UNIT-9

Hosting and Deploying ASP.NET Core Application

Once we successfully developed our web application, we may require to host the application to the server so that other people can access it. The process of deploying/installing an application into the server is called "Hosting".

Web Server: A web server is a process for hosting web applications, which responds to HTTP requests and delivers contents and services. A web server allows an application to process messages that arrive through specific TCP ports. Some of the web servers that we can use to host ASP.NET Core are:

- Microsoft IIS
- Apache
- NGINX

1) IIS web server: The IIS web server comes from the Microsoft stable and runs only on the Microsoft Windows operating system. It is actually not free, since it comes as a part of the Windows operating system. We might feel comfortable with IIS if we have already used the Windows OS ecosystem.

Advantages:

~~→As it is open source~~
→ Has the support of Microsoft.
→ We can have access to the .NET framework along with ASPX scripts.
→ Can be easily integrated with other Microsoft services like ASP, MS SQL etc.

2) Apache web server: Apache is an open source web server which was developed and maintained by Apache Software Foundation. It is a result of the collaborative efforts which was aimed at creating a robust and secure commercial grade web server which adhered to all HTTP standards. Apache is equally efficient on almost every operating system but finds can be found to be in maximum use when combined with Linux.

Advantages:
→ As it is open source, so there are no liscensing fees.
→ It is flexible.
→ Has a high level of security.
→ Strong user community to provide backend support.

3) NGIX: NGIX is a robust web server which was developed by Russian developer Igor Sysoev. It is a free open-source HTTP server which can be used as a mail proxy, reverse proxy server when required. NGIX has a lightweight architecture and is highly efficient. This is probably the only web server which can handle huge traffic with very limited hardware resources.

Advantages:
→ Open source.
→ A high speed web server which can be used as a reverse-proxy server.
→ Can be used better in a virtual private server environment.

Hosting Models in ASP.NET Core:

Q. How do you host and deploy the ASP.NET core application? [Imp]

There are two types of hosting models in ASP.NET Core:

1) Out-of-process Hosting Model: In this model we can either use Kestral server directly as a user request facing server or we can deploy the app into IIS which will act as a proxy server and sends requests to the internal Kestrel server. In this type of hosting model we have two options:

i) Using Kestrel: Kestrel is a cross-platform web server for ASP.NET Core. Kestrel is the web server that's included by default in ASP.NET Core project templates. Kestrel itself acts as an edge server which directly serve user requests. It means that we can only use the Kestrel server for our application.

ii) Using a Proxy Server: Due to limitations of the Kestrel server, we cannot use this in all the apps. In such cases, we have to use powerful servers like IIS, NGIX or Apache. So, in that case, this server acts as a reverse proxy server which redirects

every request to the internal Kestrel server where our app is running. Here two servers are running. One is IIS and another is Kestrel.

2) **In-process Hosting Model:** In this type, only one server is used for hosting like IIS, Nginx or Linux. It means that the App is directly hosted inside of IIS. No Kestrel server is being used. IIS HTTP Server (IISHttpServer) is used instead of the Kestrel server to host apps in IIS directly.

 # Steps to Deploy ASP.NET Core to IIS:

Step1: Publish to a File Folder. Publish to Folder With Visual Studio.
Step2: Copy Files to Preferred IIS Location.
Step3: Create application in IIS.
Step4: Load App!

⊛ ASP.NET CORE Module:

The ASP.NET Core Module is a native IIS module that plugs into the IIS pipeline to either:

→ Host an ASP.NET Core app inside of the IIS worker process (w3wp.exe), called the in-process hosting model.
→ Forward web requests to a backend ASP.NET Core app running the Kestrel server, called the out-of-process hosting model.

When hosting in-process, the module uses an in-process server implementation for IIS called IIS HTTP Server.
When hosting out-of-process, the module only works with Kestrel. The module doesn't function with HTTP.sys.

⊛ Docker and Containerization: [Imp],

What is docker?

⇒ Docker is the platform for deploying and building the applications which delivers the application in the packages over the operating system level virtualization. Docker requires Image and Container to run the application.

What is docker image?

⇒ Docker image is the set of configuration and instructions to create the docker container, an image is created during building the application based on steps defined in the docker file of our application. The docker image is the read-only file which cannot be modified once it's created, but we can delete the image from docker.

What is container?

⇒ The container is an independent isolated process of an operating system that has its own networking and file system to run the image or application. The container is created on the docker engine based on the image configuration.

Difference between Virtual Machine and Docker Container?

The virtual machine (Linux, ubuntu) is installed on some other operating system (windows) which shares the same lifecycle such as running and shutting at the same time. Virtual machine acts as a real PC on an actual operating system which has full features such as RAM, Hard disk, networking etc. In this case, often virtual machines are called guest PC and the actual PC which runs the VM is called host system machine.

The container is the minimal and smaller part of a virtual machine which does not use the entire operating system as a virtual machine. The container cuts the unnecessary components of the VM and creates the isolated virtualized environment called a container, which is faster than virtual machine. The container does not require the host system as a virtual machine instead, it works on the docker engine.

⊛Azure Cloud :

Q. How to publish asp.net core web app in azure cloud?

Solution:

Step1: Right click in project and select publish option.

Step2: Select Azure as target.

Step3: Click Next and Select specific target and we will find Azure App Services (Windows) in right pane.

Step4: After clicking Next we will find App Service in left pane and click (+) sign to create new App Service instances in right pane.

Step5: After clicking (+) sign we will see new popup window to add new Service instance which has following field like Name, Subscription, Hosting Plan. Now just click in new hosting plan from (+) sign button in hosting plan section. In Hosting Plan section, we will have option to select location and Size. Now we will select size option to Free and click Ok. And then click Create. Now app service instance is created in Azure.

Step6: Now we will see App Service Instance is created and select newly created instance and click Next.

Step7: We will redirect to API Management tab in right pane, this option for API management we can skip by selecting checkbox skip this step and click Finish.