brief contents

1  ■  Before you begin       1
2  ■  Deploying the web server  12
3  ■  Exploring and launching a website  33
4  ■  Managing application pools  56
5  ■  Adding more websites to your server  71
6  ■  What every administrator should know about web applications  88
7  ■  Securing your sites and web applications  111
8  ■  Securing the server  133
9  ■  Protecting data with certificates  148
10 ■  FTP and SMTP with IIS  169
11 ■  Sharing administrative responsibilities through remote management  184
12 ■  Optimizing sites for users and search engines  199
13 ■  Building a web farm with Microsoft Network Load Balancing  219
14 ■  Building a web farm with Application Request Routing  231
15 ■  High availability for ARR using Microsoft NLB  245
16 ■  Sharing content and configuration to the web farm  250
17 ■  Sharing IIS configurations for a web farm  266
18  ■  Using the central certificate store for certificate management  278
19  ■  Web farm provisioning with the Web Farm Framework  288
20  ■  Disaster recovery for IIS  304
21  ■  The final exam  313
22  ■  Never the end  317
23  ■  IIS PowerShell cheat sheet  321
24  ■  Lab setup guide  327
Deploying the web server

I remember my first IIS server deployment. Back in those days it was a lot more difficult than it is now. Walking miles to the data center from my desk, in the snow, uphill both ways. Those were the days. You younger administrators have it easy.

Well, in fact, you don’t. Sure, the initial default installation of IIS is simple, and you may have already done that, but you probably didn’t get all the software components and configuration you needed. In fact for many years most administrators would click a button to install IIS and walk away, leaving the rest of the installation and configuration for an imaginary “someone else.” This method doesn’t result in a successful web server. In this chapter I help you get your install right. As an added bonus I also show you a few tricks I doubt you’ve heard of that can turn a deployment into an enjoyable task.

I also explore the placement of the web server in your infrastructure in this chapter and in the rest of the book. The physical location can affect website configuration, access to back-end applications such as databases, and firewall settings that in turn affect your security. Your infrastructure will play a large role in the success of your production web environment, so you need to think about where to locate the web server and how it will be protected. In this book’s labs you’ll install and begin to manage the WebBikez web server in Smalltown, U.S.A.

In this chapter you’ll learn about how the physical location will impact your web server and then move on to installing your first web server. You’ll learn how to install IIS to both a GUI-based server and Server Core. When the installation is complete, I show you how to test and verify that your installation was successful.

Let’s start at the beginning by locating IIS in your infrastructure.
2.1 **Locating and protecting the physical web server**

Location, location, location. The first rule of buying a home or starting a new business applies to your web servers as well. You should have your virtual environment ready to go for the labs in this book, but at some point you’re going to need to put a real web server into production.

The location of the web server in your infrastructure will impact how people from inside and outside your network access your websites. This will affect configuration settings on firewalls, the Network Address Translation (NAT) device, and DNS. The network team will make many of these decisions for you, but you’ll need to investigate your infrastructure so you can determine whether it meets the best security and performance strategy. As we dive deeper into performance and security, you may discover that you need to request changes to the infrastructure. Perhaps you’re the network team, and if that’s the case, as we move forward, I’ll take you through all of the concerns, configurations, and troubleshooting problems that might occur. I start gradually and add infrastructure information as we go. In this section I work through some basic infrastructure concepts:

- Where to locate your server and why it matters
- How to protect your server with a single firewall
- How to create and work in the middle tier with multiple firewalls

As you install and work with IIS during the next few chapters, I recommend you think about your infrastructure and what configuration impact may occur.

#### 2.1.1 Server location matters

Imagine connecting a web server directly to the internet with no protection of any kind. Wow! It’s a fun experiment that I don’t recommend. Several years ago I did that as a demonstration and it took less than an hour for it to be script-hacked. You protect your internal network with a cascade of devices, most commonly known as firewalls.

Properly locating your web servers requires an understanding of your network environment. I find it helps to have a picture of the network structure and devices. This picture should contain the following:

- The location and type of network devices
- The location of your clients and servers
- Most importantly, the communication path from the servers to the outside internet

You can start by hand-drawing your network on a piece of paper, as shown in figure 2.1. I often do this for a quick visual guide.

