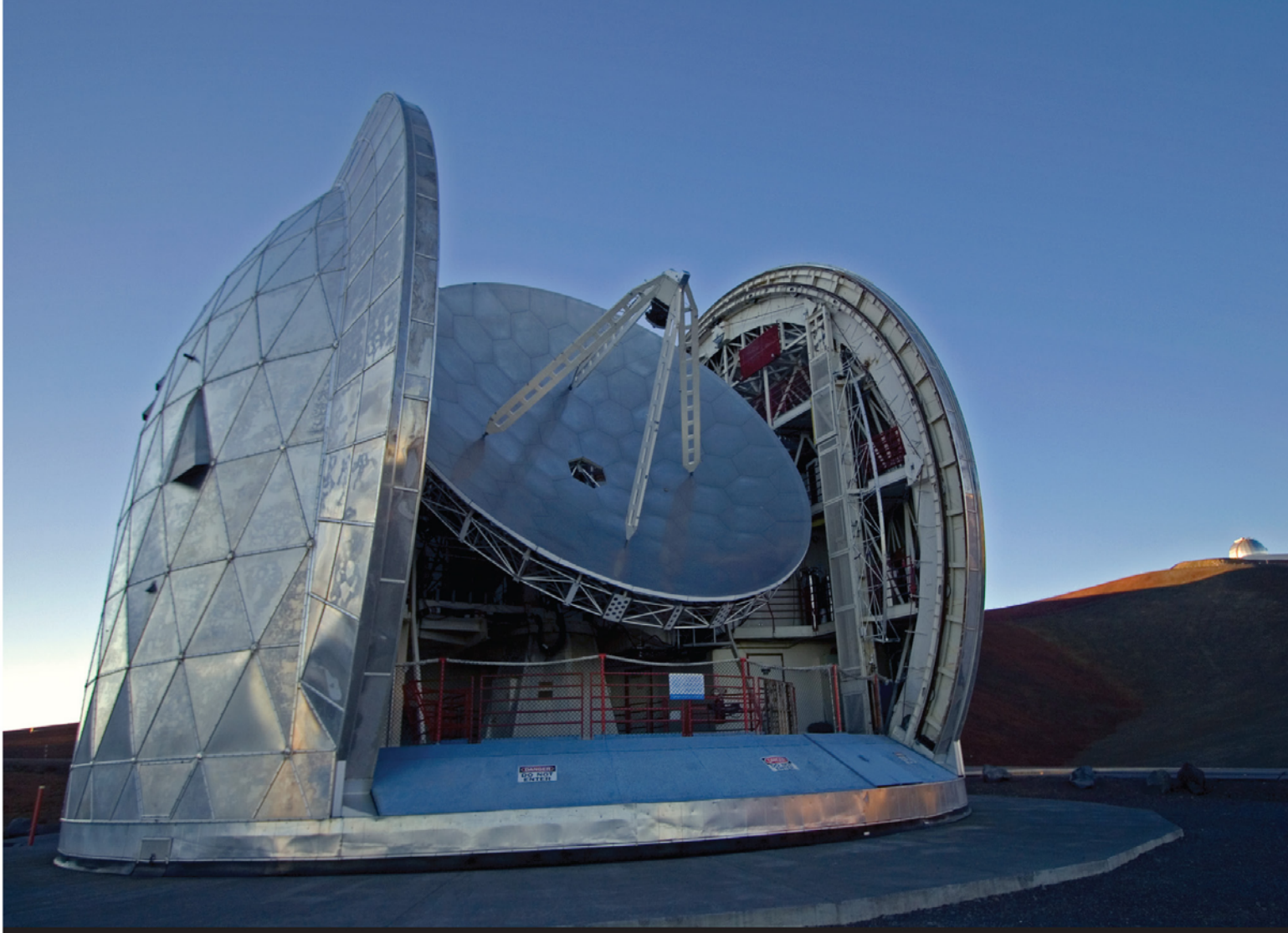


Professional Expertise Distilled

Getting Started with Windows VDI

Create, maintain, and secure scalable and resilient virtual desktops with Windows 8.1 and Windows Server 2012 R2

Andrew Fryer

[PACKT] PUBLISHING enterprise professional expertise distilled

Getting Started with Windows VDI

Create, maintain, and secure scalable and resilient virtual desktops with Windows 8.1 and Windows Server 2012 R2

Andrew Fryer

[PACKT] PUBLISHING enterprise professional expertise distilled

BIRMINGHAM - MUMBAI

Getting Started with Windows VDI

First published: July 2014

Production reference: 1070714

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

ISBN 978-1-78217-146-1

www.packtpub.com

Cover image by Andrew Fryer (andrew.fryer@live.co.uk)

Credits

Author

Andrew Fryer

Reviewers

Ed Baker

Erik Bakker

Marin Frankovic

Patrick Lownds

Puthiyavan Udayakumar

Commissioning Editor

Akram Hussain

Acquisition Editor

Neha Nagwekar

Content Development Editor

Sruthi Kutty

Technical Editors

Dennis John

Sebastian Rodrigues

Gaurav Thingalaya

Copy Editors

Insiya Morbiwala

Aditya Nair

Alfida Paiva

Laxmi Subramanian

Project Coordinator

Venitha Cutinho

Proofreaders

Ting Baker

Ameesha Green

Paul Hindle

Indexers

Mehreen Deshmukh

Rekha Nair

Graphics

Ronak Dhruv

Production Coordinator

Melwyn D'sa

Cover Work

Melwyn D'sa

About the Author

Andrew Fryer (`andrew.fryer@live.co.uk` and `@deepfat`) started out by working in the IT industry for the British government in a number of roles, including as a forensic computer expert and development team leader. This was followed by his second career as a BI consultant, mainly on the Microsoft platform in a variety of industries from MTV to Marks & Spencer.

For the last seven years, Andrew has been a Technical Evangelist for Microsoft in the UK. Essentially, this includes working with Microsoft's latest technologies and explaining the art of the possible to the TechNet community. Some of this includes presenting at big events such as TechEd, IPExpo, and TechDays Online, as well as smaller, more focused events such as IT camps and individual customer engagements. Andrew also tries to keep his blog named *Insufficient data* (`www.andrewfryer.com`) up to date with practical advice and how-to videos, as well as thought leadership around the wider issues affecting IT in the UK. You'll also find Andrew presenting at various user groups in the UK, such as Spiceworks, the Virtualization User Group, and SQLBits.

Thanks to Juliet and my other family—the TechNet UK team of Dan, Ed, Simon, Renee, Steve, and of course Andy M for their support and for letting me get a word in edgeways!

About the Reviewers

Ed Baker, BSc(Hons), FBCS CITP, FIAP, is a 25-year veteran in the IT industry. Currently employed as an Infrastructure Evangelist, Ed spends most of his time writing, blogging, talking, and playing tech. Ed holds a good number of industry certifications, including those in Microsoft's MCSE Private Cloud, Server Infrastructure, and many more. Ed is currently a Microsoft Certified Trainer (MCT) and acts as a Regional Lead for the United Kingdom.

Ed is a Freemason, and currently the Communications Officer for the Province of Worcestershire in the United Kingdom; when not talking Freemasonry or tech, Ed spends most of his time either riding his beloved motorcycle or engaging in charity fundraising by running, walking, or jumping!

Ed is a contributing author to several Microsoft Official Academic Curriculum titles covering Windows Server 2012 and Windows 8. Ed is a Chartered Professional Fellow of the British Computer Society and a Fellow of the Institution of Analysts and Programmers. Ed also spent many years as a lecturer at Open University, covering topics as varied as Project Management, Windows Server, Linux, and Robotics.

Erik Bakker is a freelance consultant / architect based in the Netherlands with a strong focus on Microsoft and Citrix virtualization technologies (SBC and VDI), and implementing and designing Microsoft Private and Public cloud solutions and the System Center Suite in large enterprise environments.

He's been a Microsoft MCSE since 1999 and has broad knowledge of almost any Microsoft product. Besides Microsoft certification, he also holds the highest technical certification for Citrix and is a Certified Citrix Integration Architect and Certified Citrix Expert for Virtualization. Erik can be contacted on Twitter using the `@bakker_erik` handle.

Marin Frankovic was born in Makarska, Croatia, in 1976, where he completed elementary and part of high school. He graduated from high school in the USA, where he attended his senior year as an exchange student. In 2003, he earned the mag. oec. degree from the Faculty of Economics in Zagreb, majoring in Business Computing. As a student, he volunteered in his faculty's IT department for a year as technical support.

After obtaining his degree, Marin started as a Microsoft MOC and IBM ACE instructor in the largest private IT education company, Algebra. There, he also started as a consultant for infrastructure, virtualization, and cloud computing based on Microsoft technologies. Later on, when Algebra opened private colleges for applied computing, he took on a position as head of the operating systems department and took on the responsibility of creating course curriculums and managing several lecturers and assistants.

He also delivers lectures on several key courses in system administration track. Five years in a row, Microsoft honored him with the MVP title for System Center and Datacenter Management. Marin is a regular speaker on all regional conferences, such as WinDays, KulenDayz, MobilityDay, NT konferenca, MS NetWork, and DevArena. In 2011, he was awarded the Microsoft ISV award for his contribution to the Microsoft community. Marin regularly writes technical articles for the *Mreža* IT magazine. His main interests today are cloud computing, virtualization as its core component, and resources consolidation based on Microsoft technologies such as Windows Server and System Center applications.

Patrick Lownds is a Master Solution Architect working in the EMEA Data Center Consulting practice for TS Consulting, which is a business unit in Hewlett-Packard's Enterprise Group division. Patrick is based in London, England, and spends the majority of his time consulting on virtualization, cloud, and workforce productivity solutions.

Patrick has been working in the IT industry since 1988, focusing mainly on core infrastructure technologies within the data center, and has in the past worked with virtual desktop solutions from Citrix, Microsoft, and VMware. In fact, one of the first VDI projects Patrick was ever involved in was for VMware VDM 2.0 back in 2008.

Patrick holds a number of certifications from Citrix, Microsoft, and VMware, and was recognized for his independent community leadership, technical expertise, and real-world knowledge of Microsoft products back in 2009, when he was awarded the title of Microsoft Most Valuable Professional (MVP) for Hyper-V. Patrick speaks around the world at numerous HP and Microsoft events, and his work has been published in magazines, articles, and books.

When not consulting, speaking, or writing about virtualization, cloud, and workforce productivity solutions, he can be most often found on a rugby pitch teaching Tag Rugby to children of various ages. Patrick can be contacted on Twitter using the handle `@PatrickLownds`.

Puthiyavan Udayakumar has more than six years' IT experience with expertise in Citrix, VMware, Microsoft products, and Apache CloudStack. He has extensive experience in designing and implementing virtualization solutions using various Citrix, VMware, and Microsoft products. He is an IBM-certified Solution Architect and Citrix-certified Enterprise Engineer. In addition, he has more than 15 certifications in infrastructure products.

He is the author of the book, *Getting Started with Citrix® CloudPortal™*, *Packt Publishing* and *Getting Started with Citrix Provisioning Services 7.0*, *Packt Publishing*. He holds a master's degree in Science with a specialization in System Software from Birla Institute of Technology and Science, Pilani, and also holds a national award from the Indian Society for Technical Education. He has presented various research papers at more than 15 national and international conferences, including IADIS (held in Dublin, Ireland) followed by the IEEE pattern.

I would like to thank Packt Publishing for giving me the opportunity to review this book. It is a very well-written book, and the project coordinator has done a great job on it.

---

www.PacktPub.com

Support files, eBooks, discount offers, and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

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.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.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.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Instant updates on new Packt books

Get notified! Find out when new books are published by following `@PacktEnterprise` on Twitter, or the *Packt Enterprise* Facebook page.

Table of Contents

Preface

Virtual Desktop Infrastructure (VDI) has been around for several years; however, it has never really become a mainstream solution. I think there are a variety of reasons for this:

- While many vendors promised enormous savings from implementing VDI, the claimants often neglected to mention the amount of backend server computation, networking, and storage that is required.
- Specialist solutions were required to make this work, for example, the remote desktop client to get VDI to run on things such as iPads and Android slates, as well as separate tools to manage VDI deployments.
- The licensing of Windows in VDI is seen as complex and ambiguous.
- Finally, the user experience in VDI has never been as rich as when using a "physical" client desktop. This has meant deployments have been limited to specific parts of some organizations, such as dealing rooms, hot-desk workers, and manufacturing environments.

So why did I want to write a book on this? Because it is still a hot topic; it not only promises to further rationalize and consolidate the IT infrastructure in a business, but it can also address one of the top trends and concerns in IT today: Bring Your Own Device (BYOD). Providing access to VDI from iPads and Android tablets means that IT still has control of corporate applications and data, without having to actually manage those devices. There is clearly an appetite for VDI, as can be seen by recent moves made by various big players in the IT industry. Dell has acquired Quest and thin-client manufacturer Wyse, and VMware continues to develop the VMWare View offering with internal development and acquisitions. Perhaps the most significant recent change is what Microsoft has been doing in VDI. They have launched Windows Server 2012 R2 to handle the backend virtualization and management of VDI along with Windows 8.1 for desktops. Significantly, they have also released up-to-date, freely available remote desktop clients for iOS and Android, removing the need to use third-party software for BYOD scenarios.

In this book, I have included details on how Remote Desktop Services (RDS) has evolved from the old Terminal Services, because RDS also provides virtual desktops. The difference is that Remote Desktop Services provides virtual desktop as a collection of sessions on a common server operating system, where VDI is the business of running of copies of Windows client, each in its own virtual machine on a virtualization host.

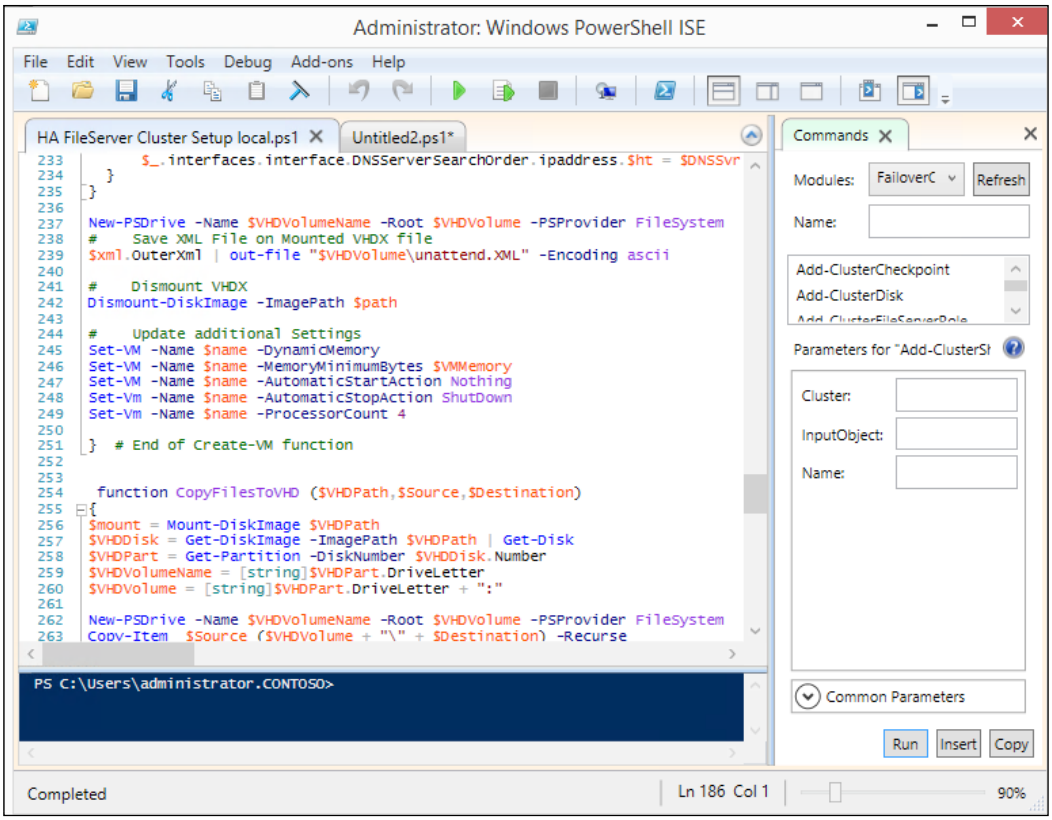
Exploring VDI

The only way I learn about a new technology is to actually use it, even if it's just a lab or demo setup. I haven't included step-by-step guides, as they already exist on TechNet, but I have given you enough guidance to follow along and try for yourself if you have had some exposure to Windows Server, and how to manage it either via the built-in Server Manager or using PowerShell.

PowerShell

I am going to be making use of PowerShell, because we all need to use PowerShell a lot more anyway. It has a lot of benefits, as follows:

- It's a lot easier to read. It has a simplified syntax and there are many more commands. So, there is a lot less effort involved if anyone needs to work with things such as Windows Management Instrumentation (WMI), DHCP, DNS, and many many others.
- The PowerShell ISE (integrated scripting environment) that ships with Windows has an array of tools to make it easier to write code such as IntelliSense and tab completion. It also helps you fill in all the switches with the command list on the right-hand side of the main window and assists with flow control (by navigating to Edit | Snippets, as shown in the following screenshot).



- It's harder to make mistakes with PowerShell, and while we will laugh at and learn from the red errors we get from PowerShell, this is going to be down to bad typing. In a user interface, on the other hand, we could easily miss a checkbox or setting, and it may not show up until later.
- It also makes for a better book. You don't want to spend hours staring at endless screen grabs on your e-book reader or carry around a 2 kg book. I don't have to explain everything click-by-click and screen-by-screen. There's lots of help on the Web, as well as courses and great books to help you get deeper into it.

- All of Microsoft's solutions and products conform to common engineering criteria, part of which requires every feature to have a PowerShell command, but not necessarily a user interface. So, there are things you can only do in PowerShell; for example, we can only configure the advanced features of data deduplication on our file shares and work with virtual machine networks, as there is no interface exposed in Windows Server for these. The same applies to other Microsoft products such as SQL Server and Exchange. Hence, to quote the inventor of PowerShell, Jeffrey Snover, we either need to learn PowerShell or give up IT and play golf!

Please note that the code in this book is designed to show what can be achieved while being as concise and clear as possible, so it's stripped of comments and error trapping that would appear if it was for use in production.

A quick introduction to PowerShell

PowerShell is made of a series of cmdlet packages of commands. When you install a Microsoft product or add in a feature to a server, there will be one or more cmdlets to go with it. These have to be loaded before they can be used with the command `import-module [module name].`.

Any PowerShell command is essentially a `Verb-Noun` combination followed by a lot of switches, for example, `get-PhysicalDisk`, and any `get` command is just looking at things. The verbs are typically get, add, and set, but the list of nouns goes on for miles! A lot of commands have switches, some of which must be completed, while others have defaults. For example, `Get-PhysicalDisk` will show all the disks as no property was specified, such as the name of a disk or a particular size. A key point about PowerShell is the pipe (`|`) command, which passes objects between commands. Have a look at the following example:

```
Get-NetAdapterBinding | Get-Member
```

The first part of this command returns the network binding of the network cards I have on my computer and then pipes these objects to display all the properties and methods of those cards. Having worked out properties I am interested in, I can refine my command as follows:

```
Get-NetAdapterBinding | where enabled -EQ $true | Select-Object -Property
Name, DisplayName, ifDesc| Out-GridView
```

Now, the various network bindings are piped into a filter to return those bindings that are enabled. The surviving objects are then piped to `select-object`, which will return the specified properties (`Name`, `DisplayName`, and `IFDesc`) of these objects and pipe those into `Out-GridView`, which shows them in a graphical table interface.

PowerShell can be run remotely in this case on another server called Orange:

```
Invoke-command –ComputerName Orange –ScriptBlock {get-physicalDisk}
```

This can be extended to run persistent sessions on remote servers that will survive a reboot, and we can actually fire a PowerShell script file (`.ps1`) at another server as well. We have barely touched the surface of PowerShell, but it is enough to help you get an idea of the code in this book.

What this book covers

Chapter 1, *Putting the V in VDI – An Introduction to Virtualization in Hyper-V*, provides the ground work for the rest of the book, as Hyper-V is the foundation of Microsoft's VDI architecture.

Chapter 2, *Designing a Virtual Desktop Infrastructure*, introduces the architecture and roles in a Microsoft VDI deployment.

Chapter 3, *Putting the D in VDI – Creating a Desktop Template*, shows how to use the tools we use to deploy Windows to laptops and desktops and apply them for creating templates for VDI.

Chapter 4, *Putting the R in Remote Desktop*, introduces the gateway role to provide secure access to VDI over the Internet.

Chapter 5, *High Availability*, explores how to make each role in a VDI deployment resistant to failure by adding in redundancy. This also allows us to carry out planned maintenance without affecting our users.

Chapter 6, *Scale and Performance*, discusses how to expand a basic deployment to provide access to thousands of users and best practices to optimize performance.

Chapter 7, *Maintenance and Monitoring*, shows how we can update our VDI deployments with patches for new applications. This also looks at troubleshooting and tuning for performance.

Chapter 8, *Managing User Profiles and Data*, looks at how to ensure our users get the same experience each time they log in and have access to their data. This includes several techniques for doing this depending on whether our users use VDI all the time or make use of physical desktops as well.

Chapter 9, *Virtual Applications*, describes two key topics for VDI: Remote Desktop RemoteApp, which allows us to serve individual applications rather than whole desktops, and App-V, which allows us to stream and deploy applications to virtual desktops without the need to go through the traditional installation process.

Chapter 10, *Licensing and the Future of VDI*, explains how to efficiently license the various deployment scenarios for VDI, such as Bring Your Own Device, and options for remote workers.

What you need for this book

This book is built around a hands-on lab experience, which will need you to have access to a server or laptop with 16 GB RAM, at least an i5 processor, and 200 GB disk space, preferably SSD (if not two disks), to separate the management OS from the disk used to store and run virtual machines.

Microsoft VDI uses Hyper-V as a server role in Windows Server 2012 R2. This is the OS that is needed for the host server. This can be downloaded (`http://msdn.microsoft.com/en-us/library/dn205286.aspx`) as a 180-day evaluation from TechNet or with an MSDN subscription. We'll explore a number of tools from Microsoft to enhance our VDI deployment, including the following free tools:

- The Windows Assessment and Deployment Toolkit (ADK) for Windows 8.1, downloadable from `http://www.microsoft.com/en-us/download/details.aspx?id=39982`
- The Microsoft Deployment Toolkit (MDT) 2013, downloadable from `http://www.microsoft.com/en-in/download/details.aspx?id=40796`
- The Remote Desktop Diagnostic Tool (RDVDiag), downloadable from `http://www.microsoft.com/en-us/download/details.aspx?id=40890`

We'll also make use of other paid Microsoft products:

- The Microsoft Desktop Optimization Pack (MDOP) is only available to customers with software assurance, or MSDN subscribers.
- We will also make use of SQL Server to enable high availability, and support the metadata used for updating services and MDOP. SQL Server 2014 Evaluation Edition can be downloaded from `http://technet.microsoft.com/en-us/evalcenter/dn205290` for a 180-day free trial.

The only third-party applications needed are Foxit Enterprise Reader and VLC media player, but these are just used as example applications to show deployment, and any application you may have would be equally suitable to learn the techniques in this book.

Who this book is for

VDI bridges two skill sets: the desktop specialist looking after the deployment of applications, desktops, and management of user profiles; and the datacenter specialist, who manages the backend roles and services that servers provide, such as virtualization. I also recognize that while Hyper-V and Windows VDI are growing in popularity, there will be many experts in other technologies who will understand the concepts but not necessarily the way things work in Windows VDI, or some of the Microsoft terminology.

I am going to assume you have heard of PowerShell and PowerCLI and have had some exposure to that, as well as the basics of core infrastructure, such as TCP/IP and Active Directory, and have used Windows to some degree. However, as no one is an expert in everything, I have introduced all of the topics relevant to VDI in this book. Also, there are many subtle changes in Windows Server 2012 R2 and Windows 8.1, specifically for VDI, that are hidden in the advanced dialog boxes and new switches in PowerShell. I'll be looking into a lot of those.

Conventions

PowerShell commands within the text are shown as follows: "We can see the disks available on a server by using `get-disk`".

If I need to use a block of PowerShell, it will appear as follows:

```
$CompNames =@("orange","grey")


foreach ($CompName in $CompNames)
{
    Invoke-Command -ComputerName $CompName -ScriptBlock {


    Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\
VmHostAgent\Parameters" -Name "Concurrency" -value 5
    Restart-Computer -Force}
}
```

New terms and **important words** are shown in bold.

I shall refer to the tile-based interface in Windows 8.1 and Windows Server 2012 R2 as the modern UI on which we can run modern applications. In contrast to this, there is also the traditional UI that looks much like Windows 7, where we can run our Microsoft Management Console (MMC) snap-ins, Server Manager, and what I will refer to as traditional applications.

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title through the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website, or added to any list of existing errata, under the Errata section of that title.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Putting the V in VDI – An Introduction to Virtualization in Hyper-V

Most IT professionals would define VDI as a collection of **Virtual Machines** (**VMs**), each running a Windows client OS (Windows 8.1 in this book) that our users will connect to from a variety of devices, such as thin clients, tablets and smart phones, laptops, and so on. In fact, there are other ways of doing this, but these can also rely on VMs, so a basic knowledge of this server virtualization is essential before we can understand **Virtual Desktop Infrastructure** (**VDI**) itself.

Strictly speaking, the business of running multiple operating systems, each in its own VM on one physical host is called server virtualization. This may seem to be picky, but as we'll see in later chapters, there is also application and user virtualization, both of which have their part to play in Windows-based VDI.

In this book, we'll be using **Hyper-V**, the Microsoft solution to virtualize the operating system away from the underlying hardware. Hyper-V is now in its fourth generation and is included inside Windows Server 2012 R2, but may be new to some readers. So, in this chapter, we'll explore the basics of Hyper-V by creating a simple VM. We can then see how to manage this VM and lay the foundation for deploying VDI by implementing a **Domain Controller** (**DC**) in the VM.

Server virtualization and Hyper-V

The Hyper-V role has been a part of the Windows Server operating system since Windows Server 2008. It is also included in the Windows 8 and Windows 8.1 client operating systems. This enables developers to run virtualized servers on a Windows client, but in Windows 8/8.1, the Hyper-V role does not contain advanced features that are available in the full Server edition, which we need for VDI.

It's technically possible to run nearly any x86 or x64 OS in Hyper-V, but if you want support from Microsoft, the Guest OS must either be a supported version of Windows Server or Client, or a supported version and distribution of Linux. For example, Windows Server 2000 and Windows XP aren't supported at all, so they aren't supported on Hyper-V. The full details of Guest OS support in Hyper-V can be found at `http://technet.microsoft.com/en-us/library/cc794868(v=WS.10).aspx`.

Hyper-V can be deployed in a variety of ways, and the easiest of these is simply to add the role into Windows Server running on a physical server. When the hypervisor is installed, the host operating system sits in a parent partition, which is essentially like a VM, and is just there to manage the hypervisor. Technically speaking, Hyper-V is like VMware's ESXi, a true Type 1 or bare-metal hypervisor. This deep integration with the latest virtualization hardware is reflected in the high performance limits for Hyper-V in Windows Server 2012 R2, as shown in the following table:

Resource	Limit
Host – CPU	320 cores
Host – memory	4 TB
Host – limit of logical processors that can be assigned to VMs	2,048
Host cluster – nodes	64
Host cluster – highly available VMs	8,000
VM – logical processors	64
VM – memory	1 TB
VM – virtual disk size	64 TB (VHDX)

These limits are beyond the capabilities of most modern servers and enable all but the largest workloads to be virtualized, including substantial VDI deployments.

Most modern servers are designed to run Hyper-V and, by association, support VDI. However, you should check that your servers have been tested with Windows Server 2012 R2 before using it for VDI in production, either with your hardware vendor or directly on the Microsoft **Hardware Compatibility List** (**HCL**) at `http://windowsservercatalog.com/results.aspx?bCatId=1283&avc=10`.

For Hyper-V servers running VDI VMs, there's also a further hardware prerequisite – **Second Level Address Translation** (**SLAT**) on the CPUs. This is needed for RemoteFX, the technology used to virtualize a **Graphics Processing Unit** (**GPU**). **Advanced Micro Devices** (**AMD**) refers to this as **Nested Page Table** (**NPT**) or **Rapid Virtualization Indexing** (**RVI**), and Intel calls it **Extended Page Table** (**EPT**). Either way, you'll need a CPU that supports this if you want a good graphics experience for users in Windows VDI.

Before any servers or laptop can run Hyper-V, virtualization support must be enabled in the BIOS / **Unified Extensible Firmware Interface** (**UEFI**), specifically the following features:

- **Data Execution Protection** (**DEP**)
- CPU-assisted virtualization depends on which CPU manufacturer you have, and you'll either need to enable Intel VT-x or AMD-V

If Hyper-V is installed without these supporting features in place, the Hyper-V service will not start, and we'll get errors in the event log of the server.

Virtual Hard Disks

In server virtualization, a physical disk is represented by a single file, a **Virtual Hard Disk** (**VHD**), and a VM will have one or more of these. In Hyper-V, VHDs have either the VHD or VHDX extension, where VMware uses the VMDK format. The VHD format came out with Hyper-V in Windows Server 2008, while the newer VHDX format was introduced with Windows Server 2012. VHDX has a number of advantages over the older format, but both are supported in Windows Server 2012 R2. They have the following properties:

- VHD is limited to 2 TB, whereas VHDX can be up to 64 TB, the limit for a **New Technology File System** (**NTFS**) volume
- VHDX has the concept of physical and logical block sizes to run much faster on larger physical disks of 4 KB block size
- VHDX is more resilient to failure and is extensible for use by third parties, as it has additional XML metadata to rack updates and store custom information

VHDs have other uses outside of server virtualization, for example, Windows backups are in VHD format and now, with Windows Server 2012, they are in VHDX format. In this book, I will refer to Virtual Hard Disks as VHDs, and I'll point out when there is a specific need to use VHDX.

We have three different options when creating VHDs or VHDXs, as follows:

- **Fixed-size disks**: These are also known as thick provisioning disks, where VHDs reserve space equal to their size on a physical disk.
- **Dynamically expanding disks**: These are also known as thin provisioning disks, and these just put a placeholder on the physical disk they are stored on. The disk then grows as data is written into it.

Originally, fixed disks were much faster than differencing disks, but at the expense of reserving space on the disk and making them larger and more difficult to move around. The gap has closed in Windows Server 2012 R2, but it's worth remembering that a fixed disk on a clean physical disk will be contiguous and not fragmented, and a dynamic disk has to store all the information about where all the physical blocks are. So, there will always be a performance penalty.

- **Differencing disks**: They work by creating a VHDX where changes are written to the differencing disk, but they are based on a parent disk that never changes. The parent disk can either be a fixed or dynamic disk, and the differencing disk will inherit this property.

Microsoft's VDI makes extensive use of both differencing disks and dynamically expanding disks, but this is all done automatically for us.

Installing and configuring Hyper-V

In order to explore Hyper-V and then overlay VDI on it, we are going to need a server to work with. For evaluation purposes, this could simply be a modern laptop; I have a 16 GB laptop with an Intel i7 processor, which I have modified by adding a 750 GB **Solid State Drive** (**SSD**) to reduce disk contention when I am running multiple VMs. I am assuming that you have something like this to work on. The key thing is that it can support Hyper-V, and you have enough RAM and disk space to support the running of multiple VMs on it. Microsoft's Windows Server 2012 R2 is available as a free evaluation, which is good for 180 days once activated. This can be downloaded from the following link:

`http://technet.microsoft.com/en-us/evalcenter/dn205286.aspx`

An option is available to alter the **Boot Configuration Database** (**BCD**) to boot directly from a VHD or VHDX (this requires a minimum of a VHD or VHDX running Windows 7 or Windows Server 2008 R2). One advantage of this option is that it gives us the ability to change different VHDs or VHDXs quickly and create alternative OS configurations. The full details are available on TechNet at `http://technet.microsoft.com/en-us/library/dd799299(v=WS.10).aspx`. Navigate to **Manage** | **Add Roles and Features**.

Our first task is to add in the Hyper-V role from Server Manager. Perform the following tasks to achieve this:

1. Select the server you are working on in the **Before You Begin** page and click on **Next**.
2. On the **Installation Type** page, select the option **Role-based or feature-based installation**.
3. Select the server you wish to work on and click on **Next**.
4. Select **Hyper-V** from the **Server Roles** page and click on **Next**.
5. You'll see that the option to install the Hyper-V management tools is already selected, so click on **Next**.
6. Don't create any virtual switches on this screen. We'll want to explore this and create our own as we configure Hyper-V after it's installed.
7. On the **Confirmation** page, be sure to select **Restart the Destination Server Automatically If Required** and click on **Install**.

We can do the same thing with just one PowerShell command as follows:

```
Add-WindowsFeature Hyper-V -IncludeManagementTools -Restart
```

Downloading the example code

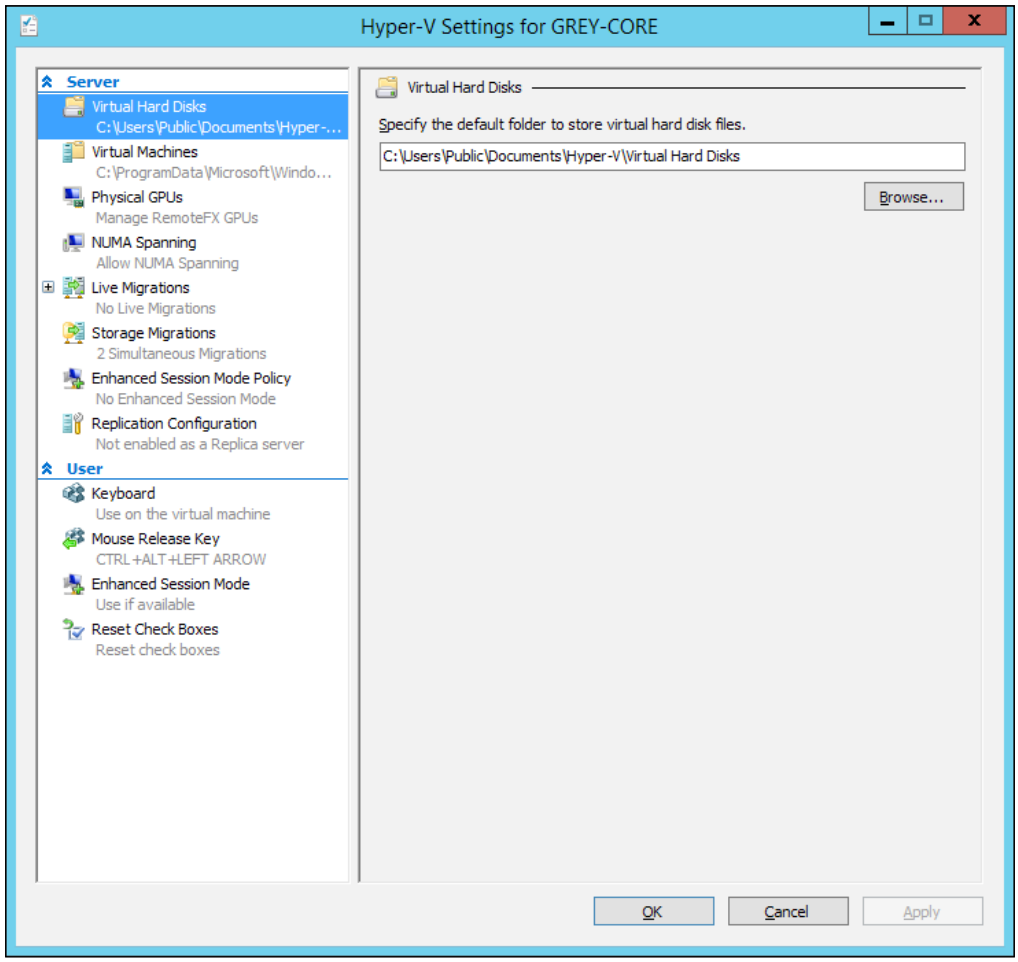
You can download the example code files for all Packt books you have purchased from your account at http://www.packtpub.com. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.

No other server roles should be installed on a server with the Hyper-V role installed, with one exception: failover clustering. Failover clustering is needed to provide **High Availability** (**HA**) of Virtual Machines.

After the server reboots twice, we are ready to use Hyper-V.

Configuring Hyper-V

Before we can use Hyper-V to create VMs, we need to perform some initial configuring to provide the resources that our VMs will need, such as CPU, graphics, storage, and networking. This is done through the **Hyper-V Settings** option in **Hyper-V Manager**, the **Microsoft Management Console** (**MMC**) snap-in that we've just installed, as shown in the following screenshot:

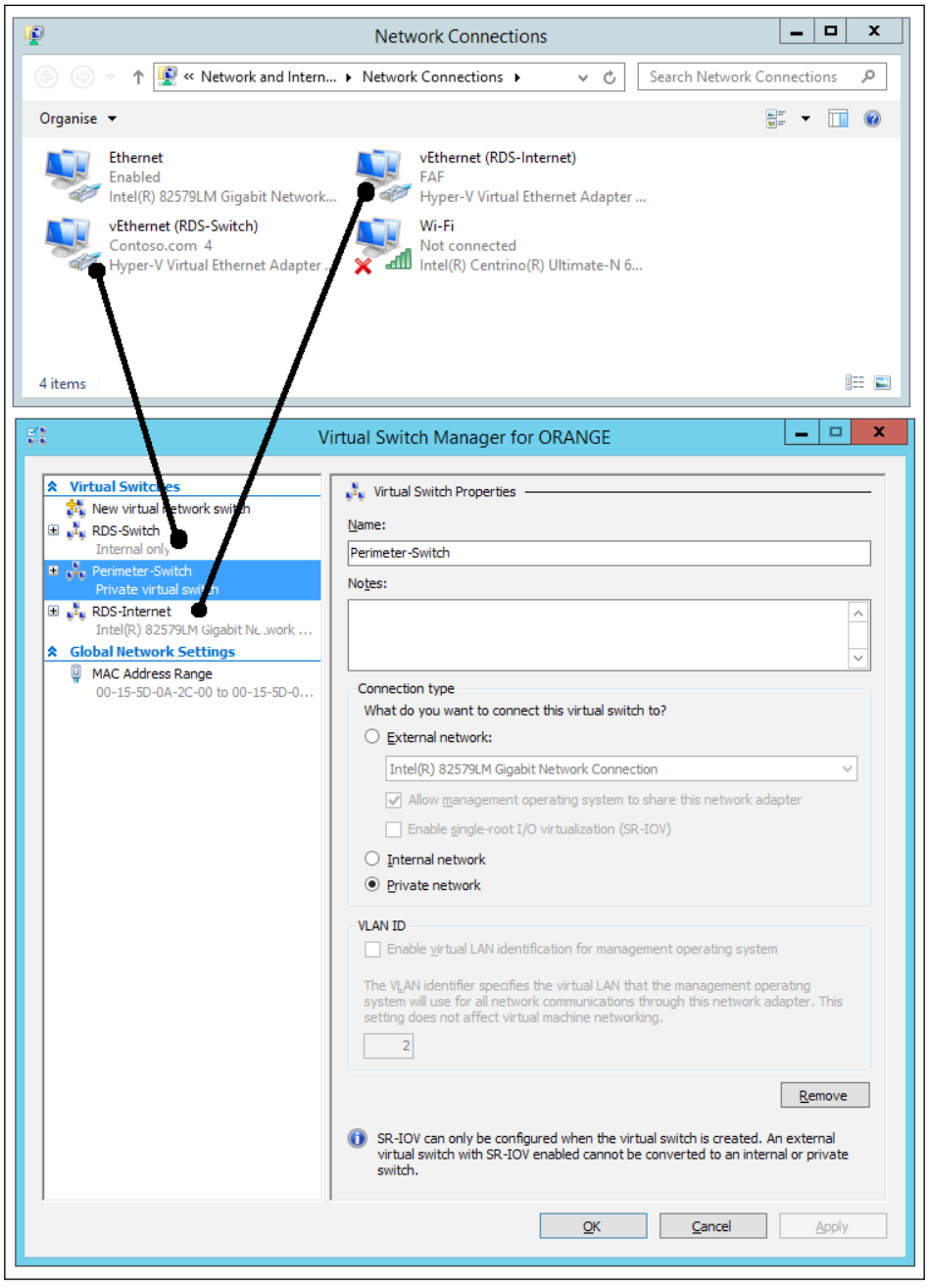


The first two options are how we set the default locations for VMs and their VHDXs. Notice that, by default, new VMs and their associated disks get stored on `C:`, which is probably the last place we want them, especially if we have just configured Boot to VHD, as the VHDXs for the VMs will now reside inside another VHDX file. So, we need to change this to a large fast disk that will be a suitable home for our VMs (in my case, `e:\TempVMStore`, which is sitting on an SSD). We'll return to the other settings for Hyper-V in later chapters, as it would be good to get a basic VM up and running quickly so that we can explore Hyper-V. The final step before we create a VM is to configure networking in Hyper-V by creating one or more virtual switches that we can then connect to our VMs. There are three types of virtual switches, as follows:

- **External virtual switch**: This is bound to an actual network adapter on our host (physical NIC). There's an important setting for external networks to allow the management operating system to share this network adapter. If this is set, then a host server can communicate with other physical and virtual servers over the physical NIC. If it's left unchecked, then the physical host cannot use that physical NIC.
- **Internal virtual switch**: This type of virtual switch is *not* bound to a physical switch at all. We could use this for our first look at VDI if we are doing everything on one physical server, as it allows the physical host and the guest VMs to connect over it.
- **Private virtual switch**: This can also be useful for some types of sandbox or restricted services as it allows the VMs to communicate over it, but there is no communication with the physical host at all.

Any internal virtual switches and external virtual switches that are set to be shared with the management operating system will be visible in the network connections on the host, and they will appear with vEthernet (the name of the virtual switch in Hyper-V). Private switches and external virtual switches that are not shared will only show up as virtual switches in Hyper-V and are completely hidden from the host.

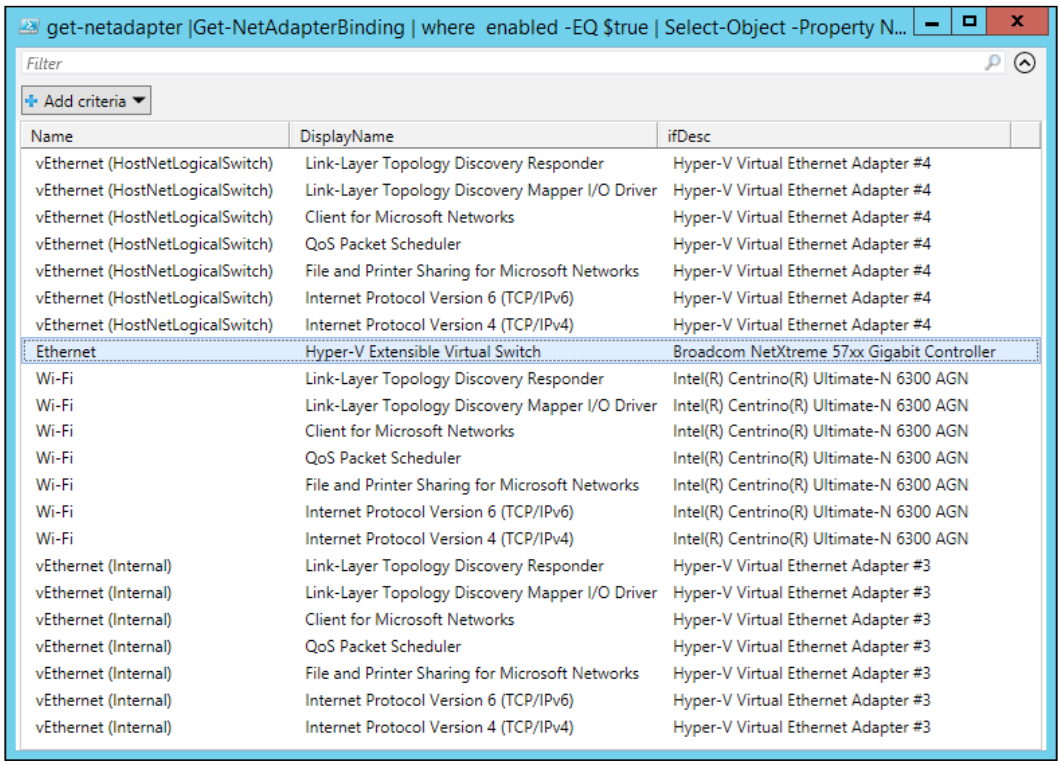
On my demo setup, shown in the following screenshot, we can see what's going on if we look at the network connections and the Hyper-V switches:



Here, we can see that I have created two virtual switches: **RDS-Switch** and **RDS Internet** in Hyper-V. These correspond to the two network connections: **vEthernet (RDS-Switch)** and **vEthernet (RDS Internet)**. What isn't obvious until we examine its settings is that the Ethernet network connection is controlled by Hyper-V. The only protocol that is enabled when Hyper-V is managing a physical NIC is the Hyper-V Extensible Virtual Switch. The easiest way to see all of this is using PowerShell, as I can get all the relevant information on one screen using the following command:

```
Get-NetAdapterBinding | where enabled -EQ $true | Select-Object -
Property Name, DisplayName, ifDesc| Out-GridView
```

This works by looking for all the protocols on each of the network connections and filtering out the ones that are enabled before outputting the specific properties. I am interested in a grid view, like the one shown in the following screenshot:



Name	DisplayName	ifDesc
vEthernet (HostNetLogicalSwitch)	Link-Layer Topology Discovery Responder	Hyper-V Virtual Ethernet Adapter #4
vEthernet (HostNetLogicalSwitch)	Link-Layer Topology Discovery Mapper I/O Driver	Hyper-V Virtual Ethernet Adapter #4
vEthernet (HostNetLogicalSwitch)	Client for Microsoft Networks	Hyper-V Virtual Ethernet Adapter #4
vEthernet (HostNetLogicalSwitch)	QoS Packet Scheduler	Hyper-V Virtual Ethernet Adapter #4
vEthernet (HostNetLogicalSwitch)	File and Printer Sharing for Microsoft Networks	Hyper-V Virtual Ethernet Adapter #4
vEthernet (HostNetLogicalSwitch)	Internet Protocol Version 6 (TCP/IPv6)	Hyper-V Virtual Ethernet Adapter #4
vEthernet (HostNetLogicalSwitch)	Internet Protocol Version 4 (TCP/IPv4)	Hyper-V Virtual Ethernet Adapter #4
Ethernet	Hyper-V Extensible Virtual Switch	Broadcom NetXtreme 57xx Gigabit Controller
Wi-Fi	Link-Layer Topology Discovery Responder	Intel(R) Centrino(R) Ultimate-N 6300 AGN
Wi-Fi	Link-Layer Topology Discovery Mapper I/O Driver	Intel(R) Centrino(R) Ultimate-N 6300 AGN
Wi-Fi	Client for Microsoft Networks	Intel(R) Centrino(R) Ultimate-N 6300 AGN
Wi-Fi	QoS Packet Scheduler	Intel(R) Centrino(R) Ultimate-N 6300 AGN
Wi-Fi	File and Printer Sharing for Microsoft Networks	Intel(R) Centrino(R) Ultimate-N 6300 AGN
Wi-Fi	Internet Protocol Version 6 (TCP/IPv6)	Intel(R) Centrino(R) Ultimate-N 6300 AGN
Wi-Fi	Internet Protocol Version 4 (TCP/IPv4)	Intel(R) Centrino(R) Ultimate-N 6300 AGN
vEthernet (Internal)	Link-Layer Topology Discovery Responder	Hyper-V Virtual Ethernet Adapter #3
vEthernet (Internal)	Link-Layer Topology Discovery Mapper I/O Driver	Hyper-V Virtual Ethernet Adapter #3
vEthernet (Internal)	Client for Microsoft Networks	Hyper-V Virtual Ethernet Adapter #3
vEthernet (Internal)	QoS Packet Scheduler	Hyper-V Virtual Ethernet Adapter #3
vEthernet (Internal)	File and Printer Sharing for Microsoft Networks	Hyper-V Virtual Ethernet Adapter #3
vEthernet (Internal)	Internet Protocol Version 6 (TCP/IPv6)	Hyper-V Virtual Ethernet Adapter #3
vEthernet (Internal)	Internet Protocol Version 4 (TCP/IPv4)	Hyper-V Virtual Ethernet Adapter #3

My suggestion for a simple lab setup is to create an external switch and allow the management operating system to share it so that we can reach our VMs from outside the physical server.

Creating a simple virtual machine

Before we build our first VM, we need to understand what we are creating. In Hyper-V, the VM is made up of three things, as follows:

- One or more Virtual Hard Disks (VHDX), one of which will have the OS.
- Metadata about how the VM is configured, such as how many logical processors it has, how much memory, how many **Network Interface Cards** (**NICs**) it has, and what these are connected to.
- Its memory execution state. If a VM is running and has 2 GB of RAM memory, then this is what is in that 2 GB of RAM. VMs can be paused/saved in the same way as a physical laptop has hibernation. VMs can be set to be automatically saved when the host is shut down. When a VM is resumed, this memory state is copied back into the RAM. The file for this is the `BIN` file, and its size is related to the amount of memory allocated to the VM.

One of the simplest ways to create a VHDX complete with an operating system is with PowerShell. Actually, the PowerShell code is pretty complex, but it's encapsulated in a script available from the TechNet gallery (`http://gallery.technet.microsoft.com/scriptcenter/Convert-WindowsImageps1-0fe23a8f`).

You can either work out the command-line switches for this script or invoke it with `-showUI`, as shown in the following command:

```
.\Convert-WindowsImage.ps1 -ShowUI
```

The preceding command will give you a simple UI to fill in to create an image. One reason for doing it this way is that there are multiple installation options on a given Windows ISO, for example, in Windows Server, there will be the option to install Server Core or the full user interface, and the dropdown in the UI will allow you to select that. We'll be using this script later to create a VHDX with Windows 8.1 as a template for our VDI VMs. For now, we can use this script to make a VHDX from the Windows Server 2012 R2 ISO. I suggest that we call this `WS2012R2 Sysprep.VHDX` to denote that the OS is in a sysprepped state.

If we just copied the disk of an existing VM, then its **System Identifier** (**SID**) remains the same even though we might rename the server or client, and we won't be able to have both the VMs in the same domain. We use sysprep to remove the SID, and when each copy is started, a unique SID is created so that each VM can then properly join a domain.

We could at this stage go and build a VM around it, but as we will be creating several VMs and the disk space is limited in our demo lab, we will make use of differencing disks. To create a differencing disk from this VHDX, we can use the **New Virtual Hard Disk Wizard** dialog box in Hyper-V Manager. We will perform the following steps:

1. From **Hyper-V Manager**, navigate to **New | Hard Disk** in the task pane. Click on **Next** to skip past the **Before You Begin** screen.
2. In the **Choose Disk Format** screen, select the VHDX format and click on **Next**.
3. In the **Choose Disk Type** screen, select **Differencing**.
4. In the **Specify Name and Location** screen, give the disk a name and path. It's a good idea to name disks used to store and run a server OS on a VM with the same name as the VM they belong to. We'll use `Test.VHDX` for this and put it on a path where you have good disk speed, and away from the host server OS if possible (I am using `e:\TempVMStore` throughout this book). Click on **Next** to continue.
5. In the **Configure Disk** screen, we need to identify the sysprepped VHDX we created earlier, `WS2012R2 Sysprep.VHDX`, and click on **Next** to continue.
6. We can then confirm our choices on the **Summary** screen and click on **Finish** to create the disk.

The equivalent command in PowerShell is as follows:

```
New-VHD –Path "E:\TempVMStore\Test.VHDX" -Differencing –ParentPath
"E:\TempVMStore\ WS2012R2 Sysprep.VHDX"
```

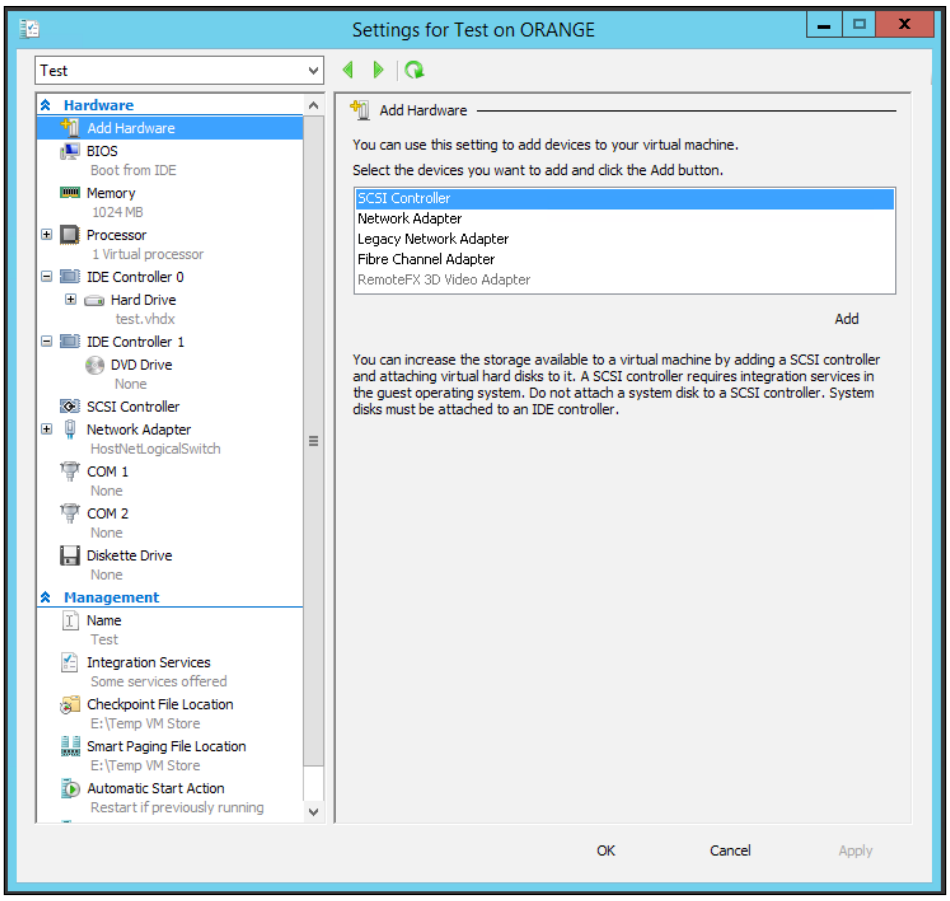
If we look at the `Test.VHDX` disk that we just created, it's just 4 Mb in size because it's dynamically expanding (thin provisioned), which means we have declared what the size it is (in this case, based on the size of the parent disk). But, as yet we haven't used any of it, so it's just a placeholder. Now we need to specify the other resources that our test VM will need. The **New Virtual Hard Disk Wizard** dialog box could be used for this. If you do decide to do this, you'll realize that there are a lot of settings to specify, such as the number of processors, memory, disks, and networking. Rather than fill this book with lots of screenshots of dialog boxes like this, it's simpler and more efficient to use PowerShell. In fact, we can create a working VM with just one (rather long) command line, as follows:

```
New-VM -Name Test -VHDPath 'E:\Temp VM Store\test.vhdx' -
MemoryStartupBytes 1Gb -BootDevice IDE -SwitchName
HostNetLogicalSwitch
```

The terms used in the command line are explained as follows:

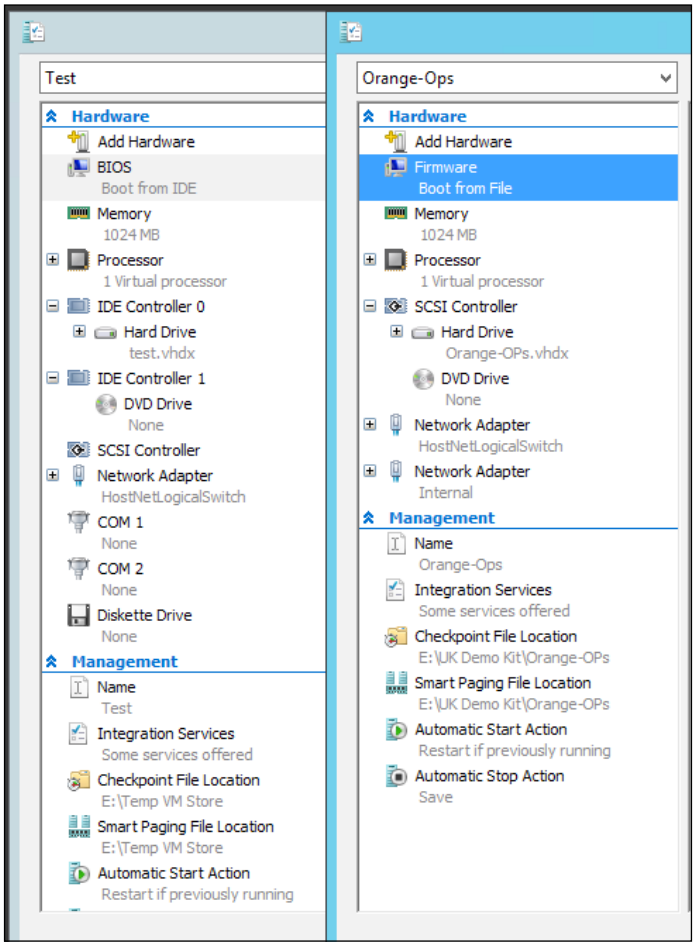
- `-Name`: This is the name given to the VM
- `-VHDPath`: This is the VHDX we just created, and this gets mounted onto the virtual IDE connector in the VM
- `-MemoryStartupBytes`: This is how much memory we are giving the VM, and PowerShell knows what MB, GB, TB, and even PB (petabytes) are
- `-BootDevice`: This means we are going to boot from our new VHDX
- `-SwitchName`: This is the name of the external virtual switch that we have created

If we go back to Hyper-V Manager, we can see our new VM `Test`. It's currently turned off, and before we start it, we can review the settings we have made. This is shown in the following screenshot:



Here, we can see that it has 1 GB memory, that the VHDX we made is connected to the IDE controller, and the network adapter is connected to our virtual switch. Note that some of the hardware settings here can only be made while the VM is off for example, we can add in another network adapter or a second hard disk could be connected to the IDE controller. Some operations can be performed online and it is possible to add or remove virtual disks that reconnected to the **Small Computer System Interface** (**SCSI**) adapter while the machine is on. But, in this type of VM, we can only boot from a disk connected to an IDE controller. There is actually no difference in performance when you connect a VHD or VHDX to either type of controller.

There is a new type of VM in Windows Server 2012 R2, a second-generation VM that removes a lot of these restrictions. It's based on UEFI rather than BIOS and, if we look at its settings alongside the first-generation VM we created, there are a number of changes. This is shown in the following screenshot:



As you can see, there is **Firmware** instead of **BIOS**. In a second-generation VM, we are booting from a file. It's not shown in the screenshot, but we can also enable a secure boot so that the VM won't start if the VHDX with the OS has been modified externally. Second-generation VMs only have SCSI connectors to attach virtual disks and DVD drives. Only VHDX format virtual disks can be used, and these can be expanded while the VM is online. There are other changes that are not visible here; the VMs make less demands on the hypervisor, and the attack surface of the VMs is reduced. However, I am going to stop there because at the time of writing, we can't use a second-generation VM as a VDI template. So all the desktop VMs that we are going to create in this book are going to have to be from the first-generation VM. Hopefully, this will be possible soon so VDI can make use of the benefits of this new type of VM.

Second-generation VMs don't support RemoteFX, so USB redirection won't work. If we try to use a second-generation VM as a template for VDI, then the process will fail. For more on second-generation VMs, refer to `http://technet.microsoft.com/en-us/library/dn282285.aspx`.

We can now start our VM either by right-clicking on it or using the following simple PowerShell command:

```
start-VM -Name Test
```

In the preceding command, `-Name` is our test VM. When we run this command, our new VM will come out of sysprep and all the settings and changes made during that process will be written to our new differencing disk, not the read-only parent disk (`WS2012R2 Sysprep.VHDX`). This differencing disk will now take up about 700 MB. It's also worth noting that the VM is using resources on the host via the Hyper-V integration components (similar to VMware tools), which provide driver support for the virtual devices we have specified, such as the SCSI and network adapters. We can connect to our VM at any time after start up by right-clicking on it in Hyper-V Manager, and we can watch it complete its initial configuration. After that, we can enter the administrator password and see that it is now running Windows Server 2012 R2.

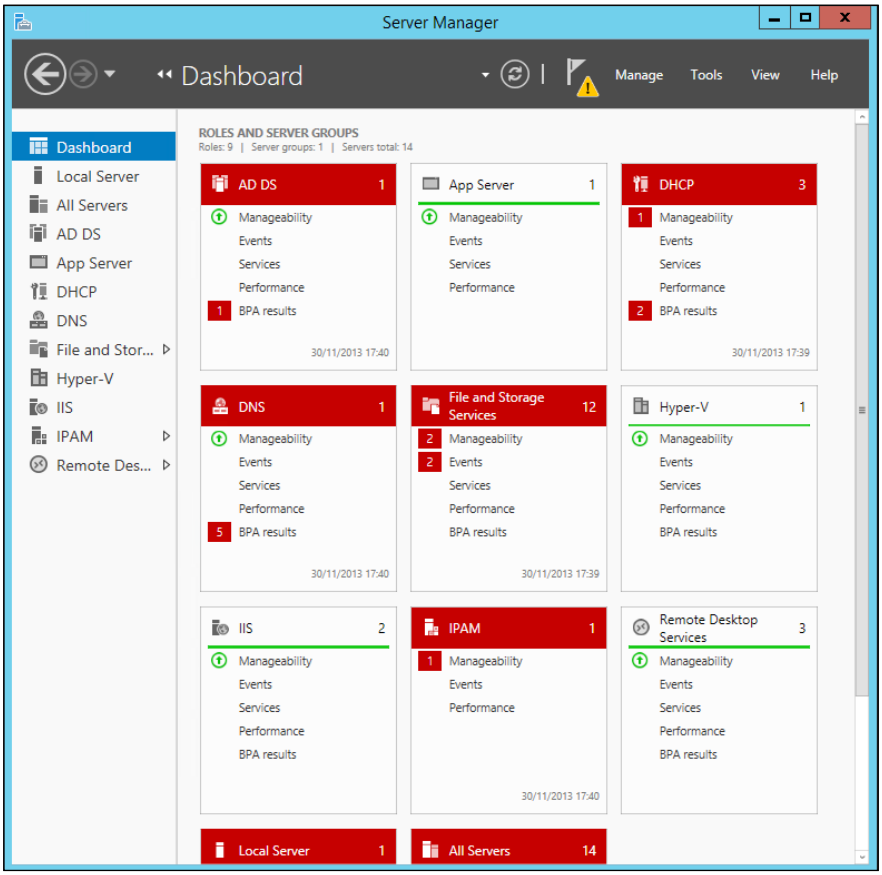
Checkpoints

In Hyper-V, it's possible to capture the state of a VM at a point in time and roll back by creating a checkpoint (in VMware and older versions of Hyper-V, this is called a snapshot). This uses the same differencing system. However, this is controlled by Hyper-V, so don't delete or merge these disks directly! When a checkpoint is made, the state of the VHD is frozen by making it read only, and a new VHD (an AVHD in Hyper-V) is created, into which any changes are written. Reverting to a checkpoint removes the AVHD and re-enables the original VHD. When checkpoints are deleted, the checkpoint AVHDs are merged back into the original VHD or chain of AVHDs if you have more than one checkpoint. It's also important to understand that checkpoints will impact the performance of a VM just as differencing disks do, and that creating checkpoints for a VM is not a substitute for creating a backup.

Managing Windows Server and Hyper-V

The Test VM we created is largely useless as it is; it has no roles and features on it and we can only connect to it using the Hyper-V console, the virtual equivalent of wandering into a data center and logging on to it directly. Windows Server is designed to be remotely managed, whether it's used for Hyper-V or performing a role in VDI. There are several ways of doing this from another server or desktop with the **Remote Server Administration Tools** (**RSAT**) via System Center, and of course with command-line tools such as PowerShell.

In older versions of Windows Server, remote management had to be enabled, but now it's already configured for servers and desktops in the same **Active Directory** (**AD**) domain. This has greatly increased the power of Server Manager as it's now possible to see how groups of servers are performing either by their function or by some grouping of your own, as shown in the following screenshot:



The boxes with the dark gray (red in reality) headers have issues, and these can be identified by simply clicking on the problem identified in the box: **Manageability**, **Events**, **Performance**, or **BPA results**. With RSAT installed, all the individual servers can be managed by using tools appropriate to the roles they are performing. For example, if the server is running Hyper-V, then you are presented with the option to use Hyper-V Manager, where you would get the DNS console option for a DNS server. You can also remotely connect to them and run those RSAT tools remotely from the server you are working on. Server Manager can also add and remove roles and features on any of the managed servers, or to a VHDX while the VM is turned off.

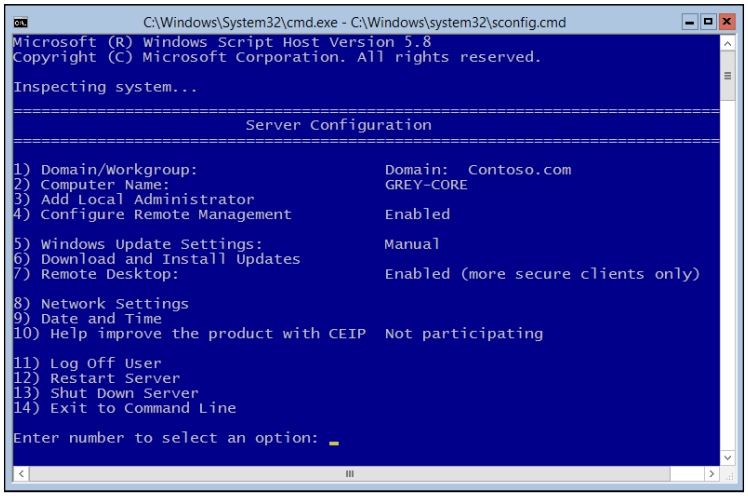
Hyper-V Server and Server Core

Remote management is very important when it comes to putting Hyper-V into production. We have already established that having any roles and features on a Hyper-V host apart from Hyper-V is not best practice. So why have the management tools and any kind of user interface on these servers at all? There is no good answer to that and there are two ways of achieving this, as follows:

- If we are going to use a physical server to host VMs running Windows Server, we should use the default way of installing Windows Server, Server Core.
- If the physical host is going to host VMs running Windows client VMs for VDI or we plan to deploy Linux VMs, then we should use the free, cut-down version of Windows Server, Hyper-V Server 2012 R2 (`http://technet.microsoft.com/en-us/evalcenter/dn205299.aspx`). This only has the Hyper-V role, the file server role, and the failover clustering feature included in it.

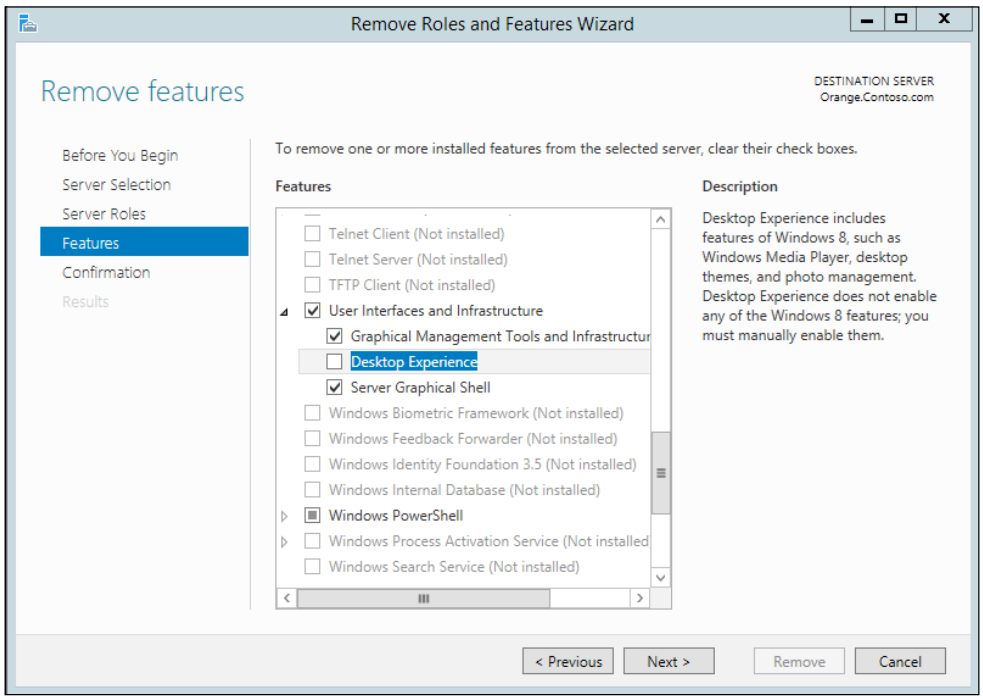
In both cases, there is no graphical shell included, such as MMC. There are just six essential tools that we can use in an emergency for local management, as follows:

- PowerShell
- Command line
- Task Manager
- Registry Editor
- Notepad
- SConfig, a lightweight shell for basic tasks



The SConfig shell – just enough options to configure the operating system before it can be remotely managed

Both Server Core and Hyper-V Server require far less patches and updates (only about 25 percent compared to a full installation of Windows Server), and at about 4 GB, they are under half the size of a full installation of Windows Server. The other benefit is that with no Internet Explorer, no one can go surfing the Web on our servers! Just to be clear, this is the only interface available in Hyper-V Server. The new feature for Windows Server 2012 is the option to post-install the server graphical shell and management tools on top of Server Core, or remove them just like any other role or feature. Have a look at the following screenshot:



The effects of changing the options under **User Interfaces and Infrastructure** are as follows:

- Removing all of the options leaves us with the Server Core option
- If we just keep the **Server Graphical Shell** feature, we can still run MMC snap-ins and Server Manager, but there won't be a modern desktop, an explorer bar, file explorer, or Internet Explorer; this is often referred to as "MinShell"
- If we elect to install Windows with a user interface, we'll get the **Server Graphical Shell** and **Graphical Management Tools and Infrastructure** features

- If we add in the **Desktop Experience** feature, our server will look even more like Windows 8.1, complete with the Windows Store

Getting started with server management

The simplest way to show the power of management in Windows Server is to have some domain-joined servers that we can manage, as remote management in Windows Server 2012 R2 is enabled by default across a domain. Domain membership is also needed for VDI as we'll see in the next chapter, so we need to build a DC. We already know that we shouldn't add roles like AD to a server running Hyper-V, but we could deploy the AD role to our test VM and set it up as a DC. However, I suggest we make a new VM from scratch and configure this as a DC, if there is room on your server for that, so we can see how to manage another VM and a host. We'll also make this VM a **Dynamic Host Configuration Protocol** (**DHCP**) server so that any new VMs we create can be assigned IP addresses. Finally, we'll add the RSAT tools in here so that we can manage everything from this one VM.

Creating the RDS-DC VM

We can manually create a new VM called `RDS-DC` by following these steps:

1. Create a new internal virtual switch called **Remote Desktop Services** (**RDS**) using the following command:

```
New-VirtualSwitch –Name "RDS Switch" –type internal
```

2. Create a new VHD called `RDS-DC.VHDX` based on `WS2012R2 Sysprep.VHDX` using the following command:

```
New-VHD –Path "E:\TempVMStore\RDS-DC.VHDX" -Differencing –
ParentPath "E:\TempVMStore\ WS2012R2 Sysprep.VHDX"
```

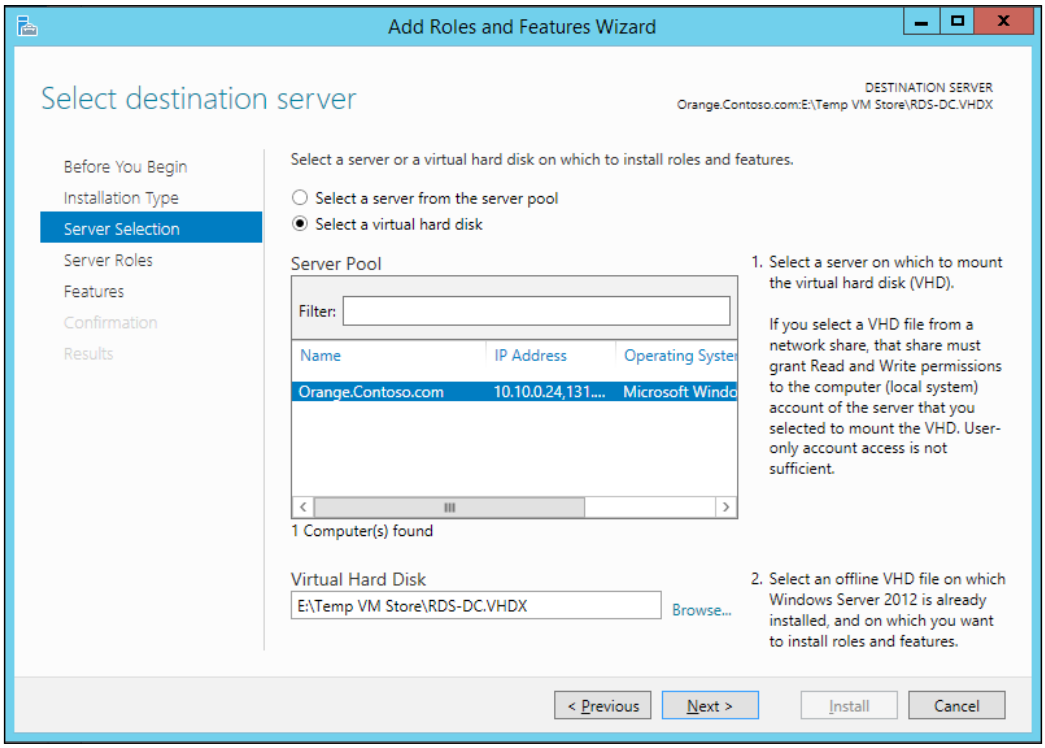
3. Create a new VM as we did for the test VM, but with the following properties:
    - Call it `RDS-DC`
    - Give it at least 512 MB RAM and one logical processor
    - Connect it to the RDS switch we just created

The command line will look like the following:

```
New-VM -Name RDS-DC -VHDPath 'E:\Temp VM Store\RDS-DC.vhdx' -
MemoryStartupBytes 512Mb -BootDevice IDE -SwitchName "RDS
Switch"
```

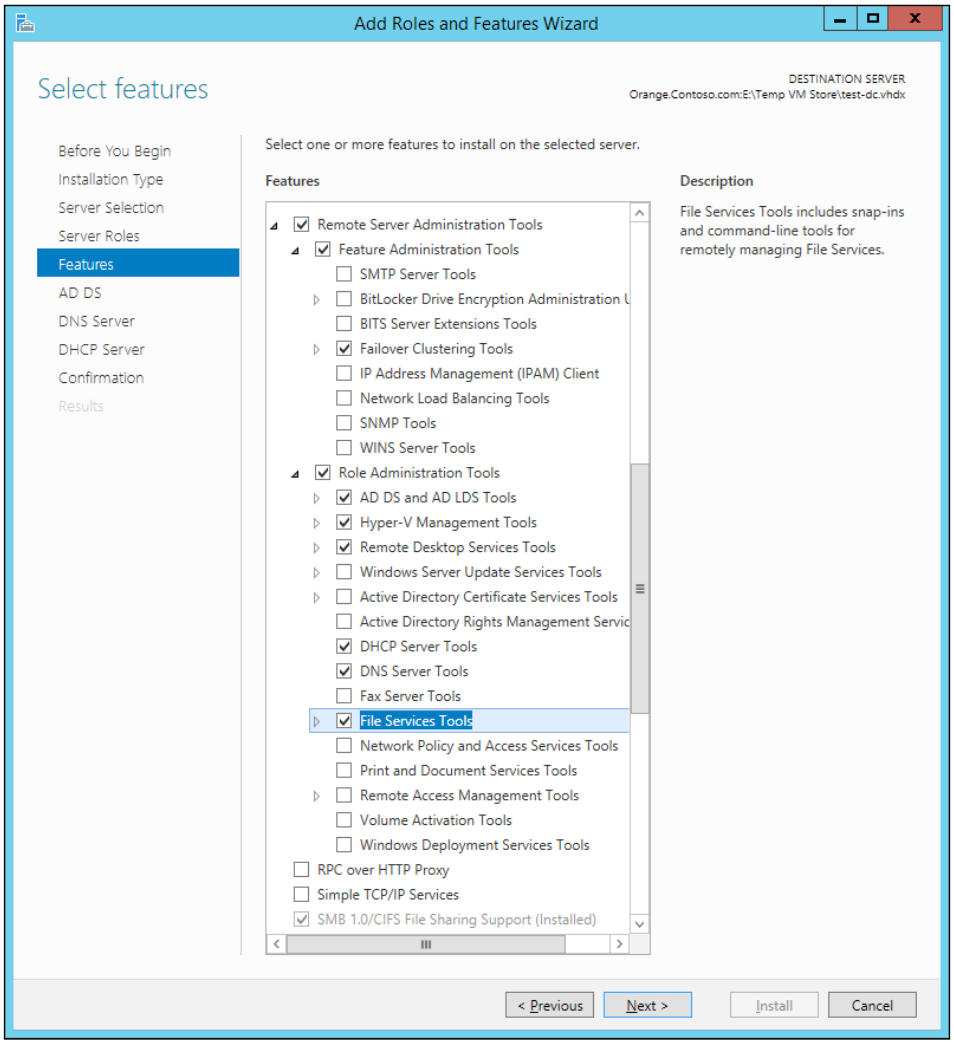
1. Add the roles and features directly into the new VHD that we just made without starting it.

