Setup Shinydocs Visualizer Behind Reverse Proxy in IIS

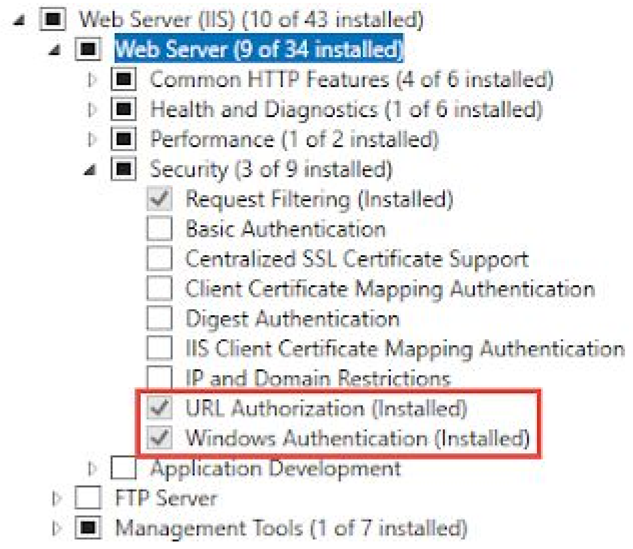
User Guide

Table of Contents

Setting up IIS

To set up IIS, you will need to make a site and download some additional features.

1. In **Server Manager**, go to **Manage > Add Roles and Features**
2. Select "**Role-based or feature-based installation**", click Next
3. Select the server from the server pool, click Next
4. You should now be in **Server Roles**. Make sure you have **URL Authorization** and **Windows Authentication** checked



- ■ Web Server (IIS) (10 of 43 installed)
  - ■ Web Server (9 of 34 installed)
    - ■ Common HTTP Features (4 of 6 installed)
    - ■ Health and Diagnostics (1 of 6 installed)
    - ■ Performance (1 of 2 installed)
    - ■ Security (3 of 9 installed)
      - ✓ Request Filtering (Installed)
      - ☐ Basic Authentication
      - ☐ Centralized SSL Certificate Support
      - ☐ Client Certificate Mapping Authentication
      - ☐ Digest Authentication
      - ☐ IIS Client Certificate Mapping Authentication
      - ☐ IP and Domain Restrictions
      - ✓ URL Authorization (Installed)
      - ✓ Windows Authentication (Installed)
    - ☐ Application Development
  - ☐ FTP Server
  - ■ Management Tools (1 of 7 installed)

5. Proceed through the remainder of the wizard
6. Download and install:

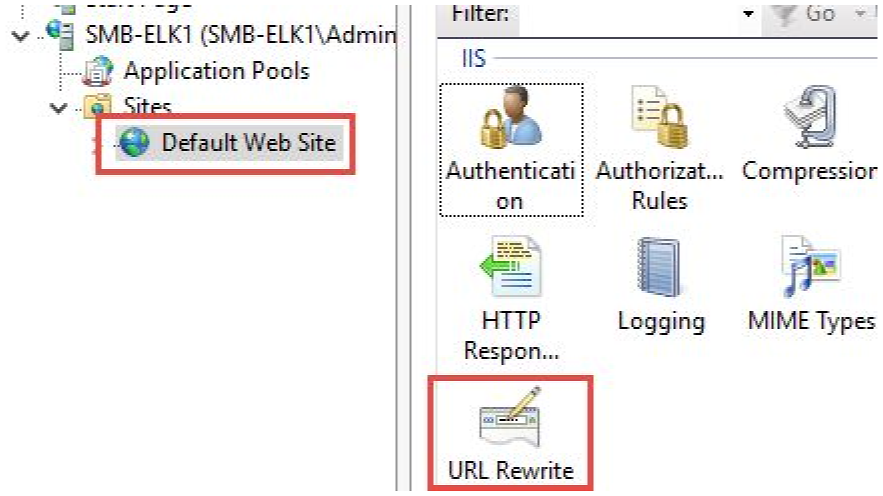
- URL-Rewrite: https://www.iis.net/downloads/microsoft/url-rewrite
- Application Request Routing: https://www.iis.net/downloads/microsoft/application-request-routing

Make your Site

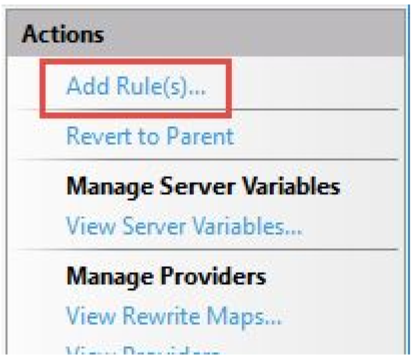
1. Open IIS Manager
2. Right-click on "**Sites**" and click **Add Website**
3. Give the site a name, DefaultAppPool
4. Give a physical path where the site will be located
5. For bindings:
   Type: http (https can be used here, you will need a cert)
   IP address: All Unassigned
   Port: 80 is default, this can be changed if needed
   Host name: Specify a host name
6. Now you have your site!