For larger and more complicated networks I prefer to use a good visual-diagraming tool such as Microsoft Visio. Figure 2.2 shows you what a complex network diagram looks like. Don’t worry about all the pieces right now—this complex network resembles what you’ll build up to as you work through the book.
CHAPTER 2  Deploying the web server

Figure 2.1  Hand-drawn representation of a small network

Back office servers like Active Directory domain controllers and database servers

Figure 2.2  Documenting your complex network with a diagramming tool like Microsoft Visio

Internal clients can access websites.

Open ports 80 and 443.

Internet-based clients can access your websites through ports on the firewall.
Once you’ve completed your network diagram, you’ll be able to decide the best location for the web servers and make corrections to your network to best protect them. For the first part of this book, your lab environment will be fairly simple to diagram, but I want you to start thinking about your production environment. You can diagram your network later on; I remind you to do so in the “Ideas to try on your own” section. For now I give you a couple examples of the kinds of diagrams and decisions that you’ll need to consider, starting with a basic network environment with a single firewall.

2.1.2 Protecting the web server with a single firewall

Perhaps you want to install a web server with some websites at home or in a small office. Placing even a simple firewall between IIS and the internet will help reduce your attack surface. Starting with a simple network diagram like figure 2.3 will help you make decisions about the best location of the web server and security considerations.

In a network with one firewall, the web server is located behind the firewall for best protection. The decision becomes how to best configure the firewall based on client traffic to the website.

![Figure 2.3 Small network with a single firewall for basic protection](image)
CHAPTER 2  

Deploying the web server

- If the web applications you run on the web server are for internal clients only (common with a product such as SharePoint), then you can block almost all inbound traffic to the web server at the firewall.
- If you want folks from the outside to have access to your web applications, you’ll need to carefully open up holes in that firewall to permit access.

**NOTE** Each manufacturer of firewalls has its own method of configuring port access. You’ll need to review documentation for the one you choose for configuration specifics.

In most situations you want clients to be able to access your websites from the internet. The goal then is to open as few ports in the firewall as necessary to accomplish that goal. Common ports for websites are port 80 for HTTP traffic and port 443 for HTTPS. These may not be the only ports needed for your web applications, but I’ll cover that as we progress through the book.

Whether you’re building a new network environment as a test lab or for a small office, having the firewall protection in place before you deploy IIS is important. The risk of being attacked is too great.

---

**What about the firewall for your virtual lab environment?**

Keep in mind that in the virtual lab machine you set up for this book, Microsoft Windows Server has already enabled a firewall for protection by default. Don’t turn this off. In the next few chapters, you’ll work with this firewall to open up access to your websites when needed.

---

If a simple single firewall is good, then two must be great, right? Web servers often need access to back-end servers such as Active Directory domain controllers for authentication and database servers for storage. Many network teams add additional protection for these back-end servers, so let’s discuss that protection next.

### 2.1.3 Protecting the back-end with multiple firewalls: Working in the middle tier

Back-end servers such as domain controllers and database servers are precious and contain confidential information. Many network teams add an additional firewall between the web servers and those back-end servers. This creates a middle tier in which to place servers such as publicly accessible web servers. This middle tier enhances security for the back-end servers and is one of the most common infrastructure designs.

It’s always a physical firewall. Most firewalls today come with three network segments:

1. *Internal network*—For the prize back-end servers
17 Locating and protecting the physical web server

2 Dirty demilitarized zone (DMZ)—The dangerous unprotected segment that connects directly to the internet

3 Clean DMZ—A middle tier between the internal network and the dirty DMZ

These segments are protected by firewalls that block unwanted traffic. IIS is usually placed in the middle tier if outside access is desired. The firewall closest to the internet needs to have several ports open to accommodate public access to your various websites. The firewall closest to the back-end servers will be much more restrictive, preventing anyone from hacking past the web server into your internal network environment. Figure 2.4 displays a typical network design for this approach.

With this setup you have firewall configuration considerations in both directions. You may be wondering which ports need to be open to support outside access to your websites. (Hint: It could be more than ports 80 and 443.) Or which ports need to be open so IIS has access to the domain controller and the database? I answer these questions and more in the coming chapters. For now I want you to get an idea of the

![Figure 2.4 IIS protected in the middle tier by two firewalls](image-url)
current infrastructure you have in place. Remember, you don’t want to deploy IIS into an unsecured location. With your virtual lab machine, the Windows firewall is providing some protection for your deployment. In the future you’ll need to address the full infrastructure.