2. From **Server Manager** on the host, go to **Add Roles and Features**. From the **Installation Type** screen, select **Role-based or feature-based installation** and click on **Next**.
3. In the **Select destination server** screen, check the **Select a virtual hard disk** option, as shown in the following screenshot:



4. In this dialog box, we select our physical host to mount the VHD on to (**Orange.contoso.com** in the screenshot), and the path to the VHD we just created. Click on **Next** to continue.
5. From the **Select Server Roles** screen, select **Active Directory Domain Services**, **DHCP Server**, and **DNS Server**. We'll get warnings as we add these roles, such as the need to install DNS and not having a static IP address. These can be ignored as we'll be addressing them once the VM is started. Click on **Next** to continue.

6. From the **Features** screen, expand the **Remote Server Administration Tools** option and select the features as shown in the following screenshot:



7. On the **Confirm installation selections** screen, note that we can save our choices onto an XML file and then use this again to create identical server installations by simply using the PowerShell command `Add-windowsFeature`. We'll do that and save the file as `RDS-DC installation config.xml`. Then click on **Install** to add the features to the VHD.

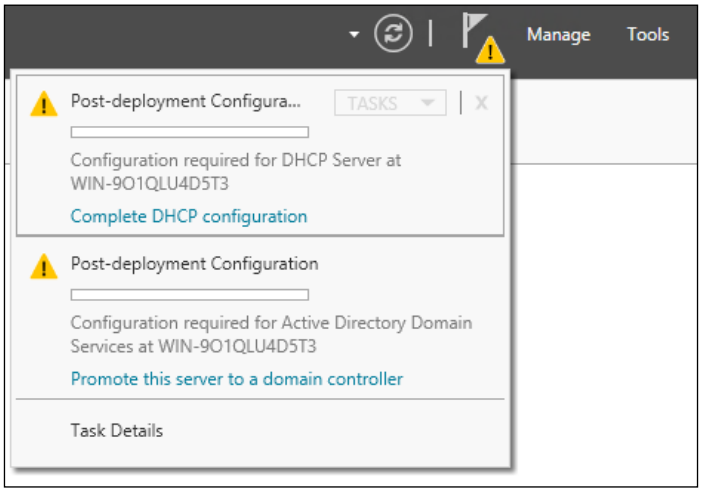
At this point, the new VM has the features installed on it but they aren't configured. The VM has not been turned on and it's still in a sysprepped state. This allows VMs to be configured exactly as we want and stored as templates. We can even patch the OS in the same way, using the Disk Image Servicing Management tool (`DISM.exe`). We can now start to configure the options we have installed by performing the following steps:

1. Start the VM from **Hyper-V Manager** and wait for it to come out of sysprep. You should see the preview screen at the bottom of Hyper-V Manager change from the Windows logo to the initial post sysprep setup screen.
2. Connect to the VM from **Hyper-V Manager** by double-clicking on it.
3. Set the language for your region and accept the license agreement.
4. Set the administrator password (we'll be using `Passw0rd!` as the standard for all passwords). The VM will complete its initial setup.
5. Log in to the VM, and in **Server Manager**, select the local server in the navigation pane and rename the VM to `RDS-DC`. Leave it in a workgroup and reboot it.
6. Configure the network card with a static IP address (192.168.0.1).
7. From **Server Manager**, select the local server and right-click on the link next to **Ethernet IPv4 address assigned by DHCP,IPv6 enabled**.
8. Right-click on the Ethernet adapter and select **Internet Protocol Version 4**. Then click on **Properties**.
9. Select the checkbox **Use the following IP address**. Set the **IP address** field to `192.168.0.1` with a **Subnet mask** value of `255.255.255.0`. Leave the **Default gateway** field blank and set the **Preferred DNS server** field to `127.0.0.1`. Click on **OK** and **Close** to accept these settings and close the **Network Connections** window.

Configuring the new VM as a DC

Now that we have the AD role installed on `RDS-DC`, we can promote it to a DC from Server Manager, while in the earlier versions of Windows, we used DCPromo. We need to perform the following steps:

1. In **Server Manager**, select the yellow warning flag that displays **Promote this server to a domain controller** to open the **Active Directory Domain Services Configuration Wizard** window. Note that DCPromo (`dcpromo.exe`) is gone, and that this is how domain controllers are created in Windows Server 2012 R2. This is shown in the following screenshot:



2. In the **Deployment Configuration** screen, select the option **Add a new forest** and set the root domain name to `Contoso.com`. Click on **Next** to continue.
3. In the **Domain Controller Options** screen, leave all the options as they are and set the **Directory Services Restore Mode** (**DSRM**) password to `Passw0rd!`. Click on **Next** to continue.
4. On the **DNS Options** screen, ignore the warning and click on **Next** to continue.
5. In the **Additional Options** screen, leave the NetBIOS domain name as **Contoso** and click on **Next** to continue.
6. Click on **Next** through the next couple of screens, and in the **Prerequisites Check** screen, ignore the warnings about NT4 and DNS delegation and click on **Install**. The server will now reboot.
7. Reconnect to the `RDS-DC` VM and log in as `Contoso/Administrator`. Then configure the new VM as a DHCP server by selecting the **Complete DHCP Configuration** warning flag in Server Manager, which will launch the **DHCP Post-Install configuration wizard**.
8. Authorize the service in our new domain with the `Contoso/Administrator` domain credentials and click on **Commit**. Close the wizard.

9. From the **Tools** menu in Server Manager, select **DHCP** to open the DHCP MMC snap-in. Expand **rds-dc.contoso.com** in the left navigation pane.
10. Right-click on **IPv4** and select **New Scope** to launch the **New Scope Wizard** window. Call the scope `VDI-Scope` and set its description to `Pool for VDI desktop virtual machines`.
11. In the **IP address** screen, set the **Start** IP address to `192.168.0.100` and the **End** IP address to `192.168.0.254`, and click on **Next**.
12. Click on **Next** on the **Add Exclusions and Delay** and **Durations** screens.
13. On the **Configure DHCP Options** screen, select **Yes, I want to configure these options now** and click on **Next**.
14. Click on **Next** on the **Router** screen.
15. On the **Domain Name** and **DNS Servers** screens, you should see the parent domain already set to **Contoso.com**. The only address we need in the **IP addresses** window is `192.168.10.1` so that the DHCP clients can have the DNS server set as well as being granted an IP address.
16. On the **WINS scope** screen, click on **Next**.
17. On the **Activate Scope** screen, select **Yes, I want to activate this scope now** and click on **Next**.
18. Click on **Finish** to complete the DHCP scope wizard.
19. Refresh the IP address on the host and confirm that the host can get an IP address in this range and can ping `RDS-DC`.

Adding users and groups

Now that we have a DC, it would be good to have some users and groups in there that we can use in our VDI lab setup in the next chapter. We will add users and groups by performing the following steps:

1. Open the **Active Directory Administrative Center** dialog box and create a new **Organizational Unit** (**OU**) called `RDS-VDI`. In this OU, create the following users and groups:
    - Three new users, `RDSUser1`, `RDSUser2`, and `RDSUser3` with password as `Passw0rd!`, and set the password to never expire
    - Two new groups, `VDI-Users` and `Session Users`
2. Add `RDSUser1` and `RDSUser2` to the `VDI-Users` group.
3. Add `RDSUser2` and `RDSUser3` to the `Session-Users` group (note that `RDSUser2` belongs to both groups)

Joining the physical host to the domain

The included script to create a DC stops here, but for the VDI we need to have our physical host in our new domain. Our physical host will be registered in the domain because it will be able to resolve the new DC; it's connected to the internal switch we created and set to use DHCP by default. Our new DC is a DHCP server, and part of this means that the domain and domain controller can be resolved in DNS. So, all we have to do is perform the following steps:

1. In **Server Manager**, on the host, select the local server on the left-hand navigation pane and click on the name of the server to bring up its system properties.
2. Click on **Change** and select the **contoso.com** domain. When asked to enter the domain credentials, use the `contoso/administrator` account with the password `Passw0rd!` to join the domain.
3. Reboot the host. Note that the VMs on the host will restart automatically if they were on when the host is shut down. This is a default that can be overridden in the setting for each VM.

Join the `Test` VM we made earlier to the domain in the same way so that we can contrast managing a physical host with a VM. To do this, we perform the following steps:

1. In **Hyper-V Manager**, on the host, right-click on the `Test` VM we made already and select **Settings**. Change the network adapter to **RDS-Switch**.
2. Connect to the VM and complete the initial setup of this VM (set the local administrator password to `Passw0rd!`).
3. Log in to the `Test` VM and in Server Manager, select the local server in the navigation pane and click on the server name. Rename this server to `RDS-Test` and join it to the `Contoso` domain.

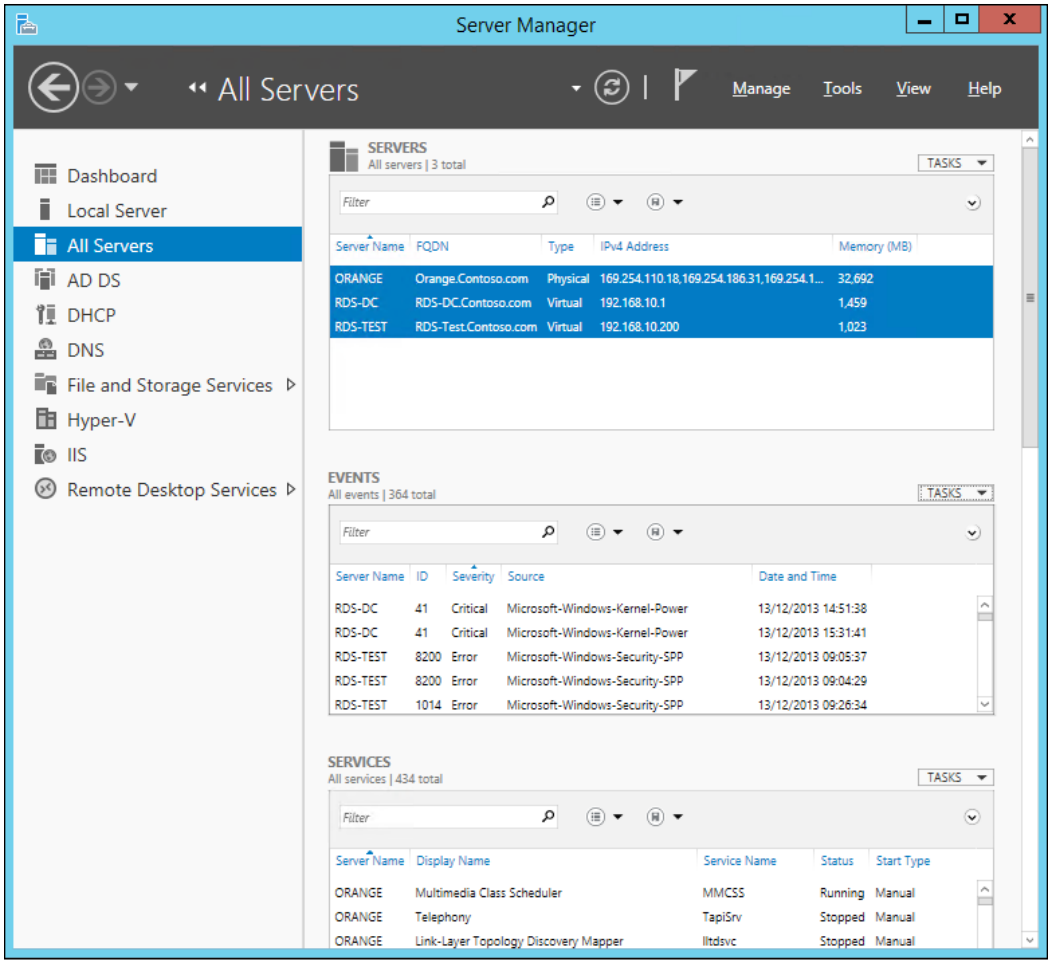
Managing multiple servers in Server Manager

Now we can begin to see the management abilities of Windows Server. We will perform the following steps:

1. Log in to `RDS-DC` and open Server Manager.
2. From the **All Servers** icon on the left-hand side navigation pane, select **Add Servers**.

3. Select all the servers in the `Contoso` domain. Note that we can manage other servers that aren't in our domain if we can discover them and have the credentials to log in to them.

   Now you should see something like the following screenshot:



I can manage all my servers from here. You'll notice that as you added the host server, the **Hyper-V** server group was added to the navigation pane. If we select it, you'll see your host server. If we right-click on it, we can get to Hyper-V Manager because Server Manager knows what roles and features are on each server that it's managing. Hyper-V Manager is only on our DC because we installed RSAT, and if you go to the **Tools** menu in Server Manager, you can see all of the consoles and utilities that are provided.

Underneath any list of servers in Server Manager, there is a complete readout of the servers in a given group, which includes event logs and the roles and features on each server by default. We can also turn on performance counters for our servers and enable the **Best Practices Analyzer** (**BPA**) to get meaningful advice and guidance on any issues that affect all our managed servers.

It's possible to manage earlier versions of Windows Server (2008 and 2008 R2) by adding in PowerShell 3 and Windows Remote Management 3. The only thing we can't do with these older versions is remotely install roles and features.

PowerShell also has extensive support to manage multiple servers, and we have the ability to run remote PowerShell sessions that can persist after a reboot to carry out complex tasks. Many of the operations we carry out in Server Manager are in simple PowerShell scripts running behind the scenes, such as when we promoted the `RDS-DC` VM to being a DC.

Desired State Configuration

While Server Manager can show us what's happening with all our servers, it doesn't actually do anything. It just shows us what problems there might be and gives us the tools to investigate and fix them. One aspect of this is that we often want our servers to stay exactly in the configuration, and with Windows Server 2012 R2, PowerShell 4 can now be used for **Desired State Configuration** (**DSC**). This allows servers to be in a known configuration with the following aspects:

- The roles and features that are enabled
- Files and directories
- Installed software
- Registry settings
- Running services
- Environment variables

DSC can both check and enforce depending on how it is set up. Behind the scenes, a **Microsoft Operations File** (**MOF**) is created based on your script. It's then possible to configure a central server with the DSC service (`Add-WindowsFeature -name DSC-Service`), which can then check the configurations on other servers for application installation files, specific files, and settings. If they are not there, then DSC can replace them to put them back into the required state.

No doubt, Microsoft and third-party vendors such as Dell and Idera will be bringing out tools to simplify the creation of these files to the point where a given template server can be examined, as well as the MOF files and content built from it, so that a series of servers can be kept in a given configuration. That's pretty much what we need to do in VDI, except that we want to manage lots of VMs running desktops and keep them in a desired state.

Summary

In this chapter, you acquired basic knowledge of how Microsoft performs server virtualization with Hyper-V. You saw how to perform an initial setup and configuration of a Hyper-V host, sufficient to create a simple virtual machine running on that host. This allowed us to create a virtual DC that then gave us a quick way to manage our virtualization hosts and other VMs joined to that domain. This is very important because it's so simple to create lots of VMs, and we can only manage them at scale by managing them from one place rather than connecting to each in turn to check their health and configuration. Another key part of the management story is PowerShell, and it's important that we all get used to it as it makes our actions less repeatable and gives us the tools to quickly create VMs with a given set of specifications and keep them in a desired configuration.

Besides being an exercise in itself, all of the practical sections in this chapter provide the foundation we need to create VDI deployment; we need virtualization hosts, we need a domain controller, and we need those hosts to belong to our new domain. We are now ready to start looking at VDI itself and, in the next chapter, we'll examine the different forms of VDI and its architecture. On the way, we'll also create a simple VDI lab setup.

We'll also be using more of PowerShell!

2
Designing a Virtual Desktop Infrastructure

In this chapter, we'll explore what exactly is meant by VDI and how that differs from the remote desktops that were provided by Terminal Services from as far back as Windows NT4, and are still in use today. This will help us choose the right option for our users based on their needs as well as the resources we'll need to provide each type of VDI. We'll then explore the various roles that go into making up Windows VDI; some of these can be run as virtual machines (VMs), but some of which must be real physical hosts running Hyper-V to serve up the virtual desktops themselves. We can then put this theory into practice to create a simple lab setup to offer two different kinds of virtual desktops running on a single host. Finally, we can review what we have created by accessing VDI from a variety of modern devices such as Windows- or Mac-based laptops, as well as Android, iOS, or Windows tablets and phones.

Remote Desktop Services and VDI

What Terminal Services did was to provide simultaneous access to the desktop running on a server / group of servers. The name was changed to **Remote Desktop Services** (**RDS**) with the release of Windows Server 2008 R2, but actually this encompasses both VDI and what Terminal Services was, and it is now referred to as **Session Virtualization**. This is still available alongside VDI in Windows Server 2012 R2; each user who connects to the server is granted a **Remote Desktop Session** (**RD Session**) on an RD Session Host and shares this same server-based desktop. Microsoft refers to any kind of remote desktop as a virtual desktop and these are grouped into collections made available to specific group users and managed as one, and a Session Collection is a group of virtual desktops based on Session Virtualization.

It's important to note here that what the users see with Session Virtualization is the desktop interface delivered with Windows Server, which is similar to but not the same as the Windows client interface, for example, it does have a modern UI by default. In the previous chapter, we saw that we could add/remove the user interface features in Windows Server to change the way it looked. One of these, the **Desktop Experience** option, is there specifically to make Windows Server look more like Windows client, and in Windows Server 2012 R2, if you add this feature option in, you'll get the Windows store just as you do in Windows 8.1.

VDI also provides remote desktops to our users over **Remote Desktop Protocol** (**RDP**), but it does this in a completely different way. In VDI, each user accesses a VM running a Windows client (Windows 8.1 Enterprise edition for this book), and so the user experience looks exactly the same as it would on a laptop or physical desktop. In Windows VDI, the collection of VMs run on Hyper-V and our users connect to them with RDP just as they can connect to the RD Sessions described earlier. The other parts of RDS are common to both as we'll see shortly, but what is important for now is that RDS manages the VDI VMs for us and organizes which users are connected to which desktops and so on. So, we don't directly create VMs in a collection or directly set up security on them. Instead, a collection is created from a template, which is a special VM that is never turned on as the Guest OS is sysprepped, and turning it on would instantly negate that. The VMs in a collection inherit this master copy of the Guest OS and any installed applications as well. The settings for the template VM are also inherited by the virtual desktops in a collection: CPU memory, graphics, networking, and so on. As we'll see, there are a lot of VM settings that specifically apply to VDI rather than for VMs running a server OS as part of our infrastructure.

To reiterate, VDI in Windows Server 2012 R2 is one option in an RDS deployment, and it's even possible to use VDI alongside RD Sessions for our users. For example, we might decide to give RD Sessions for our call center staff and use VDI for our remote workforce. Traditionally, the RD Session Hosts have been set up on physical servers as older versions of Hyper-V, and VMware wasn't capable of supporting heavy workloads like this. However, as we saw in the last chapter, we can put up to 4 TB RAM and 64 logical processors into one VM (physical hardware permitting) and run large RD Session deployments virtually.

Our users connect to our virtual desktop collections of whatever kind with an RDP client, which connects to the RDS servers using RDP.

When we connect to any server with the Microsoft Terminal Services client (`MSTSC.exe`), we are using RDP for this. But, without setting up RDS, there are only two administrative sessions available per server.

Many of the advantages and disadvantages of running any kind of remote desktop apply to both the solutions.

Advantages of remote desktops

Given that the desktop computing for our users is now going to be done in the data center, we only need to deploy powerful desktops and laptops to those users who are going to have difficulty connecting to our RDS infrastructure. Everyone else could either be equipped with thin client devices, or given access from other devices they already have if working remotely, such as tablets or their home PCs and laptops. Thin client computing has evolved in line with advances in remote desktop computing, and the latest devices from 10ZiG, Dell, and HP among others support multiple high-resolution monitors, smart cards, and webcams for unified communications, which are also enabled in the latest version of RDP (8.1). Using remote desktops can also reduce overall power consumption for IT, as an efficiently cooled data center will consume less power than the sum of thin clients and servers in an RDS deployment. In one case, I saw that this saving resulted in a green charity saving 90 percent of its IT power bill.

Broadly speaking, managing remote desktops ought to be easier than managing their physical equivalents. For a start, they'll be running on standard hardware, so installing and updating drivers won't be so much of an issue. RDS has specific tooling for this, which we'll see in *Chapter 5*, *High Availability*, to create a largely automatic process for keeping our collections of remote desktops in a desired state. VDI doesn't exist in a vacuum, and there are other Microsoft technologies to make any desktop management easier with other types of virtualization as follows:

- User profiles have long been a problem for desktop specialists. RDS in Windows Server 2012 introduced the concept of **User Profile Disk** (**UPD**), which are special differencing disks specifically available to store users' settings. These disks are then reattached when a given user logs back in, and all their files' application settings are available. These can be used with both Session Virtualization and VDI, but are not interchangeable across the two. If we want our users to roam between RDS and physical desktops, there are the traditional techniques such as folder redirection and new ones such as **User Environment Virtualization** (**UE-V**), and we'll look at that in detail in *Chapter 8*, *Managing User Profiles and Data*.

- **Application Virtualization** (**App-V**) allows us to deploy applications to the user who needs them when and where they need them, so we don't need to create desktops for different types of users who need special applications; we just need a few of these and only deploy generic applications on these and those that can't make use of App-V. This is covered in detail in *Chapter 9*, *Virtual Applications*. Even if App-V is not deployed, VDI administrators have total control over remote desktops, as we have many number of security techniques at our disposal. In the worst case scenario, we can remove any installed applications every time a user logs off!

The simple fact that the desktop and applications we are providing to our users are now running on servers under our direct control also increases our IT security. Patching is now easy to enable particularly for our remote workforce, as when they are on site or working remotely, their desktop is still in the data center. RDS in all its forms is then an ideal way of allowing a **Bring Your Own Device** (**BYOD**) policy. Users can bring in whatever device they wish into work or work at home on their own device (WHOYD is my own acronym for this!) by using an RDP Client on that device and securely connecting with it. Then there are no concerns about partially or completely wiping users' own devices or not being able to, because they aren't in a connected state when they are lost or stolen.

VDI versus Session Virtualization

So, why are these two ways of providing remote desktops in Windows there and what are the advantages and disadvantages of each? First and foremost, Session Virtualization is always going to be much more efficient than VDI. It's going to provide more desktops on less hardware. This makes sense if we look at what's going on in each scenario. If we have 100 remote desktop users to provide for, the following techniques could be adopted according to the two ways mentioned earlier:

- In Session Virtualization, we are running and sharing one operating system and this will comfortably sit on 25 GB of disk. For memory, we need roughly 2 GB per CPU, and we can then allocate 100 MB of memory to each RD Session. So, on a quad-core CPU Server, for 100 RD Sessions we'll need 8 GB + (100 MB*100) of memory, so less than 18 GB RAM.

We'll look at scale and performance for Session Virtualization in more detail in *Chapter 6*, *Scale and Performance*.

- If we want to support those same 100 users on VDI, we need to provision 100 VMs each with their own OS. To do this, we need 2.5 TB of disk and if we give each VM 1 GB of RAM, then 100 GB of RAM is needed. This is a little unfair on VDI in Windows Server 2012 R2, as we can cut down on the disk space need and be much more efficient in memory than this. But, even with these new technologies, we would need 70 GB of RAM and say circa 400 GB of disk.

Remember that with RDS our users are going to be running the desktop that comes with Windows Server 2012 R2 and not Windows 8.1. Our RDS users are sharing one desktop, so they cannot be given administrative rights to that desktop. This may not be that important for them but can also affect what applications can be put onto the server desktop, and some applications just won't work on a Server OS anyway. Users' sessions can't be moved from one session host to another while the user is logged in. So if we need to take a session host down for any reason, we may need to do this out of office hours to patch session hosts and update the applications on them, if we want to avoid interrupting our users' work.

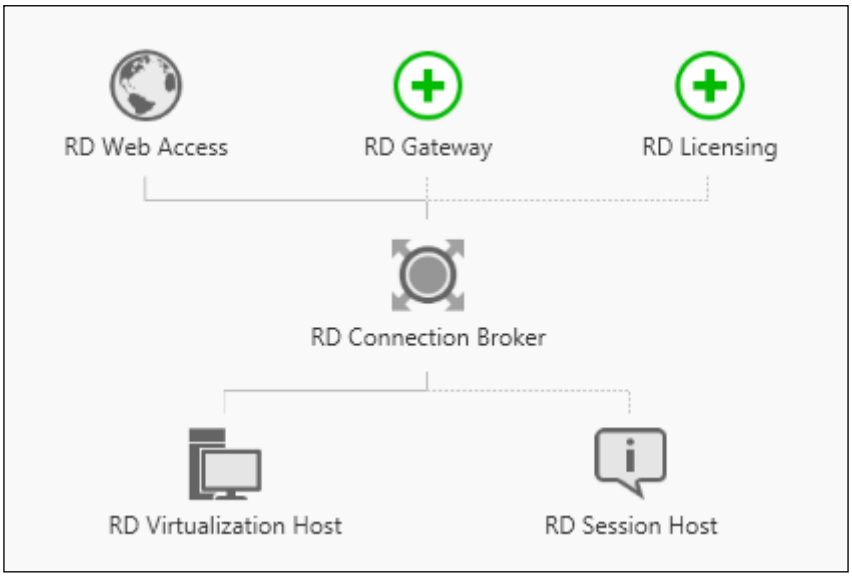
VDI on the other hand means that our users are using Windows 8.1, which they will be happier with. This may well be the deciding factor regardless of the cost, as our users are paying for this and their needs should take precedence. Application installations are the same as they would be for a physical desktop, where certain software won't install onto a server-based OS or work when we use a server for Session Virtualization. It can be easier to manage VDI as well; but, this will only apply if we are good at managing desktops in the physical world and are using the right tools for managing desktops such as App-V and UE-V in the **Microsoft Desktop Optimization Pack** (**MDOP**) suite.

Remote applications in RDS

Another part of the RDS story is the ability to provide remote access to individual applications rather than serving up a whole desktop. This is called RD RemoteApp and it simply provides a shortcut to an individual application running either on a virtual or remote session desktop. I have seen this used for legacy applications that won't run on the latest versions of Windows and to provide access to secure applications, as it's a simple matter to prevent cut and paste or any sharing of data between a remote application, and the local device it's executing on. RD RemoteApps work by publishing only the specified applications installed on our RD Session Hosts or VDI VMs.

VDI roles

VDI is made up of a number of roles. This is very clear once we have set it up as we are presented with a deployment overview screen in Server Manager in Windows Server 2012 R2, as shown in the following screenshot:



This diagram is not only an architecture overview of how RDS works but it's also used to manage our deployments. Now we can see what the roles are. Let's look at what each of them does in more detail.

Remote Desktop Virtualization Host

This is a Windows Server with the Hyper-V role or the free Hyper-V Server just as we set up in *Chapter 1*, *Putting the V in VDI - An Introduction to Virtualization in Hyper-V*. It's where our VDI VMs will be run from. We saw how to set this up in the previous chapter and how to create a VM to run on it. When we set up RDS, we declare which of our servers will host these VMs, as the RDS service will manage these servers directly and allow us to deploy collections of VMs and manage which user is connected to which VM in which collection. While this host must obviously be one or more physical servers, all of the other roles in RDS can be run inside VMs. These VMs could also be run on the same physical server as is used in a **Remote Desktop Virtualization Host** (**RD Virtualization Host**), so alongside the VDI VMs and that's what we'll do in our lab. Note that we should use Hyper-V Manager to control the VM running the RDS roles, but the VDI VMs should be managed from the RDS console.

Remote Desktop Connection Broker

Creating virtual machines one by one across a group of physical hosts is fairly easy to script in PowerShell, but managing and maintaining them is difficult without some sort of tool. We also need to allow users to access them and do this in a way that each user gets their own desktop or application backed by one of those VMs. This orchestration function is exactly what **Remote Desktop Connection Broker** (**RD Broker**) does. It's a role that can be assigned to any Windows Server and can co-exist with all the other RDS roles but one—we have already established that no other roles should be installed on a physical server running Hyper-V, so in this case our RD Virtualization Host.

If we think of RDS as a classic three-tier service, then the RD Broker is the middle tier with all the business logic and deployment metadata in it. So when we make changes to RDS via the browser or using PowerShell, this is where the configuration changes we make are stored. The RD Broker is also the gatekeeper of RDS and controls who has access to what and who is currently using both VDI and RD Sessions. It has a simple load-balancing mechanism so that users' sessions are distributed across the session hosts in a given collection.

Remote Desktop Web Access Server

The **Remote Desktop Web Access Server** (**RD Web Access Server**) is the web frontend of RDS where users can access the desktop they have permission to. Both VDI and RD Sessions can be made accessible from here, and really, all this role service does is to present what is in the RD Broker to the users. RD RemoteApps are also surfaced on the web portal and one nice touch introduced in Windows 7 is that if you configure your user's desktop to point to the web portal, any application a user has access to will just show up in the start options alongside the real applications on their desktops.

Remote Desktop Gateway

This is a variation on the Web Access role, specifically designed to allow users to access remote desktops over the Internet without the need to create a VPN connection. This works by running the RDP protocol over HTTPS to ensure the connection is encrypted and secure. It is possible to deploy the Gateway with one or two firewalls and connect to an **Active Directory Domain Services** (**ADDS**) in different ways as well.

Remote Desktop licensing server

This server is there to pass out the licensing **Client Access Licenses** (**CAL**) to the rest of the RDS infrastructure. We'll take a look at this role in more detail in *Chapter 10, Licensing and the Future of VDI*, but essentially it stores the license keys for our CALs and integrates with AD to hand these out as our users' login to RDS.

Remote Desktop Session Host

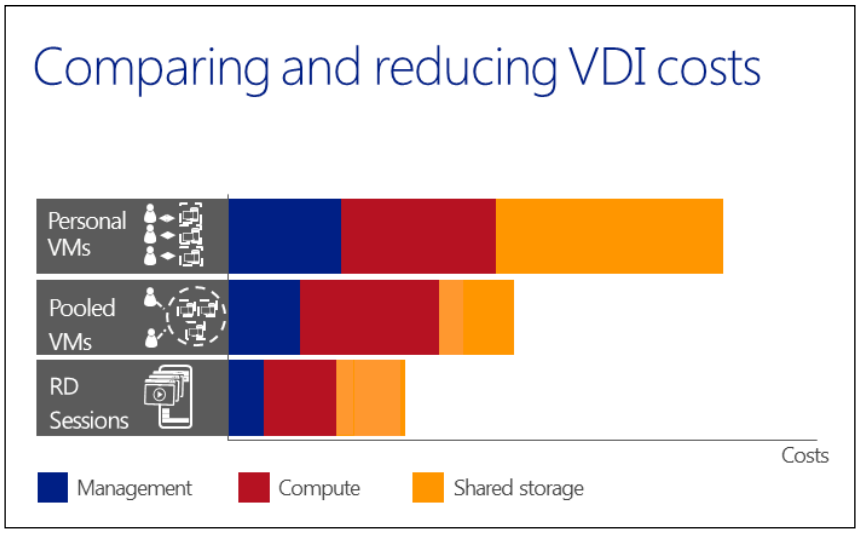
This is the server role that provides Session Virtualization via RD Sessions that we have already compared to VDI. Traditionally, these have been physical servers because they demand a lot of CPU, RAM, and I/O. But with the much higher limits for these in Hyper-V, we now have the option to run them as VMs with a small loss of performance running them as Hyper-V VMs.

Types of VDI collections

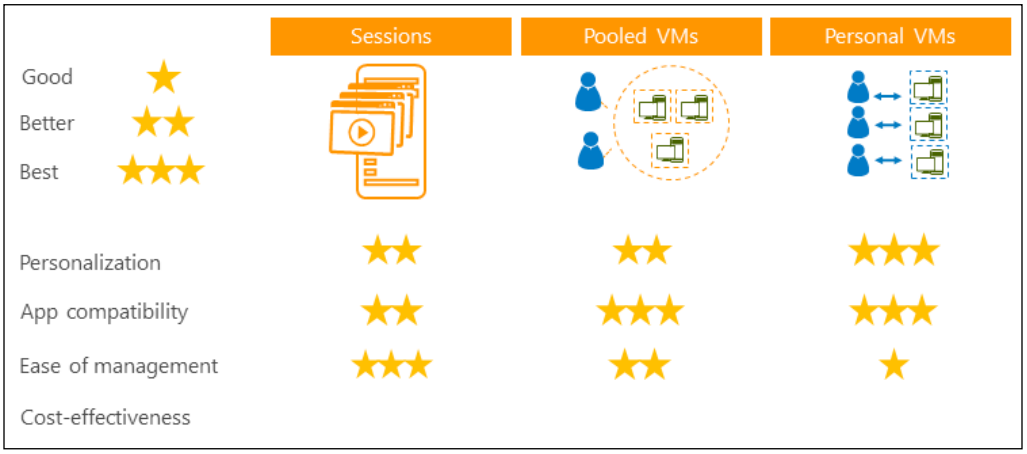
There are two basic types of VDI collections, Personal and Pooled, alongside the Session Collections in Session Virtualization. Each type of collection can be automatically managed by the RDS infrastructure as follows:

- **Personal Collections**: These are created for each of our users, and whenever they log in, they get the same VM. All we have done here is to move the OS running on a physical desktop into a VM on our data center. Because this is so similar to real desktops, it gives our user the best personal experience; but, at the same time, it is most expensive to run and maintain.
- **Pooled Collections**: These are much like a pool car system where a company buys a set of identical vehicles, which employees can sign out and use as they need. When they are returned, they are cleaned and refueled to be ready for the next user. In the same way, Pooled VMs are signed out as users request them and they won't get the same VM the next time they log in. Also, these VMs will typically be reset to a starting state after the user logs out, ensuring that each user gets the same experience each time. A collection of these Pooled VMs can be managed as one VM as far as patching and updating is concerned, and we don't need to create a VM for each user; just sufficient VMs for the maximum concurrent user demand.

If we compare the costs of these two types of VDI against Session Virtualization, we can see that Pooled VMs offer a useful compromise on cost. This is shown in the following screenshot:



While at the same time, Pooled VMs offer a balance of usability for our users and application compatibility, as shown in the following screenshot:

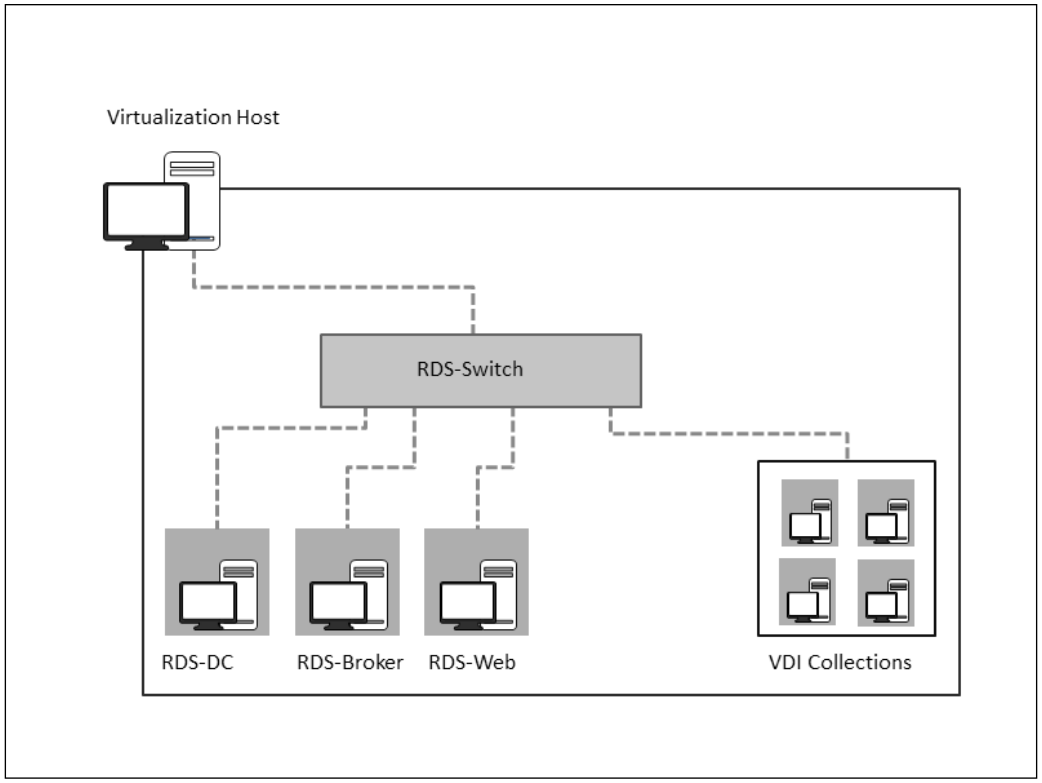


It's alright to combine Pooled and Personal Collections into one VDI deployment and to have more than one of each type to provide collections suited to particular groups of users. Whichever type of VDI collection you decide to create, you'll need to create this from a virtual desktop template. This is simply a VM built around the client OS (in our case, Windows 8.1 Enterprise Edition) configured as required and sysprepped. The VM settings such as memory, CPU, and networking are inherited by all the VMs in the collection, and the VM is never actually turned on, as the OS would come out of sysprep.

You could also include applications in a virtual desktop template, but then we need endless collections to provide the right applications to each group of users. However, there is a much better way, and that's to use App-V to stream applications to users when they log in to a VDI desktop, a real desktop, or an RD Session. We'll see how this works in *Chapter 8*, *Managing User Profiles and Data*.

Getting started with VDI

The various RDS roles can be combined for smaller deployments, and one option in the RDS deployment wizard is to put the RD Web Access, Broker, and Hosts of whichever type onto one server (a quick deployment). That might be acceptable for Session Virtualization, but the virtualization host role is Hyper-V, which should only be used for this. So, for our first foray into VDI, it would be good to have a semi-production look and feel and do things a bit more manually to see what's going on. Our target is to build a simple VDI deployment, as shown in the following diagram:



We have already done some of the setup we need in the *Creating the RDS-DC VM* section of *Chapter 1*, *Putting the V in VDI - An Introduction to Virtualization in Hyper-V*. We have the RDS-DC and RDS virtual switches and our physical host is domain-joined. We also have the DHCP role configured with a scope, which we can use to assign IP addresses to our VDI VMs. What we need to do next is to create the other two VMs in the diagram for the RDS Broker and RD Web Access roles. We could do this manually for each of these VMs in the same way as we did in the *Creating a simple virtual machine* section of *Chapter 1*, *Putting the V in VDI - An Introduction to Virtualization in Hyper-V*. However, I have included a PowerShell script for this on my blog and also at the end of the book to do all of this for you. This will create a further VM RDS-Session Host, which will be used to create a Session Collection later in this chapter.

Creating the virtual desktop template

We also need to create a virtual desktop template that will be the basis for our first VDI collection. The quickest way to create a basic virtual desktop template is to use the same `Convert-WindowsImage.ps1` PowerShell script that we used in the previous chapter to make a VHD based on Windows Server 2012 R2. The only difference is, this time we will create a Windows client VHD by running it against an evaluation edition of Windows 8.1 Enterprise edition (`http://technet.microsoft.com/en-us/evalcenter/hh699156.aspx`), which is the ISO file referred to in the following script:

```
<#Variables for the name of the DC and where the VM is going to be
stored, and what Virtual Switch it's connected to #>
$VMName = "RDS-VDITemplate"
$VMSwitch = "RDS-Switch"
$WorkingDir = "E:\Temp VM Store\"
$VMPath = $WorkingDir + $VMName
$SysPrepVHDX = $WorkingDir + $VMName +"\RDS-VDITemplate.VHDX"
<# Housekeeping to delete the VM and associated files from the last time
this was run:
  #1. delete the VM from Hyper-V #>
  $vmlist = get-vm | where vmname -in $vmname
  $vmlist | where state -eq "saved" | Remove-VM -Verbose -Force
  $vmlist | where state -eq "off" | Remove-VM -Verbose -Force
  $vmlist | where state -eq "running" | stop-vm -verbose -force -Passthru
| Remove-VM -verbose -force
  #2. get back the storage
  If (Test-Path $VMPath) {Remove-Item $VMPath -Recurse}
# Create the VHD from the Installation iso
md $VMPath
```

```
cd ($WorkingDir + "resources")
.\Convert-WindowsImage.ps1 -SourcePath ($WorkingDir +"Resources\
9600.16384.WINBLUE_RTM.130821-1623_X64FRE_ENTERPRISE_EVAL_EN-US-IRM_CENA_
X64FREE_EN-US_DV5.ISO") -Size 100GB -VHDFormat VHDX -VHD $SysPrepVHDX
-Edition "Enterprise"
#Create the VM itself
New-VM –Name $VMName –VHDPath $SysPrepVHDX -SwitchName $VMSwitch -Path
$VMPath -Generation 1 -BootDevice IDE
<# feel free to change these settings to tune your VDI VMs and remember
each VM will get the same settings as these#>
Set-VM -Name $VMName –MemoryStartupBytes  1024Mb
Set-VM -Name $VMName -DynamicMemory
Set-VM -Name $VMName -MemoryMinimumBytes  512Mb
Set-VM -Name $VMName -AutomaticStartAction StartIfRunning
Set-Vm -Name $VMName -AutomaticStopAction ShutDown
Set-Vm -Name $VMName -ProcessorCount    2
```

Having created the virtual desktop template with the preceding script, we need to be aware of the following aspects:

- This machine needs to be left in a sysprepped state, and so it must not be started.
- Any settings we make for memory CPU and so on will be inherited by all the VDI VMs in our collection. We need to be particularly careful with memory as Hyper-V doesn't overcommit memory, so we need to be aware that the number of VMs in our collection multiplied by the memory setting for this template need to be less than we have on our host, and also factor in the three other VMs we'll be running. So, the `Set-VM .. –MemoryStartupBytes` and `-MemoryMinimumBytes` settings in the above script will need to be planned carefully.

We also need to understand what's happening when a user logs into our VDI development. They'll be given some way of navigating to the RDS Web Access portal, where they'll log in. The Broker service will assign them a virtual desktop—either their own in a Personal Collection, or a random virtual desktop from a Pooled Collection. When a user logs in each time, they'll see that their desktop is the way they left it when they last used it; that's pretty simple to understand because a Personal Collection gets the same virtual desktop each time. This is where the UPD mentioned earlier comes into play. RDS mounts these automatically for us when a user logs in and intercepts which settings we want to store on the UPD, so we don't have to do any configuration of the desktop itself, such as to modify application paths and user profile paths. When a user then logs out of a virtual desktop in a Pooled Collection, the VM underpinning the desktop is rolled back to an initial checkpoint and the UPD is disconnected from it.

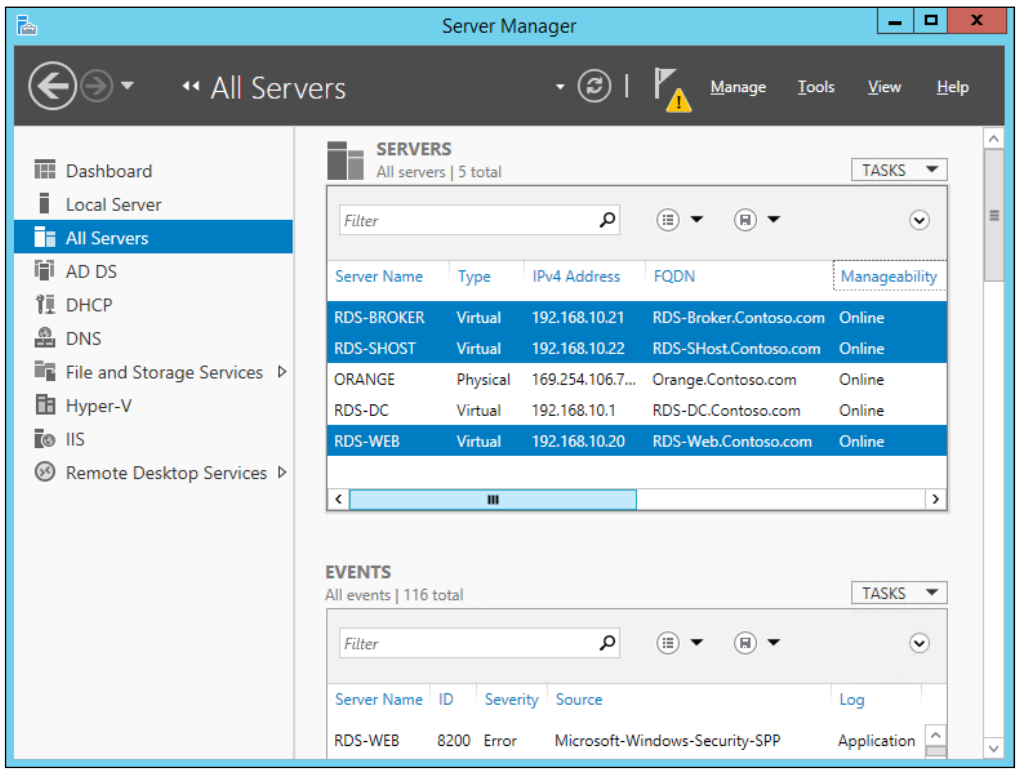
Next time they log in, their UPD is then simply attached to the VM running their next virtual desktop. UPDs do not capture any application users might install in a session; and if that is a requirement, then Personal Collections will allow for that. Therefore, in Pooled Collection, we want to set up our users to not have this ability possibly by using the AppLocker features of **Group Policy** to prevent this.

Setting up and configuring the RDS roles

In order to use the RDS deployment wizard, we need to manage all of the servers we are going to use. For this, we need to perform the following steps:

1. On the physical host, open **Hyper-V Manager**. Right-click on the **RDS-DC** VM and click on **Connect**.
2. Open **Server Manager** and navigate to **Manage** | **Add Servers** from the menu.
3. Add in all the VMs created by the script (`RDS-Broker`, `RDS-Web`, and `RDS-SHost`) and your physical host.

You should now see those servers in **Server Manager** as shown in the following screenshot:

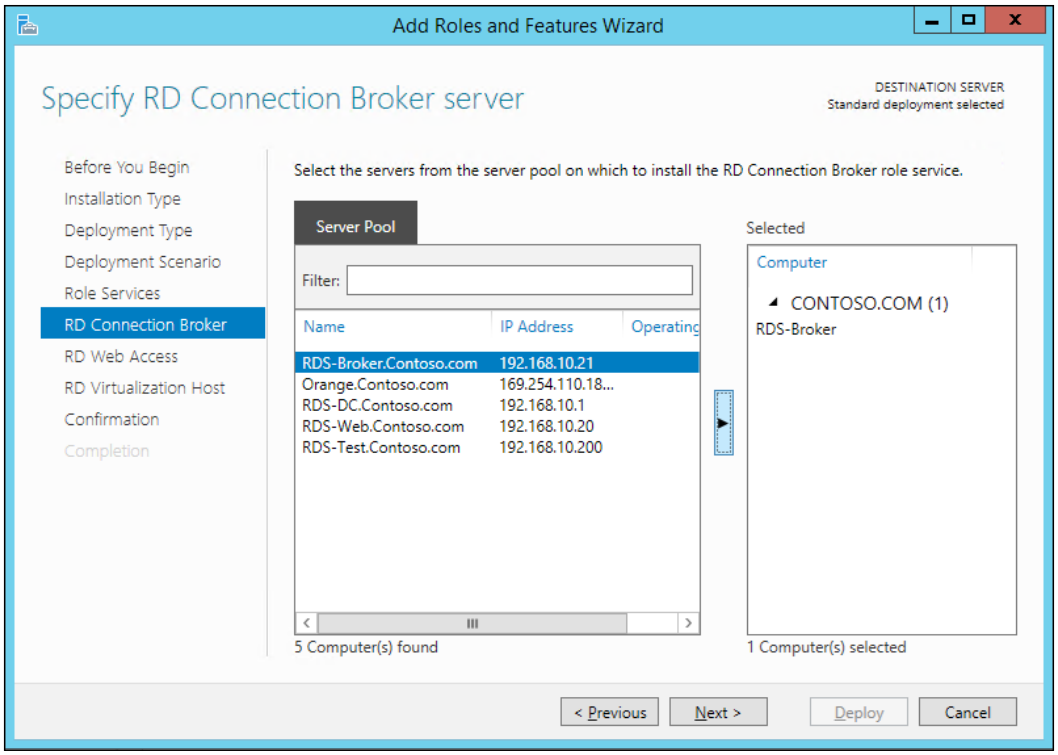


Here, I have changed the columns displayed and my host is called **Orange** (note that it says **Physical** in the **Type** column).

While **Remote Desktop Services** appears on the navigation pane, here that's only because our DC is running **Remote Server Administration Tools** (**RSAT**); so, the tools are there but aren't configured.

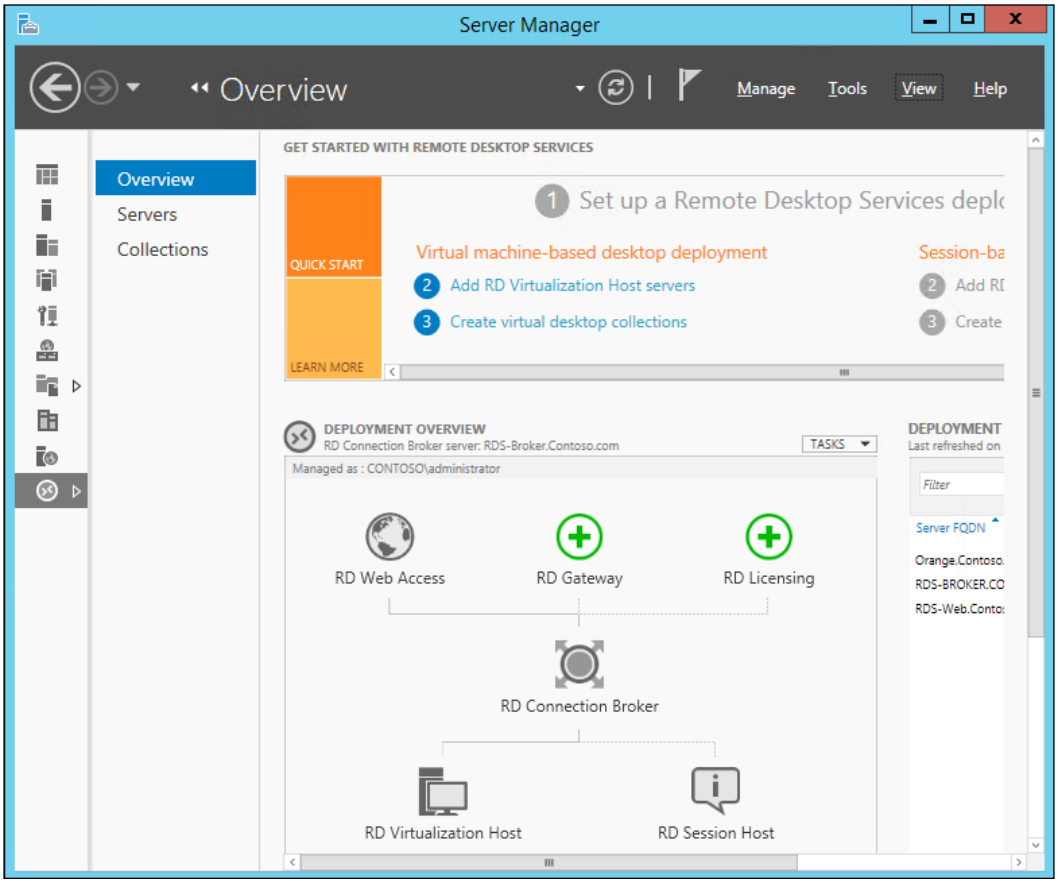
Now, we are in a position to run the RDS deployment wizard, which will allow us to provision the roles on our Broker, Web Access Host, and Virtualization Host in one go, and get us to the point where we can create our first VDI collection. For this, we need to perform the following steps:

1. On the DC (RDS-DC), open **Server Manager**, and from the menu, navigate to **Manage | Add Roles and Features**.
2. Click on **Next** to skip to the **Before You Begin** screen.
3. On the **Select installation type** screen, there is a special option, **Select Remote Desktop Services Installation**, to orchestrate our deployment. So, select this option and click on **Next**.
4. In the **Select deployment type** screen, there are two ways to set up RDS: standard and quick. Quick puts all of the roles and features onto one host (which must be a physical host for VDI) and creates a collection automatically. This is definitely not the best practice and is automated, that is, it's hard to see what the options are. So, we will select a standard deployment as you would do for a production environment of any size. Click on **Next** to continue.
5. In the **Select deployment scenario** screen, we can see that RDS allows us to select the two desktop solutions we want to use. As we'll see later, we can combine these into one infrastructure, but for now, select the **Virtual machine-based desktop deployment** option and click on **Next**.
6. In the **Review Role Services** screen, click on **Next** as we have already discussed what these are.
7. In the **Specify RD Connection Broker server** screen, select **RDS-Broker** from the **Server Pool** section shown in the upcoming screenshot.
8. Only servers already managed by Server Manager will show up in the **Server Pool** section.
9. Click on the arrow to the right of the pool to add this server to the selected column, as shown in the following screenshot:



10. Then click on **Next** to continue.
11. In the **Specify RD Web Access server** screen, select **RDS-Web** from the **Server Pool** section and again click on the arrow to the right of this column to add the server to the selected column. Click on **Next**.
12. In the **Specify RD Virtualization Host server** screen, select your physical host. In my lab, that's a laptop called **Orange** that's running all of my VMs. Click on **Next**.
13. On the **Confirm selections** screen, click on the **Restart the destination server automatically if required** checkbox. This will only force a restart if the Hyper-V role is not already installed on the physical server used as an RD Virtualization host.
14. Click on **Deploy** to complete the wizard. Behind the scenes is a PowerShell Script, which, on larger deployments, will perform the installations in parallel across the various servers selected. Click on **Close** when the wizard has finished.

We can now see what it has done in the following screenshot by clicking on that previously unconfigured **Remote Desktop Services** option in **Server Manager** on our DC:



The diagram in the center, which we have already seen, is not just a diagram. We can hover over it to configure it, and the green crosses indicate that we haven't set up licensing or a Gateway server yet. Confusingly, we haven't set up an **RD Session Host**, which is also in gray like the roles we have configured. But, it's connected by a dotted line to the rest of the infrastructure, and it's the only icon here that isn't a hyperlink to install that feature. The wizard has also created some default settings and before we create a collection we should review and change some of these by navigating to **Tasks | Edit deployment options** above the **Deployment Overview** diagram.

At the moment, we only need to worry about three of these options as follows:

- **RD Web Access**: This shows us the web page from which users can connect to our collections; in our case, `http://rds-web.contoso.com/RDWeb`. Following the link now will be pointless as we haven't created a collection, so all we'll get is a blank page. If you do follow that link, you'll get a certificate error (certificates will be covered in *Chapter 7*, *Maintenance and Monitoring*), which you can ignore. You can then log in as an administrator to see there's nothing there yet.
- **Active Directory**: We do want all the VMs in our collection to belong to a discrete **Organizational Unit** (**OU**) RDS-VDI. There's a warning when we elect to do this, which occurs because the RDS Broker needs permission to create computer entries in this OU for the VMs in our VDI collections. Click on **Apply** after doing this, and note that it's possible to capture the PowerShell script to do this as follows:

  ```
  Grant-RDOUAccess -ConnectionBroker "RDS-Broker.Contoso.com" -
  OU "RDS-VDI"
  ```

- **Export Location**: The first thing that will happen when we create a collection is that the virtual desktop template will be exported from Hyper-V to a specified share. This is because it will then be copied multiple times for each VM, and in a production environment, this would be copied across the multiple virtualization hosts running a collection. By default, this is a share on the broker (in our case, `\\RDS-BROKER\RDVirtualDesktopTemplate`), and you may wish to change this to a more appropriate location because as it stands, it's inside a VM which may have limited space.

We also need to think about where all our virtual desktops will be stored. For our lab, we might only have one server available, and so we can just store these locally. However, in larger deployments, it's possible to host either one of the following:

- **Clustered Shared Volumes (CSV)**: These are shared storage fronted by a Windows Server Failover Cluster (which we'll shorten to Cluster)
- **File share**: This is not the simple share that you create by right-clicking on a folder and selecting **Share**; it's an advanced file share for applications created on a server configured as a File Server from Server Manager. File Servers in Windows Server 2012 R2 can be scaled out for high availability and exhibit many of the features of a SAN, such as deduplication and tiered storage.

Finally, there are a couple of prerequisites, as follows that we need to configure before we can create our first collection:

- Our virtualization host must have an IP address on one of its physical switches that we'll need to physically connect our server to a network. This shouldn't be a problem as when we created our VMs, we used an internal virtual switch, which effectively isolated our VDI from any other physical or virtual servers on any network that the physical host is connected to. In a real environment, we would want to expose VDI to the internal network and possibly the Internet for remote workers.
- We need to create a simple share to store the UPDs that we are going to create. So, in the directory where the VMs for our labs are stored, create a folder called `VDI-UPD` and share it (in my case, that will be on my Orange laptop so `\\Orange\VDI-UPD`).

Creating a Pooled Collection

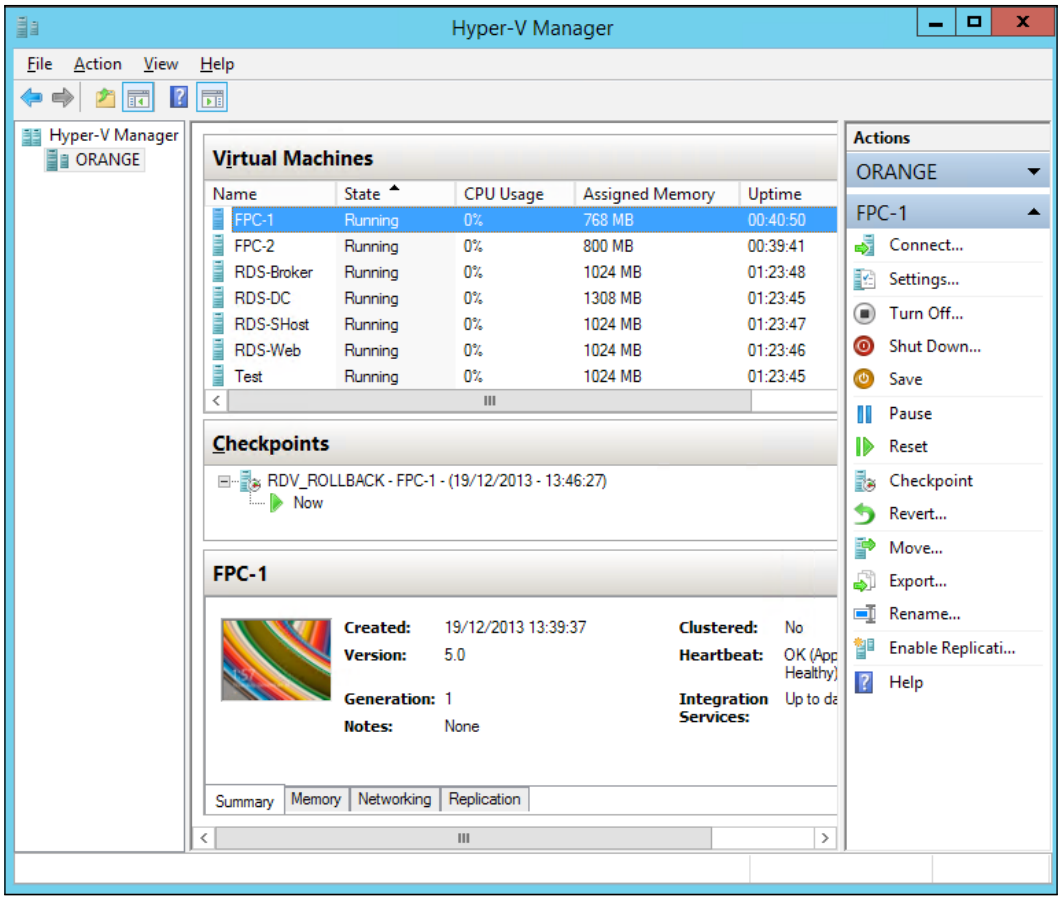
We can now create our first collection and given that the most efficient type of VDI is a Pooled Collection, this is the one we'll create first. To get started, we need to go back into Server Manager on our RDS-DC domain controller and go to the RDS **Overview** tab. From there, we can launch the **New Collection** wizard by right-clicking on our virtualization host in the **Overview** diagram and selecting **Create New Virtual Desktop Collection**. To complete the wizard, perform the following tasks:

1. Read the **Before You Begin** screen and click on **Next**.
2. In the **Name the Collection** screen, give the collection a name such as `First Pooled Collection` and optionally a description, and then click on **Next**.
3. In the **Collection Type** screen, select a Pooled Collection. Leave the **Automatically create and manage virtual desktops** option checked and click on **Next**.
4. In the **Specify the virtual desktop template** screen, select the **RDS-VDITemplate** template that we made earlier and click on **Next**.
5. In the **Desktop Settings** screen, note that we can provide an existing sysprep unattend answer file (typically called `unattend.xml`, which I'll refer to as an answer file), which would contain all of the necessary information about how to configure the desktops as they come out of sysprep. We'll go into this in the next chapter, but for now we can enter the key settings in the wizard by selecting **Provide unattended installation settings** and clicking on **Next**.

6. In the **Unattended installation settings** screen, select your local time zone and, then select **Contoso.com** as the domain and **RDS-VDI** as the OU we made earlier. Click on **Next**.
7. In the **Specify users and groups** screen, we can select who will have access to use the collection. If you have used the supplied script to create the `RDS-DC` domain controller, you can remove all the users and add `Contoso / VDI Users`. This screen also allows us to set how many virtual desktops we want to create and how they will be named. Leave the number at **2** for now, and change the prefix to `FPC-` and the suffix to `1` (so our VMs will be called `FPC-1` and `FPC-2`). Click on **Next**.
8. In the **Specify virtual desktop allocation** screen, we could spread our virtual desktop VMs across our virtualization hosts, but since we only have one host, we can just click on **Next** to continue.
9. In the **Specify virtual desktop storage** screen, we can decide where our VMs are stored. Since we only have a single host, we can select **Store on each RD Virtualization Host server**, and enter the path to the location where our VMs are stored (which in my lab setup is `E:\Temp VM Store`). There is also an option to store the parent disk in a separate location. A shared SSD would be a good location for this, as it will be read often and only overwritten when the collection is patched. Finally, notice the option to roll back the collection when the user logs off. This will automatically revert the underlying VM to an initial checkpoint (snapshot). Leave this option checked and click on **Next**.
10. In the **Specify user profile disks** screen, we can opt to use UPDs and specify the share where they'll reside. In our case, this is the share we created earlier, that is, `\\servername\VDI-UPD`. Click on **Next** to continue.
11. We can check all our settings in the **Summary** screen and click on **Finish** to complete the wizard.

This wizard will take about 10 minutes to run, and we can see what's happening behind the scenes by opening **Hyper-V Manager** on our host server. The **RDS-VDITemplate** VM will be exported from Hyper-V to the template share we specified in the wizard. This will be used to create the two virtual desktops we asked for (`FPC-1` and `FPC-2`). These will appear as VMs in Hyper-V, and we can see that they will be started by the Collection creation wizard then bought out of sysprep and configured with the settings we specified (time zone, domain, OU, and so on).

Once they have been configured, the wizard will then restart them, as it would happen as part of the sysprep process. Then, they'll be checkpointed so that when a user logs off, they'll revert to this initial state, as you can see in the following screenshot of **Hyper-V Manager**:



If we then look at what's on the disk, we'll see that in our VM store there will be a folder named as `Fast_Pooled_Coll`, which will contain two of our VMs and a local copy of the virtual desktop template in a folder called `IMGS`. In the share we created for our UPDs, there is a `UVHD-template` VHD, which is a thin-provisioned disk, set to the size we specified in the wizard. Returning to the RDS overview on our `RDS-DC`, we can see our fast-pooled collection underneath the virtualization host on the **Overview** diagram. If we navigate to **Collections** | **Fast Pooled Collection** in the navigation pane on the left, we can see our collection with its two virtual desktops and the status **Running**.

VDI always leaves two spare virtual desktops as running so that users can log in quickly. The surplus will be put into a saved state to save resources.

We can now test our virtual desktop by connecting to our Web Access server by performing the following steps:

1. Open Internet Explorer on our physical host and go to `https://rds-web.contoso.com/RdWeb`.
2. We'll get a certificate warning, which we can ignore, and then we'll be asked to sign on. Use `RDSUser1/Passw0rd!`.
3. Click on the Fast-Pooled Collection.
4. On the **Remote Desktop Connection** screen, we can select to pass through various resources on the local machine such as the clipboard drives and printers. Check everything and click on **Connect**.

Notice that as `RDSUser1` is signing on for the first time, there's a lot of initial configuration going on. So, login will be slower than usual before the Windows 8.1 desktop appears. Also notice that the remote desktop connection bar at the top of the screen thinks we are connected to `RDS-Broker.Contoso.com` and not a particular VM in our collection (`FPC-1` or `FPC-2`). If we go back and look at our collection in Server Manager (click on **refresh** in the menu bar first), we can see that `FPC-1` is running and that `RDSUser1` is signed in to it.

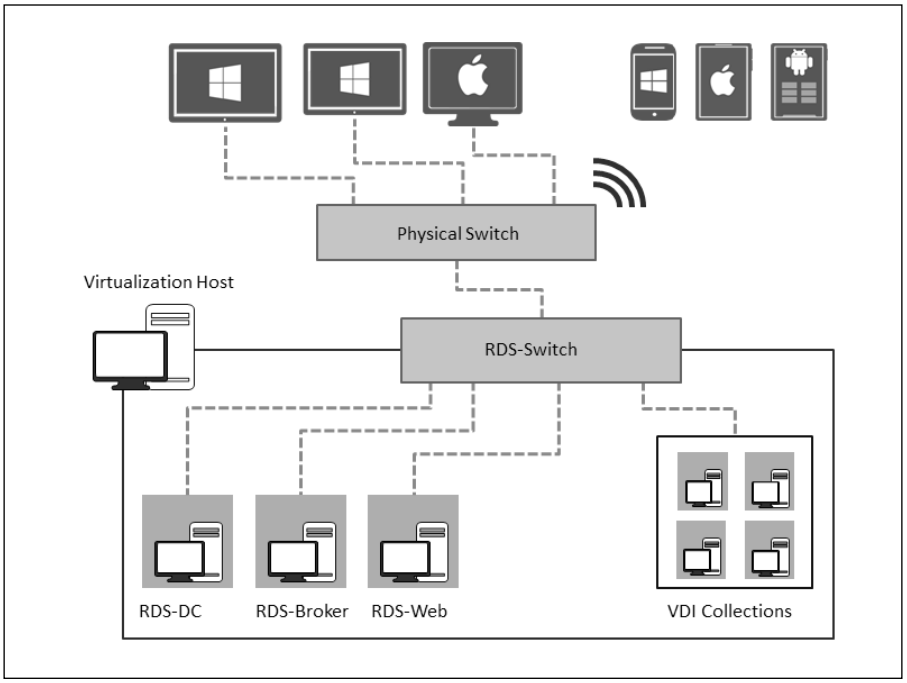
If we want to assess VDI from another device, then we can quickly do this by changing the RDS switch from being an internal switch to an external one in Hyper-V Manager on our host, or by running the following PowerShell command lines on the host:

```
$PhysicalNIC=Get-NetAdapter|where interfacedescription -like "*Broadcom*"
$RDSSwitch=Get-VMSwitch|Where Name -EQ "RDS-Switch"
Set-VMSwitch -NetAdapterInterfaceDescription $PhysicalNIC.
InterfaceDescription -VMSwitch $RDSSwitch -AllowManagementOS $True
```

This works by finding the physical NIC on the host (the physical NIC on my laptop is a Broadcom) and then altering the properties of our RDS switch to use that physical NIC, and still allow the Management OS on the physical host to use the NIC.

A physical NIC can only be bound to one virtual switch at a time, so this script might fail because there isn't an available NIC for the virtual switch to connect to.

Once we have done that, we could connect a physical network switch/router to that NIC to allow our VDI deployment to be accessed from other devices, as shown in the following diagram:



Now, any device with an RDP client on it can use the VDI collection we just created by simply connecting to that switch and going to our Web Access portal (`https://rds-web.contoso.com/RdWeb`) in a browser.

We need to use the right remote desktop client to get the best from our VDI deployment, which is RDP 8.1. This is included in Windows 8.1, and there are also clients available for Windows 7 and the other devices that are increasingly found in the workplace. The following links are sources for downloading the remote desktop clients for these operating systems:

- Android: `https://play.google.com/store/apps/details?id=com.microsoft.rdc.android`.
- iOS: `https://itunes.apple.com/us/app/microsoft-remote-desktop/id714464092`
- Mac: `https://itunes.apple.com/us/app/microsoft-remote-desktop/id715768417`

There is of course PowerShell support for all of this and, in fact, there is just one command to create a new collection—in this case a Personal Managed Collection. The command that can be used is as follows:

```
New-RDVirtualDesktopCollection `
  -ConnectionBroker RDS-Broker.contoso.com `
  -PersonalManaged `
  -CollectionName "Fast Personal Collection" `
  -VirtualDesktopTemplateName RDS-VDITemplate `
  -VirtualDesktopTemplateHostServer Orange.Contoso.com`
  -Domain Contoso.com `
  -OU RDS-VDI`
  -UserGroups VDI-Users`
  -VirtualDesktopAllocation 2 `
  -StorageTypeLocalStorage `
  -LocalStoragePath 'E:\Temp VM Store'`
  -VirtualDesktopNamePrefix FXC`
  -AutoAssignPersonalVirtualDesktopToUser`
  -CustomSysprepUnattendFilePath "\\Orange\tempVMstore\resources\ ??"
```

You can see that this command needs all of the same configuration settings that we put into the **Create Collection** wizard. However, this command has no mechanism for inserting unattend settings into the virtual desktop template like we did in the wizard (such as the time zone), and so the last parameter is a reference to an **unattend file**. This setting is optional and we'll use unattend (that is **answer files**) files inside the VDI templates in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*.

Creating an RD Session Collection

Given that Session Virtualization offers greater efficiency at the expense of user experience, we should compare how it works and what it can offer against VDI. We already have most of the infrastructure that we need for this; all that's missing is a session host; actually, we have one already as the script to create RDS-Broker and RSD-Web also included a third VM RDS-SHost, which we can use for this. Before we configure it and create a Session Collection (a collection of virtual desktops using Session Virtualization), we need to create a share for the UPDs just as we did for our Pooled Collection. However, these are different from the UPDs in a Pooled Collection, and we can't store them in the same location. So, we'll need to create another folder in our working directory (I suggest `RDS-UPD`) and share that out as `<hostname>\RDS-UPD`.

UPDs are only really appropriate for a group of users who use one type of virtual desktop (via VDI or Session Virtualization) and don't ever use a physical desktop. For those users who do use a variety of different desktop types, there are the usual mechanisms for handling user profiles such as folder redirection. However, with Server 2012 and the latest MDOP; there is also UE-V and we'll look at this in *Chapter 6*, *Scale and Performance*.

