Windows Server 2012 R2 Administrator Cookbook

Over 80 hands-on recipes to effectively administer and manage your Windows Server 2012 R2 infrastructure in enterprise environments

Jordan Krause
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 “Learning the Interface”
- A synopsis of the book’s content
- More information on Windows Server 2012 R2 Administrator Cookbook

About the Author

Jordan Krause is a Microsoft MVP in the Enterprise Security group. He has the unique opportunity to work with the Microsoft networking technologies daily as a senior engineer at IVO Networks. Jordan specializes in Microsoft DirectAccess and has authored Microsoft DirectAccess Best Practices and Troubleshooting, Packt Publishing, one of the few books available worldwide on this subject. His additional technical expertise includes Microsoft Unified Access Gateway (UAG) and Microsoft Threat Management Gateway (TMG). He spends the majority of each workday planning, designing, and implementing DirectAccess for companies around the world.

Committed to continuous learning, Jordan holds Microsoft certifications as an MCP, MCTS, MCSA, and MCITP Enterprise Administrator. He regularly writes tech notes and articles reflecting his experiences with the Microsoft networking technologies; these can be found at: http://www.ivonetworks.com/news.

Jordan also strives to spend time helping the DirectAccess community, mostly by way of the TechNet forums. Always open to direct contact, he encourages anyone who needs assistance to head over to the forums and find him personally. Jordan lives and works in the ever-changing climate that is Michigan.
Windows Server 2012 R2
Administrator Cookbook

I still remember the day that I was first thrown into the server world. I had been working with Desktop computers for years, even had quite a bit of experience with other items such as printers, copy machines, and phone systems. Technology was familiar to me, Microsoft Windows was familiar to me, but the company where I worked was large enough that Desktop support and Server support were separated, and there wasn't a lot of technical interaction between the sides. Other than office pranks and lots of humorous emails forwards flying around, naturally. So when I had a sudden chance to change jobs squarely into a server support role, I was a little hesitant at first. When I found out my first task would be implementing DFSR, and I thought that stood for "Digital something-or-other", I may have been a little freaked out. I start with this little story not to try to make you chuckle at my expense, but because, if we fast forward to several years down the road, I suddenly find myself with an opportunity to put together a book on the new version of this incredible operating system, Windows Server 2012 R2.

Yes, this is getting to the point. There were many different angles from which the recipes contained within these pages could have come. Do we stick to features that are brand-new in Windows Server 2012 R2 and not in previous versions of the operating system? How advanced should the topics be? What about an inexperienced admin who is reading this book? What about those who are trying to get familiar with the server world because they don't have the opportunity to have hands-on experience of it during their day job? These questions, combined with my own situation as described above have led to my writing the hundreds of pages that make up this book.

What would I have wanted to know when I started working with servers? What are some of the things that I have learned over the years that I still continue to pass down to new customers that I work with? How can I make use of this new Server operating system from Microsoft in my own company? What are some of the lesser known topics in Server 2012 R2 that I myself am interested in digging deeper into? These kinds of questions became the new baseline for putting together recipes for this book. I sincerely hope that anyone who reads this—from the fresh, brand-new server administrator who has nothing but a server to learn on, to the tenured, experienced admin who is coming up to speed on the latest iteration (and certainly the best ever Windows Server operating system)—can all take something out of this text and apply it to their jobs and lives in a way that benefits them.
The beauty of the Windows Server operating system has always been how incredibly vast it is, and Server 2012 R2 is certainly no exception. There are loads of different roles, features, and functions that you can run in order to do almost anything within your network. Need a file server? Check. Directory services to house your user and computer accounts? Look no further. Secure remote access solution? Yes sir. Need to publish certificates for a website that you want to present to the internet? Yup, we can even do that if we so desire.

We are going to start with some recipes that get us familiar with the graphical interface of Server 2012 R2. I have the unique opportunity to work with new admins almost daily, from all kinds of different companies and corners of the world. Navigating around inside the Server 2012 and Server 2012 R2 systems is still a struggle that I see constantly. Once we know how to get around comfortably, we will move on to the configuration and utilization of core infrastructure technologies that are absolutely need-to-know information for any admin interested in working within a Microsoft network. Following that, we will dig down into some of the individual facets of technology that are provided by this amazing operating system. I will let the chapters speak for themselves, but I strive with these recipes to cover a lot of different bases, and to keep things interesting as we round those bases.

We will certainly discuss new material and features that are only available in this newest version of Windows Server, but additionally there will be recipes that apply not only in R2 but also to systems across your entire network. By starting to use these functions that may even be new to an experienced admin, the hope is that your daily chores and tasks can become more streamlined and efficient along the way.

PowerShell is also discussed and scattered throughout this book. At one point, there was an intention for a separate chapter dedicated to PowerShell but, as the recipes came together, it became clear that they should not stand on their own but ought to be embedded within the chapters to which they apply. PowerShell is fully integrated into every aspect of Windows Server 2012 R2. Any task that you want to perform, any task, can be performed via a PowerShell command or script if you so choose. It is the core of the way that Server 2012 R2 is built. In fact, many of the things that you configure in the GUI are really just calling for pre-built PowerShell scripts to run in the background to do the actual configuration. Because PowerShell is integral to the entire operating system, we incorporated it throughout this book rather than lumping it all together.