Configure the IIS Reverse Proxy

1. Launch **IIS Manager** and select the website you'll be configuring as the reverse proxy. Click on the **URL Rewrite** feature in the center panel.

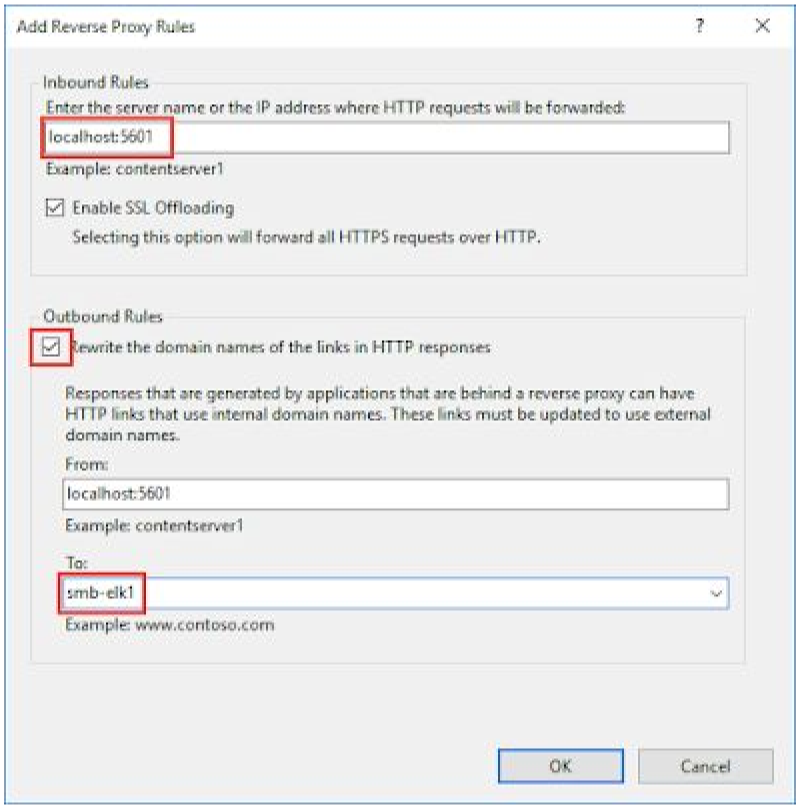


2. Then, **Add Rule(s)...** in the Actions panel on the right.



3. In the Add Rule(s) dialog, select **Reverse Proxy** and click OK.
4. Click **OK** again
5. In the **Add Reverse Proxy Rules** dialog under **Inbound Rules**, we'll give it our Kibana URL (localhost:5601) as the location where requests will be forwarded. We also want to enable Rewriting of domain names under Outbound Rules and populate the external URL for our server under the To: field. In this case the external URL will be whatever our clients on the network will type into their browsers to access Kibana. I'm just using the server name in my

lab environment. Click OK to complete the dialog.


Add Reverse Proxy Rules

Inbound Rules

Enter the server name or the IP address where HTTP requests will be forwarded: localhost:5601

Example: contentserver1

☑ Enable SSL Offloading

Selecting this option will forward all HTTPS requests over HTTP.

Outbound Rules

☑ Rewrite the domain names of the links in HTTP responses

Responses that are generated by applications that are behind a reverse proxy can have HTTP links that use internal domain names. These links must be updated to use external domain names.