We can then use the RDS installation wizard (from our `RDS-DC` VM) to add in this session host to the rest of our RDS infrastructure, much as we did to set up the roles earlier. For this, we need to perform the following tasks:

1. From the **Server Manager** menu, navigate to **Manage | Add Roles and Features** to launch the wizard. Click on **Next**, and on the **Installation Type** screen, select the **Select Remote Desktop Services Installation** option and click on **Next**.
2. In the **Select deployment type** screen, notice that the RD Connection Broker is already set to **RDS-Broker.Contoso.com** and so, we are extending our RDS infrastructure and not building a new one this time. Leave the **Standard Deployment** option selected and click on **Next**.
3. In the **Select deployment scenario** screen, the **Virtual machine-based desktop deployment** option is grayed out as we have already created it, and so our only option is **Session-based desktop deployment**. Click on **Next**.
4. As before, we get the **Review role services** screen to review. Click on **Next**.
5. We can ignore the **Specify RD Connection Broker** screen as we already have it selected for us. Click on **Next**.
6. We can also ignore the **Specify RD Web Access server** screen as we already have one of those as well. Click on **Next**.
7. In the **Specify RD Session Host server** screen, highlight **RD-SHost.contoso.com** in the **Server Pool** column and click on the arrow to the right of this column to add it to the selected column. If it's not there, then check that this server is being managed in **Server Manager** and rerun the wizard. Click on **Next**.
8. In the **Confirm selections** screen, check the **Restart the destination server automatically if required** option and click on **Deploy**.

The PowerShell equivalent to this is just a one liner with two switches as follows:

```
New-RDSessionDeployment -SessionHost "RDS-SHost.contoso.com" -
ConnectionBroker "RDS-Broker.contoso.com"
```

Here, we just have to specify the right command and declare our RD Broker and the server to use as a session host, making sure we use fully qualified domain names (FQDN). Either way, the remote desktop overview diagram will now have subtly changed; there is now a solid line connecting the **RD Connection Broker** to the **RD Session Host**, and this icon will turn blue when we hover over it to enable us to add more session hosts as we scale up our deployment. Having added a session host, we are now in a position to create a Session Collection just as we created a Pooled Collection earlier. We will perform the following steps:

1. Right-click on the **RD Session Host** icon in the RDS **Deployment Overview** diagram and select **Create Session Collection**.
2. In the **Name the collection** screen, call it `Fast Session Collection` and optionally enter a suitable description. Click on **Next**.
3. In the **Specify RD Session Host server** screen, highlight **RD-SHost.contoso.com** in the **Server Pool** column and click on the arrow to the right of this column to add it to the selected column. Click on **Next**.
4. In the **User Groups** screen, remove **Contoso\Domain Users** and click on **Add**. In the **Select Users or Groups** pop-up box, type `session` and click on **Check Names**. **Session-Users** should appear. (You should have used the supplied script to create a DC; if not, select a suitable group or use the default of **Domain Users**.) Click on **OK**.
5. In the **User Profile Disks** screen, we will again make use of UPDs and enter the path to the share we created for this (for example, `\\<hostname>\RDS-UPD`).
6. In the **Confirmation** screen, review the options and click on **Create**.

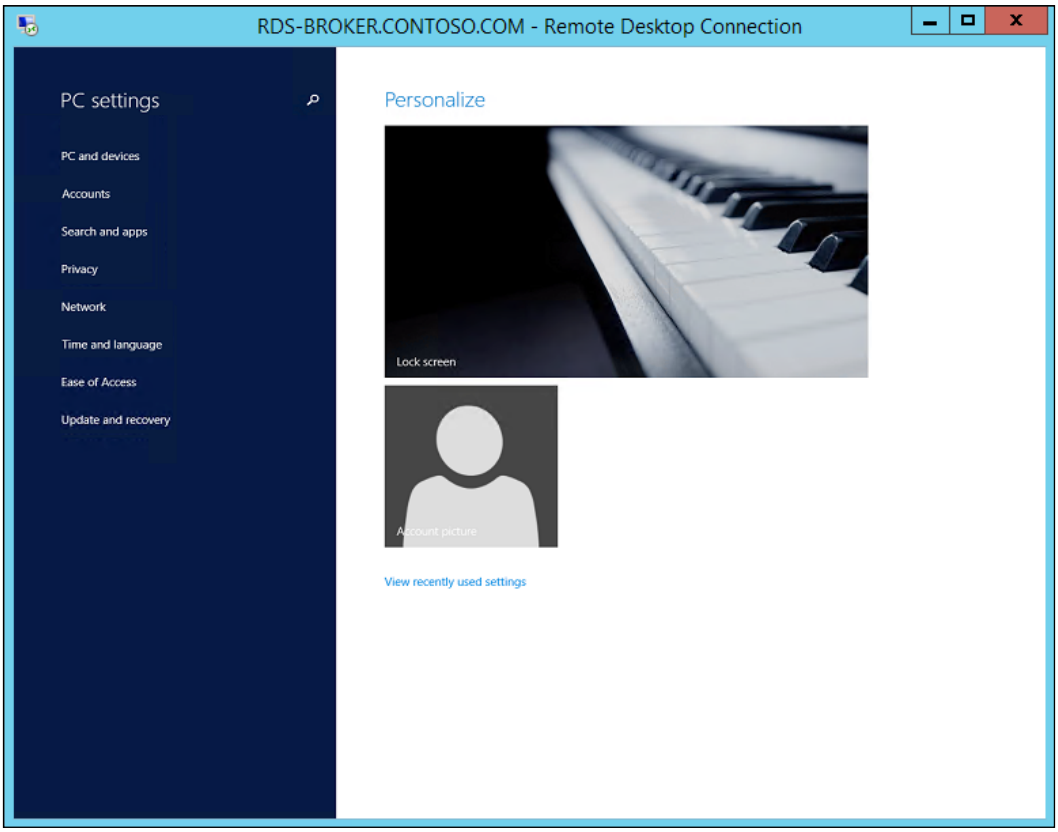
Our Session Collection will now show up in the navigation screen on the left and underneath the **RD Session Host** icon on the RD **Deployment Overview** diagram. Our users can access as before; if we go back to the Web Access portal now in the Internet Explorer and log in as `contoso/RDSUser3` (password = `Passw0rd!`), we can see our **Fast Session Collection** and login (you may well be asked to log in again). However, this brings up the traditional UI that we are used to as IT professionals, where our users would expect to see the modern UI. There are other capabilities our users would expect, such as video playback, and audio; all of which are encapsulated in the Desktop Experience server feature. However, it's important not to install this or any applications on the session host before it has the session host role service installed, as applications won't work unless installed in a special way. We'll see this in *Chapter 8*, *Managing User Profiles and Data* (when we look at application deployment). We can either do this by navigating to **Server Manager** | **Add Roles and Features** from our `RDS-DC` domain controller or with the following line of PowerShell command:

```
Add-WindowsFeature –Name Desktop-Experience -ComputerName RDS-SHost -
Restart
```

Here, the `-Restart` option is needed as this is required when this feature is added in. The audio service on Windows Server 2012 R2 isn't on by default either, so to rectify that we use the following commands:

```
Get-Service -Name AudioSrv | Start-Service
Get-Service -Name AudioSrv | Set-Service -StartupType Automatic
```

If we reconnect to our Fast Session Collection for the Web Access portal as `RDS-User3/Passw0rd!`, we can now see the modern desktop at login and notice that the store is now there, and so is the modern UI control panel on the modern UI start screen, which we can get to by clicking on the down arrow underneath the tiles to show all the applications currently available to this user, and then selecting the **PC settings** icon, as shown in the following screenshot:



Summary

Microsoft RDS includes two techniques for providing virtual desktops, Session Virtualization and VDI based on a collection of Windows 8 VMs. While Session Virtualization uses far less hardware resources, it is based on a server OS, which can be a jarring experience for our users and limit the applications we can offer using this technique. VDI consumes more resources, but offers our users a first-class experience. If we have good tools and practices in place to manage real desktops, then we can quickly deploy VDI without too much additional training. There's no right answer here, so it's about what is right for the department or business that will use VDI.

We have now got a basic VDI deployment setup, but the virtual desktops we have created are based on a clean copy of Windows 8.1, which is not what we'd want to provide in a production setting.

In the next chapter, we'll use some of the tools that we would use to deploy physical desktops and use these to create a properly configured desktop with applications, and the configuration settings needed to optimize Windows 8.1 for VDI.

Now that we have got a good understanding of how VDI works and what our options are, we can turn our attention to the design of the desktop itself. So far, all we have done is given our users a clean copy of Windows 8.1, but there are no applications in it nor is it at all configured as it would be in production. So in the next chapter, we'll look at how to design and configure our virtual desktop template in a similar way, and with similar tools to how we would build physical desktops.

3
Putting the D in VDI – Creating a Desktop Template

This chapter is all about the desktop, because VDI is only as good as the desktop, and a good desktop design will make life easy for our users, while keeping our management efforts to a minimum. Desktop design is a well-established discipline, and there are many tools available for it; however, there are special considerations for deploying virtual desktops, and if our VDI is to be a success, we must understand them. Many of these tools are free or are already included in both the Windows client and server OS. So, we will review these tools to see what they do and then use one of them to create a semiautomated process to create a more sophisticated Virtual Desktop Template and deploy a simple application as part of the process. We'll also take a look at what is present in the Group Policy that can help us in setting up and managing VDI.

Desktop deployment for VDI

The business of looking after all of the devices that are used for work by our users was hard enough when all we had to worry about were the various generations of desktops and laptops from different manufacturers. Today, our users are getting the majority of their work done on all sorts of devices, tablets, laptops, and smartphones, and these also have to be managed, even if they are owned by our users and not corporately funded. That is a topic for another book, but what is important here are two things:

- Our virtual desktops have to be deployed and managed with even more care than real desktops, because if something goes wrong, it won't just affect a single user whose laptop we broke; it will affect the whole group of users who will access the VDI we just made or modified.

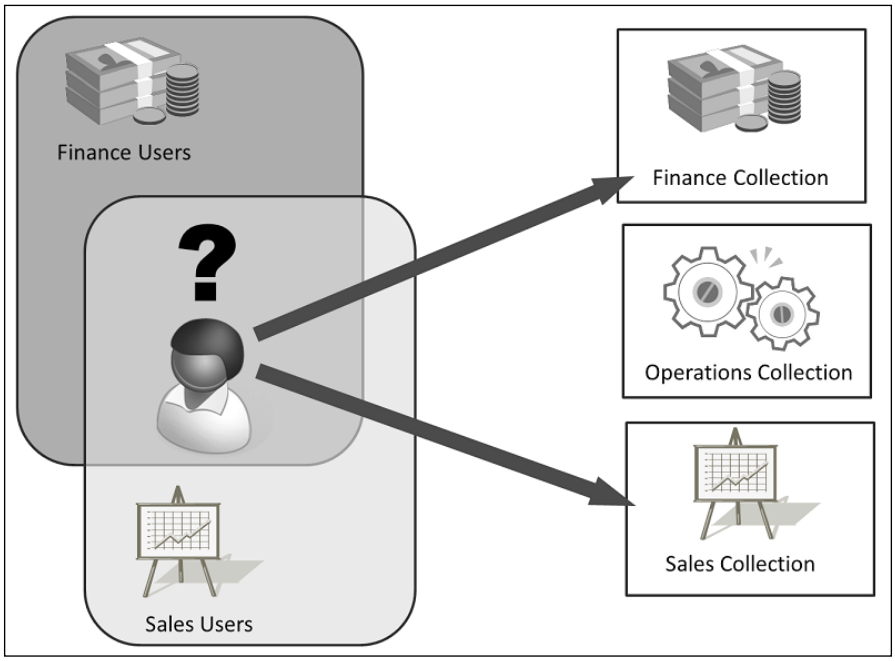
- As we saw in the previous chapter, VDI can be accessed from most of the new kinds of devices our users have. Furthermore, if we grant them access to corporate resources, we can ensure that no data (or application) will be running on those devices as it's all inside our data center and under our control.

The key difference with VDI is that we run our desktops as VMs in Hyper-V, and so the drivers we need are already in the OS and are the same for every deployment. Hyper-V has **RemoteFX** built into it to make VDI more efficient, while providing a better user experience. For example, in Windows Server 2012 R2, graphics-intensive applications can be supported in VDI either by the virtualization of a high-end graphics card in the physical host or by the emulation of a card similar to it with spare CPU cycles. RemoteFX also includes **USB redirection**, where USB peripherals on a local device, such as smart card readers for authentication and web cameras for unified communications, are available inside the virtual desktop.

The other thing that makes VDI easier to set up is that the actual deployment process is taken care of, as we saw in the previous chapter. All we had to do to make a collection was to provide a Virtual Desktop Template, which was then copied and cloned to create our virtual desktops. However, the way we created that template was very simple—all we did was create a new VM based on a VHD converted from Windows 8.1 Enterprise Evaluation media. For VDI, in production, we would want to customize this image in a number of ways, and when it comes to Session Virtualization, we also need to configure the Windows Server OS to match what we have on our Windows client desktops as far as possible.

Desktops are not much use without applications. While users might install applications on their real desktops, in VDI, we will only want to allow this for Personal Collections and set these to not roll back to an initial state when the user logs off, or they will lose the application they just installed. In that case, we are now giving our users the responsibility to manage their own VM, just as some organizations let users manage their own laptops. If we think about Pooled or Session Collections, the virtual desktop for each user is identical except for the users' settings, so each user accessing that collection will get the same applications. If a group of users needs a certain set of applications that won't be available to others, we can create a collection for that particular group based on a Virtual Desktop Template with those applications, plus any others that are used by the whole business. This is acceptable as far as it goes, but we live in a world of matrix management and dotted reporting lines. While we want each of our collections to be accessed by many users, we don't want to give user access to multiple collections.

The following screenshot provides a description about the collection to be used as the head of sales with access to finance:



The solution to this problem is **Application Virtualization**, where users get the applications they need as they log on to any desktop, whether it is real, session based, or via VDI. Microsoft's solution to this is **App-V** (we will take a look at this in detail in *Chapter 8*, *Managing User Profiles and Data*). However, it's not free per se—it's part of the MDOP that comes with **Software Assurance**. As a result, if you don't have this solution or have applications that won't work with App-V, you may still have to contend with the problem of matching users to applications and collections.

The other issue that affects desktop deployment, be it physical or virtual, is the settings and profiles of the users. We created **User Profile Disks** (**UPDs**) in the previous chapter to allow our users to keep these settings from session to session, but UPDs only work inside a collection. So, if a user has access to multiple collections, their settings will look different inside each of these and also won't be carried across to any physical desktop that they log in to.

We can see this by logging in to our RD Web Access portal as Contoso\RDSUser2 (with the password, `Passw0rd!`). Click on the fast-pooled collection, and change the lock screen picture to the one of piano keys and log out of it. Connect back into the fast-pooled collection again, and note that the lock screen is still the same picture. Now, click on the session collection, and you will see that this isn't the same. Any changes you make here will only work if you go back into that same collection.

So, we might need to use some of the traditional approaches to manage user profiles, enable folder redirection, and so on. We will cover all this in *Chapter 7*, *Maintenance and Monitoring*.

Microsoft deployment tools

There are all sorts of tools that can deploy a Windows OS to a desktop, but a lot of the older tools rely on capturing an image in binary form, which can't be tuned or modified. All three of Microsoft's deployment tools are file based and allow us to inject applications, patches, and user settings into deployments so that we can create and keep an updated library of resources that we can deploy Windows from at scale. These tools are as follows:

- **Disk Image & Service Management** (**DISM**): This is a command-line tool included in the OS, which replaced ImageX in Windows 7 / Windows 2008. It can work on a running OS (in `/Online` mode) or on WIM files and VHDs. It has a lot of switches and is quite tricky to get working. DISM is also the only way to sideload modern apps designed for Windows 8.1.
- **The Windows Assessment and Deployment Toolkit** (**ADK**): This is a free tool that replaced the **Windows Automated Installation Kit** (**WAIK**) when Windows 8 was released. The deployment tool uses DISM and allows you to edit answer files in a managed way through its interface. However, deployment using the ADK is quite a manual process. There are other tools in ADK for testing, managing licenses, and a **User State Migration Tool** (**USMT**) to capture existing users' settings and put them into special files (MIG) that can be replayed back to an image.

- **The Microsoft Deployment Toolkit** (**MDT**): This is a free solution accelerator that makes the deployment of images with the ADK a lot easier, and is widely used for this reason. It allows us to set up a desktop in a known state and use this as a template to roll out an image to all of our PC estate. It is very similar to the way VDI uses a virtual desktop in the final chapter. What MDT does that RDS doesn't is that it gives us a workbench to make this initial template, including providing options to add applications and capture users' settings.
- **System Center Configuration Manager 2012 R2** (**CM12R2**): This is a paid, comprehensive desktop management solution that is not only designed to handle very large-scale Windows deployments, but also update and monitor these deployments to ensure that they are in compliance with the desired baselines that we configure. CM12R2 doesn't just manage the Windows OS; it also handles application management and includes System Center Endpoint Protection, which is Microsoft's enterprise-grade, anti-malware solution. However, it doesn't have any specific tools to deploy VDI. It actually uses MDT to perform deployment under the covers, and so works in much the same way.

The question is, *which tool should we use for VDI deployment?* If we have access to CM12R2, we should use it, but if not, then MDT is the best bet as it is free and makes using the ADK easier. So, let's take a look at how MDT works so that we can adapt it for VDI.

Installing MDT

We really need MDT to run on its own VM, which I will refer to it as RDS-Ops. MDT can run on either Windows 8.1 or Windows Server 2012 R2, but I am going to suggest using the Server OS and creating a VM with extra VHD for the resources used by MDT.

I have included a script to create RDS-Ops that does this and enables deduplication in Windows Server to save more than 80 percent space on this second drive.

If hardware resources are limited on your host, you may want to shut down all of the running VMs, except the RDS-DC, manually or with the following PowerShell command:

```
Get-VM |where status -NE "Running" | where name -NE
RDS-DC | stop-VM
```

Once RDS-Ops is in place, install MDT 2013 by performing the following steps:

1. Install the parts of the ADK 8.1 (`http://www.microsoft.com/en-us/download/details.aspx?id=39982`) that are the prerequisites for MDT: the deployment tools, **Windows preinstallation environment** (**winPE**), and **User State Migration Tools** (**USMT**). The initial ADK download is tiny because it pulls the rest of the installation files as part of the setup. There is a way to pull this down and deploy it to other computers; run `adksetup /layout <Path>` and then run `adksetup` again, as follows, to install just the three components that MDT needs:

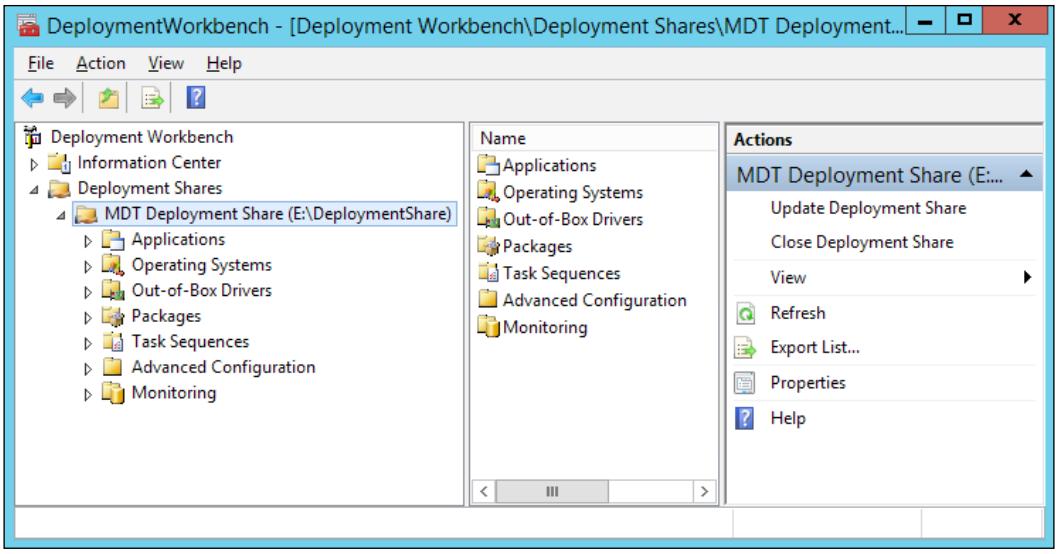
   ```
   adksetup.exe /quiet /installpath <the path specified in the
   layout option> /features OptionId.DeploymentTools OptionId.
   WindowsPreinstallationEnvironment OptionId.UserStateMigrationTool'
   ```

2. Now, download and install MDT 2013 (`http://www.microsoft.com/en-us/download/details.aspx?id=25175`). MDT will then be able to be installed from the command line with `MicrosoftDeploymentToolkit2013_x64.msi /Quiet`.

The MDT download includes a series of quick start guides in a zipped documentation folder, which is good background reading; we'll refer to these later. Before we start using MDT, we need to create a special share where MDT will store all the files we use. Do this on the new VHD that was created (in my case, `E:`) with the following steps:

1. In RDS-Ops, open **Deployment Workbench** (using the down arrow from the start screen or by searching for it).
2. Right-click on the **Deployment Shares** icon in the navigation pane and select **New Deployment Share**.
3. On the **Path** screen, leave this as is (`C:\DeploymentShare`), and click on **Next**.
4. Leave **Share name** as is, and click on **Next**.
5. Give it a descriptive name or leave it as is, and click on **Next**.
6. On the options screen, uncheck everything except **Ask if an image should be captured**, and click on Next.
7. Click on **Next** on the **Summary** screen to create the share. When it's finished, click on **Finish** to close the wizard.

We now have a deployment share that has a folder structure within which we can store all the resources we need to deploy images, as shown in the following screenshot:

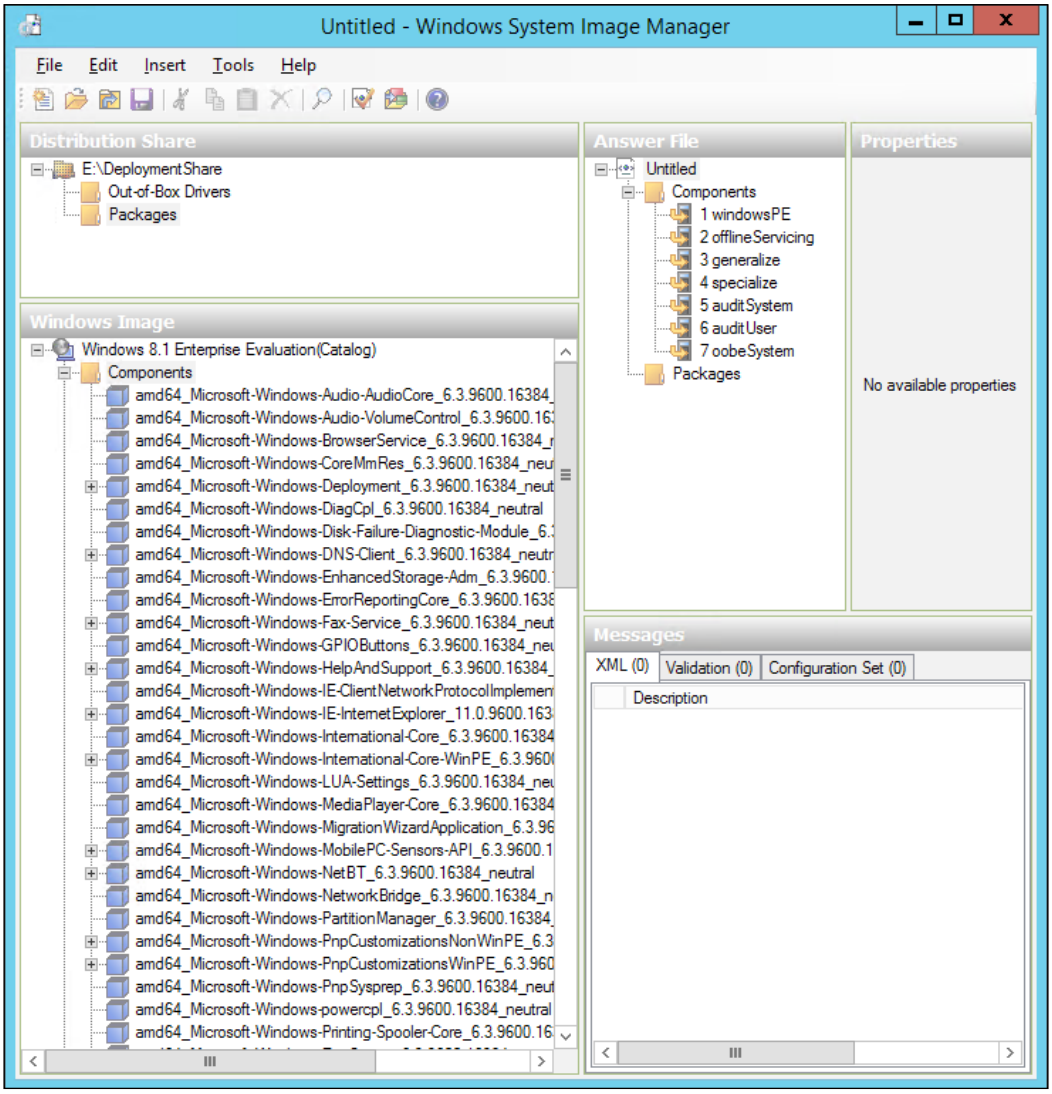


Working with answer files

The tools that were downloaded include sample answer files that we can adapt, but before we can use them, we need to import an operating system to work with as follows:

1. Mount the Windows Enterprise Evaluation ISO into the VM you are using for MDT (RDS-Ops). To do this, go to the menu at the top of the **VM console**, then go to **Media** | **DVD drive** | **Insert disk**, and browse to the ISO.
2. On the RDS-Ops, go back into the **Deployment Workbench**, expand **Deployment Share** that was just created, and select **Operating Systems**. Right-click and select **Import Operating System**.
3. In the **OS Type** screen, select **Full set of Source Files** and click on **Next**.
4. On the source screen, browse to the DVD drive in the VM (probably `D:`) and you should see the ISO file mounted. Select this and click on **Next**.
5. On the destination screen, the destination directory will already be populated (which you can change). Click on **Next**.
6. Review the choices in the summary screen and click on **Next** to import the OS. When it's done, you'll see this OS under the list of operating systems.

We can now see the operating system in the Workbench. We can't get to the corresponding answer file in MDT directly; we have to use **Windows System Image Manager**, which will be present in all the applications below the start screen on the modern desktop in RDS-Ops. It should already have our operating system, as MDT has configured it to point to the operating systems in our deployment share. The following screenshot shows the **Windows System Image Manager** window:



The **Answer File** section in the mid section has nothing in it, just a series of placeholders for different parts of the OS deployment process. We can add components to it from the components in our Windows image on the left-hand side by right-clicking on one. If we do that for any one of these, we will see that each component in the image only applies to certain components in the answer file—in other words, certain configuration items are only set at specific points in the deployment process. There's a lot of complexity here, which would need a book in its own right, but what we could do to get started is to open a sample answer file and look at how it works. Perform the following steps:

1. In the **Windows System Image Manager**, go to **File** | **Open** and navigate to `C:\Program Files (x86)\Windows Kits\8.1\Assessment and Deployment Kit\Deployment Tools\Samples\Unattend`. Select the `Autounattend_x64_BIOS_sample.xml` file and click on **Open**.
2. Click on **Yes** to accept the message about associating the answer file to the Windows image.

The components **1 Windows PE** and **4 Specialize** now have the settings, but still don't really contain anything useful. A simple, practical example of the sort of thing that can be done is to adapt this answer file to join a domain. Perform the following steps:

1. In the **Windows Image** pane, search for **amd64_Microsoft-Windows-UnattendedJoin_6.3.9600.16384_neutral** and right-click on it.
2. Select **Add Setting to Pass 4 specialize**.
3. In the **Answer File** pane, expand **4 Specialize** and navigate to **amd64_Microsoft-Windows-UnattendedJoin_neutral**, and right-click to get help about this setting; note that this will only work for Windows 8.1 Enterprise. Then, click on the identification hyperlink in the help section to find out how to configure this setting, and note that we can grab the XML and also hack another answer file with it if we want to.

We can add in a vast array of options here: we can specify networks, set up devices and the user experience, and add user accounts with predefined scripts that will be executed when the user logs in. We can also add and remove Windows features and configure licensing and activation. Having made the changes here, all we need to do for VDI is to grab this file and use it as we create a collection. For example, we could now create a Personal Collection based on this file using the script at the end of the previous chapter. However, this is a very detailed way of affecting changes and is limited to just the OS itself. If we want a richer solution, we need to start looking at how MDT itself is built on top of the ADK.

Building a new Virtual Desktop Template with MDT

There is an easier way to create our template, and that is to adapt the **Lite Touch Installation** (**LTI**) process; we can use it in the following way:

1. We import an image of an operating system from the installation media, which we already did in the previous section.
2. We create a process (MDT task sequence) to describe how to deploy this OS to a reference computer.
3. We create a boot image from the OS and use the task sequence to install the OS to the reference computer.
4. We can now use this image to do the installation by just mounting the ISO image that MDT produced and boot the reference computer VM from that.
5. We can then run the **Deployment Wizard** from our reference computer and, rather than capturing it for use in LTI to target computers, as described in the MDT documentation, we can just sysprep the reference computer for it to become our Virtual Desktop Template.

For a more detailed description, refer to **Quick Start Guide for Lite Touch Installation.docx**, which is included in the MDT documentation, and the complete installation is actually much longer than this book!

So, let's try building a template with MDT in our lab.

Creating a task sequence to deploy the captured OS to the reference computer

Now, you need to create a task sequence to deploy the captured image of Windows 8.1 to your reference computer:

1. In **Deployment Workbench** on RDS-Ops, navigate to **Task Sequence Node**. Right-click and select **New Task Sequence**.
2. On the **General Settings** screen, give the sequence a unique name, for example, `Win81Ref`, and a name like `Windows 8.1 Reference Deployment (x64)`. Click on **Next**.
3. On the **Select Template** screen, choose what you want to do in the workbench, such as a server deployment, and deploy to VHD. Select **Standard Client Task Sequence** and click on **Next**.

4. You might think that we would want to deploy to VHD, but this task sequence is designed around the Boot to VHD feature in Windows 7/2008R2, and later, where the OS on a physical host can reside on a VHD, it is mounted as part of the boot process. This option is *not* present to deploy an OS to a virtual machine.
5. On the **Select OS** screen, select the OS we have already imported and click on **Next**.
6. In **Specify Product Key**, click on **Next** to accept the default (**Do not specify a product key at this time**).
7. On the **OS Settings** screen, enter some meaningful details for the name, organization, and home page, and click on **Next**.
8. On the **Admin password** screen, use your standard lab password (in my case, `Passw0rd!`) and click on **Next**.
9. Review the summary information you entered and click on **Next**.

The new task sequence is now stored in our deployment share. Before we can use it, we should enable monitoring to troubleshoot any issues by performing the following steps:

1. Navigate to the deployment share you created and right-click on it to open its properties.
2. Select the tab in the rightmost corner, that is, **Monitoring**. Check the **Enable monitoring for this share** option and leave the ports as they are. Click on **Apply**.
3. Click on the **Rules** tab and confirm whether the monitoring is enabled by noting that the entry, **EventService=http://RDS-MDT:9800**, is now present. Click on **OK** to close the properties.

Updating the deployment share

We can now update the deployment share to publish the work we have done. Right-click on our deployment share and select **Update Deployment Share**. Leave the default options as they are and click through the wizard.

Before closing the wizard, it's useful to see what actually happened. Two boot images have been created, one for x86 and the other for x64 in both the WIM and ISO formats. If we open the file explorer and navigate to the deployment share on the disk, there is now a folder in there called `boot`, and that's where these images are. Note that there is a **WIM** (**Windows Imaging Format**) file for each and a corresponding ISO file.

Creating the reference computer

The reference computer is going to be used to make our Desktop Template, so whatever we do here will be inherited by the template and any VDI collections created from that template. Our reference computer is a VM, and we can create it using a couple of lines of PowerShell:

```
$VMName = "RDS-Ref"
$VMSwitch = "RDS-Switch"
$WorkingDir = "E:\Temp VM Store\"
$VMPath = $WorkingDir + $VMName
$VHDXPath = $WorkingDir + $VMName +"\" + $VMName +".VHDX"

# Housekeeping - delete the VM from Hyper-V
$vmlist = get-vm | where vmname -in $vmname
$vmlist | where state -eq "saved" | Remove-VM -Verbose -Force
$vmlist | where state -eq "off" | Remove-VM -Verbose -Force
$vmlist | where state -eq "running" | stop-vm -verbose -force
-Passthru | Remove-VM -verbose -force
# Housekeeping - get back the storage
If (Test-Path $VMPath) {Remove-Item $VMPath -Recurse}

# Create the VHD from the Installation iso
md $VMPath
New-VHD -Path $VHDXPath -Dynamic -SizeBytes 30Gb

#Create the VM itself
New-VM –Name $VMName   –VHDPath $VHDXPath -SwitchName $VMSwitch -Path
$VMPath -Generation 1

# change these setting to suit your lab setup
Set-VM -Name $VMName –MemoryStartupBytes    1024Mb
Set-VM -Name $VMName -DynamicMemory
Set-VM -Name $VMName -MemoryMinimumBytes    512Mb
Set-VM -Name $VMName -AutomaticStartAction StartIfRunning
Set-Vm -Name $VMName -AutomaticStopAction   ShutDown
Set-Vm -Name $VMName -ProcessorCount          2

#Mount the iso from the deployment share
Set-VMDvdDrive -VMName $VMName -Path '\\rds-ops\DeploymentShare$\Boot\
LiteTouchPE_x64.iso'

Start-VM -Name $VMname
```

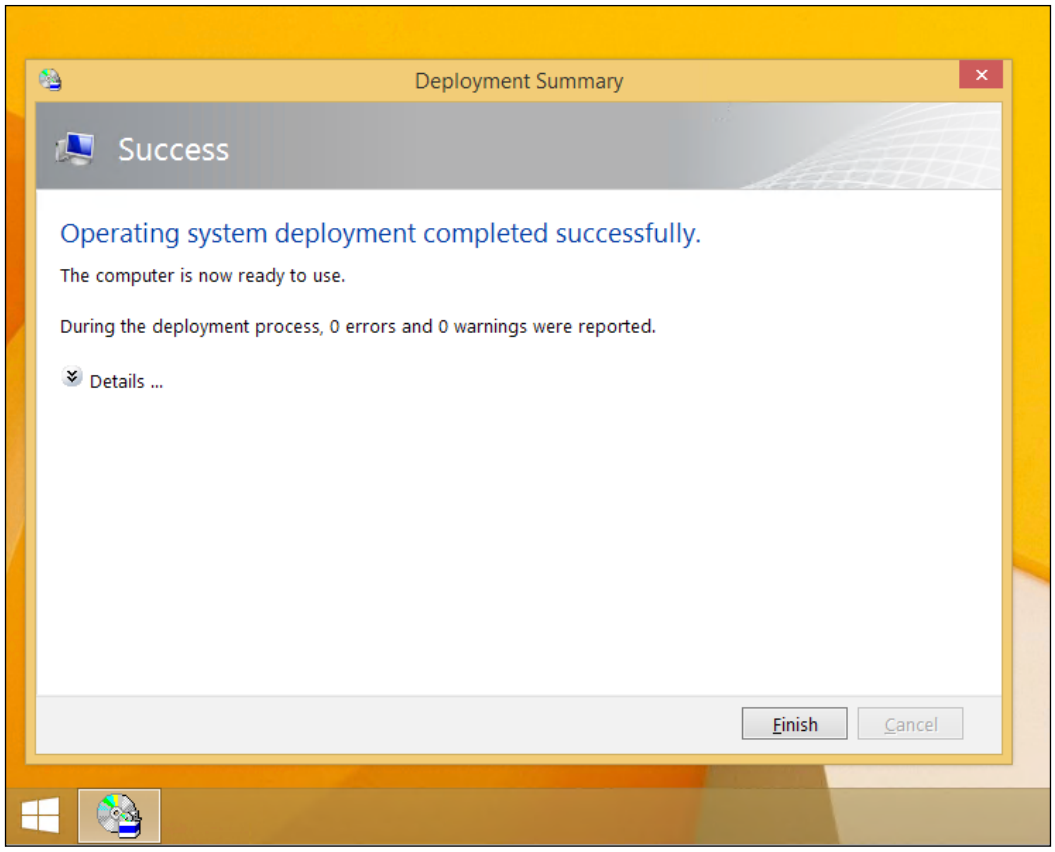
Here, we are simply creating a VM with the settings we want, as before. However, this time, we are creating a blank, dynamic VHDX and mounting the ISO file we made previously into the DVD drive so that when it starts, we will have the OS we just made by updating our deployment share. There are also some housekeeping commands at the start of this script to remove the VM and its associated storage if it's already there, so we can run this script again and again as we test our settings in MDT.

Running the deployment wizard

When we connect to our VM after it's come out of sysprep, we will be presented with an MDT flash screen that we can run the deployment wizard from to install a new operating system. Before we select that, it would be good to set the right language. However, we won't need to worry about static IP as we connect to the VM from the console. The steps to run the deployment wizard are as follows:

1. First of all, we need to access our deployment share. So, on the **Credentials** screen, we enter the values as follows:
    - **User Name**: `administrator`
    - **Password**: `Passw0rd!`
    - **Domain**: `contoso.com`
2. In the **Task Sequence** screen, we see the task sequence we created earlier to create our reference computer—**Windows 8.1 Reference deployment (x64)**; we select it and click on **Next**.
3. In the **Computer Details** screen, we fill out the details of the domain and give the VM a name. As we have seen, the VDI wizard gives us the option to join a domain, and that option is also part of the PowerShell command to create a collection, so all we need to do here is give the machine a name, such as `RDS-Ref`, and click on **Next**.
4. On the **Move Data** and **Settings** screens, we click on **Next** as there is no data to move—it's a new VM!
5. On the **User Data** (restore) screen, we click on **Next** as we aren't using USMT.
6. On the **Locale and Time** screen, we put in the details of our region and click on **Next**.
7. Now, we depart from the LTI process in the MDT documentation. All we need to do at this stage is to sysprep this VM, and it's ready to be our Virtual Desktop Template. We select the **Sysprep this Computer** option and click on **Next**.
8. On the **Ready to begin** screen, we review what we have set and click on **Begin** to start the installation.

We will now see the various stages of the Lite Touch installation execution, as the OS is installed and when it has got to the point where we can log in (with `administrator\Passw0rd!`), we can see the sysprep process being executed. The **Deployment Summary** screen is as follows:



We can then eject the DVD from the VM from its properties in Hyper-V Manager on our host or we can run `Set-VMDvdDrive –VMName RDS-OPs –Path $Null` from the host instead, to do the same thing.

This VM is now ready to either form the basis of a new collection or update an existing collection. And, we can confirm this by following the steps to create a pooled collection in the previous chapter, but this time we would select **RDS-Ref** as the VM to be the Virtual Desktop Template. What is more important is that we now have a basic but repeatable process to create our Virtual Desktop Template, but there are two areas we need to address to make it more useful:

- We don't really have to connect to the reference computer to complete the deployment wizard as we know all the answers. In other words, we really want ZTI and not LTI.
- We haven't really done any customization in MDT yet, such as adding applications and removing features.

Automating MDT

There are two files we need to work with to automatically enter all of the information we supplied to run the deployment wizard, both of which are in the properties of our MDT deployment share. On the **Rules** tab for the share, we observed some settings when we set up the monitoring. These settings are applied once the deployment wizard is running, but before that, another INI file is used—`Bootstrap.ini`—which can be customized from this screen as well. Perform the following steps to automate MDT:

1. Connect to **RDS-Ops**, and in **Deployment Workbench**, right-click on **MDT Deployment Share** and select **Properties**.
2. From the **Rules** tab, select **Edit Bootstrap.ini** and modify it as follows:

```
[Settings]
Priority=Default

[Default]
DeployRoot=\\RDS-OPS\DeploymentShare$
UserID=Administrator
UserDomain=CONTOSO
UserPassword=Passw0rd!
KeyboardLocale=en-US
SkipBDDWelcome=YES.
```

3. Navigate to **File | Save** and then close the file, but stay in the **Rules** tab.

What we have done here is supply the credentials we were first asked for in the deployment wizard to access the deployment share. Since we are doing this automatically, we can also suppress the welcome screen with `SkipBDDWelcome=YES`. However, one of the things we initially did when we ran this manually was to select our task sequence and also specify which options we want inside the task sequence, such as the name of the machine, locale and time zone. We can enter all of this directly in the **Rules** tab.

Edit the **Rules** tab to contain the following settings—the order doesn't really matter and the line gaps are just there for clarity:

```
[Settings]
Priority=Default
Properties=MyCustomProperty

[Default]
DeploymentType=NEWCOMPUTER
OSInstall=YES
SkipAdminPassword=YES
SkipProductKey=YES
SkipComputerBackup=YES
SkipBitLocker=YES
EventService=http://RDS-Ops:9800
SkipBDDWelcome=YES

SkipTaskSequence=YES
TaskSequenceID=Win81Ref

SkipCapture=YES
DoCapture=SYSPREP
FinishAction=SHUTDOWN

SkipComputerName=YES
SkipDomainMembership=YES

SkipLocaleSelection=YES
KeyboardLocale=en-US
UserLocale=en-US
UILanguage=en-US

SkipPackageDisplay=YES
SkipSummary=YES
SkipFinalSummary=YES

SkipTimeZone=YES
TimeZoneName=Central Standard Time

SkipUserData=Yes
```

Technically, these rules are the `CustomSettings.ini` file and most of these settings are documented on TechNet (`http://technet.microsoft.com/en-us/library/bb490304.aspx#E0CB0AA`); however, this is a bit out of date. For example, in MDT2013, it's possible to sysprep and shut down the OS, but these settings are not documented here. So, you will have to rely on community posts and other related data to get more information, for example, `http://blogs.technet.com/b/chuck_kiessling/archive/2013/02/05/mdt-2012-quot-customsettings-ini-quot-switches.aspx`.

What we are doing here is skipping various screens, for example, `SkipCapture=YES`, and then supplying the answers these screens would have asked us; in this case, `DoCapture=SYSPREP` and `FinishAction=SHUTDOWN`. These screens will sysprep and shut down **RDS-Ref** at the end of the deployment wizard. When you have made these changes, remember to update the MDT deployment share with the following steps:

1. Right-click on **MDT Deployment Share** and select **Update Deployment Share**. Accept the defaults and click on **Finish** to close the wizard when it has completed.
2. We can now test if it works by rerunning PowerShell to create the reference computer (**RDS-Ref**) mentioned previously.

We can get a good idea of what our new process is doing by keeping an eye on the console of **RDS-Ref** as it goes through the installation process. We can see where we are in the task sequence from the **Monitoring** section in **Deployment Workbench**.

Select **Monitoring** in the **Deployment Workbench** navigation pane. Select the log that's running in the mid section and double-click on it to open it. This will show you any errors and where the deployment has got to. If there's nothing here, check the MDT deployment share property rules to ensure that the **Monitoring** option is still enabled (something like `EventService=http://RDS-Ops:9800`).

If the process doesn't work the first time, don't worry. You are learning, and all you need to do to retest is turn off the reference computer (**RDS-Ref**) in Hyper-V manager, make any changes you want in the deployment workbench, and rerun the script to create the reference computer again.

Doing all of this to create our VDI template might seem like a lot of extra work right now, but we have created a framework that we can build on, and so we now have the ability to do the following:

- We can quickly create variations on what we have done to create templates for each of our different collections

- We can keep our environment up to date using MDT to add in updates, as we will see in *Chapter 5*, *High Availability*
- We can extend what we have done to include the applications we want in each of our collections, which is what we will look at in the next section

Deploying applications with MDT

MDT can easily manage our application deployment needs from a simple MSI/EXE file to applications with dependencies. For a quick demo, we can use a PDF reader such as Foxit.

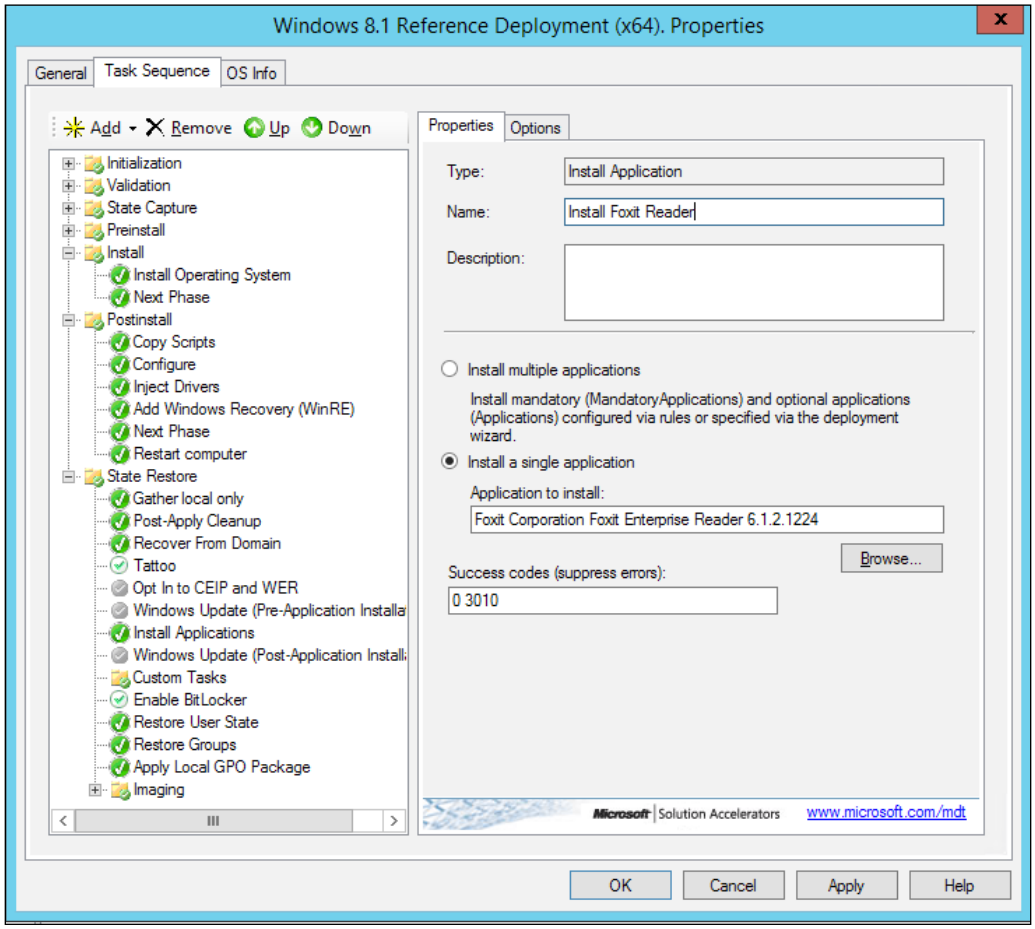
You will have to register with Foxit to get an MSI file of their enterprise reader to do this, but it is free. If you don't want to do that, you can follow along with an MSI file of your own. When you have the MSI file, copy it onto RDS-Ops in its own folder called `Foxit`. Perform the following steps to deploy applications with MDT:

1. First, import the application. Connect to RDS-Ops and open **Deployment Workbench**. Expand **MDT Deployment Share** and right-click on **Applications**, and select **New Application** to launch **New Application Wizard**.
2. In the **Application Source type** screen, note that you can get the application from a share and deploy bundles of applications. Select the **Application with source files** option and click on **Next**.
3. On the details screen, enter **Publisher** as `Foxit`, enter **Application name** as `Foxit Enterprise Reader`, enter the value for **Version** if you know it, and optionally, enter the value for **Language** if you want to. Click on **Next**.
4. On the source screen, locate the folder that contains the MSI file and click on **Next**.
5. On the destination directory screen, enter `Foxit Enterprise Reader` and click on **Next**.
6. On the command details screen, set the installation command to the following:

   ```
   msiexec /i EnterpriseFoxitReader612.1224_enu.msi /quiet
   ```

   Set the working directory to `\\RDS-OPS\DeploymentShare$\Applications\Foxit Enterprise Reader`, in other words, to the physical path of the installation file.
7. Check the settings on the summary screen and click on **Next** to import the application. Review the output and click on **Finish** to close the wizard.

8. Expand the **Foxit** directory under **Applications** in the navigation pane on the left-hand side of the screen, and the new application will show up in the center of the screen. Right-click on it and select **Properties**. On the **General** tab, you will see that the application has a GUID, which we should know, so copy this to the clipboard. On the **Details** tab, set what OS it can be deployed on, and on the **Dependencies** tab, set any other applications or fields that this depends on.
9. All you have to do now is customize the task sequence to install this application. To do this, expand **Task Sequences** in the deployment share and right-click on the task sequence you already created to set its properties. By default, the type of task sequence you selected earlier already has an application install step included. To find it, expand the **State Restore** folder on the task sequence list and then edit the application settings as shown in the following screenshot:



10. Set the **Name** field to `Install Foxit Reader` and click on the **Browse...** button to locate the Foxit Reader application you just imported. Click on **OK** to apply this.
11. To avoid being given the option to install Foxit in the deployment wizard, re-edit the rules (control `Settings.ini`) of the MDT deployment share properties with the GUID of the application in the default section (under `[default]`) as shown in the following code:

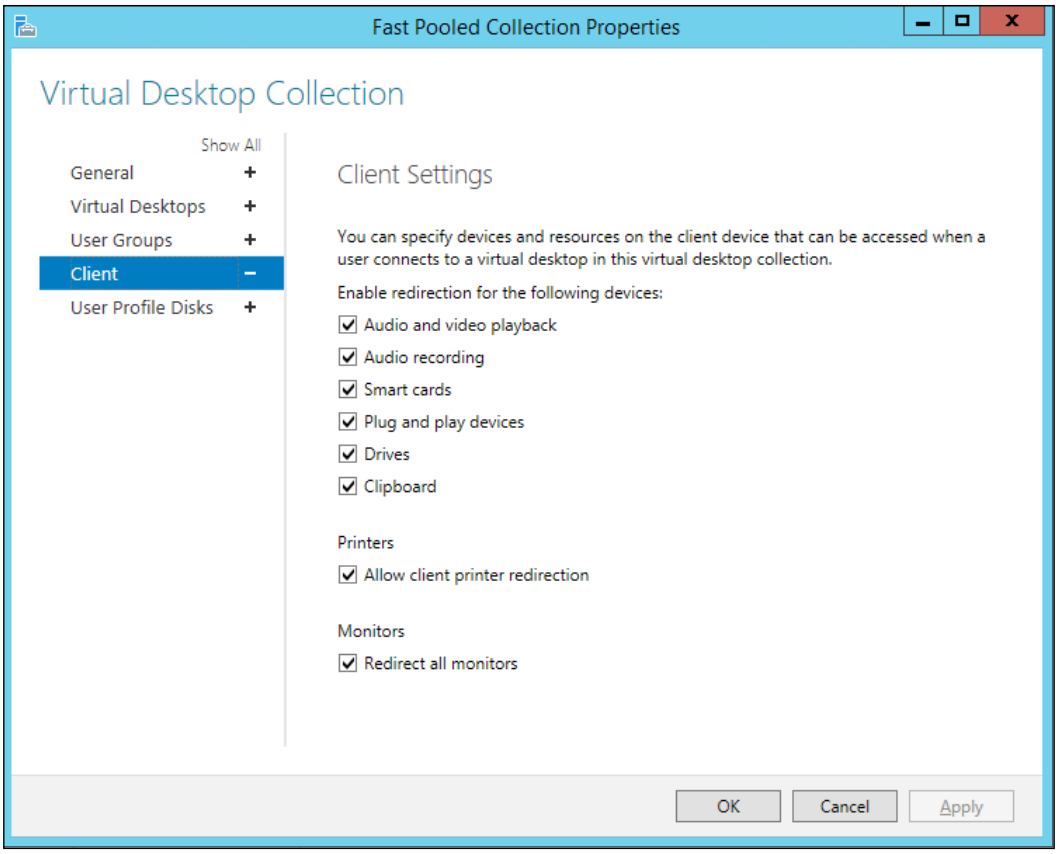
    ```
    SkipApplications=Yes
    Applications001 ={ec8fcd8e-ec1e-45d8-a3d5-613be5770b14}
    ```

    Here, the **SkipApplications** setting will hide the confirmation, and the GUID for the application can be obtained from the **General** tab of the properties of the application.

12. Now, rerun the script to create the reference VM (remember to disconnect the ISO file and update the deployment share before running the script again). Obviously, some applications such as Office will require more effort than this, but the principle is the same:
    - Test whether the command-line installation works without using MDT
    - Import the application into MDT
    - Modify the task sequence to install the application (there can be multiple install application steps in a task sequence)
    - Edit the deployment share rules
    - Update the deployment share
    - Rerun the deployment

Configuring collection properties

When we created our collections in the previous chapter, we went with the defaults for everything. However, there are a number of options we can set that will determine the desktop experience our users get. If we go back to our RDS-DC domain controller and go into the **Remote Desktop Services** node of **Server Manager** and then select **Fast Pooled Collection**, we can see its properties by selecting **Edit Properties** from the task option of the **Properties** pane. The following screenshot displays the **Fast Pooled Collection Properties** window:



There are two groups of settings we are interested in here: the **Client Settings** in the previous screenshot and use of **User Profile Disks**. The client settings allow us to quickly restrict what our users can do with the local devices they're using to connect to our virtual desktop. This is where we can quickly lock down the clipboard and plug-and-play devices and drives, which would otherwise enable our users to take data away from the virtual desktop, if we want to use RDS in a secure environment. However, we might only want to restrict certain users or home users from doing this. In this case, we can do the same thing we did in Group Policy, with filters on the users that it will affect, as we will see shortly.

Setting restrictions here does not show up in Group Policy anywhere. So, be aware that the principle of least privilege applies if you make settings here and in Group Policy.

The client settings for Session Collections are nearly the same, except that we have a couple of extra options for printer redirection. We can also limit the number of monitors a user can use. Since Session Collections handle many users from one host, we also get the option to load balance sessions across several hosts. In VDI, we saw that we can decide how many virtual desktops we want to deploy to each virtualization host and load balance it that way.

We had a quick look at **User Profile Disks** (**UPDs**) in the previous chapter. In the details setting for the UPDs, we can refine them to exclude certain folders, such as the standard folders (pictures, documents, videos, and so on), and also include any folders of our own that might be needed by certain applications.

Group Policy and the virtual desktop

Group Policy can be applied in a more fine-grained way to control the desktop experience our users get when accessing virtual desktops. There are also some other specific settings we need to be aware of, specifically the special settings used for Session Virtualization and to restrict what applications our users can use. This is not strictly something we need to implement in the VDI; however, it's not a well-known feature of Group Policy either. It can be used to reduce the number of different collections we maintain by blocking certain virtual desktop users from running an application even though it's installed in the collection.

Group Policy with Session Virtualization

There's a whole collection of settings to configure Session Virtualization, which can be found under `Computer Configuration\Policies\Administrative Templates\Windows Components\Remote Desktop Services\Remote Desktop Session Host`. The following options give us more control than the collection properties we saw earlier:

- **Application compatibility**: There's a special compatibility mode that is needed by some applications to run inside an RD session, which can be set here. Also, the session host has just one IP address, so we might need to assign virtual IP addresses for our session users so that they have different connections to a backend service that is discriminating on IP addresses.
- Connections and their limits can be set here, including the ability to shadow a user's session, which is available in Windows Server 20012R2 again.

- **Device and resource redirection**: This is the fine-grained approach to restricting users from using various local devices. Here, we can restrict which users this applies to, whereas setting this at the collection level means it is universal for all users.
- **Licensing**: This connects the environment to a licensing server, which we will look at in the chapter on licensing.
- Printer redirects to set the default printer.
- **Profiles**: This controls user profiles.
- **RD Connection Broker**: These settings are for scaling out a farm of brokers. We'll configure those in the next chapter from a special wizard for this.
- A remote session environment contains detailed settings, such as the screen resolution.
- Security can be restricted here by limiting the kinds of clients that are allowed to connect.
- Session time limits allow us to log out of an inactive session after a given time.
- **Temporary folders**: This lets us set whether to assign an individual temporary folder for each user and the option to delete these when they log out.

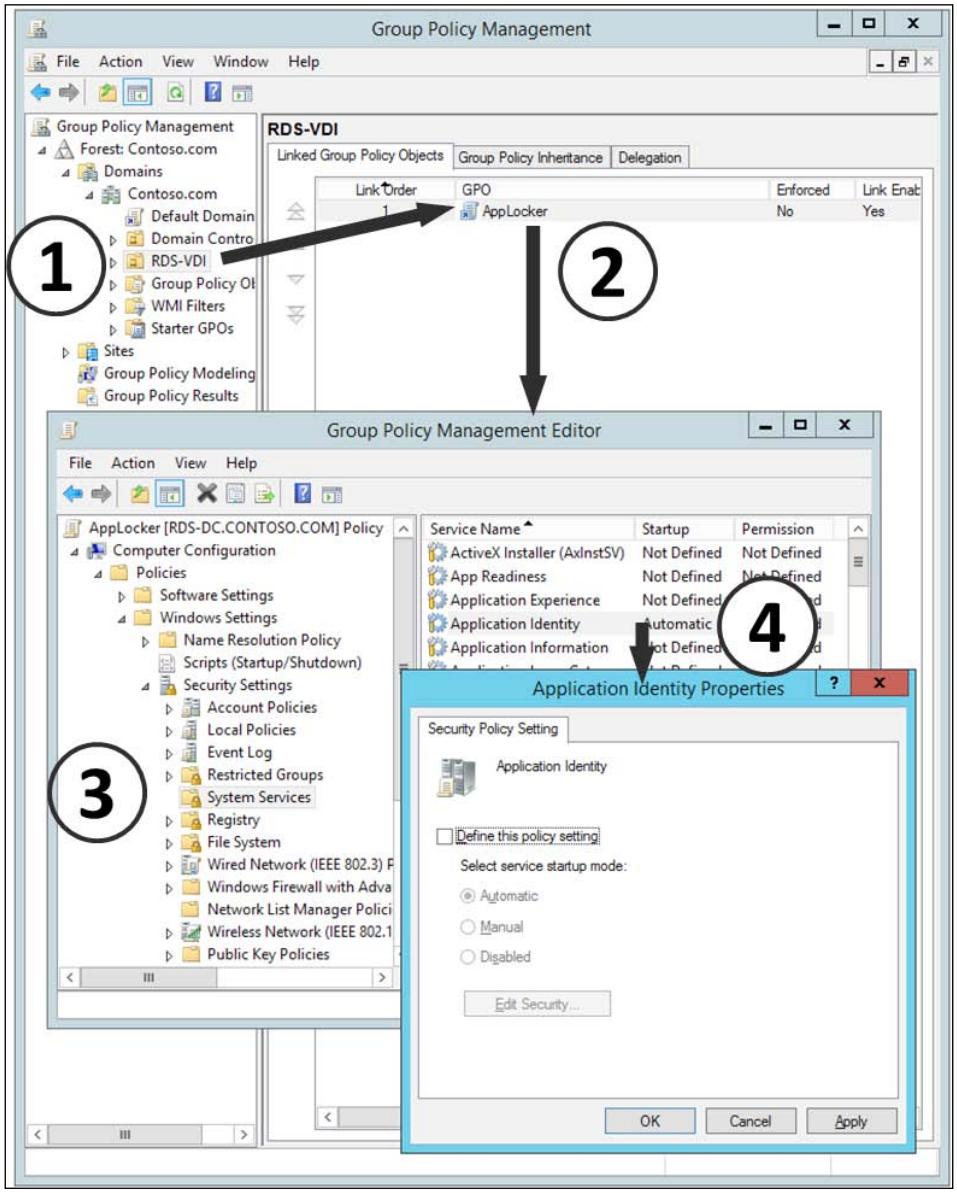
Application control

AppLocker replaced software restriction policies in Windows Server 2008 R2 as a more comprehensive set of tools to manage which applications our users can and cannot run. There is a special service built into Windows called the **Application Identity Service**, which is used by AppLocker to collect which applications are running them, to audit, block, or allow them. The rules themselves are defined in the AppLocker wizard built into Group Policy, and there are special PowerShell commands for AppLocker as well. In Windows Server 2012 R2 / Windows 8.1, AppLocker can also be used to control modern applications. There are two approaches that we can adopt when using it, and while these can be combined, the principle of least privilege applies to both of them. They are as follows:

- **Blacklist applications**: We declare which applications our users are forbidden to run, so if the application is *not* listed, the user *can* run it.
- **Whitelist applications**: We declare only those applications that we want our users to run.

Before you can work with these policies, you need to enable the application identity services on your virtual desktops, which you can do from Group Policy as well, as follows:

1. On the domain controller RDS-DC, open the **Group Policy Management** tool. Expand the **Contoso.com** domain and select the organizational unit, **RDS-VDI**, as shown in the following screenshot:



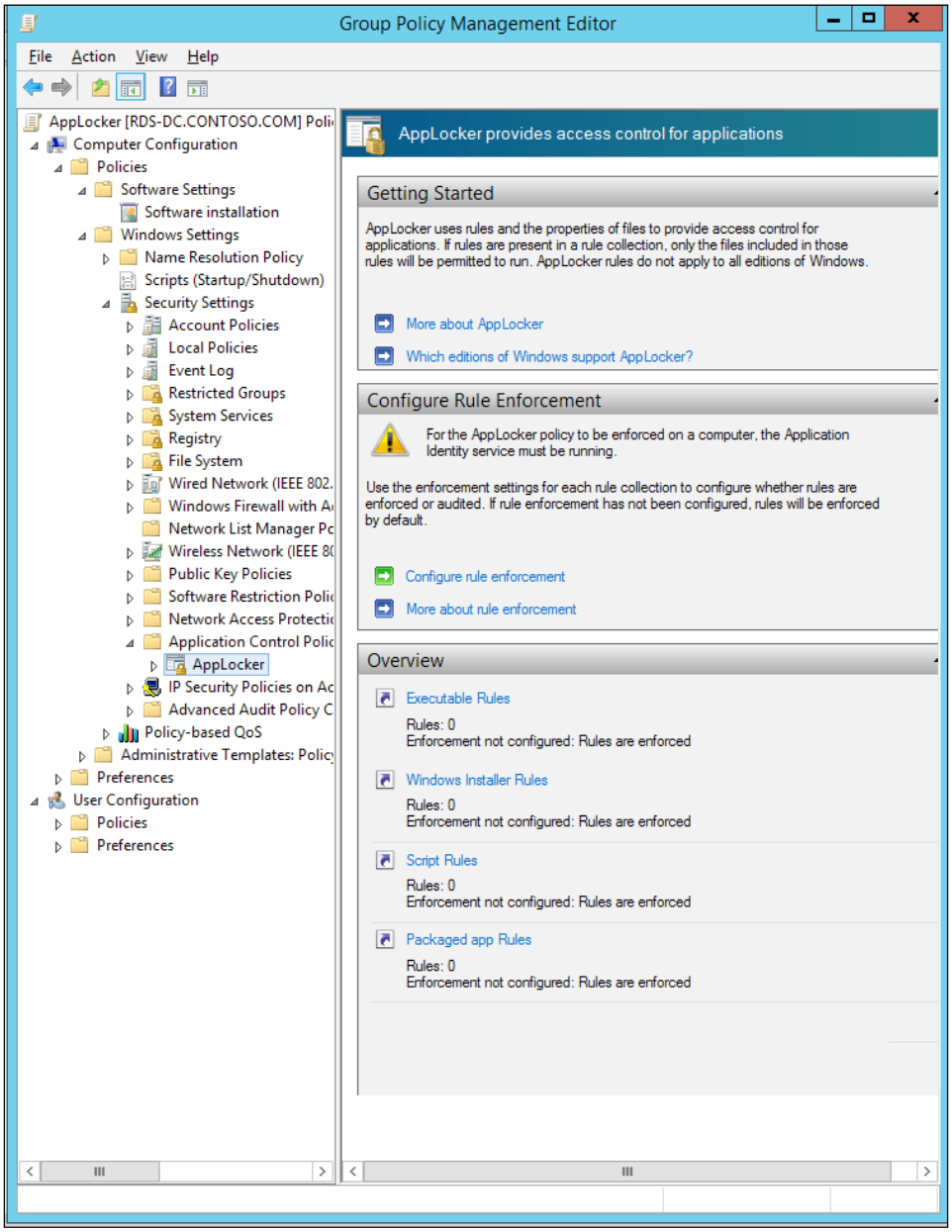
2. Right-click on this, select **Create a GPO in this domain**, and link it here.
3. Right-click on the new Group Policy and select **Edit**.
4. In **Group Policy Management Editor**, navigate to **Computer Configuration | Policies | Windows Settings | Security Settings | System Services**; then, select the application identity service, right-click on it, and select **Properties**. Check **Define this policy setting** and set the **Service Setup** mode to **Automatic**.
5. Click on **OK** to close this, and then close **Group Policy Management Editor**.

You could also change the preferences in Group Policy to start this service, which is the newer way to do this sort of thing in Group Policy. Perform the following steps:

1. In **Group Policy Management Editor**, navigate to **Computer Configuration | Preferences | Control Panel Settings | Services**.
2. Right-click on **Services** and click on **New**. In the **General** tab, click on the ellipsis to find **Application Service** (**AppIDSvc**), and choose the service action as **Start Service**. The clever thing here is that if there's a problem, you can handle it by setting options in the **Recovery** tab. You can set the first, second, and subsequent failures as well and select the retry interval.

The policy has precedence over a preference if both are applied to the same thing.

3. Now, you are ready to set up and test the rules you want to enforce on the users. Add this to the policy you just created using **Group Policy Management Editor** and by navigating to **Group Policy Management Editor expand Computer Configuration | Policies | Windows Settings | Security Settings | Application Control Policies | AppLocker**:



First, we can opt to enforce our policies. If we click on the link on the wizard screen on the right to **Configure Rule Enforcement**, we can not only control each type of executable file, but we can also elect enforce rules or click on each dropdown and select **Audit Only**. The **Advance** tab here also allows us to control DLLs, but this can be dangerous as we might easily block a DLL that an application depends on. DLL control will also mean a lot of management and, potentially, huge policy files if we decide to blacklist or whitelist them.

For executable files, Windows installers (`.msi`), and scripts, we can choose which level to declare rules at:

- The application publisher
- The application path
- The file hash—a unique signature restricted to a version of an application

This is similar for modern Windows 8.1 applications as well (which AppLocker refers to as **Packaged Apps**):

- Publisher
- Package name
- Package version

AppLocker can also be run from any machine, such as our reference computer. We just navigate to it from inside **Group Policy Management Editor** (**GPEdit.msc**). This means that we can scan the programs on a desktop, create a template we can export, and then import it to our domain policy. This is particularly useful for modern apps in Windows 8.1.

The Windows Store and modern control panel are both modern applications, so remember to include rules to allow or deny these as well in your AppLocker policies.

Having done a scan and decided which users will be subject to the rules, we can then export them by right-clicking on the **AppLocker** node in the editor and selecting **Export** to create an XML file to be imported elsewhere. In fact, we can load this into MDT and deploy directly as well, if necessary.

Summary

We had a good first look at what can be done to design and configure our virtual desktops using MDT, to store our configuration setting for the OS and for any applications we want in our collections. We have the workings of a process where we can automate the creation of a collection from a PowerShell script. We have also seen that we can configure Group Policy packs inside MDT if we want to. Although, for VDI, it is simpler to rely on our virtual desktops and users that belong to an OU that we can apply policies to. Finally, we saw that there are a number of Group Policy settings specific to RDS and VDI, such as controlling USB redirection, as well as other useful features, such as AppLocker, to restrict our users from running designated applications. Now that we have the tools to build virtual desktops for the enterprise, we can turn our attention to delivering these to our users, some of whom might be working at remote locations, such as from home, in a branch office, or on site with a customer. So, in the next chapter, we are going to put the R in Remote Desktop Services.

4
Putting the R in Remote Desktop

Many users now spend more and more time doing work away from the office, driven partly by flexible working practices, and because reliable mobile communications are available more widely and at a reduced cost. So, in this chapter, we will look at how to extend VDI outside of the local network so that our remote workers can use a virtual desktop wherever they can get a connection and on whatever device they have. This will include looking at several technologies not specifically related to VDI; we'll need to create trusted certificates to enable secure communications with these remote users. We'll need to look at firewalls and see how to authenticate our users without exposing too much of our **Active Directory** (**AD**) infrastructure.

We'll need to work with quite a few VMs to emulate the infrastructure surround in VDI, as it gets extended to the Internet, if the lab environment is to properly reflect what even a basic production environment looks like. I shall indicate which VMs we can shut down to keep this to a minimum.

Introducing the Remote Desktop Gateway

Modern work styles, branch offices, and having our users work closely with our clients at their locations are just some of the reasons why users will want access to corporate resources when they aren't in the office (or head office). If we can extend our VDI to these users, then they will have nearly the same experience as they would at work (if the communications are fast and reliable enough!).

We could just leave our VDI as is and implement a VPN to create a connection to our RD Web Access Server, and if the device they are connecting from is joined to our domain, then we could go a stage further and implement the Microsoft Direct Access and Built-in VPN to create a connection without any third-party software at either end. However, there is no need to do any of this if all we want to do is extend VDI to our remote users as there is a special role in RDS called the Remote Desktop Gateway (I shall just call it the RD Gateway from now on) that helps with this. Like any gateway, it is designed to be secure and is typically installed on a perimeter network, that is, it is installed on a network that is separated from the internal network by at least one firewall. It will use **Secure Socket Layer** (**SSL**) to establish an RDP session with remote users; this means that we will need the certificate infrastructure used in secure web traffic to support it. The RD Gateway also needs to identify and authenticate our users, and therefore, it needs access to our AD infrastructure. There are several ways of configuring the RD Gateway to achieve these objectives depending on our requirements and on the core infrastructure we already have in place to expose IT resources externally. Before we consider this, we need to understand more about the infrastructure it relies on.

Certificates

We need to have a basic understanding of **Public Key Infrastructure** (**PKI**) and how certificates are used in SSL to ensure that the remote access to our RD Gateway works. A full discussion on this is outside the scope of this book, so I will direct you to TechNet (`http://technet.microsoft.com/en-us/library/dd277320.aspx`) and to this popular primer on the subject by Steve "Planky" Plank (`http://blogs.msdn.com/b/plankytronixx/archive/2010/10/23/crypto-primer-understanding-encryption-public-private-key-signatures-and-certificates.aspx`) for more information.

SSL has actually been superseded by **Transport Layer Security** (**TLS**), but many of us still refer to this as SSL. However, you will also see references to TLS in TechNet. I'll call it SSL in this book to keep things simple.

We only need certificates in VDI to encrypt SSL between the gateway and our external users and not for the many other uses they also have. To do this, we need to have what's called an X.509 certificate, where X.509 is the industry standard definition for PKI. The simplest certificate we often see in demos is the self-signed certificate that can be created from inside **Internet Information Services** (**IIS**) or via PowerShell on a given machine and is typically used to create a secure website. We could use self-signed certificates for VDI, but in production, these aren't really feasible as they need to be managed manually. We would need to distribute and

install them on any other device used to connect to our VDI, and we would have to devise a process to renew them when they expire. This would cause us more security problems than it would solve, and our help desk would be flooded with calls from users struggling to connect. We actually saw evidence of this problem in the *Creating a Pooled Collection* section of *Chapter 2*, *Designing a Virtual Desktop Infrastucture*, when we connected to the RD Web Access portal (`http://rds-web.contoso.com/RDWeb`), where we got a certificate error, which we ignored, before we could connect. This occurred because the client we used didn't trust the certificate from the RD Web Access Server IIS site.

Fortunately, there is a much better way. Windows Server includes **Active Directory Certificate Services** (**AD CS**) to provide certificates so it can act as a **Certificate Authority** (**CA**). CAs do more than issue certificates; they also automatically revoke them and issue new ones, as needed, by maintaining **Certificate Revocation Lists** (**CRLs**) that are digitally signed for security purposes. The validity of certificates from clients connecting to our VDI is checked by an Online Responder service against the CRL that is using the industry standard **Online Certificate Status Protocol** (**OCSP**) that runs over HTTP.

When the role is installed, we get three options to choose from, as follows:

- **Standalone root CA**: This makes the server the highest level in a PKI hierarchy and is not integrated with AD at all
- **Enterprise root CA**: This also makes the server the highest level in a PKI hierarchy but is integrated with AD
- **Subordinate CA**: The server is part of, but not the root of, an existing PKI hierarchy

So, which of these options do we want for VDI? Before we answer that, we need to understand how our VDI will be accessed by our remote workers if we equip our remote workers with domain-joined Windows laptops or Windows To Go memory sticks.

Windows To Go is a Windows 8 / 8.1 Enterprise technology for running the OS from a special USB memory stick (`http://www.microsoft.com/en-us/windows/enterprise/products-and-technologies/devices/windowstogo.aspx`). The user boots from this technology, and the OS they run is domain joined, possibly with BitLocker enabled and the OS and storage on the actual laptop invisible to them; `C:` is just the storage on the memory stick.

All we have to do is use an Enterprise CA to generate our certificates as the remote devices will trust the certificate since they are in the same domain.

However, many of us might want to deploy certificates that can be trusted on any device that may not be joined to our domain and may not be a Windows device at all. To do this, we need to invest in a third-party root CA from companies such as DigiCert, GoDaddy, Symantec (VeriSign) Thawte, and so on (the full list can be viewed at `http://social.technet.microsoft.com/wiki/contents/articles/14215.windows-and-windows-phone-8-ssl-root-certificate-program-member-cas.aspx`), and then create our PKI subordinate for this root CA. Alternatively, we could just buy an SSL certificate from these same providers, in which case the certificates will be part of the third parties' PKIs. This is a much cheaper option, as we will just be using the certificate for one purpose on one or two gateway servers rather than deploying a multifunction PKI throughout our entire infrastructure. One important option to keep in mind when buying or configuring certificates in AD CS is the use of wildcards in the certificate name, technically known as **Subject Alternate Names** (**SANs**)—not to be confused with Storage Area Networks referred to elsewhere in this book!. These SAN certificates allow for the wider use of a certificate; for example, `*.contoso.com` allows us to get the certificate for `mail.contoso.com`, `vdi.contoso.com`, and so on.

For our initial evaluation, I am going to suggest that we simply use self-signed certificates, as the work done to set up AD CS is considerable and usually the responsibility of a dedicated team in a large organization.

Creating a self-signed certificate

The simplest way to create a self-signed certificate is to do this from any server that is running IIS, for example, RDS-DC. However, creating a certificate in IIS Manager doesn't give us much control over the certificate, and the one thing we can't do is define any alternative names. Fortunately, in PowerShell 4, there is now a set of PKI commands that allow us to do exactly what we want.

Don't do this in production! Self-signed certificates are, at best, difficult to manage and, at worst, a big security risk.