**Should I set up a firewall now?**

Keep in mind that the virtual machines you’re using for the labs in this book already have a Windows firewall protecting them, so you’re good to go for the labs. When you begin to deploy IIS in your own network environment, you should verify your firewall protection. If you’re the person that configures this, you probably already know the answer, but if you’re part of a network team that performs this operation, check with them.

Because your lab environment has protection enabled, and you’ve started to think about a secured production environment, let’s look at how you can get IIS deployed so you can start to see how all this works.

### 2.2 Installing IIS on Microsoft Windows Server

You can choose from several different methods of installing IIS. Why? Over the years, Microsoft has tried to find more efficient ways to accomplish IT tasks. A quick search on the internet is bound to turn up command-line tools such as Pkgmgr.exe and ServerManagerCMD.exe. Each one of these has its own set of parameters and syntax. Instead of reliving old commands such as these, we’re going to install IIS using the two best methods: the graphical Server Manager and Microsoft’s standardized management tool, PowerShell.

You also have a choice of Windows Server operating systems on which to install IIS. Most administrators start by installing IIS on a typical Windows Server 2008 R2 or Windows Server 2012. The other option is to install IIS on Windows Server Core. My favorite, and the one I think is the most powerful way, is Core, but I’m saving that for last because you may not have started using Core in your company yet. In this section, I walk you through two installations:

- Installing IIS using the graphical Server Manager, which is great for single-server installations and configurations
- Installing IIS using PowerShell for rapid deployment (my favorite way)

We’ll start with the single-server installation method.
2.2.1 Installing IIS with Server Manager

When you first install a Microsoft server, you get a simple server that can perform basic file sharing. All the other software features must be added, including IIS. This is a good practice Microsoft uses, and it’s called secured by default. It’s a practice you’ll apply to IIS as well. You don’t want to install software you don’t need: less to update, less to secure.

**IMPORTANT** Although I’m walking you through the how-to-install steps here, don’t try the installation until you reach the lab. As we move through the chapter, I walk you through two installation methods that you can use for Windows Server 2008 R2 or Windows Server 2012. When you get to the lab at the end of the chapter, you’ll be able to choose which method works best for you.

Microsoft loosely categorizes additional software into roles and features. Roles are services that affect the entire network, such as Active Directory. Features only impact the server, such as clustering. IIS has network impact, so it’s considered a role.

**Locating the IIS role by name on Server 2008 R2**

IIS is a role that can be added using Server Manager. In figure 2.5 I’ve opened Server Manager and selected the Add Roles link in preparation for installing the IIS role.

If you’ve run Server Manager before, and I assume you have, you know that from here you only need to select the role you want to install, click the Next button, and install the software. Installing IIS is no different, except for one challenge: what’s the role called?

---

**How big does your server need to be?**

I teach IIS to both budding administrators and highly experienced enterprise administrators. It’s interesting that at the point of our first deployment, without fail, the first questions I get concern server hardware. For example, “How big does my server need to be?” Translation: how much RAM, processing, and disk space will I need to support my web applications?

At this stage, early in learning about IIS, this is like asking, “How many MPGs does this vehicle get?” without knowing whether the vehicle in question is a car, truck, motorcycle, or bicycle. You can’t determine this until you understand more about what the vehicle is, what you need it to do in your environment, and what’s under the hood.

Hold on to those questions about capacity planning. I promise I’ll get to them as you progress through the book. For now, let’s get IIS installed and start figuring out what’s under the hood.
If you installed IIS in Windows NT, 2000, or 2003, the name of the software went through a series of changes. In Microsoft Server 2000 it was named Internet Information Server (IIS), and in Microsoft Server 2003 it was called Application Server. Take a look at figure 2.6 and notice both—the dilemma and the solution.

Bingo. We have both Application Server and Web Server (IIS). So which one is it? You probably guessed correctly that the role you need to install is Web Server (IIS).

**INSTALLING IIS ON MICROSOFT WINDOWS SERVER 2012**

Fortunately most of the administrative tasks between IIS on Windows Server 2008 R2 and Windows Server 2012 are similar if not identical. The installation is no exception. You’ll notice some visual screen differences between the two versions of Server Manager, but adding the IIS role is virtually the same. In figure 2.7 notice that the Add Roles and Features selection is located in a slightly different place. After selecting Add Roles you’ll be able to select IIS and its additional services as we did earlier.
After selecting the role and clicking next, you’re confronted with a long list of additional software components that you can add to IIS. Those selected by default are the minimum components you need to run a static web page (figure 2.8).