From: localhost:5601

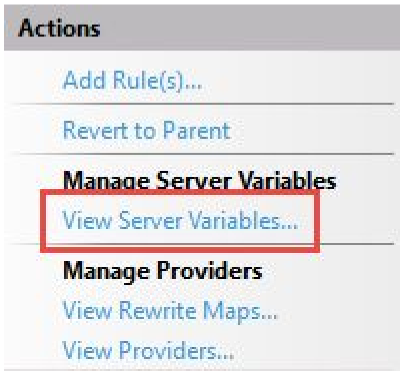
Example: contentserver1

To: smb-elk1

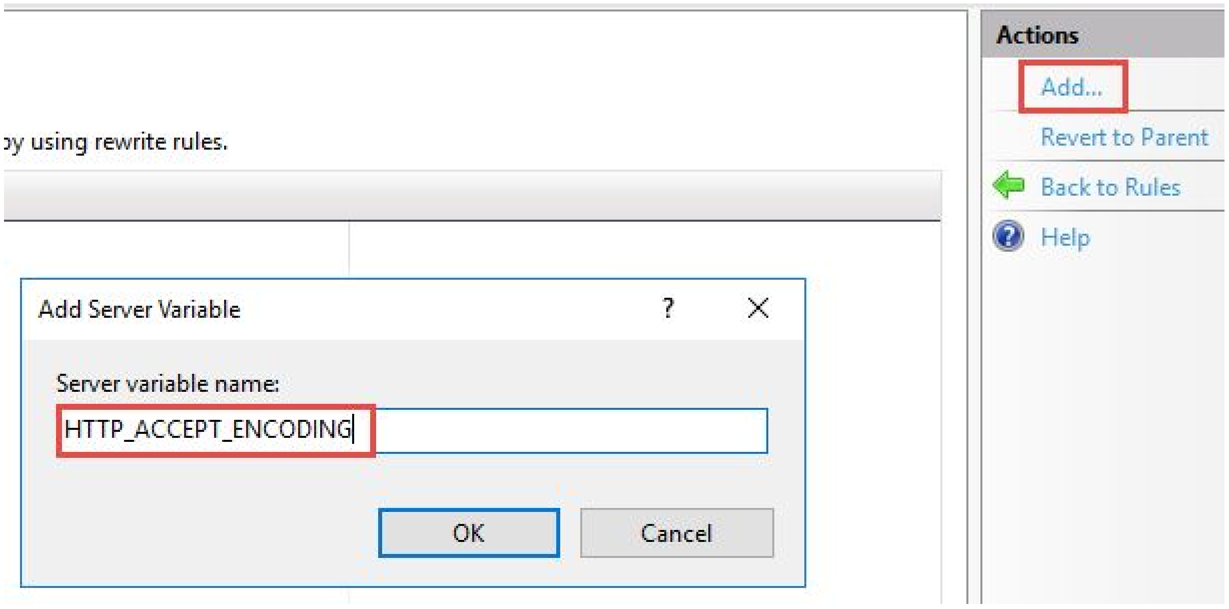
Example: www.contoso.com

Configure Server Variables

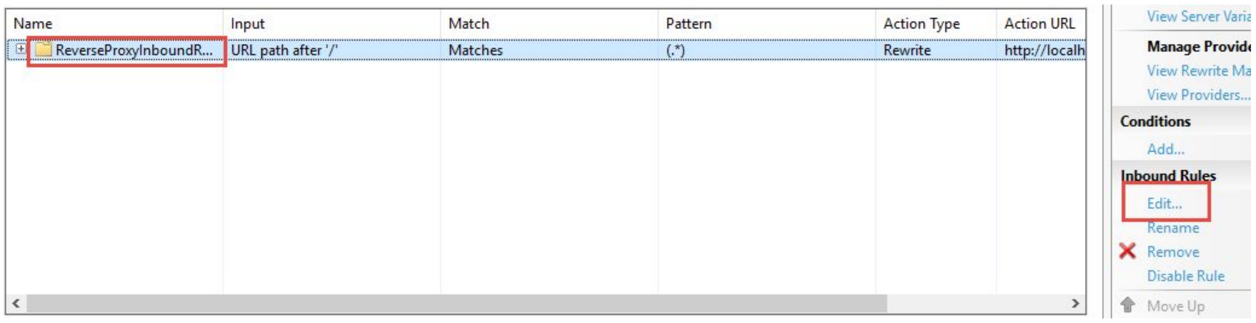
1. With our website selected let's go back to the URL Rewrite module. This time we'll choose View Server Variables…



2. On the Allowed Server Variables screen, choose Add… to add a new server variable called **HTTP_ACCEPT_ENCODING**, and click OK. Follow the same process to add a second variable called **HTTP_X_ORIGINAL_ACCEPT_ENCODING**.