Firstly, we need to create the certificate on our domain controller RDS-DC, and the following command will create it at `LocalMachine\My`:

```
New-SelfSignedCertificate `
    -CertStoreLocation cert:\LocalMachine\My `
    -DnsName rds-Broker.contoso.com,rds.contoso.com,rds-gateway.com
```

Now, we can export the certificate with its private key for use on our other servers with the following commands:

```
$Cert = Get-ChildItem -Path cert:\localmachine\My | `
    where subject -EQ "CN=RDS-Broker.contoso.com, CN=RDS-
Web.contoso.com, CN=rds.contoso.com"


$pwd = ConvertTo-SecureString -String "Passw0rd!" -Force –AsPlainText
Export-PfxCertificate `
    -cert $cert `
    -FilePath 'C:\RDS-Web self signed certificate 1.pfx'  `
    -Password $pwd
```

There is also a PowerShell command called `add-PFXCertificate` that we could use to actually deploy the certificate to a designated part of a certificate store on any of our servers. However, these certificates are better controlled from the deployment properties in the RDS overview, and now that we have created our certificate, we can configure our gateway to use it.

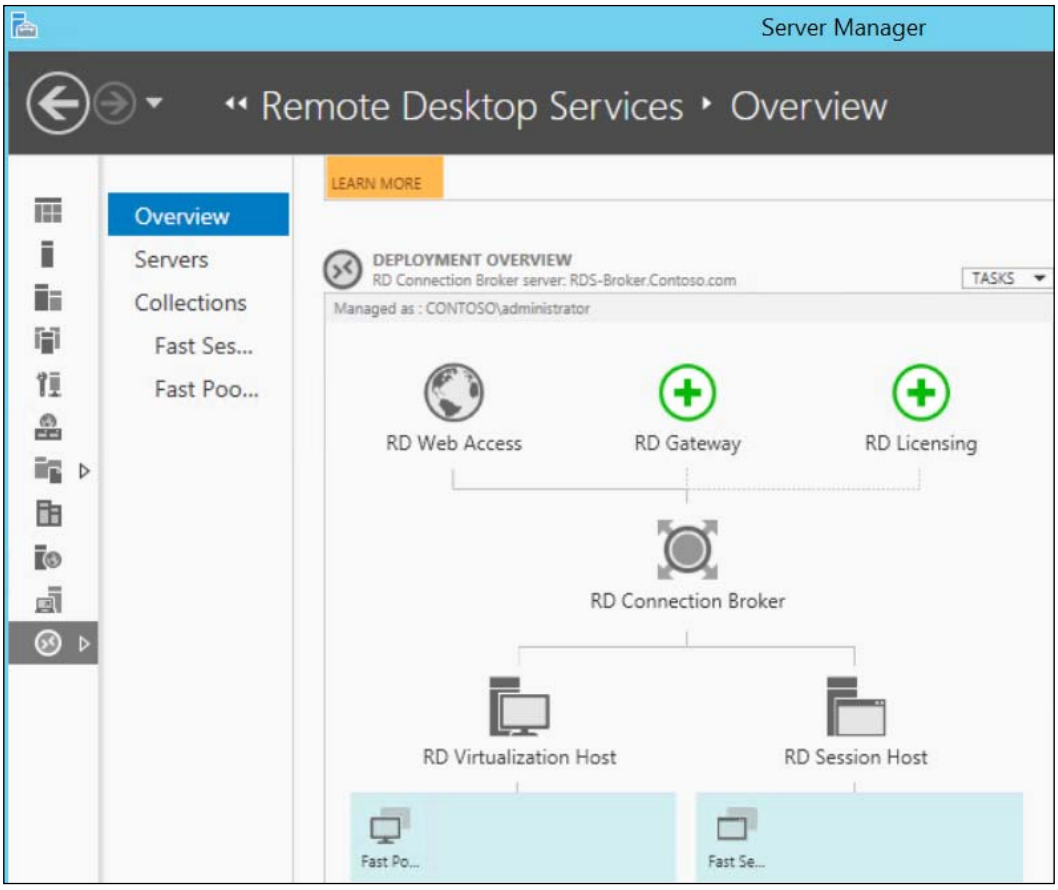
Getting started with the Remote Desktop Gateway

We can easily set up an RD Gateway with a special wizard within the RDS Overview of Server Manager. It's recommended best practice to co-locate the RD Web Access and RD Gateway roles, so that's what we'll do here.

Once this VM has been created, we need to manage and configure it. To do this from Server Manager, perform the following tasks:

1. Connect to the RDS-DC VM from Hyper-V Manager on the physical host. Open **Server Manager** and navigate to **Remote Desktop Services**.

2. In the RDS overview diagram, which was shown earlier, click on the **RD Gateway** icon (currently just a green plus sign) to launch the **Add RD Gateway Servers** wizard. The following screenshot shows the RD overview diagram:



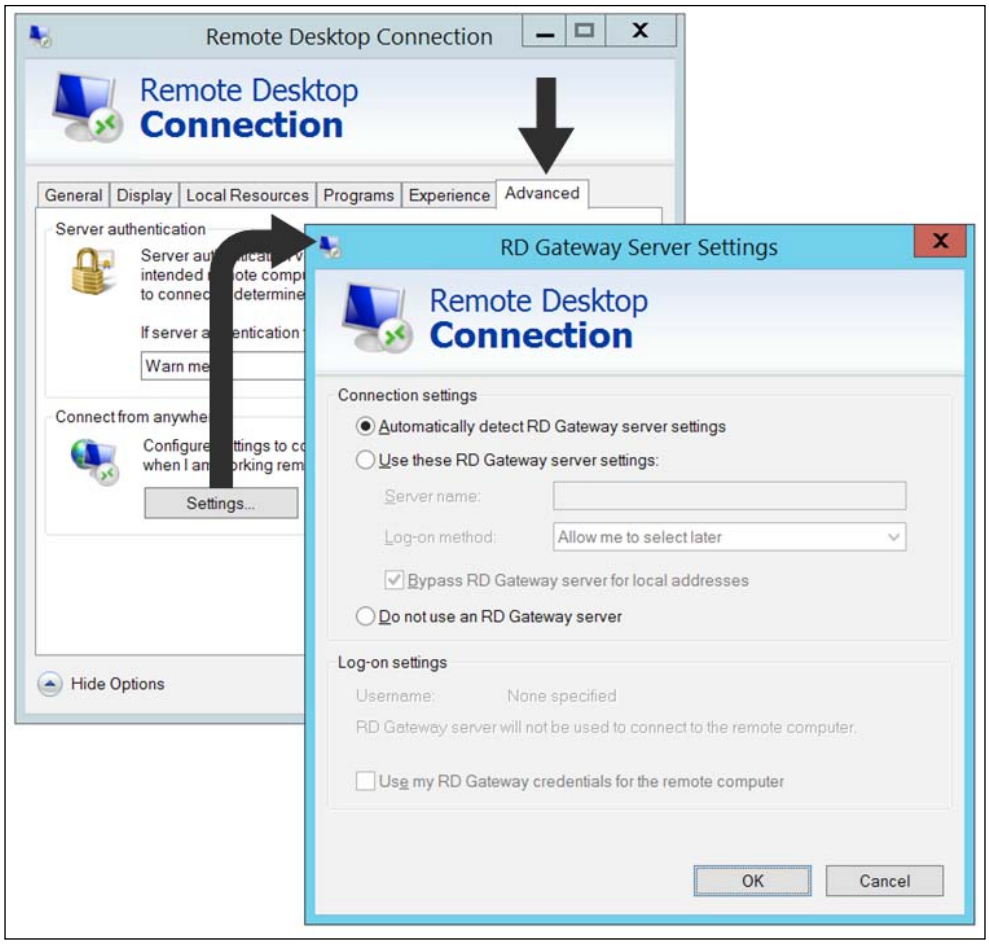
3. On the **Select a server** screen, click on the **RDS-Web** server from the server pool. Click on the middle arrow to add it to the list of selected servers and click on **Next**.
4. On the **SSL Certificate Name** screen, we are being reminded that the SSL certificate we use must correspond to the **Fully Qualified Domain Name** (**FQDN**) of our gateway server. Enter `RDS.contoso.com` and click on **Next**.
5. Click past the **Confirmation** screen, and click on **Add** to complete the wizard.

The gateway role has now been installed on RDS-Web; however, there is a warning at the bottom of the final **Progress** screen that says **The following role services require a certificate to be configured** and there is a **Configure** certificate hyperlink to do exactly this under the warning. The hyperlink will take you to the **Certificate** screen in the **Deployment Properties** wizard that we saw in the *Creating the virtual desktop template* section of *Chapter 2*, *Designing a Virtual Desktop Infrastructure*. We can use our new self-signed certificate to make then necessary changes to our deployment by doing the following:

1. Click on the **Configure** certificate hyperlink on the **Progress** screen to open the **Manage Certificates** screen of the **Configure Deployment** wizard that we saw in the *Setting up and configuring the RDS roles* section of *Chapter 2*, *Designing a Virtual Desktop Infrastructure*.
2. In the **Manage Certificates** screen, highlight the row marked **RD Connection Broker – Enable Single Sign On** and click on **Select Existing Certificate**.
3. Check the **Choose a different certificate** option and browse to the certificate we created earlier: `\\rds-DC\c$\RDS-Web self signed certificate.pfx`. Enter the password, `Passw0rd!`, and check the **Allow the certificate to be added to the Trusted Root Certificate Authorities certificate store on the destination computers** option. Click on **OK** to select the certificate and close the screen.
4. Note that since the certificate is now available, the **State** column for the **RD Connection Broker – Enable Single Sign On** row now says **ready to apply**. Click on **Apply** to complete the process.

The certificate for this role is now in place, and its level is set to **untrusted**. We can now repeat steps 2, 3, and 4 to add our certificate to each of the role services (RD Connection Broker-Publishing, RD Web Access, and RD Gateway). Note that when we highlight either the RD Web Access or RD Gateway role services, we are warned that because we have the web access role on the same server, we must use the same certificate for both of the roles. Before we close the **Deployment Properties** wizard, we need to look at the RD Gateway settings as well. From here, we can explicitly set the name of our RD Gateway server (in our case, `RDS-Web.contoso.com`), identify how we authenticate against the RD Gateway, and determine whether these credentials are then used for the remote computers, which for us are our virtual desktops.

We can also elect to bypass the gateway for local addresses; however, for testing purposes, we should uncheck this option so that the RD Gateway is always in use. All of these settings show up in two other places in the Remote Desktop client under **Advance Properties** | **Gateway Settings**. This brings up the following screenshot:



This screenshot shows that our users can change the settings, but actually, if they use the remote desktop client directly against our VDI like this, then what the client will try to do is actually log on to the gateway and not access our VDI collection at all; this attempt will fail as our users don't have rights to log on to this server. If they are on a device that is connected to our domain, we can control the gateway settings from Group Policy by navigating to **User Configuration** | **Administrative Templates** | **Windows Components** | **Remote Desktop Services** | **RD Gateway** and preventing them from logging on to our servers directly.

When we created the gateway, we specified the name for the gateway to be associated with our SSL certificate and set it to `rds.contoso.com`. However, this name doesn't actually exist; the name of our gateway is `RDS-Web.contoso.com`. So, for the RDS clients to resolve this name, create an alias (or `CName`) in DNS. Perform the following steps:

1. Connect to the **RDS-DC** VM, and from the **Tools** menu in **Server Manager**, select **DNS** to open the **DNS Manager** window.
2. Navigate to **rds-dc | Forward Lookup Zones**.
3. Right-click on **Contoso.com** and select **New Alias (CNAME)**.
4. Set the alias name to `RDS`, leave the FQDN as **rds.contoso.com**, and set the FQDN for the target host to `rds-web.contoso.com`. Click on **OK** to create the entry.

As usual, there is a simple PowerShell command to do this, as follows:

```
Add-DnsServerResourceRecordCName -Name rds -ZoneName contoso.com -
ComputerName rds-dc.contoso.com -HostNameAlias rds-web.contoso.com
```

We can now check whether our gateway is working by opening Internet Explorer either from RDS-DC or from the physical host and navigating to the URL `https://rds.contoso.com/rdweb`. Note that we still get the certificate warning we saw before, but since we just ignored it and signed in just now, it will prevent us from accessing our virtual desktops. We can confirm this by ignoring the warning again, proceeding to the site, and signing in as `contoso/RDSUser2` (password: `Passw0rd!`). When we try and access our Fast Session Collection, we get an error: **This computer can't verify the identity of the RD Gateway RDS-Web.contoso.com**. In order to fix this, you need to install the SSL certificate you have been using on the local machine that you are connecting from. Perform the following steps:

1. Click on the red **X** of the certificate in the address bar and select **View Certificates**.
2. Look at the **Details** tab of the certificate to confirm whether it's the one we made earlier with all our alternate names.
3. On the **General** tab, select **Install Certificate**.
4. Select the **Current User** option and click on **Next**.
5. Select the **Place all certificates in the following store** option and browse to **Trusted Root Certification Authorities**. Click on **Next**.
6. Click on **Finish** to install the certificate and then close the browser.

7. Go back to `https://rds.contoso.com/rdweb`. Note that we don't have a certificate error anymore.
8. Sign in as `contoso/RDSUser2` again (password: `Passw0rd!`), and select the **Fast Session Collection** option. You will now be logged in without any further requests for your credentials.

We now have a working RD Gateway before we have begun with our VDI deployment. However, before we proceed any further and begin to add more complexity to it, we should check whether the basic configuration works so that we know what to fix in case of any issues.

Active Directory authentication

We saw that when we connected to our VDI in the *Creating a Pooled Collection* section of *Chapter 2*, *Designing a Virtual Desktop Infrastucture*, we were asked to sign in with our domain credentials so that the broker role could identify who we are and assign us with the correct collection in our VDI. We also need to do the same thing for the RD Gateway, and in fact, the RD Gateway needs to be joined to our domain. So, what are our options, given that we will want to put the RD Gateway on some sort of perimeter network?

Opening additional ports on the firewall

We could just open up the ports we need from the perimeter network so that the gateway server can communicate with AD and the rest of our VDI. In this scenario, we would need to unblock the ports for a wide variety of traffic, for authentication and authorization to the domain, DNS, CRL, and the actual RDP traffic. Understandably, this isn't very secure and would probably be opposed by your network security team.

Relying on a forest trust relationship

It is possible to set up AD so that different domains trust each other, and this can be done in one or both directions. So, we could install a DC in our perimeter network and create a domain called `rds.dmz.com`, for example, and then join the gateway to that domain. We can then set up a one-way trust system between that domain and our corporate domain, `contoso.com`. This would cut down the ports we would need to open to just the AD Forest traffic and the RDP protocol. However, in this scenario, we have a fully working DC in our perimeter network, which might not be acceptable to the IT security team either.

Using a read-only domain controller

This was new for Windows Server 2008 and was originally designed for branch offices where the link to the head office could occasionally fail, but users could still log in to the branch by authenticating against the in-branch r**ead-only domain controller** (**RODC**). It works by caching specified users credentials, and we can control which accounts are cached. In our case, all we need is our VDI users, and we can block any caching of privileged domain accounts so that they are never stored on this DC. There does have to be one-way synchronization from the other DCs to the RODC for the maintenance of the designated RODC accounts, for example, as passwords are changed. Another consideration is DNS, and while it would be quite usual to have this running alongside a DC, in a branch scenario, this would be another security headache in a perimeter network.

I mention all of this because using one or more RODCs in the perimeter network is the recommended approach to AD authentication in VDI. Similarly, the recommended method to keep DNS up to date is to use DHCP to maintain DNS records.

There is a complete setup guide to deploy AD in a perimeter network, which includes the specifics of RODCs as well, available at `http://technet.microsoft.com/en-us/library/dd728028(v=ws.10).aspx`.

Creating an RODC

Creating an RODC used to be quite difficult, but among the many features in Windows Server 2012 R2, there are now more wizards and the option to save from the PowerShell script. For RODCs, there are two wizards. We can preauthorize a designated RODC from the Active Directory Administrative Console, or we can set up the RODC when we add the Active Directory Domain Services role. The reason there are two options here is because it is possible to create an RODC using an offline domain join when it's not connected to the network where our DCs are. However, we are doing this on a VM in Hyper-V so that we can create the RODC VM on our existing network for now and move it around once it is configured.

We are going to use two more VMs in this chapter to create the RODC (RDS-RODC) itself, and another to spoof the perimeter network; as usual, there is a script for this. If we don't have the resources to run any more VMs, then it's possible to test what we are doing by deleting any collection or stopping the initial VM we used in *Chapter 1*, *Putting the V in VDI – An Introduction to Virtualization in Hyper-V*. So, what you need to have running is the RDS-SHost (to provide session-based virtual desktops), RDS-Broker, RDS-Web (which is now our gateway), and RDS-DC VMs.

I think it's important to understand the basics of RODCs, so step through the wizard to make one on a clean VM that's already a member of the `contoso.com` domain (`RDS-RODC`, if you have used my script). Perform the following tasks:

1. Connect to the VM from **Hyper-V Manager** as `contoso/administrator`.
2. Install the Active Directory Domain Services role from the **Add Roles and Features** option or with the following PowerShell command:

   ```
   Install-WindowsFeature -name AD-Domain-Services -
   IncludeManagementTools
   ```

3. If you have used PowerShell, click on **Refresh** in **Server Manager**.
4. Click on the yellow warning in the top-right corner of **Server Manager** and select **Promote this Server to a domain controller** by launching the **Active Directory Services Configuration** wizard.
5. Review the settings on the **Deployment Configuration** screen and click on **Next** to add this server as a new DC into the `contoso.com` domain.
6. On the **Domain Controller Options** screen, check all of the options: **Domain Name System (DNS) Server**, **Global Catalog (GC)**, and **Read-only Domain Controller (RODC)**. Set the **Directory Services Restore Mode** (**DSRM**) password to `Passw0rd!`, confirm it, and click on **Next**.
7. On the **RODC options** screen, specify which accounts will be stored on the RODC and which will not as well as setting a delegated administrator for the RODC. We'll set `RDSAdmin` as the delegated administrator and leave the other settings as given on the screen. This will prevent key accounts from being stored on the RODC, and you can add in your VDI users to the Allowed RODC Password Replication Group later to ensure that their credentials are stored on the RODC. Click on **Next**.
8. Leave the settings on the **Additional Options** screen as they are and click on **Next**.
9. Leave the settings on the **Paths** screen as they are and click on **Next**.

10. On the **Review Options** screen, check whether the options are as you intended. You can see how to do all of this in PowerShell by clicking on **View Script**. The following command lines will be available in **View Script**:

```
Import-Module ADDSDeployment

Install-ADDSDomainController `
-AllowPasswordReplicationAccountName @("CONTOSO\Allowed RODC
Password Replication Group") `
-NoGlobalCatalog:$false `
-CriticalReplicationOnly:$false `
-DatabasePath "C:\Windows\NTDS" `
-DelegatedAdministratorAccountName "CONTOSO\RDSAdmin" `
-DenyPasswordReplicationAccountName
@("BUILTIN\Administrators", "BUILTIN\Server Operators",
"BUILTIN\Backup Operators", "BUILTIN\Account Operators",
"CONTOSO\Denied RODC Password Replication Group") `
-DomainName "Contoso.com" `
-InstallDns:$true `
-LogPath "C:\Windows\NTDS" `
-NoRebootOnCompletion:$false `
-ReadOnlyReplica:$true `
-SiteName "Default-First-Site-Name" `
-SysvolPath "C:\Windows\SYSVOL" `
-Force:$true
```

    This could easily be adapted to roll out multiple RODCs, for example, to branch offices.

11. Click on **Next** to run a prerequisite check.
12. On the **Prerequisite Check** screen, you'll get the usual warning about cryptography and NT4, which you can ignore, and you can configure the RODC by clicking on **Install**.

You now need to make a few changes in AD on your RDS-DC so that your users' accounts are on the RODC, and you need to remove the `RDSAdmin` account from the domain admins group; otherwise, it won't be replicated to the RODC, and you won't be able to manage it. Perform the following steps:

1. Connect to **RDS-DC** as `contoso/administrator` and open **Server Manager**. In the **Tools** menu, select **Active Directory Administrative Center**.
2. Select **Global Search** at the bottom of the navigation pane, and enter `Allowed` in the **Global Search** window. Right-click on **Allowed RODC Password Replication Group** and select **properties**. Navigate to **Members** and click on **Add** on the right-hand side of the screen.
3. In the **Select Users** dialog box, enter `VDI-Users;SessionUsers; RDS Administrators (RDSAdmin)` and click on **OK**.

4. Navigate to **Contoso (local)** | **RDS-VDI** and right-click on **RDS-Administrator**. Then, select **Properties**.
5. Navigate to **Member Of** and select **Domain Admins**. Click on **Remove** and then on **OK** to confirm and close this screen.
6. Repeat this for Enterprise Admins, as in both cases, the RDS admin account won't be replicated to the RODC if it's in either of these groups.

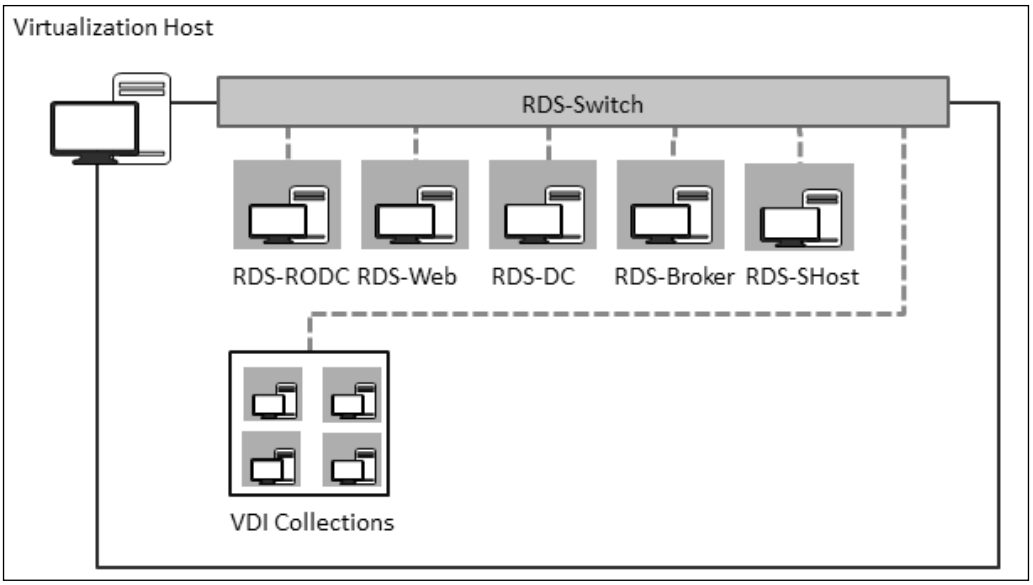
The equivalent PowerShell script can be captured from the bottom of the Active Directory Administrative Console as follows:

```
Set-ADGroup -Add:@{'Member'="CN=VDI-Users,OU=RDS-VDI,DC=Contoso,DC=com",
"CN=Session-Users,OU=RDS-VDI,DC=Contoso,DC=com", "CN=RDS
Administrator,OU=RDS-VDI,DC=Contoso,DC=com"} -Identity:"CN=Allowed RODC
Password Replication Group,CN=Users,DC=Contoso,DC=com" -Server:"RDS-DC.
Contoso.com"

Remove-ADPrincipalGroupMembership -Confirm:$false -Identity:"CN=RDS
Administrator,OU=RDS-VDI,DC=Contoso,DC=com" -MemberOf:"CN=Domain Admins,C
N=Users,DC=Contoso,DC=com" -Server:"RDS-DC.Contoso.com"

Remove-ADPrincipalGroupMembership -Confirm:$false -Identity:"CN=RDS
Administrator,OU=RDS-VDI,DC=Contoso,DC=com" -MemberOf:"CN=Enterprise Admi
ns,CN=Users,DC=Contoso,DC=com" -Server:"RDS-DC.Contoso.com"
```

Our RODC is now ready for use, but it has no real value as it stands because our RDS-DC is available and our VDI doesn't know about this new DC.

Creating the perimeter network

So far, in this book, we have created several VMs to create a simple VDI in a box for our labs. All of these VMs that we just created, including the gateway and RODC, are connected to the same Hyper-V virtual switch (RDS-Switch)—it can be considered our private or corporate LAN—and so our current lab setup looks like the following:

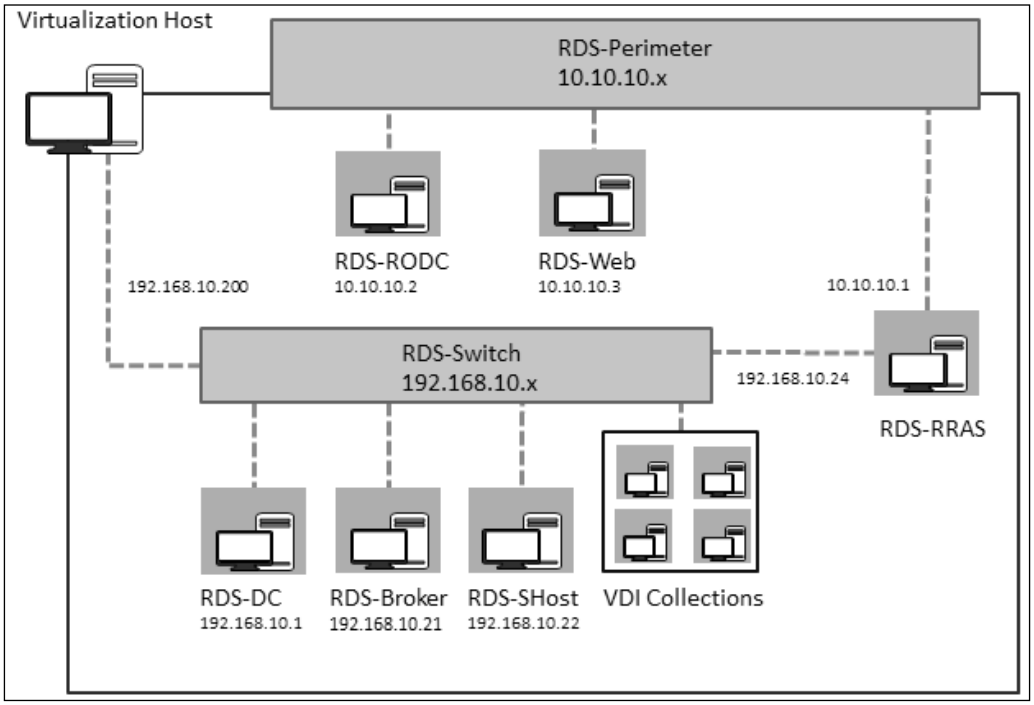


The current lab setup

Since RDS-Switch is an external virtual switch bound to a physical NIC on our host, the LAN extends beyond the physical host, and we are able to connect to our VDI from whatever was connected to that NIC. However, we can create more than one virtual switch on a Hyper-V host, and as we saw in *Chapter 1*, *Putting the V in VDI – An Introduction to Virtualization in Hyper-V*, each of these can be private, internal, or external; additionally, the type can be changed from one to the other even if running VMs are using the switch (it's a bit like changing the patch cable in our real switches). So, we can create a new virtual switch to handle our perimeter network (we'll call it RDS-Perimeter) alongside our existing RDS-Switch and then make the RDS-Switch an internal virtual switch so that it's only visible to our host and the VMs connected to it. Our new RDS-Perimeter switch will now be our external virtual switch; note that only one virtual switch can be bound to a physical NIC, but this NIC will *not* be available to the host (remember to use the setting **Allow the management operating system to share this network adapter**, which if not set, will mask this network adapter from the host).

In production, you might well decide to run all of your VMs in a perimeter network on a dedicated host. The host would never be connected to the perimeter network itself, but would typically have another physical NIC used expressly to manage it.

Just to be clear, at this point, a VM connected only to RDS-Perimeter cannot communicate at all with a VM connected only to RDS-Switch. In a real perimeter network, there would be limited connectivity between the two, but this would be subject to routing, and they would probably be separated by a firewall so that only certain ports, protocols, and applications were able to traverse the two networks. This is easy to replicate in Windows Server with the built-in firewall and **Routing and Remote Access Services** (**RRAS**) role. Putting all of this together results in a setup like the following diagram:



The design for our RDS Gateway lab

Here, we now have the **RDS-Perimeter** on a new `10.10.10.x` subnet, and our physical host is only connected to **RDS-Switch**, which is now internal. I am going to suggest that we use a dedicated VM (**RDS-RRAS** in the diagram) as this mimics what there would be in production; this VM also has two virtual NICs and is connected to both of the virtual networks. Once we have this VM configured, we can move our gateway and RODC VMs to the new perimeter network by simply changing their static IP address, connecting them to the RDS-Perimeter virtual switch, and updating DNS to reflect their new location.

Configuring the virtual switches

Before you can configure anything else, you need to create the new virtual network and adjust the settings of your existing virtual switches (RDS-Switch). Perform the following tasks:

1. Create the RDS-Switch VM and internal virtual switch by performing the following steps:
    1. On the host server that you are using, open **Hyper-V Manager**.
    2. From the **Actions** pane, select **Virtual Switch Manager**.
    3. Select **RDS-Switch** and check the **Internal Network** option. Then, click on **Apply**.
2. Create the RDS-Perimeter switch and make it an external virtual switch by performing the following steps:
    1. While still inside the Virtual Switch Manager, select **New Virtual network switch** on the left-hand pane.
    2. Select **External** and click on **Create Virtual Switch**.
    3. Name the switch `RDS-Perimeter`, select the appropriate switch from the drop-down box under **External** network, and deselect **Allow the management operating system to share this network adapter**.
    4. Click on **OK** to confirm the settings and close the Virtual Switch Manager.

The equivalent PowerShell script is as follows:

```
Get-VMSwitch | where name –eq RDS-Switch | Set-VMSwitch –SwitchType
Internal
$PhysicalNIC = Get-NetAdapter | where interfacedescription -Like
"*gigabit*"
New-VMSwitch -NetAdapterInterfaceDescription $PhysicalNIC.
InterfaceDescription -Name RDS-Perimeter
```

Here, you'll need to change `*gigabit*` to an appropriate search term to find the physical NIC in your host.

Configuring Routing and Remote Access

The script used to create the RODC that we configured earlier also created another VM: RDS-RRAS. This VM is not a member of the `contoso.com` domain; it has two virtual NICs and the Routing and Remote Access feature already installed on it. However, it's not configured as it's better to be able to configure a server like this when testing similar VDI scenarios.

If you want to create a blank VM for this, then you just need to preconfigure it with the following PowerShell script:

```
#Add in the routing role into the RRASVM while the VM is
off but has the server OS installed
Add-WindowsFeature -Vhd (path to your new VHD) -Name
"routing" -IncludeManagementTools -ComputerName orange

#Add a second NIC and leave it unconnected
Add-VMNetworkAdapter -VMName $RRASVM
```

The traditional Routing and Remote Access role is now inside the much more powerful Remote Access role in Windows Server 2012 R2, which also covers Direct Access, reverse proxy capabilities, and VPN. All you need here is a simple router; to get this router, perform the following steps:

1. Connect to the **RDS-RRAS** VM in **Hyper-V Manager** as `.\administrator` (password: `Passw0rd!`). Open **Server Manager** and navigate to **Local Server**. Click on the IP address for **Ethernet** to open the **Network Connections** dialog box.
2. Right-click on **Ethernet2** and select **Properties**.
3. Highlight the row for **Internet Protocol Version 4(TCP/IPv4)** and click on **Properties**.
4. Set the IP address to `10.10.10.1`, the subnet to `255.255.255.0`, and the gateway to `10.10.10.2`. Click on **OK** to close this window, and click on **Close** on the **Ethernet 2 properties** window.
5. Go back to **Server Manger**, and click on the refresh icon to confirm whether the IP address for Ethernet2 is now set.
6. From the **Tools** menu in **Server Manager**, select **Routing and Remote Access**.
7. Right-click on the **RDS-RRAS** server and select **Configure and Enable Routing and Remote Access** to launch the **Routing and Remotes Access Server Setup** wizard.
8. Click on **Next** on the welcome screen.

9. Select **Custom Configuration** and click on **Next**.
10. Select **LAN Routing** and click on **Next**.
11. Confirm that the **LAN Routing** option has been selected, and click on **Finish** to complete the wizard.
12. Click on **Start Service** to enable routing.

If we have a look at the **RRAS** manager now, we will see that our service is green. If we navigate to **IPv4 | General**, there are four interfaces, namely, **Loopback**, **Internal**, **Ethernet**, and **Ethernet2**, and the last two have a type named **dedicated**. If we right-click on either Ethernet interface to look at its properties, we will see that **Enable IP router manager** is enabled. All we need to do now is move our VMs across to the new virtual network and alter some network and DNS settings, and we will have implemented what is in the preceding diagram.

Completing the gateway design

You now need to move the VMs, RDS-Web and RDS-RODC, into the perimeter network, which is simply a matter of connecting them to the RDS-Perimeter switch. Perform the following tasks:

1. In **Hyper-V Manager** on the host, right-click on **RDS-Web** and select **Settings**.
2. In the **Network Adapter Settings** dialog box, change the virtual switch to RDS-Perimeter and click on **Apply**.
3. Repeat the process for RDS-RODC by selecting **RDS-RODC** from the top-left corner of the **Settings** screen, set its network adapter to **RDS-Perimeter**, and click on **OK** to close the **Settings** window.

The VM can be left running. This is the virtual equivalent of swapping the router that a real server is connected to.

We now need to reconfigure the IP addresses of these servers to reflect the new subnet we are using in the perimeter network (`10.10.10.0/24`) and the location of our IP gateway (our RDS-RRAS server), as shown in the design diagram for the network.

We can do this in each VM by navigating to **Server Manager** | **Local Server** and clicking on the current IP address for the server to change it in **Network Connections**. The settings we need to change are the IPv4 properties of the Ethernet adapter, and they need to be set as follows:

- The RDS-RODC IPv4 address (`10.10.10.2`), subnet mask (`255.255.255.0`), gateway (`10.10.10.1`), and DNS server (`127.0.0.1`)
- The RDS-Web IPv4 address (`10.10.10.3`); subnet mask (`255.255.255.0`); gateway (`10.10.10.1`); and DNS server (`10.10.10.2`), that is, RDS-RODC

The PowerShell script to do this is harder to understand as there is no native command, so we have to call a **Windows Management Interface** (**WMI**) using the following command:

```
$wmi = Get-WmiObject win32_networkadapterconfiguration -filter
"ipenabled ='true'";
```

There's a good article on how to run commands like this across multiple servers at `http://technet.microsoft.com/en-us/library/ff730958.aspx`.

In the same way, you also need to alter the IP gateway settings on the various servers in your internal network to point to the RRAS Server (`192.168.10.24`) so that the DC can update the RODC and the RDP traffic can be routed between the RD Gateway and the RD Broker. You'll have to do this manually as these servers have static IP addresses.

Finally, you need to update the DNS records on your internal network to point to the new addresses for the RD Gateway and RODC by performing the following steps:

1. Connect to the **RDS-DC** VM, and from the **Tools** menu in **Server Manager**, select **DNS** to open the **DNS Manager** wizard.
2. Navigate to **rds-dc** | **Forward Lookup Zones**.
3. Right-click on **RDS-Web** and select **Properties**.
4. Set the IP address to `10.10.10.3` and then click on **OK**.
5. Repeat this for RDS-RODC, but set its new IP address to `10.10.10.2`.

As usual, there is a PowerShell command to do this, as follows:

```
CLS
$DNSZone = 'contoso.com'
$DNSServer = 'RDS-DC.contoso.com'
$OldDNS = get-DnsServerResourceRecord -Name RDS-RODC -ZoneName $DNSZone
```

```
-ComputerName $DNSServer
$NewDNS = get-DnsServerResourceRecord -Name RDS-RODC -ZoneName $DNSZone
-ComputerName $DNSServer
$NewDNS.RecordData.IPv4Address = "10.10.10.2"
Set-DnsServerResourceRecord -OldInputObject $OldDNS -NewInputObject
$NewDNS -ZoneName $DNSZone -ComputerName $DNSServer -PassThru

$OldDNS = get-DnsServerResourceRecord -Name RDS-Web -ZoneName $DNSZone
-ComputerName $DNSServer
$NewDNS = get-DnsServerResourceRecord -Name RDS-Web -ZoneName $DNSZone
-ComputerName $DNSServer
$NewDNS.RecordData.IPv4Address = "10.10.10.3"
```

This PowerShell script also shows how a variable (in this case, `$NewDNS`) inherits the complete methods and properties of the object it is set to.

After all of these changes to your networking, you now need to test that your VDI is still working, as all you have done so far is to put your servers on a different subnet. Perform the following tasks:

1. In **Hyper-V Manager** on the host, connect to the RDS-RODC VM and open Internet Explorer.
2. Browse to `https://rds.contoso.com/rdweb`.
3. You will get the familiar certificate warning in Internet Explorer again, and you need to install this certificate in the folder located at `personal/trusted root certificates`.
4. Sign in as `contoso/RDSUser2` (password: `Passw0rd!`) and connect to **Fast Session Collection**.

The next thing you can do is test whether the RODC is working by temporarily disabling the main DC (RDS-DC). To do this, right-click on the **RDS-DC** VM in **Hyper-V Manager** on the host and select **Pause**.

Saving a VM is a bit like putting a laptop into hibernation; the memory state is written to disk, and when the laptop or VM is resumed, the memory state is read back into the RAM. Most of our VMs in a large Pooled Collection will be in the saved state when they aren't in use, as they won't take up any CPU or RAM in this state and can be restored to the point they were at before the save.

We can now rerun our previous test to ensure that our user accounts are available on the RODC and that our VDI infrastructure still works properly. If there are authentication problems, we can resume the DC (from Hyper-V Manager, right-click on the VM and select **Resume**) and fix any issues and retest. When everything is working, resume the DC as we'll need it again later.

Locking down the perimeter network

At the moment, our RRAS router is not blocking any traffic; it's just that our RODC and RD Gateway are on a different subnet from the rest of our VMs. In reality, there would be firewalls blocking all but the essential traffic between these two networks. We can simulate this on our RRAS box now that we have our routing working and our RODC is authenticating our users as we want. If anything breaks as we implement any firewall rules, we'll know that it's these rules that are causing the issue and not anything else. Before we implement the rules, we need to plan what we are doing.

Active Directory

The DC needs to update the RODC as we change passwords, create new users, and permit them to use our VDI collections by adding them to the relevant groups. As noted earlier, there is quite a list of ports that are needed for the AD replication traffic, but the good news is that we only need to allow this traffic into the perimeter network from our domain controllers and block all of the traffic coming back the other way. The static ports are listed in the following table:

Service	**TCP**	**UDP**
The RPC endpoint mapper	135	135
The RPC static port for AD replication	These are dynamic but can be limited by the setting of registry keys on the DC	
Kerberos	88	88
LDAP	389	LDAP Ping 389
LDAP over SSL	636	-
Global catalog LDAP	3268	-
Global catalog LDAP over SSL	3269	-
SMB over IP (Microsoft-DS)	445	445
DNS	53	53

Service	TCP	UDP
NTP	123	-
FRS	Normally dynamic, but can be changed in the registry; for example, in `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters`, change `"TCP/IP Port"=dword:0000c000`, where `dword(which is in hex) = 0000c000` is port `49152`.	

There is also a **Remote Process Call** (**RPC**) service, and the ports for this are dynamic, and several are needed. However, we could adopt a different approach and implement **IP security** (**IPsec**) between the DC and RODC. If we do this, we can just create a specific route on our router to allow all traffic from the DC (`192.168.10.1`) to the RODC (`10.10.10.2`) on our RRAS server.

There's a complete guide to configuring IPsec at `http://technet.microsoft.com/en-us/library/bb742429.aspx`.

What you can do in your lab to emulate this is just create the rules you want in RRAS. There are two mechanisms. You can use inbound filters in RRAS itself, or you can configure the Windows Firewall ports. Note that if you do both, then the traffic will have to satisfy both sets of rules to get through to a destination. You can perform a simple test to start with, using the following steps:

1. Connect to the **RDS_RRAS** VM as `.\Administrator` (password: `Passw0rd!`) from **Hyper-V Manager** on the host.
2. From the **Tools** menu in **Server Manager**, select **Routing and Remote Access** to open the Routing and Remote Access console.
3. Navigate to **RDS-RRAS** | **IPv4** | **General**, right-click on **Ethernet** (which should have an IP address of `192.168.10.24`), and select **Properties**.
4. Click on **Inbound Filters**.

5. Click on **New** and enter the following:

Property	Setting
Source Network	X
IP Address	`192.168.10.1`
Subnet Mask	`255.255.255.255`
Destination Network	X
IP Address	`10.10.10.2`
Subnet Mask	`255.255.255.0`
Protocol	Any

6. Click on **OK**.
7. Check the **Drop all packets except those that meet the criteria below** option, and click on **OK** to close the **Inbound Filters** screen.
8. Click on **OK** to close the **Ethernet Properties** screen.

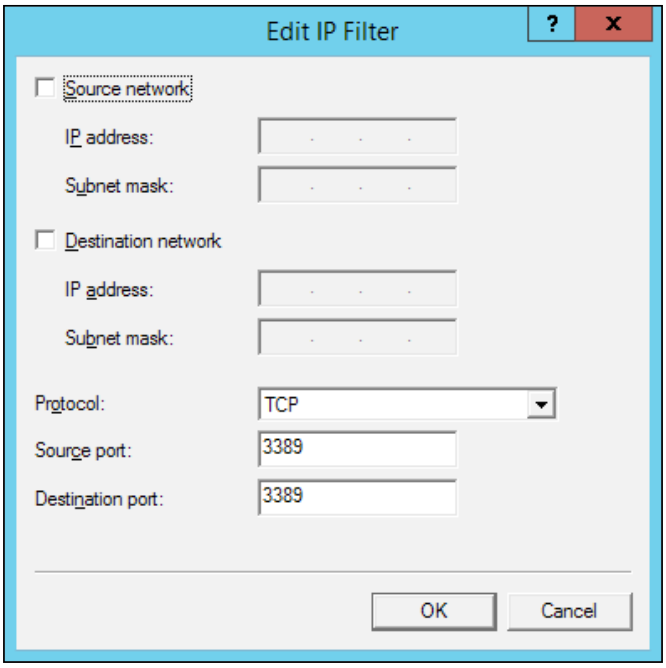
If you now connect to the RDS-DC VM and ping RDS-RODC, you should get a reply, but if you ping RDS-Web, it will fail. If you connect to RDS-Broker and try these tests, they will all fail.

The remote desktop

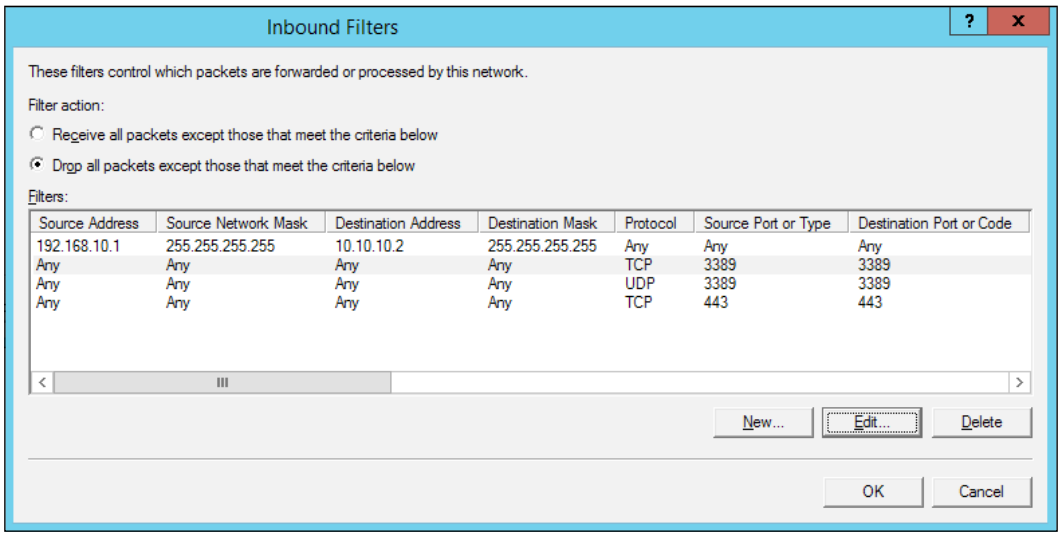
Clearly, we also need to allow **Remote Desktop Protocol** (**RDP**) traffic in and out of our internal network. We'll need a few other ports open as well, as shown in the following table:

Service	TCP	UDP
RDP	3389	3389
		3391
SSL	443	-
The RD Web Access connection to the RD Broker as the web server is now on our perimeter network	5504	-
WMI and remote PowerShell	5985	

We can apply these rules in RRAS by specifying the protocol, source, and destination ports as well as the source and destination IP addresses, as shown in the following screenshot:



The resulting filters will look like the following screenshot:



As soon as we apply any of these rules, all other kinds of traffic coming from the perimeter network into the internal network will be blocked, so I recommend testing at each stage to save time and frustration!

Remote access without using the gateway

When we set up the RRAS service to bridge our network, we installed the routing role service into our RDS-RRAS VM, which is part of the Remote Access role. The other parts of this role allow us to create connectivity for our remote users that can be used for a wide variety of purposes, including VDI. Direct Access allows domain-joined Windows 7/8 clients to seamlessly join the corporate network over the Internet using a variety of secure protocols, and will resort to SSL if that's all that will work. Direct Access is very easy to set up as there is no extra software on the client, and all the configuration changes are made up centrally with Group Policy. There is also a fully functional VPN server in Windows Server 2012 R2 that can be accessed from any device. The VPN technology can work with most VPN security providers (Dell SonicWall, Juniper, RSA, Cisco, F5, and so on) and is supported on IOS4 VPN and IOS Lion as well as on the Android smartphone client (the details are available at `http://technet.microsoft.com/en-us/library/jj613765.aspx`). If any of these technologies are used, then we won't really need the RD Gateway, and all we'll need to do is work with the security team that implements remote access to ensure that our VDI users have a good experience.

Summary

In this chapter, we have seen that Windows VDI has a built-in role, the RD Gateway, which allows us to provide virtual desktops to our remote workers over a secure Internet connection from nondomain-joined devices. The security of these connections is based on SSL, and so the gateway makes use of trusted X.509 certificates, much the same way as a secure website would. We typically put the gateway in a perimeter network or DMZ, but it will only work if we have a way of authenticating remote users with AD. We did this by putting a special read-only domain controller in the same perimeter network. This complexity has made this chapter the hardest one to understand in this book, but that's basically because security is hard, and in many organizations, this is done by a dedicated team. It's also worth mentioning that while we did some basic work with certificates, we would never use self-signed certificates in production. Finally, for production VDIs, there would be some sort of enterprise firewall in front of the perimeter network and possibly between it and the internal network, and there would be a public IP so that our users can get to it from wherever they are.

Now that we can offer secure virtual desktops to our remote workers, we will probably need to think about resiliency, as our remote users might be working late or be connecting in from different time zones. In the next chapter, we will see how to scale up VDI and, at the same time, make it highly available so that we can survive an outage of any of the parts of our deployment caused either due to a failure or because we need to do some sort of planned maintenance.

5
High Availability

This chapter is all about making VDI more available, and to do that, we need to ensure that each role in our VDI deployment has some sort of redundancy so that if there is any kind of failure, there is a standby server that can immediately provide the same service. So we will look at how to enable HA for each role (the RD Broker, the RD Gateway, and RD Web Access server) and how to enable HA in each of the different types of virtual desktop collections. At each stage, we'll implement this in our lab setup wherever possible.

Why high availability matters for VDI

What makes VDI different from some other parts of the IT fabric is that if it's very close to our users and if we give virtual desktops to a group of users, then those users are wholly dependent on them to access their work and other systems. So, if we are going to put VDI into production, we are going to need to implement HA completely for it. Given that the physical hardware is pretty reliable these days, the main reason we implement HA is to allow us to do planned maintenance on parts of a service without stopping it, or because we need to reverse a change we made that has caused our VDI to become unstable. Of course, things will go wrong as well, but that's more often caused by configuration changes than something physical that ceases to work. In either case, we need to minimize the impact of these issues on downtime for our users and the amount of work they might lose. Looking at all of this from our users' point of view, we need to ask, "Can we connect to our VDI and get a new virtual desktop?" and "What happens to our sessions and work if something fails while we are already logged on to a virtual desktop?" Even with the best design, our users will notice if part of our VDI fails, and they may have to log in and out again either because we have patched their virtual desktops or some unplanned event has occurred, for example, in cases where a host (be that a virtualization or session host) has failed in some way.

The meaning of **high availability** is to a certain extent open to interpretation, but it usually means that there will be some outage as the standby system comes online and that this should be automatic. I will use the term *continuous availability* to indicate that there is no outage at all when a failure occurs.

Designing HA for VDI

To implement any kind of HA in a system, we need to ensure that there is no single point of failure, and, if we are extra cautious, we may even want to have three components in place so that even if we take one out for maintenance, there are still two left to provide resiliency. VDI doesn't live in isolation, so if we are implementing HA, then we must ensure that the resources needed for VDI are similarly protected from failure such as Active Directory, networking fabric, and storage. However, in this chapter, we will just concern ourselves with the different roles in VDI: the RD Web Access Portal, the RD Gateway, the RD Broker, and the servers that provide our collections of any type. We also need to understand the impact of server virtualization on both the VDI role server and our VDI collections because we have to allow for planned maintenance of our Hyper-V hosts and be able to cope if they fail unexpectedly. As we'll see, different parts of VDI make use of different techniques to enable high availability, but all of these are built into the Windows Server and are well established.

HA for the RD Broker role

HA for the **Remote Desktop Broker** (**RD Broker**) role changed in Windows Server 2012. We used to use Windows Failover Clustering and add the RD Broker as a role in a cluster, whereas it is now based around a central SQL Server database to which all the RD Brokers connect and there is no cluster. The individual RD Brokers are then put into an RD Broker Farm, which is just a set of entries in DNS, and make use of DNS Round Robin to spread out the incoming connections and provide active-active high availability. It is possible to use a third-party solution in place of DNS Round Robin, but we'll use it in our lab, and one requirement is that we must use static IP addresses for our RD Brokers (which we have already done). The SQL Server database behind the RD Broker Farm is now another point of failure, and so it also needs to be made highly available. A lengthy discussion of HA for SQL Server is not really in the scope of this book, but the following are the principal options available in SQL Server 2012, any of which will be suitable to host our broker database:

- A SQL Server cluster, where cluster nodes share an instance of SQL Server hosted on a shared storage of some kind (2 x nodes for Standard and OS for an Enterprise edition)
- Availability groups, where copies of databases are kept in sync across cluster nodes with no shared storage (Enterprise edition only)
- Database mirroring, where a copy of a database is kept up-to-date in a read-only state on another server (Standard and Enterprise editions), which can be failed over automatically by a witness server

Creating an RD Broker Farm

We won't concern ourselves here with setting up SQL Server in any kind of HA deployment as that's not the focus of this book; instead, we'll do a straight SQL Server install and store this on a single VM. I am going to suggest that you keep the number of VMs down; we'll use the RDS-Ops VM that we used for MDT in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*. The detailed instructions are as follows:

1. Download SQL Server 2012 SP1 evaluation (`http://technet.microsoft.com/en-us/evalcenter/hh225126.aspx`).
2. Go to the Hyper-V Manager on your host and start RDS-Ops VM if it's not already running (right-click on it and click on **Start**).
3. Open the console for **RDS-Ops**, and in the **Console** menu, navigate to **Media | DVD Drive | Insert Disk**.
4. Mount the ISO that contains Windows Server 2012 R2 onto the RDS-Ops VM.
5. Log in to RDS-OPs as `contoso\Administrator` (password as `Passw0rd!`).
6. Install the .NET Framework 3.5 with the following PowerShell command:

   ```
   Add-WindowsFeature NET-Framework-Core -Source D:\sources\sxs
   ```

   Here, `D:\` points to the DVD drive with the Windows Server ISO on.
7. On the **Console** menu bar, navigate to **Media | DVD Drive | Insert Disk**.
8. This time, mount the ISO that contains SQL Server 2012 SP1.
9. Install SQL Server from the command line using the following command:

   ```
   start-process "D:\Setup.exe" -ArgumentList '/Q /
   IAcceptSQLServerLicenseTerms /Action=Install /
   AgtSvcStartupType=Automatic /BrowserSvcStartupType=Disabled /
   Features=SQL,Tools /IndicateProgress /InstanceName=MSSQLSERVER
   /RsSvcStartupType=Automatic  /SQLSvcAccount="NT Authority\
   Network Service" /AgtSvcAccount="NT Authority\System"   /
   SQLSysAdminAccounts="BUILTIN\Administrators"' -NoNewWindow -Wait
   ```

10. This will do a basic unattended install of the database engine and management tools using local accounts to run the database and agent services (and assuming that the SQL Server media is on `D:`).
11. We can check if SQL Server is now installed by going to the Windows Start screen on RDS-Ops and typing `SQL` and then selecting **SQL Server Configuration Manager**. In this tool, expand **SQL Server Services** and confirm that SQL Server and the SQL Server Agent are both running.
12. We now need to ensure that we can connect to SQL Server remotely from our RD Broker by opening up a firewall rule on RDS-Ops to allow inbound access to SQL Server using either the Windows Firewall manager or the new PowerShell firewall commands in Windows Server 2012 R2, which are as follows:

```
New-NetFirewallRule -DisplayName "Allow SQLServer" -Direction
Inbound -Program "C:\Program Files\Microsoft SQL Server\MSSQL11.
MSSQLSERVER\MSSQL\Binn\sqlservr.exe" -RemoteAddress localsubnet
-Profile Domain
```

13. Now, we have a working SQL Server installation; we need to grant permission to it so that our brokers can create our broker database on it. The most efficient way to do this is to create a group in AD and add our RD Brokers to that group, which we can do on our RDS-DC from the **Active Directory Administrative Center** (**ADAC**) by right-clicking on **Contoso** and selecting **New Group**. We'll call the group `HA-Brokers` and we'll need to add in the RDS-Broker and RDS-DC computers to it using the following PowerShell commands:

```
New-ADGroup -GroupCategory:"Security" -GroupScope:"Global"
-Name:"HA-Brokers" -Path:"DC=Contoso,DC=com" -SamAccountName:"HA-
Brokers" -Server:"RDS-DC.Contoso.com"

Add-ADPrincipalGroupMembership -Identity:"CN=RDS-BROKER,CN=Compute
rs,DC=Contoso,DC=com" -MemberOf:"CN=HA-Brokers,DC=Contoso,DC=com"
-Server:"RDS-DC.Contoso.com"

Add-ADPrincipalGroupMembership -Identity:"CN=RDS-
DC,CN=Computers,DC=Contoso,DC=com" -MemberOf:"CN=HA-
Brokers,DC=Contoso,DC=com" -Server:"RDS-DC.Contoso.com"
```

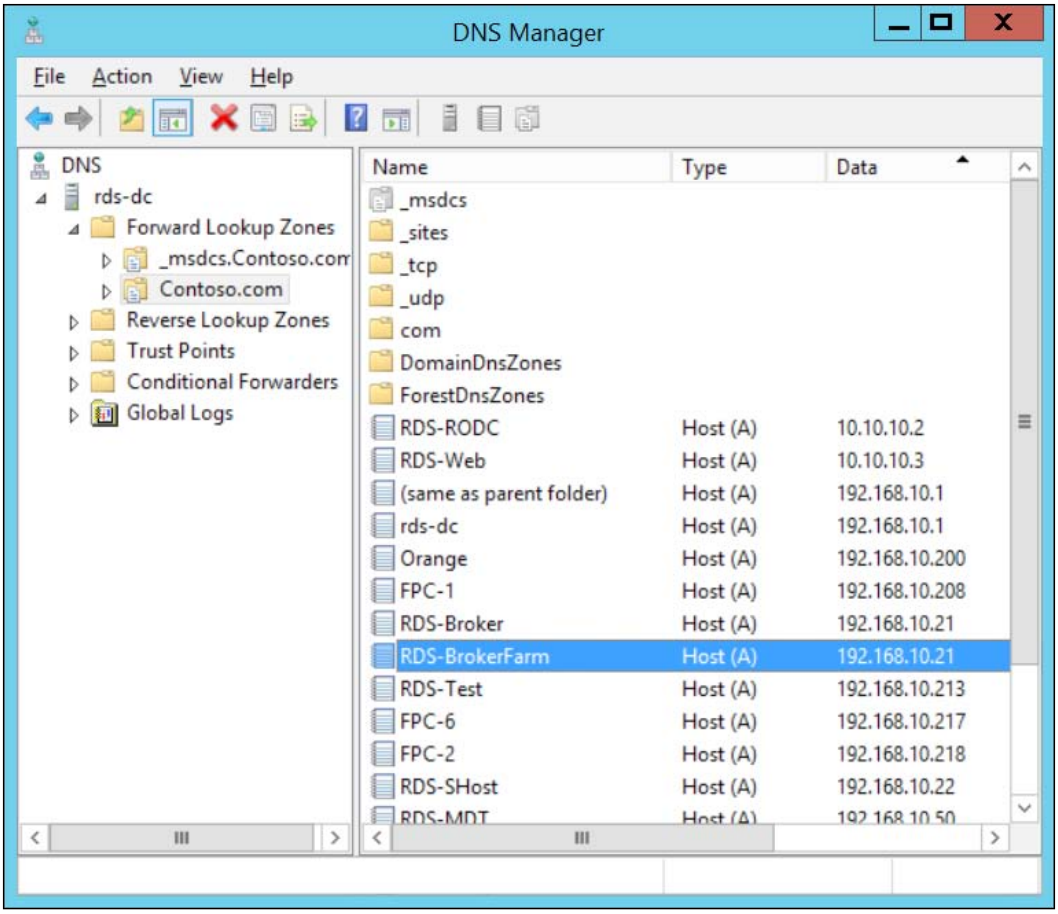
The wizard to create the SQL Server database executes from the server we are using, not the RD Broker. So in the preceding commands, the RDS-DC is also added to the HA Broker group, but it can be removed once the database is in place.

14. We can then grant permissions in SQL Server to that group. Initially, when we run the wizard, we need to create a database, so we need to grant the `dbcreator` privilege. We can do this from SQL Server Management Studio, but in a production setup, we probably won't have the privileges to do that, and instead, we would send the following PowerShell script over to the DBA team:

```
#Load up the PowerShell for SQL Server
Import-Module sqlps

#Create a SQL Server login for the group
$DBName      = "RDBroker"
$SQLInstance = "RDS-Ops"
$ADGroup     = "contoso\HA-Brokers"
$NewLogin = New-object Microsoft.SqlServer.Management.Smo.Login`
              -ArgumentList $SQLInstance, $ADGroup
$NewLogin.LoginType = "WindowsUser"
$NewLogin.Create()
#Assign the new login DB Creation rights
$NewLogin.AddToRole("dbcreator")
```

This is a very odd looking PowerShell that is using the **Windows Management Framework** (**WMF**) to do the work for us. If you type this out and use tab completion in the PowerShell ISE, you'll see that it will tab-complete all of this, so it is then just a matter of understanding the class, in this case, SQL **Server Management Objects** (**SMO**).

15. Each of our RD Broker(s) needs to connect to the same SQL Server to create and access the central broker database, and to do that, we also need to install the SQL Server native client (available at `www.microsoft.com/fwlink/?LinkID=239648&clcid=0x409`) on them by simply running `sqlncli.msi`. The final piece of the puzzle is to create a DNS entry (an "A" record) for each RD Broker in our farm with the name of the Broker Farm. We can do this from the DNS console on our RDS-DC (by navigating to **Server Manager** | **Tools**) by creating new records in **Forward Lookup Zones** | **Contoso.com**, as shown in the following screenshot:



The RDS-BrokerFarm entry has the same IP address as RDS-Broker

The following is the PowerShell command to do this:

```
Add-DnsServerResourceRecord  -Name RDS-BrokerFarm -IPv4Address
192.168.10.21 -A -ZoneName contoso.com -ComputerName RDS-DC.
contoso.com
```

For simplicity, I suggest that we disable any inbound filters and firewall rules we created on our RDS-RRAS box in *Chapter 4*, *Putting the R in Remote Desktop*, until we have our HA working properly.

We can now configure the broker for high availability in the **RDS Overview** diagram using the following steps:

1. Connect to the RDS-DC VM as `contoso\Administrator` (password: `Passw0rd!`).
2. Expand **Remote Desktop Service** and navigate to the **Remote Desktop Services Overview** screen.
3. Right-click on the **RD Connection Broker** icon in the overview diagram and select **Configure High Availability**.
4. Note the information in the **Before you begin** screen and click on **Next**.
5. Set the database connection string as `DRIVER=SQL Server Native Client 11.0;SERVER=RDS-Ops; Trusted_Connection=Yes;APP=Remote Desktop Services Connection Broker;DATABASE=RDBroker` and the value of **Database Location** as `c:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA`, which is the default location for SQL Server databases. We also need to enter the **DNS Resource Record Name**, which corresponds to the DNS entries we just created for the RD Broker Farm we just created, `RDS-BrokerFarm.contoso.com`.

If we have implemented some sort of HA solution for SQL Server and we want to use that for the RD Broker database, then the connection string needs to reflect the solution we have used.

The following command is for the SQL Server failover cluster:

```
DRIVER=SQL Server Native Client 11.0;SERVER=<cluster>;
Trusted_Connection=Yes;APP=Remote Desktop Services Connection
Broker;DATABASE=<broker_database>;
```

The following command is for SQL Server Availability Group:

```
DRIVER=SQL Server Native Client 11.0;SERVER= <availability_group_liste
ner>;MultiSubnetFailover=True;Trusted_Connection=Yes;APP=Remote Desktop
Services Connection Broker;DATABASE=<broker_database>;
```

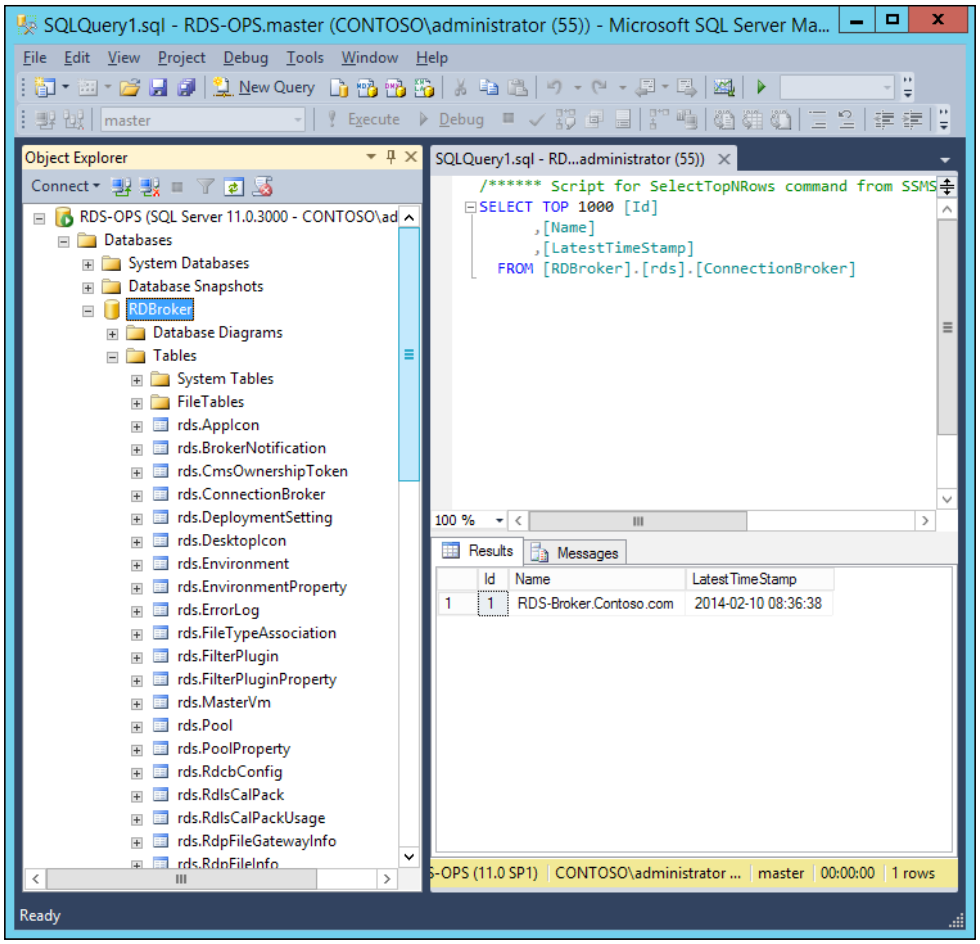
The following command is for SQL Server mirroring:

```
DRIVER=SQL Server Native Client 11.0;SERVER=<sql_server_
instance1>;Failover_Partner=<sql_server_instance2>;Trusted_
Connection=Yes;APP=Remote Desktop Services Connection
Broker;DATABASE=<broker_database>;
```

There's just one line of PowerShell to set up HA for the broker once we have declared the necessary variables. This is as follows:

```
$RDBroker = "RDS-Broker.contoso.com"

$DBConnection = "DRIVER=SQL Server Native Client 11.0;SERVER=RDS-
Ops;Trusted_Connection=Yes;APP=Remote Desktop Services Connection
Broker;Database=RDBroker"

$DBPath = "c:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\
MSSQL\DATA\RDBroker.mdf"

Set-RDConnectionBrokerHighAvailability -ConnectionBroker $RDBroker
-DatabaseFilePath $DBPath -ClientAccessName "RDS-BrokerFarm.contoso.com"
-DatabaseConnectionString $DBConnection

<#Notice that when using PowerShell we specify the path to the actual
database file we'll create not just the folder#>.
```

If we go back to the **RDS Overview** diagram, we'll notice that the RD Broker icon has now changed to reflect that it is now configured for high availability. More importantly, the Web Access server has now been configured to reference the Broker Farm that we created in DNS rather than the single RD Broker we had before, and we now have a database on RDS-Ops, as we can see by going into **SQL Server Management Studio** (**SSMS**) on RDS-Ops.



The RD Broker database now has all of the information needed by the broker including the brokers that are present

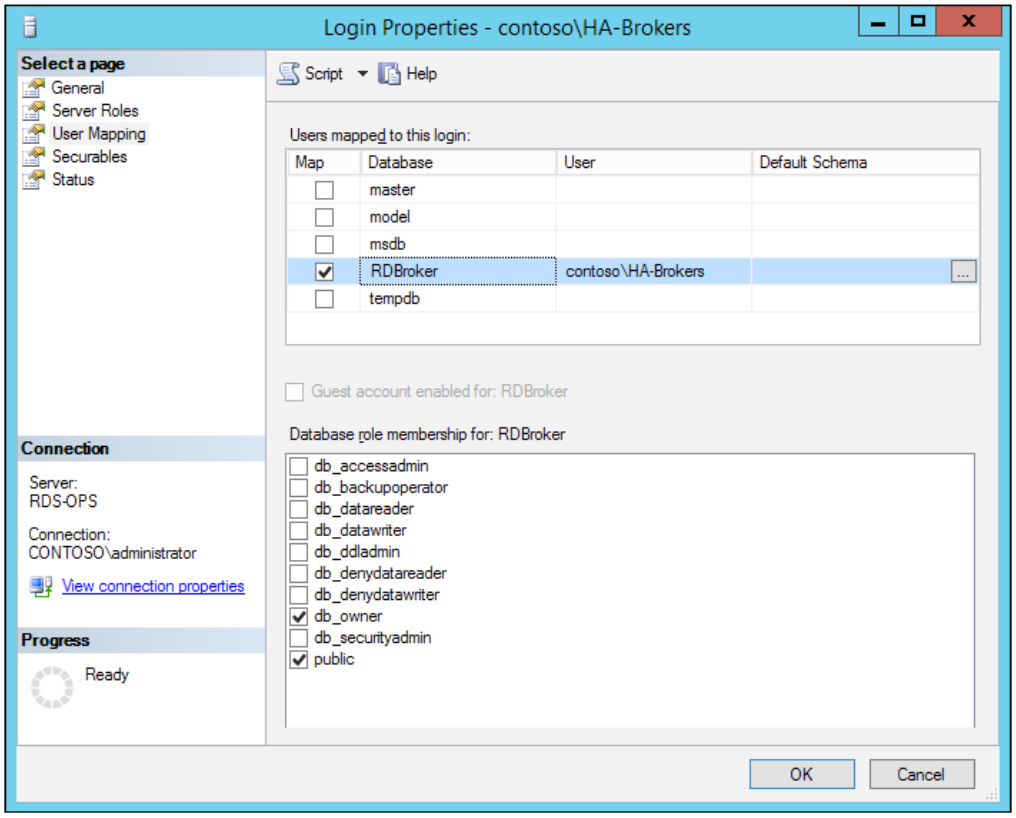
The various tables in this database give us an insight into what the broker does, but we should not edit these directly from here. Now, since we have this database, we don't need our AD Group of Broker (HA-Brokers) to have **dbcreator** rights to SQL Server anymore; all we need to do is grant database ownership to the actual broker database (RD Broker).

We can do this in SSMS by navigating to **RDS-Ops** | **Security** | **Logins** | **contoso\HABrokers**, selecting **server roles** from properties, and unchecking **dbcreator**. The PowerShell equivalent is as follows:

```
Import-Module sqlps
cd SQLServer:\SQL\RDS-Ops\default\roles
$dbcreator = Get-ChildItem | where name -eq dbcreator
($dbcreator).dropmember("contoso\HA-Brokers")
```

Notice here that we can traverse the SQL Server hierarchy just as we would in a folder, and the same applies to the registry; for example, `cd HKLM:` takes us to the local machine hive.

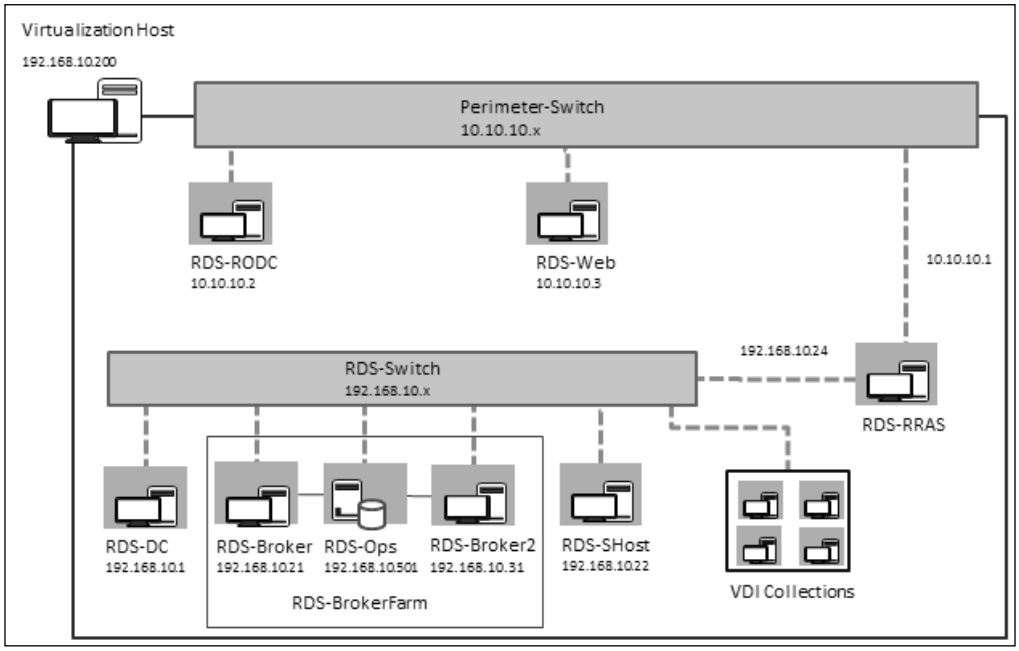
Instead of those permissions, we can just grant the database ownership (db_owner) privilege on the database we have just created by navigating to the properties of the login again and selecting User Mapping and checking the options as shown in the following screenshot:



Again, we can use the following PowerShell script for this:

```
#Load up the PowerShell for SQL Server
Import-Module sqlps
$DBName    = "RDBroker"
$SQLInstance = "RDS-Ops"
$ADGroup   = "contoso\HA-Brokers"
$SQLserver = new-object Microsoft.SqlServer.Management.Smo.Server`
-ArgumentList $SQLInstance
$DB        = $SQLserver.Databases[$DBname]
$SQLlogin  = $SQLserver.Logins[$ADGroup]
#this creates a new user in the RDBrokerdatabase
$DBUser   = New-Object Microsoft.SqlServer.Management.Smo.User`
-ArgumentList $DB, $ADGroup
$DBUser.Login = $ADGroup
$DBUser.Create()
$RoleName = "db_owner"
$DBrole = $DB.Roles[$RoleName]
$DBrole.AddMember($ADGroup)
```

Now that we have got HA for the RD Broker role properly configured for one server, we can quickly add in more servers to actually make this role highly available and to check that it works. The following design is what we are aiming for:

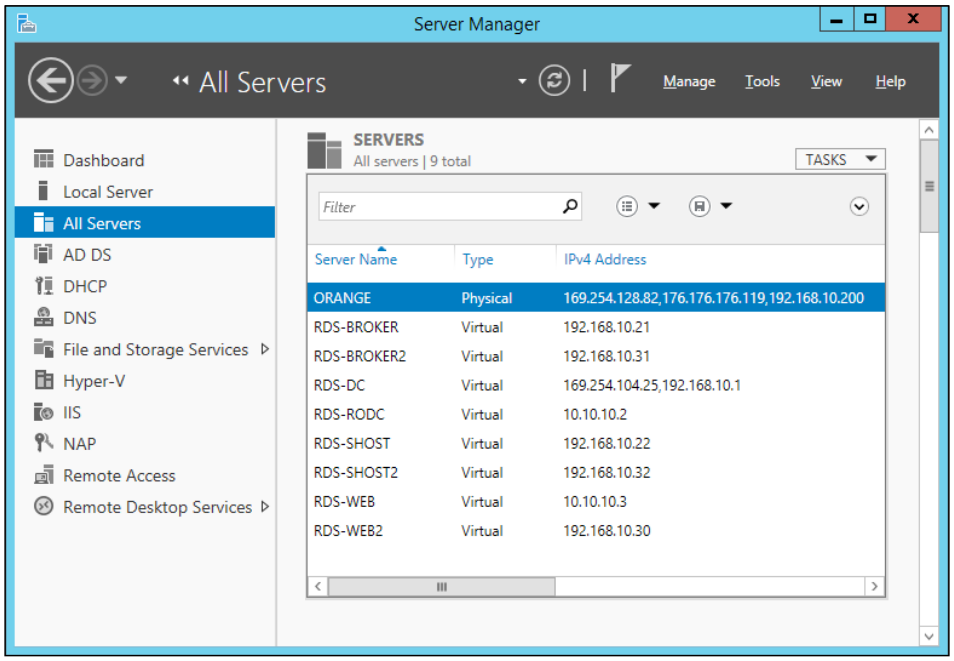


Our new design that shows the RDS Broker Farm with our SQL Server database on RDS-Ops

We are going to add in another RD Broker (`RDS_Broker2`). To do that, we are going to need another VM, and as we will also need VMs to set up HA for the Web Access / Gateway role and another RD Session Host, I suggest that we edit the PowerShell script in *Chapter 2*, *Designing a Virtual Desktop Infrastructure*, which we used to create our VDI VMs and create three new ones:

Line No	Value to change	To be replaced with
19	`$RDWebVM = "RDS-Web"`	`$RDWebVM = "RDS-Web2"`
20	`$RDBrokerVM = "RDS-Broker"`	`$RDBrokerVM = "RDS-Broker2"`
22	`$RDSHostVM ="RDS-SHost"`	`$RDSHostVM ="RDS-SHost2"`
276	`-IPAddr "192.168.10.20"`	`-IPAddr "10.10.10.4"`
276	`-Network $VirtualSwitch`	`-Network "Perimeter-Switch"`
276	`-DNSSvr "192.168.10.1"`	`-DNSSvr "10.10.10.2"`
277	`-IPAddr "192.168.10.21"`	`-IPAddr "192.168.10.31"`
287	`-IPAddr "192.168.10.22"`	`-IPAddr "1092.168.10.32"`

Once these VMs have been created, we need to bring them under management in Server Manager on our RDS-DC, as shown:



A Server Manager on RDS-DC with all the HA VMs we'll need

Now, we can use **RDS-Broker2** as another RD Broker in our Broker Farm by performing the same steps as we did for the first broker:

1. Install the SQL Server native client on **RDS-Broker2**.
2. Add **RDS_Broker2** into the HA-Brokers group in AD so that it will have access to our SQL Server broker database, as shown:

```
Add-ADPrincipalGroupMembership -Identity:"CN=RDS-Broker2,CN=Comput
ers,DC=Contoso,DC=com" -MemberOf:"CN=HA-Brokers,DC=Contoso,DC=com"
-Server:"RDS-DC.Contoso.com"
```

3. Create another DNS record for the farm with the IP address of **RDS-Broker2**, as shown:

```
Add-DnsServerResourceRecord  -Name RDS-BrokerFarm -IPv4Address
192.168.10.31 -A -ZoneName contoso.com -ComputerName RDS-DC.
contoso.com
```

Now, we can make this VM a part of our RD Broker Farm with the following steps:

1. Open **Server Manager**, and in the **RDS Overview** diagram, right-click on the **RD Connection Broker** icon and select **Add RD Connection Broker Server**.
2. Click on **Next**, and in the **Select a Server** screen, double-click on **RDS-Broker2.contoso.com** to add it to the selected server list, and click on **Next**.
3. Click on **Add** to add the server to the RD Broker Farm.

