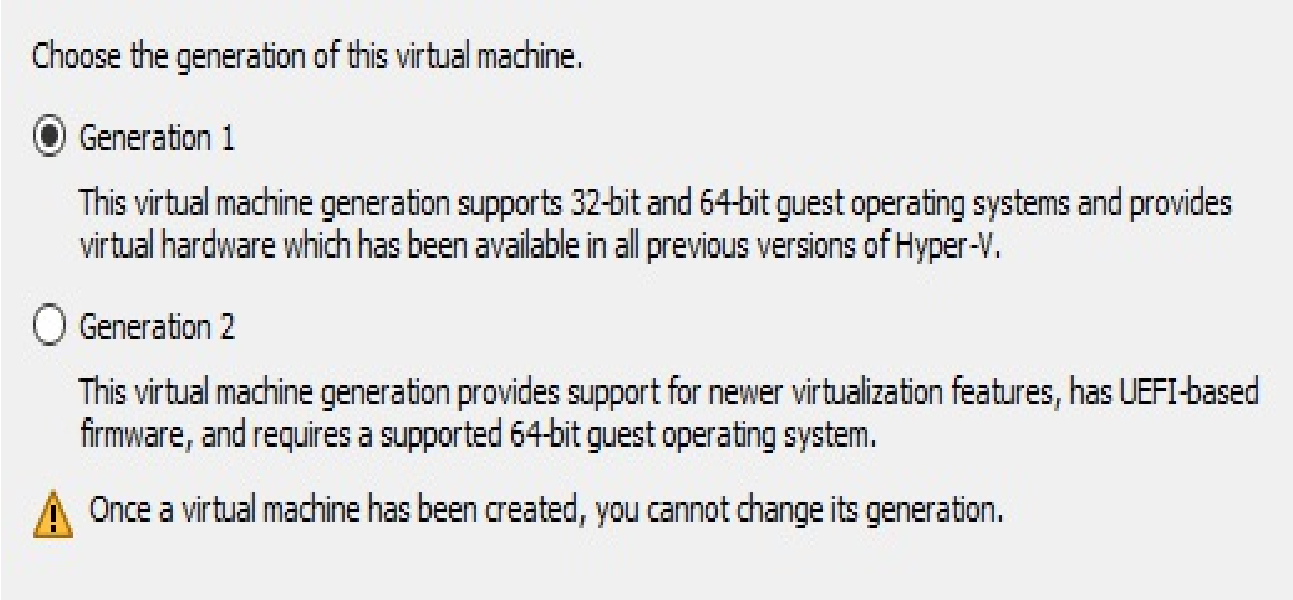
The first screen where we need to make some decisions is Specify Name and Location. Create a name for your new VM, that is easy enough. But

then you also have the chance to store your VM in a new location. If you set a good default location for your virtual machines during Hyper-V role installation, chances are that you won't have to modify this field. But in my case, I chose the default options when I installed the role, and so it was going to place my VM somewhere in `C:\ProgramData`, and I didn't like the look of that. So I selected this box, and chose a location that I like for my VM. You can see that I am using a dedicated disk to store my VMs, which generally a good practice. An even better practice in a larger network would be to utilize resilient disk space that was accessed through the network, such as a Storage Spaces Direct infrastructure:



Next, you have to decide whether you are creating a Generation 1 or Generation 2 VM. We don't need to discuss this in very much detail, because explanations of the two are clearly stated on the page and in the following screenshot. If your VM is going to be running an older

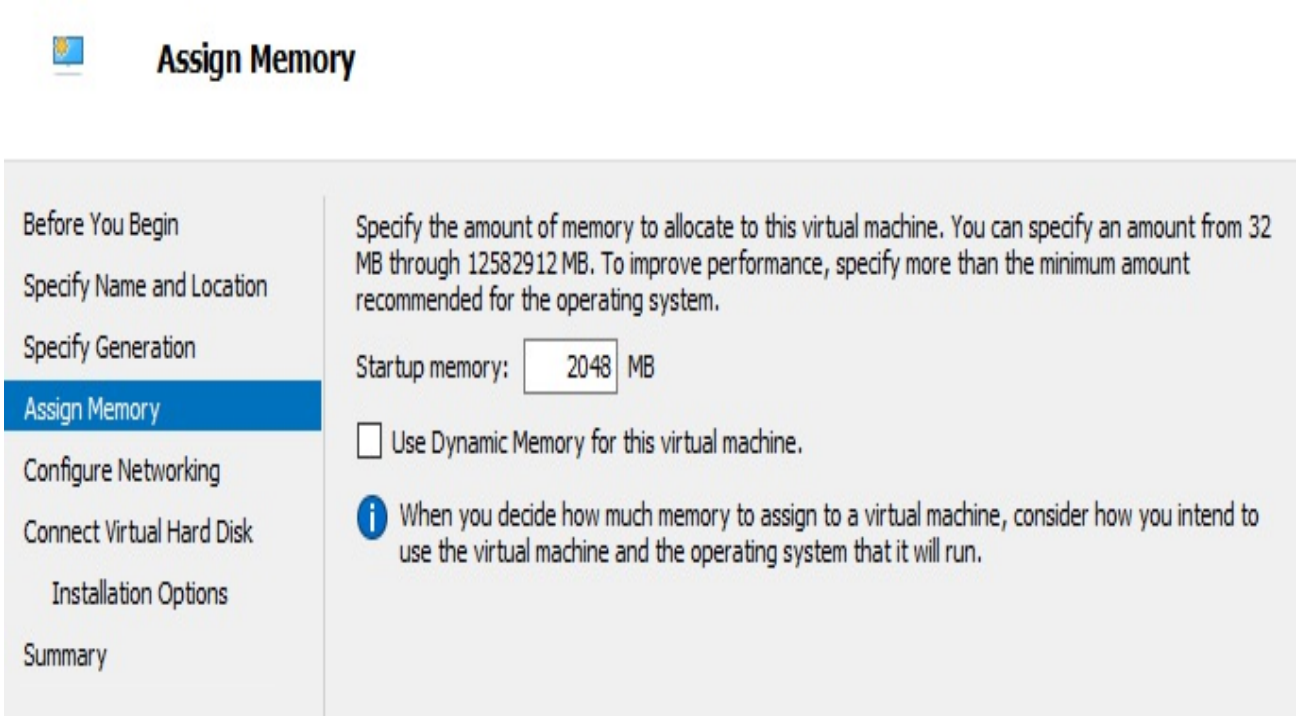
operating system, you should likely go with Generation 1 to ensure compatibility. Alternatively, if you are planning for a recent operating system to be installed on this new VM, selecting Generation 2 is probably in your best interests from a new features and security perspective:



Now, define how much memory you want to assign to this particular VM. Keep in mind that this is a setting that you can change in the future on this server, and so you don't have to plan too hard for this. The amount of RAM you dedicate to this virtual machine will depend on how much RAM you have available in the Hyper-V host system, and on how much memory is required to run whatever roles and services you plan to install on this VM. You can specify any amount of memory in this field. For example, if I wanted roughly 2 GB, I could type in around 2,000 MB. However, what I find in the field is that most people still stick with the actual amount of MB, because that is what we have always done with hardware. So instead of rounding to 2,000, I am going to set my 2 GB VM to an actual 2 GB—or 2,048 MB.

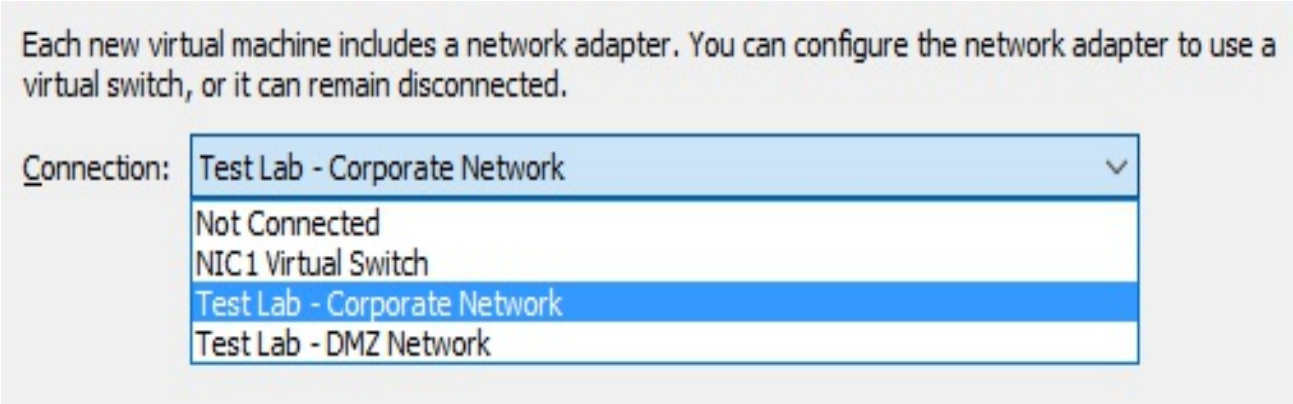
Leaving the box unchecked for dynamic memory means that Hyper-V

will dedicate an actual 2,048 MB of its physically-available RAM to this specific VM. Whether the VM is using 2,048 MB or 256 MB at any given time, the full 2,048 MB will be dedicated to the VM and will be unusable by the rest of the Hyper-V Server. If you select Use Dynamic Memory for this virtual machine, the VM only takes away from the Hyper-V host what it is actually using. If you set it to 2,048 MB, but the VM is sitting idle and only consuming 256 MB, it will only tax Hyper-V with a 256 MB load:

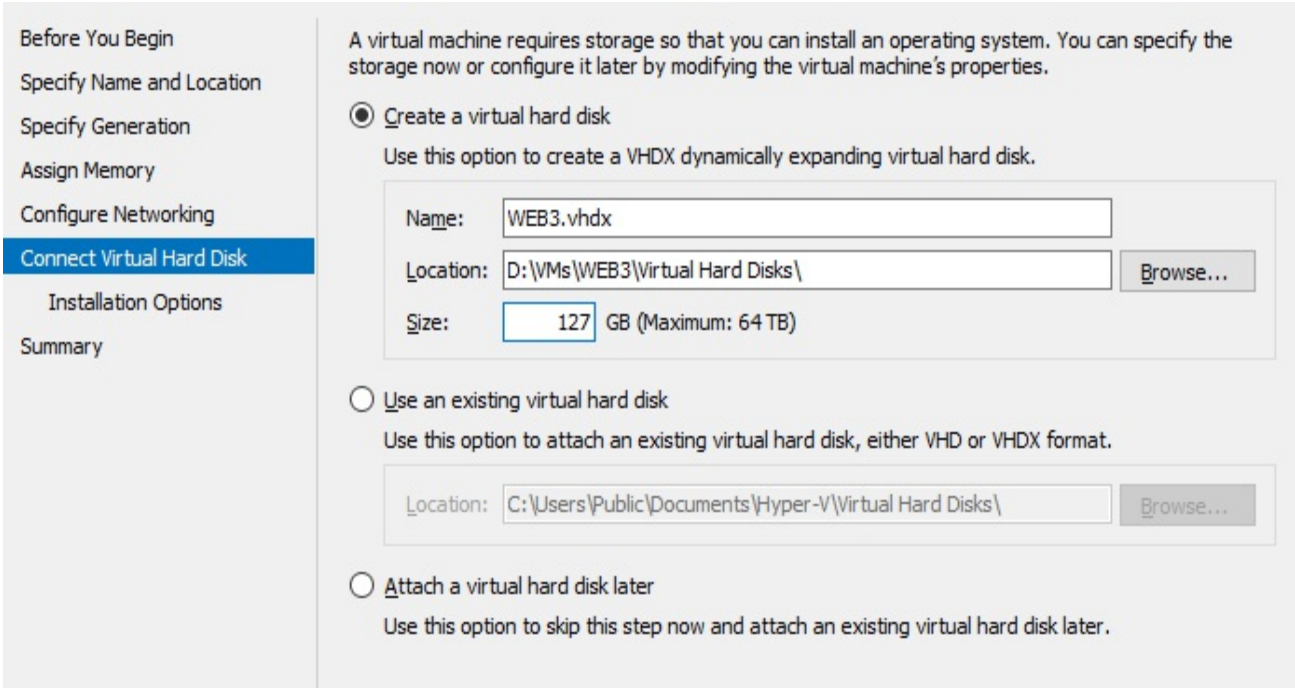


Configure Networking is the next screen we are presented with, and here we are simply choosing which virtual switch our VM's NIC gets plugged into. We do have the ability to add additional NICs to this VM later, but for now we get a standard single NIC during the creation of our new VM, and we just need to choose where it needs to be connected. For the time being, this new web server I am building will be connected to my `Test Lab` internal corporate network, so that I can build my web app and test it out, before introducing it into a real production network. If I