By the time you reach the end of this book, my hope is that even someone completely unfamiliar with the Windows Server operating system will have the ability to build their own Microsoft-centric network from the ground up. Even though the recipes are grouped together in chapters and follow a fairly logical sequence, there is no need to read this front-to-back. Each recipe has the power to stand on its own, and to be useful with or without the recipes that precede it. Feel free to skip around, grab the data you are currently interested in, or use this book as a reference guide in the future. Enjoy!
What This Book Covers

Chapter 1, Learning the Interface, starts us on our journey working with Windows Server 2012 R2 as we figure out how to navigate the look and feel of this new operating system, and gain some tips and tricks to make our daily chores more efficient.

Chapter 2, Core Infrastructure Tasks, will take us through configuring and working with the core Microsoft technology stack. The recipes contained in this chapter are what I consider essential knowledge for any administrator who intends to work in a Windows network.

Chapter 3, Security and Networking, teaches us some methods for locking down access on our servers. We will also cover commands that can be very useful tools as you start monitoring network traffic.

Chapter 4, Working with Certificates, will start to get us comfortable with the creation and distribution of certificates within our network. PKI is an area that is becoming more and more prevalent, but the majority of server administrators have not yet had an opportunity to work hands-on with them.

Chapter 5, Internet Information Services, brings us into the configuration of a Windows Server 2012 R2 box as a web server in our network. Strangely, in the field I find a lot of Microsoft networks with Apache web servers floating around. Let's explore Internet Information Services (IIS) as a better alternative.

Chapter 6, Remote Access, digs into using your Server 2012 R2 as the connectivity platform that brings your remote computers into the corporate network. We will discuss DirectAccess and VPN in this chapter.

Chapter 7, Remote Desktop Services, encourages you to look into using Server 2012 R2 as a virtual session host or VDI solution. RDS can be an incredibly powerful tool for anyone interested in centralized computing.

Chapter 8, Monitoring and Backup, covers some of the capabilities included with Server 2012 R2 to help keep tabs on the servers running in your infrastructure. From monitoring system performance and IP address management to backing up and restoring data using the tools baked into Windows, these recipes will walk you through some helpful tasks related to monitoring and backup.

Chapter 9, Group Policy, takes us into the incredibly powerful and far reaching management powers contained within Active Directory that are provided out of the box with Windows Server 2012 R2.

Chapter 10, File Services and Data Control, provides us with information and step-by-step recipes on some of the lesser known ways that data can be managed on a Windows server. We will cover technologies such as DFSR, iSCSI, and Server 2012 R2 Work Folders.
In an effort to become familiar with the look and feel of Windows Server 2012 R2, you will learn how to navigate through some daily tasks using the graphical interface. On our agenda in this chapter are the following recipes:

- Shutting down or restarting the server
- Launching Administrative Tools
- Using WinKey + X for quick admin tasks
- Using the Search function to launch applications quickly
- Managing remote servers from a single pane with Server Manager
- Using PowerShell to accomplish any function in Windows Server
- Installing a role or feature
- Administering Server 2012 R2 from a Windows 8.x machine
- Identifying useful keyboard shortcuts in Server 2012 R2
- Setting your PowerShell Execution Policy
- Building and executing your first PowerShell script
- Searching for PowerShell cmdlets with Get-Help
Introduction

I meet new server administrators almost daily, from all types of businesses and walks of life. Often during our projects, we introduce the very first Windows Server 2012 R2 server into an environment. This gives me the unique opportunity to observe first interactions with the updated interface, and most of the time first impressions are pretty terrible. Now don't get me wrong, after using Windows Server 2012 R2 for only a little while, I think you'll all agree that it has some of the best functionality and stability we have ever seen out of a server operating system. What I see people struggle with all the time when starting to use 2012 R2 is the user interface. This is the same challenge that has plagued Windows 8 since its release. Everything seems to look and feel different than the Windows we have always known, trying to find applications can be futile, and what in the world is that "tablet-touch-screen-wannabe-thing" that consumes my entire screen all the time? And the charms bars that pop out? Don't get me started.

These big changes in the way that we interact with the operating system can be a nuisance on a client Desktop machine, but can be downright dangerous on a server until you have a good handle on managing the interface. When you are working within three levels of Remote Desktop Protocol (RDP), bouncing from one server to another, all of these little differences are compounded. It suddenly becomes difficult to know which server it is that you are changing. Let's have a show of hands, how many of you have mistakenly rebooted the wrong server? Or even more likely, how many of you have rebooted your own computer while you were trying to reboot a remote server? I know I have! And not just once.

Hope is not lost! I promise you that once you learn to manage the interface, rather than letting it manage you, some of these changes may start to seem like good ideas. They can increase productivity and the ease of accomplishing tasks—we just need some pointers on making the best use of the new interface.