The equivalent PowerShell is just one line and is as follows:

```
Add-RDserver -Role RDS-CONNECTION-BROKER -Server RDS-Broker2.contoso.com
-ConnectionBroker "RDS-broker.contoso.com"
```

In *Chapter 4*, *Putting the R in Remote Desktop*, we enabled the RD Gateway and built a perimeter network. So, we must check that our certificates are ok and set the gateway on the new RD Broker to point to our RRAS server. Perform the following steps to do so:

1. From the **RDS Overview** diagram, select **Task | Edit Deployment Properties** and navigate to **Certificates**. You should see that the entries for the two **RD Connection Broker** settings have a status of **Error**. Highlight **RD Connection Broker –Enable Single Sign On** and click on **Select existing certificate**.
2. In the **Select Existing Certificate** screen, select **Apply the certificate that is stored on the RD Connection Broker Server** and enter the password (`Passw0rd!`). Check the option **Allow the trusted certificate to be added to the Trusted Root Authorities certificate store on the destination computers** and click on **OK**. Click on **Apply** on the **Deployment properties** screen.

3. Repeat step 2 for the **RD Connection Broker – Publishing Role Service**.
4. Connect to **RDS-Broker2** and set its gateway by going to **Server Manager** and navigating to a local server. Click on its IP address (`192.1768.10.31`) to bring up its **Network Connections**. Right-click on the Ethernet connection and go to the properties of the IPv4 protocol. Set the value of **Gateway** to `192.168.10.24` (the RDS-RRAS server) and click on **OK** and on **OK** again to close the network connection.

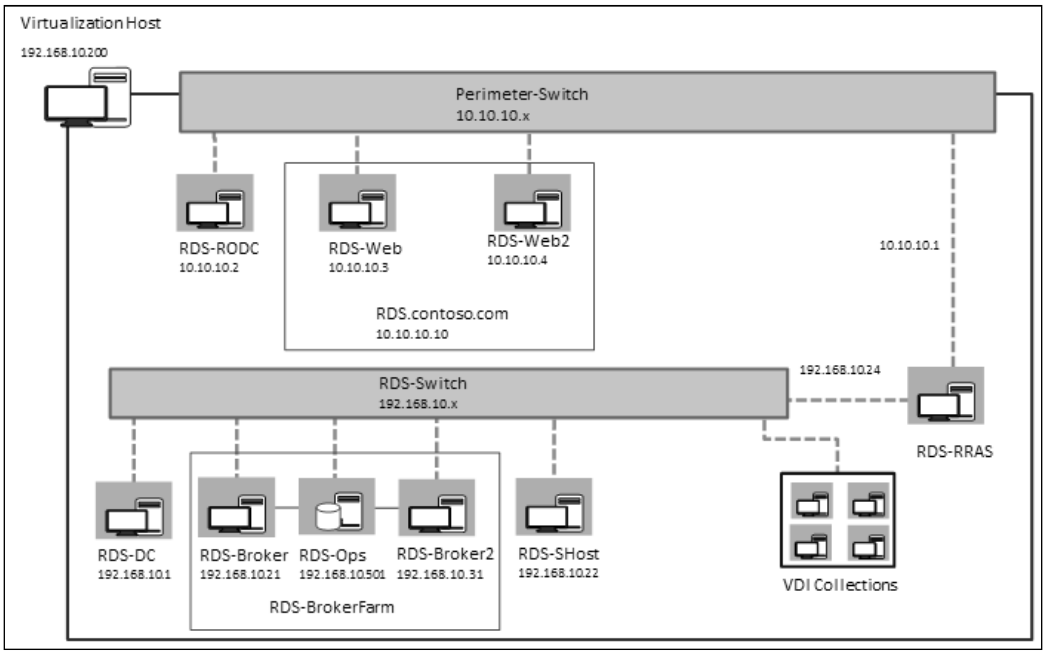
Now that we have two RD Brokers in place, we can retest if our collections are working. We can see that when we connect to a virtual desktop, the remote desktop title bar reflects the name of our RD Broker Farm (`RDS-BrokerFarm.contoso.com`). We can also test failover by pausing one of the RD Brokers, which we can do from **Hyper-V Manager** by right-clicking on **RDS-Broker** and selecting **Pause**. We can then check if we can still connect to our VDI deployment and log in to a virtual desktop. We can also try logging into a virtual desktop and then stop one of the brokers. In this case, while we may lose the connection momentarily, we will automatically be connected back into the same session we were using and continue to work from where we left off.

HA for the RD Web Access and RD Gateway roles

We can enable HA for Web Access and Gateway roles by using **Network Load Balancing** (**NLB**), and it's possible to use the built-in NLB server role in Windows Server 2012 R2 to do this or use a third-party appliance. In the spirit of this book, we'll look at how to do this with Windows Server because it's good to see what's possible in the box before looking at the extra expense of a third-party tool. In order to set up NLB, each participating server has to have a fixed IP address on one of its NICs, which is what we have already.

In older versions of Windows Server, we needed to have a dedicated NIC for NLB; however, in Windows Server 2012 R2, there is now NIC teaming, the idea being we can use the entire capacity of all of our NICs and then isolate traffic that go over the same network using VLANs, subnets, and so on.

Our VDI design will evolve again to achieve this by putting another server (**RDS-Web2**) into our perimeter network, which will have the RD Web Access and RD Gateway roles to form a two-node NLB cluster for these roles.



RDS-Web and RDS-Web 2 with NLB

Setting up NLB

The first thing we need to do is install the NLB role on our Web Access / Gateway Server (RDS-Web). Perform the following steps to do so:

1. Connect to RDS-Web, and from the **Server Manager** menu, select **Manage | Add Roles and Features**.
2. In the **Select installation type** screen, select **Role-based** or **Feature-based** installation and click on **Next**.
3. For **Select destination Server**, select **RDS-Web** and click on **Next**.
4. In **Select Roles**, click on **Next**.
5. In **Select Features**, select **Network Load Balancing**, and in the popup that appears, include the management tools and click on **Add Features**. Click on **Next**.
6. In the **Confirmation** screen, click on **Install**.

In PowerShell, we can do this in one line, which is as follows:

```
Add-WindowsFeature -Name NLB -IncludeManagementTools
```

We can now go back to **Server Manager** and configure the NLB cluster by navigating to **Tools | Networking Load Balancing Manager** and performing the following steps:

1. In the **Network Load Balancing Manager** screen, right-click on **Network Load Balancing Cluster** and select **New Cluster**.
2. In the **New Cluster Connect** screen, enter `RDS-Web` as the value of **Host**. The interfaces on this server will then be populated. Click on **Next**.
3. In the **Host Parameters** screen, the **Priority** indicates a unique ID for this node, and we'll leave it at **1**. We can also see the fixed IP address (`10.10.10.3`) that will identify this host on the cluster. We'll leave the default start state as **Started** and click on **Next**.
4. For **Cluster IP Addresses**, we will set up an IP address that will be used for the whole cluster. We can add multiple addresses in order to serve different websites, but for our needs, we will just enter `10.10.10.10` with a subnet mask of `255.255.255.0` and click on **Next** and on **Next** again.
5. In the next **Cluster Parameters** screen, we can enter an Internet name for the cluster. We'll enter `VDI`, but this actually doesn't do anything. What we must do is set the value of **Cluster Operation** to **Multicast** so that we can discover it across our RRAS server.
6. We can set port rules for load balancing in the same way that we did in *Chapter 4*, *Putting the R in Remote Desktop*, for routing and remote access. For now, we can leave this blank and click on **OK** to set up the cluster.

The PowerShell for this is as follows:

```
New-NlbCluster `
         -ClusterName VDI`
         -HostName RDS-Web`
         -InterfaceName Ethernet
         -ClusterPrimaryIP 10.10.10.10`
         -SubnetMask 255.255.255.0 `
         -OperationMode Multicast
```

We can now ping our NLB cluster, but it can't really do anything because it doesn't appear in DNS, although it does have an IP address `10.10.10.10`. Back in *Chapter 4*, *Putting the R in Remote Desktop*, we added in a special (CName) DNS record to direct users to our RD Gateway, and this corresponds to the name in our certificate that signs the Web Access portal and Gateway. So what we need to do is delete that record and create a conventional host or a record for our NLB cluster that points to its IP address (`10.10.10.10`). Perform the following steps to do so:

1. Connect to the **RDS-DC** VM, and from the **Tools** menu in **Server Manager**, select **DNS** to open **DNS Manager**.
2. Expand **rds-dc** | **Forward Lookup Zones**.
3. Right-click on **Contoso.com** and navigate to the **rds** CName record and delete it.
4. Right-click on **Contoso.com** and select **New Host (A or AAAA)**.
5. Set the name to RDS and leave and set the IP address to `10.10.10.10`. Click on **OK** to create the entry.

The PowerShell command to do this is as follows:

```
Remove-DnsServerResourceRecord -Name RDS -RRType CName -ZoneName contoso.
com -ComputerName RDS-DC.contoso.com
```

```
Add-DnsServerResourceRecordA -Name RDS -IPv4Address 10.10.10.10 -ZoneName
contoso.com -ComputerName RDS-DC.contoso.com
```

Although our NLB cluster is working, it only has one node in it, and so it's not really balancing anything nor is it highly available. So let's add in another web server, RDS-Web2. As with the first server, we need to add in the NLB feature, which we can do using the following PowerShell command from our RDS-DC VM:

```
Add-WindowsFeature -Name NLB –ComputerName "RDS-Web2"
```

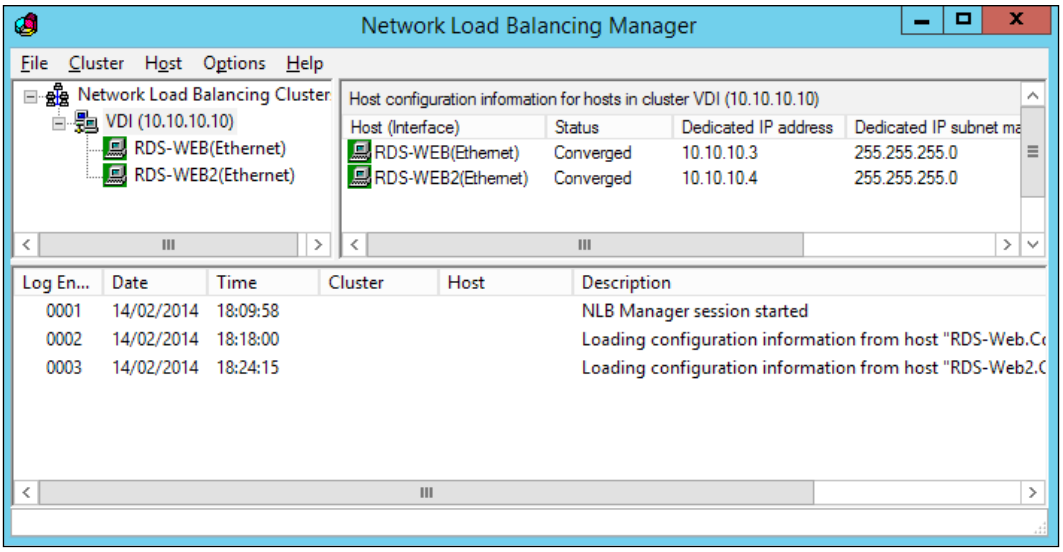
We don't need the management tools, but we do need to specify the machine to install the feature on. Because we have the remote administration tools on RDS-DC, we can also add the node into the cluster from here by navigating to **Server Manager Tools** | **Networking Load Balancing Manager** and performing the following steps:

1. Right-click on **Network Load Balancing Cluster** and select **Connect to Existing**.
2. In the **Connect RDS-Host** screen, enter **RDS-Web** as the value of **Host**. The interfaces on this server will then be populated. Click on **Finish**.
3. Right-click on **VDI(10.10.10.10)** and select **Add Host to Cluster**.
4. In the **Connect** screen, enter **RDS-Web2**, click on **Connect**, and we can see the **Ethernet** interface with an address `10.10.10.4`. Click on **Next**.
5. In the **Host Parameters** screen, leave all the defaults as they are and click on **Next**.
6. In the **Port Rules** screen, we'll leave the rules as they are to simplify things and click on **Finish** to add the node to the cluster.

The PowerShell command for this is as follows:

```
Add-NlbClusterNode -HostName "RDS-Web" -InterfaceName Ethernet
-NewNodeName "RDS-Web2" -NewNodeInterface Ethernet
```

We can now see our NLB cluster with the two nodes, as shown in the following screenshot:



Our NLB cluster is now able to balance traffic; however, the new RDS-Web2 server is not part of our RDS deployment. So we need to go into **Remote Desktop Services Overview** on the RDS-DC diagram and add it in, using the following steps:

1. Right-click on the **RD Web Access** icon and select **Add RD Web Access Servers**.
2. In the **Select a server** screen, double-click on **RDS-Web2** to select it and click on **Next**.
3. In the **Confirm selections** screen, click on **Add** to add the server into the deployment.

You can run the following PowerShell command instead:

```
Add-RDServer -Server "RDSWeb2.contoso.com" -Role RDS-WEB-ACCESS
-ConnectionBroker RDS-Broker.contoso.com
```

When I was testing the PowerShell scripts in this chapter, I found that because I had suspended RDS-Broker, RDS-Broker2 had become the active management server, and this meant that I had to use this server wherever I had to set the active management server either by navigating to **Deployment Overview** | **Tasks** | **Set Active RD Connection Broker** or with the following PowerShell command:

```
Set-RDActiveManagementServer -ManagementServer RDs-
broker.contoso.com
```

Now, we need to repeat the process to add this server in as a gateway:

1. Right-click on the RD Gateway icon and select **Add RD Gateway Servers**.
2. In the **Select a server screen**, double-click on **RDS-Web2** to select it and click on **Next**.
3. In the **Confirm selections** screen, click on **Add** to add the server into the deployment.

The following `Add-RDSServer` PowerShell command can be used for this as before:

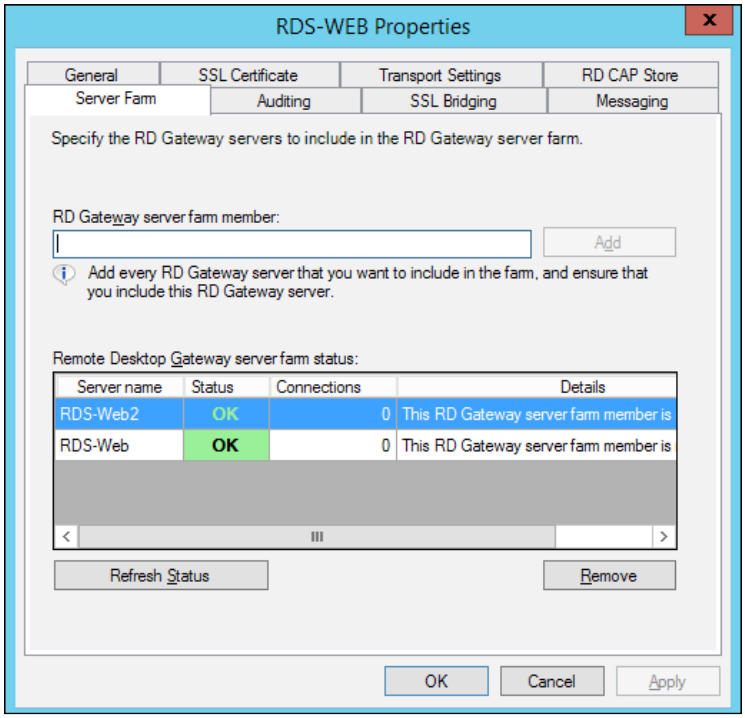
```
Add-RDServer -Server RDS-Web2.contoso.com" -Role RDS-GATEWAY
-GatewayExternalFqdn "RDS.CONTOSO.COM" -ConnectionBroker "RDS-Broker.
contoso.com"
```

However, we also need to provide the FQDN of the Gateway while adding a new RD Gateway server. Now, we need to install the certificate we have been using to secure the RD Gateway onto this new server using the following steps:

1. From the **RDS Overview** diagram, navigate to **Task** | **Edit Deployment Properties** and navigate to **Certificates**. You should see that the entries for the **RD Web Access** and **RD Gateway** roles have a status of **Error**. Highlight **RD Web Access** and click on **Select Existing Certificate**.
2. In the **Select Existing Certificate** screen, select **Apply the certificate that is stored on the RD Connection Broker Server** and enter the password (`Passw0rd!`). Check the option to allow the trusted certificate to be added to the Trusted Root Authorities certificate store on the destination computers and click on **OK**. Click on **Apply** on the **Deployment properties** screen.
3. Repeat steps 1 and 2 for the RD Gateway role.

The management of RD Gateway is one of the few parts of RDS that still has its own console, the RD Gateway Manager, and we need to use this to configure the Gateway Farm using the following steps:

1. Open the **Remote Desktop Gateway Manager** (search for it from the start menu on RDS-DC).
2. Right-click on the **RD Gateway Manager** icon and select **Connect to RD Gateway Server**. Select **Remote Server** and enter **RDS-Web**.
3. Right-click on **RDS-Web** and select **Properties**.
4. In the **Server Farm** tab, add in **RDS-Web** and **RDS-Web2**. They should both have a status of **OK**, as shown:



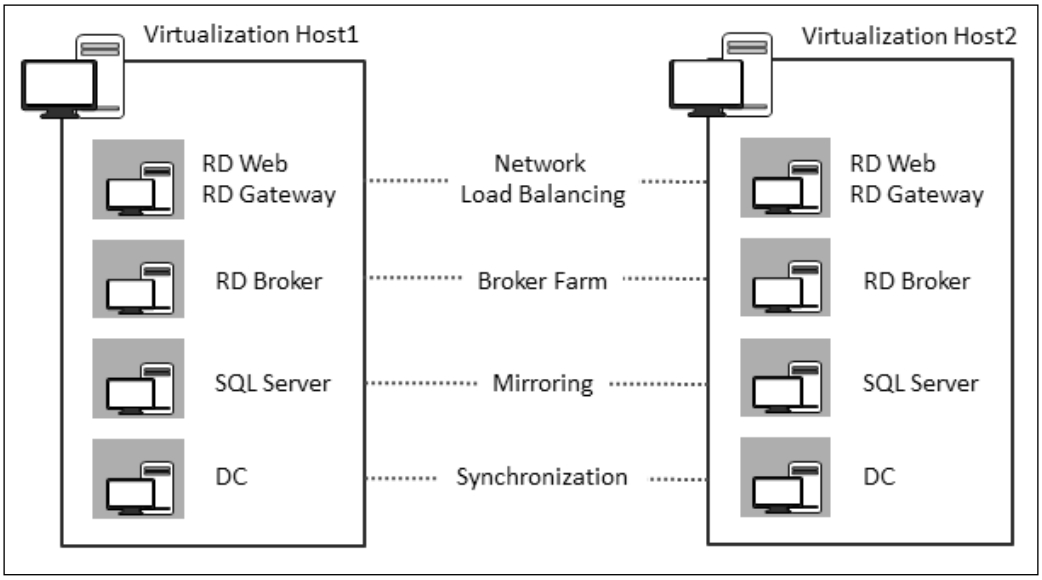
5. The only other thing we need to do on our new Web Access server is to set its gateway to `10.10.10.1` (the RRAS server).
6. Connect to **RDS-Web2** and set its gateway by going to **Server Manager** and navigating to the local server. Click on its IP address (`10.10.10.4`) to bring up its **Network Connections**. Right-click on the Ethernet connection and go to the properties of the IPv4 protocol. Set the **Gateway** to `10.10.10.1` (the RDS-RRAS server) and click on **OK** and again on **OK** to close the network connection.

We should now test if we have actually got high availability working for our RD Gateway by pausing the RDS-Web VM in Hyper-V manager and connecting to `https: RDS.contoso.com/RDWeb` to confirm if we can still connect to our collections. We can also see that if we are already connected to a virtual desktop and either one of the gateways is paused, we can continue to work as if nothing has happened. Hence, we can now carry out panned maintenance on any of our RDS role servers without affecting our users, and our users are also protected from failure of any one of these servers.

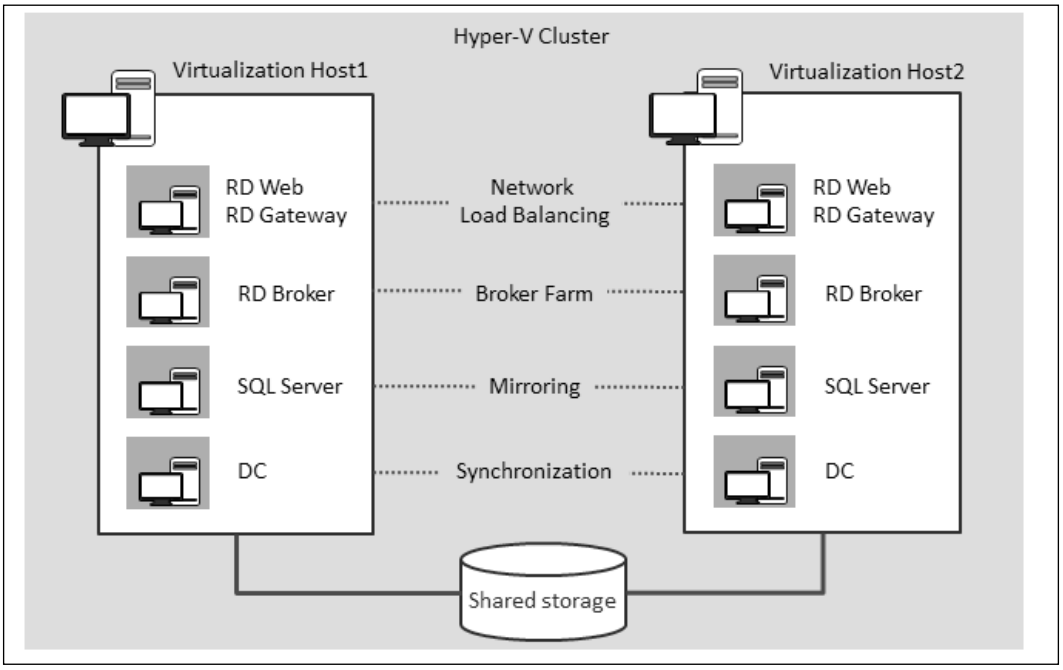
HA and Hyper-V

So far in this chapter, we have considered each of the RDS server roles as though they are individual physical servers; however, we have all of these roles running as Hyper-V VMs, and there is no reason why we wouldn't virtualize them in a production deployment. However, this means that we have another point of failure—the physical host on which these VMs are running. So what can be done to ensure we can either take a physical host offline to maintain it or survive an unexpected failure?

The simplest approach is to have a second host and put one set of our servers on each host, as illustrated by the following diagram:



Then, if either host fails, we have a complete set of servers to continue to support our RDS deployment. We can further enhance this by putting our physical hosts into a Windows cluster and enabling HA for the VMs on that cluster by storing them on a shared storage that is accessible to all nodes in the cluster. This is illustrated in the following figure:



This means that we have the option to drain a host of its VMs for planned maintenance with Hyper-V Live Migration; hence, the VMs don't stop in the process. If a host fails, the VMs that were running on it will automatically start on the surviving nodes. This isn't easy to do in our lab setup, as we would need a second physical host joined to our domain and both hosts would need access to a shared storage. If we already have Hyper-V in production and have implemented HA for it, then these servers are easily added to that.

I have made videos of how to set/create an HA VM in a lab setup with just two laptops. If you do want to try this out, check out my blog at `http://blogs.technet.com/b/andrew/archive/2013/01/14/evaluate-this-high-availability-virtual-machines.aspx`.

HA for virtual desktop collections

So far in this chapter, we have managed to ensure that when our RDS management servers fail, our users are hardly affected—they can still connect and get a new virtual desktop, and if they are logged in, they won't lose any work and only suffer a minor interruption. However, the collections we have created are running on individual hosts, and if these fail, our users won't be able to connect, and if they are already connected, they will lose their session. So what can be done to make our collections more resilient? The answer depends on the type of collection, so let's look at each of these in turn.

HA for session collections

We can easily add in more session host servers and use these to load balance and scale our collections; however, this will only provide a minimal HA capability, as all this will do is allow our collection to be available when users need a new session-based virtual desktop. What this doesn't do is to allow our users' sessions to survive the loss of the session host that's running their particular virtual desktop. This is because our users' sessions just exist in memory, so they are not preserved when the server is lost. When we do add in a new session host along with adding it into our deployment, we also need to specify that it is being used to host a particular collection.

One key thing to note is that each session host in a given collection must be configured in exactly the same way; so each server must have the same version of the OS with the same patches. The applications, folder layout, and so on must also be identical. This is because we don't have fine-grain control over which user ends up using which session host to provide their virtual desktop and they will want a consistent experience across these hosts.

We can quickly set up a second session host by using the last of the new VMs we created earlier in this chapter (RDS-SHost2), using the following steps:

1. On **RDS-DC**, connect to **Server Manager** and bring up the **Deployment Overview** in **Remote Desktop Services**.
2. Right-click on the **RD SessionHost** icon and select **Add RD Session Host servers**.
3. In the **Select a server** screen, double-click on **RDS-SHost2** to select it and click on **Next**.
4. In the **Confirm selections** screen, click on **Add** to add the server into the deployment.

The `Add-RDServer` command in PowerShell does the same thing, as illustrated in the following line of command:

```
Add-RDServer -Server RDShost2.contoso.com -Role RDS-RD-SERVER
 -ConnectionBroker RDS-Broker.contoso.com
```

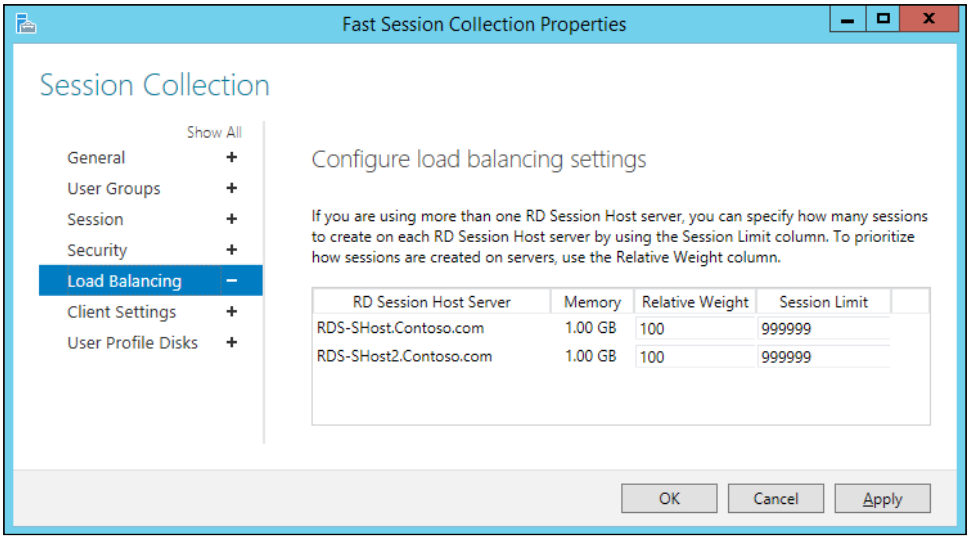
We can then allocate this server to the session collection that we already have, using the following steps:

1. Still in **Server Manager** on RDS-DC, navigate to **Remote Desktop Services | Collections | Fast Session Collection** to display all the properties and settings of our collection.
2. In the **Host Servers** pane, navigate to **Tasks | Add RD Session Host servers**.
3. In the **Select a server** screen, double-click on **RDS-SHost2** to select it and click on **Next**.
4. In the **Confirm selections** screen, click on **Add** to add the server into the deployment. Click on **Close** to complete the process when the server is added.

The equivalent PowerShell command for this is as follows:

```
Add-RDSessionHost -SessionHost RDS-Shost2.contoso.com -ConnectionBroker
RDS-Broker.contoso.com -CollectionName "fast session collection"
```

In Windows Server 2012 R2, we can quickly set up load balancing across these two session hosts by going to the **Properties** pane for the collection and selecting **Edit Properties** from **Tasks**. The following screenshot shows the output of the mentioned steps:



Now, if we want to do a planned maintenance on a session host, we could deny new connections to the given host from here and wait until none of our users are on it before taking it offline to work on it. If a session host fails unexpectedly, our users will lose their session, but the situation is not too disastrous if we have their data and the user profile disk on a shared storage of some kind—they will be able to log in to a new session and get back to their last saved copy of any work they were doing very quickly.

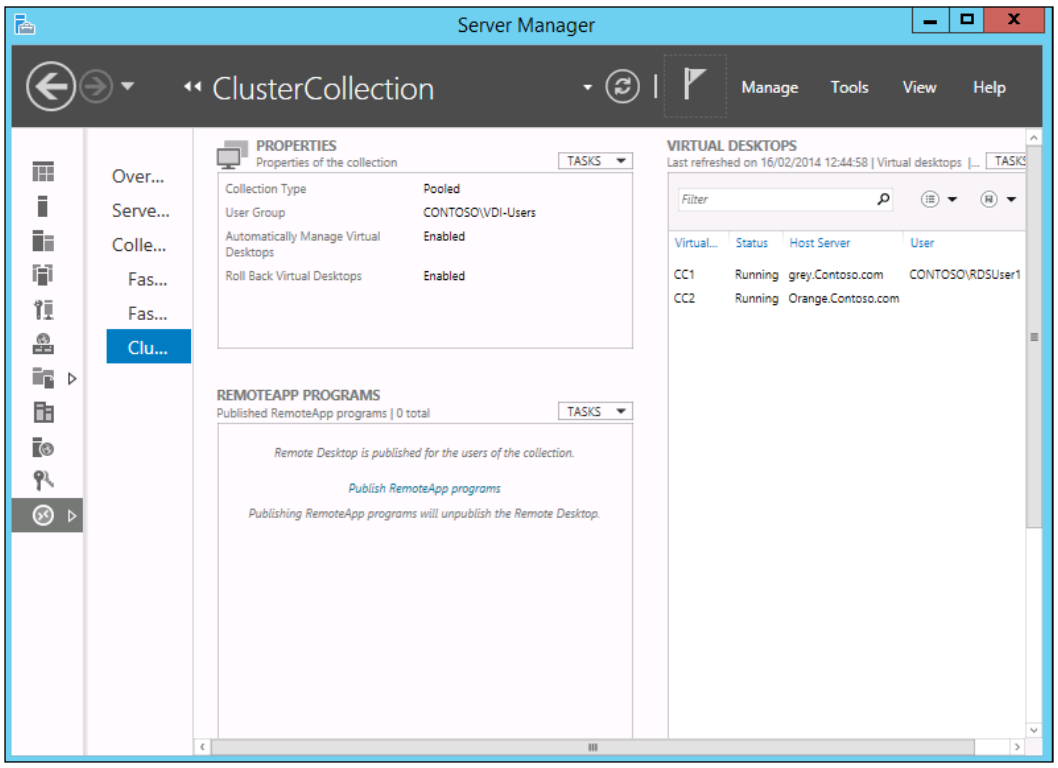
HA for VDI collections

We also need additional hosts if we want to provide any kind of redundancy for VDI collections of whatever type, but these will be virtualization hosts (physical servers that run Hyper-V). If a host actually fails, the VMs on it will restart after the host comes back online. We can also manually move VMs from one virtualization host to another with the `move-RDVirtualDesktop` PowerShell command so that we can take a virtualization offline for planned maintenance.

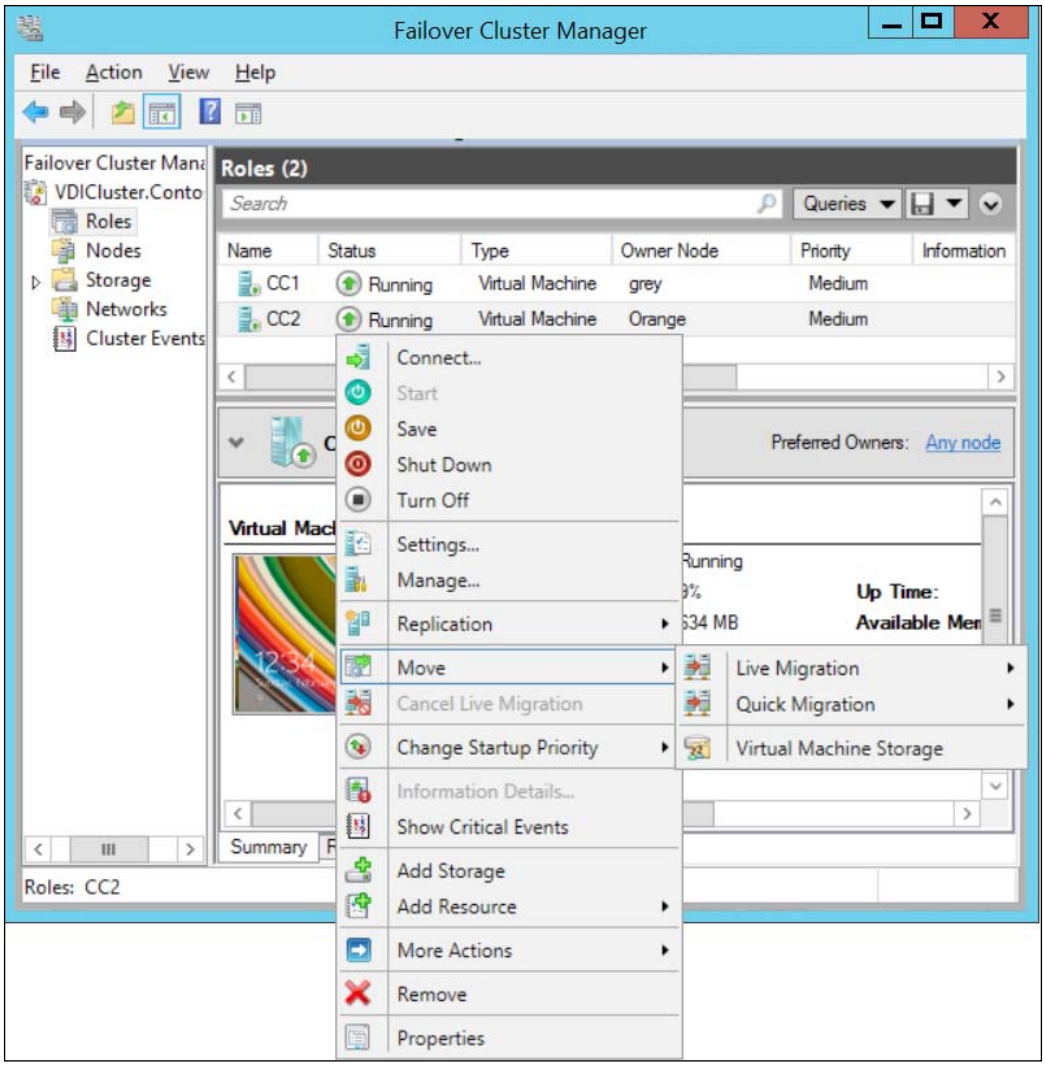
`move-RDVirtualDesktop` allows us to move virtual desktops between hosts even if they aren't in a cluster; however, to do that, the two hosts need to trust each other, and the simplest way to do that is to set up **Credential Security Support Provider Protocol** (**CredSSP**) between them. If we want to move a VM FPC-01 from server 1 to server 2, then we should run the following PowerShell commands:

```
Enable-WSManCredSSP -Role "Client" -DelegateComputer *
-Force
Enable-WSManCredSSP -Role "Server" -DelegateComputer *
-Force
$Creds = Get-Credential #which will prompt for credentials
to use
Move-RDVirtualDesktop -SourceHost "Server1.contoso.com"
-DestinationHost "Server2.contoso.com" -Name "FPC-01"
-Credential $Creds
```

We can provide better and faster resilience for our VDI collections by creating them on shared storage rather than local storage as we did in *Chapter 2*, *Designing a Virtual Desktop Infrastructure*. We can then combine our hosts into a Hyper-V cluster that allows us to automatically patch nodes on the cluster and perform very fast live migrations of our virtual desktops to move them from node (virtualization host) to node. In the following screenshot, I have created a new pooled VDI collection of two virtual desktops; each of these resides on a different virtualization host (gray and orange).



However, RDS isn't aware that these hosts are in a cluster and we can't add a whole cluster as a host. So, if we use a cluster, then it's best to use the tools in the cluster (either via PowerShell or in Cluster Manager), as shown in the following screenshot:



We can see that the two VMs, CC1 and CC2, are in the cluster and that I can easily move them with Live Migration from one host to another. If I do this, the user won't be aware of it and neither will it show up in the RDS overview in **Server Manager** until we refresh it. If I break the connection or stop one of these servers (for example, gray), then a connected user (RDSUser1) will lose their session and any unsaved work. However, they will be able to connect straight back in, and their profile and saved work will be there, so for them, it's exactly like their laptop crashed in the real world.

Summary

In this chapter, we have seen that Windows Server 2012 R2 has a range of built-in tools that can be applied to different roles in our VDI deployment to minimize the impact of any server failure. Just as importantly, we can take parts of the infrastructure offline for maintenance without affecting our users, and we will cover more of that in *Chapter 7*, *Maintenance and Monitoring*. In older versions of Windows, the price of HA was that we had a redundant server that was largely idle until a failure occurred. However, the HA that we implemented here not only provides us with HA, it also allows us to scale up our deployment. In the next chapter, we'll see how to get the best VDI performance from what we have, how to test and plan for larger deployments, and what the limits are in Windows Server 2012 R2.

6
Scale and Performance

Now that we have made our VDI deployment as reliable as possible, we can apply some of the same techniques for scaling it up to provide virtual desktops for more and more users. In this chapter, we will review each role and service in a VDI deployment to see how to tune and scale it up. This will include a look at the impact of virtualizing the roles as well as our virtual desktops. We will look at what RemoteFX is and how it balances rich user experiences such as video streaming, device redirection, and multitouch over limited bandwidth connections. We will also revisit the Virtual Desktop Template and look at how its settings and those of Windows 8.1 can be optimized.

Understanding scale and performance

In the previous chapter, we made our VDI deployment highly available, but if our virtual desktops don't perform well or it takes a long time to connect, our users will think it's failed anyway and will call the helpdesk. The challenge here is to try and provide consistent performance even when there is peak demand. If our users are doing different things at different times, a high performance task such as opening a large spreadsheet in Excel will be offset by a user who is doing nothing more than reading their e-mail. However, if our users all sign in at the same time, certain parts of our infrastructure will come under pressure at the same time, which we must allow for as well. So how can we predict what resources our users will need? The following are some of the ways we can do this:

- We can take advice from Microsoft and its partners. In this chapter, I have tried to include as much best practice as I could find in white papers, from Microsoft Premier Field Engineers who implement this in production, and from the engineering team that develops RDS.
- We can pilot an implementation and put a host under more and more load to see how it performs.

- Use testing tools and simulators to put a VDI deployment under load to see how it performs.
- Assess the resources used by sample users on their real desktops and use those as a basis for capacity planning.

None of these are perfect as no two users behave in the same way and idealized testing must be a sort of an average; nonetheless, these are the best guides we have. It's also worth bearing in mind that VDI is as much about winning the hearts and minds of our users as it is about technology, so we should aim to give our users the best experience we can, consistent with the work they are required to do rather than limiting their experience for the sake of economy. For example, kiosk-like terminals on a factory floor are a good candidate for session-based collections serving out just one or two applications to a group of thin client terminals. On the other hand, hot-desking knowledge workers would benefit from full-fidelity Windows 8 on multiple monitors, so they can use rich applications and make use of unified communications. This is why we have the choice of collection types in RDS, which we covered in *Chapter 1, Putting the V in VDI – An Introduction to Virtualization in Hyper-V*.

In addition to capacity planning, we also need to tune all of the resources we have in our VDI deployments: networking, I/O, memory, and CPU, as any one of these will slow our users' response time down. The good news is that VDI in Windows does a pretty good job of balancing the resources we give it to a large extent, so each of our users gets a fair share without us having to constantly tune everything. Our main focus will be on tuning the settings of our Virtual Desktop Template and the hosts on which our collections will run.

We can break down our examination of scale and performance into three areas:

- The server roles managing our remote desktop services (the RD Brokers, RD Gateway, and so on)
- The hosts, be they session or virtualization hosts, that serve our virtual desktops
- The virtual desktop VMs if we are using VDI rather than Session Virtualization

We also need to be aware of the scalability limits. Actually, the limit on any pure Microsoft VDI deployment is not down to any hard performance limit of Windows Server—it is largely limited by the lack of management tools to control VDI. The only tools that Microsoft supplies out of the box in Windows Server to manage RDS are Server Manager and PowerShell. Server Manager simply can't handle deployments where there are hundreds of users online—it's simply too hard to find who is connected to what and fix any problems they might have in the list of users and connections in the collections view. We can, of course, use PowerShell scripts

to mash together a set of tools to do management at scale, but this wouldn't really be enterprise and production ready. So, Microsoft's unpublished but recommended guidance is that RDS by itself is really only suitable for deployments of about 500 desktops, despite the fact that it has tested more than 2,500 VDI deployments and has published performance white papers on how that was done. As in the past, Microsoft still recommends implementing partner solutions from Citrix, Dell, and so on.

Microsoft's white paper on the performance of a 2500-seat VDI deployment can be found at `http://download.microsoft.com/download/6/1/8/618657D7-9D3F-42BD-89F8-8C8963F9EC91/Windows_Server_2012_VDI_Deployment_Guide.pdf` and a confusingly similar but more detailed white paper at `http://download.microsoft.com/download/2/4/B/24B5EC7D-1D03-49A2-B792-C7EDF24549EE/Windows_Server_2012_Capacity_Planning_for_VDI_White_Paper.pdf`.

Testing RDS

If we want to test a VDI deployment, we need to put it under a predictable load and have repeatable tests to see the effect of any changes to our deployment. We also need an automated system for this, as we can't ask all our users or the IT department to log in and run a series of scripts. There are two sets of tools available, the Remote Desktop Simulation tools (`http://www.microsoft.com/en-us/download/details.aspx?id=2218`) from Microsoft and the more sophisticated but expensive cross-platform solution from LoginVSI (`http://www.loginvsi.com`). The Microsoft tool is very manual to set up and is only suitable for testing Session Collections. The following are required for its working:

- A test controller to orchestrate the testing
- A number of clients that simulate the load (the tool has been tested for 50 sessions per client)
- A server component that runs on the RD Session Host server being tested.

There are detailed instructions included in the download and a couple of sample test files. LoginVSI can be used for VDI of whatever type of collection and has the advantage of allowing us to contrast and compare the VDI solutions from more than one vendor on the same configuration. It was used by Microsoft for the white paper referred to in this chapter and so can easily handle simulation and testing for large deployments. It comes with a set of sample workloads and we can use these as is or configure a more realistic model for our organization, for example, to include applications that we use and to do the kind of printing we need.

One other mechanism that can be employed is using sizing tools such as Lakeside SysTrack (`http://www.lakesidesoftware.com/`) to determine what kind of resource requirements a typical user in a given department needs before they move to VDI and then making a collection from this sizing study. This method is less about simulation and more about the current resource profile.

Hyper-V

We have made use of Hyper-V in three ways in this book: to run the role services that manage our RDS deployment, to host virtual desktops in our pooled and personal VDI collections, and to provide two session hosts for our session-based collection. If we decide to use VMs for any of these, we must understand what we can do to tune Hyper-V itself and how to scale up if needed. Broadly speaking, we lose about seven percent of our performance by virtualizing a workload, but many new servers are over-specified on the CPU, so this isn't a big penalty for the agility and manageability we get in return.

When we first configured Hyper-V in *Chapter 1*, *Putting the V in VDI – An Introduction to Virtualization in Hyper-V*, all we did was add in a virtual switch having installed the role. Actually, there is hardly any need to set anything here; however, it is worth mentioning networking because at the moment we have just one NIC attached to our host that is handling all our traffic, which is not good for HA or performance. We could add in separate NICs and assign each to a separate kind of traffic, such as for connectivity to shared storage, management, and live migration; however, what we can do instead is to create a team of our NICs in Windows Server and rely on VLAN and subnets to separate out the traffic.

We can try out NIC teaming by just adding another NIC of any make (I use a simple USB NIC in my laptop for this) to our server. You then need to remove the existing NIC from Hyper-V by setting the virtual switch connected to the NIC to be internal. You can then team the two NICs by navigating to **Server Manager** | **Local manager** | **NIC teaming**. The final step is to make your virtual switch external again by associating it to the Microsoft Network Adapter Multiplexor Driver. I have a post on this at `http://blogs.technet.com/b/andrew/archive/2012/12/11/evaluate-this-nic-teaming.aspx`.

This means that our network traffic is now load-balanced efficiently across our NICs. If an NIC fails, our users won't be affected. We'll look at the specific impact of Hyper-V on each role and the collection type as we cover each role service.

RDS role servers

In the previous chapter, we spent a lot of time implementing HA for the brokers, and the Web Access and gateway servers in our VDI deployment. In the process, we also implemented load balancing so that each role is active rather than active-passive, where the standby server is idle most of the time. We also virtualized all of these server roles and discussed how they should be deployed on two Hyper-V hosts for HA. From a performance point of view, it is best practice not to put the role services on the same host as the one we use for our collections. In Microsoft's performance white papers, a 2,000-seat deployment is fronted by two Hyper-V servers, just as we set up for our HA configuration, where the two servers were Dell R620s with 16 cores and 96 GB of RAM, which was sufficient to run the RD Broker, Gateway, Web Access, and SQL Server for the broker-shared database.

Microsoft has general tuning guidelines for Windows Server 2012 R2, which includes computing, networking, storage, as well as the RD and Hyper-V roles. See `http://msdn.microsoft.com/en-us/library/windows/hardware/dn529133.aspx`.

RD Broker

The performance of the RD Broker role determines the connection rate and time to connect to our virtual desktops, so what can we do to ensure we are getting the best performance? We can scale up by adding more resources to a single RD Broker or scale out by creating an RD Broker farm, as we did for HA in the previous chapter. To recap, a broker farm has multiple RD Brokers accessing a shared SQL Server database. Load is balanced across the individual RD Brokers using DNS round-robin so that our HA solution is active. So does this improve performance?

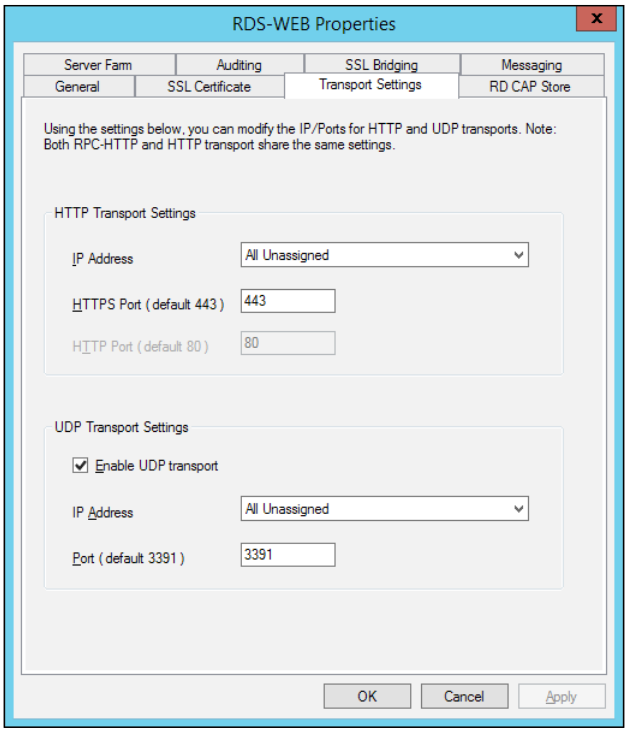
In smaller collections of less than 100 VMs, the broker can handle up to 10 connections per second, where the slowest connection takes less than a second and adding in a second broker will have little impact on this. Doubling the number of brokers doesn't give double the benefit. In Microsoft's tests on two quad-core CPUs, the connection time was reduced from 4.6 to 3.3 seconds when there were 2,500 VMs in the collection, and this rose to 3.7 seconds for a 5,000 VM collection. So, the only reason for having more than one RD Broker in all but the largest deployments is for high availability.

The Microsoft white paper on RD Broker performance based on using quad-core servers in an HA configuration using a shared SQL Server database can be found at `http://www.microsoft.com/en-us/download/details.aspx?id=38779`.

This means, if we decide to add in a third broker so that two are always available in case of a failure, the third server will have little or no impact on improving the connection response time. The broker is also capable of handing off connections to multiple session hosts. A two-broker farm can comfortably handle connections to over 100 session hosts in under 4 seconds. Again, adding more brokers doesn't really affect this. In these situations, it's the SQL Server database server that's under pressure, although it is far more efficient than the **Windows Internal Database** (**WID**) used in single RD Broker deployments. In the Microsoft white paper, a quad-core SQL Server instance is used with a minimum of 4 GB RAM. So, if more scale or performance is needed, we should look at adding more resources to SQL Server, such as using a higher edition and using the new in-memory engine in SQL Server 2014, or using SSD storage for at least the system tempDB database and increasing the memory available to it.

Tuning the RD Gateway and RD Web Access roles

RD Gateway performance is all about connectivity; our users connect over SSL port 443, which is then translated into UDP and TCP/IP traffic on port 3389. If performance is slow for our internal users, we should check that we have set **Bypass the RD Gateway for local addresses** by navigating to **Remote Desktop Services** | **Overview** | **Deployment Properties**. We should also ensure that we have got UDP properly configured on the RD Gateway by checking the **Transport Settings** in the properties of each Gateway server in the RD Gateway Manager, as shown in the following screenshot:



Microsoft has released a performance white paper on a 2,500 VDI deployment based on Windows Server 2012. In addition to detailing how to set up the collections, it also covers the setup of the role servers. In these tests, an RD Gateway with 4 cores and 8 GB RAM could support 1,000 connections a second, each with a throughput of 60 KB/sec.

The only other adjustment we can try is to tune the idle timeout setting of the RD Web Access Application pool that our Gateway is using in IIS. Have a look at the following example:

```
appcmd set config /section:applicationPools /[name='PoolName
'].processModel.idleTimeout:0.00:30:00
```

In this code, `PoolName` is the name of the application pool.

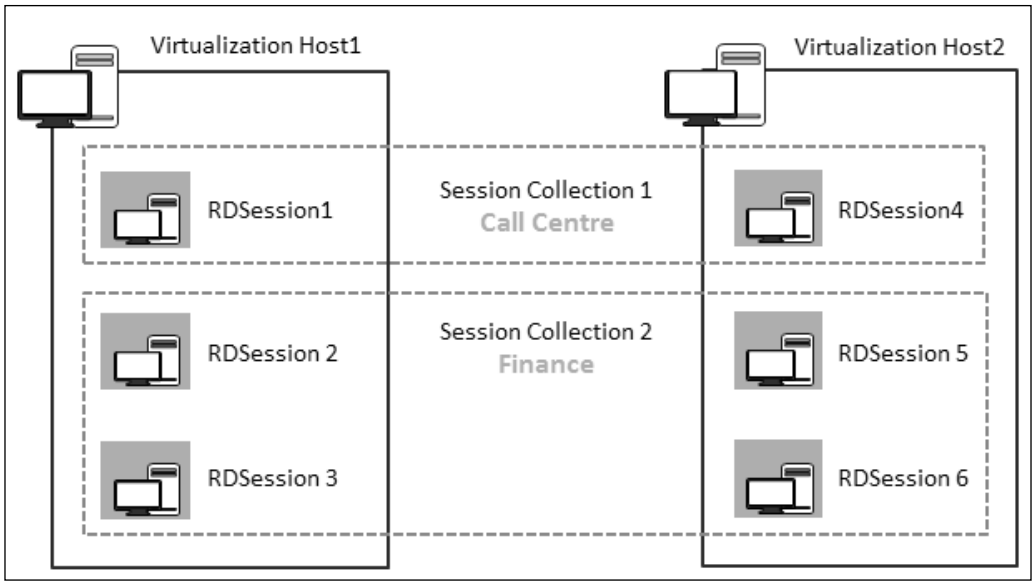
Session Collections

In a Session Collection, we need to ensure that a single server has enough resources for each user session running on it and enough left to support the underlying OS. Because everything is running on one copy of the OS and everyone has the same copy of the desktop, we need to be mindful to keep what is on each desktop to the minimum required.

Hyper-V in Windows Server 2008 R2 only allowed a maximum of four logical processors to be assigned to a VM, which isn't sufficient for larger Session Collections. However, with Server 2012 R2, we can add up to 64 logical processors to a VM if the hardware supports it, and so even the largest session hosts can now be virtualized and we'll only see a small loss in performance.

A logical processor can be likened to a core on a physical host, and Hyper-V can only assign logical processors to a particular VM up to the number of cores on the host or 64, whichever is lower. So, on a 32-core host, we can only have 32 logical processors in any VM, and on a 128-core host, it's the 64 core limit in a VM that would apply.

If we do decide to virtualize our session hosts, we should not over-commit the logical processors on that host. In other words, the sum of logical processors of all the VMs on the host should not exceed the cores on that host. One reason why I think virtualizing session hosts is a good idea is because it allows us to carve up a big server into several session hosts, which gives us resilience if one VM fails and allows us to patch and update each of these in turn. Our broker can handle and balance across multiple session hosts, so performance won't be a factor there. We can also use virtualization to use two hosts in an active-active scenario across two Session Collections, which might allow us to load-balance different types of users, as shown in the following diagram:



Multiple session hosts providing two Session Collections across two virtualization hosts

Testing Session Collections

Microsoft hasn't updated its performance testing white paper for Windows Server 2008 R2 (`http://www.microsoft.com/en-us/download/confirmation.aspx?id=17190`), but there's lots of useful information on how to test and on considerations for I/O memory and CPU. If we look at the testing of the most demanding user, from these tests, we can see that an 8-core, 2.4 GHz server of five years ago can support up to 310 users if we load the server up with a 128 GB RAM, so round about 400 MB of RAM per user. The other interesting statistic is the worst-case network traffic of 14 KB per user, which means that commodity servers will easily handle hundreds of users. What is hard to assess is disk I/O, as this will depend on where the profile disks are, the users' data files, and so on. If our session hosts are physical servers, a modern array of disks in the same rack will easily handle this throughput; the only recommendation would be to put the paging files on a local SSD.

Pooled and Personal Collections

The performance of these collections depends on the performance of the virtualization hosts our virtual desktops are running on and the settings and configuration of the Virtual Desktop Templates used to create the collections. Having got this right, we can then look at how many virtual desktops we can run on a given server for a given workload.

Virtual Desktop Template optimization

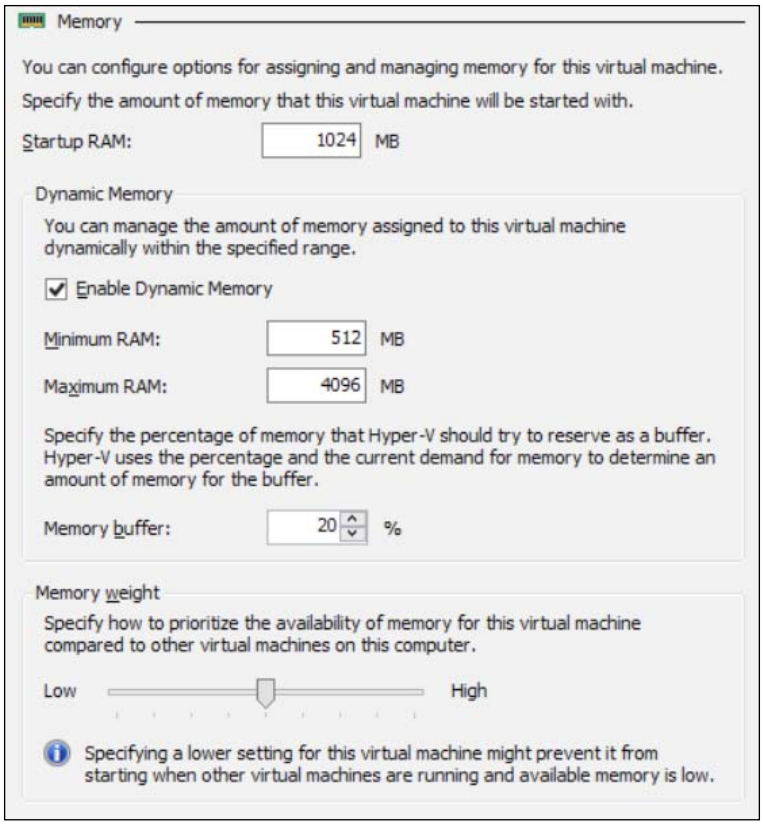
There are two areas to consider when configuring our templates: the settings of the VM itself and the settings inside the guest operating system. There are a lot of settings in a Hyper-V VM that are present specifically to configure the performance of VDI VMs.

Dynamic memory

Unlike VMWare, Microsoft does not over-commit memory except in one special case. So, if our virtualization host has 64 GB of memory, it has to be shared among the VM and the management OS.

The memory reserved for the management OS is set internally by Windows Server and should not be overridden unless advised to do so by Microsoft support.

What Hyper-V does instead is dynamically manage memory. The following screenshot shows the settings for a VDI template VM in Hyper-V:

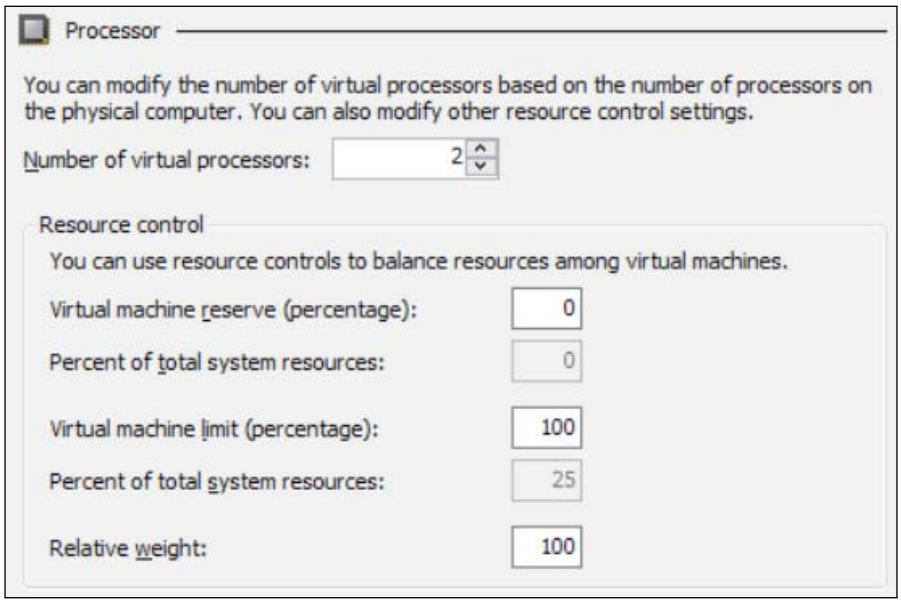


This allows the VM to start with a **startup RAM** setting of 1,024 MB to enable it to boot quickly. Once it's running, the memory allocated will drop to the **minimum RAM** setting of 512 MB unless the VM needs more, in which case the other settings come into play. The **memory buffer** allows us to determine when a VM asks for more memory; in this case, if the buffer falls below 20 percent (so 80 percent or more of the memory is in use), more memory will be allocated, up to the **maximum RAM** setting (if there is memory available). However, if the host is under memory pressure, the memory weight setting will determine which VMs get allocated more memory and which will not. This last setting will be the same for all the VMs in a collection, so each user will get a fair share. If we want to prioritize some users, we should create separate collections based on a VDI template that reflects this. Dynamic memory will also hand memory back from the VM to the host when the memory buffer is above the specified setting. As our users open and close applications, Hyper-V will automatically ensure that there is memory there to support them.

There is that one exception to over-committing memory that I mentioned earlier: if a VM has to restart when the server is under memory pressure, there may not be enough free memory to meet the startup RAM requirement. In this single case, Hyper-V will use a special **smart paging file** to allow the VM to restart. We can set the **Smart paging file location** in the properties of a VM. If we have a local SSD on our server, this would be a good use of that, as the file is only in temporary use.

Processor

We can set how many logical processors each VM will get and set relative weights, as shown in the following screenshot:



Processor settings for a VDT template VM

The resource controls will only be of value if there are different collections with different settings on the same host, as these will be the same for each VM in a VDI collection.

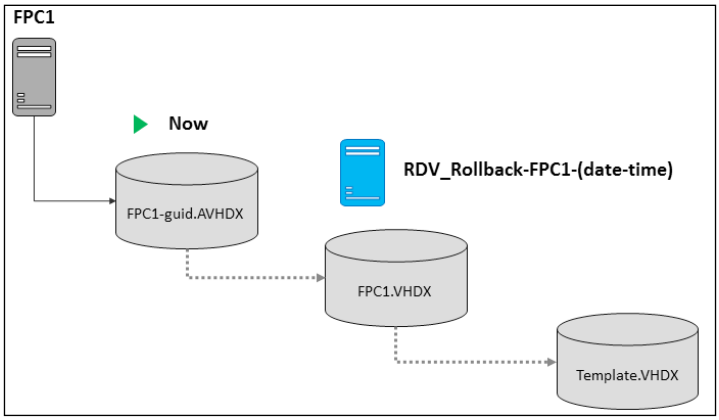
Networking

Quality of Service (**QoS**) can also be set for network bandwidth, but again, this will only really be relevant where there are multiple collections with different settings on the same host. Hyper-V can also leverage the latest improvements in NICs, such as IPSec offloading and **Server Root IO Virtualization** (**SR-IOV**) to free up CPU on our hosts, but these are not really suitable for Hyper-V as they bypass the physical

switch, and more importantly, you can't use SR-IOV and NIC teaming. As a result, we would lose HA and flexibility if we used this for our collections. The only other setting of interest in here is **Advanced Features** under **Protected Network**, which will automatically move a VM if its network is lost.

VM storage

I/O performance is a key part of VDI performance, but before we look at that, we need to understand how the VMs in our VDI collection are stored on disk. If we go into Hyper-V on our host and look at one of the VMs (FPC1) we made in *Chapter 2*, *Designing a Virtual Desktop Infrastructure*, we can see the chain of disks it has by going to **Settings** | **IDE Controller 0** | **Hard drive** and clicking on the **Inspect** button. If we keep clicking on the **Inspect** button, we will see that we have three hard disks that relate to our VM, as shown in the following diagram:



The relationship between virtual hard disks and snapshots for our FPC1-pooled VM

The current state of the VM is written to the checkpoint disk (`FPC1-guid.avhdx`) as the user works on their session. The disk associated with the rollback checkpoint is a VHDX with the same name as the VM. This captures the saved state of the VM as it started for the first time after sysprep and is what the VM will roll back to when a user logs off, so it's ready for the next user. This VHDX is a differencing disk based on a copy of the VHDX that was part of our Virtual Desktop Template VM. This copy (`Template.VHDX` on the previous diagram) is read-only, and we can control its location as part of the collection deployment properties.

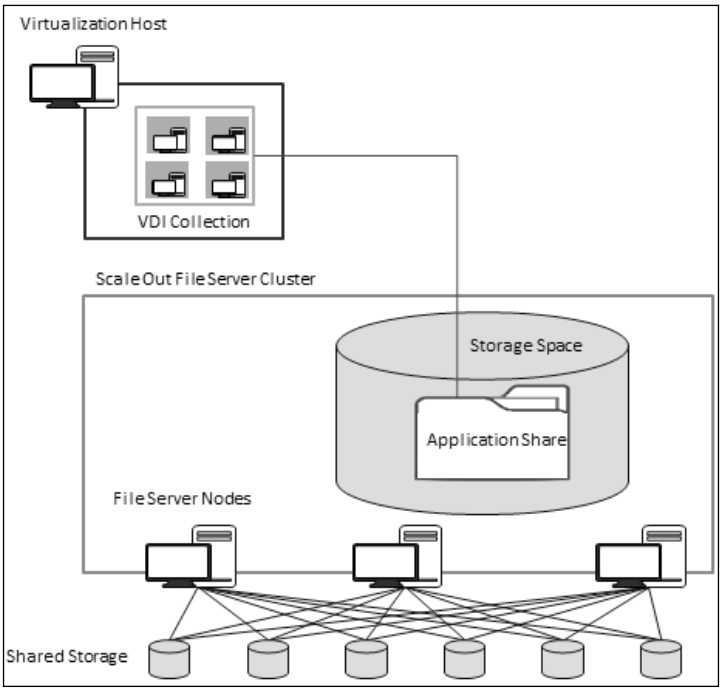
RDS was not the only technology that got a significant overhaul in Windows Server 2012; there were also major improvements to the storage engine, which gives us new options for storing our VMs. And these were further enhanced in Windows Server 2102 R2. Storage is a big topic and we'll just skim the surface here to cover what is relevant for hosting our virtual desktops. First of all, there are **Storage Spaces** and **Storage Pools**. Storage Pools allow us to group a bunch of commodity **Serial-Attached SCSI** (**SAS**), **Serial ATA** (**SATA**), or USB into something like a **Redundant Array of Independent Disks** (**RAID**).

Storage Pools also support disk enclosures, commonly known as **JBOD** (**just a bunch of Disks**), where the disks in the enclosure are not already in a RAID configuration. However, we can't use fiber channel and iSCSI-controlled storage for Storage Pools. For more details, refer to the Storage Spaces FAQ at `http://social.technet.microsoft.com/wiki/contents/articles/11382.storage-spaces-frequently-asked-questions-faq.aspx#What_types_of_drives_can_I_use_with_Storage_Spaces`.

We can then lay a special virtual hard disk, a storage space over this, format it and create shares on it, and use it just like any file server. This is relevant to VDI because instead of creating simple shares like we might do on any folder, we can create a special **Application Share** specifically designed to host VMs and SQL Server databases and use this as a location for our virtual desktops in our VDI collections.

Dell has a whitepaper on Storage Spaces at `http://en.community.dell.com/techcenter/extras/m/white_papers/20439139.aspx`.

The following figure shows how this fits together to provide storage for our virtual desktop VMs:



In the preceding diagram, we have three file servers in a scale-out file server cluster that are connected to our shared storage. This is a bit like a SAN, where each file server can be considered a storage controller. Like a SAN, Storage Spaces also allows us to declare RAID-like configurations, such as mirror, parity, and hot spare, to survive disk failures. If a File Server Node fails, then the next node automatically picks up the connection so quickly that if we were playing video, we would only lose one or two frames. Modern storage is increasingly moving toward hybrid solutions, where SSDs are integrated into the solution to provide faster access to "hot" blocks of data that are in continuous use. Windows Server 2012 R2 emulates this capability with tiered storage in storage spaces so that when we create a space, we can declare how much SSD and how much hard disk to use. The Windows Server 2012 R2 filesystem will then work out which are the hot blocks and automatically store them on SSD for us. We can also override this behavior by pinning files to the SSD tier. For example, we could pin our VHD template VHDX on the SSD tier just to be sure it's there:

```
$SSDTier = Get-StorageTier –FriendlyName "Collection Space" | Get-
StorageTier -MediaType SSD
Set-FileStorageTier -FilePath S:\TemplateParent.VHDX
-DesiredStorageTier $SSDTier
```

In this code, `Collection Space` is the storage space we have already created. Within that, there is already an SSD Tier. `S:\TemplateParent.VHDX` is the copy of the VHDX template specified in our deployment, and `S:` is the drive to our storage space. We can also do this from the file server inside the Server Manager.

Another new storage technology introduced in Windows Server 2012 is deduplication, which modifies the way the Windows File System (NTFS) works by examining common chunks across files and removing the duplicates. This is an out-of-band process that is normally scheduled to run on a low priority in the background or out of hours. Initially, Microsoft did not support putting VHD for running VMs on deduplicated volumes, but this was changed in Windows Server 2012 R2 for one use case: pooled VDI Collections. If we look at how collections create and use VHDXs, we have one parent template VHDX that's read-only and we have a differencing disk for each VM in our collection that are practically identical. In fact, apart from their unique ID (SSID) and name, they are identical and make excellent candidates for deduplication as they don't change; it's only the VHDX at the end of the chain that changes while our users are logged on.

I have two step-by-step posts on storage:

- How to set up tiered storage (`http://blogs.technet.com/b/andrew/archive/2013/10/02/labb-ops-part-3-storage-in-windows-server-2012r2.aspx`).
- How to set up a scale-out file server using two VMs rather than two physical hosts (`http://blogs.technet.com/b/andrew/archive/2013/12/02/lab-ops-part-10-scale-out-file-servers.aspx`).

Two other performance tips that might be useful are as follows:

- Enable caching for the **Cluster Shared Volumes** (**CSV**), which is where we can allocate memory on the file server nodes in our cluster to be used, in much the same way as the cache on a SAN Controller. CSV cache is not compatible with tiered storage or deduplication, but deduplication already has a built-in caching system and this would be redundant anyway.

For details on how to configure this, refer to this MSDN post: `http://blogs.msdn.com/b/clustering/archive/2012/03/22/10286676.aspx`.

- Look at the setting storage QoS in the properties of the VM template to ensure fair use between the VMs in a VDI Collection. This can be done in the properties of the VDI template VM by expanding the plus sign next to the hard disk and navigating to the advanced features. We can set the maximum and minimum IOPS that the selected disk on that VM can have. Remember that each VM will get the same setting and that setting this also enables it to be monitored in the event logs of the host and through WMI. We can also set this on the fly for a given VM by using the `Set-VMHardDisk` PowerShell command with the `-minimumIOPS` and `-maximumIOPS` switches.

Tuning Windows 8 for VDI

In *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, we looked at how MDT enabled us to manage the creation of a Virtual Desktop Template; however, we didn't do any specific tuning of the desktop for VDI. For a start, Microsoft recommends that we use the 64-bit version of Windows 8, although this takes up a bit more space as it works better with RemoteFX, especially if we plan to virtualize a real GPU on our host. We can eliminate the 350 MB partition Windows 8 creates for BitLocker, as we won't need it for VDI, by adding a rule to our Task Sequence `DoNotCreateExtraPartition=YES`. We should also review the configuration of the services that are running in our template. Microsoft recommends that we should disable the following (note that a lot of these are already set to manual):

Service	Default setting	Description
Application Layer Gateway Service	Manual	This service manages mobile broadband [**Global System for Mobiles** (**GSM**) and **Code Division Multiple Access** (**CDMA**)], data card / embedded module adapters, and connections by autoconfiguring the networks. Microsoft strongly recommends that this service be kept running for the best user experience of mobile broadband devices.
Background Intelligent Transfer Service	Manual	VDI infrastructure is usually connected to fast LAN or WAN links to infrastructure servers hosting data.
BitLocker Drive Encryption Service	Manual (TS)	BitLocker is not available for use in VMs.
Block-Level Backup Engine Service	Manual	This service is used to back up data on the client computer; it is not used for VMs.
Bluetooth Support Service	Manual (TS)	Bluetooth Wireless is not supported from VMs.

Service	Default setting	Description
Computer Browser	Manual (TS)	This maintains an updated list of computers on the network and supplies this list to computers designated as browsers.
Device Association Service	Manual (TS)	This enables pairing between the system and wired or wireless devices.
Device Setup Manager	Manual (TS)	This enables the detection, download, and installation of device-related software. If this service is disabled, devices may be configured with outdated software and may not work correctly.
Diagnostic Policy Service	Automatic	This service enables problem detection, troubleshooting, and resolution for Windows components. If this service is stopped, diagnostics will no longer function.
Diagnostic Service Host	Manual	The Diagnostic Policy Service uses the Diagnostic Service Host to host diagnostics that need to run in a Local Service context. If this service is stopped, no diagnostics that depend on it will function.
Diagnostic System Host	Manual	The Diagnostic Policy Service uses the Diagnostic System Host to host diagnostics that need to run in a Local System context. If this service is stopped, no diagnostics that depend on it will function.
Family Safety	Manual	This service is a stub for the Windows Parental Control functionality that existed in the Windows Vista operating system. It is provided for backward compatibility only.
Fax	Manual	This enables you to send and receive faxes using fax resources available on this computer or on the network.
Function Discovery Resource Publication	Manual	This publishes the computer and resources attached to this computer so that they can be discovered over the network. If this service is stopped, network resources will no longer be published and will not be discovered by other computers on the network.
Home Group Listener	Manual	This is used to establish Home Groups; it is not used with VMs in a corporate environment.
Home Group Provider	Manual (TS)	This is used to establish Home Groups; it is not used with VMs in a corporate environment.
Microsoft iSCSI Initiator Service	Manual	**Internet SCSI** (**iSCSI**) will not be used on virtual desktops.