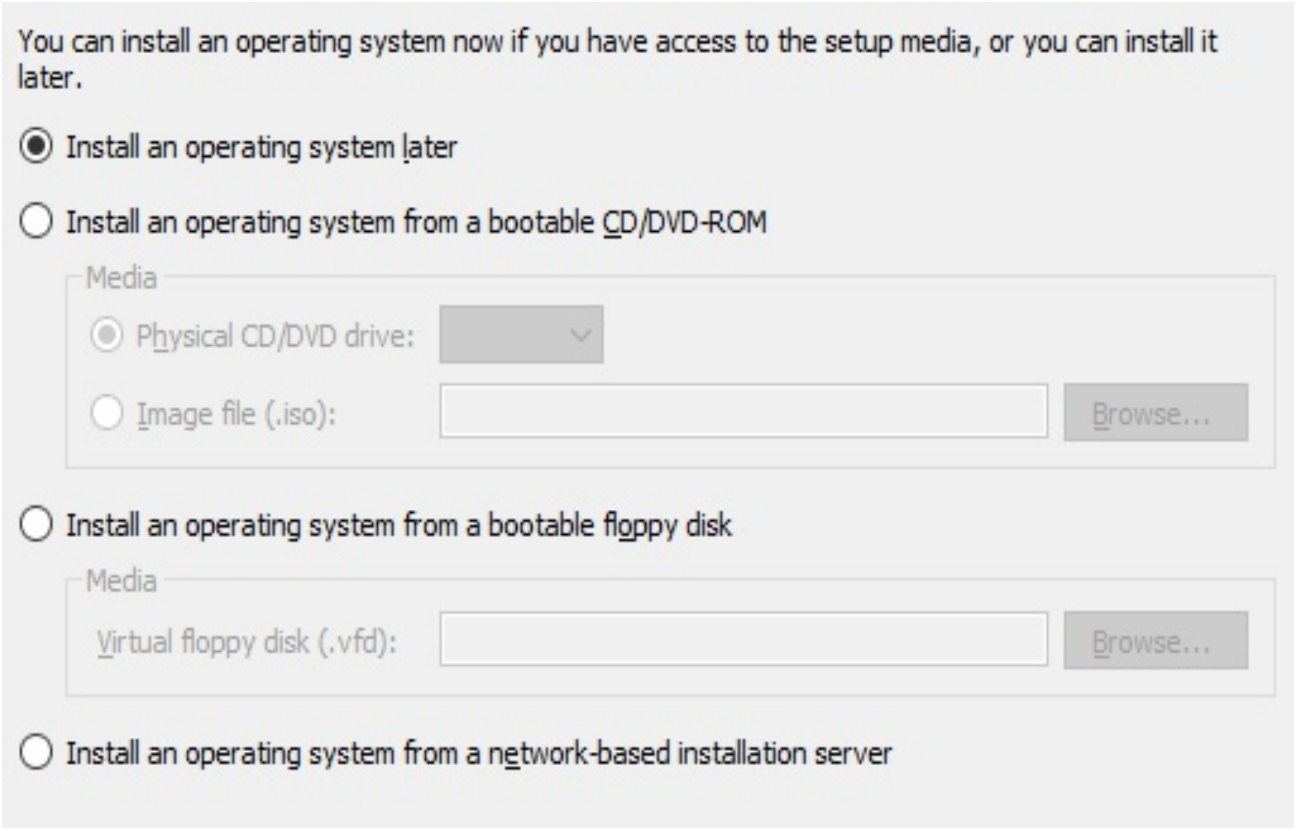
drop down a list of available connections here, you will see that my original external virtual switch, as well as the two new private virtual switches that I created, are available to choose from:



A few details are also needed so that this new VM can have a hard drive. Most commonly, you will utilize the top option here so that the new VM gets a brand new hard drive. There are also options for using an existing virtual hard disk if you are booting from an existing file, or to attach a disk later if you aren't prepared yet to make this decision. We are going to allow the wizard to generate a new virtual hard disk, and the default size is 127 GB. I can set this to whatever I want, but it is important to know that it does not consume the full 127 GB of space. The disk size will only be as big as what is actually being used on the disk, so only a fraction of that 127 GB will be used. I mention this to point out that the number you specify here is more of a *maximum* size, so make sure to plan your disks appropriately, specifying enough size so that you and your applications have the necessary room to grow:



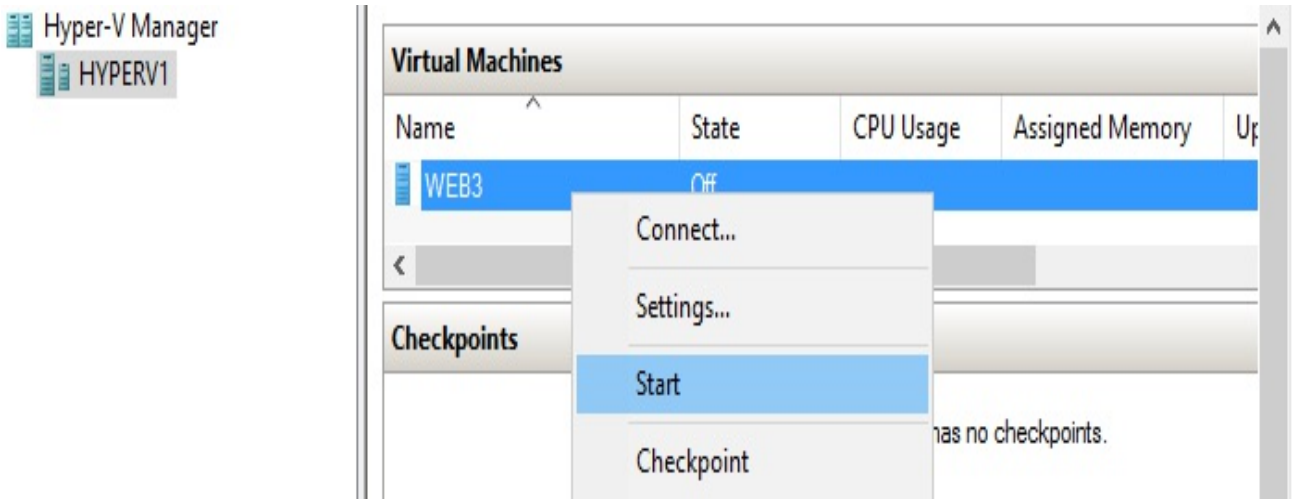
Our last screen of options in the wizard allows us to define the specifics of the operating system our new VM is going to run on. Or rather, where that operating system will be installed from. We are going to purposefully leave this set to Install an operating system later, because that is the default option, and it will give us the chance to see what happens when you do not specify any settings on this screen:



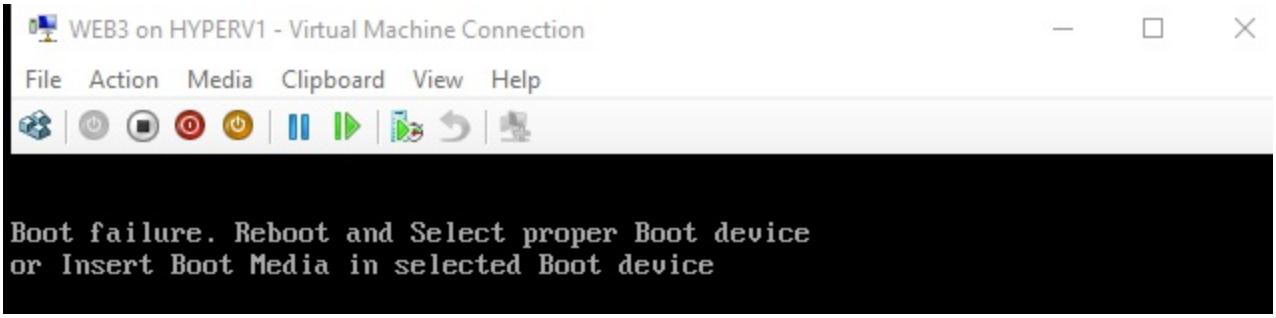
HivaNetwork.Com

Starting and connecting to the VM

We have now created a VM, which you can see inside the Hyper-V Manager console. Starting the VM is as simple as right-clicking on it, and then selecting Start. After selecting the option to start the VM, right-click on it again and click on Connect…. This will open a console window from which you can watch the boot process of your new server:



Now that our new VM has been started, *what can we expect to see inside the console window?* A boot failure error, of course:

