**IIS History:**

Windows 2000/IIS 5 -> Pre Installed. Had security issues of getting hit by worm due to pre installation

Windows 2003/IIS 6 -> Not pre installed. Security issue resolved. Explicitly decide which pages to run like Active server pages, asp .net pages etc.

Windows 2008/IIS 7 & Windows 2008 Server R2/IIS 7.5 -> expanded iss by breaking out what component of IIS are installed. Now you can choose, windows authentication, basic authentication, http logging, types of application development installation etc. also additional extensions from Microsoft and third parties can also be installed.

**Installing IIS:**

Go to server manager on the server and windows features on the client machine to install IIS.

On server:

Go to server manager -> roles -> add roles -> choose web server (IIS) -> select specific components ->

Generally choose

1. Static content compression
2. Dynamic content compression
3. Management Service - > allows this server to be managed from a remote machine

Finally install it.

**Web platform installer:**

Download this tool from Microsoft which tells the latest components available for IIS. Very handy for installing latest extensions.

**Creating Websites:**

Out of the box we get Default Web Site Under Sites in IIS manager. We can delete this as there is nothing special about and setup our own.

Click Add Web Site and fill the necessary details

1. Site Name -> Site1.com -> this also creates an application pool with the same name automatically
2. Physical path -> C:\inetpub\site1.com (this is the name of the folder where your starting html file is located. It can be any path on the physical drive of the server)
3. Define the binding information -> what happens is that the request comes to IIS when the url is hit and then IIS will base off of this binding information and route the request to the appropriate site. IpAddess, Port and hostname. One of the three has to be unique between all the other sites so that IIS is able to determine which site the request should go to.

Always leave IPAdress to UnAssigned. This means that which ipaddress the request come into. Web servers often have multiple ipaddresses. This particular setting comes in play when we use **SSL certificate.**

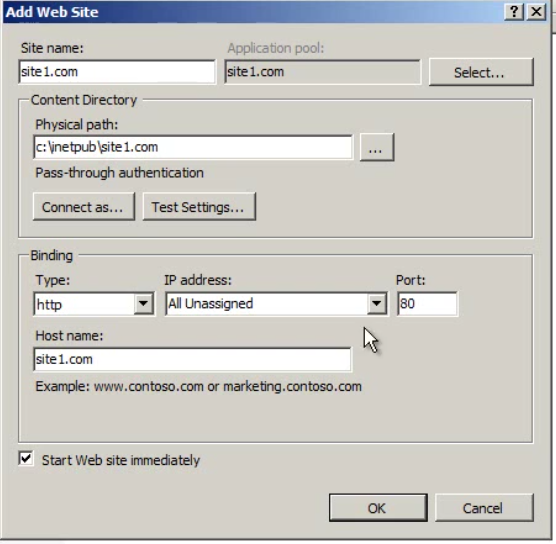
**Port 80** is the default for **http site.** When the user types in the url for the site, it is automatically resolves to port 80. We can change this to any arbitrary number let’s say 8080 and then the user have to type site1.com:8080 to hit the site. This is something not to be done so consider port number to be a not changeable variable.

**Hostname ->** what happens with the web browser is when it connects to the web server, one of the pieces of information it passes along is the host header, the hostname. When you type google.com, it passes google.com along with it in the host header. We will also keep it same.

**Set hostname to site1.com**

1. No we go to browser and type <http://site1.com>, we see it will work.
2. Most of the times user types in <http://www.site1.com> and volia we get a 404 error. Why is that? This is because in the bindings we have our hostname as site1.com whereas the web browser request in [www.site1.com](http://www.site1.com). A mismatch. We can resolve this by adding additional bindings information.

Go to edit site -> bindings -> add and then provide all the information similar as before except for hostname which will be [www.site1.com](http://www.site1.com).



**Adding SSL Certificates:**

For this we are going to add an additional binding with https. But to add an https binding we need to provide the SSL certificate for the site.

To add the ssl certificate, go to the root of the iis server (below the start page) and click server certificates. this is where we can manage the server certificates. You might see a certificate already there starting with the name like WMSvc -\*. WMSvc stands for windows management service. This is service tat allows us to manage iis remotely. No need to do anything to this certificate.

For your site, click on Create Certificate request and fill in the required details. It will create a text file called a certificate request and then we would hand that off to a third party such as Thawte or VeriSign or GoDaddy, and then we would purchase a certificate from them. For you r personal use, you can create a self-signed certificate.