Service	Default setting	Description
Microsoft Software Shadow Copy Provider	Manual	This manages software-based volume shadow copies taken by the Volume Shadow Copy service. If this service is stopped, software-based volume shadow copies cannot be managed. If this service is disabled, any services that explicitly depend on it will fail to start.
Offline Files	Manual (TS)	
Optimize Drives	Manual	This helps the computer run more efficiently by optimizing files on storage drives.
Secure Socket Tunneling Protocol Service	Manual	This service publishes a machine name using the Peer Name Resolution Protocol. The configuration is managed via the netsh context `p2p pnrp peer`.
Shell Hardware Detection	Automatic	This provides notifications for AutoPlay hardware events.
SNMP Trap	Manual	This receives trap messages generated by local or remote **Simple Network Management Protocol** (**SNMP**) agents and forwards the messages to the SNMP management programs running on this computer. If this service is stopped, SNMP-based programs on this computer will not receive SNMP trap messages. If this service is disabled, any services that explicitly depend on it will fail to start.
SSDP Discovery	Manual	This discovers networked devices and services that use the **Simple Service Discovery Protocol** (**SSDP**), such as **Universal Plug-and-Play** (**UPnP**) devices. It also announces SSDP devices and services running on the local computer. If this service is stopped, SSDP-based devices will not be discovered. If this service is disabled, any services that explicitly depend on it will fail to start.
Telephony	Manual	This provides Telephony API support for programs that control telephony devices on the local computer and, through the LAN, on servers that are also running the service.
UPnP Device Host	Manual	This allows the computer to host UPnP devices. If this service is stopped, any hosted UPnP devices will stop functioning and no additional hosted devices can be added. If this service is disabled, any services that explicitly depend on it will fail to start.

Service	Default setting	Description
Windows Backup	Manual	This provides Windows Backup and Restore capabilities.
Windows Color System	Manual	The WcsPlugInService service hosts the non-Microsoft Windows Color System color device model and the gamut map model plug-in modules.
Windows Connect Now – Config Registrar	Manual	WCNCSVC hosts the Windows Connect Now Configuration, which is Microsoft's implementation of the Wi-Fi-Protected Setup protocol.
Windows Error Reporting Service	Manual (TS)	This allows errors to be reported when programs stop working or responding and allows existing solutions to be delivered. This also allows logs to be generated for diagnostic and repair services.
Windows Media Player Network Sharing Service	Manual	This shares Windows Media Player libraries with other networked players and media devices using UPnP.
WLAN AutoConfig	Manual	The WLANSVC service provides the logic required to configure, discover, connect, and disconnect from a wireless LAN (WLAN), as defined by **Institute of Electrical and Electronics Engineers** (**IEEE**) 802.11 standards.
WWAN AutoConfig	Manual	This service manages mobile broadband (GSM and CDMA), data card / embedded module adapters, and connections by autoconfiguring the networks. Microsoft strongly recommends that this service be kept running for the best user experience of mobile broadband devices.

Microsoft also recommends that we consider disabling the following services:

Service	Default setting	Description
Microsoft BranchCache	Manual	This service caches network content from peers on the local subnet.
Distributed Link Tracking Client	Automatic	This tracks NTFS filesystem links locally and across the network (but only if the service is also running on the remote system).
Encrypting File System Manual	Manual (TS)	This provides the core file encryption technology used to store encrypted files on NTFS filesystem volumes. If this service is stopped or disabled, applications will be unable to access encrypted files.

Service	Default setting	Description
Extensible Authentication Protocol (EAP)	Manual	The EAP service provides network authentication in such scenarios as 802.1x wired and wireless networks, virtual private network (VPN), and Network Access Protection (NAP).
File History Service	Manual (TS)	This protects user files from accidental loss by copying them to a backup location.
Microsoft Account Sign-In Assistant	Manual (TS)	This enables user sign-in through Microsoft account identity services. If this service is stopped, users will not be able to log on to the computer with their Microsoft account.
Sensor Monitoring Service	Manual (TS)	This monitors various sensors to expose data and adapt to system and user state. If this service is stopped or disabled, the display brightness will not adapt to lighting conditions. Stopping this service may affect other system functionality and features as well.
Themes	Automatic	This provides user experience theme management.
Volume Shadow Copy	Manual	This manages and implements volume shadow copies used for backup and other purposes. If this service is stopped, shadow copies will be unavailable for backup and the backup may fail. If this service is disabled, any services that explicitly depend on it will fail to start.
Windows Defender	Automatic (TS)	This helps protect users from malware and other potentially unwanted software.
Windows Search	Automatic (Delayed)	This provides content indexing, property caching, and search results for files, email, and other content.

Finally, we should enable one service that's actually manual by default: the Network List Service, which identifies the networks to which the computer has connected, collects and stores properties for these networks, and notifies applications when these properties change.

Rather than manually edit this list, Microsoft Premier Field Engineer Jess Stokes has produced a VB Script to modify Windows 8 in this fashion on his blog *Hot off the presses, get it now, the Windows 8 VDI optimization script, courtesy of PFE!* at `http://blogs.technet.com/b/jeff_stokes/archive/2013/04/09/hot-off-the-presses-get-it-now-thewindows-8-vdi-optimization-script-courtesy-of-pfe.aspx`.

We can then continue to thoroughly review other aspects of the OS, such as:

- Remove as many items as possible from the task scheduler and review what is running, particularly for third-party applications
- Set the event logs to a much smaller size and ensure they are set to overwrite
- Disable System Restore and Hibernate

Finally, before we sysprep the image, we should perform a disk cleanup with the built-in tool in Windows 8 plus conduct a thorough check of our own, as this will reduce I/O in itself and the impact of antivirus I/O if that's installed as well.

Clearly, we will want to limit what applications we put on our Virtual Desktop Template, but there is one we'll certainly need, and that is the one that will have the biggest impact on performance: our antimalware solution. This kind of software has a bad reputation on real desktops. So, having multiple copies of it running inside our collections will compound any performance problems. We essentially have two options here:

- Run the agent inside each VM as we would on a real desktop. Microsoft estimates that we will experience a 10 to 12 percent increase in disc I/O per VM.
- Scan our VDI template and create exceptions for operating system files. If a user gets a virus notification, we can shut down the VM and it will revert to its snapshot and the user just has to log in again.

We will cover keeping AV signatures up to date along with patching the desktop OS and application in the next chapter.

Capacity planning for VDI collections

In Microsoft's white paper on testing a 2,000-seat pooled VDI collection, 14 identical virtualization hosts were used to run 150 VMs per host. This was then put under load by using the LoginVSI tool running the standard medium workload that comes with it. Each of the virtualization hosts had 16 cores and 256 GB RAM, and the VMs were stored locally on each server across a set of 10 x SAS disks (72 GB 15K RPM). The Virtual Desktop Template was configured as per the guidance in this chapter and Office 2013 was installed. The conclusion of the tests was that:

- One core is needed per 10 users to keep the core below 80 percent utilization.
- Each VM needed about 1 GB RAM when running Windows 8 and Office 2013. However, this test didn't make use of RemoteFX for graphics-intensive tasks. We should allow a maximum of another 1 GB of RAM for those users who will be running more graphics-intensive applications, as will be discussed in the upcoming sections.
- I/O was about 10 IOPS per VM for a running VM.
- 400 KBPS bandwidth was needed per user.

Client settings

Now that we understand what to do to scale and tune the remote desktop role servers, we can consider how to ensure our virtual desktops of whatever type are running efficiently while providing our users with a rich desktop experience. Our users are going to want access to the capabilities of their local devices, to make use of unified communications, to print, and to touch the screens on their tablets. This is all provided in RemoteFX, which is available for both session- and VDI-based virtual desktops. RemoteFX was originally based on virtualizing the GPU in a graphic card, and it will still do this if there is a supported card in the physical server we are using. Now, in Windows Server 2012, it covers a range of technologies to enrich the client experience and make the best use of limited bandwidth. We have already discussed USB redirection to enable local devices such as web cameras and printers to be used inside a virtual desktop, but what's important here is the impact on performance, which comes down to how much bandwidth we have. RemoteFX will automatically compensate for slower bandwidth WAN connections by making use of UDP as well as TCP on port 3389. UDP is used for traffic such as audio and video, where having smooth rather than choppy streaming is really important at the expense of error checking and waiting for missing packets. UDP is also used where possible for interactive graphics and touch control, but if UDP is blocked, TCP will be used instead. Apart from the progressive rendering of web content and images and smoother video streaming, RemoteFX in RDP 8 also allows support for unified communications such as Lync inside a virtual desktop and will work over WAN connections as well. All of these benefits come at a price, and that is the memory overhead needed for RemoteFX, which we need to be aware of. In the physical world, a graphics card has both dedicated and shared memory (VRAM) available for graphics-intensive workloads, and these concepts are then virtualized in RemoteFX for VDI.

We can go into our VDI VM and look at the VRAM by running the **DXDiag** (**DirectX diagnostic**) tool, but this number will have no relation to the VRAM on a real graphics card if we are using one. Things changed for Windows 8.1 and we'll get either 128 MB out of 256 MB of dedicated VRAM and anywhere from 64 MB to 1 GB of shared VRAM depending on the startup RAM allocated to the VM based on the following calculation:

```
TotalSystemMemoryAvailableForGraphics = MAX(((TotalSystemMemory - 512)
/ 2), 64MB)
```

The definitive post on VRAM for RemoteFX in Windows 8.1 is at `http://blogs.msdn.com/b/rds/archive/2013/12/04/remotefx-vgpu-improvements-in-windows-server-2012-r2.aspx`.

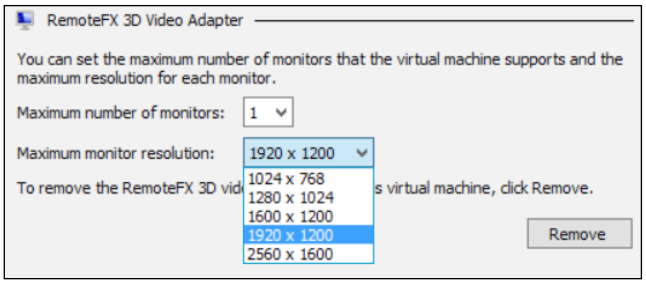
This all becomes useful over remote connections, but there's no point in having this capability if it's not secure. So, the UDP traffic also makes use of SSL to ensure that this traffic is encrypted as well. There's very little we can or need to do to enable RemoteFX.

RemoteFX makes use of **Second Level address Translation** (**SLAT**) on modern CPUs, and this technology will only work if this capability is present. Intel refers to SLAT as EPT and AMD as **Nested Page Tables** (**NPT**).

However, if we do want to use this technology to virtualize graphics cards in our hosts, we need to do two things: we need to declare the graphics card to Hyper-V in Hyper-V settings and then add in a RemoteFX graphics card in to our VMs. Remember, if we do this for a VDI template, all the VMs in a given collection will have this feature; also remember that we'll need this same setup on all our virtualization hosts if we want to move those VMs from host to host.

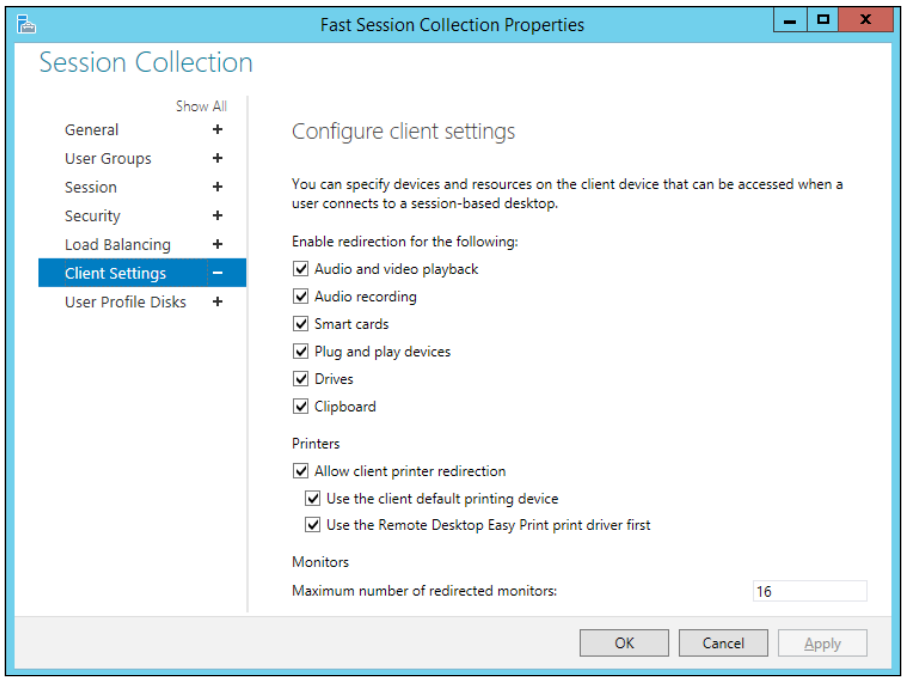
If we want to test it, the easiest way to do that is to connect to our VDI deployment from a VM running Windows 8 and edit its settings in Hyper-V to set the network adapter QOS to simulate a typical WAN bandwidth, for example, 2 MBPS, and then connect to our VDI and run a desktop.

If you want to see what RemoteFX does when using a real graphics card, you'll need a supported card (the list of supported cards is at `http://server.windowsmarketplace.com`) in the physical server. You can then go into a VM in Hyper-V and add in a RemoteFX graphics card:



If this VM is your Virtual Desktop Template, you could build a collection that would have the ability to access the GPU on the host for graphics-intensive programs such as CAD and market trading solutions.

When our users connect to a virtual desktop, we can either allow them to set up their preferences or control them centrally from the deployment properties of our collection in Server Manager:



Client settings for a Session Collection

We have already noted in *Chapter 4*, *Putting the R in Remote Desktop*, that we might disable some of these for security reasons. The defaults can be left as is for good performance; for example, if we do allow local printing, the Remote Desktop Easy Print Driver is a much better option for this, as we won't need to install printer drivers on our virtual desktops, which can soak up resources and which need to be maintained. The more monitors we allow our users to have, the more bandwidth and processing power we will need, but again RemoteFX will automatically minimize the impact of this on our users and on our servers.

Desktop as a Service

The future of RDS may well mean that we start to use **Desktop as a Service** (**DaaS**), where we can simply rent desktops based on Windows Servers acting as RD Session Hosts from cloud providers such as Amazon, Google, and Microsoft. Amazon already has this service in place, but it's based on the older Windows Server 2008 R2 technologies, and there are also some major hosting providers who also offer this. Microsoft has not offered its own DaaS, but that is about to change and we will be able to license RDS on VMs running in Microsoft Azure's cloud platform for running our software and our VMs. This will allow us to either upload or create a VM on Azure configured to provide session-based collections that can be accessed from anywhere with an Internet connection. We can also configure a Gateway and integrate this infrastructure with what we have running in our data center with Active Directory Services and VPN tunneling so that Azure just becomes an extension of what we have already. This can also be integrated with another Microsoft online service, Office 365, so that our users can work completely independently of our data center and we can scale this up and down as we need, for example, if we decide to take on more staff or contractors. We still have work to do to make this happen, such as the configuration of our session desktops to provide any applications our users need and to secure and integrate these Azure VMs with our own infrastructure for seamless sign-on, group policy, and patching. At the time of this writing, there is no sign yet of a cloud-based VDI service, where we would be able to spin up Pooled or Personal Collections of Windows client-based VMs on platforms such as Azure, but this is technically possible; it's just not licensed yet.

Summary

Remote desktop services are designed to be scalable, and the main challenge we have is to plan the capacity we need to support a given user base and profile. In this chapter, we have seen that there is a lot of guidance out there and that LoginVSI can be used to verify the performance of our collections against standard or customized workloads. We have also covered performance tuning of the role services and the setting of our collections and how to configure Windows 8 for VDI. Now that we have a secure, fast, and reliable VDI deployment, we need to think about how to maintain and manage it and that's what we will look at in the next chapter.

7
Maintenance and Monitoring

In this chapter, we will look at how to keep our VDI deployments up to date with the latest patches and updates, while minimizing the impact of planned maintenance on our users. We'll look at monitoring our deployment and how to troubleshoot the problems that might occur. We'll finish up with a quick introduction to Microsoft System Center and the parts of the suite that are relevant to VDI.

Maintenance

Our VDI deployment needs to be maintained to keep it up to date, and this affects the Remote Desktop role services and hosts as well as the virtual desktops. We will need to deploy patches and fixes from Microsoft that could affect Windows Server, the Windows Client OS, and any Microsoft applications we have deployed on our desktops. The same applies to any third-party applications and, in particular, whatever anti-malware solution we have in place to protect our virtual desktops. Finally, we will need to modify our VDI deployments in response to change requests from the business, such as to provide more virtual desktops to new users and update them with new applications.

Windows Server Update Services

The update and patching element of VDI maintenance incurs a big overhead for little return, as the best that can happen is that VDI isn't broken, infected, or compromised. Hopefully, it's a process that's already established in our IT department, as this applies to all our real desktops and other infrastructure servers. Typically, this is all done with **Windows Server Updates Services** (**WSUS**).

This is a Windows Server role service that is a central management point where patches and updates from Microsoft can be managed. We can then use Group Policy settings to direct our servers and desktops to get updates from our WSUS server rather than directly from Microsoft, which gives us a number of benefits, as follows:

- We can have a process to test patches and updates against our server and then deploy them once we are satisfied; they won't break anything in production. However, we can elect to deploy certain classes of patches automatically, for example, critical updates and security fixes.
- We only have to expose our WSUS server to the Internet to get the patches, and we don't clog up our Internet connection by repeatedly downloading the same patch for each desktop or server that needs it.

WSUS stores all our updates and fixes centrally on a designated share, and is backed by a database to store the metadata about each update and data about what patches have been installed, approved, and so on. So, we can get rich reports on how the services are working. In a production environment, this is typically SQL Server, although it is possible to use WSUS with the database that comes with Windows Server, the **Windows Internal Database** (**WID**). WSUS can also be extended by using System Center Configuration Manager, which allows us to have desired baselines for our desktops and servers and report on compliance against these.

WSUS applies equally well to VDI and allows us to manage updates for all parts of our deployment: the Remote Desktop role services, virtualization, and session hosts, as well as the virtual desktops themselves.

Installing and configuring WSUS

We have already deployed SQL Server in our lab for the RDS Broker HA database running on RDS–Ops, so I am going to suggest we deploy WSUS on that server as well.

If you haven't installed SQL Server on RDS-Ops already, then please refer to the *Creating an RD Broker Farm* section of *Chapter 5*, *High Availability*, which will help you get do a basic setup of SQL Server that we can use for WSUS.

We'll also need to expose this VM to the Internet to see how WSUS works. The easiest way to achieve that is to add in a second virtual NIC into the RDS-Ops VM that is connected to a virtual switch on our host and has Internet access. We will perform the following steps:

1. From **Hyper-V Manager** on the physical host, go to **Virtual Switch Manager**.
2. Select **New Virtual Switch** and select **External**, and then click on **Create Virtual Switch**.
3. Set the name of the switch to `RDS-Internet`, and in the **External Network** drop-down menu, select an NIC on the host that is connected to the Internet. Click on **OK** to create the virtual switch.

On my laptop, I create a new external switch, which is either bound to the onboard wireless NIC or plugin a USB NIC, depending on whether I have access to the Internet over a wireless or wired connection

4. Right-click on the **RDS-Ops** VM and select **Shutdown**.
5. When the VM has shut down, right-click on it again and select **Settings**.
6. From the **Add Hardware** icon, select **Network Adapter** and click on **Add**.
7. In the **Virtual Switch** drop-down menu, select **RDS-Internet** and click on **OK** to add the new network.
8. Right-click on **RDS-Ops** and select **Start**. RDS-Ops should now be connected to the Internet.

WSUS needs to store the updates in a folder for onward distribution to individual servers and desktops. When we set up **Microsoft Deployment Toolkit** (**MDT**) in the *Installing MDT* section of *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, we also created a separate volume `E:` on RDS-Ops using the new deduplication feature in Windows Server 2012; this will ensure that the space taken up by updates is also kept to a minimum. All we need to do before we install WSUS is create a folder, `E:\Updates`, to use this volume. We can now log on to RDS-Ops and install WSUS by performing the following steps:

1. From the **Server Manager** menu, navigate to **Manage | Add Roles and Features**.
2. Click on **Next** on the **Before You Begin** screen.
3. On the **Installation Type** screen, select **Role-based or feature-based installation** and click on **Next**.

4. On the **Destination server** screen, click on **Next** as RDS-Ops is already selected for us.
5. In the **Server Roles** screen, select **Windows Server Update Services**. In the pop up that appears, simply click on **Add Features**. Then click on **Next**.
6. In the **Server features** screen, click on **Next**.
7. Read the notes in the **Windows Server Updates Services** screen and click on **Next**.
8. In the **Role services** screen, select **WSUS Services** and **Database** only, as we will be using SQL Server and not the WID database. Then click on **Next**.
9. In the **Content Location** screen, enter `E:\Updates` and click on **Next**.
10. In the **Database Instance selection** screen, enter `.\` to specify the default instance on the local server. Click on **Check connection** to ensure it's working and then click on **Next**.
11. In the **Web Server Role (IIS)** screen, click on **Next**. Click on **Next** to skip off the web services screen as the wizard has made the right choices for us.
12. Review the settings and click on **Install**.

You might see a warning triangle in **Server Manager** telling us that we need to perform some post-deployment tasks, but if we click on this, it doesn't do anything as the wizard has already done everything.

This is the equivalent PowerShell script:

```
#Install the WSUS features the database integration and management
tools
Add-WindowsFeature ("UpdateServices-Services","Updateservices-DB") -
IncludeManagementTools
#Create the folder for the updates
if(!(test-path E:\updates)){Md e:\Updates}
#Configure WSUS to use the local SQL Server instance and use the
folder we just made
Start-Process "C:\Program Files\Update Services\Tools\wsusutil.exe" -
ArgumentList 'postinstall SQL_INSTANCE_NAME=localhost
CONTENT_DIR=E:\Updates'
```

We can now configure WSUS in our environment to select where to get updates from, what updates to get, which servers to apply them to, and how approvals of different types of updates are set. We will perform the following steps:

1. On RDS-Ops, navigate to **Server Manager** | **Tools** | **Windows Server Update Services**.
2. In the **Update Services** console, navigate to **RDS-Ops** and expand it. Select **Options** and click on the bottom option, **WSUS Configuration**.
3. Click on **Next** on the **Before You Begin** screen, and again on the **Join the Microsoft improvement Program** screen.
4. Click on **Next** on the **Choose Upstream Server** screen. In a small lab setup like this, we'll be downloading updates directly from Microsoft. In larger production environments, there might need to be a hierarchy of WSUS servers to distribute updates across large and distributed sites, where child WSUS servers get updates from another parent WSUS server.
5. On the **Proxy Server** screen, click on **Next** or configure a proxy if you have one in place.
6. Click on **Start Connecting** and grab a coffee, as this will take a few minutes!
7. On the **Choose languages** screen, check **Download updates only in these languages** and choose your local language. Click on **Next**.
8. On the **Product** screen, we'll select just the following options, and click on **Next** to continue:
    - Everything related to **Windows 8.1**
    - **Windows Defender**
    - All the **Windows Server 2012 R2** options
    - **Microsoft SQL Server 2012**
9. In the **Classifications** screen, note the options and click on **Next**.
10. In the **Set Sync** screen, leave the option **Sync Manually**, but note that we could schedule this and click on **Next**.
11. In the **Finished** screen, check the option **Begin initial synchronization** and click on **Next**.
12. Review the other things we will need to do and click on **Finish** to complete the wizard.
13. After the initial download has been completed (which can take a while!), expand **Computers** and right-click on **All Computers** and select **Add Computer Group**.
14. Call the group `RDS-Servers` and click on **Add**.
15. Finally, we will be using Group Policy to assign computers to groups, and so, we need to go to **Computers** in the **Options** tab and check the **Use Group Policy or registry** setting on **Computers**; then click on **OK** to confirm this.

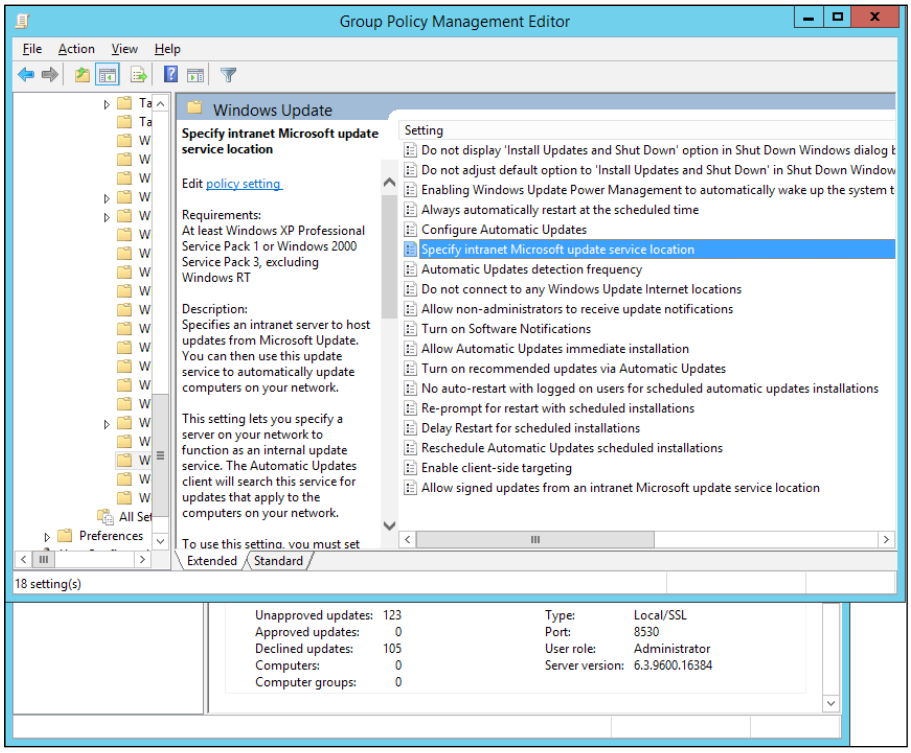
Now we can assign updates to our RDS servers by selecting one or more of them, right-clicking on them, and selecting **Approve**. When we do this, we can assign which group the updates are approved for. There is a limited set of PowerShell commands to approve updates and check status, for example, to approve all the critical updates to our `RDS-Servers` group, we can use the following command:

```
Get-Wsusupdate | where classification -eq "critical updates" |
approve-wsusupdate -Action Install -TargetGroupName "RDS-Servers"
```

We can also configure WSUS to approve all updates of a given type automatically from **Automatic Approvals** in the **Options** tab.

If we want to look at the status of an individual update, then we'll get an error: **Microsoft Report Viewer Distributable is required for this feature**. Fortunately, we can click on the link in the error and install it!

This will now go away and pull down an initial set of updates, and when it completes, we will get an overview of how WSUS is set up, as shown in the following screenshot:



As the preceding wizard shows, we need to do a few things to start using this service, such as approving updates and assigning updates to computers, but before we dive in and set up our update process, we need to think about how to use WSUS in a VDI deployment. For our servers, we will apply all the updates we approve directly to our servers, but we might want to have a different approach for our personal and pooled virtual desktops.

Maintaining the RDS servers and hosts

We are now in a position to set up our RDS servers and hosts to receive updates, and the way this is usually done is with Group Policy. To keep things simple, we will create an **Active Directory** (**AD**) group with all our servers in and create a Group Policy linked to that group. To create the group in AD, we will perform the following steps:

1. Connect to the **RDS-DC** VM, and in **Server Manager**, navigate to **Tools** | **Active Directory Administrative Center**.
2. Navigate to **Contoso** | **Computers** and right-click on the middle pane. Then navigate to **New** | **Group**.
3. Call the group **RDS-Servers**, and in the **members** tab, click on the **Add** button.
4. In the **Select Users, Contacts, Computers** dialog box, change the object type and only select **Computers**. Type in `RDS` and click on **Check Names**. Select all the objects and click on **OK**. We could also add in our physical host (mine is called orange). Click on **OK** to create the group.

The following two lines of PowerShell create the group and add in the computers (which you can see in the PowerShell history windows in ADAC):

```
New-ADGroup -GroupCategory:"Security" -GroupScope:"Global" -
Name:"RDS-Servers" -Path:"CN=Computers,DC=Contoso,DC=com" -
SamAccountName:"RDS-Servers" -Server:"RDS-DC.Contoso.com"

Set-ADGroup -Add:@{'Member'="CN=RDS-
BROKER,CN=Computers,DC=Contoso,DC=com", "CN=RDS-
BROKER2,CN=Computers,DC=Contoso,DC=com", "CN=RDS-DC,OU=Domain
Controllers,DC=Contoso,DC=com", "CN=RDS-
OPS,CN=Computers,DC=Contoso,DC=com", "CN=RDS-RODC,OU=Domain
Controllers,DC=Contoso,DC=com", "CN=RDS-
SHOST,CN=Computers,DC=Contoso,DC=com", "CN=RDS-
SHOST2,CN=Computers,DC=Contoso,DC=com", "CN=RDS-
WEB,CN=Computers,DC=Contoso,DC=com", "CN=RDS-
WEB2,CN=Computers,DC=Contoso,DC=com",
"CN=ORANGE,CN=Computers,DC=Contoso,DC=com"} -Identity:"CN=RDS-
Servers,CN=Computers,DC=Contoso,DC=com" -Server:"RDS-DC.Contoso.com"
```

We can now create a **Group Policy Object** (**GPO**) to set up this group of servers to use WSUS for updates by performing the following tasks:

1. Connect to the **RDS-DC** VM, and in **Server Manager**, navigate to **Tools | Group Policy Management**.
2. In the **Group Policy Management** console, navigate to **Group Policy Management | Forest | Contoso.com**.
3. Right-click on **Contoso.com** and select **Create a GPO in this domain and link it here**.
4. Call the policy **Update RDS-Servers** and click on **OK**.
5. Click on the new **GPO** and ignore the warning.
6. In the **Security Filtering** window, highlight **authenticated users** and click on **Remove** and **OK** to confirm.
7. Click on **Add**, and in the **Select Users, Computer or Group** screen, enter `RDS-Servers` and click on **Check Names**. Click on **OK** to add the group to the GPO.

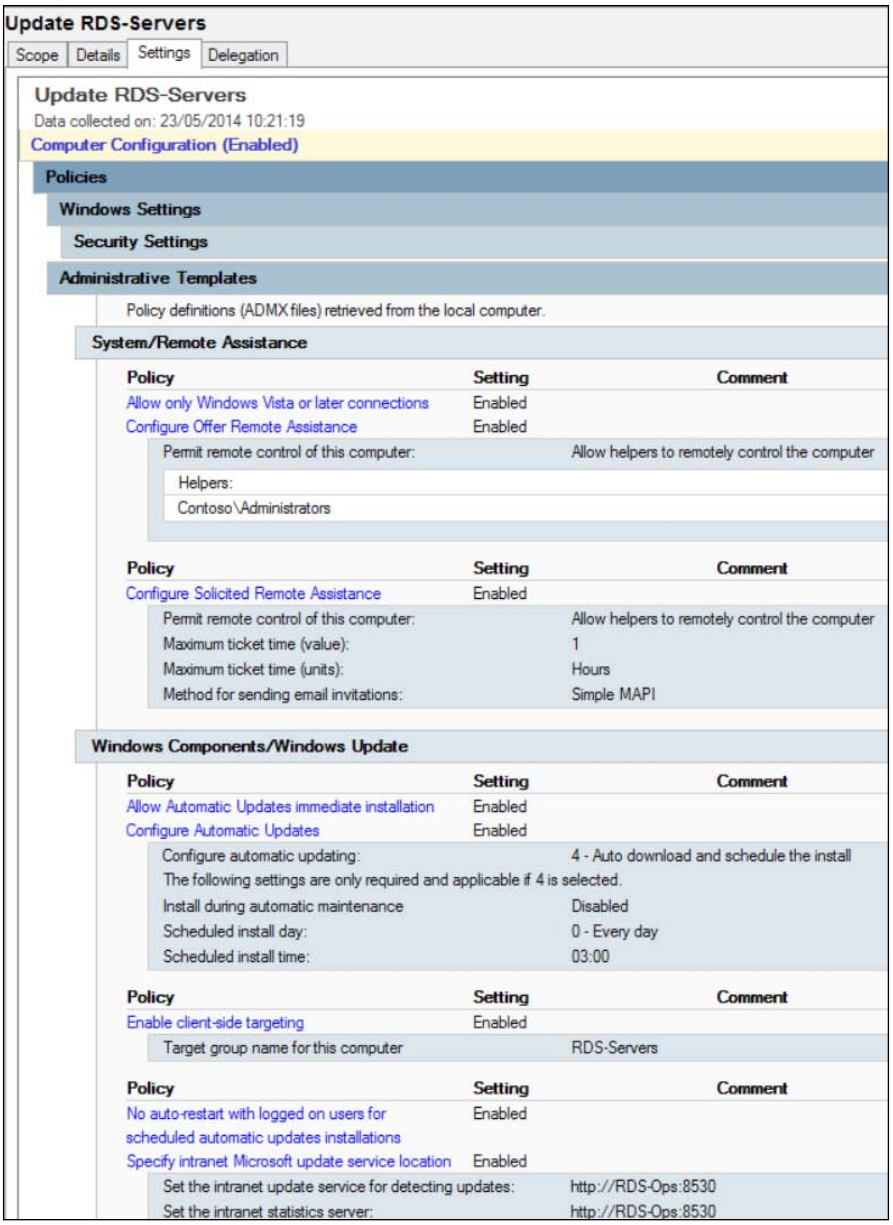
Before we dive in and edit the GPO, we need to think about how we should apply updates to our RDS servers and hosts, preferably without affecting our users. We have already seen how to implement HA in *Chapter 5*, *High Availability*, and we can build on this by updating each server role in such a way that at least one instance is in place while the other is offline. For example, for the RD Broker role in our lab, we will update and reboot RDS-Broker and bring it online before updating RDS-Broker2. This is only important if an update needs a reboot. WSUS doesn't really have a mechanism for this, so what can we do to stop WSUS installing updates that could reboot all of our servers at the same time? We could split our servers into separate groups in WSUS and configure separate update schedules for each of these in Group Policy so that they are set to update at different times. Another approach would be to prevent any of our servers from rebooting automatically after updates and have a separate maintenance script to test if there are any servers pending a reboot, and then do that in a sequence. For now, let's edit our GPO to see how to get WSUS working at a basic level by performing the following steps:

1. Right-click on the GPO and select **Edit** to open the **Group Policy Management Editor** dialog box.
2. Navigate to **Computer Configuration** | **Policies** | **Computer Management** | **Administrative Templates** | **Windows Components** | **Windows Update**.
3. We need to specify `RDS-Ops` as our WSUS server. For that, we perform the following steps:
    1. Double-click on **Specify intranet Microsoft update services location**.
    2. Click on **Enabled**.
    3. Set the option **Set the intranet update service for detecting updates:** to `http://RDS-Ops:8530`.
    4. Set the option **Set the Intranet Statistics server:** to `http://RDS-Ops:8530` and click on **OK**.
4. We want our RDS servers to be assigned the RDS-Servers group in WSUS. We will perform the following steps for that:
    1. Double-click on the **Client Side Editing** policy.
    2. Click on **Enabled**.
    3. Set the **Target group name for this computer** option to **RDS-Servers** and click on **OK**.
5. We can then schedule when our updates are applied by performing the following steps:
    1. Double-click on the **Configure Automatic Updates** policy.
    2. Set the **Configure Automatic updating** option to **4 – Auto download and schedule the install**.
    3. Note that we have the option to set when this is done and click on **OK**.
6. We then need to enable the following policies in the same way:
    - **No auto-restart with logged on users for scheduled automatic updates installations**. This will prevent our servers from restarting until we want to initiate that process.
    - **Allow Automatic Updates immediate installation**.

The final thing we will want to do is configure our domain firewall via Group Policy to allow our servers to see the WSUS intranet site, which is on port 8530 by default, by performing the following steps:

1. Still in Group Policy, navigate to **Computer Configuration** | **Policies** | **Windows Settings** | **Security Settings** | **Windows Firewall with Advanced Security**. Right-click on **Inbound Rules** and select **New Rule**.
2. In the **Rule Type** screen, select **Port**.
3. In the **Protocol and Ports** screen, enter `8530` in the **Specific local ports** field and click on **Next**.
4. Click on **Next** to skip over to the **Action** screen.
5. In the **Profile** screen, uncheck **Private and Public** and click on **Next**.
6. In the **Name** screen, call the rule **WSUS default update port** and click on **Finish** to create the rule in the policy.

We can quickly review the changes we have made by selecting the new policy (**Update RDS-Servers**), clicking on the **Setting** tabs, and then expanding the sections that have an enabled flag on them, as shown in the following screenshot:



Update RDS-Servers

Scope | Details | Settings | Delegation

Update RDS-Servers

Data collected on: 23/05/2014 10:21:19

Computer Configuration (Enabled)

Policies

Windows Settings

Security Settings

Administrative Templates

Policy definitions (ADMX files) retrieved from the local computer.

System/Remote Assistance

Policy	Setting	Comment
Allow only Windows Vista or later connections	Enabled	
Configure Offer Remote Assistance	Enabled	
Permit remote control of this computer:	Allow helpers to remotely control the computer	
Helpers:		
Contoso\Administrators		

Policy	Setting	Comment
Configure Solicited Remote Assistance	Enabled	
Permit remote control of this computer:	Allow helpers to remotely control the computer	
Maximum ticket time (value):	1	
Maximum ticket time (units):	Hours	
Method for sending email invitations:	Simple MAPI	

Windows Components/Windows Update

Policy	Setting	Comment
Allow Automatic Updates immediate installation	Enabled	
Configure Automatic Updates	Enabled	
Configure automatic updating:	4 - Auto download and schedule the install	
The following settings are only required and applicable if 4 is selected.		
Install during automatic maintenance	Disabled	
Scheduled install day:	0 - Every day	
Scheduled install time:	03:00	

Policy	Setting	Comment
Enable client-side targeting	Enabled	
Target group name for this computer	RDS-Servers	

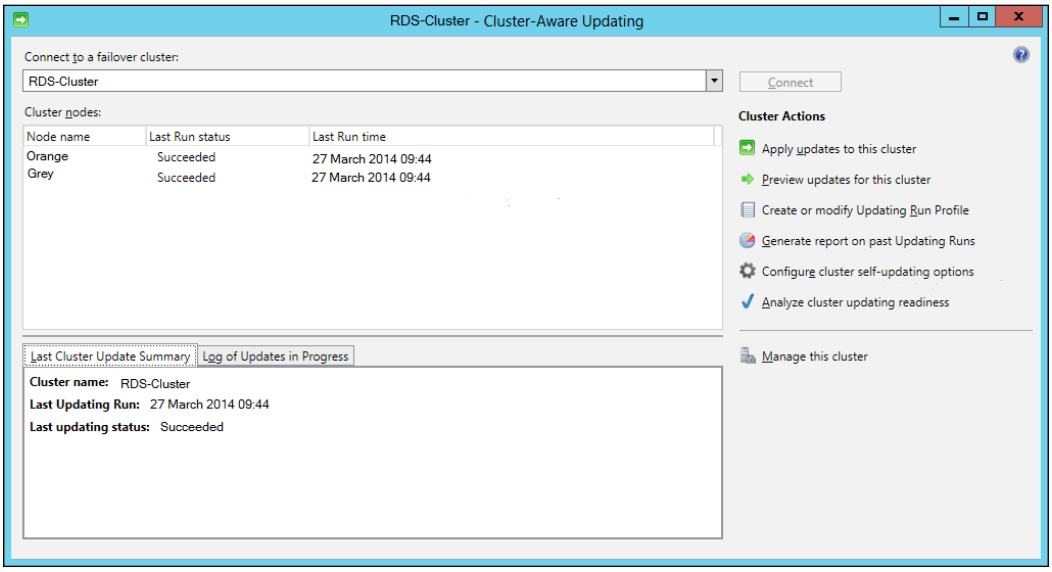
Policy	Setting	Comment
No auto-restart with logged on users for scheduled automatic updates installations	Enabled	
Specify intranet Microsoft update service location	Enabled	
Set the intranet update service for detecting updates:	http://RDS-Ops:8530	
Set the intranet statistics server:	http://RDS-Ops:8530	

Group Policy and WSUS will take some time to settle in and start working. If you haven't used Group Policy before, you can use the wizard under **Group Policy Modeling** and **Group Policy Results** to troubleshoot.

If we have to restart a virtualization host as part of an update, then any VMs running on them won't be available while the physical host reboots, such as our Pooled and Personal Collection, as well as any server roles we have virtualized. If we have implemented Hyper-V clusters as part of our HA solution to protect our RDS role services or as the location for pooled and personal desktops, then we can use a new feature of Windows Server 2012 to automatically move the VMs to other nodes in the cluster during an update. **Cluster Aware Updating** (**CAU**) is the technology used, and it is installed as part of the tools for managing a cluster along with Failover Cluster Manager. CAU can either be implemented as a role on the cluster so it updates itself, or it can be run remotely from another server or desktop with RSAT installed. PowerShell cmdlets exist to support this as well. We'll need special rules and Group Policy in place to use this feature, and they are outlined as follows:

- Automatic updates should not be applied to the cluster
- All cluster nodes should be in the same target group so that they get the same updates from the same source
- Have a process in place to ensure that the right updates are applied to the cluster

Apart from those, it is simply a matter of implementing the update process and monitoring the status of the updates in WSUS. The **Cluster Aware Updating** console is shown in the following screenshot:



The Cluster Aware Updating console

Refer to `http://technet.microsoft.com/en-us/library/hh831694.aspx` for more on setting up and using CAU.

Virtual desktops

We can patch our virtual desktops with WSUS in exactly the same way as we can for the server roles; we can create a GPO for them and create a group in WSUS to control which updates they get. However, our Pooled and Personal Collections may contain hundreds or even thousands of desktops, all of which will need to be patched in the same way at the same time, and this will create a massive spike in I/O, which is referred to as an I/O storm. This could bring our whole VDI deployment to a standstill. So, what can be done to minimize this problem?

- If we are using Pooled Collections or Personal Collections that are reset after a user logs off, we could update the entire collection by patching just the VDI template and then using the option in RDS to recreate the collection.
- If we have to use persistent Personal Collections, then these will have to be individually patched. So, we would have to write some code of our own to try and patch groups of desktops rather than updating them all, possibly using different scheduling windows for desktops sharing the same storage or host servers.
- We can use **System Center Configuration Manager 2012 R2** (**CM12**), the latest version of Microsoft's device management solution. This has a non-controllable randomized mechanism for patching built-in virtual desktops. We'll see how this works later in the chapter.

Recreating pooled virtual desktops

We can quickly extend the task sequence we created in the *Building a new Virtual Desktop Template with MDT* section of *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, to automatically pick up any approved updates from WSUS. We will perform the following steps:

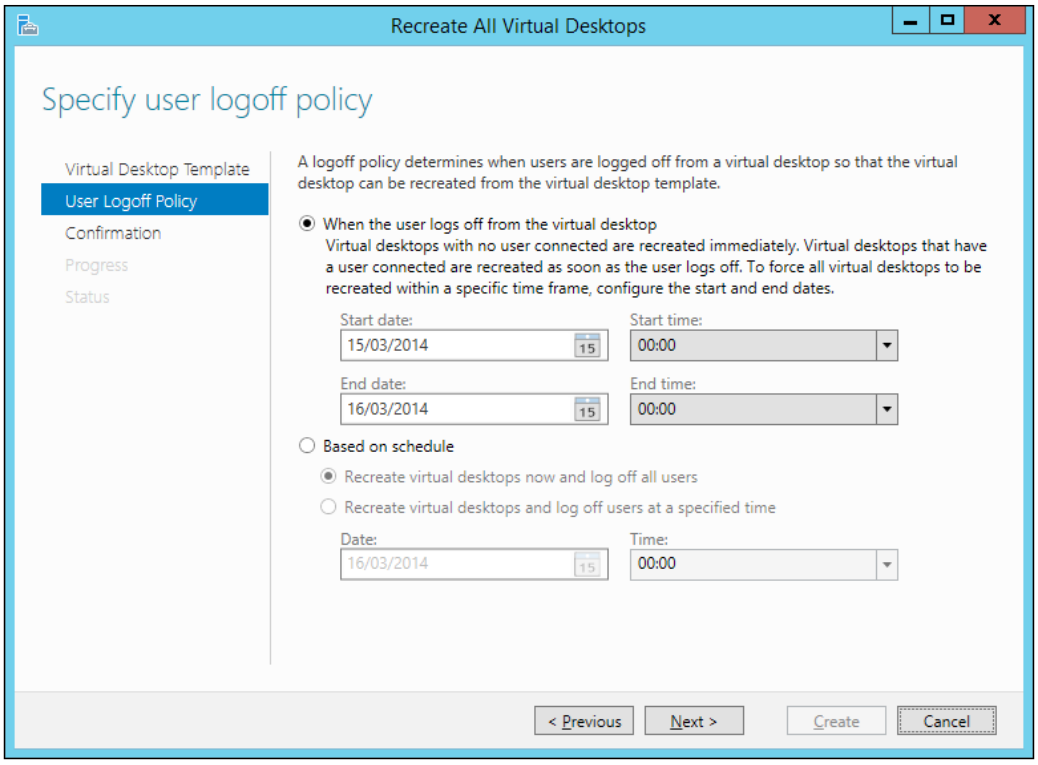
1. Connect to RDS-Ops and open the **Deployment Workbench** console.
2. Navigate to **Deployment Workbench** | **Deployment Shares** | **MDT Deployment Share** (`E:\Deployments`) | **Task Sequences**.
3. Right-click on **Windows 8.1 Reference Deployment** and select **properties**.

4. There are two places where updates can be applied and both of these are in the **State Restore** node: **Windows Update (Pre-application Installation)** and **Windows Update (Post-application Installation)**. Select **Windows Update (Post-application Installation)** and uncheck **Disable this step**. Click on **OK** to close the task sequence.

We need to set the location of our WSUS server as there won't be any Group Policy applied during the OS installation. We will perform the following steps:

1. Right-click on the **MDT Deployment Share** environment and select **Properties**. From the **Rules** tab, select **Edit Bootstrap.ini** and add the following under the **[Default]** section to point to our WSUS server web service, `WSUSServer=http://RDS-Ops:8530`.
2. Right-click on the **MDT Deployment Share** environment and select **Update Deployment Share**. Accept the defaults and click on **Finish** to close the wizard when it has completed.

We can now rerun the PowerShell script in the *Creating the reference computer* section of *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, to create the reference computer (RDS-Ref). We then need to recreate our desktop collection, which we can do by navigating to **Server Manager Remote Desktop Services** | **Collections** | **Collection Name** (in our case, Fast Pooled Collection), and in the **Virtual Desktops** pane, navigating to **Tasks** | **Recreate All Virtual Desktops**. We can then decide which new template to use (for example, RDS-Ref created in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*) and how our users are affected by this change. The following screenshot shows the **Recreate All Virtual Desktops** wizard:



We can wait for our user to log off, set a grace period for this, after which they will be forced to log off, or immediately log them off (first option in the preceding screen) or at a scheduled time (second option). This is going to depend on how critical the update is and how much we wish to disrupt our users for the sake of security. If we decide to do this, we can use the following PowerShell command:

```
Update-RDVirtualDesktopCollection `
   -ConnectionBroker          RDS-Broker.contoso.com `
   -CollectionName           "Fast Pooled Collection"`
   -VirtualDesktopTemplateHostServer orange.contoso.com `
   -VirtualDesktopTemplateName    RDS-Ref `
   -DisableVirtualDesktopRollback  `
   -ForceLogoffTime          12:00am `
```

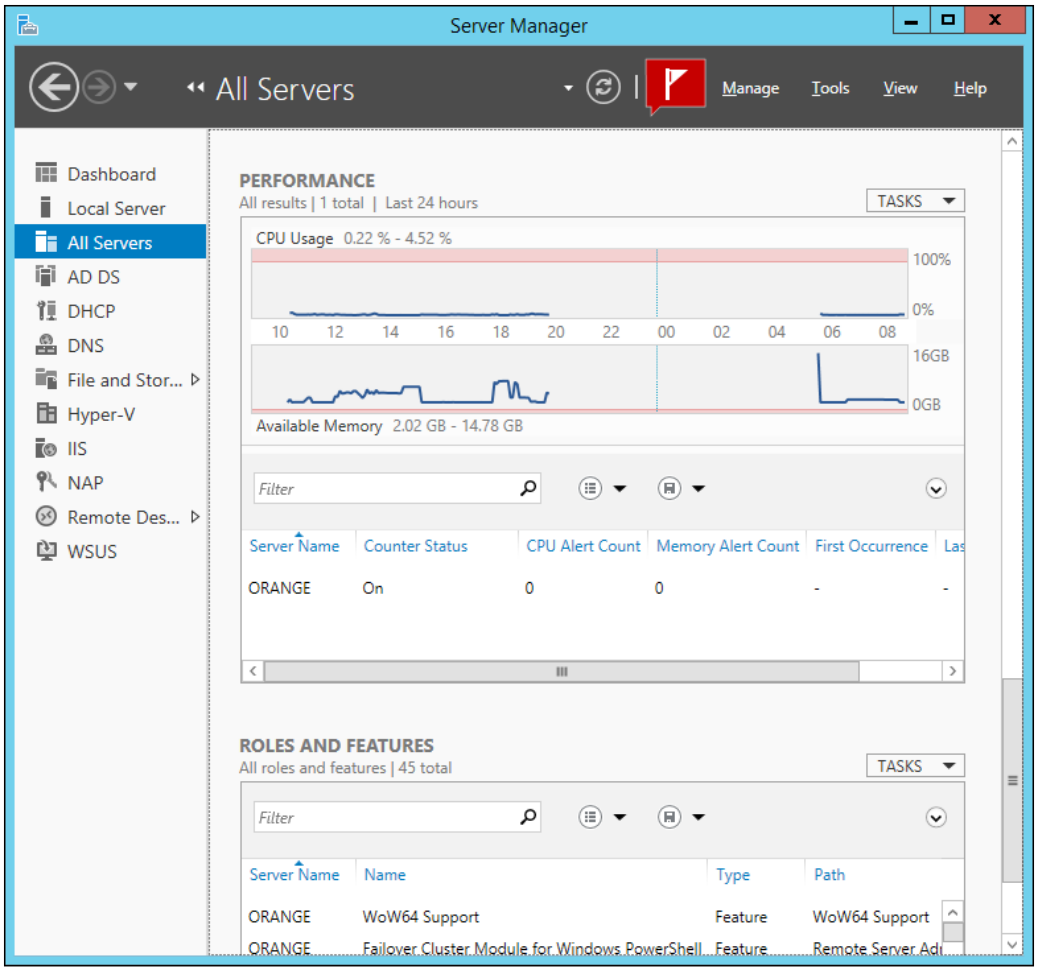
The preceding code is easier to read because I have split the line up using the ` character, which is the PowerShell line continuation character. This means it's easy for us to read and it will still execute properly, so it is a bit like the ; character in SQL.

Then, we can add this command to the script to create the reference computer and schedule this process to run once a week after Microsoft releases its patches to fully automate the update process after we have approved any patches.

If we have created a Personal Collection with unmanaged desktops—in other words, they will keep their virtual desktops as it is between sessions—then we have no choice but to patch these machines directly as though they are real desktops. There is no UI in Remote Desktop Services to control this, but there is PowerShell support with `Set-RDPersonalVirtualDesktopPatchSchedule`. As the name suggests, we can set up schedules for individual desktops to avoid the I/O storm problem. We can manage which patches the desktops get by assigning a GPO to the OU that the virtual desktops are part of. This GPO should reference a named group of computers in WSUS in the target group name in the client-side settings entry, as we did earlier for our role servers.

Monitoring

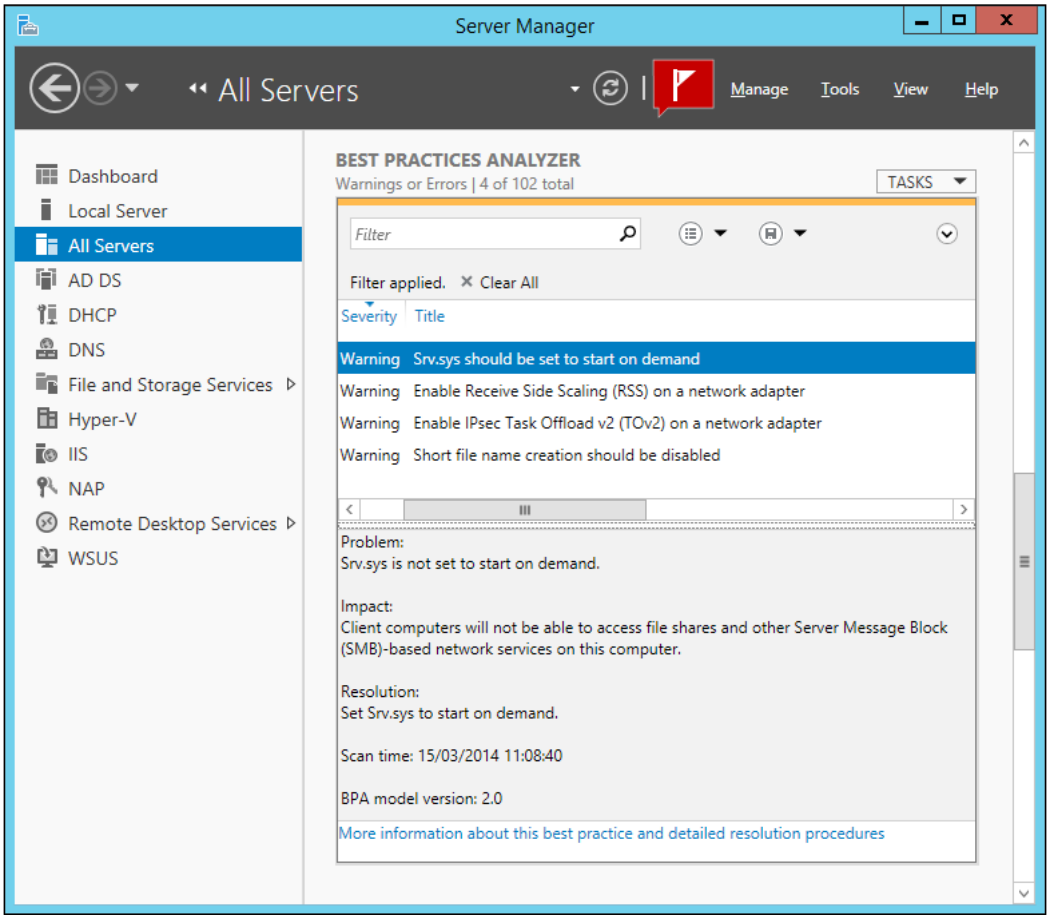
There are a variety of built-in tools that we can use to ensure our VDI deployments are working properly. Firstly, any errors in our VDI deployment will show up in the Server Manager dashboard view, and the section for our RDS roles will turn red and direct us to the problem. We can then enable a couple of other features to give us more information about the performance and smooth running of our deployments. We can turn on performance counters for our role servers and hosts by right-clicking on each server and selecting **Start performance counters**. This is particularly useful for keeping an eye on the session and virtualization hosts. The following screenshot depicts performance counters:



Performance counters for a Virtualization Host

This will show up any problems with the servers that are coming under pressure (if any), and we can set the memory threshold that triggers the alert; the default is 2 MB and a more realistic figure might be 500 MB. There is also the built-in **Best Practice Analyzer** (**BPA**), which will pick up any errors and warnings that might not be obvious in the Event Viewer.

To run the BPA, go to **Server Manager** on RDS-DC and scroll down to the section heading, **BEST PRACTICES ANALYZER**. Click on **Tasks** and select **BPA**. This brings up the following screen:



Best Practices Analyzer results for RDS-Broker

Managing and shadowing users' sessions

If our users encounter problems on their remote desktops, then it can be useful to log in to their session and help them resolve their problem. Clearly, this could be a security risk but we can do this by only allowing our helpdesk staff to have this capability, and our users can see what is happening and grant permission to allow this. If our users are using a session-based desktop, then we can do this by navigating to **Server Manager** | **Remote Desktop Services** | **Collection** |

<Collection name> and finding their connection in the **Connections** pane by right-clicking and selecting **Shadow**. We can let the user know that this is what we are doing and ask their permission to interact with their session. We can also send them a message, perhaps to ask them to log off before we have to restart their session for some planned maintenance, and there's a PowerShell command for that too, as follows:

```
Send-RDUserMessage -HostServer "rds-SHost.contoso.com" -
UnifiedSessionID 1 -MessageTitle "Message from Administrator" -
MessageBody "Please save your work. You will be logged off in 15
minutes"
```

If we want to use a command line to shadow a session, then we need its session ID from PowerShell; for example, in our lab, we use the following command:

```
Get-RDUserSession -ConnectionBroker "RDS-Broker.contoso.com" -
CollectionName "Fast Pooled Collection"
```

And, then we can run `MSTSC` from the command line or in PowerShell with a new (for RDP 8.1) option to shadow the session as follows:

```
mstsc /v:<ServerName> /shadow:<SessionID>
```

This capability doesn't exist for Pooled and Personal Collections, so we will have to rely on the user knowing that they can ask for Remote Assistance via Easy Connect, which is part of Windows 8.1. This is all enabled in Group Policy. Firstly, we need to set up Remote Assistance for a GPO linked to the OU for our collections, which can be done by navigating to **Computer Configuration** | **Policies** | **Administrative Templates** | **System** | **Remote Assistance**. We enable the following features over there:

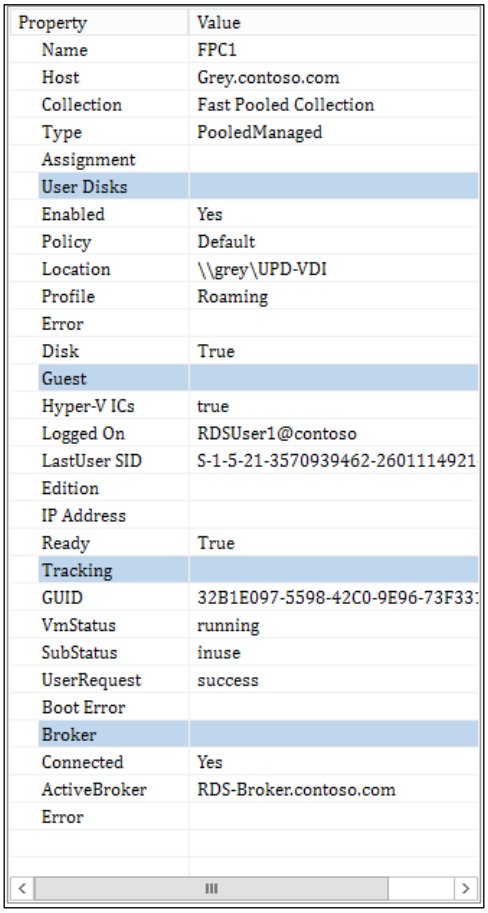
- Enable **Configure Offer Remote Assistance** and set **Permit helpers to remotely control the computer** to **Allow helpers to remotely control the computer**. Set the **Helpers** group to a domain group for the helpdesk team, and we can test this with `Contoso/Administrators` in our lab setup. Click on **OK**.
- Enable **Configure Solicited Remote Assistance**. Set **Permit remote control of this computer** to **Allow helpers to remotely control the computer**.

We then need to alter the firewall for this policy (**Computer Configuration** | **Policies** | **Windows Settings** | **Security Settings** | **Windows Firewall with Advanced Security**) and create two new inbound rules on TCP port 135: one for the program `%WINDIR%\System32\msra.exe` and the other for `%WINDIR%\System32\raserver.exe`.

The Remote Desktop Diagnostic tool

The Microsoft RDS engineering team have released a beta tool for RDS diagnostics called RDV Diag (`http://www.microsoft.com/en-us/download/details.aspx?id=40890`). It needs to be installed on an RD Broker and needs to be run with an administrator account. It has five different tabs within it for looking at each aspect of our VDI deployment as follows:

- The **Virtual Machines** tab gives us a complete readout of every VM on our virtualization hosts just by clicking on each one. This is shown in the following screenshot:



Property	Value
Name	FPC1
Host	Grey.contoso.com
Collection	Fast Pooled Collection
Type	PooledManaged
Assignment	
User Disks	
Enabled	Yes
Policy	Default
Location	\\grey\UPD-VDI
Profile	Roaming
Error	
Disk	True
Guest	
Hyper-V ICs	true
Logged On	RDSUser1@contoso
LastUser SID	S-1-5-21-3570939462-2601114921
Edition	
IP Address	
Ready	True
Tracking	
GUID	32B1E097-5598-42C0-9E96-73F33
VmStatus	running
SubStatus	inuse
UserRequest	success
Boot Error	
Broker	
Connected	Yes
ActiveBroker	RDS-Broker.contoso.com
Error	

Diagnostic details for the FPC1 VM in a Pooled VM Collection

All is well in the preceding screenshot, but if there were problems, such as the VM not being managed by the broker, then there would be appropriate errors in the **Broker** section. The other tabs give us a lot more detail about the health of our deployment.

- The **Collections** tab also gives us a lot more information than we can get from Server Manager just by clicking on the collection we want to examine, such as the details of the job that created the collection. This is shown in the following screenshot:

Property	Value
Name	Fast Pooled Collection
Alias	Fast_Pooled_Coll
Description	
Type	PooledManaged
User Groups	contoso\VDI-Users
Auto User Assignment	False
User has Admin Rights	False
Is Rollback Enabled	True
Is shown in RDWeb	True
Size	2
Percent in Use (%)	100
Prefix	FPC
Domain	Contoso.com
OU	OU=RDS-VDI,DC=contoso,DC=com
HighlyAvailable	False
Export Path	\\RDS-BROKER\RDVirtualDesktopTemplate
Storage	
Type	LocalStorage
Local Location	default
Central Location	
Export Location	\\RDS-BROKER\RDVirtualDesktopTemplate\Fast_Pooled_Coll\IMGS_1
User Disks	
Is Enabled?	True
Path	\\grey\UPD-VDI
Maximum Size (GB)	20
IncludeFolderPath	
IncludeFilePath	
ExcludeFolderPath	
ExcludeFilePath	
Virtual Desktop Template	
Name	RDS-VDITemplate
Host	Grey.contoso.com
Provisioning	
Status	JOB_COMPLETED
Start Time	16/03/2014 07:28:57
Last Modified Time	16/03/2014 07:37:09
Total VMs	2
Percent Completed	100%
Failed VMs	0
Job Deadline	

Diagnostic information for the Fast Pooled Collection

This gives information about the job that created the collection and pulls in all of the collection information onto one screen. The **Provisioning** tab shows all of the jobs we have invoked to create or change collections with timings and status.

- The **Connections** tab gives a history of all of the last 300 sessions, and we can filter this either by user or by VM.
- The **Database** tab allows us to extract all of the tables that sit in the RD Broker database. This is more useful if we aren't using HA, as by default this is hidden away in the WID; whereas, in *Chapter 4*, *Putting the R in Remote Desktop*, we used SQL Server, which means that this data is more accessible.
- **Events & Traces** allows us to go even deeper into what's going on in the broker. If we press *F1*, we can collect all of the relevant event log entries into one file to examine them. We can invoke tracing with *F4*; this brings up a dialog box to start tracing. We can then run whatever the problem is, and once it has occurred, click on **Repro and Stop**. The files can then be collected from the RDV Diag installation folder, and if we can't figure out what they are telling us, then we can forward these to Microsoft support. The following screenshot displays the diagnostic information for the Fast Pooled Collection:

Property	Value
Name	Fast Pooled Collection
Alias	Fast_Pooled_Coll
Description	
Type	PooledManaged
User Groups	contoso\VDI-Users
Auto User Assignment	False
User has Admin Rights	False
Is Rollback Enabled	True
Is shown in RDWeb	True
Size	2
Percent in Use (%)	100
Prefix	FPC
Domain	Contoso.com
OU	OU=RDS-VDI,DC=contoso,DC=com
HighlyAvailable	False
Export Path	\\RDS-BROKER\RDVirtualDesktopTemplate
Storage	
Type	LocalStorage
Local Location	default
Central Location	
Export Location	\\RDS-BROKER\RDVirtualDesktopTemplate\Fast_Pooled_Coll\IMGS_1
User Disks	
Is Enabled?	True
Path	\\grey\UPD-VDI
Maximum Size (GB)	20
IncludeFolderPath	
IncludeFilePath	
ExcludeFolderPath	
ExcludeFilePath	
Virtual Desktop Template	
Name	RDS-VDITemplate
Host	Grey.contoso.com
Provisioning	
Status	JOB_COMPLETED
Start Time	16/03/2014 07:28:57
Last Modified Time	16/03/2014 07:37:09
Total VMs	2
Percent Completed	100%
Failed VMs	0
Job Deadline	

Diagnostic information for the Fast Pooled Collection

Microsoft System Center

Microsoft's System Center suite is a separately licensed solution for managing all aspects of the IT infrastructure. There are a number of components in the suite that used to be separate products, and while none of them are explicitly for VDI, they can add some useful capabilities that aren't present in Server Manger and other free tools that we have looked at so far. Each of the System Center components would warrant a separate book, so I just want to give you a taste of the parts of System Center that are relevant to VDI and cover the basics of what they are for and how they work.

Configuration Manager

For larger VDI deployments, we may want to consider managing our virtual desktops with the same tool used in many businesses for managing real desktops, System Center Configuration Manager 2012 R2 (CM12). This builds on the capabilities of WSUS as well as MDT, which we saw in the *Microsoft deployment tools* section of *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*. This makes patching and deployment of our virtual desktop easier to set up and will also manage our role servers and hosts as well. It also includes **System Center Endpoint Protection** (**SCEP**) with its management agent, as an anti-malware tool that has minimal impact on VDI, allowing this to be easily added into our virtual desktop template and for us to keep this up to date. CM12 also minimizes the I/O storm problem by having a built-in randomized delay for any scheduled updates on VMs, and how CM12 works will depend on the type of collection we have. The types of collections are as follows:

- **Personal Collections**: They are treated exactly as though they are real desktops. As they are created, they will be registered in CM12, which will install the agent on the desktop if it's not there already. Given that the CM12 agent includes SCEP, we will probably create a template for all our virtual desktops (be they pooled or personal), which will have this included already. In fact, if we have CM12, we could and should use it to build the virtual desktop template for us rather than the hand-crafted work we did in the *Building a new Virtual Desktop Template with MDT* section of *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*. CM12 actually has a built-in mechanism to create a VHD, but as with MDT, we will want a sysprepped VHD as the end result so that we can simply import what we did in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, into CM12.
- **Pooled Collections**: They will confuse CM12 because if we have a process to continually recreate our Collection to keep it up to date, possibly by using CM12's deployment tools, then the new desktops will be seen by CM12 as being completely new. So, Microsoft's advice is to exclude them in any inventory task.
- **Session-Based Collections**: They will not show up at all; CM12 will just see the session host as one server, and won't be aware of users' sessions.

CM12 also has Remote Assistance built in and allows for much easier management of the Group Policy we'll need.

Operations Manager

Possibly the most widely known out of the System Center suite, Operations Manager 2012 R2 (OM12) is how we can get a deep insight into the health of any part of our data center, including the fabric (switches, storage, and servers) in far more detail than in Server Manager. Along with gathering data, there are extensive tools to show health and report warnings and alerts, as well as customizable dashboards. The key to the Operations Manager's flexibility is the use of free and paid **Management Packs** (**MPs**). Each Management Pack provides monitoring and alerting for a given part of the infrastructure; this might be hardware, the operating system, or applications from Microsoft and third parties. Many of these are free and are provided by Microsoft; others are paid for and there are tools to author our own if we want to. While Management Packs are great, some of them can be very noisy because they give us far too much information, so it's important to tune and deploy them one by one so that real problems are not masked by a lot of irrelevant information.

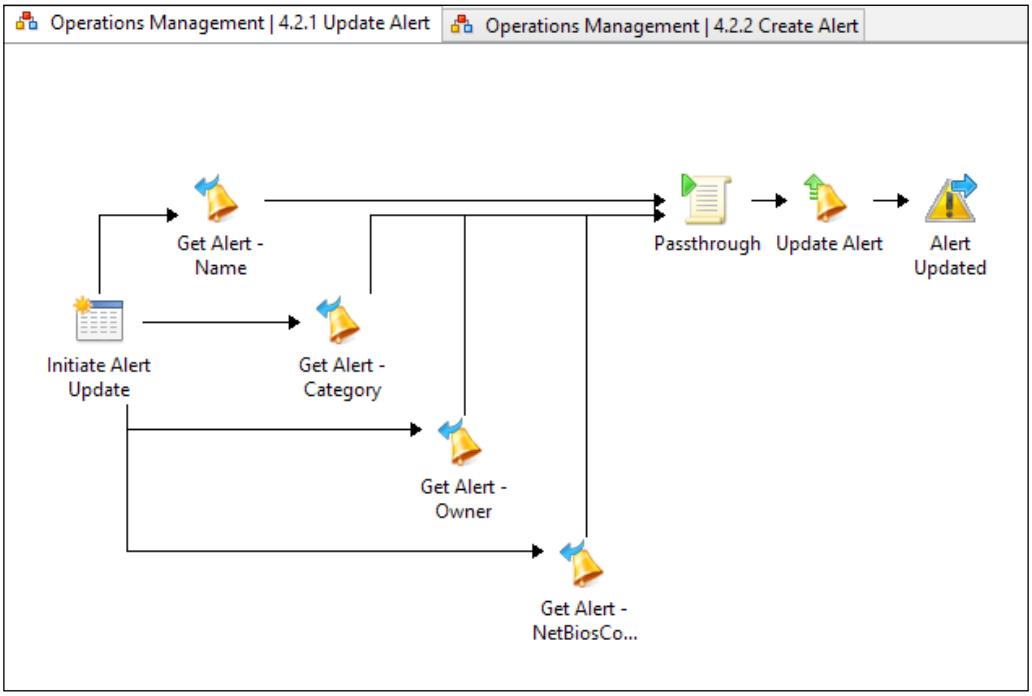
The best place to start with Management Packs is to refer to this series of articles on TechNet at `http://technet.microsoft.com/en-us/library/hh212709.aspx`. Management Packs for all versions of Windows Server can be downloaded from `http://www.microsoft.com/en-us/download/details.aspx%3Fid%3D9296`. Third-party packs are on Microsoft System Center Marketplace at `http://systemcenter.pinpoint.microsoft.com/en-US/home`.

Downloaded Management Packs are normally sealed (read-only), and rather than unsealing them, we can possibly break them or have our changes overwritten by a newer version we download. It is best practice to put any changes (overrides) into our own Management Pack along with other settings, group reports, and so on. There is a specific Management Pack for Remote Desktop Services (`http://1drv.ms/1f6BSjg`). This gives really good insight to the health of the all the server roles in our deployment, and so if OM12 is deployed and there are in-house skills to configure it for RDS monitoring, then it will definitely be of use, but it's probably not worth investing in if no other part of System Center is in use elsewhere.

Like other parts of System Center, OM12 uses an agent on each of our servers. Once we have this deployed, the server we have under monitoring will be automatically added to the folder structure within OM12 in much the same way that Server Manager knows that a server is running RDS or Hyper-V, and places it in the appropriate grouping.

Orchestrator

Orchestrator is best described as PowerShell meets Visio, as you can see in the following screenshot:



An Orchestrator runbook to update an Operations Manager alert

It allows us to use graphical tools to represent a process (runbooks) that can reach out to any other management tools we might have in our data center through an extensive set of integration packs. This integration extends from Microsoft products such as Exchange SharePoint, Azure, and so on to non-Microsoft solutions such as VMware, Cisco, HP, and IBM (the complete list is at `http://technet.microsoft.com/en-us/library/hh295851.aspx`). There aren't any special integration packs or sample runbooks for RDS, so we would have to start from scratch. But, if we find ourselves doing the same maintenance task on our VDI deployments more than once a week, then it should be possible to automate this. For example, if we have a lot of temporary employee turnaround, say in our call center, we will want to add these users to the right groups so that they have access to VDI and have some sort

of algorithm to work out whether we need to provision or decommission a number of virtual desktops. We can invoke runbooks directly from CM12, so we could come up with a comprehensive patching process that would allow us to recreate or apply patches to our virtual desktops on the back of an approval of updates in CM12, and which will give us logging and error trapping that is not in the scripts in this book.