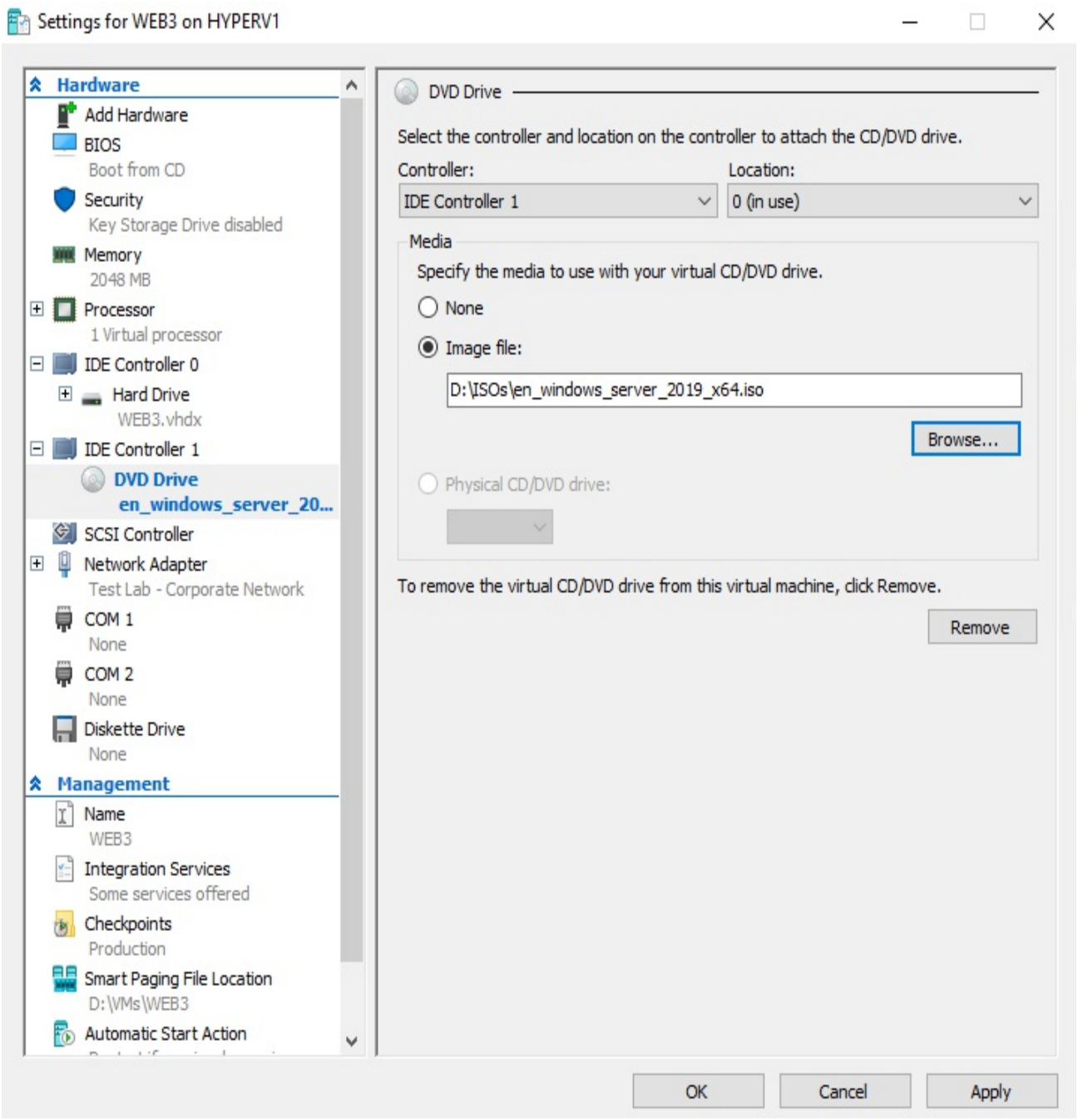


Installing the operating system

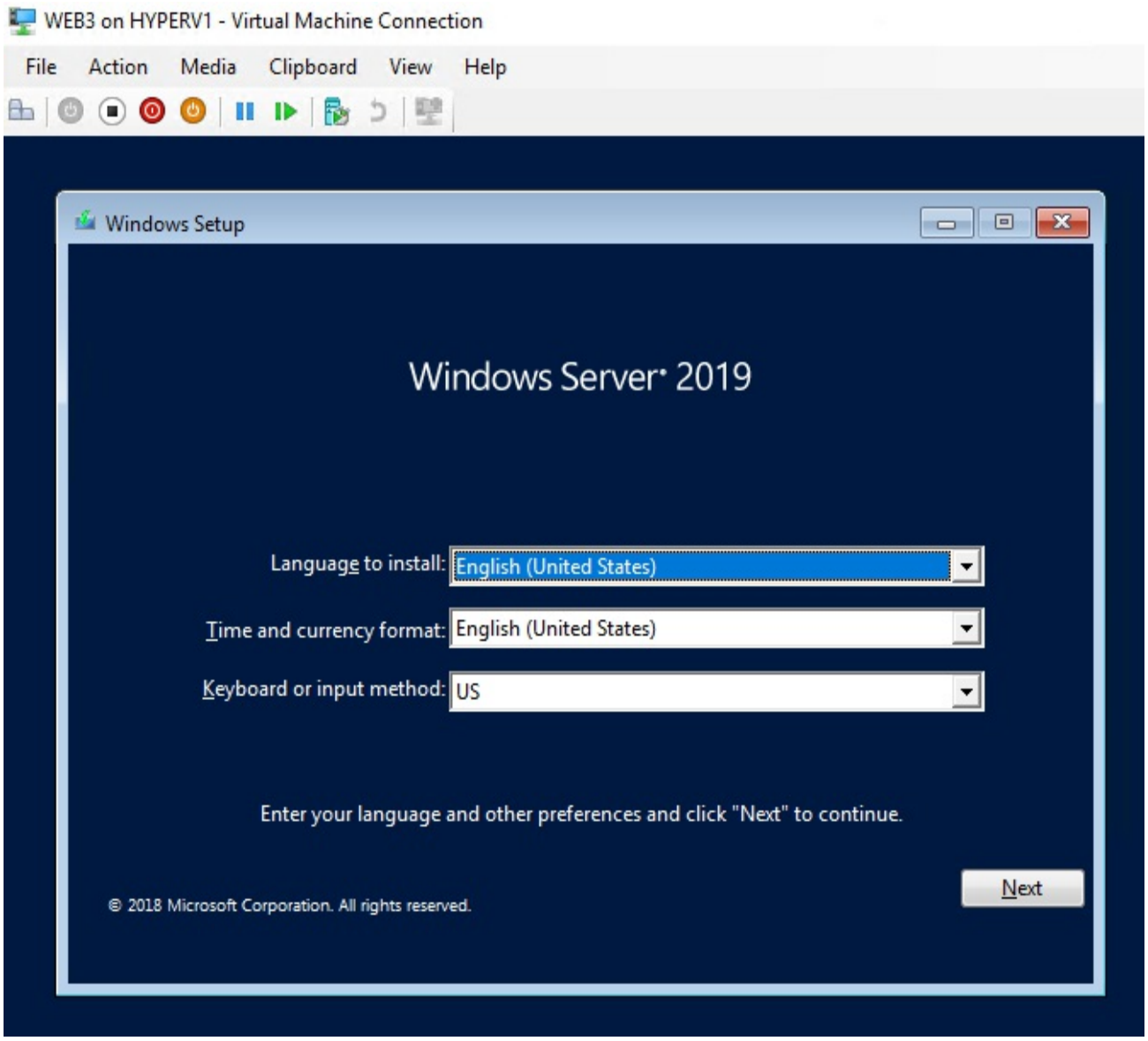
We get a boot failure message because we didn't specify any operating system media during our wizard, and so Hyper-V has created our VM and our new hard disk, but just like when you build a new server out of fresh hardware, you need software to be installed onto that hard disk in order for it to do something. Luckily, installing an operating system onto a VM is even easier than installing it onto a physical server. Heading back into Hyper-V Manager console, right-click on the name of your new VM and go to Settings...

Inside the settings, you will see that this VM has a DVD Drive automatically listed in IDE Controller 1. If you click on DVD Drive, you can easily tell it to mount any ISO to that drive. Copy the ISO file of the operating system installer you wish to run onto the hard drive of your Hyper-V Server.

I typically place all of my ISOs inside a dedicated folder called `ISOs`, right alongside my `VMs` folder, and then Browse... to it from this screen. Connecting an ISO to your VM is the same as if you were plugging a physical installation DVD into a physical server:



After mounting the media, restart the VM and you will see that our operating system installer kicks off automatically:

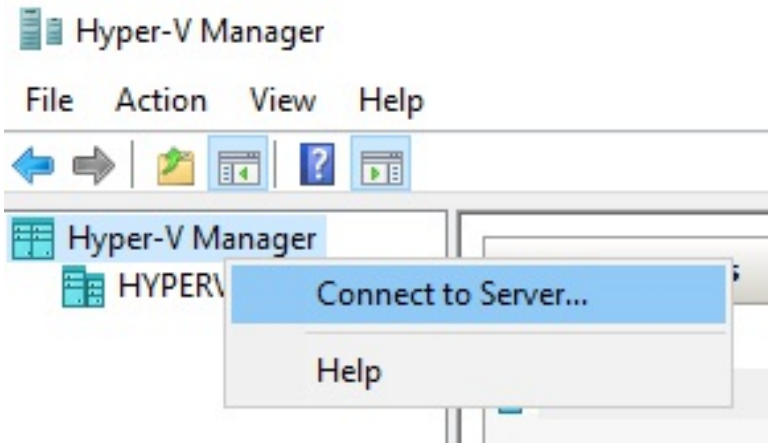

WEB3 on HYPERV1 - Virtual Machine Connection
File Action Media Clipboard View Help
Windows Setup
Windows Server 2019
Language to install: English (United States)
Time and currency format: English (United States)
Keyboard or input method: US
Enter your language and other preferences and click "Next" to continue.
© 2018 Microsoft Corporation. All rights reserved.
Next

Managing a virtual server

We have made use of Hyper-V Manager in order to manage our virtual switches, and to create a virtual machine. This tool is all-powerful when it comes to manipulating your VMs, and I find myself accessing it frequently in my daily job. Let's take a look at a few of the other things you can do from inside Hyper-V Manager, as well as discussing other methods that can be used to work with the new virtual machines that are being created on your Hyper-V Server.

Hyper-V Manager

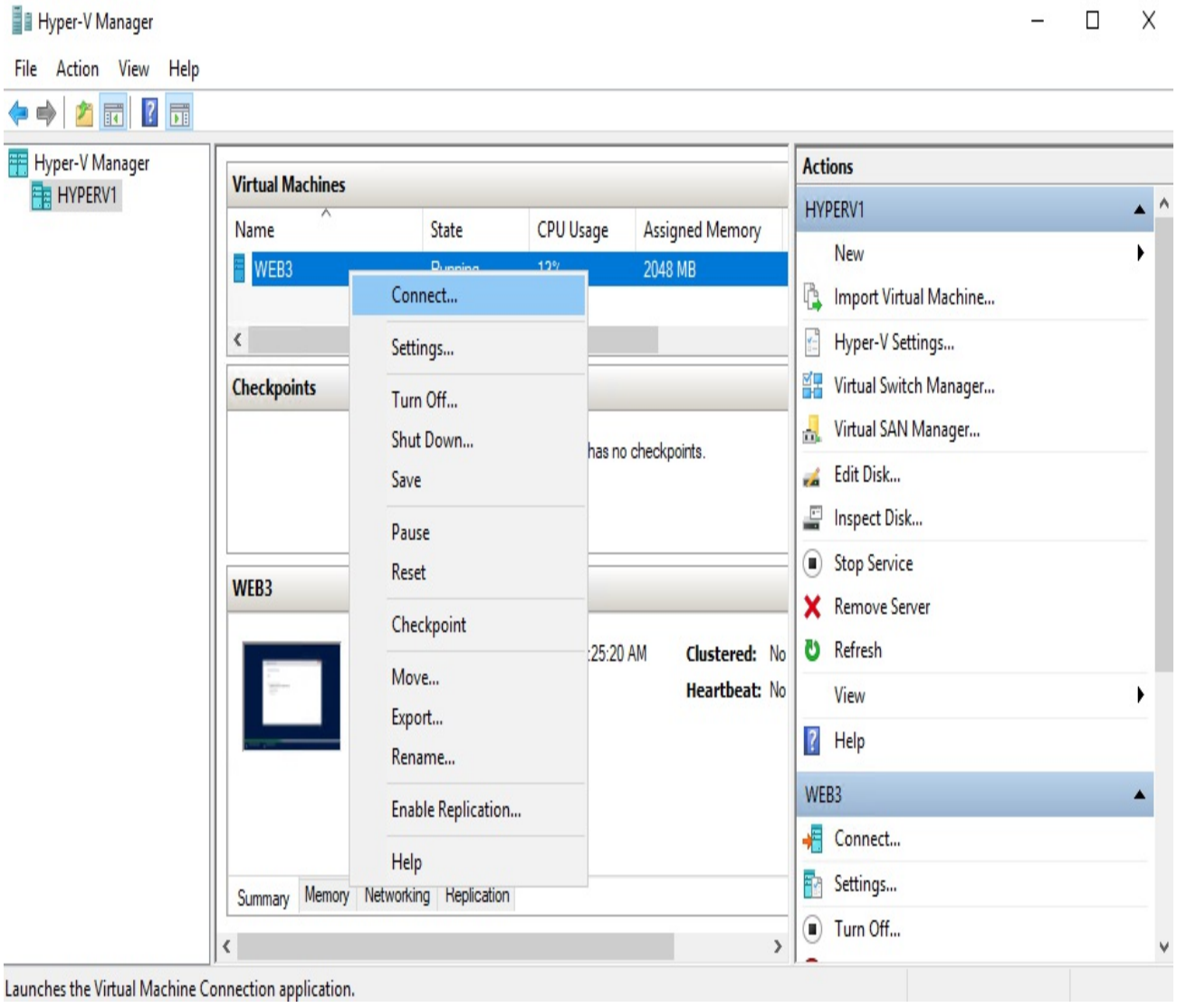
As you know, Hyper-V Manager is the primary tool for managing a Hyper-V Server. It is a nice console that gives you a quick status on your virtual machines, and allows you to manage those VMs in a variety of ways. Something we did not cover, because I only have one Hyper-V Server running, is that you can manage multiple Hyper-V Servers from a single Hyper-V Manager console. Just like any MMC-style console in the Microsoft world, you are able to right-click on the words Hyper-V Manager near the top-left corner of the screen, and select an option that says Connect to Server.... By using this function, you can pull information from other Hyper-V Servers into this same Hyper-V Manager console:



Furthermore, this enables you to run Hyper-V Manager software on a client computer. You can install the Hyper-V role onto a Windows 10 machine, which will also install this console, and then use that local copy of Hyper-V Manager running on your Window 10 desktop in order to manage your Hyper-V Servers, without needing to log in to those

servers directly.

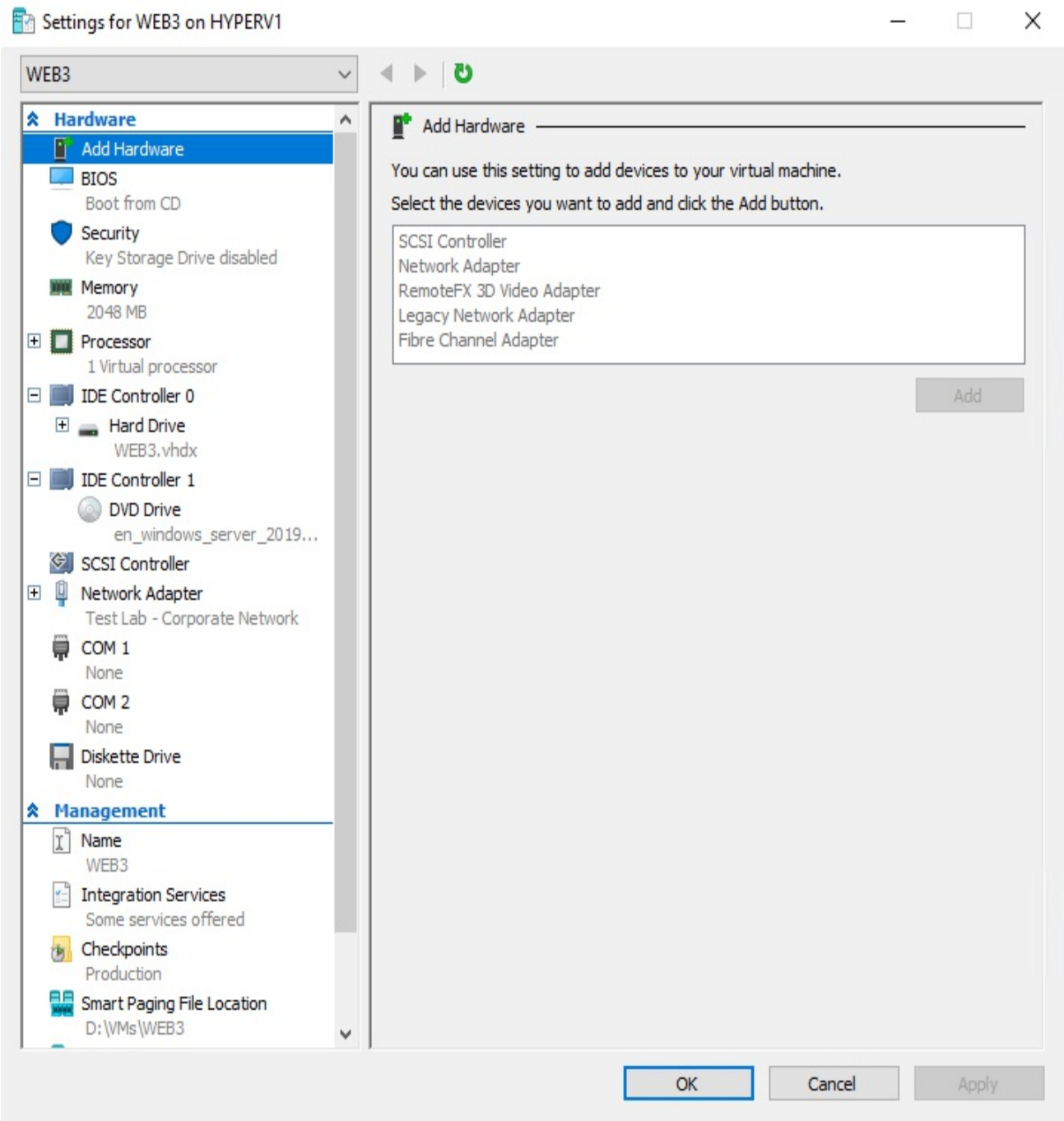
Some of the most useful actions inside Hyper-V Manager are listed along the right side of the console in the Actions pane, features such as Virtual Switch Manager and the ability to create a new VM. Once you have VMs up and running, you will find a lot of useful functions listed inside the context menu that appears when you right-click on a VM, as you can see in the following screenshot:



Some of these are self-explanatory, and some are worth playing around with. We have already used Connect... to connect to our VM's console. Settings... opens up a ton of possibilities, and we will take a look further inside the **Settings** menu immediately following this text. One of the most common reasons I open up this right-click menu is for power functions on my VMs. You can see that you have the ability to Turn Off... or Shut Down... your VM. Turning it off is like pressing the power button on a server: it cuts off power immediately to that server and will cause Windows some grief when doing so. The shutdown function, on the other hand, initiates a clean shutdown, at least when you are using Microsoft operating systems on the VMs. Shutting down a server is no big deal, but the real power here comes from the fact that you can shut down multiple VMs at the same time. For example, if I were running a dozen different VMs all for my test labs, and I decided that my lab was taking up too many resources and causing problems on my Hyper-V Server, I could select all of my VMs at the same time, right-click on them, then click on Shut Down... just once, and it would immediately kick off the shutdown process on all of the VMs that I had selected. Once a VM is shut down or turned off, right-clicking on that VM will give you a Start function; you can also select many servers and start them all at once by using this right-click menu.