Click on self-signed certificate and give it a name as it is for internal use only. Let’s name it site1.com. now go to site1.com edit the bindings, add additional binding with https, all Unassigned, 443, and select the site1.com ssl certificate. You will notice that the hostname value is disabled now. The reason is that as the request comes in, the hostname is still encrypted so IIS can’t determine what the hostname is and so it can’t use it to route request to the appropriate site*. This is why whenever we have a hosted web site in a shared environment, we always have to purchase an SSL certificate along with the IP Address because that’s how we distinguish between the different SSL sites.* We specify which IP Address that request is going to come into, because we want to leave the port alone to 443. If this is only SSL site then we can leave it to All Unassigned otherwise we need to select the IPAddress that we are going to bind this specific site to.

Now we can run our website with <https://www.site1.com>

Normally browser checks for 3 things with the certificate.

1. The expiry date is in the current date range. Normally we buy certificates for 1 or 2 years. The time on the computer must fall with in the range of the certificate is valid for.
2. Second things it checks for that was the certificate created by an authority it trusts. When windows ships, it comes with a set of certificates from trusted certificate authorities. Because this is how PKI, public key infrastructure works. We always know who created which certificate. The browser checks if the certificate is trusted by a higher lever authority and in our case it’s not. This is because we have created the certificate ourselves.
3. Third thing it checks for the hostname. Does the hostname match with what is in the certificate.

**You can still access the site and the https encryption will occur. But the browser will display a certificate error.**

**SSL Wildcard Certificate:**

What a wildcard certificate is, instead of certificate being issued to site1.com, it can be issued for ildcard.site1.com. there is no way to create a wildcard certificate through IIS. We can create it though using the **MakeCert executable** which is a part of windows sdk.

Remove the SSL binding for site1.com and start over with site2.com

Firstly, import the certificate. Go to the root of the IIS under the start page and click import. Give it the password that you used when you exported it. And now you got a certificate for \*.site2.com. now you add a SSL binding to the site2.com. now as soon as you select the wildcard certificate in the SSL certificate, hostname unlike previous will become enabled. Now we can use this wildcard certificate for as many site as possible as long as it is something.site2.com. type in the hostname as [www.site2.com](http://www.site2.com) and then navigate to <https://www.site2.com> in the browser. And this time it will work just fine and we won’t get any certificate error if you have trusted the authority that created the certificate.

Now if we create another site with the wildcared certificate let’s say beta.sie2.com and navigate to it, it will also work just fine.

You can only have one wildcard certificate for one ip address. If you want to turn the address bar green in case of https, you need to have an extended validation certificate. This is not possible with the wildcard certificates.

**Configure IIS:**

We use web.config to configure settings for our sites. When you go to the url of a site, the page it displays is being told to it by default document settings. You can click on the site in the iis and then select the default document icon to see the default document list. This is how the website when you don’t specify a specific page routes to the default page. Let’s say our site has 2 pages

Default.html and home.html.

The way that the site knew to pull up the default.html page is because this page is at the top of the default document list in the iis. If it does not find the page, it will break and show us a 403 error. Let’s say we renamed our default.html file to xyz.html and in the document default list there is no page listed with name xyz.html then, web server will not be able to find the listed html pages in the root directory of the server and will break.

*By default iis forbid to list the contents of a directory.*

Now suppose you want to have xyz.html as you default page for the site, then you need to add this page to the document default list. *Now as soon as you add the xyz.html to the list, iis will create a web.config in the root folder of the site. Also noticeable here is that, the added page will have an entry type of local type instead of inheritance.*

Since IIS 7, certain settings are stored in the web.config xml file under the system.webserver namespace.

The generated web.config will look something like this.

<?xml version=”1.0” encoding=”UTF-8”?>

<configuration>

<system.webserver>

<defaultDocument>

<files>

<add value=”xyz.html” />

<add value=”xyz2.html” />

</files>

</ defaultDocument>

</ system.webserver>

</configuration>

----------------------

Let’s understand what is inheritance entry type.

Let’s create a folder say Contact under the root folder of the site with a Contact.html page inside it. Now this page will not be listed in the default document list and we also don’t want it to be listed.

Now in the iis under the site2.com web site, you will be able to see a folder named Contact. And as soon as you will click on that folder and go to the default document, you will see that the xyz.html is listed as entry type inherited because xyz.html is the member of the parent folder. Similarly if we add the contact.html to the default document of the folder, we can see a web.config file similar as before is created inside the contact folder.