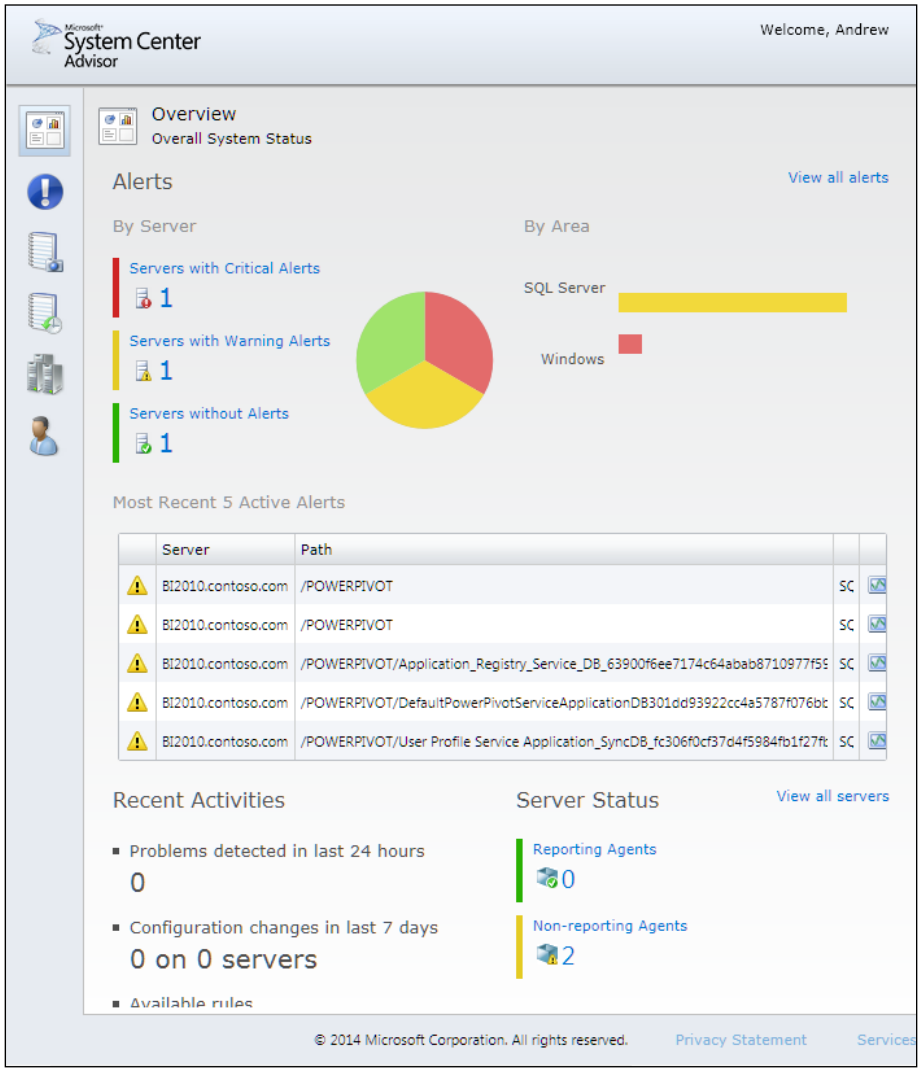
Virtual Machine Manager

Virtual Machine Manager (**VMM**) manages all of the fabric that underpins virtual machines: virtualization hosts, networking, and storage; however, it is not aware of VDI except that it will see the virtual desktops we create in RDS as virtual machines. So, my recommendation is not to add VDI virtualization hosts into VMM so that our virtual desktop VMs are just managed from Server Manager, PowerShell, and the RDV Diagnostic tools, and thus kept away from VMM administrators.

System Center Advisor

Advisor is the odd man out in the System Center suite, in that it is free to use and is a cloud service that is part of Microsoft's online services (such as Azure One Drive and Office 365). It gathers information from our servers and produces an online dashboard of alerts, errors, and warnings, which is based on Microsoft's current best practices. Advisor works by installing an agent on our servers; actually, it's the Operations Manager agent, which collects data every day and posts it to the Microsoft Advisor service.

There is also the option to use a gateway server so that only one of our servers needs an Internet connection to use the service. The following screenshot displays the System Center Advisor screen:



Advisor doesn't have any specific features for RDS, but will give you advice on AD, Hyper-V, SQL Server (which might be useful for our RD Broker database in HA mode), and general server health. We can also integrate the information from Advisor into Operations Manager so that we have a complete left-to-right view of the health of our servers.

Summary

In this chapter, we have seen how we can use the standard mechanism, WSUS, to keep all of our VDI deployments up to date with the latest patches and updates. We have been able to monitor our deployment with some basic counters and the BPA in Server Manager, and we have a basic understanding of how System Center can help us manage and automate VDI. We also saw how the VDT enabled us to get a deeper insight into problems that might occur. Now that we have all the foundations in place for a secure and reliable VDI, we need to turn our attention to managing our users, and in the next chapter, we'll see how we can manage them and their profiles and settings, whether they are just using VDI or a mix of VDI and real desktops.

8
Managing User Profiles and Data

In this chapter, we will look at how to manage our users' profiles and settings so that they get a consistent experience in VDI, which we may also want to be in sync with to get a look and feel of their physical desktops. There are a number of ways of handling this depending on what types of desktops our users will connect to. We'll look at all of the various options and the considerations for using each of them and then move on to examining how to implement each of them. We will also cover the newest technology for managing user profiles: **User Environment Virtualization** (**UE-V**).

Background and options

When a user logs on to Windows, all of their settings are written into a collection of files, collectively known as their **profile**. A user's profile will contain a record of how they have configured their desktop and certain applications, such as the following:

- Desktop shortcuts and display settings.
- Internet settings, favorites, history cookies, as well as cached sites.
- Applications settings that include any customization that are written in the user's profile folder structure. A good example is Office, where templates, menu options, styles, and so on are stored on a per-user basis.
- The various `My*` folders (`My Documents`, `My Pictures`, and so on) with our users' work.
- OneDrive for personal or business use (part of Office 365), if that is being used.

Leaving all of the mentioned components on a given desktop makes it difficult to manage and puts the data at risk as the OS and files can get damaged or corrupted. Hence, IT managers have tried to store key parts of this data on servers. Back in the days of Windows XP, this was a tough process both to implement and for users as it delayed logon and logoff times and was often responsible for the very problem we were trying to avoid: corrupting our users' profiles. The two main techniques we used were as follows:

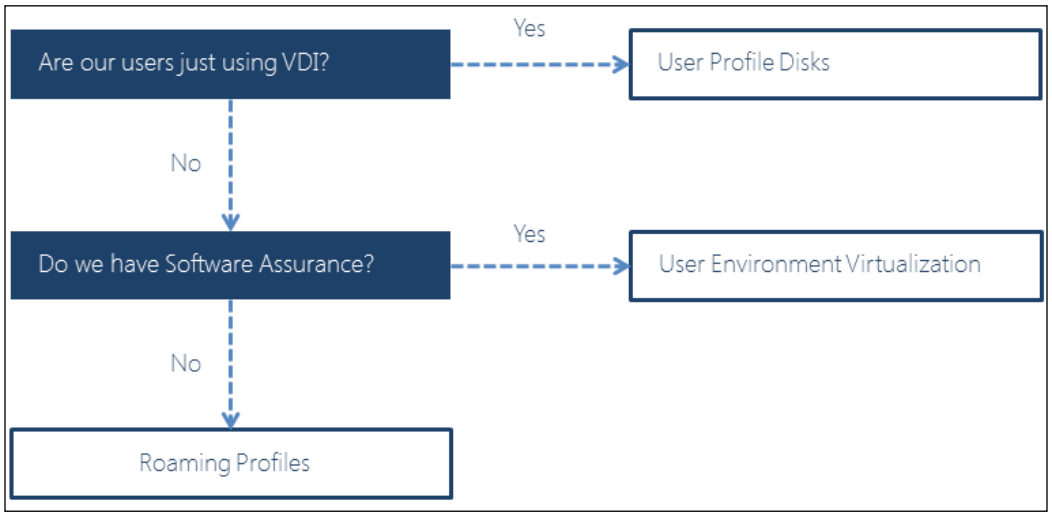
- **Roaming Profiles**: Here, a user's settings were stored on a server and downloaded to whichever machine they were using, and then any changes were written back to the server when they logged off
- **Folder Redirection**: This set users' folders to point to file server shares rather than the local `C:`, for example, `My Documents`

For virtual desktop users, this was OK, provided that users only used the old terminal services, but if they did need to log in to a physical desktop as well, then they could also wait ages to log in and log off as their profile was copied between the local machine and the central server storing the profiles. There was also a feature called **Offline Files**, where local copies of shared files were continually synchronized with a central share to reduce the big hit on I/O during logoff and logon. This feature also made key files available for use when a server connection wasn't available. Later versions of Windows have refined these tools and improvements to storage traffic (for example, Server Message Block Version 3) have made this more efficient. However, as well as making things easier, there are also additional considerations in Windows 8, specifically the modern Start screen and the modern applications that run on it that are available from the Microsoft Store. So we need to understand what are our options and what they are for. One option to simplify the management of the profile is to simply enforce a **mandatory profile** with a standard look and feel for all our users, which they cannot change. This might be suitable for kiosks in shops and on a factory floor, and also for users who are only allowed to perform a specific set of tasks.

For those organizations with Microsoft **Software Assurance** (**SA**), there is an additional set of free tools collectively known as **Microsoft Desktop Optimization Pack** (**MDOP**), and one of these is **UE-V**. These tools are all there for use with physical desktops; for virtual desktops, there is another approach we can use, **User Profile Disks** (**UPDs**), which we briefly saw back in *Chapter 2*, *Designing a Virtual Desktop Infrastructure*. UPDs are simple to set up and work with, so why wouldn't we just use this option for VDI? The problem with UPDs is that they are bound to a specific collection, so if all our users ever do is use a virtual desktop from a single collection, then this would be sufficient. However, if users use different collections as

well as physical desktops, then we need a separate mechanism, and the techniques for managing profiles across multiple physical desktops need to be used in conjunction with VDI. The best mechanism for this is UE-V, so we should use that if we are licensed for it. UE-V handles profiles very effectively across the network and has the ability to learn about how applications store settings and capture those very efficiently along with users' Windows settings. If this isn't an option, we can only make use of the built-in options for Roaming Profiles and Folder Redirection. In all cases, we can use Folder Redirection and Offline Files to ensure the users' work is stored centrally.

The following figure will help us in selecting the tool to be used for profile management:

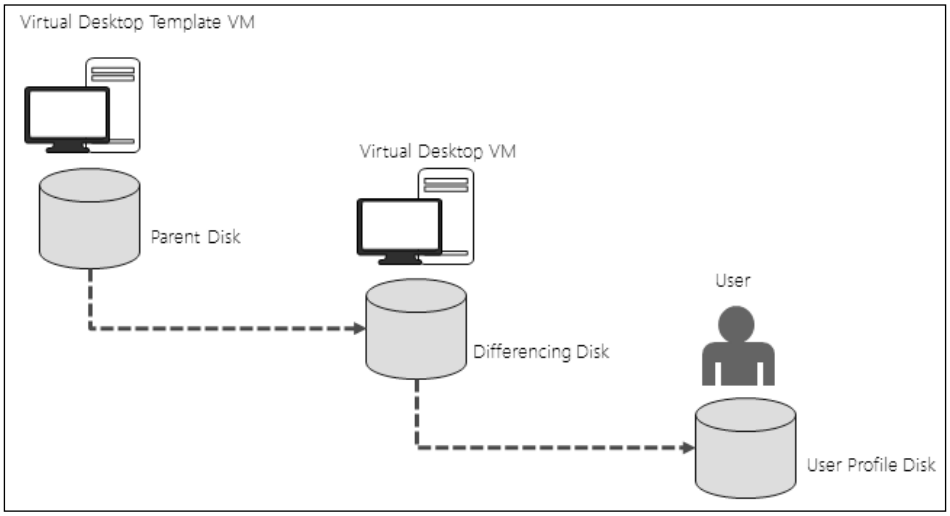


Now that we understand when to use each of these, let's look at how they work in detail.

User Profile Disks

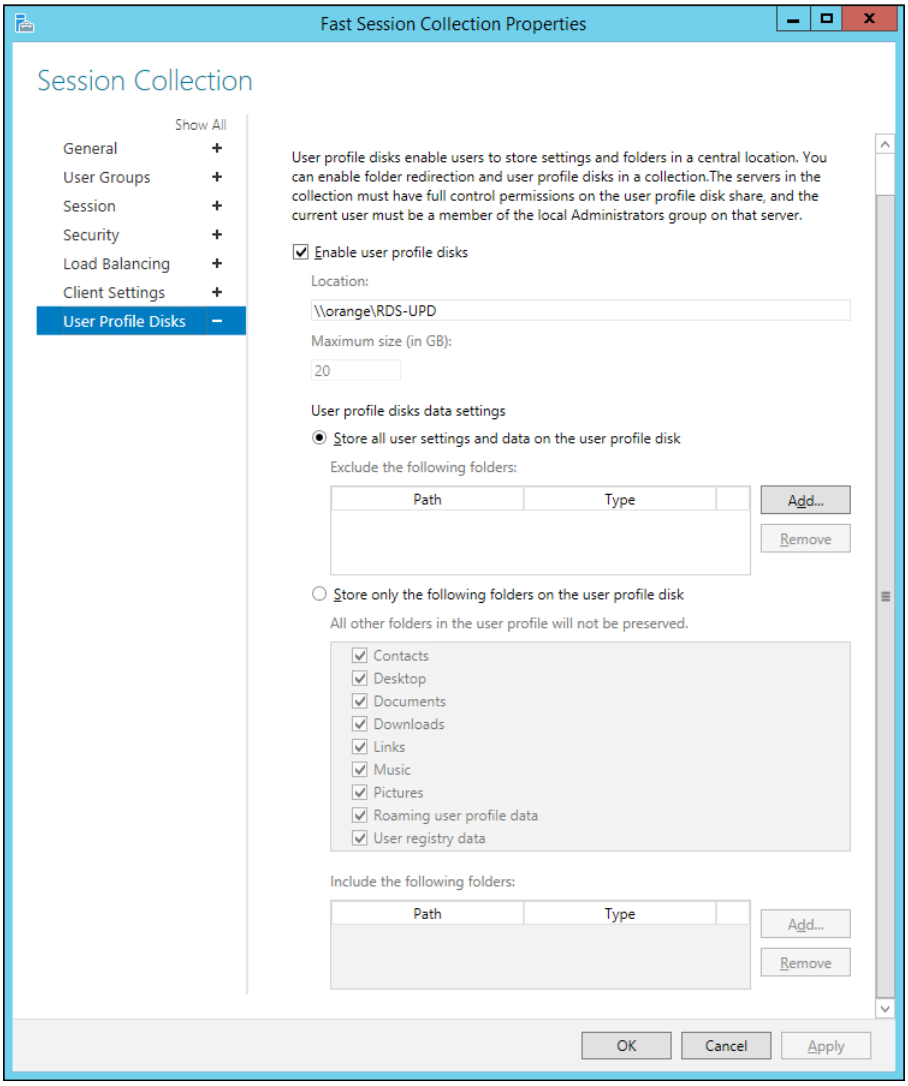
In *Chapter 1*, *Putting the V in VDI – An Introduction to Virtualization in Hyper-V*, we saw that Hyper-V uses three kinds of **Virtual Hard Disks** (**VHD**): Fixed, Dynamic, and Differencing. When we create a personal or pooled collection from a virtual desktop template, the VHD associated with the template is copied to a destination we specify, for example, to an SSD-backed volume on a file server, and a differencing disk is then created from that for each **Virtual Desktop** (**VM**) in the collection.

If we then decide to enable UPDs, then each user gets a new differencing disk based on the VM differencing disk and this works because each of the VM differencing disks are identical. This is illustrated in the following figure:



Normally, it is not possible to break a chain of differencing disks, but UPDs allow this, and this means that a particular user's UPD will get attached to whichever pooled desktop VM he/she is assigned to by the broker. It allows us to slide in an updated parent disk when we recreate a collection to update it. UPDs are also used for Session Virtualization, but in this case, there is simply a separate dynamic VHD for each session user that is simply mounted on `C:` when they log in and detached when they log out. This is a simple matter as Windows Server 2012 allows us to mount VHD by just clicking on it; however, by default, it will be assigned a drive letter. Because both types of UPDs are dynamic or thin-provisioned, they only occupy the space that would be needed to store profile settings.

By default, all of our users' settings get stored in their UPD. We can override this behavior in the collection properties if we want to by excluding some folders and adding in others that our users might need, as shown in the following screenshot:



The User Profile Disks section in the properties of a collection

We can also limit the maximum size of the UPDs and hopefully encourage our users to think about what they need to store. To see how these disks work in our pooled collection, perform the following steps:

1. Log in to our RDS Web portal (`https://rds.contoso.com/rdweb`) as `contoso\RDSUser2` (password is `Passw0rd!`) and then select the **Fast Pooled** collection.
2. Once we have signed in, move some of the tiles around on the Start screen, change the background color of the desktop, create a new WordPad file on the desktop, and rename it `RDSUser1`.
3. Log out of the desktop and then sign in again to check whether the settings have been preserved.
4. Repeat all of this for `contoso\RDSUser2`.

If we have a look at the current state of one of our VMs while a user is logged on (go to **Hyper-V Manager**, right-click on it, and select **Settings**), we can see that Hyper-V has registered the UPD as the system disk of the VM; when we sign out, this will revert back to the differencing disk for that VM (**FPC1** or **2**, followed by a GUID). We can also go to our physical host and mount one of the UPDs by simply right-clicking on it. We'll get a new drive (say `S:`) and, if we explore it, we'll see exactly the same structure as we would if we looked at a user's personal folder.

If you have created the documents detailed in the previous steps, then they'll be visible in the desktop folder.

Using the built-in tools in Windows for managing the users' settings

If our users need to use physical desktops as well as VDI as part of their role, we have to resort to using the same set of tools that the desktop team has traditionally used to manage those physical desktops. This includes controlling the profile itself and using Offline Files and Folder Redirection to keep our users' data on our central file servers. This is largely controlled through Group Policy, and so we don't have to make any changes to our virtual desktop templates to accommodate this. If you have used these tools on previous versions of Windows, you may note that there are three new enhancements to Windows 8/ Server 2012:

- **The Always Offline mode**: If we configure folders using the Always Offline mode, all the files are cached to the local disk even on high-speed internal networks. Changes are periodically written back to the central file servers using more Group Policy settings.

- **Cost-aware synchronization**: Windows 8 knows when our users are connecting over a metered network, and if we configure this, then our users will automatically be in the offline mode on these metered networks to avoid incurring excessive data charges.
- **Primary Computer Support**: This allows us to restrict folder redirection to only a user's primary devices, such as their main laptop or a VDI collection. If they use anything else, then they won't get their profile. A user's primary computers are tagged in AD either through the UI or with PowerShell.

Enabling Roaming Profiles

The basic principles of enabling our users' profiles to move from machine to machine have not really changed since Windows XP.

Best practices, such as minimizing what's in the profile and not using EFS, for Roaming Profiles may still be relevant and can be found at `http://technet.microsoft.com/en-us/library/cc784484(v=WS.10).aspx`.

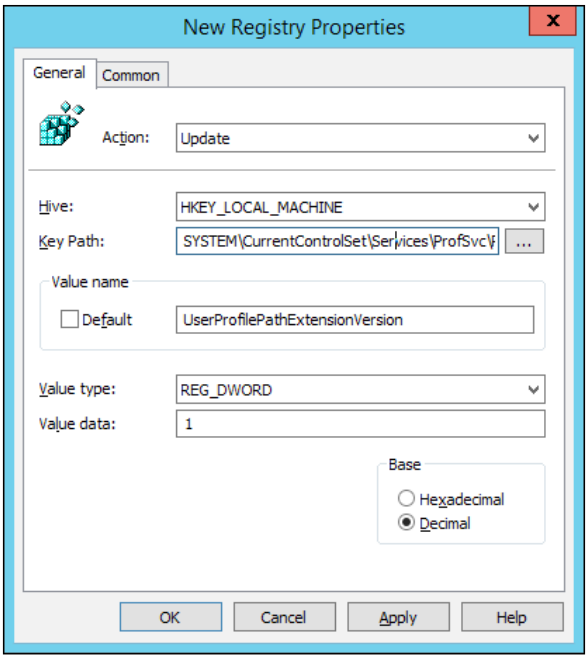
There are one or two things that have changed and there are also some special considerations for VDI. If we want to use Roaming Profiles for users accessing our Virtual Desktops, we could enforce this on our desktops rather than per user, which was Microsoft's best practice. However, if our users are exclusively using VDI, then we can just implement UPDs and not worry about Roaming Profiles and so on. So what we need to do is set up Roaming Profiles in Active Directory for these users. We also need to be aware that if we are using session-based virtualization and then allowing our users to also use a real or virtual desktop based on Windows 8.1, then our users' profiles are moving between a Server OS and the Client OS. To make that work, we need to enable different profile versions, and this requires a Microsoft **Knowledge Base** (**KB**) update, for example, **KB 2887595** (`http://support.microsoft.com/kb/2887595`), and a new registry key. These changes need to be made to our session hosts and our VDI template, and the good news is we can take advantage of WSUS, which we set up in *Chapter 7*, *Maintenance and Monitoring*. We can do this by doing the following:

1. Connect to **RDS-DC** and from **Server Manager**, navigate to **Tools | Windows Server Update Services**.
2. In the **Update Services** console, right-click on **Update Services** and select **Connect to Server**. Enter `RDS-Ops.contoso.com` and click on **OK**.
3. Expand **RDS-Ops.contoso.com** and navigate to **Updates**.
4. Right-click on **Critical updates** and select **Search**.

5. Enter `2887595` and click on **Find Now**, which should return three entries: the ones we need to approve for Windows 8.1. Right-click on each one and select **Approve**. Approve the update for **All computers** as our VDI desktops aren't in a WSUS-targeted group.
6. Now we will just need to recreate our VDI collection using the RDS-Ref script we used in *Chapter 7*, *Maintenance and Monitoring*.

We also need to alter the registry of our session hosts and virtual desktops to add in a new entry. Rather than doing this on each machine, we can use Group Policy. We'll edit the default domain policy, using the following steps, as this will cover our virtual desktops and hosts:

1. Connect to **RDS-DC** and in **Server Manager**, navigate to **Tools** | **Group Policy Management**.
2. Navigate to **contoso.com**, right-click on **Default Domain Policy**, and select **Edit**.
3. In the **Group Policy Management Editor** screen, navigate to **Default Domain Policy** | **Computer Configuration** | **Preferences** | **Windows Settings**.
4. Right-click on **Registry** and go to **New** | **Registry Item**. Leave the value of **Hive** as is, and in the **Key path**, navigate to **My Computer** | **HKEY_LOCAL_MACHINE** | **SYSTEM** | **CurrentControlSet** | **Services** | **ProfSvc** | **Parameters** and then click on **Select**.
5. Set the value of **Value name** to `UseProfilePathExtensionVersion`, **Value type** to `REG-DWORD`, and **Value data** to `1`. Click on **OK** to create the new registry item, as shown in the following screenshot:



This is a good example of how we can manage our VDI deployments without having to dive into each one as we need to make changes and apply updates. However, we now need to get on with the business of creating our Roaming Profiles. We can do this by doing the following:

1. Create a security group in AD for Folder Redirection and so on, and assign this group to our roaming users.
2. Create a file share(s) on a Windows Server 2012 or R2 server for our users' profiles and folders (`My Documents` and so on). In order to properly manage the share, we should enable the File Server Resource Manager to limit what our users can store and to retrieve quotas on how much they can store.
3. Use Active Directory to set up Roaming Profiles.

Creating the Security Group

We can create a new Security Group in our lab environment and just add in one user, **contoso\RDSUser2**, as they already have access to both our session and pooled collections and we can see their settings roam across these. Perform the following steps to create Security Groups:

1. Connect to **RDS-DC** and in **Server Manager**, navigate to **Tools** | **Active Directory Administrative Center** from the menu.
2. Right-click on **Contoso** and go to **New** | **Group**.
3. Name the group `Roaming-Users`.
4. Scroll down to the **Members** section and click on **Add**.
5. Type in `RDS User2`, click on **Check Names**, and click on **OK**.
6. Click on **OK** at the bottom of the screen to create the group.
7. The PowerShell History window will show you the following scripts to perform the preceding steps in PowerShell:

```
New-ADGroup -GroupCategory:"Security" -GroupScope:"Global"
-Name:"Roaming-Users" -Path:"DC=Contoso,DC=com"
-SamAccountName:"Roaming-Users" -Server:"RDS-DC.Contoso.com"
Set-ADGroup -Add:@{'Member'="CN=RDS User2,OU=RDS-
VDI,DC=Contoso,DC=com"} -Identity:"CN=Roaming-
Users,DC=Contoso,DC=com" -Server:"RDS-DC.Contoso.com"
```

Creating the file share

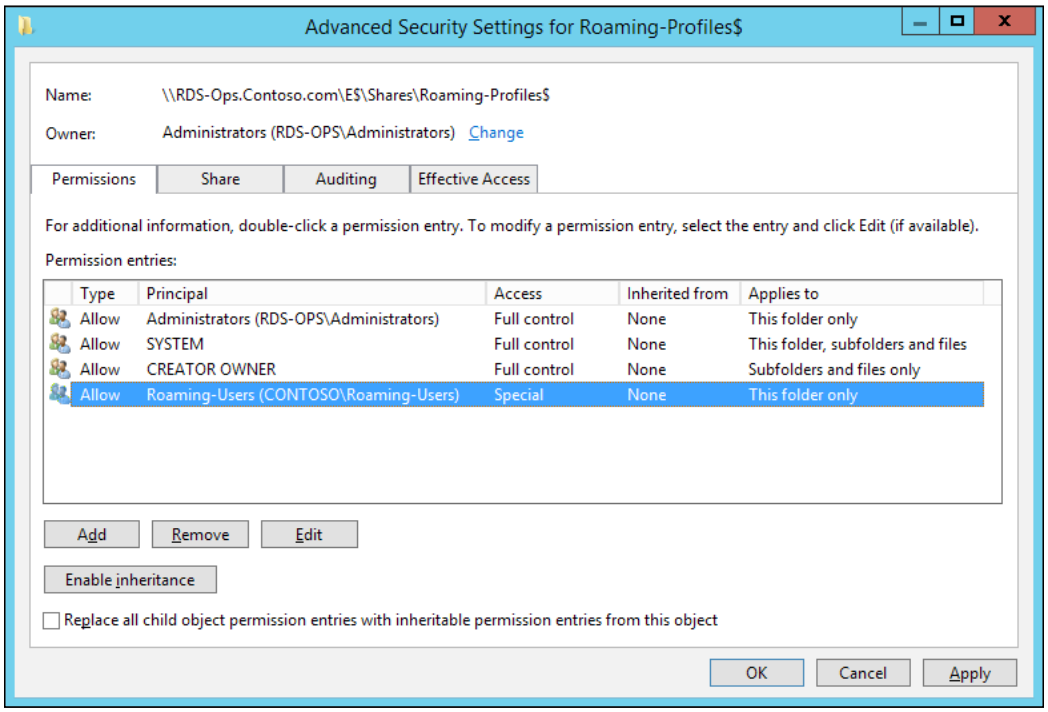
We'll use RDS-Ops for our file share as it has a number of shares on it already and is enabled for deduplication, which means we'll get good disk savings (70 percent) by storing users' data on that. We might also want to use the **File Server Resource Manager** (**FSRM**) to control quotas on our users' files and profiles. This is simply a Role Service (an optional part of the File Server role) that we can turn on by navigating to **Server Manager** | **Add Roles and features**, as we have done several times before, or with the following line of PowerShell command:

```
Add-WindowsFeature –ComputerName RDS-Ops FS-Resource-Manager –
IncludeManagementTools
```

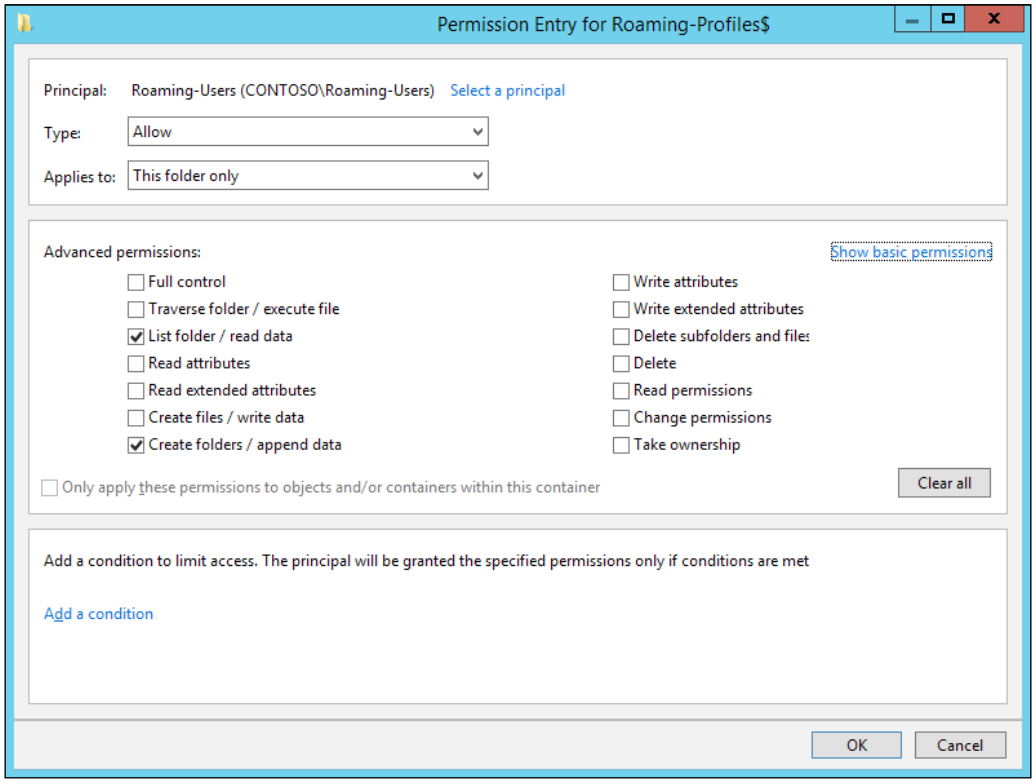
Now, since we have enabled FSRM, we can create the file share using the following steps:

1. We can go back to **Server Manager** and navigate to **File and Storage Services** | **Shares**.
2. Right-click on **RDS-OPs** and select **New Share** to open the **New Share** wizard.

3. In the **Select the profile for this share** screen, select **SMB Share – Advanced** and click on **Next**.
4. In the **Select the server and path for this share** screen, select **RDS-Ops** as the **Server**, then select the **E:** volume as the **Share location**, and click on **Next**.
5. In the **Specify share name** screen, we'll call the share **Roaming-Profiles$** (the $ suffix hides the share from casual browsing).
6. In the **Configure share settings** screen, we can select various advanced sharing options, and certainly, access-based enumeration should be enabled so our users can't see what they don't have access to. Encryption might be important, and **Allow caching of share** should be enabled so that we can enable offline files on the users' PCs if not on our virtual desktops. Click on **Next** to make the selections.
7. In the **Specify permission to control access** screen, we need to allow our users access to just their personal folders, not the whole share. Hence, select **Customize Permissions**.
8. Click on **Disable inheritance**, and then click on **Convert inherited permissions into explicit permission on this object**.
9. Set the permissions as shown in the following screenshot:



The specific special permissions we need to set for our **Roaming-Users** group are the advanced permissions **List folder/read data** and **Create folders/ append data**, which should be applied to this folder only, as shown in the following screenshot:



10. Click on **OK** to apply the permissions and on **Next** to move on to the next screen in the **New File Share** wizard.
11. In the **Specify folder management properties** screen, we can set that this folder is for user files and click on **Next**.
12. In the **Apply a quota to a folder or volume** screen, we can use the capabilities of FSRM to set limits on how much data our users can use, and we'll use the template **Monitor 500MB share**. Click on **Next**.
13. Click on **Create** to enable the new file share.

Using Active Directory to enable Roaming Profiles

We'll just enable Roaming Profiles for one user, **RDS User2**, to see how it works. We can do this by doing the following:

1. On **RDS-DC** in **Server Manager**, navigate to **Tools** | **Active Directory Administration Center**.
2. In the **Global Search** option enter `RDS`. Double-click on **RDS User2**.
3. In the properties for **RDS User2**, scroll down or navigate to the **Profile** section and set the value of **Profile path** to `\\RDS-Ops\Roaming_Profiles$\%username%`, where `%username%` will automatically replace the username when the user signs in.
4. Some applications use the **Home** setting, and we can set that to `\\rds-ops\roaming-Profiles$`, where the username will be appended during login.

We can now test this by connecting to the RD Web portal (`http://rds/rdweb`) and logging in to our fast-pooled collection and then logging out of that and into our fast session collection. You should see that our user **RDS-User2** now has a folder in our Roaming Profiles share and the only way to get to that is by being signed in as this user and navigating to `\\RDS-Ops\Roaming-Profiles$\RDS-User.V2`. Note the `V2` suffix; it is the direct result of the version changes we made earlier.

Super-mandatory profiles

In some organizations, we can get away with everyone having the same profile and so any change they make are reset when they log in again by using mandatory profiles, and if this is not available, they'll just be given a temporary profile for that session. There is also a stronger option of **super-mandatory profiles**, and if a user can't get this profile, they will not be able to log in at all. If we want to do this for our VDI users, we need to do the following:

1. Create a clean Windows 8.1 VM as we did in *Chapter 2*, *Designing a Virtual Desktop Infrastructure*, and modify the script to create a new VM, for example, **RDS-Profile**.
2. Start the new VM and log in. Configure the appearance icons desktop background as you wish.

3. From **RDS-Ops**, create a new answer file (`unattend.xml`) using the **System Imaging** tool we have already installed as part of the WADK in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, then do the following:
    1. Select the option **Create a new answer file to use with Sysprep**.
    2. Select **File | New Answer File**. An empty answer file appears in the **Answer File** pane.
    3. In the **Windows Image** pane, expand **Components**, right-click on **amd64_Microsoft-Windows-Shell-Setup**, and then click on **Add Setting to Pass 4 specialize**.
    4. In the **Answer File** pane, select `amd64-Microsoft-Windows-Shell-Setup_neutral` from under the `Components\4_specialize\` folder.
    5. In the **Microsoft-Windows-Shell-Setup Properties** pane, in the **Settings** section, type in the value `CopyProfile = true`.
    6. Save this new answer file to `\\RDS-Profile\C$` (our new VM) and name it `CopyProfile`.
4. Back on our **RDS-Profile** VM, open a command prompt, run `CD C:\Windows\System32\sysprep`, and then run `sysprep /oobe /reboot /generalize /unattend: c:\unattend.xml`.
5. Complete the out-of-box experience, and then log back on to the computer using an account that has local administrator privileges.
6. Right-click on the Windows icon on the Start menu and select **System**.
7. In the **System** screen, select **Advanced system settings**.
8. In the **System Properties** screen, in the **Advanced** tab, click on **Settings** in the **User Profiles** section.
9. Click on **Default Profile** and then click on **Copy To**.
10. In the **Copy To** dialog box, set the **Copy Profile to:** path to `\\RDS-Ops\Roaming-Profiles$\Mandatory.V2`. Under **Permitted to use**, click on **Change**, type in `Everyone`, and then click on **OK**.
11. Click on **OK** to copy the default user profile.
12. Open File Explorer, navigate to `\\RDS-Ops\Roaming-Profiles$\Mandatory.V2`, and rename the hidden system file from `NTUser.dat` to `NTUser.man`.
13. The final step is to go back into Active Directory and set the **Profile path** of a test user (we could use **RDSUser 3**) to point to the mandatory profile we have just created, just as we did to enable Roaming Profiles.

Configuring Folder Redirection and Offline Files

There is no dependency between Folder Redirection and Roaming Profiles – they are two tools to help us do a better job of allowing users to move from desktop to desktop. However, enabling Folder Redirection with Offline Files removes the Big Data transfer that occurs when a user logs on and off because the files the user is working on are continually synced with the local copy and file server. We might want to keep Roaming Profiles away from our user's other folders, and we might also want to apply Folder Redirection to users who don't have Roaming Profiles enabled. I am going to suggest we continue to work on our roaming user (**RDS User2**) in our Roaming Users Group and add in Folder Redirection and Offline Files on a new share to see how this all works with a minimum of extra effort. So we need to create another share in exactly the same way as we just did to create `\\RDS-Ops\Roaming-Profile$`, but this time the share will be called `Redirected-Folders$` on `E:` of **RDS-Ops** and will look like this: `\\RDS-Ops\Redirected-Files$`. We can then edit our Roaming User GPO using the following steps:

1. Right-click on the **Roaming Users GPO** and click on **Link Enabled** to unlink the policy until we have finished editing it.
2. Right-click on **Roaming Users GPO** again and click on **Edit**.
3. In the **Group Policy Management Editor** screen, navigate to **User Configuration** | **Policies** | **Windows Settings** | **Folder Redirection**.
4. Right-click on **Documents** and select **Properties**.
5. In the **Properties** dialog box, select **Basic - Redirect everyone's folder to the same location**. In the **Target folder location** section, select **Create a folder for each user under the root path**, and for the **Root Path**, enter `\\RDS-Ops\Redirect-Files$`, which will automatically create a folder structure for each user so **RDS User2** will get to store their documents on `\\RDS-Ops\Redirected-Files$\RDSUser2\Documents`.
6. Click on the **Settings** tab of the GPO, and in the **Policy Removal** section, click on **Redirect the folder back to the local userprofile location when the policy is removed**. This means that if we remove this GPO, the users' profiles will revert to their original settings.
7. Close the **Group Policy Management Editor** window.

In a production VDI deployment, we would only allow our users' sessions to persist for personal collections, and even then we might decide not to do that. This means that if we aren't using UPDs in our collections, we will need to repeat the last six steps for all the users' folders and settings we want them to keep between sessions.

8. To enable our new GPO, right-click on **Roaming Users** and select **Link Enabled**.
9. The quick and dirty way to apply the GPO in our lab is to restart the relevant VMs; otherwise, we will need to connect to each one (**RDS-host**, **RDS-Host2**, and the VMs in our pooled collection).
10. In the **Select User**, **Computer**, or **Group** dialog box, type in `Roaming-Users`, click on **Check Names**, and then on **OK** to add the group.

Once we test this is working ok, then we can move on to enable one of the new features of Folder Redirection in Windows Server 2012—Always Offline—for our users by performing the following steps:

1. Right-click on the **Roaming Users** GPO and click on **Link Enabled** to unlink the policy until we have finished editing.
2. Right-click on the **Roaming Users** GPO again and click on **Edit**.
3. In the **Group Policy Management Editor** screen, navigate to **Computer Configuration** | **Policies** | **Administrative Templates** | **Network** | **Offline Files**.
4. Right-click on **Configure slow-link mode** and select **Edit**.
5. In the **Configure slow-link** mode, click on **Enabled**.
6. In the **Options** box, click on **Show**.
7. In the **Show Contents** screen, specify `\\RDS-Ops\Redirected-Files$` to enable the Always Offline mode on our user's redirected folders. In the **Value** box, type in `Latency=1` to set the latency threshold to one millisecond and then click on **OK**.

User Environment Virtualization

UE-V is only available as part of the **Microsoft Desktop Optimization Pack**, which is free for customers with Software Assurance. The download to evaluate and test it is not freely available, so you'll either need access to the SA resources or an MSDN subscription (`http://msdn.microsoft.com/en-us/subscriptions`). UE-V is unique among the tools that we have looked at so far, in that it is designed around Windows 8, so it's aware of managing the new user settings and the modern apps

that form the tiles in the Windows 8 start screen. It also works in a very different way to Roaming Profiles as it is application aware, and when an application like PowerPoint launches, it will update the user's profile (registry settings and so on) based on what is stored centrally. This avoids the following problems that could be encountered by our users:

- The settings needs are only downloaded for an application or a part of the OS when a user opens it, which means the big hit of pulling down a profile is avoided.
- Switching between Session-based desktops and Windows desktops is handled better.
- UE-V is designed to work with **Application Virtualization** (**App-V**), another part of MDOP that allows applications to be separated from the underlying OS in the same way as Hyper-V separates the OS itself for the physical hardware. The use of App-V means that we don't have to build Virtual Desktops with applications in them, as we'll see in the next chapter, but for now, we just need to know that UE-V works with installed and virtualized applications.

By default, UE-V understands how common applications such as Office work through the use of a set of XML-based templates, and we can add our own template or override what is included with UE-V. There is a Microsoft-hosted gallery of IE-V templates at `http://gallery.technet.microsoft.com/site/search?f%5B0%5D.Type=RootCategory&f%5B0%5D.Value=UE-V&f%5B0%5D.Text=UE-V` that includes Microsoft products such as Office and third-party software from vendors such as Firefox, Adobe, and Google. The following three mechanisms can be used to manage Windows 8 (as opposed to desktop applications designed for Windows 7):

- We can elect to sync settings or not
- We have a list of how designated Windows 8 apps are managed
- We have a default behavior setting for Windows 8 apps that are not in the list

If we have applications we want to manage in a specific way, we can install them on to a reference computer, use UE-V Generator to monitor the behavior of an application as it launches, and create a template from that. The generator also provides an interface to modify existing templates.

How does all of this work in practice? UE-V creates settings package files within the user profile (`%userprofile%\AppData\Local\Microsoft\UEV\%computername%`) to store the settings for Windows and an application. When a user logs in for the first time, it creates an initial folder that stores the initial state of the settings.

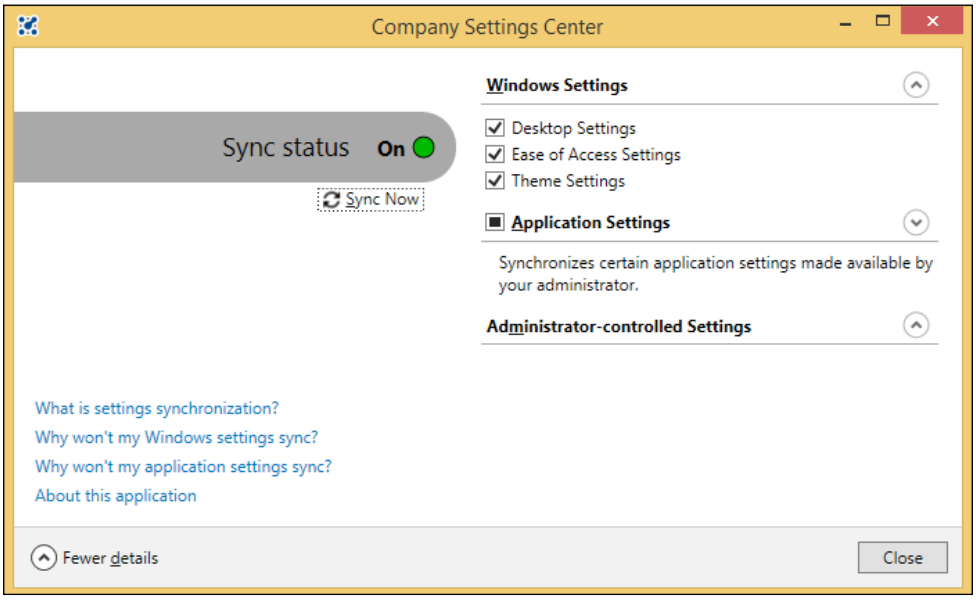
A second folder, `Current`, is also created, which records what has changed. The changes are written into the following two kinds of files:

- **PKGX**: These files contain registry information and metadata about the second kind of file
- **PKGDAT**: These files have settings normally stored in XML or INI files

Each folder for a Windows setting or application will have one or more PKGX files and zero or more PKGDAT files. This folder structure is then kept in sync to a central share and with offline files.

Note that all of the UE-V packages will be hidden, so you'll need to change the **View Files** setting in the explorer to see them. For example, when logged on as RDS User 1 with UE-V enabled, we'll see the following file structure: `C:\Users\RDSUser1\AppData\Local\Microsoft\UEV\LocalSyncFolder\SettingsPackages`.

UE-V has no server console component or user interface, and all we actually need to do to deploy UE-V is install the UE-V agent to each of our virtual desktops. We can then manage UE-V in a number of ways: via Group Policy, via PowerShell, or using WMI (Windows Management Interface). Our users will see that UE-V is working via the **Company Settings Center**, which we can optionally present to them and add in such details of how to contact the helpdesk, as shown:



The Company Settings Center screen that is part of the UE-V agent

Installing UE-V

Installing UE-V for our VDI deployment is fairly straightforward. The following steps illustrate it:

1. Set up a file share for each user's UE-V settings to be stored.
2. Set up a central location for the UE-V templates using another file share (also referred to as the **UE-V template catalog**).
3. Install the UE-V agent to our pooled virtual desktop collection. We can modify the work we did in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, to update our virtual desktop template and then recreate the collection. For our session hosts, we can install the agent directly.
4. Implement Group Policy to configure UE-V. I suggest we create a new GPO linked to a new security group so we can leave the work we have done on Roaming Profiles in place for comparison.

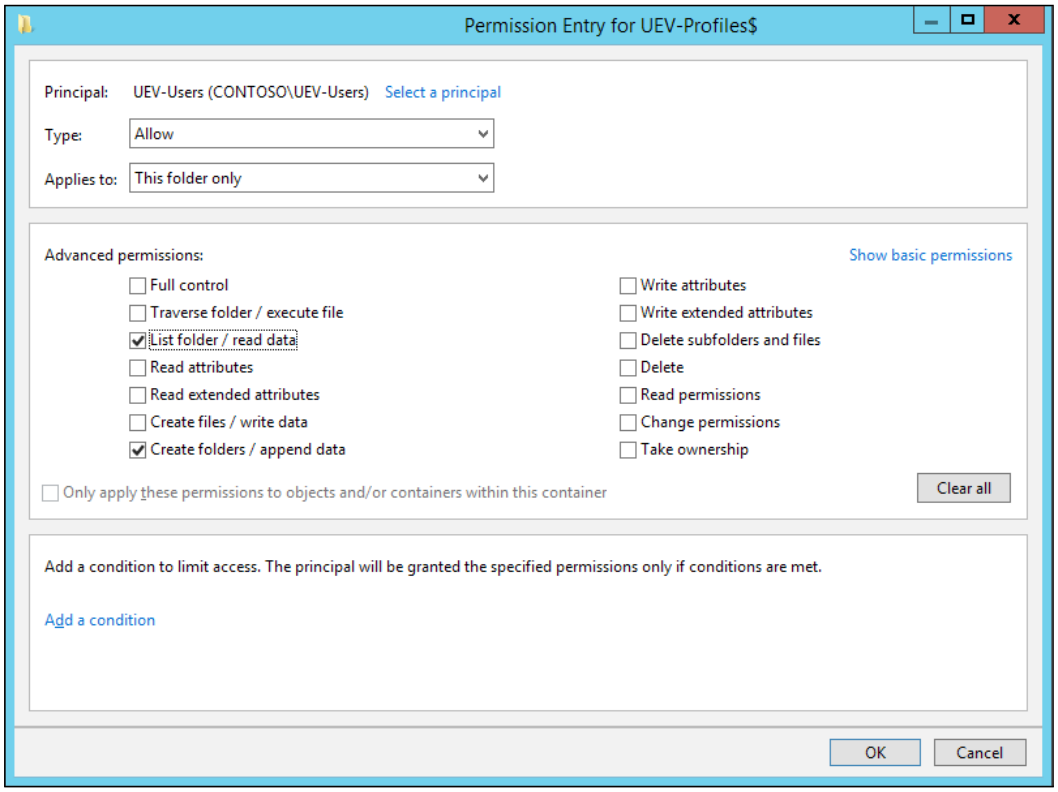
Before we do any of this, we need to have a test user and group in our lab so we can see how UE-V works. We'll create a new group in AD—`UEV-Users`—and then assign RDS User1 to this group; as with RDS User 2, we'll make this user a member of both the Session and VDI users-groups so we can see UE-V working as they change desktops. Hopefully, you now know how to do this in the ADAC; if not, the following is the PowerShell script that will make the necessary changes:

```
New-ADGroup -GroupCategory:"Security" -GroupScope:"Global" -Name:"UEV-
Users" -Path:"DC=Contoso,DC=com" -SamAccountName:"UEV-Users"
-Server:"RDS-DC.Contoso.com"
Set-ADGroup -Add:@{'Member'="CN=RDS User1,OU=RDS-
VDI,DC=Contoso,DC=com"} -Identity:"CN=UEV-Users,DC=Contoso,DC=com"
-Server:"RDS-DC.Contoso.com"
Add-ADPrincipalGroupMembership -Identity:"CN=RDS User1,OU=RDS-
VDI,DC=Contoso,DC=com" -MemberOf:"CN=Session-Users,OU=RDS-
VDI,DC=Contoso,DC=com" -Server:"RDS-DC.Contoso.com"
```

Setting up the file shares for UE-V

We'll use RDS-Ops for the two file shares we need. We can create these in exactly the same way we did for the `Roaming-Profile$` share in the earlier section, *Creating the file share*, except the following points:

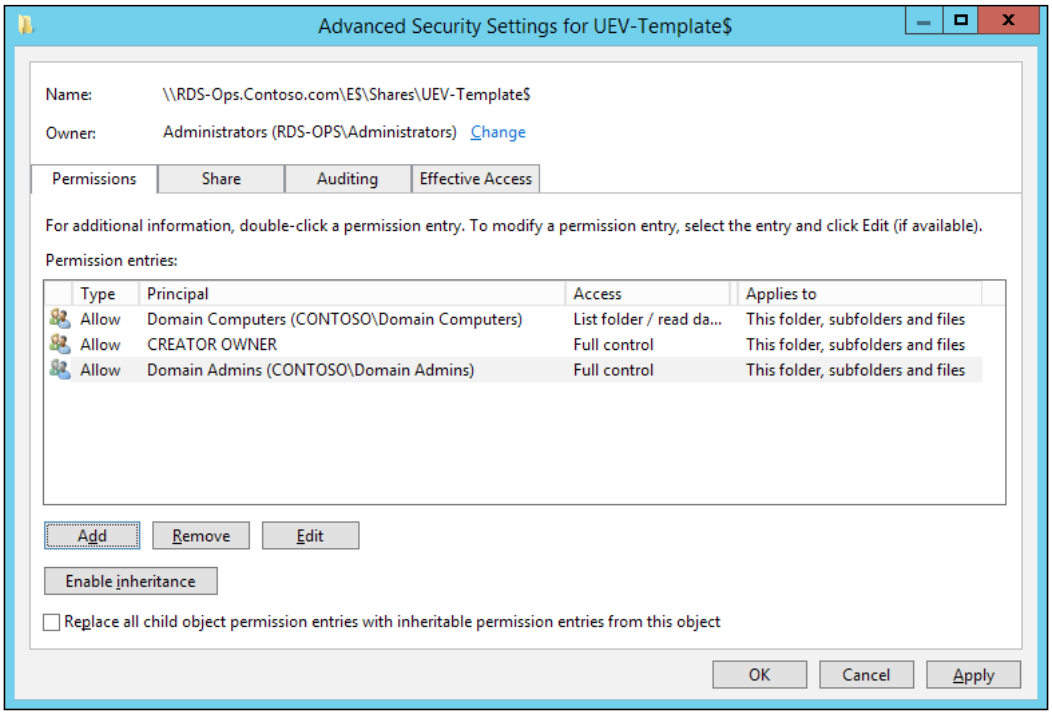
- For the share for the user's settings, we'll just create a Simple-SMB share and we'll call the share `UEV-Profiles$`. And we'll assign the advanced permissions **List folder/read data** and **Create folders/append data** for this folder only to our new AD group UEV-Users, as shown:



- For the share that will store the UE-V template, we'll create a Simple-SMB Share called `UEV-Templates$` with the following NTFS permissions:

User account	Recommended permissions	Apply to
Creator/owner	Full control	This folder, its subfolders, and its files
Domain computers	List folder contents and read	This folder, its subfolders, and its files
Everyone	No permissions	No permissions
Administrators	Full control	This folder, its subfolders, and its files

The share permissions can be left as is. The following screenshot shows the mentioned permissions:



Deploying the UE-V agent

The UE-V download is in the ISO format and contains `.exe` and `.msi` for the agent. We can use the MSI in MDT to modify our task sequence that we created in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, as follows:

1. Download the MDOP ISO to your Hyper-V host.
2. Connect to **RDS-Ops** and in the **Hyper-V** menu for this connection, navigate to **Media | DVD drive** and then navigate to the UE-V ISO (something like `mu_user_experience_virtualization_2.0_x86_x64_dvd_3220555.iso`).
3. As we saw in the *Deploying applications with MDT* section of *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, we need to import the application. Open **Deployment Workbench**, expand **MDT Deployment Share**, right-click on **Applications**, and select **New Application** to launch the **New Application Wizard**.
4. In the **Application Source type** screen, select the **Application with source files** option and click on **Next**.
5. In the **Details** screen, enter the value of **Publisher** as `Microsoft`, **Application name** as `UE-V`, and **Version** as `2.0`. Click on **Next**.
6. In the **Source** screen, locate the folder with the EXE file (which should be `D:`) and click on **Next**.
7. In the **Destination directory** screen, leave the default of **Microsoft UE-V** and click on **Next.**
8. We need to install the agent, and we can optionally set switches to say where the user's settings are to be stored and where the UE-V templates are, but we are going to use Group Policy to set these anyway. This also allows us to make changes and correct errors without the need to reinstall the agent. So all we need specify here is the following:

   ```
   msiexec.exe /i AgentSetupX64.msi /quiet /norestart
   ```

9. Set the working directory to `\\RDS-OPS\DeploymentShare$\\Applications\Microsoft UE-V 2`.
10. Check the settings in the **Summary** screen and click on **Next** to import the application. Review the output and click on **Finish** to close the wizard.
11. Expand the **Applications** directory under **Applications** in the navigation pane to the left and the new application will show up on the center screen. Right-click on it and select **Properties**. In the **General** tab, we can see that the application has a GUID, which we need to know, so copy this to the clipboard. In the **Details** tab, we could set what OS it can be deployed to, and in the **Dependencies** tab, we could set any other applications or fields that this application depends on.

12. Next, we need to customize our Task Sequence to install UE-V. To do this, expand **Task Sequences** in the deployment share and right-click on the **Task Sequence** we already created to set its properties. By default, the type of task sequence we selected earlier already has an application install step included. To find it, expand the **State Restore** folder on the task sequence list and then copy and paste the entry for **Install Foxit Reader**.
13. Edit the second of the **Install Foxit Reader** application copy as follows:
    1. Set the value of **Name** to `Install UE-V`.
    2. Check the option **Install a single application**.
    3. Set the value of **Application to install** to **Microsoft UE-V 2.0** (use **Browse** to find it).
    4. Click on **Apply**.
14. We can then add this application into the **[Default]** section of the rules of our deployment share to install it automatically. The entry should look like the following:

    ```
    Applications002 = <GUID copied from the properties of the
    Microsoft UE-V 2.0 application we just copied>
    ```

15. Now we can recreate our RDS-Ref VM and then use this as a virtual desktop template to recreate our pooled collection, as we did earlier in this chapter, to apply the special update needed for profile versions.

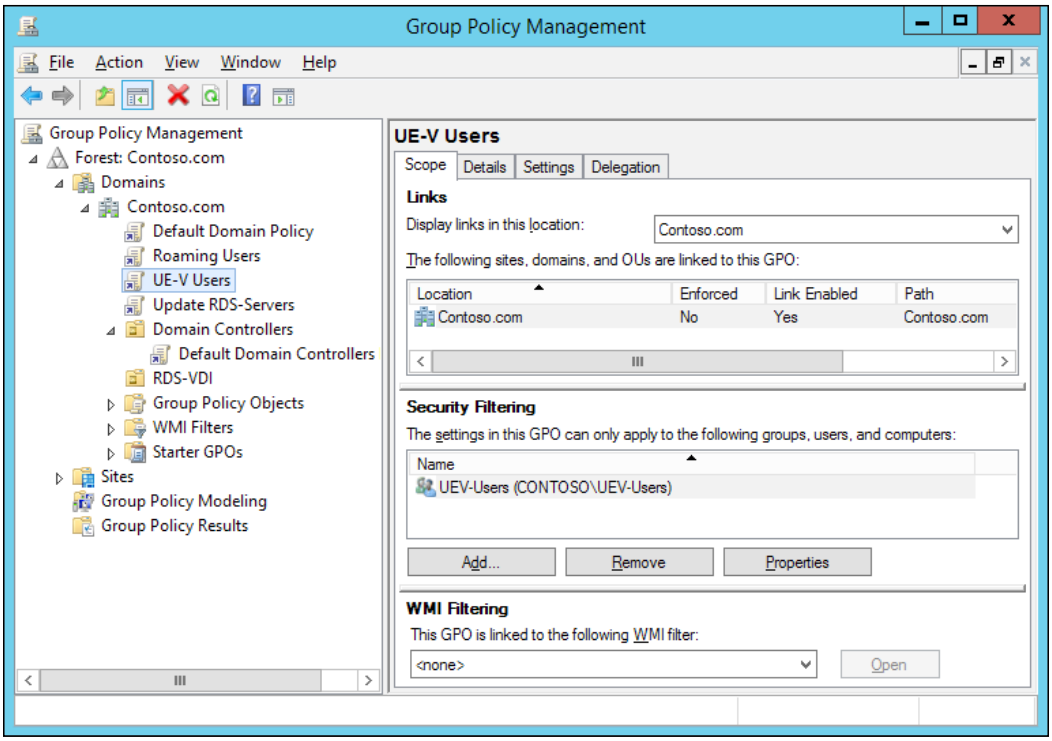
We can also install the agent on our session hosts by mounting the UE-V ISO onto each VM and installing it with the same switches mentioned previously.

In a real production, it is essential to keep all the session hosts that are used to serve out a collection in exactly the same configuration, so where we have used RDS-Host and RDS-Host2 to form a collection, UE-V must be installed on both of them as we have no direct control over which user will use which host.

Using Group Policy to manage UE-V

Because UE-V is not part of Windows Server itself, we need to import a special UE-V administrative template into Group Policy. The download is available from `http://www.microsoft.com/en-us/download/details.aspx?id=41183`, and once we have it, we just need to run it on RDS-DC and accept all the defaults.

When it has finished installing, we can create a new GPO just like we did for Folder Redirection earlier, but this time we'll call the GPO **UE-V** and we'll set the security filtering so that GPO is only applied to our new AD Group **UE-V Users**, as shown in the following screenshot:



We can then right-click **Edit** to edit the GPO and configure UE-V by navigating to **Users** | **Administrative Templates** | **Windows Components** | M**icrosoft User Experience Virtualization**. In the **Group Policy Management Editor** screen, we can review how UE-V works and make some basic changes for our lab:

- **Do not use the Sync Provider**: This will effectively disable UE-V. By default, the Sync Provider is enabled and UE-V will work.
- **Do Not synchronize**: By default, Windows 8 apps are synchronized, so we can turn this off if we need to.
- **Roam Windows Settings**: We'll enable this and check **Desktop Settings**, **Themes**, and **Ease of Access** so that our users get the same desktop look as the switch desktops.
- **Setting Package Size warning threshold**: This allows us to be alerted if a package file (the PKGX and PKGDAT files) size reaches a certain size.

- **Synchronization Timeout**: This is in milliseconds and the default is `20000`.
- **Settings Storage Path**: We need to enable this and set the storage path to `\\RDS-Ops\UEV-Profiles$\%username%`.
- **Use User Experience Virtualization (UE-V)**: This is enabled by default.
- **Sync Setting over a metered connection**: Windows 8 detects metered networks such as paring a desktop with a phone to connect to the Internet, and we can then decide whether to allow UE-V to work over this.
- **Sync Setting over a metered connection when roaming**: This disables UE-V when we are not connected to our normal service provider.

We then go to the corresponding folder for computers by navigating to **Computer Configuration** | **Administrative Templates** | **Windows Components** | **Microsoft User Experience Virtualization**.

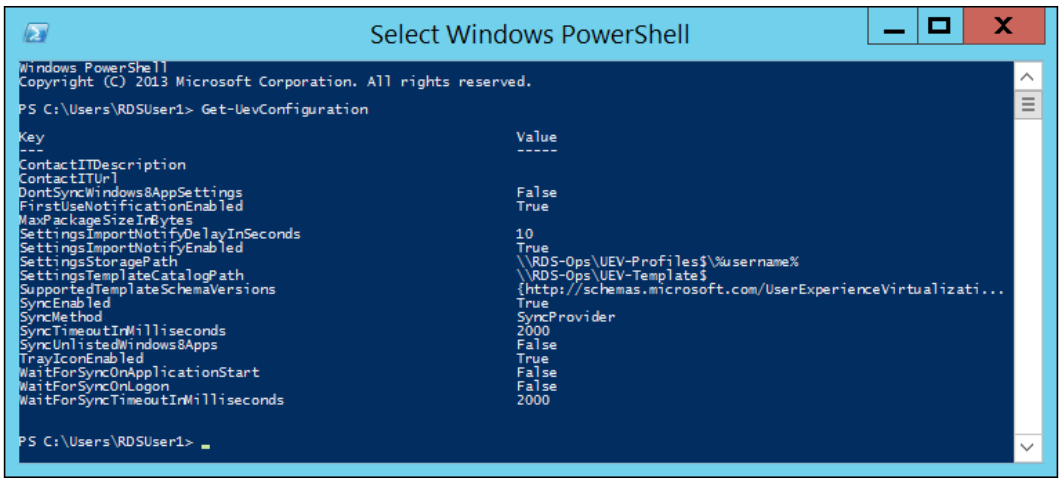
Note that as elsewhere in Group Policy, if we set something in **Users**, it takes precedence over the corresponding **Computer Configuration** setting.

There are the following additional options we'll need to be aware of that aren't included in the settings for users:

- **Contact IT Link text**: This is the text for the URL of our helpdesk on the **Company Settings Center** screen.
- **Contact IT URL**: This is the actual URL of our helpdesk in the **Company Settings Center** screen.
- **First Use Notification**: This lets a user know they are using UE-V in the system tray on their desktop. We will enable this to see it working.
- **Settings Template Path**: We need to enable this and set the path to the share we created for the UEV templates: `\\RDS-Ops\UEV-Template$`. There is an option to replace the default Microsoft templates, but in most scenarios, we will want to add additional templates, not overwrite the ones that were supplied.
- **Sync unlisted Windows 8 apps**: Because the programming model for Windows apps has sync built in, we can reliably just enable any unlisted app to get it to work if we want to.
- **Tray Icon**: This puts the UE-V agent icon into the system tray. We will enable this to see that UE-V is working.

In both **Users** and **Computer Configuration**, there are two additional folders, one for **Applications** (traditional desktop applications) and one for **Windows 8 apps**, and we can disable roaming for any of these if we want. However, this list is not an exhaustive one; for example, it didn't originally include Office 2013 because it has built-in cloud-based roaming capabilities of its own. So we need to understand how to get new templates for Microsoft and how to create our own for non-Microsoft applications.

We can also manage UE-V with our old friend PowerShell, and one thing we might want to check is whether UE-V has been configured from Group Policy by connecting to any of our virtual desktops with `Get-UEVConfiguration`, as shown:



We can also make changes, and the following script can be used to set up the template path on each of our virtual desktop computers instead of using Group Policy:

```
$ServerList = @("RDS-SHost","RDS-Shost2","FPC1", "FPC2")
$domaincred = new-object -typename System.Management.Automation.
PSCredential -argumentlist "Contoso\administrator", (ConvertTo-
SecureString "Passw0rd!" -AsPlainText -Force  )
foreach($Server in $ServerList){
    invoke-command -ComputerName $Server {
    Set-UevConfiguration -Computer -SettingsTemplateCatalogPath "\\
RDS-Ops\UEV-Template$"} -Credential $domaincred
}
```

However, this will fail on FPC1 and 2 because they don't have remote management configured and so we would need to enable this first (run `winrm quickconfig` as an administrator from the command line or in PowerShell).

UE-V can also be managed from System Center Configuration Manager 2012 R2, and there is a configuration pack for this available at `http://go.microsoft.com/fwlink/?LinkId=317263`. This allows settings to be made as part of an overall configuration baseline and to check that UE-V is working as expected.

Adding and creating UE-V settings location templates

Application templates are just XML files, but then so is a modern Word document, and just as we can edit in Microsoft Word, so there is the UE-V Generator, which is included in the UE-V download (`toolbox.exe`). We need to set this up as a clean VM and use the script we used in *Chapter 2*, *Designing a Virtual Desktop Infrastructure*, and modify the script to create a new VM, for example, **RDS-UEV**. We can then install the Generator there and the Foxit Enterprise Reader we used in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, as a basis for creating a new template. We can then launch the generator to see how it works using the following steps:

1. In the **Generator start** screen, select **Create a settings location template**.
2. Set the file path to `C:\Program Files (x86)\Foxit Software\Foxit Reader\Foxit Reader.exe`. There is no need to add any command-line arguments or set the working folder; just click on **Next**.
3. The Generator will launch the Foxit Reader and, because this is the first time we have run it, it will ask us if we want to use it as our default PDF reader. Say yes to this and confirm Foxit as our selection in the subsequent popup that gives us the choice of the reader or Foxit Reader.
4. The Generator will ask us to close Foxit Reader as soon as it's loaded, so we'll do that.
5. The Generator should respond with **Location discover is complete**. Click on **Next** to continue.
6. In the **Review Locations** screen, we can see that the Generator has found the registry key for the Foxit Reader (in my case, it also picked up **Software\Microsoft\Speech\Voices**, which I unchecked). Click on **Next** to continue.
7. In the **Edit Template** screen, we can edit the metadata for the template and we can see that five file locations have been discovered along with one registry location.
8. We can then save this template to our central share, `\\RDS-Ops\UEV-Template$`.

The Generator can't be used for the following types of applications:

- Virtualized applications
- Applications that are offered through Terminal Services
- Java applications
- Windows 8 applications

We'll be looking at the first two in the next chapter on applications and examine how to make sure UE-V works with those. Windows 8 applications are already managed by UE-V, so the only thing we would have to handcraft in UE-V are Java applications, by manually creating a template. Note, however, that the Generator can't validate it either.

There is also a link on the Generator to download templates from Microsoft (`http://gallery.technet.microsoft.com/site/search?f%5B0%5D.Type=RootCategory&f%5B0%5D.Value=UE-V&f%5B0%5D.Text=UE-V`), and a good example is that there is now an official Office 2013 template we can use once we have downloaded it and copied it to our template share. UE-V will run a scheduled task to pull down new and changed templates from the template share we set in Group Policy.

We can now test that we have UE-V working by making some changes to our desktop, opening a WordPad file or other application. UE-V will write our changes to the UE-V user setting share we created, and we can see those if we browse to it (`\\RDS-Ops\UEV-Profile$\RDS-User1`) while logged in as RDS User1. Note the files will be hidden here as well. So UE-V is pretty straightforward once we understand how it works; it quietly syncs our users' data as they work with little intervention from us other than to configure it.

Summary

In this chapter, we have seen the various methods for allowing users to keep their settings. Our lab example is now using all these technologies, but possibly not in a way that we would use in production. UPDs can only be used in isolation if we restrict our users to one collection. If not, we need to use either Roaming Profiles or UE-V. Both of these techniques can be used in conjunction with Folder Redirection and Offline Files, but UE-V is smarter in the way it synchronizes settings for applications and Windows, and if we have MDOP, we should use it. The other reason for using UE-V is that it works well with another part of MDOP: App-V. That's what we'll be looking at in the next chapter along with another kind of application virtualization—RD RemoteApp—and the business of running an application in RDS that looks just like any other application on a user's physical desktop.

9
Virtual Applications

So far in this book, we have looked at Server Virtualization, Desktop Virtualization, and most recently User Environment Virtualization, and now it's time to turn our attention to Application Virtualization. This is the separation of the application from the operating system it would normally run on, and we'll actually cover two entirely different techniques in this chapter: using remote desktop services to serve individual applications rather than whole desktops (known as RD RemoteApp); and App-V, where we deploy applications to both physical and virtual desktops that run in a sandboxed environment rather than actually installing them on the virtual desktops in our collections. We'll look at what these techniques are used for and why they matter and then deploy them into our lab to see them in action.

RD RemoteApp

RDS allows us to serve any individual application running on any of our collections rather than the whole desktop. To our users, it will look like just another app on their device, albeit a Windows app. If the user is on a Windows machine, the RemoteApp will look like just another icon on their desktop; however, we can't serve Windows 8 apps with RemoteApp. There are a number of reasons why RemoteApp is widely used for all sorts of situations:

- **Security**: We can provide an application that handles sensitive data so that all the user can do is run it. No data or details of the application will ever appear on the physical device it is accessed from, so it can co-exist on one screen. However, in the past, some organizations such as the police would have deployed a second device on a different network to do this.
- **Single use**: If all a user needs is one application, we could just deliver this to a thin client device in their location. This might be useful in museums and galleries or in a manufacturing environment. A variation of this is the branch office, where running an application from a central server makes sense because we don't have IT support to provide local help.

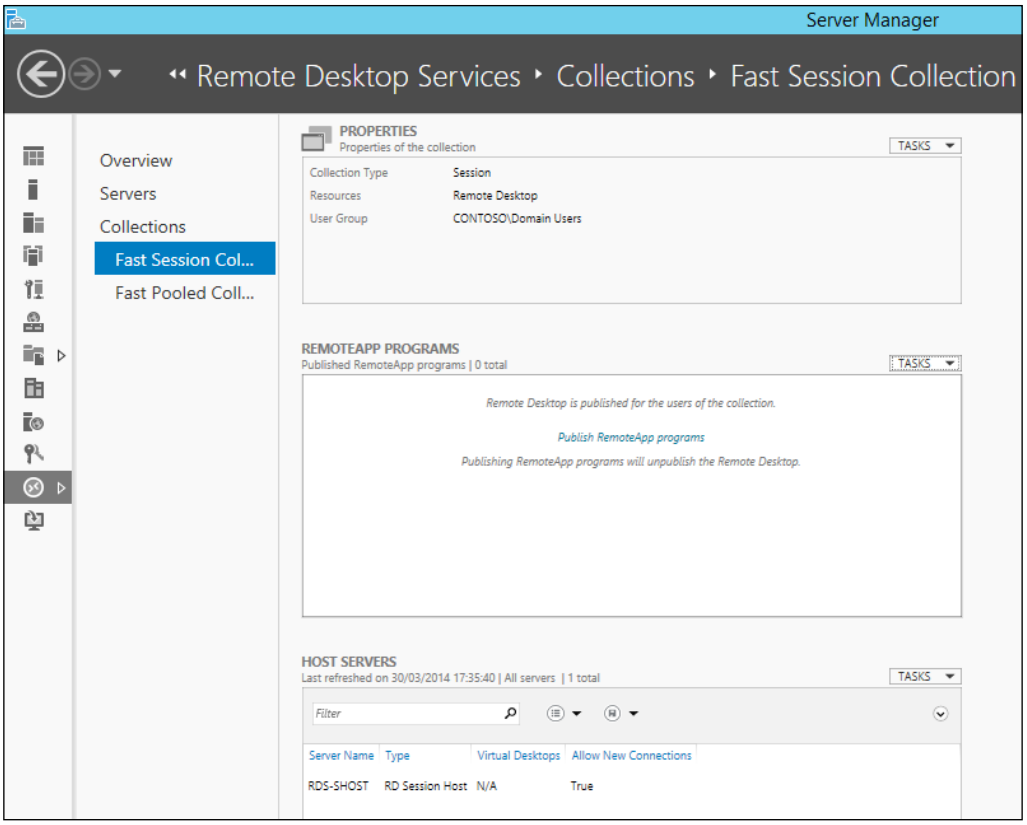
- **Legacy systems**: Despite XP being in its terminal phase and Windows Server 2003 nearly obsolete, some organizations need to run applications on these platforms. It is also possible that the rest of these systems are more up to date. While some of these scenarios are not supported by Microsoft, the applications are still essential to the business and have not or cannot be replaced. A related use is where there may be a compatibility issue between versions of the same application on a desktop.

RemoteApps are controlled inside the RDS section of Server Manager and are just an extension of the properties of a collection, and it doesn't matter if it is a Pooled or Session Collection. All we have to do is declare which applications on the virtual desktops we wish to publish to our users and which users can access each application. While it is possible to create RemoteApps based on a Pooled VDI Collection, this means each user requiring simultaneous access to the app will need access to a Pooled Virtual Desktop, which is a VM. It is a big overhead on our Virtualization Hosts to just serve out applications, so we will typically want to use RemoteApps served from Session Collections as much as possible, as this is far more efficient (as we saw in *Chapter 2*, *Designing a Virtual Desktop Infrastructure*). Given that all our users will see is the application rather than a server-based desktop, the only reason not to use RemoteApps on Pooled Collections is because we can't get the candidate application to work on Windows Server acting as a Session Host or because of the licensing restrictions of the application.

Publishing RemoteApps from a session host

We can create a RemoteApp from any program installed on our session hosts that our Session Collection is running on. Let's see how to set this up in Server Manager by just publishing on WordPad and Paint, which are already installed on the server:

1. Connect to **RDS-DC** and **Server Manager**.
2. Navigate to **Remote Desktop Services** | **Collections** | **Fast Session Collection** (which we created in *Chapter 2*, *Designing a Virtual Desktop Infrastructure*) as shown in the following screenshot:



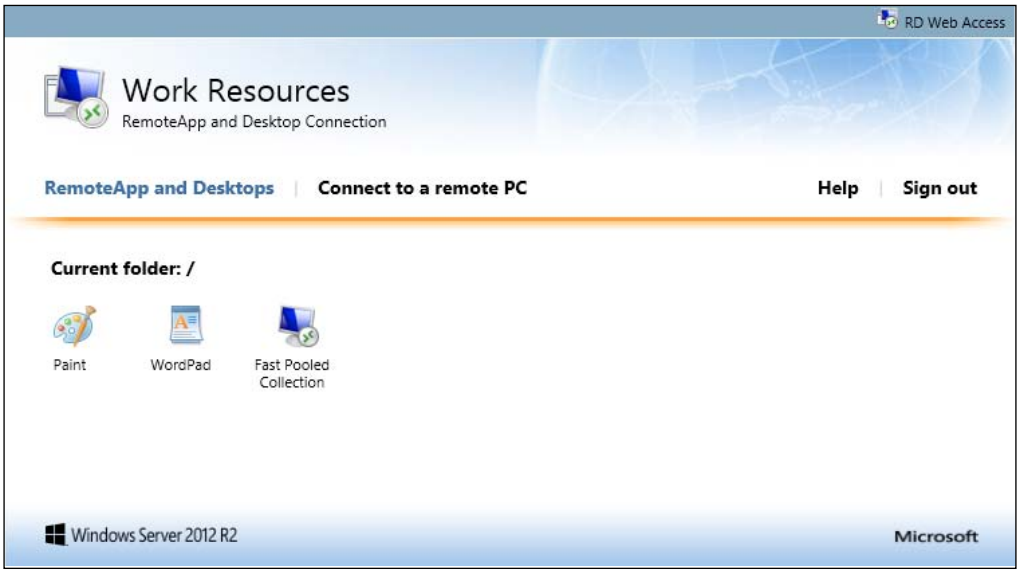
3. Click on **Publish RemoteApp programs** in the **RemoteApp Programs** section of the screen.
4. The RemoteApp wizard will launch and scan the session host to see what applications are installed on it. Select **Paint** and **WordPad** from the list and click on **Next**.
5. Click on **Publish** on the confirmation screen to close the wizard.

To do the same thing in PowerShell, we need to confirm that Paint and WordPad are installed and then set them up as RemoteApps with the following snippet:

```
Get-RDAvailableApp `
        -CollectionName "Fast Session Collection" `
        -ConnectionBroker RDS-broker.contoso.com `
|where displayname -In ("Paint", "WordPad") `
| New-RDRemoteApp `
        -CollectionName "Fast Session Collection" `
        -ConnectionBroker RDS-broker.contoso.com
```

In this code, the first command returns a list of the apps that can be published as RemoteApps in a given collection, which we can then use to look for Paint and WordPad, and then publish these with the `New-RDRemoteApp` command.