The Settings menu

Making in-depth modifications to any of your VMs typically means right-clicking on that VM, and then navigating to Settings... for that particular VM. Inside settings, you can adjust any aspect of your VM's hardware, which is the most common reason to visit this screen. Immediately upon opening the settings, you have the option to Add Hardware to your VM. This is the place you would go in order to add more hard drive controllers or NICs to your virtual server:

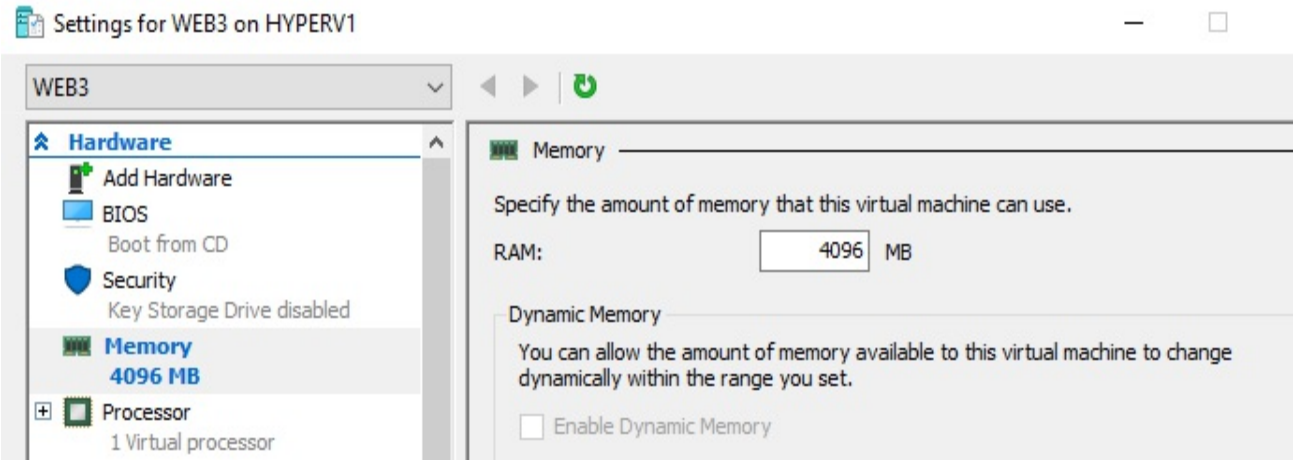


I don't know if you can tell from the preceding screenshot, but the Add button is currently grayed out. This is important. Many functions inside settings can be manipulated on-the-fly, while the VM is running. Some functions cannot be accomplished unless the VM is turned off. Adding

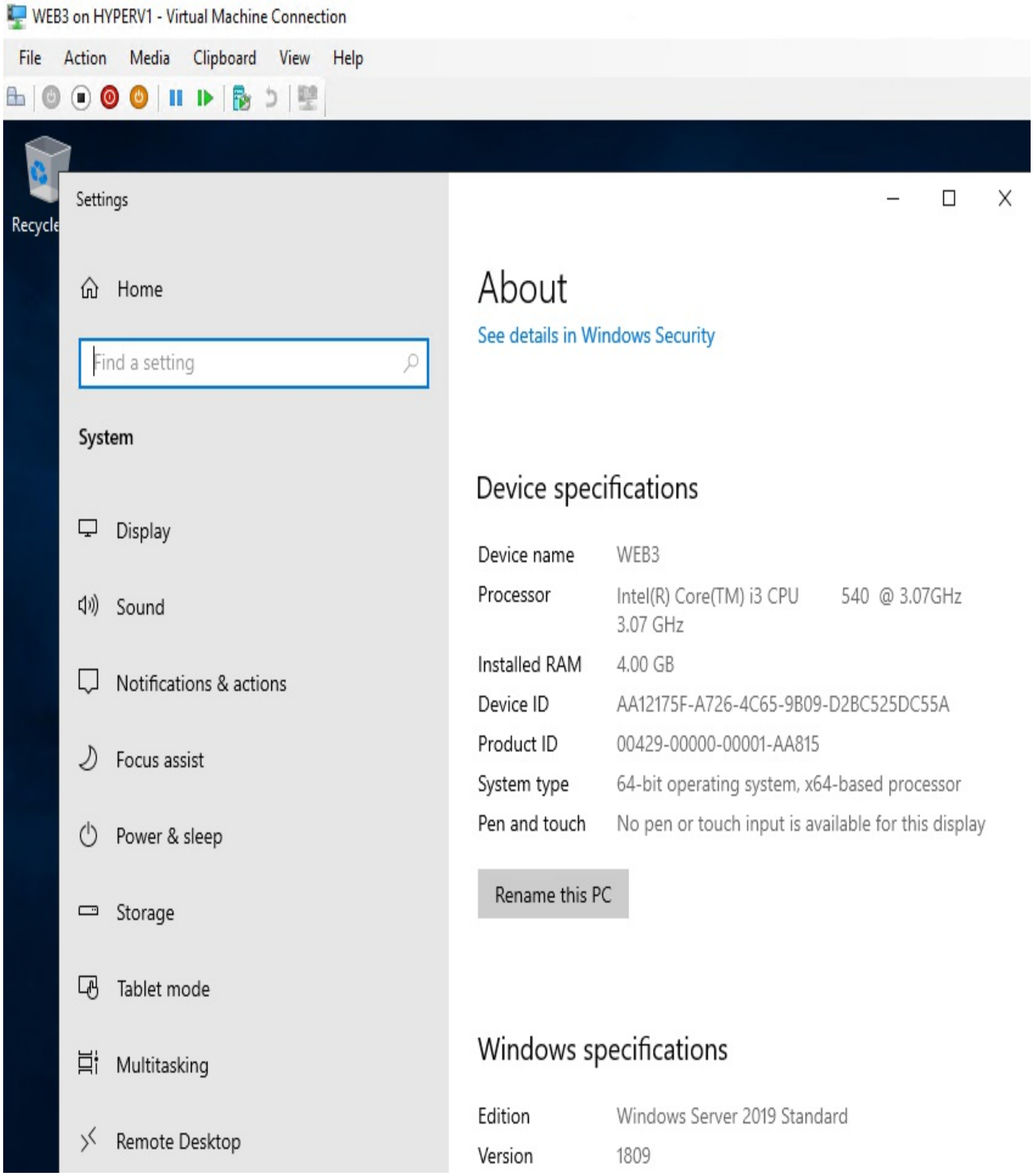
hardware is one of those functions. If you want to add a new hard drive or NIC to your VM, you will need to shut down that server before you are able to do it.

Next, we should talk about the Memory screen. This one is fairly simple, *right?* Just input the amount of RAM that you want this VM to have available. The reason that I want to point it out is that a major improvement has been made in this functionality. Starting with Windows Server 2016 Hyper-V, you can now adjust the amount of RAM that a VM has while it is running! In the previous versions of Hyper-V, you were required to shut down the VMs in order to change their memory allocation, but even though my `WEB3` server is currently running and servicing users, I can pop in here and increase RAM at will.

Let's say my 2 GB isn't keeping up with the task load, and I want to increase it to 4 GB. I leave the server running, open up the Hyper-V Manager settings for the VM, and adjust the relevant setting to 4,096 MB:



The amount of memory immediately adjusts, and if I open up system properties inside the `WEB3` server, I can see that the operating system has updated to reflect the 4 GB of RAM now installed:



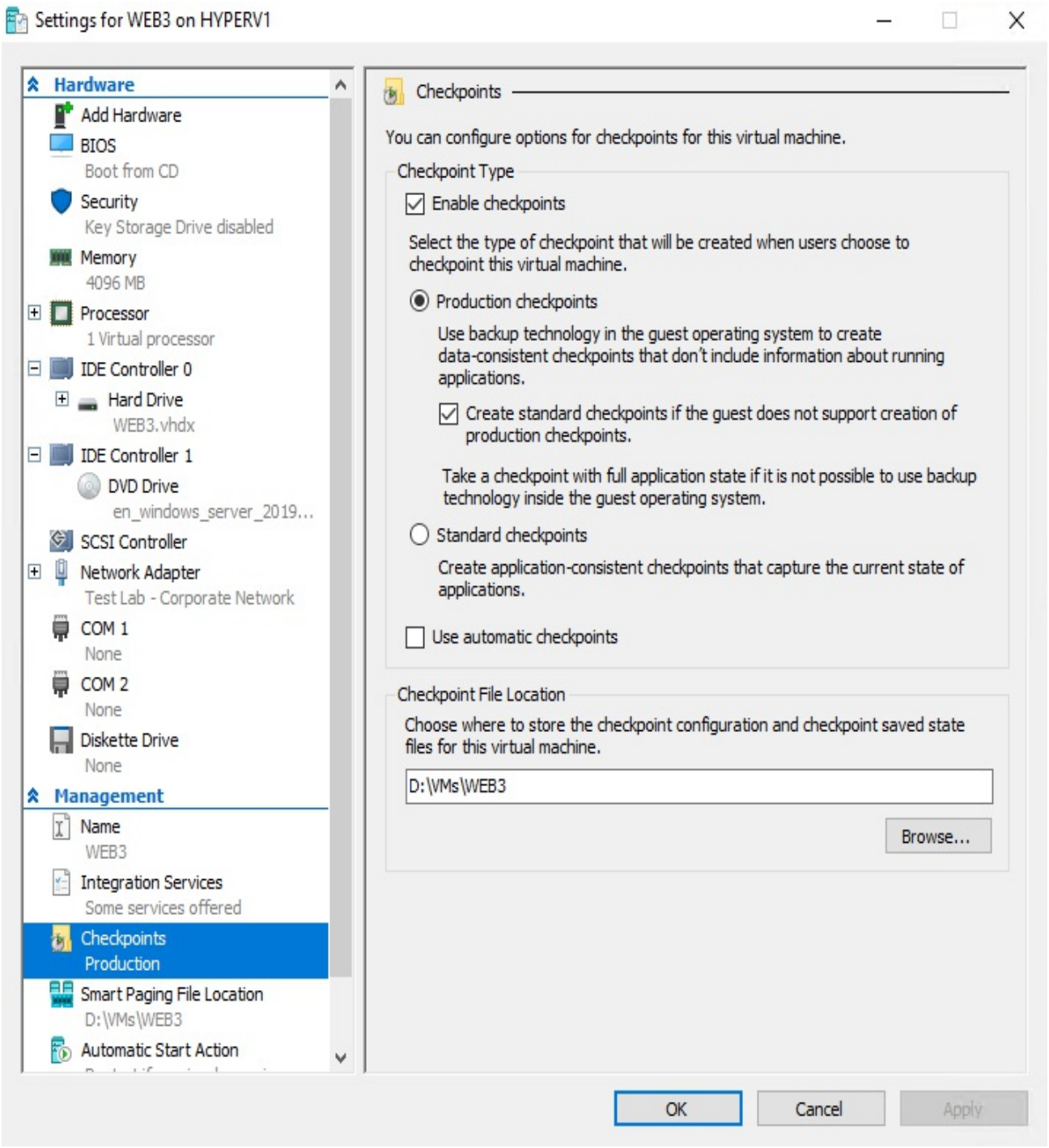
The other useful settings screens are the Processor and Network Adapter sections. You have the ability to define the number of virtual

processors currently assigned to the VM, and performance weights associated with these processors. In the Network Adapter screen, you can change which virtual switch your virtual NICs are plugged into. I find myself accessing this section often as I move servers from one location to another.