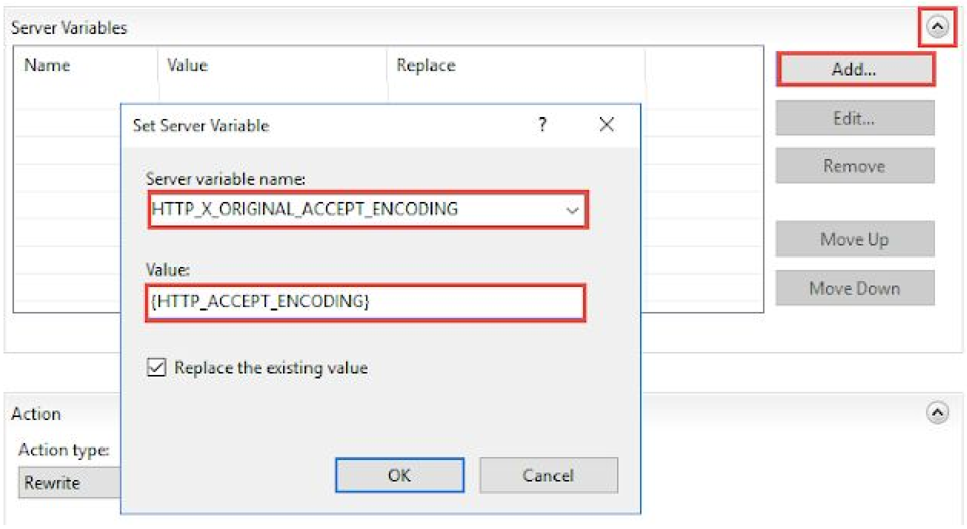


3. Next, go back to URL Rewrite rules and select the inbound rule. Then click **Edit…**



Name	Input	Match	Pattern	Action Type	Action URL
ReverseProxyInboundR...	URL path after '/'	Matches	(.*)	Rewrite	http://localh

4. On the Edit Inbound Rule screen, expand the **Server Variables** section and click **Add…** Select the **HTTP_X_ORIGINAL_ACCEPT_ENCODING** variable that we created earlier from the Server variable name: dropdown box. Under Value: type **{HTTP_ACCEPT_ENCODING}**. Be sure to include the curly braces so the rule knows to use the value of that variable. Click OK.

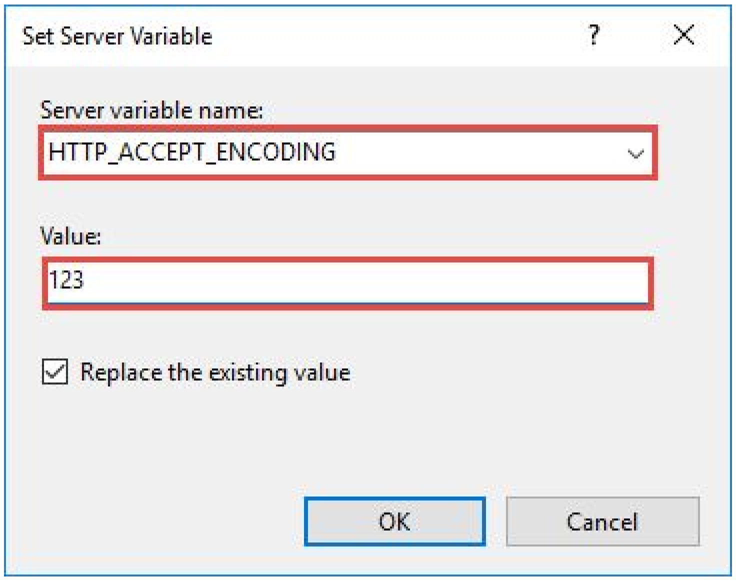


Server variable name: HTTP_X_ORIGINAL_ACCEPT_ENCODING

Value: {HTTP_ACCEPT_ENCODING}

☑ Replace the existing value

5. Click Add… again to add another server variable. This time select **HTTP_ACCEPT_ENCODING** from the drop down box, and type any text value into the Value: field. What we need to do here is set the value of this variable to be empty, but this field won't accept a blank value so we're giving it any text value so we can save the variable, and we'll update the value in the next step. I typed "**123**" as my value.