That's all there is to a basic RD RemoteApp deployment, and these RemoteApps are now available to our users; for example, if we log in to the RD Web Portal (`http://rds.contosos.com/rdweb`) as RDSUser3, we'll see Paint and WordPad listed as applications. Have a look at the following screenshot:

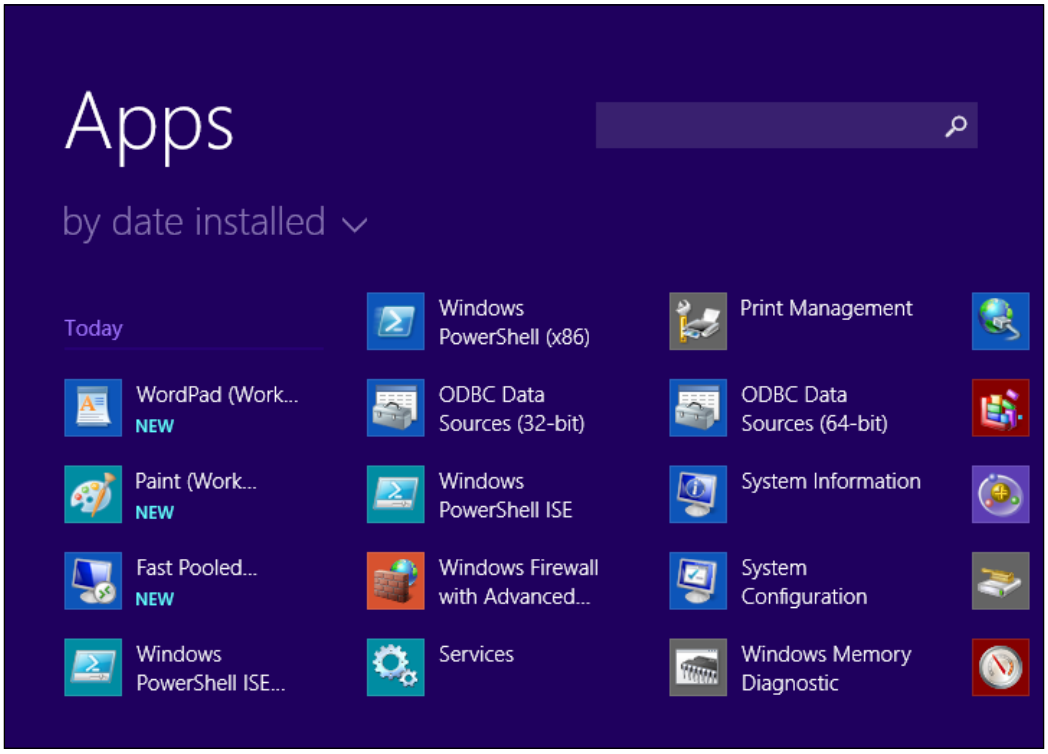


If we click on one of these, it will only open that application on the local desktop, and if we look carefully, we'll see a slightly modified icon on the taskbar to indicate it's a RemoteApp. If our users access RemoteApps from Windows (7 or later), we can save them the trouble of going to the web portal by configuring them to appear as icons on their desktop. They can also be made to appear on the Windows 8 start screen by configuring Group Policy. Follow the following steps to do so:

1. Connect to **RDS-DC**, and in **Server Manager**, go to **Tools** | **Group Policy Management**.
2. Navigate to **Contoso.com** and right-click on **Create a GPO in this domain and link it here**. Name the policy `RemoteApp` and click on **OK**. Right-click on `RemoteApp` and select **Edit**.
3. In **Group Policy Management Editor**, navigate to **User Configuration** | **Policies** | **Administrative Templates** | **Windows Components** | **Remote Desktop Services** | **RemoteApp and Desktop Connections**.

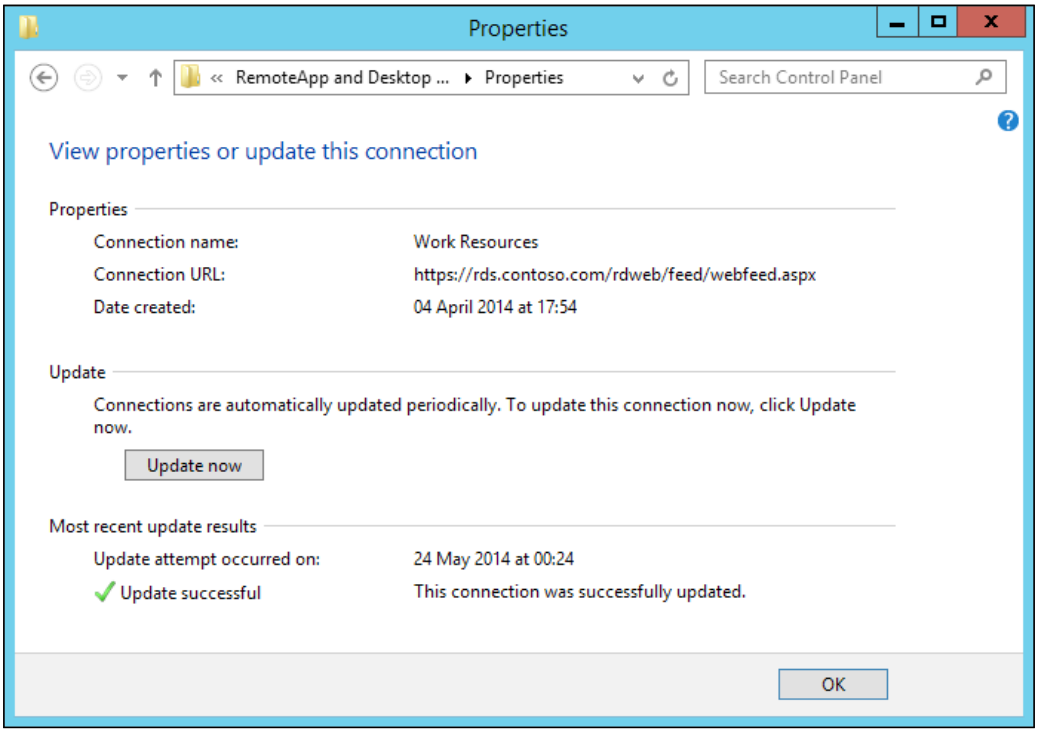
4. Double-click on the setting **Specify default connection URL**.
5. Check the **Enabled** option and enter `https://rds-contoso.com/rdweb/Feed/webfeed.aspx` as **Default connection URL**.

We will need to test this from any of our VMs by connecting to one of them and forcing a policy update to get this to take effect (from an elevated PowerShell or command prompt, type in `GPUpdate /force`). If we then examine the list of installed applications, they will show up as the latest ones installed:



Notice that we also get a link to our Fast Pooled Collection. The RemoteApps and virtual desktops will also show up in **Favourites** in the users' profile, under **Microsoft** | **Windows** | **Start Menu** | **Programs** | **Work Resources (RADC)**. What has happened behind the scenes is that a `.rdp` file has been copied from the RD Web Portal to the user's local profile and then a shortcut has been created to launch the Remote Desktop client (`MSTSC.exe`) for each RemoteApp / Virtual Desktop the user has access to.

Users can also set this up for themselves by running a search for RemoteApp on the local desktop and launching **RemoteApp and Desktop Connections** (a utility in the **Control Panel**). From there, they can connect to the RD Web Portal if we haven't set this up for them (`https://rds.contoso.com/rdweb/feed/webfeed.aspx`), as shown in the following screenshot:



However, if we have set up Group Policy to push the location of RemoteApps, we have created a **default connection**, which can only be removed by using Group Policy.

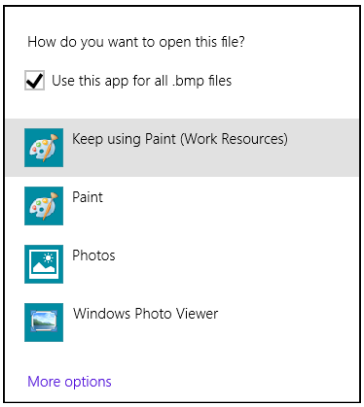
We can now go back to **Remote Desktop Services** in **Server Manager** to refine our basic RemoteApp by right-clicking a published RemoteApp and selecting **Edit** to change its properties, for example, to:

- Change the **General** properties of the RemoteApp (including its name), hide it from being shown in the RD Web Portal, and create groups of folders for them.
- Specify the command-line parameters to launch a RemoteApp and block the use of command-line parameters from the client.

- Refine which users in the collection can run the RemoteApp, or leave it as default, where all the users who have access to the collection can run all the RemoteApps in the collection.
- Set the file type associations so that if we have configured a default connection in GPO, our users will launch the RemoteApp automatically when they try and open a local file of the right type.

To see file associations in action, we change them using Paint by clicking the option for BMP. We will need to force a refresh of the RemoteApp setting on the local machine by going back into **RemoteApp and Desktop Connections** on a local machine (I have a base Win 8.1 VM in my lab for this, which is domain-joined to `Contoso.com` on RDS-DC) and clicking on the **Properties** button in **Work Resources** (default connection) and clicking on **Update Now** in the next screen.

If we have enabled a default connection to the RD Web Portal with Group Policy, these file associations are used on the local machine. For example, when we published Paint earlier, there is a file association to BMP files, and if we right-click on a local BMP file that has been configured with a default connection, we'll get the following window, which shows that the RD RemoteApp will launch automatically if we open this file. Have a look at the following screenshot:



We have already seen that it's a simple matter to create a Session Collection spread over multiple session hosts, whether this is for HA, for scale, or both. If we are going to use a multihost collection like this to serve RemoteApps, the app must be installed identically on each host as we have no control over which user accesses which host. We'll create usability problems and users will possibly lose their work if this is not done. RDS will try and enforce this when we set up a new RemoteApp, as the search for installed applications at the start of the RemoteApp wizard or by using the PowerShell command `Get-RDAvalaibleApp` will only show what's installed on all of our session hosts in a given collection.

Publishing RemoteApps from a Pooled Collection

The process to create RD RemoteApps from a Pooled Collection is much the same, but at first sight, this might look like a waste of a VM, which is being used just to publish one application to one user. For many scenarios, session-based RemoteApps is indeed the way to go, but there are those awkward and powerful apps that a certain set of users might need, particularly if they are not in the office. A good example is CAD, where an architect or engineer might be on-site with a lightweight or rugged device but needs to amend the drawings away from office. Doctors accessing diagnostic tools or an organization wishing to offer access to expensive and complex software to external contractors might also make use of this. The concerned applications won't run under session virtualization.

Application virtualization

One of the biggest challenges in providing VDI of any kind to our users is that we need to include all the applications they need across our various collections. At one extreme, we could deploy all our applications on just one collection that everyone uses; at the other, we could end up using just Personal Collections, with each virtual desktop replacing a physical desktop. The problem here is that we install applications on operating systems that run on specific computers, but it's our users who need access to them. So what we really want to do is assign applications to users. This is exactly how App-V works, and if that's all it did, it would still be an essential tool in VDI; however, it also has other useful features:

- Applications deployed as App-V packages are completely decoupled from the OS they are running on, so we don't install them on our session hosts or in our Virtual Desktop Template. This reduces the need to patch and update them and we only need to apply the relevant OS updates.
- App-V isolates applications from each other, so we can run potentially incompatible applications on the same desktop where we might have had two different collections. A good example is Office. We put Excel 2007 alongside Excel 2013, but a finance user might need both of these to run modern BI analysis as well as some old macros that won't work in the new version.
- App-V is smart enough to know when to update cached applications and will do this for us, so if we update the applications deployed by App-V centrally, the new version will cascade out to all our virtual desktops.

However, there are some things it can't do that we need to be aware of, as follows:

- It won't virtualize Internet Explorer, as that's part of the OS and not an app
- We can't virtualize Java
- Certain types of other applications can't be virtualized, such as anything that is installed at boot time or that needs access to drivers, uses COM+, or uses DLLs that run `DLLHost.exe`

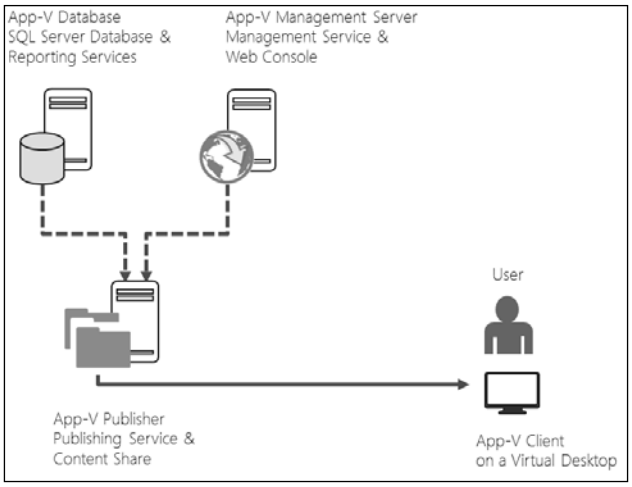
It's also important to understand that some applications virtualize well and are easy to implement while others require more effort. The applications that are tougher to virtualize will have more dependencies on other software and will have more registry setting as well as store configuration information in unusual places. We'll see how this plays out as we start to use App-V in our lab.

App-V, like UE-V, is a component of the **Microsoft Desktop Optimization Pack** (**MDOP**) and the only way to license it is to have Software Assurance in place. It is possible to test and evaluate App-V with an MSDN subscription.

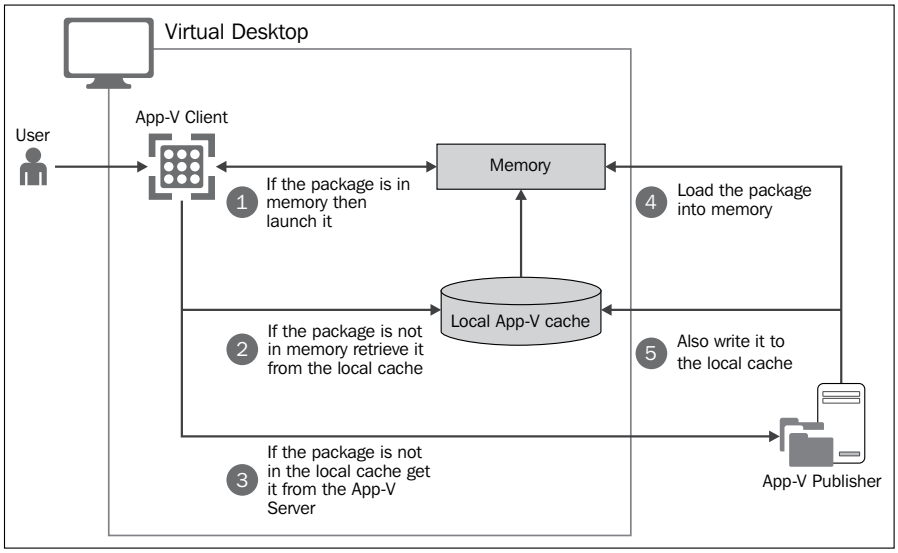
App-V architecture and components

App-V works by capturing an application we want to virtualize and putting this information into a package that is then deployed to a desktop. The process of capturing all the information about how a package works is called **Sequencing** and works in a similar way to how we captured user settings for UE-V in the previous chapter.

The package is then stored on a file share controlled by an App-V console and a web service that stores metadata about each package in a SQL Server database. Packages are controlled on each desktop by an **App-V client**. The App-V architecture is shown in the following figure:



When a user wants to run an application, the App-V client intercepts the requests and checks to see if the relevant package is already in memory. If it is, it will open; if not, the local App-V cache is checked to see if the package has been downloaded before and it'll load that if it has. However, if the package isn't available locally, it will be streamed from the designated **App-V Publisher**, as shown in the following figure:



That's fine for scenarios where users are on physical desktops, especially if they are working remotely, as this works well for that. However, the local cache is of limited use in VDI. Therefore, in App-V 5, this could be replaced by directing the App-V client to a **Shared Content Store** (**SCS**), which means that we don't end up with lots of copies of packages in our Pooled or Personal Collections. SCS also works really well with Session Collections as it can detect that a package is in the shared memory on the session hosts, so if another user asks for the same application, it will just use the copy in memory.

The App-V server side roles can all be combined for smaller deployments, but App-V can also be scaled out if needed. A single publishing server can handle up to 20,000 desktops and the **Management Server** and database in turn can manage 50 publishing servers, that is, 1,000,000 desktops! HA is also a necessary part of App-V as our users won't be able to run any App-V applications if the infrastructure is unavailable and each App-V server role uses the same techniques we have already looked at in *Chapter 5*, *High Availability*:

- The SQL Server databases can be mirrored or we can use SQL Always if we have SQL Server 2012 Enterprise Edition or later. Note that we can't use SQL Express for this, as App-V has a dependency on the SQL Server Agent service, which is not present in the free version.
- We can load-balance the IIS management console in the same way we have load-balanced the RD Web Portal.
- We can use a Scale-Out File Server (a Windows Server cluster with this role) to store the packages on a generic file share that is fast and highly available.

There are more details on capacity planning for App-V at `http://technet.microsoft.com/en-us/library/dn595131.aspx`.

There is also a reporting system built into App-V to provide monitoring. App-V can also be integrated into System Center Configuration Manager for complete device and user management. Once the App-V infrastructure is in place, we need to deploy the packages for the applications we want to virtualize. There are predefined package accelerators already available to help us deploy applications, such as Microsoft Office 2010 and 2013, but if there isn't one for a given application, we will need to sequence the application ourselves.

App-V packages

Now that we understand how App-V deploys packages, we need to know what a package actually is and how we create one. A package is nothing more than a collection of files of the following types:

File type	Description
`.appv`	The Virtual Application Package file containing all assets and states organized into streamable feature blocks
`.msi`	Executable deployment wrapper that allows the manual deployment of the `.appv` files or deployment via existing third-party deployment platforms
`_DeploymentConfig.XML`	This is used for customizing the default publishing parameters for all applications in a package
`_UserConfig.XML`	This is used for customizing the publishing parameters directed to specific user groups for all applications in a package
`.cab`	An optional file if a Package Accelerator file is used to automatically rebuild a previously sequenced virtual application package
`.appvt`	An optional Sequencer Template file used to retain commonly reused sequencer settings

These files are created by a special sequencing tool included in App-V. We launch this on a clean reference computer (a VM with a fresh installation of Windows 8.1 with no other applications installed) and then it sits in the background capturing the setting of an application as we install it. Having completed the application installation, we reopen the sequencer to make additional configuration changes and then deploy the package to our App-V servers.

App-V is a big topic in its own right, so all we are going to do in this chapter is to see how it fits in with VDI using a simple application as an example. For more information, you may wish to refer to *Getting Started with Microsoft Application Virtualization 4.6*, *Packt Publishing*, which is downloadable from `http://www.packtpub.com/article/faq-virtualization-and-microsoft-app-v`.

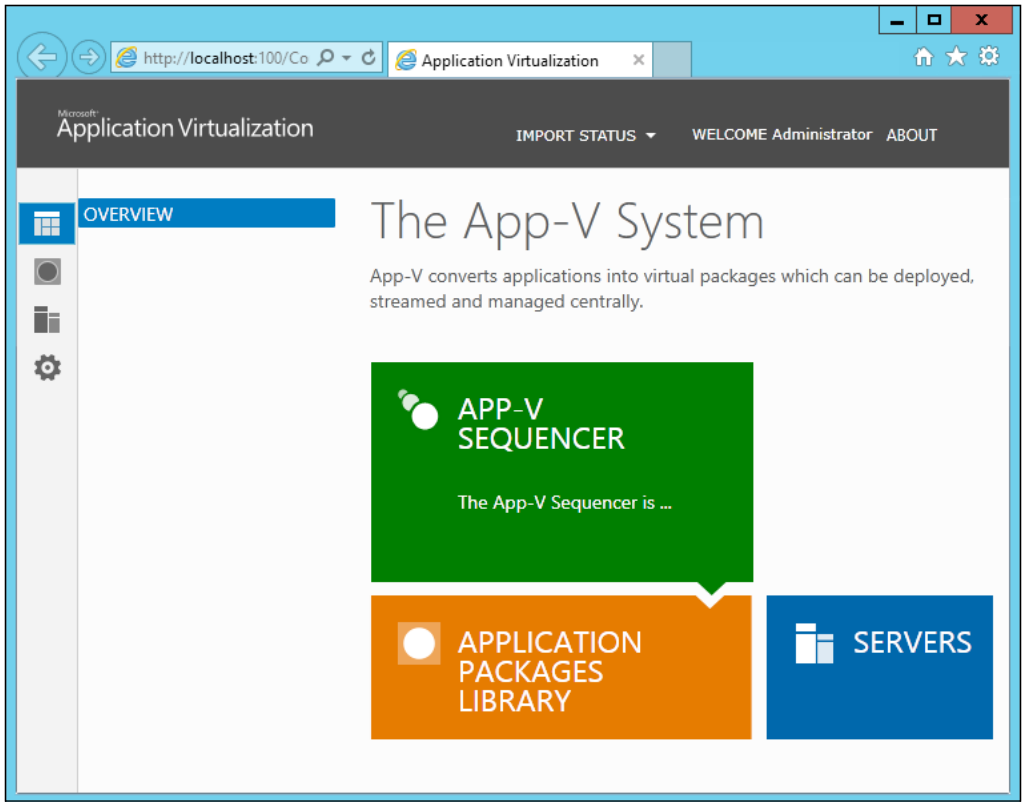
Installing the App-V infrastructure

We are going to use App-V 5.0 sp2 as it available on MSDN and works with Windows Server 2012 R2 and Windows 8.1. We will put all the App-V server roles (the server databases and publisher) on RDS-Ops, as we have SQL Server on there and a deduplicate volume on which we can create the share to host the App-V packages:

1. Mount the App-V ISO file (something like `mu_application_virtualization_hosting_for_desktops_5.0_service_pack_2_x86_cd_3210319.iso`) on **RDS-Ops** from **Hyper-V Manager** and connect to **RDS-Ops**.
2. Open the DVD drive and you should see three folders: one for the client, one for the sequencer, and one for the server; open the server folder, and launch the App-V Server installer (`APP_SERVER_SETUP.exe`).
3. Click on **Install** on the start screen, accept the license terms on the **Getting Started** screen, and click on **Next**. Select **Use Microsoft Update...** on the update screen and click on **Next**.
4. On the **Feature Select** screen, select all of the options and click on **Next**.
5. On the **Installation Location** screen, leave the location as it is and click on **Next**.
6. On the first **Configure** screen, we will install the default instance of SQL Server and the default configuration of the Management Server database, so just click on **Next** to continue.
7. On the second **Configure** screen, click on **Next** as we are using the default options.
8. On the third **Configure** screen, we will use the default instance of SQL Server for the reporting and the default Reporting Server database configuration, so click on **Next**.
9. On the fourth **Configure** screen, click on **Next** as we are using the default options for the reporting database.
10. On the fifth **Configure** screen, enter `contoso\administrator`, where in a production environment, we would create or use an AD Group for this. We also need to specify a port for the management website and we'll use **100** for that. Click on **Next**.
11. On the sixth **Configure** screen, we need to specify the web service port for the publishing services, and we'll use **101** for this. Click on **Next**.

12. On the seventh **Configure** screen, we need a port binding for Reporting Services, and we'll use **102** for that. Click on **Next**.
13. Click on **Install** on the **Ready** screen to complete the wizard and install the App-V server roles.

We can check whether the installation is ok by opening SQL Server Management Studio and connecting to RDS-Ops with Windows authentication, and we'll see that two new databases are installed: **AppVManagement** and **AppVReporting**. If we launch IIS Manager from RDS-Ops, we should see our three new websites: the management service, the publishing service, and Reporting Services. To connect to the management console, we'll need to install Silverlight on RDS-Ops and then we will see something like the following:



Finally, we will need a share to host our packages. We will call this share **App-V Packages** and host it on our App-V Server RDS-Ops. It needs just one extra permission: the machine account, RDS-Ops, must have full control of the share. Now that we have our server in place, we need to deploy the App-V client to our VDI deployment.

Installing the App-V client to virtual desktops

We can make use of MDT and the techniques we used in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*, to recreate our Pooled VDI Collection to include the App-V Client, just as we did in the previous chapter to add the UE-V agent:

1. We need to connect to RDS-Ops if we aren't already, launch the **MDT Workbench**, and mount the App-V ISO we used in the last section.
2. As in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template* and *Chapter 8*, *Managing User Profiles and Data*, we need to import the App-V client application. Open **Deployment Workbench**. Expand **MDT Deployment Share**, right-click on **Applications**, and select **New Application** to launch the **New Application Wizard**.
3. In the **Application Source type** screen, select the **Application with source files** option and click on **Next**.
4. In the details screen, for **Publisher**, enter `Microsoft`, **Application name** as `App-V`, and **Version** as `5.0 sp2`. Then click on **Next**.
5. In the **Source** screen, locate the folder with the EXE file (which should be `D:\APP-V 5.0 CLIENT SP1\CORE MSI`) and click on **Next**.
6. In the **Destination directory** screen, leave the default of **Microsoft App-V 5.0 sp2** and click on **Next**.
7. The App-V client has a lot of switches to configure it, but as with UE-V in the previous chapter, we can use Group Policy to set these centrally; therefore, all we need to specify here is:

   ```
   msiexec.exe /i APPV_CLIENT_MSI_X64.msi /quiet /norestart
   ```

8. Set the working directory to `\\RDS-OPS\DeploymentShare$\\Applications\Microsoft UE-V 2.`
9. Check the settings in the summary screen and click on **Next** to import the application. Review the output and click on **Finish** to close the wizard.
10. Expand the **Applications** directory under **Applications** in the navigation pane to the left, and the new application will show up on the center screen. Right-click on it and select **Properties**. In the **General** tab, we can see that the application has a GUID, which we need to know, so copy this to the clipboard. In the **Details** tab, we could set what OS it can be deployed on. And in the **Dependencies** tab, we could set any other applications or fields that this depends on.

11. Now we need to add in a step into our Task Sequence to install the App-V client. To do this, expand **Task Sequences** in the deployment share and right-click on the Task Sequence we have already created to set its properties. By default, the type of Task Sequence we selected earlier already has an application install step included. To find it, expand the `State Restore` folder on the Task Sequence list and then then copy and paste the entry for **Install Foxit Reader**.
12. Edit the second of the **Install Foxit Reader** application copy as follows:
    1. Set **Name** to **Install App-V**.
    2. Check the option **Install a single application**.
    3. Set **Application to install** to **Microsoft App-V 5.0 sp1** (use **Browse** to find it). Click on **Apply**.
13. We can then add this application into the **[Default]** section of the rules of our deployment share to install it automatically. The entry should like the following:

    ```
    Applications003 = <GUID copied from the properties of the
    Microsoft App-V 5.0 sp2>
    ```

14. Now, we can recreate our RDS-Ref VM and then use this as the Virtual Desktop Template to recreate our Pooled Collection, as we did earlier in this chapter, to apply the special update needed for profile versions.

Finally, we will open up the firewall on the client so that App-V can communicate with the publisher web service, which, for the lab, is port 101. We could do this as part of the design of our virtual desktop or implement this rule in Group Policy, which would be the best approach if we plan to use App-V across several VDI and Session Collections or to physical desktops.

Installing the App-V Client to session hosts

There's a special App-V Client for RDS, as session hosts are running Windows Server and not Windows client and our users are in session on the same OS and not on separate copies of the OS. All we need to do is mount the special ISO for App-V on RDS (mine was called `mu_application_virtualization_for_remote_desktop_services_5.0_service_pack_2_x86_cd_3209756.iso`), navigate to the `D:\App-V Client for RDS <XX>` folder, and run `APPV_CLIENT_FOR_RDS.exe`.

Remember that we need to put the session host into the user mode—as we did to install the UE-V agent in the previous chapter—before we perform the installation either by navigating to **Control Panel** | **Install Application on Remote Desktop Server** or running `Change /user install` before the installation and `change /user execute` afterwards.

We can accept the default options during the installation as we will manage App-V centrally across our VDI deployment with Group Policy. We need to install App-V on all of our RD Session Hosts (RDS-SHost and RDS-SHost2 in our lab) to keep them in sync.

Configuring App-V

We could have installed the App-V client with a multitude of command-line switches to configure it to connect to our server, configure SCS, and various other options. We can use the PowerShell cmdlets that come with App-V, such as `set-appvclientconfiguration` to configure settings on a client and `add-appvpublishingserver` to direct the client to the publishing server. However, the most effective way to ensure all our App-V clients are in a desired state is to use Group Policy. However, Group Policy, as shipped, doesn't know anything about how to manage App-V, so we need to add in an **Administrative Template** (**ADMX**) file to our Group Policy infrastructure. The ADMX files for MDOP are included in one download, so if you already set up Group Policy for UE-V, the App-V template is already installed as well. If not, download the installer from `http://www.microsoft.com/en-us/download/details.aspx?id=41183` and run it on **RDS-DC**.

Ideally, we should apply the GPO to configure all the virtual desktops in our lab, so on both the Pooled Collections and our session hosts. However, as we are just going to deploy an application to our Pooled Collection, all we need to do is to create a new GPO in our RDS-VDI OU and restrict it to the computers in it:

1. On **RDS-DC**, open **Group Policy Management** (by going to **Server Manager** | **Tools**).
2. In the **Group Policy Management** console, navigate to **Group Policy Management** | **Forest: Contoso.com** | **Domains** | **Contoso.com**.
3. Right-click on **RDS-VDI** and select **Create a GPO in this domain and link it here**.
4. Name the GPO `App-V Computers` and click on **OK**.
5. Highlight the new **App-V Computers** GPO, and in the **Security Filtering** section, select **Authenticated Users** and select **Remove**.

6. Still in **Security Filtering**, click on **Add** and enter `Domain Computers` and click on **OK**.
7. Right-click and edit the **App-V Computers** GPO.
8. In **Group Policy Management Editor**, navigate to **Users | Administrative Templates | System | App-V** to review the options we can control in App-V.

There are eight separate folders of settings for App-V, and it is worth going through them to see how the client and server can be configured:

- **CEIP**: There's just a single object to enable the Customer Experience Improvement Program, which we can ignore.
- **Client coexistence**: The one object in this folder allows earlier versions of App-V to work with packages created in earlier versions.
- **Integration**: This folder is for advanced integration of App-V with roaming profiles. Given that App-V and UE-V are both in MDOP, you have the other the best option is to use them together.
- **Publishing**: Here, we can specify whether the user sees App-V in action as it refreshes and the setting for up to five publishing servers to enable redundancy if one or more of the servers specified is missing. We'll enable the Enable Publishing Refresh UX policy and configure the Enable Publishing Server 1 Settings, as follows, to see it in action in our lab:
    1. Click on **Enabled** to enable the policy.
    2. Set the display name to `RDS-Ops`.
    3. Set **Publishing Server URL** to `http://RDS-Ops:101`.
    4. Set **Global Publishing Refresh on Logon** to `True`.
    5. Set **User Publishing Refresh Interval** to `1`.
    6. Set **User Publishing Refresh Interval Unit** to `Hour`.

These setting changes we are making here are just to test App-V. In a production environment, you'll want to set the publishing refresh differently to suit how often you add and change packages.

- **Reporting**: We edit the one policy in this folder, Reporting Server, to send telemetry about App-V so we can monitor it. Enable the policy and set the **Reporting Server URL** to `http://RDS-Ops:102/` and the **Reporting time** to `12`.

- **Scripts:** This allows us to run any scripts in the packages.
- **Streaming**: This controls how packages are sent to the desktop from the App-V Server:
    - **Specify what to load in the background** (that is, **Autoload**): This allows us to load all the applications automatically, just the ones that have been used previously, or none of them.
    - **Allow first time application launches if on a high cost Windows 8 metered connection**: Windows 8 can detect if it's connected via a metered network connection (for example, 4G), and so we can control how App-V behaves in this instance.
    - **Certificate filter for client SSL**: We can define a certificate in the local store to encrypt App-V streaming over SSL.
    - **Location provider**: This specifies the CLSID for a compatible implementation of the `AppvPackageLocationProvider` interface such as System Center Configuration Manager.
    - **Package installation root**: This specifies where the applications and updates will be installed. Normally, this is in the user profile.
    - **Package source root**: This overrides the source location for downloading package content.
    - **Package store access control**: This determines whether access to the package store should be controlled based on the package publishing status.
    - **Reestablishment interval**: This determines the number of seconds between attempts to reestablish a dropped session.
    - **Reestablishment retries**: This specifies the number of times to retry connecting a dropped session.
    - **Shared Content Store (SCS)**: This specifies whether to use SCS and its location.

To set up SCS, we will need to create a file share (APPV-SCS) in exactly the same way as we did in *Chapter 8*, *Managing User Profiles and Data*, except that we'll need to grant the advanced permissions **List Folder/ Read** data and **Create Folders/Append data** to the RDS-VDI users, and these should only be applied to this folder only. When we have this file share in place, enable the policy and set the path to the share, for example, `\\RDS-Ops\APV-SCS`.

    - **Verify certificate revocation list**: This verifies the server certificate revocation status before streaming using HTTPS

Creating an App-V sequence

We need to create a clean VM with the App-V sequencing tool installed on it. The OS needs to reflect the OS that the App-V packages will be deployed to; so if we are using session hosts, we should use Windows Server, where, for VDI, we will use the Windows client. For our lab environment, I am going to suggest we use a Windows 8.1 VM, which doesn't need to be domain-joined; we just need the App-V sequencer installed on it.

To quickly create a clean VM for App-V, use the script in *Chapter 3, Putting the D in VDI – Creating a Desktop Template* and change the name of the VM; for example, `RDS-AppV`.

Once the App-V sequencer is installed, it's very important to checkpoint the VM, as we might make mistakes in the sequencing process and we will need to revert to the state the VM was in before we started the installation of the application we wish to sequence. The detailed steps to install App-V are as follows:

1. Mount the App-V ISO (something like `mu_application_virtualization_hosting_for_desktops_5.0_service_pack_2_x86_cd_3210319.iso`) on the VM from Hyper-V Manager.
2. Connect to the VM, go through the sysprep questions, and when the VM is ready to use, install the sequencing tool at `D:\AppV-SEQUENCER 5.0 SP2\APPV-SEQUENCER-Setup.exe`.
3. Checkpoint the VM in Hyper-V.

Now, we can begin to design the process, referred to as a **recipe** in App-V, to sequence an application. The first thing to be aware of is how the application is installed. Secondly, document the answers to the first three questions listed here and note the rest of the points:

- Does this application do something that App-V doesn't support, and is there a workaround available?
- Are there any prerequisites as far as components are concerned? Also, are there any updates and patches needed to complete the installation?
- Are there any post-installation steps required?
- If the first thing an application does is unzip and install, extract to a temporary folder before starting the sequencer; otherwise, we'll have the ZIP file and the unextracted files in the package, which will slow down the installation and take up more space.

- App-V has a **Primary Virtual Application Directory** (**PVAD**) setting that should be set to match the directory the application is installed to ensure application compatibility and best performance. The simplest way to check this is to start the installation of a package and see where it installs by default, cancel the installation, and use this as the PVAD.
- If the application has an option installed on first use, this should be disabled.
- If the application has an option to check for updates, this must be disabled as we will use App-V to sequence and deploy a new version.
- If the application has initial startup steps to configure it, document these.

We can then launch the sequencer, which will give us lots of useful links to help us use it. We can also configure its working by navigating to **Menu** | **Tools** | **Options**. We can also set up a working (scratch) directory and set its naming conventions. In the **Parse Items** and **Exclude Items** tabs, we can configure where the sequencer scans for changes as we install our applications. These items include both folders and registry settings. To show how the process works, let's look at a simple example of the VLC media player, which will allow our users to play DVDs on their local drives (this functionality is not included in Windows 8):

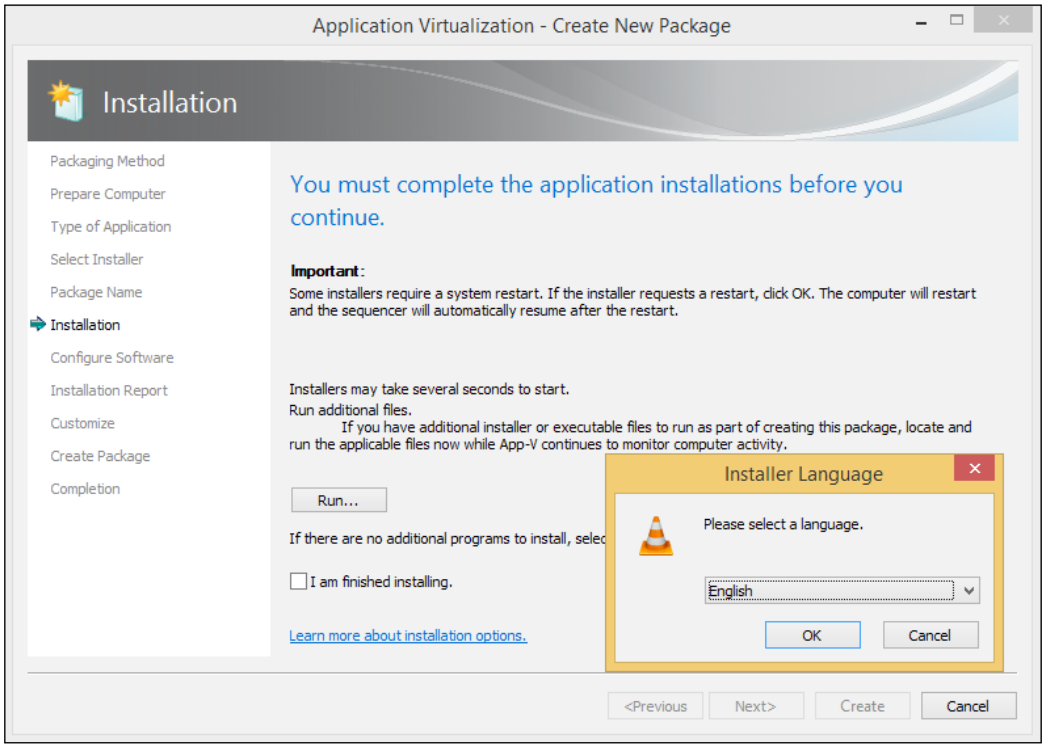
1. Connect to the reference computer we created just now and launch **Microsoft Application Virtualization Sequencer**.
2. In the **Sequencer start** screen, select **Create a New Virtualization Package**.
3. In the **Packaging Method** screen, select **Create Package (default)** and click on **Next**.

You can get App-V package accelerators to assist in deploying and then select the option to use an accelerator; for example, the Office accelerators can be found at `http://gallery.technet.microsoft.com/office/site/search?f%5B0%5D.Type=RootCategory&f%5B0%5D.Value=App-V`.

4. In the **Prepare Computer** screen, we'll get some warnings about other software that are running; in my case, it is **Windows Search**, **Defender**, and **Antivirus**. I went into the **Defender console** to turn it off and I stopped the **Windows Search** service from **Services**. It's also a good idea to ensure that automatic updates are disabled on the reference VM we are using to avoid confusing the sequencer. Click on **Refresh** to confirm that there are no warnings and click on **Next**.

If you plan to enable **User Account Control** (**UAC**) in your VDI deployment, make sure the reference computer is set up in the same way.

5. In the **Type of Application** screen, we can choose to deploy straight applications or add-ins and middleware. For example, there are Power BI add-ins for Excel our user might need to do work with business intelligence. Select **Standard Application (default)** and click on **Next**.
6. In the **Select Installer** screen, we need to point to the application installer we want to work with. I downloaded VLC media player and then installed it from there. Enter the path to the installer (`vlc-2.1.3-win32.exe`) and click on **Next**.
7. In the **Package Name** screen, enter `VLC Media player 2.1.3` and enter `c:\Program Files (x86)\VideoLAN\VLC` as **Primary Virtual Directory** and click on **Next**.
8. App-V will launch the installation of the VLC media player and sit in the background, as shown in the following screenshot:



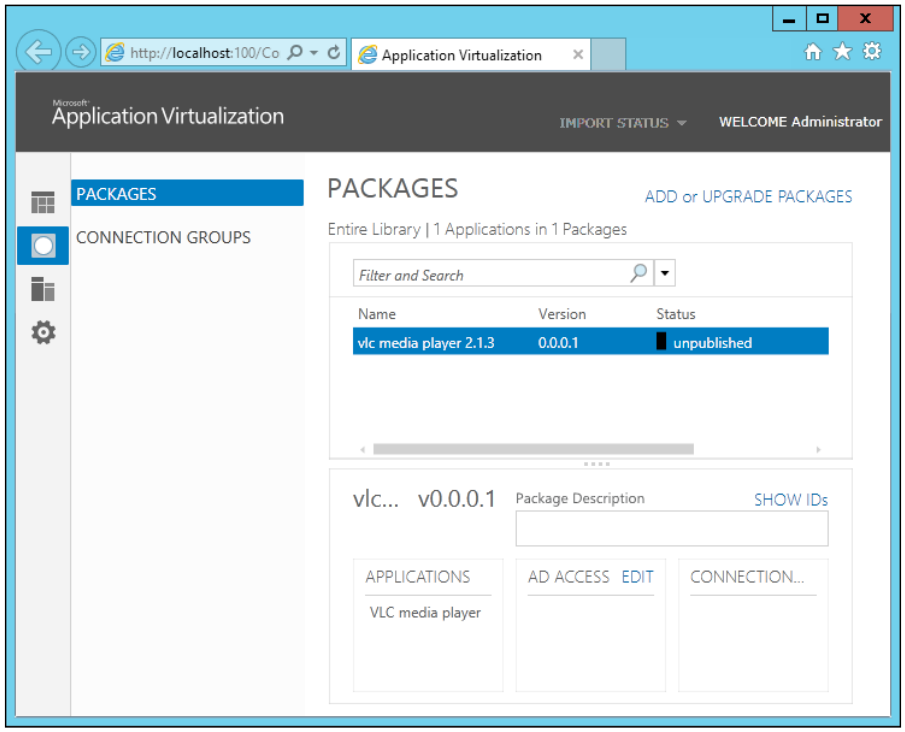
9. We can now go through the steps to install the VLC media player:
    1. Select the appropriate **Installer Language** in the VLC ,media player installer and click on **OK**.
    2. Accept the license terms and click on **Next**.
    3. Choose the components you want to install and click on **Next**.
    4. Note that by default, the VLC media player will install to `c:\Program Files (x86)\VideoLAN\VLC`, and if we decide to install it elsewhere, the PVAD and this setting should match. Click on **Next**.
    5. Click on **Install**, and when the installation has finished, uncheck the option to run VLC.
10. On the **Installations** screen, check the **I am finished installing** option and click on **Next**.
11. On the **Configure Software** screen, we have the option to run the application and configure first use. Highlight **VLC media player** and click on **Run selected**.
    1. VLC will open and pop up a screen asking us to check for updates and automatically retrieve media info. Uncheck the updates option and click on **Continue**.
    2. Close the VLC media player.
12. You might get a warning in the **Installation Report** screen that files have been excluded from the package. Click on the warning to see what the issue is; in my case, it was simply that one or more files were excluded from the package because they are located in excluded paths. Missing files can cause unexpected application failures. The following files were excluded: `C:\Users\Andrew\AppData\Local` and `C:\Users\Andrew\AppData\Local\Microsoft`. This can be ignored as the VLC media player doesn't use these folders. Close the warning and click on **Next**.
13. In the **Customize** screen, we can restrict operating systems that can run the package, and we should do this for RDS if our users are going to use both Pooled Collection and Session Collections. For the Session Collection, we need to create a separate package for the application using a Windows Server 2012 R2 VM as a reference computer. Click on the **Customize** option and click on **Next**.
14. Click on **Next** and ignore the warning to leave the streaming options as is, as our VDI environment will be connected over a fast LAN.
15. In the Target OS screen, select the x86 and x64 versions of Windows 8.1 and click on **Next**.

16. Give the package a meaningful description and select **Create** to create the package.
17. Ignore the warning if there is one and click on **Close**.

Deploying a package

We now have a folder on our desktop containing the package we created, which we can now deploy to our App-V Server (RDS-Ops):

1. Connect to **RDS-Ops** and copy the package to the share we created to host our packages (**RDS-Ops\AppV-Packages**).
2. Open the App-V web portal and, in the packages section, click on **Add or Upgrade Packages**. We'll need to select the `.AppV` file and import the package.
3. The package will now show up in the **Packages** screen as unpublished, as shown in the following screenshot:



Now we need to do two things: publish the package and grant access to it. Publishing is simply a matter of right-clicking on it and selecting **Publish**, and we can also right-click on it and set **Active Directory Access**. Enter `contoso\VDI-Users` and click **Grant Access**. We can now test this by logging on to the RD web portal (`https://rds.contoso.com/rdweb`) as `contoso\RDSUser1`, and we'll see our VLC application in the list of applications. We can check whether everything is working by looking at the event logs on both the client and server, as there is a special section for App-V, which we can reach by navigating to **Event Viewer** | **Applications and Service Logs** | **Microsoft** | **App-V**. We can also use PowerShell to check whether all is well on the client with `Get-AppvPublishingServer` and `Get-AppvClientConfiguration` to check whether our group policy has updated the location of our publisher, and `Get-AppvClientPackage` to see what packages are currently available on the client.

UE-V and App-V

UE-V and App-V are designed to work together, and the agents for each of these, although different, won't interfere with each other. There is just one thing to be careful of, and that is that UE-V can't see the settings for a package deployed with App-V. So, if we are going to deploy an application for use with both of these technologies, we'll need to train the UE-V Generator against a physical installation of an application so that it can determine where the user settings and configuration are stored.

App-V and System Center Configuration Manager

App-V is an extensible system that allows it to be controlled by other solutions—the obvious one being System Center Configuration Manager 2012 R2 (CM12R2). CM12R2 has the concept of advertisements, through which users can get applications or have them delivered automatically. This process then essentially replaces the App-V server, and while this does give a common location from where a user's devices, applications, and settings can be controlled, we have already seen that CM12R2 doesn't really work with VDI, although it does work with session hosts. So, my advice is if you have it already, it might be worth seeing how it works with VDI, but that it doesn't have sufficient integration with VDI to warrant using it just for managing our virtual desktops and App-V.

Summary

In this chapter, we have seen that we can use RD RemoteApp to offer individual applications to users rather than a whole desktop so they can use it without installing on the physical device they are using. This can be considered to be one form of application virtualization. We also saw that we can deploy applications to our virtual desktops with App-V so that they don't need to be installed and can be targeted at specific users; also, by doing this, we have a single point of management for updating them. However, App-V is not part of Windows; it is part of MDOP, and that is only licensed for customers who have signed up to Microsoft's Software Assurance program. Licensing is perhaps the hardest part of VDI to understand, and that is what we shall look at in the next chapter.

10
Licensing and the Future of VDI

In this chapter, we will review the wider licensing requirements needed to deploy VDI in our organizations. We will look at what we need across Windows Server and the more complicated licensing for Windows 8/8.1. Then we'll look at the licensing requirements of the additional software we have used in this book such as SQL Server and MDOP and conclude with a look at how Office is licensed in a VDI world. In the last section of this book, we will look at how we might use a service or cloud provider to provide VDI rather than using our own servers, and what the licensing implications are of this.

This chapter is for guidance only. I have researched this to the best of my abilities. Before making any decisions about implementing VDI or preparing a business case for it, you should definitely check with Microsoft or your Microsoft licensing reseller for up-to-date information about license compliance and costs.

Windows Server

We have used Windows Server to underpin our VDI deployment and have made use of a variety of roles and features such as RDS's own Active Directory, Update Services, File Servers, and of course Hyper-V. All of these features are built-in in the two main paid editions of Windows Server: Data Center and Standard. The other important edition is **Hyper-V Server**, which is a cut-down version that has no GUI (such as Server Core) and only has the Hyper-V and File Server roles and the Failover Clustering feature. So it's designed to be remotely managed.

When we install any of the following on a physical server, we get a certain amount of rights to use the same operating system in the guest VMs that are running on that host:

- **Windows Server Datacenter edition**: This is licensed per two physical CPUs (the number of cores doesn't matter) and allows us to run an unlimited number of VMs on that server using the same OS and certain earlier versions of Windows Server.
- **Windows Server Standard edition**: This is also licensed per two physical CPUs but only allows us to run two VMs on that host, provided that there are no other roles installed on the host OS apart from Hyper-V. Apart from this limitation, there is no restriction on any of the features compared with the Datacenter edition, so the decision to use Standard or Datacenter is simply a question of the costs of the two. Typically, if you have more than eight server VMs per host, then the datacenter is going to cost less. Also, bear in mind that if we are planning on moving VMs between hosts, say for planned maintenance, there may be licensing issues if we are using Standard edition on our hosts.
- **Hyper-V Server**: This confers no licensing rights to guest VMs, so they must be licensed themselves. This is perfect for our VDI virtualization hosts as we will be licensing Windows Client anyway; in large deployments, we will have dedicated hosts for these VMs.

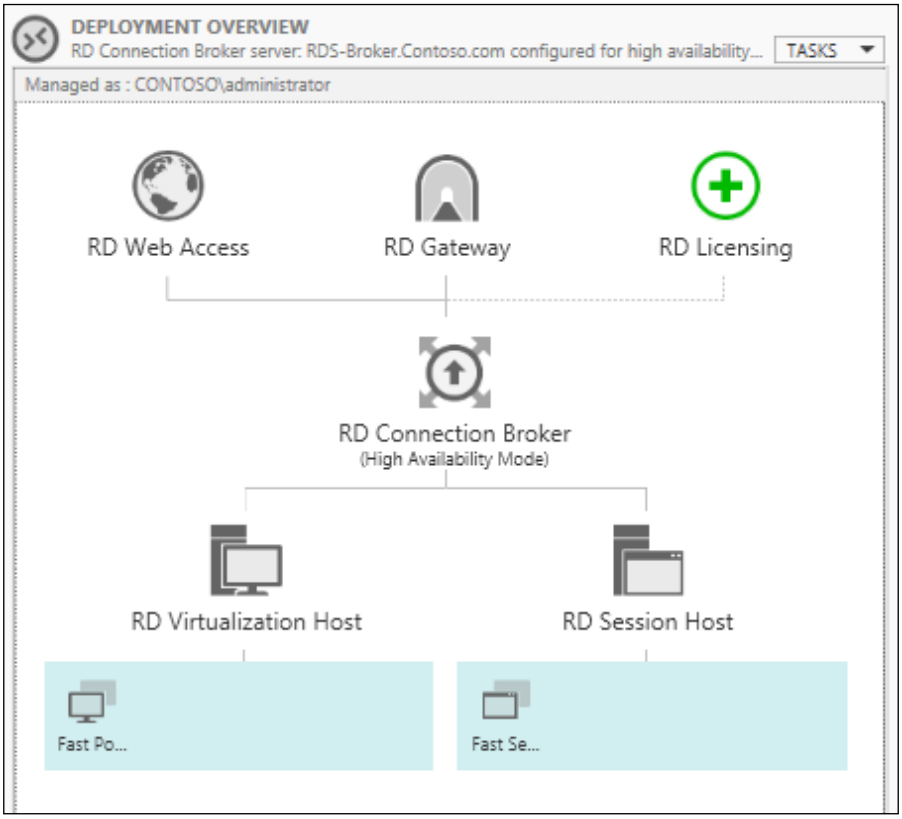
The definitive guide to licensing Windows Server can be found at `http://www.microsoft.com/licensing/about-licensing/briefs/remote-desktop-services.aspx`.

Applying this to a medium-to-large VDI deployment, we would have two or three physical hosts running the Datacenter edition, which will host our RDS role servers (for example, the Broker Web Access portal) in HA, plus the additional server we have used, such as Updates Services, AD, and App-V. Our Pooled and Personal VDI Collections will run on dedicated hosts running on Hyper-V Server. That excludes session hosts, and these are licensed to allow our users to simultaneously access them for VDI; there is a special Window Server role service for this: the RD Licensing server.

Remote desktop licensing

The licensing for session hosts is either licensed by a device or by a user, and we need to buy the appropriate **Client Access Licenses** (**CALs**) to suit our needs. If our users are hot-desking, possibly using thin clients, then we can license each of those devices and any user can access our session collections from those. Licensing in this way would be good for schools, manufacturing, and call centers where all we need to manage is the device, not the user. However, our mobile and remote workers, such as the sales and marketing teams, may need to use their laptops, tablets, and phones to access their virtual desktop, and so we'll use RDS User CALs for these users and manage the user, not the device. The problem is that a given VDI deployment can only be licensed in one way, so if we have both scenarios in an organization, we may need to create more than VDI deployments to get around this.

You may have noticed that there is one thing missing from our VDI lab on the RDS Deployment diagram (on RDS-DC): there's no licensing server, just a green cross to indicate we need to add one, as shown in the following screenshot:



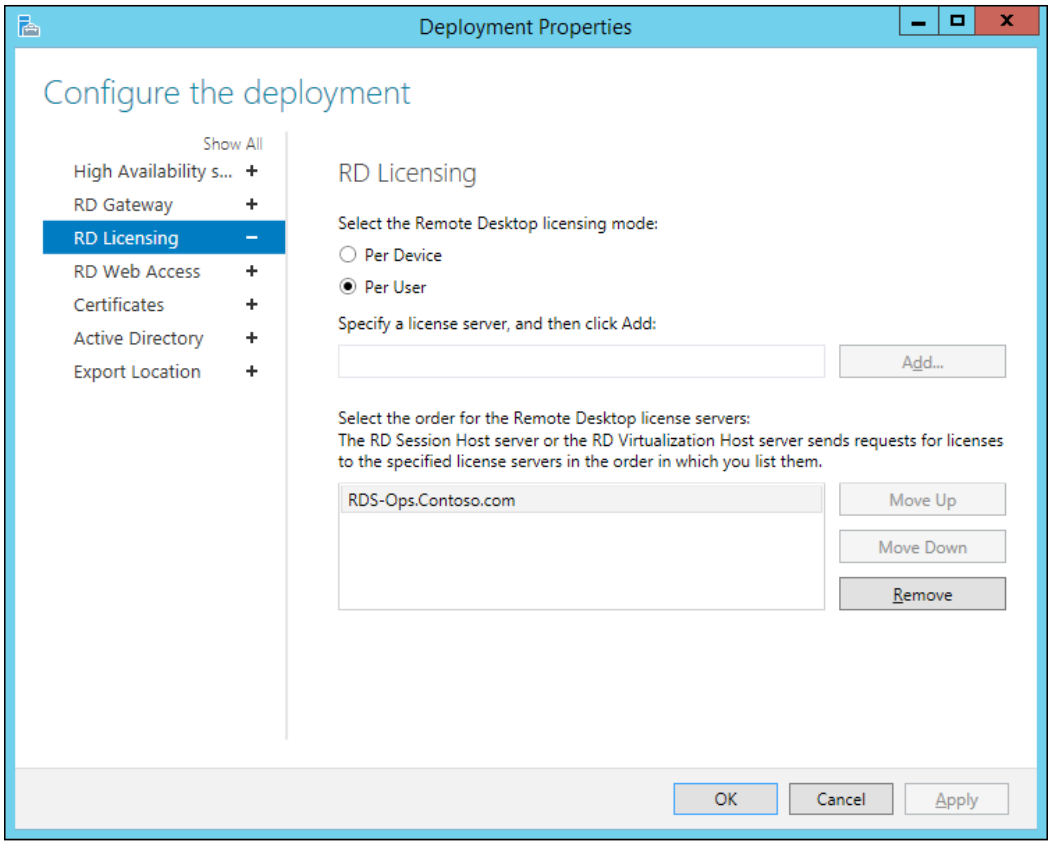
To set up and configure a licensing server using RDS-Ops again, perform the following steps:

1. Click on the green cross.
2. On the **Add RD Licensing Servers** screen, select **RDS-Ops**, and click on **Next** and on **Add** to complete the process.

However, while the RD licensing server is now in place, it has no licenses associated with it, and we'll now need to add those. Perform the following steps to do so:

1. To add licenses from RDS-DC or RDS-Ops, navigate to **Server Manager** | **Tools** | **Terminal Services** | **Remote Desktop Services Licensing** and open **RD Licensing Manager**.
2. Connect to **RDS-Ops** by right-clicking on **All Servers**.
3. We'll need to add our new license server to the Terminal Services License Servers Group in AD to allow it to issue RDS User Client Access Licenses (CALs) to our VDI users. We can fix this by clicking on **Review** in the **Configuration** column, clicking on **Add to AD Group**, ignoring the warning, and clicking on **OK**.
4. The server will still have a status of not activated as it has not been registered with Microsoft. To do this, you will need to connect RDS-Ops to the Internet. One way to do this is to stop the VM, add in an extra network adapter, and connect that to an NIC on the host that has an Internet connection.
5. Once we put in our company details, the license server will contact Microsoft Clearinghouse to register the server. When this is complete, we can add in licenses from our MSDN subscription.

We now need to configure the licensing server in the Broker to reflect the licenses we have added by navigating to **Server Manager** | **Remote Desktop Services** | **Overview on RDS-DC** and then navigating to **Tasks** | **Edit Deployment Properties**, as shown in the following screenshot:



The one problem with the licensing server is that it is a single point of failure, so what happens if we take it offline for maintenance or it fails for some reason? Even if any client device has an unexpired Per Device RDS CAL, it will still be able to connect to our RDSH servers. Any devices that do not have a Per Device RDS CAL or have an expired RDS CAL will not be able to connect until we get a new RD licensing server running. If the server is to be out of action for some time, we can install our RDS CALs on a different server. To do that, we need to contact the MS Clearinghouse, explain why the server failed, and they will help us to install the RDS CALs on the new server.

For disaster recovery, Microsoft's advice is to have a second RD licensing server (with no CALs installed) and have it listed as the second server on our VDI deployments. Then, devices that do not already have a Per Device RDS CAL will be given temporary licenses if the primary RD licensing server fails. However, devices that have an expired CAL will not be able to connect until we reinstall the RDS CALs on the secondary server with the assistance of the MS Clearinghouse.

License activation for Windows

One of the new roles in Windows Server 2012 is **Volume Activation Services**, and this allows us to activate various Microsoft licenses and works with **Active Directory-Based Activation** (**ADBA**) and a **Key Management Server** (**KMS**) where we have a **Generic Volume License Key** (**GVLK**). There are several other tools for managing license keys that can be used as well, such as the **Volume Activation Management Tool** (**VAMT**), which comes with the Windows Assessment and Deployment kit that we used in *Chapter 3*, *Putting the D in VDI – Creating a Desktop Template*. This can be backed by a SQL Server database and can centrally manage all our keys for Windows and Office. If these tools are not in place, we will need to expose a server or desktop (physical or virtual) to the Internet to be activated by Microsoft with one exception: if a physical server running Windows Server 2012 R2 has any VMs that are also running Windows Server 2012 (be it Datacenter, Standard, or Essentials), then those VMs will be automatically activated by the host using **Automatic Virtual Machine Activation** (**AVMVA**). In VDI, we can use a combination of AVMVA to activate the VMs running our role servers and our session hosts (if these are VMS themselves), and we can use the other tools to activate our virtual desktops in our Pooled and Personal Collections. The procedure for doing this would be as follows:

1. Obtain a **KMS host key** from Microsoft Volume Licensing.
2. Install the **Volume Activation** role server either from **Add roles and features** in **Server Manager** or with the following PowerShell command:

   ```
   Install-WindowsFeature VolumeActivation
   ```

3. Open **Server Manager** and navigate to **Tools** | **Volume Activation Tools**.
4. On the **Select Volume Activation Method** screen, select **Active Directory-based Activation** and click on **Next** (you'll need Enterprise administrator rights to do this).
5. Enter the KMS host key and optionally a name for the Active Directory object, and then click on **Next**.

This approach won't work for Office 2010 but does work for Office 2013. So, if the activation of an older version is needed, then we would need to set up a traditional KMS server. However, in its latest version that can also be done through the Volume Activations tools. Remember that in both cases you'll need to ensure the KMS port is open on client OS (port 1688 by default).

Windows 8.1

Licensing VDI in a modern organization is not easy to understand, especially where we might have contractors and staff bringing their own devices to work or using their own devices in remote locations. The key to simplifying this is to have Software Assurance to license all the devices, which also allows us to use MDOP and enable App-V and UE-V to simplify our VDI deployments and this gives us another benefit: Virtual Desktop Access (VDA) licensing. This is perfect to remotely access a VM running Windows that is running on another device (our virtualization host). VDA is configured per device and is a subscription, and we may need to acquire extra licenses because VDA works in the following manner:

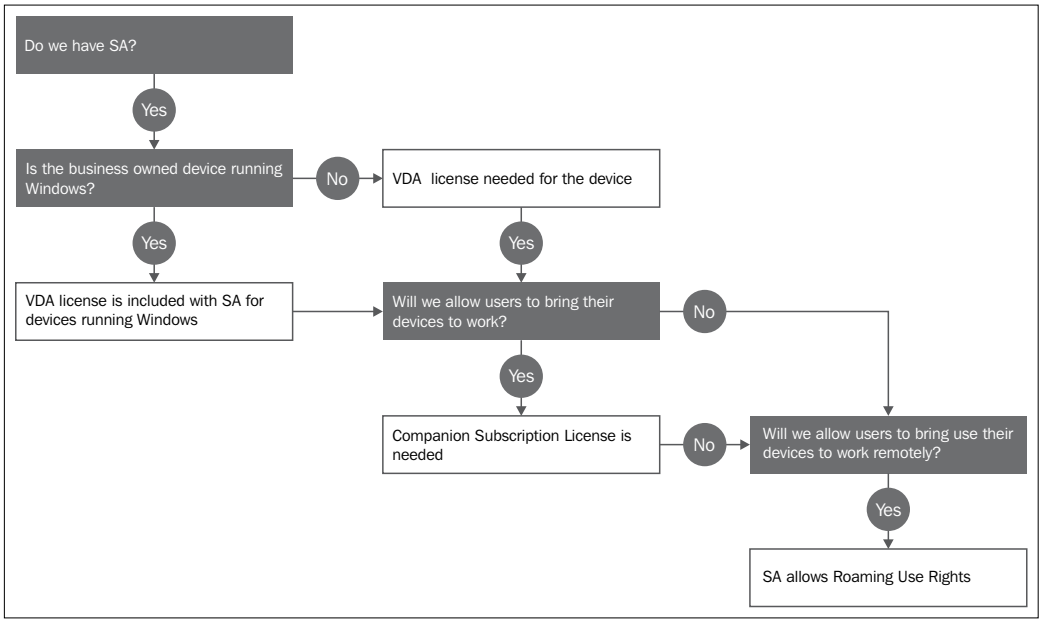
- If the device is a corporate laptop or desktop running Windows 8/8.1 that is covered by SA, then VDA is included.

Some thin client devices such as those from Dell and 10ZiG run a cut-down version of Windows: Windows 8 embedded. These already include VDA and so a separate device license is not needed but does need to be included in SA (`http://www.microsoft.com/licensing/about-licensing/briefs/windows_embedded_8.aspx`). There is also Windows ThinPC, which allows older devices to access VDI by essentially turning them into thin clients with a similar cut-down version of Windows.

- If the corporate device can't run Windows, for example, a thin client, iPad, or Android tablet, then we can buy VDA (about $100 per device per year) for that so these corporate devices can access our VDI deployment.

Either of these allows us to add an optional **Companion Subscription License** (**CSL**) that allows the user of that corporate device to access VDI from up to four companion devices such as smartphones and tablets owned by the user but used at work.

Finally, if our users are using devices away from the office, then the Roaming Use Rights that are included with SA allow them to access VDI when they are away from the office. This is illustrated in the following figure:



One thing to note is that when we employ temporary staff such as contractors, even if they have licensed copies of Windows themselves (possibly provided by another organization with SA), they still need to be licensed to access our virtual desktops, and this is definitely worth checking up on with a licensing specialist.

Some organizations and vendors use VDI but use Windows Server as the guest OS in each VM rather than a Windows Client. This is not the same as Session Virtualization and does allow the deployment of a wider set of applications where some applications can't be installed or can't be licensed. While this does allow each user to have their own VM and so be more isolated than they would be in Session Virtualization, we are still giving our users a server OS.

Other software

In addition to Windows Server 2012 R2 and Windows 8.1, we have used several other Microsoft solutions and tools in this book to manage and configure our VDI deployments. Some of these are not licensed per se, for example, the **Microsoft Deployment Toolkit** (**MDT**) and **Windows Assessment and Deployment Toolkit** (**ADK**), but others are paid licenses and so we need to be aware of how these are licensed to assess whether they are useful enough to warrant the additional cost.

MDOP

In this book, we have looked at two of the features of MDOP (the Microsoft Desktop Optimization Pack), App-V, and UE-V, but there are several other utilities that might be useful as well, particularly if we are also managing physical desktops, as follows:

- **Advanced Group Policy Management (AGPM)**: This allows for proper change control and management of group policy where thus far we have just been making changes to our "live" lab environment. This works by having a central service and then a client on each target machine, much as we have already for UE-V and App-V.
- **Microsoft BitLocker Administration and Monitoring (MBAM)**: This enables Enterprise management of the status and keys for Microsoft's disk encryption technology, BitLocker, on servers, and clients.
- **Diagnostic and Recovery Toolset (DaRT)**: This is a detailed set of tools for repairing Windows Server and Client, including repair tools for getting back partitions and file recovery utilities.

For more details on MDOP, visit the MDOP Tech Center (`http://technet.microsoft.com/en-US/windows/bb899442.aspx?ocid=wc-mscom-ent`).

MDOP is only available to customers with SA and requires a subscription of $10 per device per year at the time of this writing. However, App-V is included with RDS CALs for using application virtualization with Session Virtualization.

SQL Server

We have used SQL Server for a number of purposes in this book to enable HA Broker and as databases for WSUS and App-V. If we plan to implement VDI with high availability, our SQL Server databases for the Broker and for App-V must also be HA. The simplest way to do that is with mirroring, and the cheapest edition of SQL Server that includes this is the Standard edition. For App-V, we also need to use Standard edition because App-V makes use of SQL Agent to schedule jobs and run background tasks, and this is not included in the free Express edition.

Office 2013 and Office 365

Office 365 is the pay per user per month version of Office and can also include SharePoint, Lync, and Exchange Online. The higher-level plans allow for the deployment of desktop versions of Office to up to five devices per user; however, Office 365 cannot be licensed for any kind of shared desktop solutions such as the Session and Pooled Collections in Remote Desktop Services, so currently we can only use it for personal collections assigned to individual users. That leaves Office 2013, which can still be used in VDI with either Standard or Professional Plus editions. The difference between the two editions is that Pro Plus adds Business Intelligence and Lync to the basic products for unified communication.

Third-party VDI solutions

This book has been all about how to use VDI without using third-party technologies, notably Citrix, Dell, and of course VMWare. There is some advantage in laying these technologies over VDI, notably the ability to manage VDI at scale, but all these solutions will incur significant costs as well; a detailed examination of these would fill another book or two. All I want to mention here is that if the OS provided to the user is Windows-based, then the requirements for licensing also apply. So, if we build a VMWare View environment where the Guest OS is a Windows client, then SA, VDA, and CSLs will be needed in exactly the same way as if the solution uses Microsoft's VDI. One outcome of this is that VMWare recommends using Windows Server as the Guest OS in the VM-based virtual desktops as the licensing is cheaper and easier to manage. Note that this is *not* the same as session-based collections, where a single physical machine or VM has many users sessions; it is one VM per user (Pooled or Personal to that user), but running Windows Server with the desktop experience enabled to make it look like Windows Client and have things such as video playback enabled.

The future of VDI – Desktop as a Service

Throughout this book, we have configured our own infrastructure for VDI, but as with many other IT services, there is a push to provide VDI from third-parties, either hosting or cloud providers, to provide agility and scalability while reducing the management overhead of running VDIs. There are several ways we might achieve this—we could use a public cloud provider such as **Amazon Web Services** (**AWS**) or Microsoft Azure, or we could simply get a hosting provider to run our customized desktops for us. AWS now includes a service to do this, and this can include Office as well, and there will at some point be a similar Desktop as a Service on Microsoft Azure. However, hosting providers must use the **Service Provider Licensing Agreement** (**SPLA**) to pass on Microsoft licensing to end users, and currently this does not include Office 365 or Windows Client. So while the Amazon service might look like Windows 7, it's actually based on Windows Server 2008 R2 with the desktop experience added, but it is based on a VM per user and not Session Virtualization. In fact, none of Microsoft's license agreements for Windows Client or Office 365 allow for this kind of deployment, so even if we could use a cloud service to set up VDI, we will have to use the server OS in the same way that Amazon does today if we are to be license compliant or use Session Virtualization.

If we have software assurance, we can use RDS CALs to access these third-party solutions. For more details on this, refer to the Microsoft Product Use Rights (`http://www.microsoft.com/licensing/about-licensing/product-licensing.aspx`).

In either case, your users will want to sign is as they normally would—with their domain credentials—and when they log in they will want to access corporate data and resources. This means that we will have to create some sort of directory trust or synchronization and probably a site-to-site VPN.

Given these licensing restrictions, the use of the cloud for VDI is rather limited, but there are one or two use cases where this sort of approach is used and a good example of this is how Microsoft now provides Visual Studio Online so that developers can collaborate and work remotely on any device capable of running the Microsoft RDP client to design and test software. The code can be kept in the cloud by the project team and shared without letting the code leak to a local device, thus protecting intellectual property across teams of contractors while allowing collaboration. We could do the same thing with our software and possibly offer this as an RD RemoteApp to third-parties or internal teams, particularly if it's niche or legacy software. This can be particularly effective if the data is collocated in the cloud, for example, if we were to provide desktops with specialized tools to access Big Data in solutions such as Microsoft's HDInsight, which is based on Hadoop running on potentially hundreds of VMs in Azure.

Clearly, this is a fast-changing area so it's worth checking new offerings from cloud providers such as Amazon and Microsoft, especially as the release cycle and pricing models change far more rapidly than with traditional locally installed software and services.

Summary

In my opinion, licensing was the hardest part of a Microsoft-based VDI solution and is one of the reasons why it was not widely adopted in the past. However, the rules have changed with things like VDA. With the built-in benefits that come with SA and CSL, it's pretty straightforward to implement a BYOD strategy. There is still a lot of confusion, doubt, and restrictions when using a third party to host VDI, but this is the fastest changing part of IT, with new updates typically coming out every month, meaning that new possibilities may emerge that will completely change the rules. So it's even more important to be up to date, especially as Desktop as a Service is now the hot topic across the industry and not just with Microsoft.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Thank you for buying

Getting Started with Windows VDI

About Packt Publishing

Packt, pronounced 'packed', published its first book "Mastering phpMyAdmin for Effective MySQL Management" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike.
For more information, please visit our website: `www.packtpub.com`.

About Packt Enterprise

In 2010, Packt launched two new brands, Packt Enterprise and Packt Open Source, in order to continue its focus on specialization. This book is part of the Packt Enterprise brand, home to books published on enterprise software – software created by major vendors, including (but not limited to) IBM, Microsoft and Oracle, often for use in other corporations. Its titles will offer information relevant to a range of users of this software, including administrators, developers, architects, and end users.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to `author@packtpub.com`. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

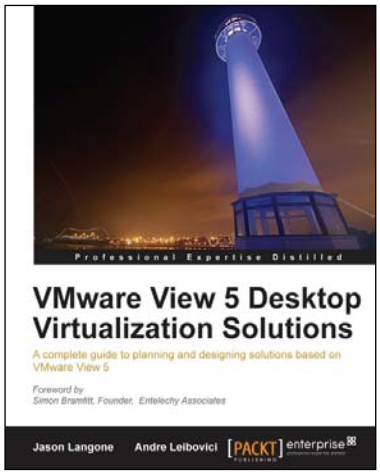
[PACKT] enterprise
professional expertise distilled
PUBLISHING

Getting Started with Citrix VDI-in-a-Box

ISBN: 978-1-78217-104-1 Paperback: 86 pages

Design and deploy virtual desktops using Citrix VDI-in-a-Box

1. Design a Citrix VDI-in-a-Box solution.
2. Get the budget for Citrix VDI-in-a-Box by building a case.
3. Implement a Citrix VDI-in-a-Box proof of concept and Citrix VDI-in-a-Box solution.



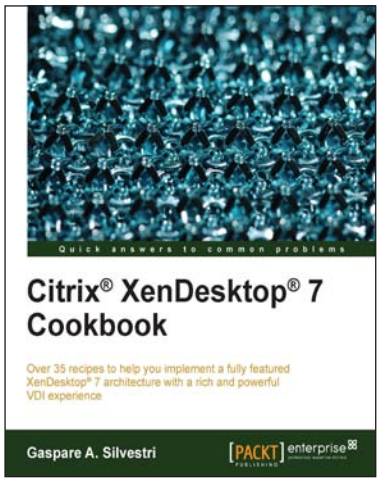
VMware View 5 Desktop Virtualization Solutions

ISBN: 978-1-84968-112-4 Paperback: 288 pages

A complete guide to planning and designing solutions based on VMware View 5

1. Written by VMware experts Jason Langone and Andre Leibovici, this book is a complete guide to planning and designing a solution based on VMware View 5.
2. Secure your Visual Desktop Infrastructure (VDI) by having firewalls, antivirus, virtual enclaves, USB redirection and filtering, and smart card authentication.
3. Analyze the strategies and techniques used to migrate a user population from a physical desktop environment to a virtual desktop solution.

[PACKT] enterprise
professional expertise distilled
PUBLISHING

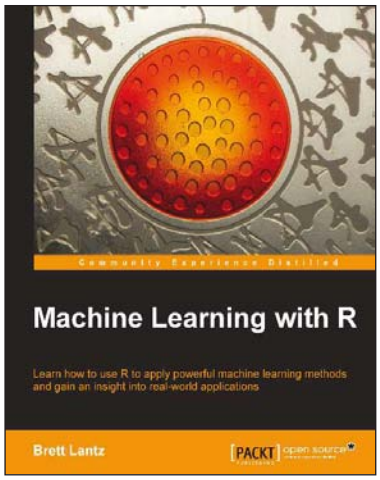


Citrix® XenDesktop® 7 Cookbook

ISBN: 978-1-78217-746-3 Paperback: 410 pages

Over 35 recipes to help you implement a fully featured XenDesktop® 7 architecture with a rich and powerful VDI experience

1. Implement the XenDesktop 7 architecture and its satellite components.
2. Learn how to publish desktops and applications to the end user devices, optimizing their performance and increasing the general security.
3. Designed in a manner which will allow you to progress gradually from one chapter to another or to implement a single component only referring to the specific topic.



Machine Learning with R

ISBN: 978-1-78216-214-8 Paperback: 396 pages

Learn how to use R to apply powerful machine learning methods and gain an insight into real-world applications

1. Harness the power of R for statistical computing and data science.
2. Use R to apply common machine learning algorithms with real-world applications.
3. Prepare, examine, and visualize data for analysis.