Checkpoints

The last part of the Settings menu that I want to discuss is called **checkpoints**. These were formerly called snapshots, which I think makes a little more sense to most of us. Checkpoints are a function that you can invoke from Hyper-V Manager by right-clicking on one or more VMs. It essentially creates a *snapshot in time* for the VM. Another way to look at checkpoints is that they are creating rollback points for your servers. If you create a checkpoint on Tuesday, and on Wednesday somebody makes a configuration change on that server that causes problems, you can restore the checkpoint from Tuesday and bring the VM back to that day's status.

There are a couple of different ways that checkpoints can be run, and the Settings menu is where we define those particulars. Simply right-click on any VM, visit the Settings screen, and then click on the management task called Checkpoints. You can see the options in the following screenshot:



These settings are individual for each VM you are running; you could treat checkpoints for `WEB1` differently than `WEB2`, for example. The default way to handle these snapshots in time is called Production checkpoints.

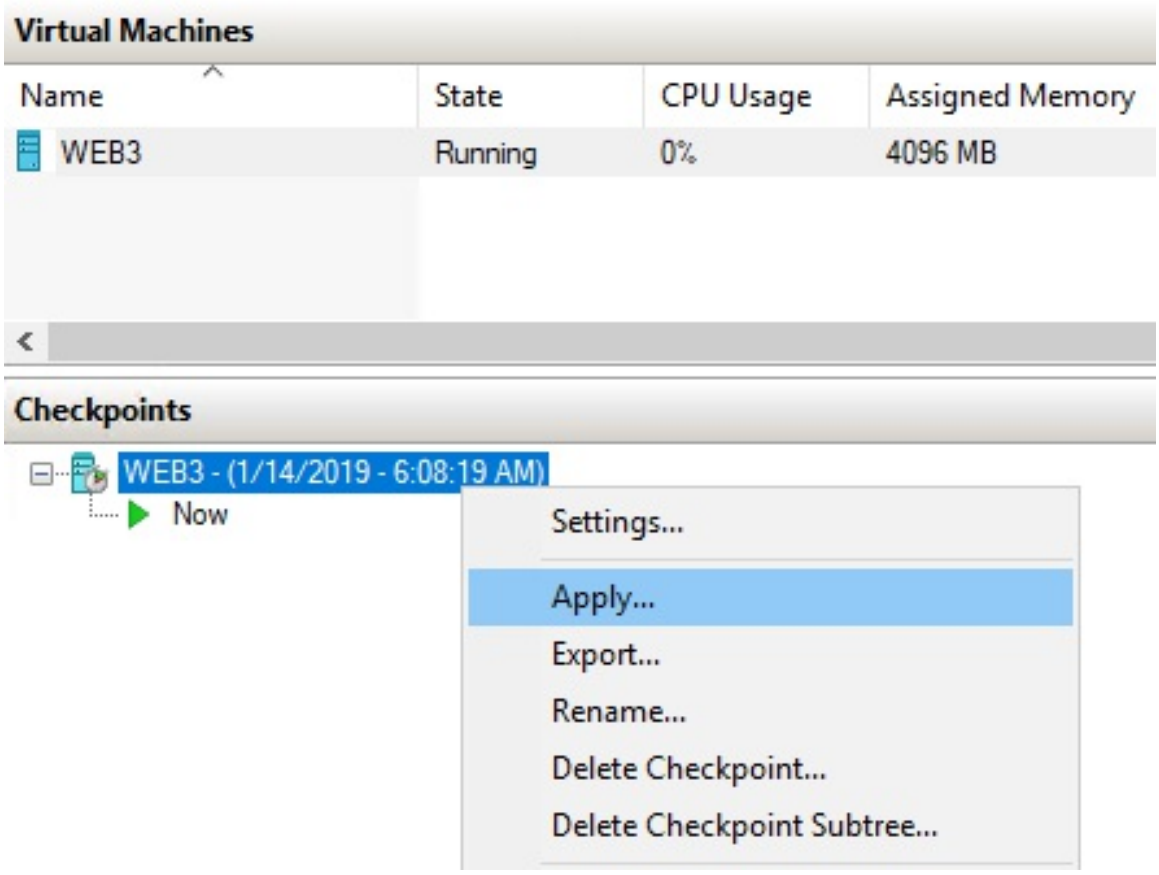
This is generally the preferred method for creating these quick images of your servers, as it is the cleanest method. When choosing to generate a production checkpoint, Hyper-V invokes Windows backup functions inside the VM's own operating system, in order to create a backup of that server. This would be similar to you logging into that VM and manually launching an operating system backup task. Keep in mind that, when you do this, and therefore when Hyper-V does this for you, it is not a block-by-block identical backup of the VM as it is running at this point in time, but rather a backup file that can then be restored in the future to bring the operating system files back to this point in time. In other words, a production checkpoint brings Windows back to the previous status, but any applications and data that are constantly changing on the server are not captured.

Alternatively, the Standard checkpoints option does just that. This takes more of a quick-and-dirty capture of the VM, kind of like just right-clicking on the VHDX hard drive file and choosing to copy and then paste it somewhere else. Restoring standard checkpoints can be a messier process, because if your checkpoint was created while an application on the server was in the middle of an important function, the restore would bring it right back to that application being in the middle of the same important function. For something such as a database write, that could get complicated.

Once you have made the decision on which kind of checkpoint is best for your VM, invoking checkpoints is very simple. Back at the main screen in Hyper-V Manager, simply right-click on your VM and select Checkpoint. After performing this task, you will see the middle pane of Hyper-V Manager receive some new information in a section you may not have even noticed earlier: Checkpoints.

The new checkpoint that we just created is now sitting here, waiting to be restored if the need arises. In the future, right-clicking on this

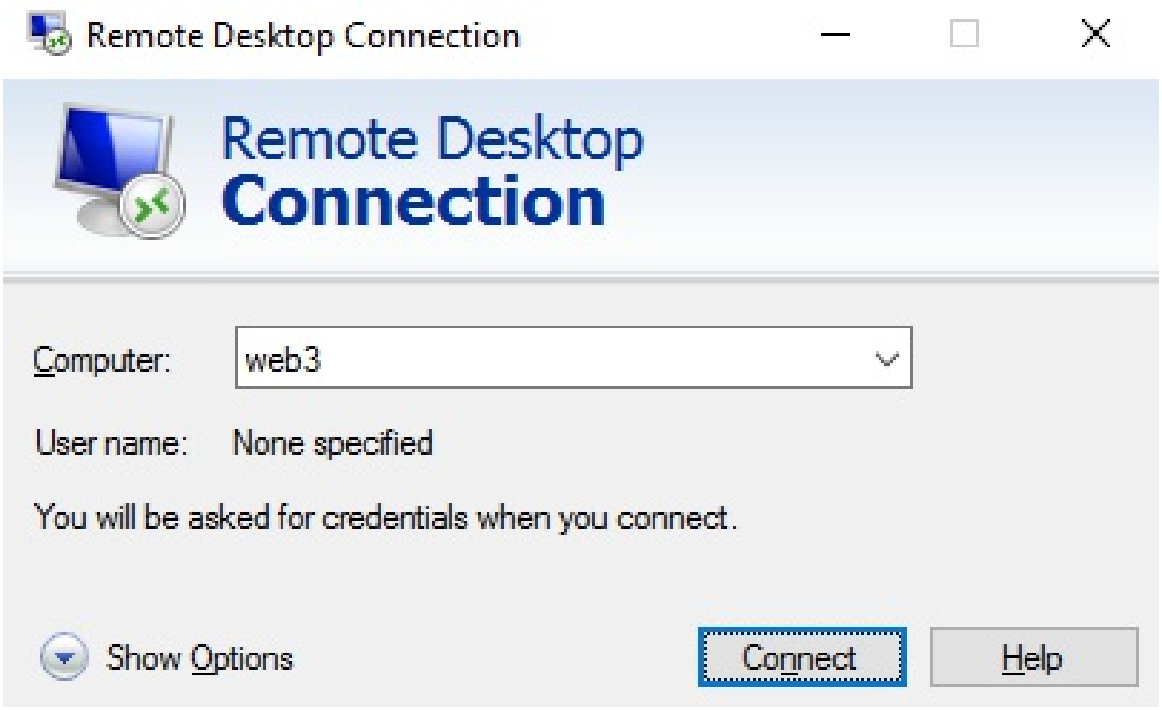
checkpoint and choosing Apply... will initiate the restoration process:



Hyper-V Console, Remote Desktop Protocol (RDP), or PowerShell

While hardware adjustments to VMs need to be made through Hyper-V Manager, your daily interaction with these VMs running as servers in your environment does not necessarily mean you have to log in to your Hyper-V Server. If you happen to be inside Hyper-V Manager anyway, you can quickly and easily use that Connect function to interact with the console of your servers, through the use of the Hyper-V Console tool. Accessing your servers this way is beneficial if you need to see something in BIOS, or otherwise outside of Windows operating system that is running on that VM, but it's not often that you require this level of console access.

When you have Windows Servers running as VMs, it is much more common to interact with these servers in the same ways that you would interact with physical servers on your network. While I have been accessing my `WEB3` server through the Hyper-V Console in this chapter, now that I have Windows Server 2019 installed onto `WEB3` and I have enabled the RDP capabilities on it, there is no reason why I couldn't just pop open `MSTSC` and log into `WEB3` that way, straight from my desktop:



The same is true for PowerShell or any other traditional way of remotely accessing services on any other server. Since this VM is fully online and has the server operating system installed, I can use PowerShell remoting to manipulate my `WEB3` server as well, from another server or from my desktop computer. Once you are finished building out the hardware and installing the operating system on a VM, it's rare that you actually need to use the Hyper-V Console in order to interact with that server. The primary reasons for opening up Hyper-V Manager to reach a VM are to make hardware-level changes on that server, such as adding a hard drive, adjusting RAM, or moving a network connection from one switch to another.

Windows Admin Center (WAC)

We have seen WAC scattered throughout this book, and for good reason. WAC is the brand new super-tool that Microsoft wants server administrators to start using in order to interact with and manage almost every single one of their servers. VM servers hosted in Hyper-V are no exception; you can make use of the WAC toolset in order to administer servers running on your Hyper-V hosts, and use WAC to manage the host servers themselves.

Shielded VMs

If your day job doesn't include work with Hyper-V, it's possible that you have never heard of shielded VMs. The name does a pretty good job of explaining this technology at a basic level. If a VM is a virtual machine, then a shielded VM must be a virtual machine that is *shielded* or protected in some way, *right?*

A shielded VM is essentially a VM that is encrypted. Rather, the hard drive file itself (the VHDX) is encrypted, using BitLocker. It sounds simple, but there are some decent requirements for making this happen. In order for the BitLocker encryption to work properly, the VM is injected with a virtual **Trusted Platform Module** (**TPM**) chip. TPMs are quickly becoming commonplace at a hardware level, but actually using them is still a mysterious black box to most administrators. Shielded VMs can also be locked down so that they can only run on healthy and approved host servers, which is an amazing advantage to the security-conscious among us. This capability is provided by a couple different attestation options, which we will discuss shortly.

In order to explain the benefits that shielded VMs bring to the table, we are going to look at an example of what happens when VMs are *not* shielded. Keep in mind that the idea of shielded VMs is quite a bit more important when you think in the context of servers being hosted in the cloud where you don't have any access to the backend, or hosted by some other division inside your company, such as inside a private cloud. Unless you have already taken the time to roll out all shielded VMs in your environment, what I am about to show you is currently possible on *any* of your existing VMs.

You already know that I am running a Hyper-V host server and on that host I have a virtual machine called WEB3. Now, let's pretend that I am a cloud-hosting provider, and that WEB3 is a web server that belongs to one of my tenants. I have provided my tenant with a private virtual switch for networking, so that they can manage the networking of that server and I don't have access to that VM at the networking level. Also, it is a fact that this WEB3 server is joined to my tenant's domain and network, and I as the cloud host have absolutely no access to domain credentials, or any other means that I can utilize to actually log in to that server.

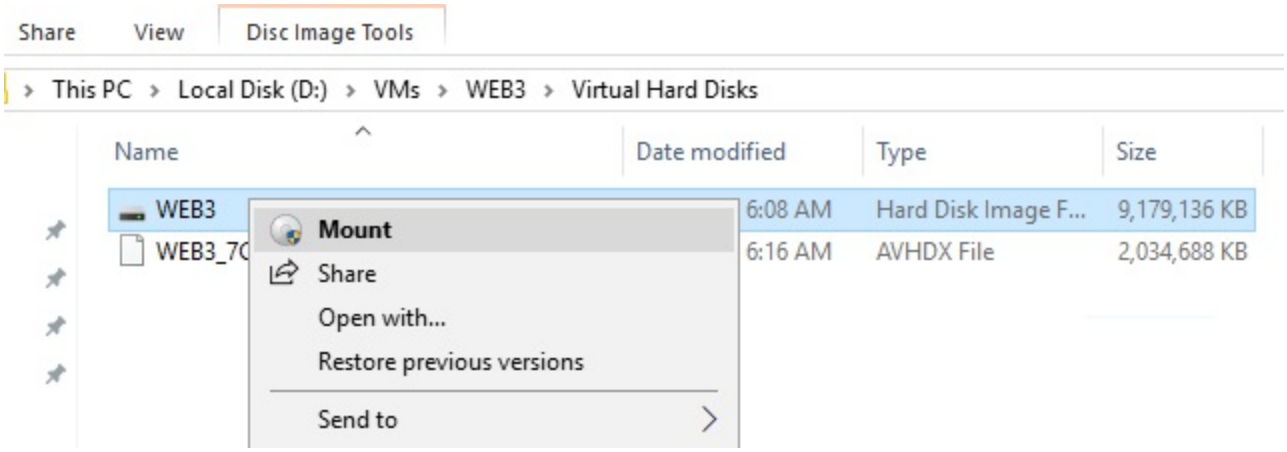
Sounds pretty good so far, *right?* You, as a tenant, certainly wouldn't want your cloud provider to be able to snoop around inside your virtual machines that are being hosted in that cloud. You also wouldn't want any other tenants who might have VMs running on the same cloud host to be able to see your servers in any way. This same mentality holds true in private clouds as well. If you are hosting a private cloud and are allowing various companies or divisions of a company to have segregated VMs running in the same fabric, you would want to ensure those divisions had real security layers between the VMs, and between the VMs and the host.