The recipes in this chapter are dedicated to doing just that. Let's work together to gain a better understanding of why the interface was built the way it is, and learn to take advantage of these new screens and settings.

Shutting down or restarting the server

I just couldn't resist starting with this one. Yes, this seems trivial. Silly even. However, the amount of times that I have watched a simple server restart consume more mouse clicks than creating a Domain Controller has convinced me that this needed to be in the book. Perhaps the shutdown and restart options were hidden away purposefully, because once your system is up and running, there is not often a need to accomplish either of these tasks. When first configuring the box though, it is very common to have to reboot a couple of times, or to shut down a machine to move it to another location.
Getting ready

To step through this recipe, you will need a Windows Server 2012 R2 system online. There are no other prerequisites.

How to do it...

Let's take a look at three different ways to shut down or restart your system. The first two are the most commonly employed. The third is less commonly known, but is by far my preference when tasked with restarting a remote server.

The first option would be to use the power button on the Start screen. So far, I don't think that many administrators have stumbled into this one, because it didn't exist in Server 2012.

1. Click on your "Start Button", or rather your button in the Taskbar that opens your Start screen. You will see a power icon in the top-right corner of that Start screen.
2. Click on this for Shut down and Restart options.

The second method to accomplish shutting down or restarting is by using the charms bar. The charms bar is the little menu that presents itself when you have your mouse sitting near the upper-right corner of your screen. On a touchscreen computer, you swipe your finger from the right side of the screen toward the middle to make it appear. Go ahead and hover the mouse in the upper-right corner of your screen until the charms bar appears. Then click on Settings, and then on Power. Unfortunately, this is the most common way to attempt to restart a Server 2012 R2 machine. I say unfortunately because it is so easy to hover in the corner and display the wrong charms bar, the one for your own Desktop for example.
Learning the Interface

Many mistaken reboots are caused by this.

The most definitive, and dare I say most fun way of restarting your server is to utilize Command Prompt. Doing this gives you the opportunity to double check that you are manipulating the correct machine. Open up Command Prompt and run a quick hostname check to make sure you are restarting the one you really intend to. Then utilize the shutdown command to take care of the rest. This process can be especially helpful when logged into remote servers using RDP. Use the following commands to perform the explained operations:

```
hostname
shutdown /r /t 0
```

If you were to simply type `shutdown`, the server would shut itself down in 60 seconds. Using `/r` indicates a restart rather than a shut down, and `/t 0` is a timing flag that indicates the number of seconds the server should wait before restarting. Specifying slash zero here tells it to wait for zero seconds before initiating the restart.
Chapter 1

How it works...

Shutting down or restarting a server doesn't require a lot of explanation, but I hope that this small recipe gets some thought going about creative ways to do regular tasks. As you will see throughout this book, you can accomplish anything in Windows Server 2012 R2 through the use of commands or scripts. You could easily turn the `shutdown` command, the last example that we tested in this recipe, into a batch file, and place it on the Desktop of each of your servers as a quick double-click option for accomplishing this task.

However, I work with RDP windows inside RDP windows very often. When you're bouncing around between a dozen servers that all have the same background, I have decided that the only sure-fire way to make sure you are restarting the correct device is to do a quick `hostname` check before you initiate the restart. If you are interested in discovering all of the available flags that are available to use with the `shutdown` command, make sure to type in `shutdown /?` some time to take a look at all of the available options.

Using the command prompt is also an easy way to log off a server. Let's say you are layers deep in RDP and want to log off from a single server (not all of them). Are you sure you clicked on the `Start` button of the right server? Instead, open up a prompt and simply type `Logoff`.

Launching Administrative Tools

Earlier versions of Windows Server placed all of the Administrative Tools in a self-named folder right inside the Start menu. Server 2012 R2 no longer has a Start menu, but rather a full Start screen. It is my experience that administrators prefer to stay on the Desktop as much as possible, and do not like switching back and forth between the two. In other words, they want to accomplish everything without having to use the Start screen. Let's take a look at some ways to launch Administrative Tools right from your server's familiar Desktop interface.

Getting ready

All you really need is a Windows Server 2012 R2 machine online. The more roles and services that you have running on it, the more options that you will see on your screen as we navigate these menus.
Learning the Interface

How to do it...

To launch Administrative Tools from your Desktop, perform the following steps:

1. Open up **Server Manager**. In fact, if you just logged into the server, it's probably already open for you.
2. Click on **Tools** in the upper-right corner.

There you go. A full list of all the admin tools installed onto that server. Heading into this list is also a quick way of taking a look into what a particular server is doing, which you can take an educated guess at based on what roles and services are installed. By looking at the screenshot below, we can see that this is a pretty standard server instance that is not yet running any big roles. Once added, the management tools for the Windows roles will present themselves in this list.
**How it works...**

Since Server Manager likes to open automatically when logging in, let's make quick use of it for opening the tools that we need to do our jobs. Another way to have easy access to your tools from the Desktop is to create shortcuts, or to pin each of them to your Taskbar. Sometimes this isn't as easy as it sounds. In the past, these tools were all grouped together in the Administrative Tools folder, so you didn't have any reason to memorize the exact names of the tools. If you were to head into the Start screen and try to recall the name of the service you are looking for to type into the Search field, it may not immediately come to you. If you're a consultant working on someone else's server, you may not want to pin anything to their Desktop anyway, and you certainly don't want to resort to using Bing in front of them to look up the name of the service. So I like to stick with launching Administrative Tools from Server Manager.