The default-selected components are the minimum components needed to get a website that can run a static web page. A static web page is one with no code other than old-fashioned HTML.
This is where the phrase “Don’t install it if you don’t need it” comes in handy. You only want to install the additional services you need for your web applications. By not adding components you don’t need, you reduce the attack surface for hackers. Think of a firewall—you don’t open ports that you don’t need because it’s risky.

How do you know what components you’ll need for your environment? You should base most of your additions on the needs of your web applications. An example would be that you’re using a web application that uses ASP.NET, in which case you’ll need to install ASP.NET. In most cases the application vendor’s documentation describes the necessary services you need to add. You’ll be installing (and removing) various components throughout your month-of-lunches journey.

**Removing the IIS role**

You can remove roles as easily as you add them. You may need to remove a web server for a variety of reasons, such as moving a web application to another server or if a failed install occurs. To remove a role, open Server Manager and select Remove Roles.

Later in the book we’ll be working with the file system and the storage of your websites. Note that when you remove the IIS role, the file system stays intact. This means when you remove the IIS role, it won’t delete any of your files and web pages.

The graphical installation of IIS is fairly quick and straightforward. Many administrators in smaller environments use it exclusively. I want to show you a faster and, as you’ll see later, better method that will work on a single server or hundreds. Time for a rapid IIS installation with PowerShell.

### 2.2.2 Performing a rapid installation using PowerShell

I hope you spent some time getting familiar with PowerShell as I suggested in the first chapter. Now you can put what you learned to work. The techniques you learn in this chapter will pay off as you move forward through the book. We’ll start with PowerShell v2 that ships with Windows Server 2008 R2 and then we’ll look at Windows Server 2012 and PowerShell v3.

**NOTE** PowerShell v3 is now available as an update to Windows Server 2008 and Windows Server 2008 R2. Although you should update to the new version, I’m including the v2 instructions for those who haven’t performed the update.

First I’d like to show you some slight differences before you do the installation in the lab.

**USING PowerShell v2 on Windows Server 2008 R2**

The capabilities of Server Manager, specifically the part that installs and removes roles and features, are also available in PowerShell. Microsoft added three cmdlets:
Installing IIS on Microsoft Windows Server

- Get-WindowsFeature—Displays the roles and features available to install or remove and displays the name of the software for the next two cmdlets
- Add-WindowsFeature—Installs one or more roles and features; you must use the name of the software as it’s displayed by Get-WindowsFeature
- Remove-WindowsFeature—Removes one or more roles or features

These three cmdlets aren’t part of the core PowerShell commands that are loaded when you open the shell. Instead they’re part of a separate module. Modules hold a collection of cmdlets that can be loaded and unloaded as needed. The module you want, called the ServerManager module, is shown in Figure 2.9. Use the following command to see a list of available modules on your system:

PS> Get-Module -ListAvailable

**A reminder about PowerShell modules**

Often PowerShell needs additional cmdlets to manage products and perform tasks. These cmdlets are stored in modules that can be imported when needed. To install additional Windows and IIS components, you need the ServerManager module.

To get access to the cmdlets in a module, you must first import the module with the Import-Module cmdlet. You can get a list of cmdlets that are in a loaded module using the Get-Command cmdlet (see figure 2.10):

PS> Import-Module ServerManager
PS> Get-Command -Module ServerManager

At this point, you can also use Get-Help to learn about the cmdlet parameters and see examples of how to use them. But because that’s what I’m here for, I’ll show you how to add IIS with the default installation using the Add-WindowsFeature cmdlet (see figure 2.11):

PS> Add-WindowsFeature Web-Server
It’s truly that simple and fast. Later in the book you’ll see examples of how to install to multiple servers at once using this same cmdlet. If you haven’t started working with PowerShell, this is a good time to start.

### Above and beyond

In PowerShell v2 the cmdlet that adds roles and features to a server is named Add-WindowsFeature. In PowerShell v3 this has been changed to Install-WindowsFeature. If you’re using PowerShell v3, you can still use the Add-WindowsFeature name because it’s now an alias to the Install-WindowsFeature cmdlet. I wanted to give you examples that worked for both versions of PowerShell, so I’m using the Add-WindowsFeature cmdlet instead of the v3 only cmdlet Install-WindowsFeature.