Now, let's have a little fun and turn into a villain. I am a rogue cloud-host employee, and I decide that I'm going to do some damage before I walk out the door. It would be easy for me to kill off that WEB3 server completely, since I have access to the host administrative console. However, that would probably throw a flag somewhere and the tenant would just spin up a new web server, or restore it from a backup. So even better than breaking the VM, I'm going to leave it running and then change the content of the website itself. Let's give this company's clients something to talk about!

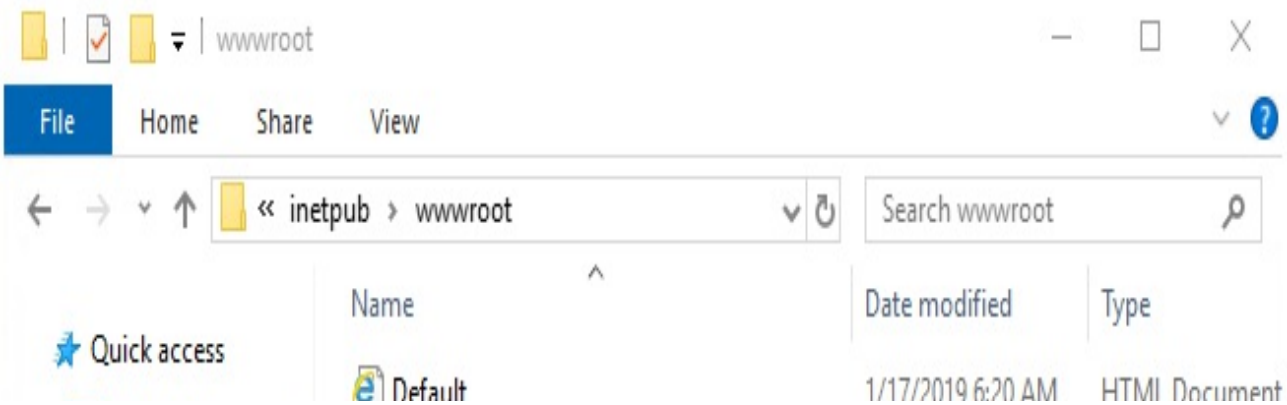
To manipulate my tenant's website running on WEB3, I don't need any real access to the VM itself, because I have direct access to the virtual hard

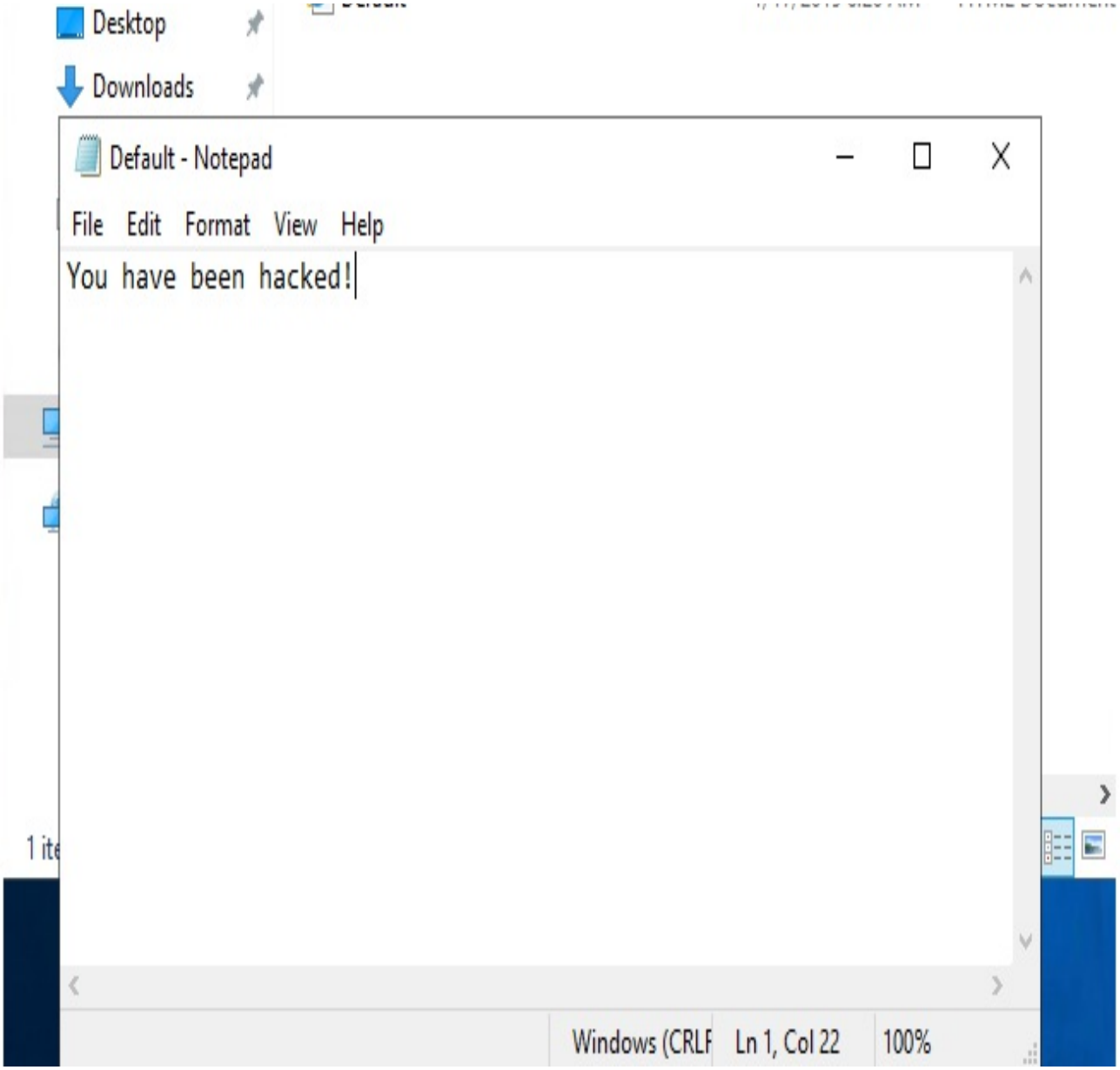
drive file. All I need to do is tap into that VHD file, modify the website, and I can make the website display whatever information I want.

First, I log into the Hyper-V Server (remember, this is owned by me since I am the host), and browse to the location of the VHD file that `WEB3` is using. This is all on the backend, so I don't need any tenant credentials to get here. Furthermore, nothing is logged with these actions and the tenant will have no way of knowing that I am doing this. I simply right-click on that VHD and select Mount:

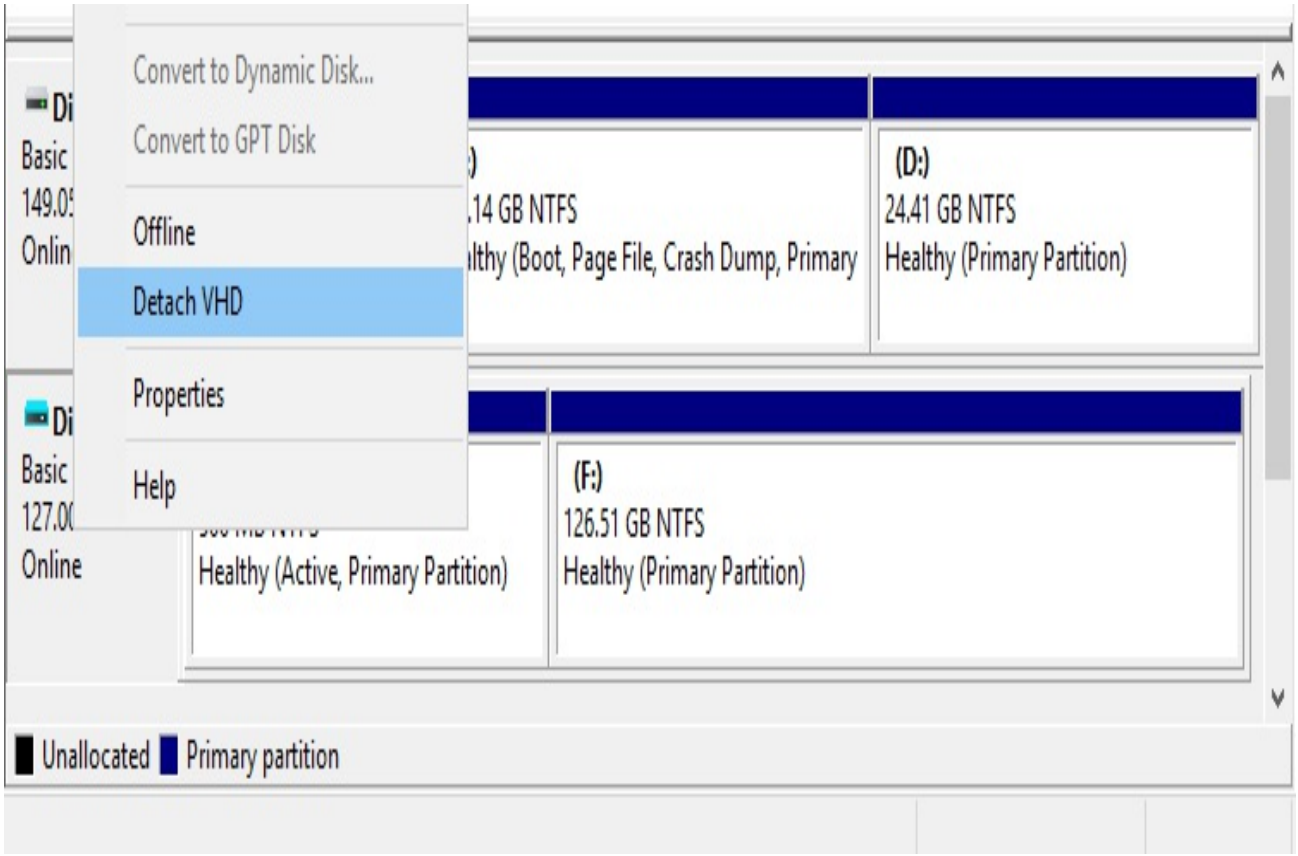


Now that the VHD has been mounted to the host server's operating system directly, I can browse that VM's hard drive as if it were one of my own drives. Navigate to the `wwwroot` folder in order to find the website files, and change the default page to display whatever you want:





When I'm finished playing around with the website, I can open up Disk Management, right-click on that mounted disk, and select Detach VHD to cover my tracks:

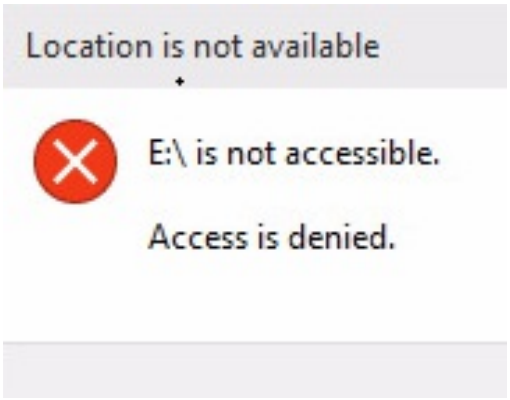


And then, just for the fun of it, I copy the entire VHD file onto a USB so that I can take it with me and mess around with it more later.

How do you feel about hosting virtual machines in the cloud now? This example cuts to the core of why so many companies are scared to take that initial step into cloud hosting—there is an unknown level of security for those environments. Thankfully, Microsoft is taking steps to alleviate this security loophole with a new technology called **shielded VMs**.

Encrypting VHDs

The idea behind shielded VMs is quite simple. Microsoft already has a great drive-encryption technology, called BitLocker. Shielded VMs are Hyper-V VMs that have BitLocker drive encryption enabled. When your entire VHD file is protected and encrypted with BitLocker, nobody is going to be able to gain backdoor access to that drive. Attempting to mount the VHD as we just did would result in an error message, and nothing more:



Even better is that; when you set up your infrastructure to support shielded VMs, you also block Hyper-V Console access to the VMs that are shielded. While this in itself isn't as big a deal as drive encryption, it's still important enough to point out. If someone has access to the Hyper-V host server and opens up Hyper-V Manager, they will generally have the ability to use the Connect function on the tenant VMs in order to view whatever was currently on the console. More than likely, this would leave them staring at a login screen that they, hopefully, would not be able to breach. But if that VM's console had somehow been left in a logged-in state, they would have immediate access to manipulating the

VM, even if the drive was encrypted. So when you create a shielded VM, it not only encrypts the VHD using BitLocker technology, it also blocks all access to the VM's console from Hyper-V Manager.

