KIT304 Server Administration and Security Assurance

Tutorial 10

Goal

This tutorial is in many ways a Windows version of the previous tutorial. You will set up several websites on Windows Server 2016, making use of ASP (in place of PHP) and MS SQL (in place of MariaDB).

Introduction

In the previous tutorial you learned that Unix, and in particular Apache, is the dominant web server platform. However, Windows Server is still a significant part of the market, and Microsoft's cloud based Azure platform hosts a large number of sites. In addition to this, it is a popular server for internal networks, and as such many businesses internal websites are hosted on the platform.

Microsoft's web server platform is called *Internet Information Services*, frequently abbreviated to **IIS**. It was first released in 1995 and is now at version 10. Looking through the documentation on the TechNet web site, you will see that it has been fairly stable in terms of usage and instructions since IIS 7.

Aside: PHP vs. ASP.NET

As part of your degree, you should have taken a unit that covers dynamic web programming. The main language of interest would have been the server scripting language PHP. Server-side scripting languages enable web developers to make dynamic web pages that interact with databases for a rich user experience. Server-side web page scripts produce HTML (and CSS + JavaScript) that is sent to a client computer to be rendered in the browser. The content displayed is a product of the scripts written by the web site developer.

PHP was released in 1995, and Microsoft quickly followed in 1996 with the launch of IIS 2.0 and Active Server Pages (ASP). ASP evolved into ASP.NET in version 5.0 of IIS in Windows 2000. ASP is also a server-side scripting language which shares many similarities with PHP. Both are executed on the server when a page is requested, and both produce a web page to be sent to the client based on an initial request.

Both languages are quite popular, but perhaps unsurprisingly PHP is the more popular of the two as it is Unix native – the predominant platform for most web servers (PHP can also be installed on

Windows). By market share PHP has 83% of the market, with ASP.NET making up 13.5% of the market. However, if you look at the top 1,000 websites on the internet that narrows to 57% usage for PHP, and 24% for ASP.NET, with the remainder being a combination of Java, Cold Fusion, Ruby and static sites.

PHP is the language that we teach in our degree as a dynamic server-side scripting language; however, it is not really a clear-cut obvious choice for developers. A strong case can be made for both languages. PHP has the advantage of being free, and arguably is one of the best documented languages for any platform or context. ASP.NET has nice GUI interfaces, enables easy integration with other .NET based applications, and has greater capabilities in relation to threaded programming. Both have their own distinct programming styles and cultures, and those are potentially more of a factor in language choice than any particular feature of either language.

In this unit we won't be teaching you PHP or ASP.NET, however, you will see a small amount of ASP.NET in this tutorial.

Activity

1. Launch the VMs of both **Windows Server 2016** and **Windows 7** and give them both IP addresses within the network **`192.168.1.x`**.

2. IIS is already set up in its default mode on the Windows 2016 server. Open **Internet Explorer** and enter **`localhost`** into the address bar. Then on the **Windows 7** VM open a web browser and enter the IP address of the Windows server in the address bar. In both cases, you should see the default IIS web page screen, showing you that the web server is running.

3. Although IIS is set up, the ASP scripting language is not yet enabled on the server. To fix this, on Windows 2016 server, open the **Server Manager**, and select **Add Roles and Features** from the **Manage** button. In the Wizard that is displayed:

    - accept any default settings, and click the **Next** button until you get to the **Server Roles** page
    - expand the **Web Server (IIS)** branch, inside it, expand the **Web Server** branch, and inside that expand the **Application Development** branch
    - check the **ASP**, **.NET Extensibility 4.6** and **ASP.NET 4.6** items – if a popup window appears asking you to confirm settings, click the **Add Features** button.
    - click **Next**, then **Next**, and then **Install**
    - when the installation is finished, close the wizard window

4. In **File Explorer** open the **inetpub** folder at the root of the **`C:`** drive, and then open the **wwwroot** folder inside. This is the default location for web server content. Here you should see the file (**`iistart.htm`**) that generated the page shown in both web browsers.

You're going to create a new file in this directory. Open **NotePad** from the start menu, and type the following into it:

```
<%
    response.write "Hello World"
%>
```

Choose **File → Save**, and navigate to the **wwwroot** folder you visited earlier. Change the **Save as type** filter to **All Files**, and save the file as **`test.asp`**. Open the local web browser and visit the site **http://localhost/test.asp** - you should see the output of the script in the browser window.

5. Next, create another file (or edit your existing one) with the following code and test it on a web browser to see some more ASP in action:

```
<%
    response.write "Date: " & date() & "<br/>"
    response.write "Time: " & time() & "<br/>"
    response.write "Browser is: " & Request.ServerVariables("http_user_agent") & "<br/>"
    response.write "IP is: " & Request.ServerVariables("remote_addr") & "<br/>"
    randomize()
    r = rnd()
    if r < 0.5 then
        response.write "You're the Best"
    else
        response.write "You're the Worst"
    end if
%>
```

If you want to learn more about ASP, follow this link:

https://www.w3schools.com/asp/asp_introduction.asp

6. Next, you will set up several websites. First, you will create the files that comprise the website's pages. In **File Explorer** go to the **`C:\inetpub`** directory and create a subdirectory called **sites**. In this folder create two subdirectories for two different websites. Inside each of those subdirectories, create a file **`index.htm`** or **`default.asp`** that contains some text that will let you distinguish between the two web sites.

7. In the File Explorer, right-click on the **`sites`** subdirectory, and then go to **Properties**. In the **Security** tab select **Edit**. You need to add permissions to this directory that enables IIS to interact with the websites and their content. Select **Add** and then in the popup window type **IIS_IUSRS** in the box and press **Check Names**, then press **OK** on the various windows to apply the changes. This permission has been applied to all the subdirectories.