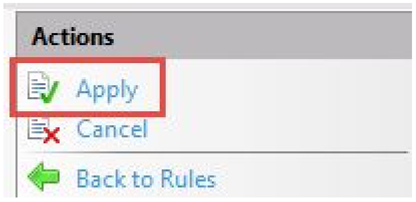


Server variable name: HTTP_ACCEPT_ENCODING

Value: 123

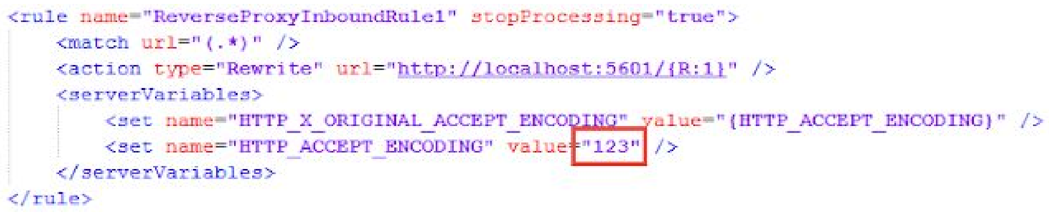
☑ Replace the existing value

6. With both variables set, click **Apply** in the Actions panel.



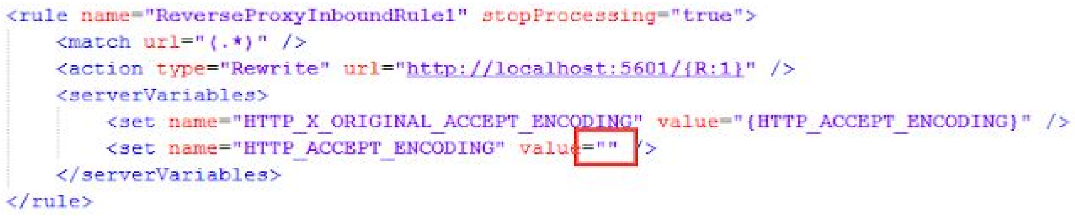
7. Now we need to replace our arbitrary text value (“**123**” in my case) with a blank. This is done in our website’s **web.config** file. Since I’m using the Default Web Site, that’s located in **C:\inetpub\wwwroot**. Open the web.config file in a text editor and find the text value that you entered. Select the value between the quotes and delete it, leaving just the quotes.

Before:



```
<rule name="ReverseProxyInboundRule1" stopProcessing="true">
    <match url="(.*)" />
    <action type="Rewrite" url="http://localhost:5601/{R:1}" />
    <serverVariables>
        <set name="HTTP_X_ORIGINAL_ACCEPT_ENCODING" value="{HTTP_ACCEPT_ENCODING}" />
        <set name="HTTP_ACCEPT_ENCODING" value="123" />
    </serverVariables>
</rule>
```

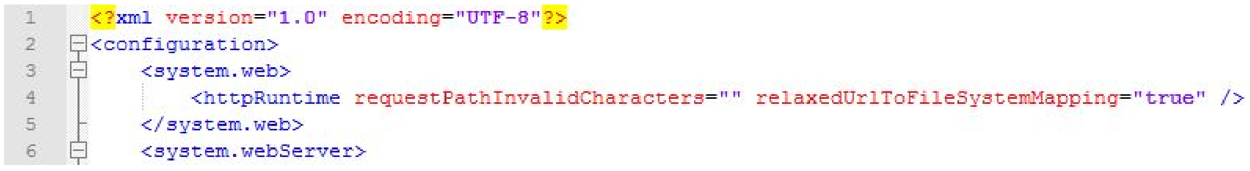
After:



```
<rule name="ReverseProxyInboundRule1" stopProcessing="true">
    <match url="(.*)" />
    <action type="Rewrite" url="http://localhost:5601/{R:1}" />
    <serverVariables>
        <set name="HTTP_X_ORIGINAL_ACCEPT_ENCODING" value="{HTTP_ACCEPT_ENCODING}" />
        <set name="HTTP_ACCEPT_ENCODING" value="" />
    </serverVariables>
</rule>
```

1. In web.config, you will also need to add (after <configuration> and before <system.webServer>:

```
<system.web>
<httpRuntime requestPathInvalidCharacters=""
relaxedUrlToFileSystemMapping="true" />
</system.web>
```