**Using WinKey + X for quick admin tasks**

There are some functions in Windows that a server administrator needs to use all the time. Instead of making shortcuts or pinning them all to the Taskbar, let's get to know this hidden menu that is extremely useful for launching these commonly used admin tools.

**Getting ready**

A running Windows Server 2012 R2 machine is all we need to highlight this one. In fact, this menu also exists on any Windows 8.x computer, so make use of it often!

**How to do it...**

There are two ways to open this little menu. While you are in the Server 2012 R2 Desktop, you can perform either of these steps:

- Hold down your Windows key on the keyboard and press X
Hover your mouse over the Windows flag in the lower-left corner of the Desktop. You know the button that opens the Start screen if you left click on it. Now go ahead and right-click on that button and you will see a menu as shown in the following screenshot:

![Menu Screenshot]

How it works...

This little "quick-tasks admin menu" is very easy to open, and is very convenient for launching programs and settings that are accessed often. I won't talk too much about what particulars are in the menu as it's pretty self-explanatory, but I use this menu multiple times per day to open up the System properties and the command prompt. Particularly the command prompt, as it has an option to open an administrative command prompt right from the menu.

Look at that, you can also shut down the server from here!
Using the Search function to launch applications quickly

We are finally going to make some use of that Start screen. While many server administrators prefer to work outside of this screen 100 percent of the time, the powerful Search functions of the Start screen are definitely something that you should be aware of.

Getting ready

For this recipe, you will need a Windows Server 2012 R2 system online.

How to do it...

To search in Windows Server 2012 R2, we have to make our way to the Start screen. Fortunately, this is extremely easy to do. No matter where you are or what application you are in, if you press the Windows key or click on the Start button, you will be taken to the Start screen.
Now you simply start typing. Search results for anything found on the system that relates to the word you are inputting are automatically displayed. Once you see the item you are looking for, you can press Enter or click on it. Additionally, if you right-click on the item or hold down the Ctrl key while clicking it, you will receive additional options that you can perform with this application. For example, right-click or Ctrl click in order to launch command prompt as an administrator.

By default, your search will encompass the entire operating system. If you would rather narrow the scope of your search down to only Settings or Files, you can use the Everywhere dropdown and choose accordingly.
You can see it is very easy not only to open applications, but also its specific settings.
Learning the Interface

How it works...

From the Start screen, we can search for anything on the server. This gives us the ability to quickly find and launch any program or application that we have installed. This includes Administrative Tools. Rather than moving into Server Manager in order to launch your administrative consoles from the Tools menu, you can also search for them on the Search screen, and launch from there. It also gives us the ability to find files or documents by name. Another powerful way to use the Search function in Windows Server 2012 R2 is to open any kind of setting that you might want to change. In previous versions of Windows, you had to either memorize the way to get into the settings that you wanted to change or you had to open up Control Panel where you had to poke and prod your way around until you stumbled upon the one that you were looking for. Now it is a very simple matter of pressing the Windows key, typing the first few characters of the setting or program you want to launch, and pressing Enter.

Another common task to perform from the Search screen is to right-click on the application that you are trying to launch and pin it somewhere. When you right-click on a program from the Search screen, you see options to pin the program to either your Start screen or to the Taskbar. This will create a quick-launch shortcut on either the main Start screen or on the Taskbar of the Desktop mode, giving you easier and faster access to launch those applications in the future.

Managing remote servers from a single pane with Server Manager

As you have already noticed, Server Manager has changed significantly as of Server 2012, and has been made even better in Server 2012 R2. Part of these changes are a shift in mindset where emphasis is now placed on remote management of servers. Server Manager in Windows Server 2012 R2 can be used to manage and administer multiple systems at the same time, all from your single pane of glass, the monitor where you are sitting. In this recipe, you are going to learn how to manage both the local server we are logged into, as well as a remote server, from the same Server Manager window.

Getting ready

For this recipe, we need two servers. One is the machine we are physically logged into. Another is a server on the same network that we can contact from our primary server, so that we can manage it from our local Server Manager.
How to do it...

To manage a local as well as a remote server from the same Server Manager window, perform the following instructions:

1. Log in to your primary server and launch **Server Manager**. You will see in the upper-left corner that the only server you have listed is the Local Server that we are logged into.

![Server Manager](image)

2. Now head over toward the top-right of Server Manager and click on the **Manage** button. In this menu, click on **Add Servers**.