Now suppose you want to have your contact.html below the interited document home.html. Once you do that, the web.config will change something like this

<?xml version=”1.0” encoding=”UTF-8”?>

<configuration>

<system.webserver>

<defaultDocument>

<files>

<clear />

<add value=”Home.html” />

<add value=”Contact.html” />

</files>

</ defaultDocument>

</ system.webserver>

</configuration>

**NOTE: any page removed from the default document list of the root of the iis server, it will be removed from all the default document of all the sites.**

Having these settings in the web.config file is great. It gives better visibility to the developer. Gives better control if you are in any type of shared hosting environment.

In a real scenario where iis settings are managed by some other team and webconfig by the development team, we do not want webconfig to get pushed into the production environment and overwrite the iis settings.

For that purpose, we nned to move some settings from the web.config file to the iisconfig. *For this purpose, go to the root of the iis server and select Feature delegation. What this does is defines which settings are going to be stored in the web.config fileand which are in the iis config. All the settings under feature delegation with the delegation set to read only will not be overwritten by the web.config as web.config can read the setting but cannot write to it. Not delegated means that now the web.config will not be able to read the setting as well.*

**Important to note here is that , the settings that are marked read only in the feature delegation of the root of the iis, if included in the web.config then it will throw a 500 errror.** This is because the configuration is set to read only and web.config will try to overwrite the setting which is not allowed giving the error. **So all these settings need to be removed from the web.config.**

**Application Pool Basic:**

It is also referred to as worker processes. This is because this is where the work actually happens in IIS. This is where you code is run. Starting in IIS 7, if you create a new site, it creates a new pool automatically.

By default, you will get one site per application pool configuration, which is generally what we want. Double clicking on the application pool shows the .net version used and the managed pipeline mode. Pipeline mode can be integrated and classic. In general go for the integrated mode and if the site fails to run under this mode then choose the classic mode.

In integrated mode, the asp .net runtime and other stuffs are part of the pipeline. It is all a continuous process. In classic mode it is not the part of the pipeline so when the request comes in, it is handed to the asp .net runtime and then gets handed back to the iis. Integrated gives better performance and better reliability.

**Application Pools Advance:**

There are a ton of settings. Let’s look at some of these.

* Enable 32-Bit Mode -> true/false: self-explanatory. if your website as any third-party component that specifically work with a 32 bit or 64-bit then change the setting accordingly, else it should not be an issue at all.
* Identity -> this defines what security context application pool is going to run in. by default, it is set to ApplicationPoolIdentity which is a special built-in account for the local computer. The other options are LocalService, LocalSystem,NetworkService. Also we can specify a custom account.
* Idle Timeout (mins): this tells that if nothing happens in this application pool for the specified minutes, it will shut down the application pool until the next request comes in. this will have save resources. Unfortunately, what tis means is that the first person who hits the site after idle timeout passes by, the user will face the first hit penalty. User has to wait for the application pool to spin up, wait for the first time compile, then wait for the any type of database connection. So the first user will have a pretty poor experience.

In the production environment, we usually set this to 0.

* Maximum worker processes: by default this is set to 1. This is where we can enable what is called web gardens meaing when application pool has multiple processes that it can divvy request out to.
* Ping Enabled: IIS periodically checks to see if your application pool is healthy. If the pool is not healthy then iis will recycle the application.
* The last section in the advance setting is Recycling. What this does is if the application pool is unhealthy, it will shut down the worker processes and starts a new one. By default, this happens every 29 hours. The only big thing to watch out is that is the session state is stored in the process, that part of the application pool, so if you are storing in session state, in process, and the app pool recycles, it is potentially gonna cause a ooor experience for the end-user.

**Application pools strategy:**

It will become cumbersome to remember the settings done for each app pool. So in order to avoid that, we can set *application pool defaults settings* which gets apply to all the new application pool that gets created.

We can also have multiple site set in the same application pool. By doing this we can save a nominal amount of resources. There is some overhead with each application pool and the less the pool the more memory we will have in hand. Nominal means that we would only save few MB of memory.

The disadvantage of combining sites in the same app pool is that if anything happens to site2.com it is going to effect site1.com running in the same app pool. Also if something causes the app pool to recycle it is going to affect both the sites. *Make it a general practice to have one site per app pool.*

Sometimes we want to have multiple app pool per site. Let’s say that we have our site2.com working in majority of cases. But there is some specific part of the site that has a lot of bugs in it and it is crashing a lot. So this specific portion of the site will affect the rest of the site. What we can do at this point is to break this specific portion into a different application pool.