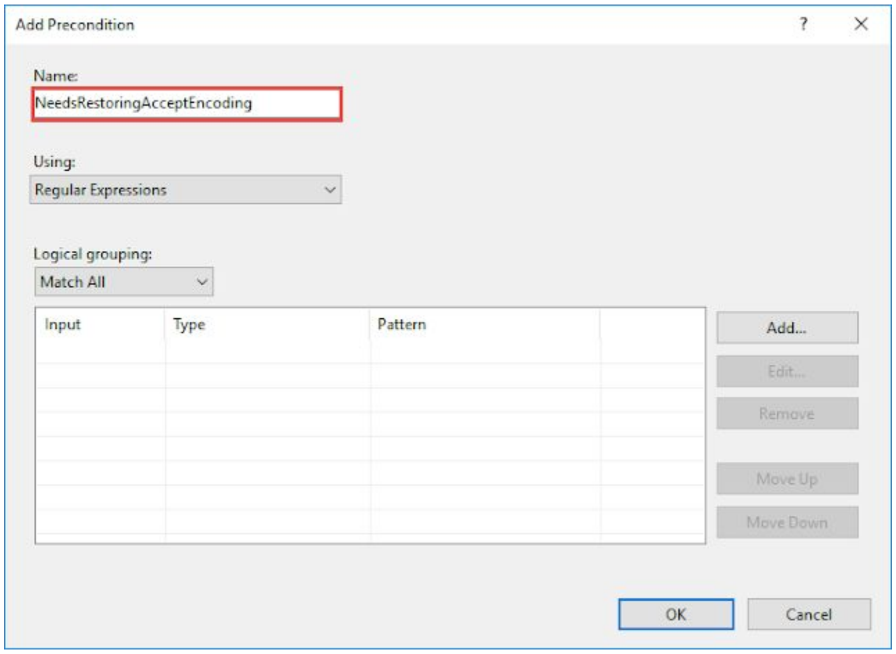


```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.web>
        <httpRuntime requestPathInvalidCharacters="" relaxedUrlToFileSystemMapping="true" />
    </system.web>
    <system.webServer>
```

1. Save the file.

That addresses the inbound portion of our configuration, now we need to address outbound traffic.

1. Under **URL Rewrite**, click **Add Rule(s)...** again, this time selecting **Blank rule** under Outbound rules.
2. We’ll name our new rule **RestoreAcceptEncoding**, and select **<Create New Precondition…>** from the drop-down menu. On the Add Precondition screen, provide the name **NeedsRestoringAcceptEncoding** and **ensure Regular Expressions** is selected from the **Using:** drop-down menu.


Add Precondition

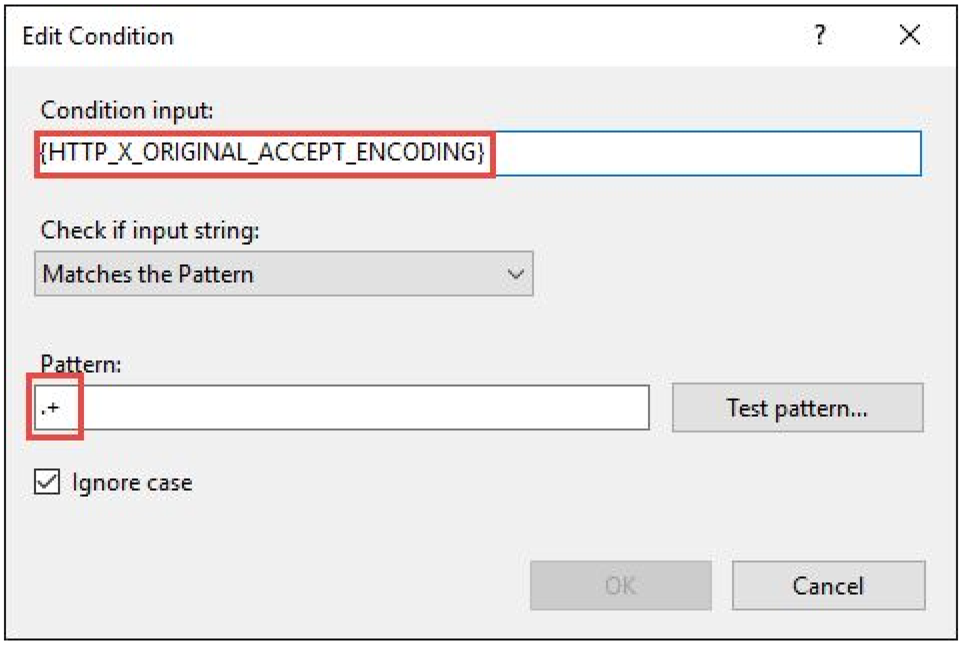
Name: NeedsRestoringAcceptEncoding

Using: Regular Expressions

Logical grouping: Match All

Input	Type	Pattern

3. Click **Add…** to add a new condition. For the Condition input: type **{HTTP_X_ORIGINAL_ACCEPT_ENCODING}**, again making sure to include the curly braces. Under pattern, type '**.+**'. Click OK. Click OK again to close the Add Precondition dialogue.