Does this hardcore blocking have the potential to cause you problems when you are trying to legitimately troubleshoot a VM? What if you need to use the Hyper-V Console to figure out *why a VM won't boot or something like that?* Yes, that is a valid point, and one that you need to consider. Shielded VMs make the security of your VMs much higher. So much so that you could, in fact, lock yourself out from being able to troubleshoot issues on that server. As is often the case with everything in the IT world, we are trading usability for security.

Infrastructure requirements for shielded VMs

There are a couple of important pieces in this puzzle that you need to be aware of if you are interested in running shielded VMs.

Guarded hosts

You will need to run one or more guarded host servers in order to house your shielded VMs. Guarded hosts are essentially Hyper-V servers on steroids. They will host VMs like any other Hyper-V Server, but they are specially crafted and configured to host these encrypted shielded VMs, and to attest their own health as part of this overall security strategy.

Guarded hosts must be running Server 2016 Datacenter or Server 2019 Datacenter, and generally you want them to boot using UEFI, and to contain a TPM 2.0 chip. While TPM 2.0 is not a firm requirement, it is certainly recommended.

These guarded host servers then take the place of your traditional Hyper-V Servers. It is their job to host your VMs.

Host Guardian Service (HGS)

HGS is a service that runs on a server, or more commonly a cluster of three servers, and handles the attestation of guarded hosts. When a shielded VM attempts to start on a guarded host server, that host must reach over to HGS and attest that it is safe and secure. Only once the host has passed the HGS attestation and health checks will the shielded VM be allowed to start.

HGS is *critical* to making a guarded fabric work. If HGS goes down, *none* of your shielded VMs will be able to start!

There are different requirements for HGS, depending on what attestation mode your guarded hosts are going to utilize. We will learn about those modes in the next section of this chapter. HGS will have to be running Server 2016 or Server 2019, and most commonly you want to use physical servers running in a three-node cluster for this service.

I also want to point out a capability related to HGS that is brand new in Windows Server 2019: **HGS cache**. A previous limitation of Server 2016 Shielded VMs was that HGS needed to be contacted every time any guarded host wanted to spin up any shielded VM. This can become problematic if HGS is unavailable for some temporary reason. New in Server 2019 is HGS cache for VM keys so that a guarded host is able to start up approved VMs based on keys in the cache, rather than always having to check in with a live HGS. This can be helpful if HGS is offline (although HGS being completely offline probably means that you have big problems), but HGS cache has a more valid use case in branch-office scenarios where a guarded host might have poor network connection to

HGS.

Host attestations

Attestation of the guarded hosts is the secret to using shielded VMs. This is the basis of security in wanting to move forward with such a solution in your own environment. The ability for your hosts to attest their health and identity gives you peace of mind in knowing that those hosts are not being modified or manipulated without your knowledge, and it ensures that a malicious host employee cannot copy all of your VM hard drive files onto a USB, bring them home, and boot them up. Those shielded VMs are only ever going to start on the guarded hosts in your environment, nowhere else.

There are two different modes that guarded hosts can use in order to pass attestation with HGS. Well, actually there are three, but one has already been deprecated. Let's take a minute to detail the different modes that can be used between your guarded hosts and your HGS.

TPM-trusted attestations

This is the best way! TPM chips are physical chips installed on your server's motherboards that contain unique information. Most importantly, this information cannot be modified or hacked from within the Windows operating system. When your guarded host servers are equipped with TPM 2.0 chips, this opens the door to do some incredibly powerful host attestation. The host utilizes Secure Boot and some code-integrity checks that are stored inside the TPM in order to verify that it is healthy and has not been modified. HGS then crosschecks the information being submitted from the TPM with the information that it knows about when the guarded host was initially configured, to ensure that the requesting host is really one of your approved guarded hosts and that it has not been tampered with. If you are configuring new Hyper-V Servers, make sure they contain TPM 2.0 chips so that you can utilize these features.

Host key attestations

If TPMs aren't your thing or are beyond your hardware abilities, we can do a simpler host key attestation. The ability for your guarded hosts to generate a host key that can be known and verified by HGS is new with Windows Server 2019. This uses asymmetric key-pair technology to validate the guarded hosts. Basically, you will either create a new host-key pair or use an existing certificate, and then send the public portion of that key or cert over to HGS. When guarded hosts want to spin up a shielded VM, they reach out to attest with HGS, and that attestation is approved or denied based on this key pair.

This is certainly a faster and easier way to make shielded VMs a reality in your network, but is not as secure as a TPM-trusted attestation.

Admin-trusted attestation – deprecated in 2019

If your environment is new and based on Server 2019, don't pay any attention to this one. However, there are folks who are running shielded VMs within a Windows Server 2016 infrastructure, and in that case, there was an additional option for attestation. Commonly known as admin-trusted attestation, this was a very simple (and not very secure) way for your hosts to attest to HGS that they were approved. Basically, you created an **Active Directory** (**AD**) security group, added your guarded hosts into that group, and then HGS considered any host that was part of that group to be guarded and approved to run shielded VMs.

Integrating with Linux

Many companies utilize Linux in some capacity or another. The use of Linux may actually be poised to make a grander entrance into the Windows Server world now that we have this higher level of integration possible inside Windows Server 2019. There are ways in which your Server 2019 can now be used in order to interact with Linux VMs:

- **Running in Hyper-V**: VMs hosted on a Hyper-V Server used to be limited to Windows-based operating systems. This is no longer the case. The scope of the Hyper-V virtualization host has now been expanded to accommodate running Linux-based VMs in Hyper-V Manager. There is even good integration with the keyboard and mouse!
- **Linux shielded VMs**: You now know about running shielded VMs in Hyper-V, and you also know about running Linux-based VMs inside Hyper-V. *Does this mean we can combine those two ideas and run a Linux VM that is also shielded?* Why yes, we certainly can. This capability was introduced in Windows Server 1709, and also exists in the newest LTSC release of Windows Server 2019.
- **Running in containers**: While most server and Hyper-V administrators won't be chomping at the bit to install Linux on their systems because they simply have no reason to do so, there

will definitely be a lot more Linux-y talk coming from anyone on the DevOps side of the IT house. When building scalable applications that are destined for the cloud, we often talk about running these applications inside containers. In the past, hosting containers on a Windows Server meant that the container itself had to be running Windows, but no more. You can now host Linux-based containers on top of Windows Server 2019. This allows great flexibility to the application-development process and will be an important consideration for the future of containers.

ReFS deduplication

While filesystems and deduplication features are technologies that you may not expect to be discussed when it comes to Hyper-V, the improvements in Server 2019 related to ReFS and the deduplication of data carry some huge advantages for Hyper-V servers. In case these are unfamiliar terms, let's take a minute and define ReFS and deduplication.

ReFS

Anyone who has worked on computers for a while will recognize FAT, FAT32, and NTFS. These are filesystems that can be used when formatting hard drives. The different versions of filesystems translate into different capabilities of how you can utilize that hard drive. For a number of years, NTFS has been the *de facto* standard for all hard disks connected to Windows machines.

That is, until Windows Server 2016 came along. We now have a new filesystem option called ReFS. Even if you work in an IT department every day, you may have never heard of ReFS because so far it isn't getting used all that much. It is primarily used in servers that are involved with **Storage Spaces Direct** (**S2D**). If it's the latest and greatest filesystem from Microsoft, *why isn't it being used as the default option on any new system?* Primarily because ReFS is not a bootable filesystem. That immediately cancels out the capability for systems with a single hard disk to be running ReFS on the whole drive. What that implies is that ReFS is for secondary volumes on servers, perhaps volumes intended to hold large amounts of data.

In those instances where you do format a second volume to be ReFS and store data on it, there are some great resiliency and performance advantages to using ReFS instead of NTFS. These advantages were designed to make S2D implementations work better.

Data deduplication

Data deduplication is simply the ability for a computing system to discover multiple bits of data on a drive that are identical, and *clean them up*. If there were six copies of the exact same file on a system, deduplication could delete five of them, retaining one for the purposes of all six locations. This idea enables some major space savings. Data deduplication itself is not new; we had some capabilities introduced way back in Server 2012 regarding this.

Windows Server 2019 is the first platform where it is possible to enable data deduplication on a volume that is formatted via ReFS.

Why is this important to Hyper-V?

Data deduplication can be incredibly advantageous to run on a volume that stores Hyper-V VM hard drive files because, as you can imagine, there will be a ton of information that is duplicated over and over and over again when you are running dozens of VMs. Think about all of those Windows operating system files that will be identical among all of your VMs running on the Hyper-V host. It's pretty obvious why it would be beneficial to enable data deduplication on a volume that stored VHDX files.

ReFS has some big resiliency and performance advantages over NTFS, and so it is also obvious that VHDX files would be best served by being stored on an ReFS volume.

Windows Server 2019 is the first platform where you can have your cake, and eat it too. We now have the ability to create an ReFS volume for storing virtual machine hard drives, and also enable data deduplication on that same volume.

Hyper-V Server 2019

It's very easy to get excited about virtualization. Build some hardware, install Windows Server 2019, implement the Hyper-V role, and bam! You're ready to start rolling out hundreds and hundreds of VMs in your environment... *right?*

Not necessarily. We haven't talked about licensing yet, and too often our technological prowess is limited by licensing requirements. The same is true with Hyper-V. Every VM that you spin up needs to have its own operating system license, of course. That requirement makes sense. What isn't as obvious, however, is the fact that you can only run a certain number of VMs on your Hyper-V Server, depending on what SKU you use for the host operating system itself.

The biggest *gotcha* is that using Windows Server 2019 Standard Edition as your Hyper-V Server will result in your ability to run two VMs. Two! That's it, no more. You will be able to launch a couple of virtual machines, and will then be prevented from running any more. Clearly, the Standard Edition SKU isn't designed to be used as a Hyper-V Server.

That leaves you with Windows Server 2019 Datacenter Edition. Fortunately, Datacenter allows you to run *unlimited* VMs! This is great news! Except for one thing—Datacenter Edition usually costs many thousands of dollars. This is a very limiting factor for deployments of Hyper-V Servers.

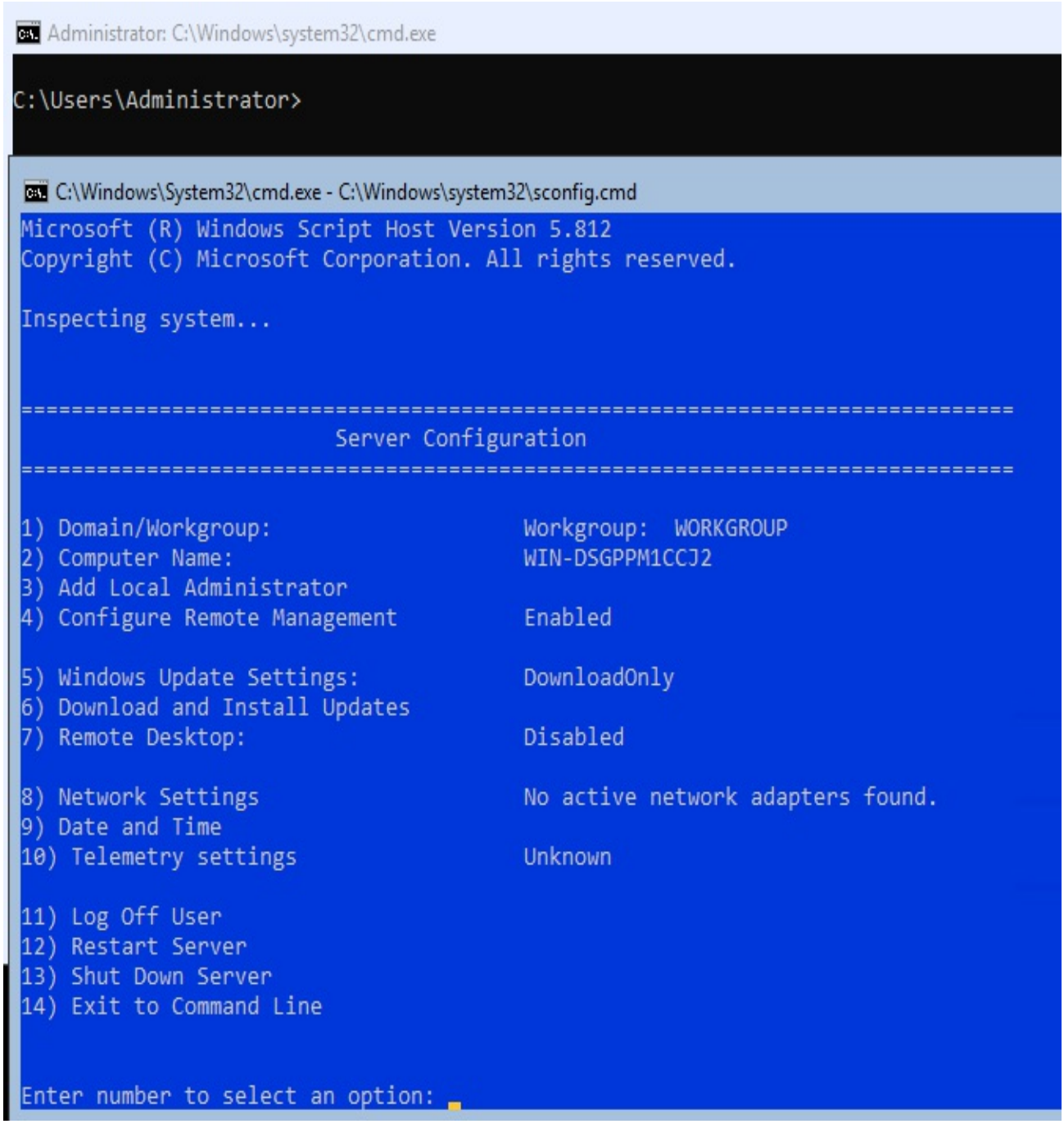
All of this talk about licensing and how messy or expensive it can be leads to one point: **Hyper-V Server 2019**. Wait a minute, *isn't that*

what this whole chapter has been about? Isn't that just Windows Server 2019 with the Hyper-V role installed? No, not at all.