### Three tips and a “gotcha”

**Tip** You can add additional services, roles, and features by separating the names with a comma:

```
PS> Add-WindowsFeature Web-Server, Web-ASP
```
Installing IIS on Microsoft Windows Server

**Tip** To use Add-WindowsFeature you need to look up the name of the component you want to install. The names of all roles and features are listed in the Name column when you run Get-WindowsFeature.

**Tip** In PowerShell v2 you need to launch modules every time you launch a new shell. A feature in PowerShell, called a Profile, can automatically load the modules for you. But because you won’t need the ServerManager module that much, it’s best not to put it in a Profile and waste environment memory.

**Gotcha** In Windows Server 2012 the GUI management console isn’t installed by default when you use PowerShell. Add the management console component using the Install-WindowsFeature Web-Mgmt-Console.

**Using Windows Server 2012 and PowerShell v3**

One of the coolest and most convenient features of PowerShell v3 is the ability to dynamically load modules when they’re needed. In short, you don’t have to import the module to use its cmdlets.

This is a feature of PowerShell v3, so if you’ve installed v3 on Windows Server 2008 R2, you can perform an IIS install even faster. In figure 2.12 I’m using Windows Server 2012 and PowerShell v3. Notice that you don’t need to import the ServerManager module:

```
PS> Add-WindowsFeature Web-Server
```

The preceding alias cmdlet works, or you can use the name of the new cmdlet, as follows:

```
PS> Install-WindowsFeature Web-Server
```

Huzzah! It takes the graphical Server Manager utility longer to initially load than it takes to fire off that PowerShell command.

**WARNING** If you search Help for the Install-WindowsFeature cmdlet, you’ll notice a switch parameter called -IncludeAllSubFeature. You may even see someone on the internet using it. To use it yourself, do the following:

```
PS> Add-WindowsFeature Web-Server -IncludeAllSubFeature
```

Don’t use the -IncludeAllSubFeature when you’re working with IIS. Doing so will install all of IIS, including every available service. Remember what I said earlier: you only want to install what you need, for security reasons.

![Figure 2.12 Installing a web server role on Windows Server 2012](image)
So far I’ve walked you through an IIS installation using the GUI-based Server Manager and the ServerManager PowerShell module. I’ll bet you’re ready to get to the lab and get IIS installed, but I want to show you one more operating system option—one that you’ll need to consider in the near future if you want the best performance from IIS.

### 2.3 Installing IIS on Server Core

Microsoft Server Core is my favorite installation option for Windows Server when working with IIS. Its performance, ease of updates and management, and performance are amazing. Yes, I said performance twice. You may not have had a chance to experience Server Core, but you should if you’re the IIS administrator.

**NOTE** I recommend that you use only Windows Server 2008 R2 Server Core or Server Core on Windows Server 2012. The prior version, Windows Server 2008, didn’t support ASP.NET, and it was hard for most administrators to configure. Starting with 2008 R2, adoption of PowerShell has made Server Core much easier to use.

Server Core has been performance-optimized by removing the entire graphical desktop. As you can see in figure 2.13, it has no start bar, no icons, and no graphical components. The desktop is a blue background with a command prompt. You perform the initial configuration of the Server Core at the command prompt using PowerShell or a utility named SConfig.exe. Once it’s installed, you can manage IIS remotely using the graphical tools or PowerShell.

In almost every case placing IIS on this lightweight version of Windows Server is the right choice. The stumbling block for most administrators is the command line. Without the GUI, they feel lost. If you’ve been paying attention to the PowerShell cmdlets I’ve been demonstrating, you’ll excel at using Server Core.

In this section you’ll prepare a Server Core for an IIS installation and install IIS using PowerShell. Server Core requires a little more initial configuration than the graphical server before installing IIS. The configuration utility SConfig.exe will help, so we’ll start there.
### Why no desktop on Server Core?

Having a graphical desktop slows down the server, preventing it from accomplishing the task of “serving” efficiently. The removal of the graphical desktop also means fewer components to update and service pack.

Don’t let the lack of a graphical desktop keep you from using Server Core. You can perform management remotely using graphical tools installed on your local computer or through PowerShell.

In chapter 11 I show you how to configure IIS on Server Core for remote management with both the graphical tools and PowerShell. You’ll find no differences in the management tasks you need to perform.