![Add Servers](image)
3. If your servers are part of a domain, finding remote machines to manage is very easy by simply selecting them from the default **Active Directory** tab. Since mine are not yet part of a domain, I need to head over to the **DNS** tab and specify the name or IP address of the server that I want to connect to.

4. Now if you go ahead and click on **All Servers** in the left window pane, you will see the additional servers listed that you have selected. If you double-click or right-click on those remote server names, you have many options available to you to remotely manage those machines without having to log into them.
How it works...

Server Manager makes use of the Windows Remote Management (WinRM) tools to remotely manipulate servers. Historically, most of us who are administering Windows Servers make extensive use of RDP, often having many windows and connections open simultaneously. This can cause confusion and can lead to tasks being accomplished on servers for which they are not intended. By using Server Manager from a single machine to manage multiple servers in your network, you will increase your administrative efficiency as well as minimize human error by having all management happen from a single pane of glass.

This recipe is written with the most common network scenario in mind, which is a domain environment where both servers have been joined to the domain. If you are working with standalone servers that are part of a workgroup, rather than being joined to a domain, you will have some additional considerations. In the workgroup scenario, WinRM will need to be enabled specifically, and the Windows Firewall will have to be adjusted in order to allow the right ports and protocols for that WinRM traffic flow to happen successfully. In general though, most of you will be working within a Microsoft domain network, in which case these items are not necessary.

See also

- The Administering Server 2012 R2 from a Windows 8.x machine recipe

Using PowerShell to accomplish any function in Windows Server

An incredibly powerful tool in Windows Server 2012 R2 is PowerShell. Think of PowerShell like a command prompt on steroids. It is a command-line interface from which you can manipulate almost anything inside Windows that you may care to. Better yet, any task that you may wish to accomplish can be scripted out in PowerShell and saved off as a .ps1 script file, so that you can automate large tasks and schedule them for later, or at regular intervals. In this recipe, let's open up PowerShell and run some sample commands and tasks just to get a quick feel for the interface. In a later chapter of the book, we will do some more specific tasks with PowerShell to go even deeper into the technology.

Getting ready

To start using PowerShell, all you need is a server with Windows Server 2012 R2 installed. PowerShell is installed and enabled by default.
To get a feel of using PowerShell, perform the following steps:

1. Right-click on your blue PowerShell icon and make sure to open PowerShell as administrator by clicking on Run as administrator. PowerShell is very powerful, but only if you have admin rights in the session.

2. Test out some commands that you are familiar with from using the command prompt, such as `dir` and `cls`. Since you are able to make use of these familiar commands, PowerShell can really be your one and only command-line interface if you choose.

3. Now let's try some of the PowerShell secret sauce, one of its cmdlets. These are special commands that are built into Windows and allow us to do all kinds of information gathering, as well as manipulation of server components. Let's start by pulling some data. Maybe take a look at what IP addresses are on the system with `Get-NetIPAddress`.

4. The previous command probably gave you a lot more information than you needed, since most companies don't make use of IPv6 inside their network yet. Let's whittle this information down to the IPv4 specific info that you are most likely interested in. Enter `Get-NetIPAddress -AddressFamily IPv4` to attain it.
PowerShell has so many commands and cmdlets, we just wanted to get a feel for launching the program and pulling some data with this particular recipe. There are countless Get commands to query information from the server, and as you have seen those cmdlets have various parameters that can be appended to the cmdlets to pull more specific data to meet your needs. To make things even better, there are not only Get cmdlets, but also Set cmdlets, which will allow us to make use of the PowerShell prompt to configure many aspects of the configuration on our server, as well as remote servers. We will dive further into PowerShell in a later chapter.

### Installing a role or feature

You've installed the Windows Server 2012 R2 operating system onto a piece of hardware. Great! Now what? Without adding roles and features to your server, it makes a great paper weight. We're going to take the next steps here together. Let's install a role and a feature into Windows so that we can start making this server work for us.

### Getting ready

As long as you have a Windows Server 2012 R2 installed and running, you are ready to install roles and features onto that machine.

### How to do it...

To install a role and a feature into Windows, perform the following steps:

1. Open **Server Manager**. In the middle of the screen, you'll see a link that says **Add roles and features**. Click on that link.

2. Click **Next** on the first summary screen and you will come to a choice on the second page. For most roles and features, we want to leave it set at the top bullet, which is **Role-based or feature-based installation**. If we were configuring Remote Desktop Services, which we will discuss in another chapter, then we would choose the second option.
Now we choose where we want to install a new role or feature. This is a neat page, as we can choose from any server that we have added into our Server Manager, or we can even choose to install a role or feature into a virtual hard disk. I am running the **Add Roles Wizard** from DC-01, but I want to install the IIS role onto WEB-01. Rather than having to log into WEB-01 to accomplish this task, I will do it right from here. In the screenshot below, you can see WEB-01 listed as a server that I can install a role onto, even though I am opening this console on the DC-01 server.

3. Scroll down and choose the role that you want to install. For DC-01, I am choosing the **Web Server (IIS)** role. Then click **Next**.