Edit Condition

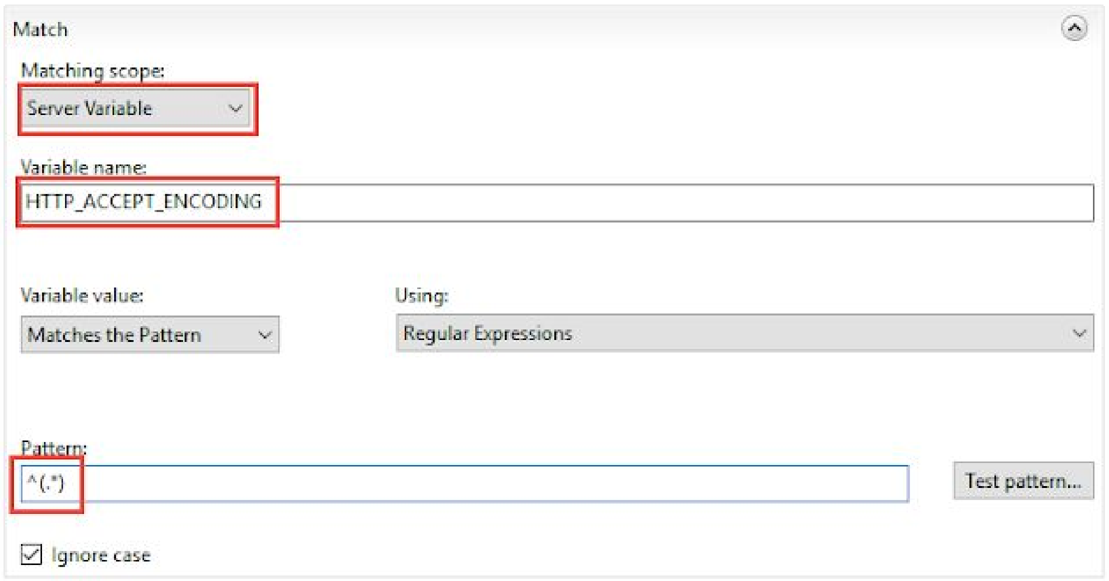
Condition input: {HTTP_X_ORIGINAL_ACCEPT_ENCODING}

Check if input string: Matches the Pattern

Pattern: .+

☑ Ignore case

4. Still under the Edit Outbound Rule screen, find the Match section and set the Matching scope: to Server Variable. Type **HTTP_ACCEPT_ENCODING** as the Variable name:. For the pattern, type '**^(.*)**'.


Match

Matching scope: Server Variable

Variable name: HTTP_ACCEPT_ENCODING

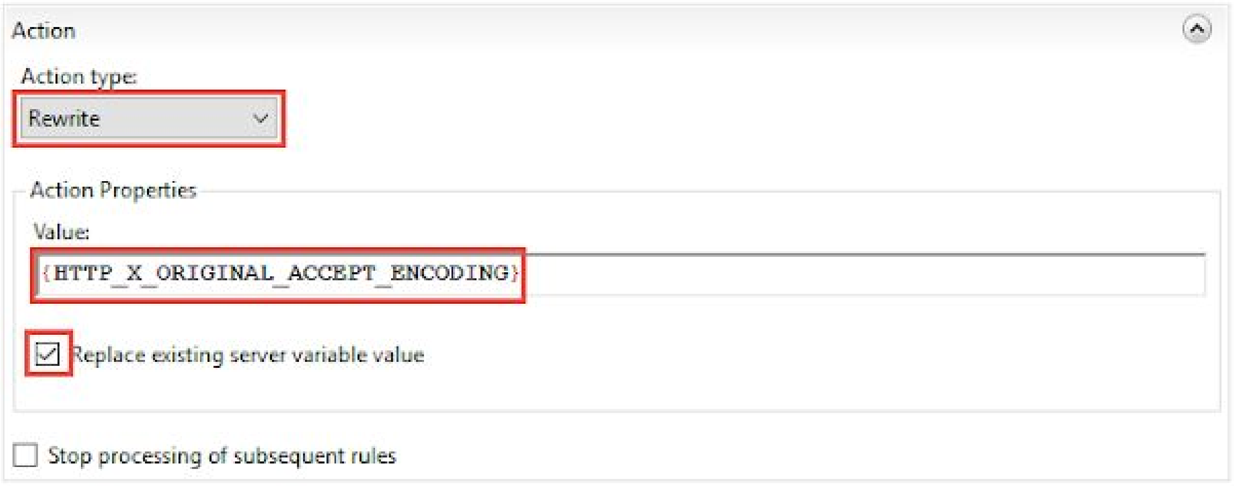
Variable value: Matches the Pattern

Using: Regular Expressions

Pattern: ^(.*)