### 2.3.1 Preparing Server Core for IIS

Windows Server 2008 R2 made configuring Server Core much simpler with the introduction of the SConfig.cmd (SConfig) utility. This utility lets you change the computer name, set an IP address, and join a domain. It also permits you to enable PowerShell and remote management.

Before you can install IIS on Server Core, you need to first enable PowerShell. As figure 2.14 illustrates, you do that by selecting option 4 and then enabling options 1, 2, and 3. This turns on PowerShell and gets you the ServerManager module you’ll need for the install.

![Figure 2.14 Enabling PowerShell on Server Core](image-url)
CHAPTER 2 Deploying the web server

Once you’ve enabled PowerShell you’ll be able to install IIS easily. How are you going to perform the install without a graphical tool? I’ll bet you already know the answer: PowerShell. Let’s dive in.

2.3.2 Adding IIS to Server Core

This is one of those times when you might search the internet on how to install IIS on Server Core and see a lot of old and bad stuff. Commands such as ocsetup.exe and oclist.exe are the wrong direction—they’re unnecessarily complex and have been replaced by a better tool: PowerShell. The only stumbling block is that when Server Core boots initially, it runs the old CMD.exe.

The fix is simple. Type `PowerShell` at the command prompt to load PowerShell. After it loads, the process to install IIS is the same as I described in the preceding section: import the ServerManager module and add the web server (see figure 2.15).

That was easy. In fact, with PowerShell managing, Server Core is faster and more efficient than the GUI-based installation. As you move through the book, all the PowerShell commands will work on Server Core.

TIP If for some reason you can’t find the ServerManager module on Server Core, open SConfig, select option 4, and then select option 3 (Allow Server Manager Remote Management) again. This will get the module installed.

As we move along in the book I talk more about Server Core and why it might be the best installation option for your server operating system and IIS. After completing any IIS installation, it’s best practice to test the web server to make sure it’s operating normally.

Our last stop before running the lab will show you a quick method of testing the web server.

2.4 Verifying a successful installation

After every installation of IIS, before I start configuring the containers that will hold websites and web applications, I check to make sure IIS is installed properly and working normally. It’s rare for IIS to have problems during installation, but it’s good practice to always check. In this section you’ll learn how to test the web server using the
default website and check for error events in the logs. To assist in the testing process, Microsoft created a default website container that contains a simple HTML web page.

2.4.1 Testing the default website

During installation of IIS, a default website container and web page is created. In your next lunch you'll dive into the details of this default website, but this is a perfect chance to use this website to make sure the web server is functioning as it should.

After you install IIS, open your favorite web browser and type the URL of the web server. If you’re physically sitting at the new web server, you can use any of the following URLs:

- http://localhost
- http://<ServerName>
- http://<IP Address>

If your test is successful, in your browser you’ll see one of the welcome pages shown in figure 2.16.

I prefer testing from another computer because it tells me that the web server and all its networking components are working. I also recommend that you test in this way when you do the lab for this chapter.

Speaking of the lab, I want to cover one last thing before I turn you loose on it. It’s always best practice to check for errors in the logs after an installation, so let’s examine that. Then you can dig in to the lab and try it yourself.

2.4.2 Checking for problems

Most network administrators check the Event Viewer for errors after an installation. I don’t like the Event Viewer. It takes too long to load, and I have to surf through hundreds of entries to locate a possible error message (see figure 2.17). You can filter and sort the views in Event Viewer in other ways, but I have a still better way. Again, it’s that “inadvertent administrator” thing, and I don’t want to waste time.
A more useful way to quickly check for events regarding IIS is to use the PowerShell cmdlet `Get-EventLog` (figure 2.18). This cmdlet searches any of the logs you have. Let me show you how to focus it on IIS and errors in the system log:

```
PS> Get-EventLog –LogName System –Source IIS* -EntryType Error
```

I added the `-EntryType` parameter so that the cmdlet would only search for error messages. This is much more efficient than dealing with Event Viewer, and if you know a little about PowerShell, you can now export this information to a CSV, HTML report, or text file for future reference. In the following example I export my results to a CSV file:

```
PS> Get-EventLog -LogName System -Source IIS* -EntryType Error |
 ➤ Export-Csv c:\IISErrors.csv
```

You now have all the information you need to try out this lab. Are you excited? Go wash your hands after your delicious lunch and get started.
2.5 **Lab**

Now it's your turn. I want you to install IIS to support the future WebBikez website on your lab computer. The bicycle shop has a network environment similar to your virtual computer: a web server using Windows Firewall plugged into the internet. This lab will only take a few minutes but will help ensure that you understand the concepts discussed in this chapter.