Suppose you have a contact folder in your site, which has a lot of bugs. Create an application pool say named site2.com-Contact

Then go to the site folder and right click and select convert to application. When we do this, it will ask for the application pool to use for it. Select the newly created application pool and click ok.

On doing that, you will see that the folder sign across the Contact folder will change into an application pool icon indication that this folder is now running as an application in its own pool. So if anything goes worng in contact, it won’t affect the rest of the site.

**Manage IIS:**

Go to the root of the IIS and select worker processes. This will show all the running processes. If you click any of the worker process, it will display any request running from more than 0 seconds. This is really handy to see in real time which page is running longer then you would expect them to.

We can see the page , http verb, client IP address and the state of the request along with the time elapsed.

Now let’s learn how to capture requests based on a certain criteria. We are going to do that with failed request tracing. *Install the Tracing Module.*

Open Server Manager. Click on add roles and features. Then under the iis -> Health and Disgonostics -> select Tracing and install it. This will allow us to do the failed request tracing.

Once it is installed then go to the site in the IIS manager and you will see, Failed Request Tracing Rules icon. Click the icon to open it and the from the right panel, select Add to create a rule. Create the rule by selecting the proper options and the click ok.

Now for this rule to work, we need to enable the failed request tracing for the site. From the right panel select Edit Site Tracing and click Enable and save it.

Now let’s say we want to find requests that are returning 404 at the highest frequency. *For this, we can use log parser, a free tool from Microsoft to parse our IIS logs and determing which page is returning the most 404’s.*

Log Parser Lizard is another such tool. Download and open the tool. You can use sql like queries to read the logs

Select cs-uri-stem, count(\*) from C:\inetpub\log\logfiles\W3SVC1\\*.log

Where sc-status = 404

Group BY cs-uri-stem

**Managing IIS through PowerShell:**

Powershell treats iis like a drive letter. CD IIS;\\

The IIS drive is provided by the WebAdministration module, so you need to install/import that module first.

How you install the module depends on your actual system and whether you use GUI or PowerShell. On a Windows Server 2008 R2 for instance you'd install the module with the following PowerShell commands:

Import-Module ServerManager

Add-WindowsFeature Web-Scripting-Tools

After the module is installed you can load it in your script like this:

Import-Module WebAdministration

Creating a new site in IIS

Cd IIS:\\

New-Item IIS:\Sites\site3.com -bindings @{ protocol=”http”; bindingInformation=”\*:80:site3.com”} -PhysicalPath C:\Inetpub\site3.com

New-ItemProperty IIS:\Sites\Site3.com -name bindings -value @{ protocol=”http”; bindingInformation=”\*:80:www.site3.com”}

**Deploying a Website in IIS:**

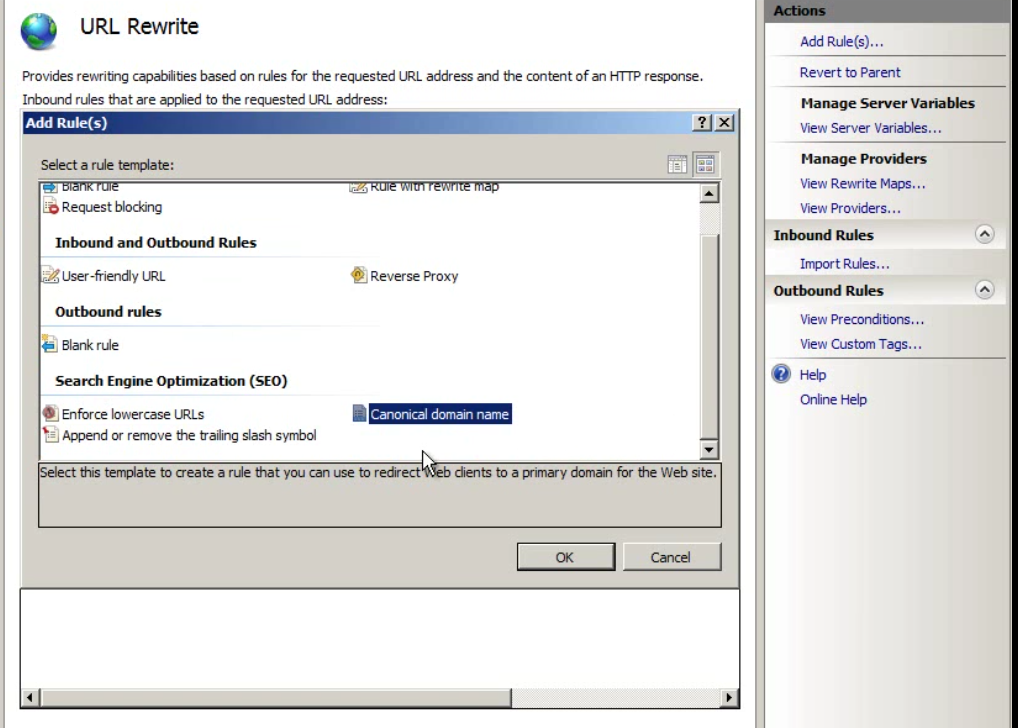
We need a tool called Microsoft Web Deploy or MS Deploy. This tool is also referred as Web Deploy. This tool allows you to pull or push different websites between different servers. So you can take the content of the web server you are on and push it to another server.