![Select a server or a virtual hard disk on which to install roles and features.](image)

- Select a server from the server pool
- Select a virtual hard disk

Server Pool

<table>
<thead>
<tr>
<th>Name</th>
<th>IP Address</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC-01.MYDOMAIN.LOCAL</td>
<td>192.168.0.1</td>
<td>Microsoft Windows Server 2012 R2 Standard</td>
</tr>
<tr>
<td>WEB-01.MYDOMAIN.LOCAL</td>
<td>192.168.0.3</td>
<td>Microsoft Windows Server 2012 R2 Standard</td>
</tr>
</tbody>
</table>

2 Computer(s) found

Some roles require additional components to be installed for them to work properly. For example, when I chose to install the IIS role and clicked **Next**, I was prompted about needing to install some management tools. Simply click on the **Add Features** button to automatically add the items that it needs to perform correctly.
4. Now choose any features that you would like to install. For example, in order to do some network connectivity testing later, go ahead and select **Telnet Client** from the list.

5. Read and click **Next** through the informational messages that are displayed. These messages will vary depending on which roles and features you have installed.

6. The final screen is your installation summary. If everything looks correct, go ahead and click on **Install**.

After your roles and features have finished installing, the server may or may not have to reboot. This depends on whether or not the role installation requires it. Following installation, or following the reboot, if your new role needs any additional configuration or setting up to be completed, you will be notified at the top of the Server Manager screen.

**How it works...**

Adding roles and features to a Windows Server is something that every administrator will have to do sooner or later. These items are necessary to turn on the functions in the server that will perform tasks that need to be performed in your environment. Adding roles is quite straightforward. However, it is interesting to see the options that are available to add more than one role or feature at a time. Moreover, the ability to remotely install these items for servers in your network that you are not logged into is intriguing.

**Administering Server 2012 R2 from a Windows 8.x machine**

In the *Managing remote servers from a single pane with Server Manager* recipe, we already discussed remotely administering another server by using Server Manager. Did you know we can accomplish the same remote management by using our day-to-day Windows 8.x computer? We will install and use the **Remote Server Administration Tools (RSAT)** to take even more advantage of Server 2012 R2's remote management ideology.

**Getting ready**

To test out the RSAT tools, we will need a Windows 8.x client machine. We will then also need a Windows Server 2012 R2 system online, and on the same network, which we can remotely control and manage.
To remotely manage a server using RSAT, perform the following instructions:


After you install these tools onto your Windows 8.x computer, you should now have a copy of Server Manager installed onto your computer. Go ahead and launch that from the Start screen. You can pin it to your Taskbar for quicker launching in the future, of course. In the same fashion, as with Server 2012 R2, you can use the Manage menu to add servers to Server Manager.

For this recipe, I do have the machines we are working with joined to a domain, so we will take a look at adding servers that are part of the domain.

2. Click on the Find Now button and you will see a list of server names that are remotely manageable.
3. Click on the server names that you want to administer and click on the arrow to move them over to the right side of the screen. Upon clicking on **OK**, you will see these new servers listed and ready for management inside your Server Manager console.

![Server Manager screenshot](image)

**How it works...**

Server Manager in Windows Server 2012 R2 is a powerful tool that can be used for the management of not only the local server, but also remote servers that you want to manage. If we take this even a step further and install the RSAT tools on a Windows 8 computer, this gives us the ability to launch and use Server Manager from our everyday Windows 8 computer. In doing so, we enable ourselves the ability to add roles, view events, and restart servers, all from our own desk. Managing servers using these tools will increase productivity and decrease errors, because your entire infrastructure of servers can be available within a single window. This is much more efficient than using the RDP client to connect to many different servers, all in different windows. If you've never tried using RSAT to manage servers, give it a try!

**See also**

- The *Managing remote servers from a single pane with Server Manager* recipe

**Identifying useful keyboard shortcuts in Server 2012 R2**

I prefer using a keyboard over a mouse any day, for almost any task. There are numerous keyboard shortcuts and tips and tricks that I employ on a daily basis and I want to test them out with you in this recipe. Some of these shortcuts have been around for years and will work with multiple versions of Windows Server; some are new in the Server 2012 R2 operating system. They will all be useful to you as you start working with servers in your network.
Getting ready

We are going to run these commands and keyboard shortcuts while logged into a Windows Server 2012 R2 machine.

How to do it...