If you get stuck on a task and need help, go to MoreLunches.com for a complete walk-through of the lab. Open up your virtual environment and let's get started.

**Task 1**

For your first task go into Control Panel and verify that your firewall is turned on. Remember, you don't want to deploy IIS to an unprotected environment.

**Task 2**

Install IIS with only the default components and services. For this installation you get to choose the installation method—graphical or PowerShell. When the installation is complete, move on to task 3 to test the web server.

**Task 3**

After IIS is installed, test IIS to see if it's working. On the web server open a browser and type http://localhost in the address bar. You should see the default test page.

I prefer to test a new installation on a different computer. If you created the multi-VM environment described in chapter 1, open your domain controller and attempt to access the new web server from a browser. The URL will be the IP address or hostname of the new web server.

**Task 4**

To make sure you had no errors during the installation, check the Event Logs for IIS errors. Do this on the new web server using PowerShell and the `Get-Eventlog` cmdlet.

**Task 5**

Here's a challenge before you put down the book and go back to work. If you installed IIS using the graphical Server Manager, consider trying the installation again using PowerShell. If you have the extended environment, you can install IIS on the remaining virtual machine. If you have only the one virtual machine, I want you to remove IIS and reinstall it using PowerShell. Yup, do the install again.

After you've successfully installed IIS, to double-check it test it again by opening a browser and typing http://localhost. You should also use PowerShell to check for any errors in the system log.

2.6 **Ideas to try on your own**

When you get a chance in your busy schedule, I have two ideas for you to try out. This won't be easy, but it's something you should consider trying before you finish the book.
First start diagramming your own network environment. A hand-drawn picture is fine, but if you already have a network schematic in something like Visio, that’s even better. Diagramming will help you to decide how you may want to implement IIS in your own environment.

Try to install IIS on Server Core. If you haven’t worked with Server Core, this may seem like a daunting task, but I’ll help you. I’ve posted a full installation on More-Lunches.com, but let’s look at the highlights to get you started:

- Install Windows Server 2008 R2 Server Core or Windows Server 2012 Server Core into a virtual environment with your existing lab computer. We’ll use your existing domain controller.
- Using SConfig change the computer name to something like Web1.
- Using SConfig assign a static IP address.
- Using SConfig assign an IP address to your DNS server. This should be the IP address of your virtual domain controller.
- Using SConfig join your existing domain.
- Using SConfig enable PowerShell.
- Using PowerShell install IIS with its default components.
- Test your installation and check for errors.
When your website slows to a crawl or Exchange stops sending internet mail, you have to get things working again. Fast. IIS is the communication gateway for most Microsoft servers as well as the primary technology for publishing websites. Learning to manage it effectively means you can keep your systems running smoothly and go home on time.

*Learn Windows IIS in a Month of Lunches* is an innovative tutorial designed for busy administrators. Follow the crisp explanations, examples, and exercises in this concise, easy-to-read book, and you’ll be managing, securing, and automating IIS faster than you thought possible. You’ll start with IIS fundamentals, and then systematically explore web farm design, website management, and high availability, along with practices to keep your Exchange, SharePoint, System Center, and Lync servers running smoothly.

**What’s Inside**

- 24 bite-sized lessons with practice examples
- High-value troubleshooting techniques
- No previous IIS experience assumed

Written for anyone who needs to manage IIS, whether you’re an IT pro or a reluctant administrator.

Jason Helmick is a professional trainer and consultant specializing in Active Directory and IIS. He’s the founder of the Arizona PowerShell User Group.

“A fundamental reference for automating everyday tasks on IIS.”
—Victor Onate, Microsoft

“Comprehensive, in-depth, and up to date!”
—Jerry Warner
Southern Tier Athletics

“Push your IIS skills to the next level.”
—Brian T Young
American Express

“Excellent examples of management in both GUI and PowerShell.”
—Shane Beacom
Independent IT Consultant

“Absolutely essential for the multiple-hat systems administrator.”
—Francis Setash
U.S. Department of State