FTP can also be used to deploy websites but it is largely a manual process. With ms deploy you just have to provide the source and the destination and rest it will figure out. Download and install it to use its powershell commands to deploy web sites.

**Url Rewrite:**

It allows you to modify the behaviour as the request comes in and out. Just go to the site in IIS and click URL Rewrite icon to open it. Click Add rule to add the rule.

As soon as you click add rule, it will ask you to select the rule template.



Lets create a canonical domain name. what this rule will do is that it will either add or drop www from the incoming request. The primary reason you would do this is the SCL.

Select the primary host name [www.site2.com](http://www.site2.com). We are creating a rule which will add www if it doesn’t exists.

Now if we type site2.com in the browser, it will add www to the url. But note here that if we type [*https://site2.com*](https://site2.com)

It will again redirect us to <http://www.site2.com>. **It will rip off the https and replace it with http.** Lets modify the rule. Open the rule and then add a condition.

Condition Input: {HTTPS}

Check if input string : Matches the pattern

Pattern: Off

Let’s create another rule now as previously made with [www.site2.com](http://www.site2.com)

This time now open the rule and add the same condition for HTTPS with the pattern **ON.** Change the Redirect URL to https://<rest of the path>

By default we will get very ugly names when creating Canonical Host name rules. All the rules created in the IIS are added into the web.config file.

Go to the weeb.config file and change the rule name as per convenience. This will reflect back in IIS.

*Let’s add another User-Friendly Url rule.*

Let change the url path containing querystring into something more convenient. Select the template and it will ask for an example of the internal url. Provide the functioning url for the site as the example. Then select the options ofr the url rewrite as per your convenience. ***This is necessary as querystring shows the variable names to the end user in the url. We don’t want that. That is bad coding.***

*Let’s create another rule: whenever soeone goes to the login page we are going to redirect them to https:*

Add a blank rule.

Name: login Traffic to https

Match on the pattern using regular expression to ^login.htm

Select action type to redirect and set the redirect url to [https://site2.com/{R:0}](https://site2.com/%7bR:0%7d)

Add the below condition otherwise you will end up in the infinite loop of redirection.

Condition Input: {HTTPS}

Check if input string : Matches the pattern

Pattern: Off

So what we are doing is,if it is already on HTTPS, we are not going to match the rule. Because Https already has SSL. We are only going to match if we are not on https

**IIS Express:**

Install IIS Express

After installing IIS, the easiest way to use it is through Visual studio.

If launching you application now does not recognize system.webserver steeings of the web config then just right click on the web project in the solution explorer and select Use IIS Express.

IIS Express also has url rewrite built in. you also need to provide the port for redirection when using url rewrite of IIS express.

Also IIS Express creates a Logs in the my documents folder for your application. It also create the trace log files for you in the same folder.

You can also turn on SSL by right clicking the project folder in the solution explorer. But it will give you a certificate warning. To avoid that. Open MMC (Microsoft Management Console) and click File and select Add/Remove snapIn.

Add the certificate snapin on a computer account. Then go to personal certificates. Select the localhost certificate and copy this to the Trusted Root Certification Authorities folder. This way we can get our site to have SSL without any certificate warning.

**IIS 8:**

New features in IIS 8 are

1. **Application Initialization Mode** – this allows your site to warm up as soon as the iis loads instead of waiting for the first user to show up. This way the first user will not have a poor experience of waiting till the site warms up and loads everything for the first time. What Application Initialization Mode does is it makes the forst request automatically happes as soon as the IIS loads.