- **Windows key**: Opens the **Start** screen where you can start typing to search for programs.
- **Windows key + X**: Opens the Quick Links menu, which we discussed in an earlier recipe.
- **Windows key + C**: Opens the main charms bar.
- **Windows key + I**: Opens the **Settings** charms bar specifically.
- **Windows key + D**: Shows or hides the Desktop. If you are in the **Start** screen, this is one way to get back over into the Desktop quickly.
- **Windows key + R**: Opens the **Run** box.
- **Windows key + M**: Minimizes all windows.
- **Windows key + E**: Opens File Explorer.
- **Windows key + L**: Locks the computer.
- **Alt + F4**: Exits the program you are currently working in. This is especially helpful in full screen apps launched from the Start screen, where it is not always obvious how to exit the program with your mouse.
- **Alt + Tab**: Displays a list of open programs so you can hop between them.
- **Shift + Delete**: Holding down **Shift** while pressing **Delete** deletes files without placing them into the Recycle Bin.
- **Using Tab inside command prompt or PowerShell**: I cannot believe that I went years without knowing about this one. When you are working inside command prompt, if you type the first letter of a file or folder that exists in the directory where you are working and then press the **Tab** key, it will auto-populate the rest of the filename. For example, you may be trying to launch a Microsoft update file with a filename that is 15 characters and comprises a mix of numbers and letters. No need to type out that filename! Let's say the file starts with ".K8". Simply navigate to the folder where your installer exists, type ".K8", and press **Tab**. The full filename is populated inside command prompt and you can press the **Enter** key to launch it.
How it works...

Keyboard shortcuts can greatly increase productivity once you are fluent with them. This is not an extensive list by any means, there are many more key combinations that you can use to launch apps, minimize and maximize windows, and do all sorts of other functions. This is a list to get you started with the most common ones that I employ often. Start using these with your daily tasks and I bet your mouse will start to feel lonely.

If you are interested in exploring more of the Windows Server 2012 R2 key combinations available, this website is a great place to start: http://technet.microsoft.com/en-us/library/hh831491.aspx.

Setting your PowerShell Execution Policy

To say that the Windows operating system can be manipulated by PowerShell is a gross understatement. They are fully intertwined, and PowerShell can be useful for so many tasks on your servers. However, the ability to run PowerShell scripts is disabled by default on many machines. The first stumbling block that many new PowerShell administrators bump into is the Execution Policy. It's quite simple, in order to allow PowerShell scripts to run on your server, the Execution Policy must be adjusted to allow that to happen. Let's introduce our first task in PowerShell by using some commands in this recipe that will set this policy for us.

This is also a good introduction to the idea of the verb-noun syntax that PowerShell utilizes. For example, we are going to make use of cmdlets called Get-ExecutionPolicy and Set-ExecutionPolicy. The Get-(parameter name) and Set-(parameter name) cmdlets are very common across all facets of cmdlets available in PowerShell. Wrap your mind around this verb-noun syntax and you will be well on your way to figuring out PowerShell on your machines.

Getting ready

We will be working within a PowerShell prompt on our Windows Server 2012 R2 box.
How to do it...

Follow these steps to set the PowerShell Execution Policy:

1. Right-click on the PowerShell icon from the Taskbar and choose Run as administrator.

2. Type Get-ExecutionPolicy and press Enter in order to see the current setting of the PowerShell Execution Policy.

3. You can see that the current execution policy is set to RemoteSigned. Here is a short description of the different options for the policy:
   - **Remote Signed**: This is the default setting in Server 2012 R2, which allows PowerShell scripts that are locally created to run. If you try running remote scripts, they must be signed by a trusted publisher in order to execute successfully.
   - **All Signed**: With this setting, all scripts will only be allowed to run if they are signed by a trusted publisher.
   - **Restricted**: With this setting, PowerShell is locked down so that scripts will not run.
   - **Unrestricted**: This setting will allow PowerShell to run scripts, with or without signing.

4. For the purposes of our recipe and to make sure scripts will run for us as we progress through these recipes, let's set our Execution Policy to unrestricted. Go ahead and use this command:
   
   Set-ExecutionPolicy Unrestricted
The PowerShell Execution Policy is a simple setting and easy to change, but can make a world of difference when it comes to running your first scripts. If configured to be more restrictive than you intend, you will have trouble getting your scripts to run and may think that you have mistyped something, when in fact the issue is only the policy. On the other hand, in an effort to make your servers as secure as possible, on machines where you don't need to execute PowerShell scripts, it makes sense to restrict this access. You may also want to read some additional information on the signing of scripts to see whether creating and executing signed scripts would make more sense in your own environment.

Building and executing your first PowerShell script

Command prompt and PowerShell are both great command-line interfaces that can acquire and configure information about our servers. Most of us are familiar with creating some simple batch files that are driven by command prompt, essentially programming out small tasks within these batch files to automate a series of commands. This saves time later as we do not have to type out the commands line by line, especially for common tasks or for items that we need to run during login.

PowerShell has similar functionality, the ability to write out multiple lines of PowerShell cmdlets inside a script file. We can then launch this script file as we would a batch file, automating tasks while taking advantage of the additional features that PowerShell brings to the table over command prompt. These PowerShell scripts are put together inside .ps1 files; let's build a simple one together to get a feel for running these scripts.
Learning the Interface

Getting ready

Our work with PowerShell today will be accomplished from a Windows Server 2012 R2 machine. PowerShell is installed by default with Windows, and there is nothing further that we need to install.

How to do it...

Follow these steps to build and execute our first PowerShell script:

1. Open the **Start** screen and type **Windows PowerShell ISE**. Right-click to launch this tool as an administrator. PowerShell ISE is an editor for PowerShell scripts that is much more useful than opening a simple text editor such as Notepad in order to build our script.