Hyper-V Server 2019 is its own animal. It has its own installer, and a whole different user interface from a traditional server. Installing Hyper-V Server 2019 onto a piece of hardware will result in a server that can host an unlimited number of Hyper-V VMs, but nothing else. You cannot use this as a general-purpose server to host other roles or services. Hyper-V Server also does not have a graphical user interface.

Hyper-V Server 2019 has one *huge* benefit: it's *FREE*. You are still responsible for licenses on each of the VMs themselves, of course, but to have a free host operating system that can run an unlimited number of VMs, now that is something my wallet can really get behind.

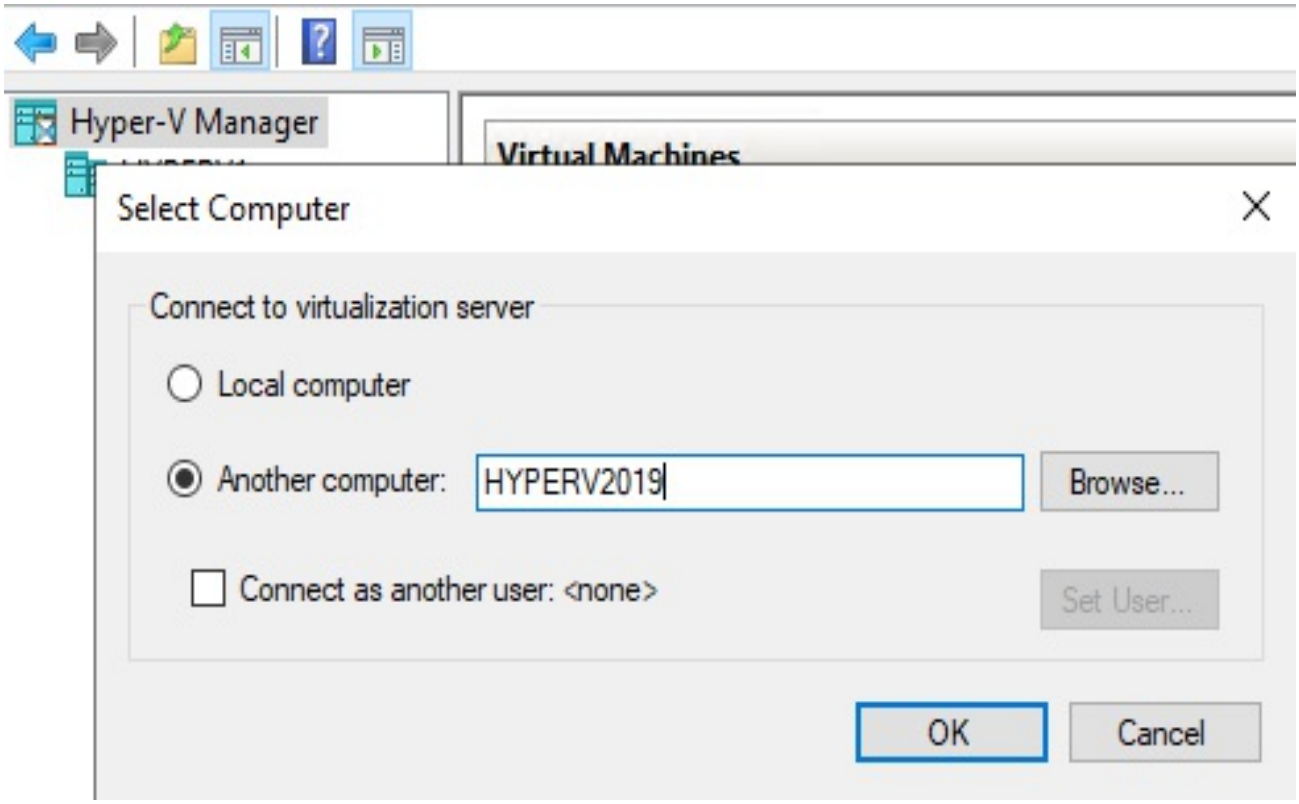
I have burned the ISO installer for Hyper-V Server 2019 onto a DVD (thankfully this one is small enough to actually fit!), and just finished installing it onto my hardware. The installation of the operating system itself was completely familiar: all of the installation screens and options were the same as if I were installing the full version of Windows Server 2019. However, now that the installer has finished and I have booted into the operating system of my Hyper-V Server 2019, everything looks completely different:



We are presented with only a Command Prompt, and inside that prompt it has auto-launched a configuration utility called **SConfig**. By using the keyboard here, I can do things such as set the hostname of this server, join it to a domain, and change networking settings. Once you

have finished using this CLI interface to set the basic requirements on the server and get it communicating with the network, we really don't need to access the console of this Hyper-V Server again, unless you need to backtrack and revisit this configuration screen in order to change something. Instead, after configuring the Hyper-V Server, you simply utilize Hyper-V Manager, or PowerShell, on another server or desktop inside your network, to tap remotely into the management of the VMs that are running on this Hyper-V Server.

In the following screenshot, you can see that I have launched Hyper-V Manager. I am running this instance of Hyper-V Manager from my Windows 10 machine where I have the Hyper-V role installed. From here, I right-click on Hyper-V Manager and choose Connect to Server.... I then input the name of my new Hyper-V Server, and the console creates a remote connection. From this remote connection, I can now utilize all functionality inside my Window 10 Hyper-V Manager as if I were logged straight into the new Hyper-V Server:



Similar to the way that most tasks performed on a Server Core or Nano Server are handled remotely—through the use of remote consoles or PowerShell—we make all ongoing maintenance and administration of this Hyper-V Server happen from a remote Hyper-V Manager console.

Hyper-V Server gives you the security benefits of a GUI-less interface, combined with the flexibility benefits of hosting an unlimited number of virtual machines, at a price point that nobody can argue with!

Summary

I don't have official numbers, but I will take a risk and say that today there are already more virtual servers running than physical servers, to keep our world online. While the battle continues to rage between which hypervisor platform is the best—typically the argument is split between either Hyper-V or VMware—you cannot ignore the fact that virtualization is the way of the future. Microsoft puts great quantities of time and resources into making sure that Hyper-V always stays ahead of the competition, and introducing more and more features with every release so that you can keep your virtualized infrastructure up and running perfectly, all the time. *Is the capacity for cloud virtualization even more powerful than on-premise Hyper-V Server?* I would say yes, because the infrastructure that is in place at a cloud service provider is going to be the all-powerful Oz compared to what a single company can provide in their own data center. *Does this mean you can forget about Hyper-V altogether and just use cloud-provided servers?* Maybe someday, but most aren't ready to take that leap just yet. The need for on-premise servers and services is still immense, and some industries are simply never going to permit their data and applications to be hosted by a third party. Understanding the capabilities of Hyper-V, and being able to build this infrastructure from the ground up, will give you a major advantage when looking for a technology job in a Microsoft-centric organization.

This brings us to the end of our story on the new Windows Server 2019. Many of the topics we discussed could fill entire books, and I hope that the ideas provided in this volume are enough to prompt you to dig further into the technologies that you plan to work with. Microsoft

technology reigns supreme in most data centers across the globe. The new and updated features inside Windows Server 2019 will ensure that this trend continues long into the future.

Questions

1. What are the three types of virtual switches inside Hyper-V?
2. If you needed to build a virtual machine that booted using UEFI, which generation of VM would you need to create?
3. True or False—In Windows Server 2019 Hyper-V, you must shut down a VM in order to change its allocated about of memory (RAM).
4. True or False—The only way to interact with a VM is through the Hyper-V console.

5. What is the name of the technology inside Hyper-V that allows you to take snapshot images of virtual machines, that can later be restored?
6. When running Shielded VMs in your environment, what is the name of the role that handles the attestation of your Hyper-V host servers?
7. Which is the most comprehensive attestation method for Shielded VMs—Host key attestation, TPM trusted attestation, or Admin trusted attestation?

Assessments

Chapter 1: Getting Started with Windows Server 2019

1. Right-click on the Start button and select Windows PowerShell (Admin) from the quick admin tasks menu
2. WinKey+*X*
3. Microsoft Azure
4. Standard and Datacenter
5. 2
6. Server Core
7. Long-Term Servicing Channel (LTSC)
8. Both, although Windows Settings is the preferred method for most configuration options

Chapter 2: Installing and Managing Windows Server 2019

1. Windows Admin Center (WAC).
2. False. Windows Server 2019 can be installed onto physical hardware, or as a virtual machine instance.
3. False. The default option for Windows Server 2019 is Server Core, which does not have a graphical user interface.
4. `Get-WindowsFeature | Where Installed`.
5. True.
6. Remote Server Administration Tools (RSAT).
7. As of this writing, Microsoft Edge and Google Chrome. Note that Internet Explorer is not supported.

Chapter 3: Core Infrastructure Services

1. Organizational Unit (OU)
2. Prestaging the account
3. Active Directory Sites and Services
4. Read-Only Domain Controller (RODC)
5. The Default Domain Policy
6. AAAA record (quad A)
7. DSA.MSC

Chapter 4: Certificates in Windows Server 2019

1. Certification Authority
2. Enterprise Root CA
3. No, this is not a recommended scenario
4. The new certificate template must be published
5. Certificate Auto-enrollment
6. Private key
7. Certificate Signing Request (CSR)

Chapter 5: Networking with Windows Server 2019

1. 128 bits.
2. 2001:ABCD:1:2::1.
3. PATHPING.
4. False. Doing so will cause routing issues. You should only ever have one Default Gateway address on a system, no matter how many NICs it has.
5. New-NetRoute
6. Windows Server 2019, 2016, and 2012 R2.

Chapter 6: Enabling Your Mobile Workforce

1. Always On VPN
2. IKEv2 and SSTP
3. Windows 10 1607
4. When you want to utilize the AOVPN Device Tunnel
5. No, your internal network can be completely IPv4
6. Network Location Server (NLS)
7. WAP can be implemented as an ADFS Proxy

Chapter 7: Hardening and Security

1. Windows Defender Antivirus
2. The Domain profile
3. Public and Private
4. ICMPv4
5. Group Policy
6. Shielded VM
7. Advanced Threat Analytics

Chapter 8: Server Core

1. True
2. False, switching back and forth is not possible in Windows Server 2019
3. Command Prompt
4. `Get-NetIPConfiguration`
5. `Rename-Computer`
6. PowerShell, Server Manager, RSAT, and Windows Admin Center
7. `Sconfig.exe`

Chapter 9: Redundancy in Windows Server 2019

1. Network Load Balancing
2. Dedicated IP address and virtual IP address
3. Unicast, Multicast, and Multicast IGMP
4. NLB is a feature
5. Hyper-V and file services
6. A USB memory stick
7. False, you may use any types of hard drives with S2D

Chapter 10: PowerShell

1. Simply type the word `powershell` and press *Enter*
2. `Get-Command`
3. `Enter-PSSession`
4. `.PS1`
5. RemoteSigned
6. *Tab*
7. The WinRM service

Chapter 11: Containers and Nano Server

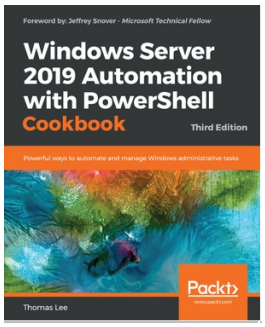
1. Server Core or Nano Server
2. Hyper-V Container
3. False, the ability to run Windows and Linux containers is new as of Windows Server 2019
4. `docker images`
5. Kubernetes
6. True

Chapter 12: Virtualizing Your Data Center with Hyper-V

1. External, Internal, and Private
2. Generation 2
3. False, you can adjust a VM's RAM count on the fly
4. False, once your VM's operating system is installed, you can interact with it through any other traditional administration methods, such as RDP
5. Checkpoints
6. Host Guardian Service
7. TPM trusted attestation

Another Book You May Enjoy

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



Windows Server 2019 Automation with PowerShell Cookbook - Third Edition
Thomas Lee

ISBN: 978-1-78980-853-7

- Perform key admin tasks on Windows Server 2019
- Employing best practices for writing PowerShell scripts and configuring Windows Server 2019
- Use the .NET Framework to achieve administrative scripting
- Set up VMs, websites, and shared files on Azure
- Report system performance using built-in cmdlets and WMI to

obtain single measurements

- Know the tools you can use to diagnose and resolve issues with Windows Server

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!