How this happens is that, when you go to the site in the browser, its application pool loads into the task manager with the process name as w3wp.exe *which is a IIS Worker Process.* This process is actually the instance of the application pool running.

Now if you open the cmd prompt and run the iisreset command to stop the iis and then turn it back on, you will see that the w3wp process is killed and is not turned on automatically. Now if you refresh the website, you will see the process kicked back in the task manager. So the worker process came in when the first user hit the site.

This is where application Initialization mode comes in for help. It starts the worker process by making the first request automatically to the site.

Select the application pool and go to advance settings. In General, select the start Mode and change it to Always Running from OnDemand. By default, it is set to OnDemand.

If you have multiple application pools, you will see multiple w3wp processes.

Now go to the site and click advance settings. Set Preload Enabled to True. By default, it is set to false.

Now suppose, some user hits the site when the site is warming up. It can happen in production. We cannot solve this other than optimising the site to load faster. We cannot solve it on individual web server. **We can solve this though in a web farm scenario. But we can at least provide the user with a message with what is happening. Why the page is taking longer to load.**  We do this in the web.config file.

First create a html page of the message you want to display . this has to be a html page as the aspx and other engine would ot have load by now. Then add the below line to the web.config file.

<applicationInitialization remapManagedRequestsTo=”Startup.html” skipManagedModules=”True”>

<add initialicationPage=”/defaule.aspx”/>

</applicationInitialization>

1. **SNI (Server Name Indication) Support** – this allows us to host multiple ssl certificates on the same ip address. Without SNI, we either have a wildcard certificate that covers a single domain space, or we have to have separate IP Addresses for each individual site. For example, we have a certificate for site1.com and a certificate for site2.com, if we want to host those on a web server, they would have to have two distinct Ip Address which can be a problem because public IP Addresses cost you real money. This is because, a computer has only one IP Address. This means we have to have two different servers to host the two applications.

***SNI does not support any version of internet Explorer on Windows XP.*** Let’s now configure the SNI

Open MMC (Microsoft Management Console) and click File and select Add/Remove snapIn. Now go to the personal certificates and then create a certificate for both site1.com and site2.com separately.

Now go to the iis manager and select the site and then click onBindings from the right panel. Click on Add and then select the type as https, ip address as All Unassigned, Port as 443. You will see a new checkbox named as Require Server Name Indication. Select the hostname to be site1.com and then check the checkbox.

Select the site1.com SSL Certificate.

Do the same for the rest of the sites.

*Now if you see a warning in the right panel saying “No default SSL site has been created”,* what it means that is for browsers that does not support SNI, IIS should have a default site to reach to that is bound to that IP Address not requiring SNI that it will reach, you can inform them that you need to upgrade the browser

1. SSL Management
2. CPU Throttling – this will help us in managing website in case of overloading.
3. Web Sockets

**SSL Management, CPU Throttling and Web Sockets:**

Go to the application root and select Centralized Certificates. So if I have a web farm, I have multiple certificates that is going to each node of my Web Farm, I can put it in a centralized location and utilize those certificates from there.

Click Edit Feature Settings.

Specify the centralized physical path of the certificates like fileshare etc, credentials to connect with and that all. More about it in the Web Farm course.

For CPU throttling, go to the application pool to limit the resources used by the pool.

Go to the pool, select Advance Settings and set the Limit to 1000 which is equivalent to 1% of the CPU. Set the limit action which will play when the limit set is reached. That action provided in the limit action will be triggered. It also generates an event log, so we can monitor which sites are consuming CPU before we can set Limit Action from NOAction to any other value. The other options are, KIllw3wp, Throttle, ThrottleUnderLoad. Throttle means do not let this application pool use more than the percentage provided in the Limit.

ThrottleUnderLoad is more of the real world scenario. Let’s say we have two we servers and we want them to be equally weighted, we can go and set each of them to use 50% of the cpu but if one of the sites is not currently in use then that site has literally 0 traffic then the first site could basically use the 100% of the CPU. As we have set the liit of 50% for both the site to equally weight them, in this case, it will not be able to use more then 50 % and that is not efficient. Setting the setting to ThrottleUnderLoad makes it possible to use 100% CPU if the other application pool is not in play. And once the other pools starts then it wil throttle it down to the limit.

Limit Interval indicates over what period of time are we going to measure the CPU

The concept of web sockets is to have a real time connection, a persistant connection to the web server. Just enable web sockets while installing the IIS windows feature under the Application Development Server roles.