2. Choose **File | New** from the menus in order to open a blank .ps1 script file.

3. In your first line, type the following: `Write-Host "Hello! Here is the current date and time:"`.

4. From the toolbar menu, click the green arrow that says **Run Script**. Alternatively, you can simply press the **F5** button. When you run the script, the command and output are displayed in the lower portion of the ISE window.
Cool! Okay, so far it's actually pretty lame. Just reflecting the text that we told it to echo, but it worked. That is the nice thing about using the ISE editing tool rather than a generic text editor, you have the ability to quickly test run scripts as you make modifications.

5. Now let's add some additional lines into our script to give us the information we are looking for. You can see a list of available commands on the right side of the screen if you would like to browse through what is available, but for our example simply change your script to include the following:

```powershell
Write-Host "Hello! Here is the current date and time:"
Get-Date
Write-Host "The name of your computer is:"
hostname
```
6. Press the Run Script button again to see the new output.

![Image of script output]

7. Now use File | Save and save your new .ps1 PowerShell script out to the Desktop.

8. Let's test this script by launching it from within a real PowerShell command window. Right-click on your PowerShell icon in the Taskbar and choose Run as administrator.


![Image of PowerShell window]

**How it works...**

In this recipe, we created a very simple PowerShell script and saved it on our server for execution. While in practice getting time and date information from your server may come faster by using the stand-alone Get-Date cmdlet, we use this recipe to give a small taste of the ISE and to get your scripting juices flowing. Expanding upon the ideas presented here will start to save you valuable time and keystrokes as you identify more and more ways to automate the tasks and information gathering that are part of your daily routines. The possibilities behind PowerShell are practically limitless, so make sure that you open it up and start becoming familiar with the interfaces and tools associated with it right away!
Searching for PowerShell cmdlets with Get-Help

With this recipe, let's take a minute to use Get-Help inside PowerShell in order to, well, get some help! I see both new and experienced PowerShell administrators going to the Web a lot in order to find commands and the parameters of those commands. The Internet is great, and there is a ton of data out there about how to use PowerShell, but in many cases the information that you are looking for resides right inside PowerShell itself. By using the Get-Help cmdlet combined with the functions you are running or searching for, you might not have to open that web browser after all.

Getting ready

We will be running some commands from inside PowerShell on a Windows Server 2012 R2 machine.

How to do it...

To use the Get-Help function inside PowerShell, run the following steps:

1. Launch a PowerShell prompt.
2. Type Get-Help.
3. You're finished! No, I'm just kidding. Using Get-Help by itself will present you with some helpful data about the Get-Help command, but that's not really what we are looking for, is it? How about using Get-Help with a search parameter, like this:

   Get-Help Computer

   PS C:Users\...
   Name Category Module Synopsis
   -------- -------- ---------
   Add-Computer Cmdlet Microsoft.PowerShell.Management ...
   Checkpoint-Computer Cmdlet Microsoft.PowerShell.Management ...
   Disable-ComputerRestore Cmdlet Microsoft.PowerShell.Management ...
   Enable-ComputerRestore Cmdlet Microsoft.PowerShell.Management ...
   Get-ComputerRestorePoint Cmdlet Microsoft.PowerShell.Management ...
   Remove-Computer Cmdlet Microsoft.PowerShell.Management ...
   Rename-Computer Cmdlet Microsoft.PowerShell.Management ...
   Reset-ComputerMachinePassword Cmdlet Microsoft.PowerShell.Management ...
   Restart-Computer Cmdlet Microsoft.PowerShell.Management ...
   Restore-Computer Cmdlet Microsoft.PowerShell.Management ...
   Stop-Computer Cmdlet Microsoft.PowerShell.Management ...
   Test-ComputerSecureChannel Cmdlet Microsoft.PowerShell.Management ...
   Get-SiComputer Function SoftwareInventoryLogging ...

   PS C:Users\...
Learning the Interface

Cool! That searched the available cmdlets and presented us with a list of the ones that contain the word Computer. Nice.

4. Now what if we wanted to find out some more particular information about one of these cmdlets? Maybe about Restart-Computer, that sounds like something we might use often. Use the following command:

```
Get-Help Restart-Computer
```

Now we're really cooking! This is wonderful information. Basically, this is exactly what you would find if you were looking for information about the Restart-Computer cmdlet and went searching on TechNet for it.

How it works...

The Get-Help cmdlet in PowerShell can be used with virtually any command in order to find out more information about that particular function. I often use it when the specific name of a cmdlet that I want to use escapes my memory. By using Get-Help as a search function, it will present a list of available cmdlets that include the keyword you specified. This is a brilliant addition to PowerShell, and makes it so much more powerful than command prompt.

Also included with the Get-Help files are all of the special syntax and parameter options for each cmdlet that you might be working with. This saves you time from having to go to the Web in order to search for these functions, and it is just way more fun doing it at the command line than in a web browser.
Where to buy this book

You can buy Windows Server 2012 R2 Administrator Cookbook from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.