8. Next, open up the link to Microsoft's TechNet site to learn how to view which web sites IIS is currently configured to serve, and follow the steps there to create two new web sites. Create

them with actual domain names like you did on CentOS in the previous tutorial (for example, one of the two sites you created was **`www.networks.com`**):

https://technet.microsoft.com/en-us/library/cc771341(v=ws.10).aspx

Once you have added both websites to IIS, but before you can view them in a web browser, you need to make their domain names resolvable. While you do have a DNS server running on the Windows server, you won't be configuring it today as that will be the subject of a future tutorial.

So, again, you will use the technique you used in the last tutorial, of adding the domain names directly to the **`hosts`** file, which on Windows systems is located here:

```
C:\Windows\System32\drivers\etc\hosts
```

To edit this file:

- locate **NotePad** in the **Start** menu
- right click on it and select **More > Run as Administrator**
- in the new Notepad that opens, go to the **File** menu and use the path above to find and open the file – you will also need to change the file name filter from "Text Documents (*.txt)" to "All files"

Entries in this file are single lines with the IP address of the server, a space or tab, and then a list of host or domain names, for example:

```
192.168.1.2     test.com
```

9. Once you have added both websites verify that you can load them in the browser and easily jump between them, seeing that they have been mapped to the two domains and the different local directories.

10. As discussed in the previous tutorial, modern web servers usually generate sites dynamically, and pull content from a database. Typically, the database is often run on a separate computer, and indeed you could set up CentOS as a database server, running MariaDB, and providing content to web applications running on Windows Server 2016.

    Both PHP and ASP.NET are able to operate with MS SQL server and MySQL, and many other types of databases. Today you use MS SQL which has already been installed on the Windows 2016 Server.

    In the previous tutorial you installed WordPress, which is a PHP based content management system. Today you will install one of the most popular ASP.NET content management systems: **DotNetNuke**.

11. Create a directory in **`C:\inetpub`** to contain your DNN website. Extract into this folder the archive called **DNN_Platform_9.1.1_Install** which is located in the Administrator's **Downloads** folder.

12. In the **Internet Information Services (IIS) Manager** tool, right click on **Sites** and select **Add Website** and fill out the dialog. The website name is important, as you will need it again shortly.

13. Open **Microsoft SQL Server Management Studio** from the **Microsoft SQL Server 2016** folder in the Windows **Start** menu. Once it has loaded click **Connect** to log in using the default Windows Authentication.

14. In the **Object Explorer** window that opens, right-click on the **Databases** item and select **New Database**. When the dialog appears, enter a name for your DNN database and click **OK**.

15. Expand the **Security** branch in the **Object Explorer** window, right click on the **Logins** item and select **New Login**. The login name you need is going to be the application pool that you created (that is, the site name) in **IIS Manager** and is listed in the *Application Pools* section under the *Windows Server.* Type it in as the database login, formatted like this:

    ```
    IIS APPPOOL\siteName
    ```

    You won't be able to search to confirm it, you will need to type it correctly. Note that the path has three **P**s as it is **App Pool** merged together.

    Next select the **User Mapping** section from the list of the left-hand side of the **Login - New** window. Check the box for the database you created in the previous step, and then check **db_owner** in the lower box as well to specify that the login you are creating is the owner of the database you created. Finally, click **OK**.

16. Next, you need to modify the security settings of your new site folder to provide your web application with full access to it:

    - In **File Explorer**, locate your new site folder, right-click on it, and select **Properties**
    - Select the **Security** tab, and then click **Edit…** and then click **Add…**
    - In the window that appears, click the **Locations…** button, and click on the server name at the top of the tree (**CorpDC1**), then click **OK**
    - In the **Enter the object names to select** field, enter the full path and site name as you did in step 15 and then click **Check Names** – if you've done this correctly, you should see just the site name in the field, now underlined – then click **OK**
    - In the permissions area for your new user, give them **Modify** permissions on the folder (in addition to the other, default settings), then click **Apply**, and finally click **OK**.
    - Close the remaining dialogs

17. Add your domain to the **hosts** file on both **Windows 2016** and **Windows 7** (don't forget to run **Notepad** as Administrator).

18. Try to visit your new domain on both **Windows 2016**, and **Windows 7**. (The first time you visit the site will take a little longer while the DNN software initialises itself). By default, Windows 2016 server has a more locked down version of Internet Explorer than you may be used to on other Windows platforms. If you see a prompt telling you that **Content from the website listed below is being blocked…**, click the **Add…** button to add it to the trusted sites list, then click **Add** and then **Close** in the popup dialog.

19. Since DNN is running in setup mode, enter an admin username and password that you would like to use for the DNN CMS, and website name. Then under **Database Information** select **Custom** for the **Database Setup** and select **SQL Server/SQL Server Express Database.** Finally type in the name of the database that you created earlier.

20. Press **Continue** and DDN will then install. Once completed experiment with the platform, comparing it to your experience with WordPress in the previous tutorial.

Conclusion

In the last two tutorials, you have configured both CentOS and Windows Server 2016 to be web servers. It wasn't especially complicated – largely just a case of installing the application and turning it on. You have also setup WordPress and DotNetNuke as examples of typical web platforms that many servers run.

In the coming tutorials, you will focus a bit more on the use of databases, DNS and SSL (TLS) in conjunction with a web server that is running.

Skills to Master

To successfully complete this tutorial, you must have mastered the following skills:

- enabling the ASP scripting language on IIS
- creating simple ASP scripts to demonstrate the language is working
- creating and configuring named virtual web hosts on IIS

You could be required to demonstrate any of these skills in this module's practical exam.