☑ Ignore case

5. Lastly under the Action section, ensure that Action type: is **set to Rewrite**. For the Value: type **{HTTP_X_ORIGINAL_ACCEPT_ENCODING}**, again being sure to include the curly braces. Then, click Apply.


Action

Action type: Rewrite

Action Properties

Value: {HTTP_X_ORIGINAL_ACCEPT_ENCODING}

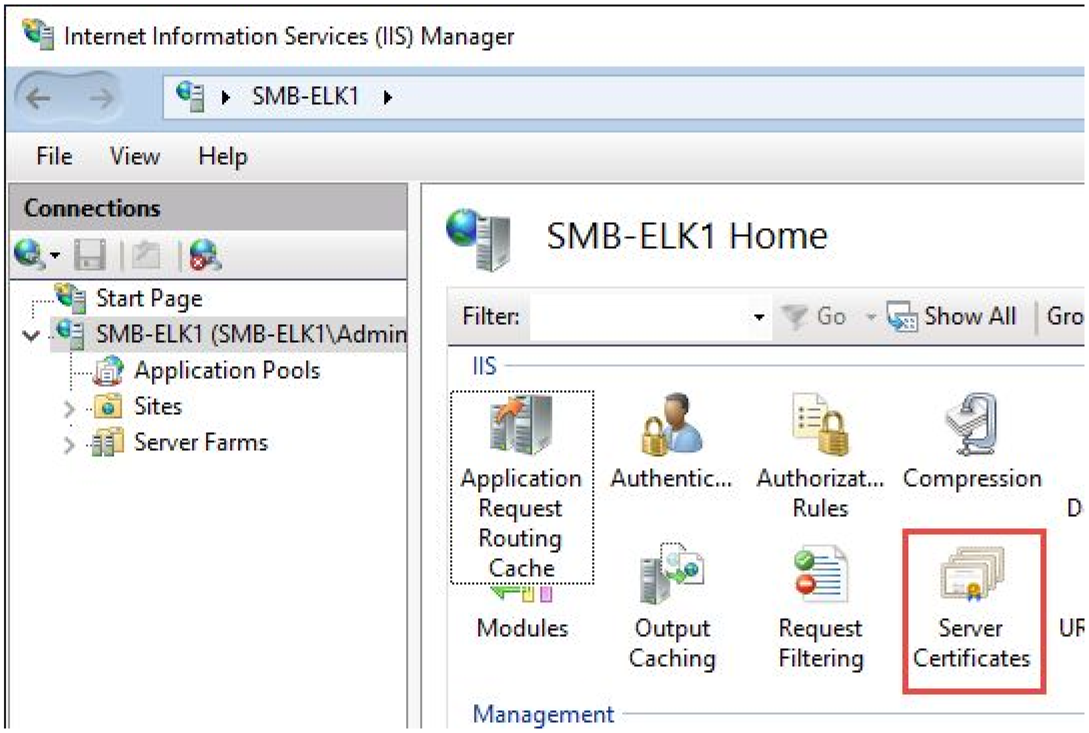
☑ Replace existing server variable value

☐ Stop processing of subsequent rules

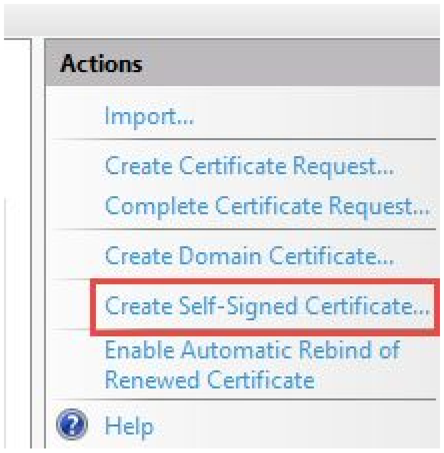
If everything has gone according to plan, reverse proxying from IIS to Kibana should now be working. If you type http://localhost
into a web browser on the Elastic server, you should see Kibana being served via IIS over port 80.

Configure SSL Certificate

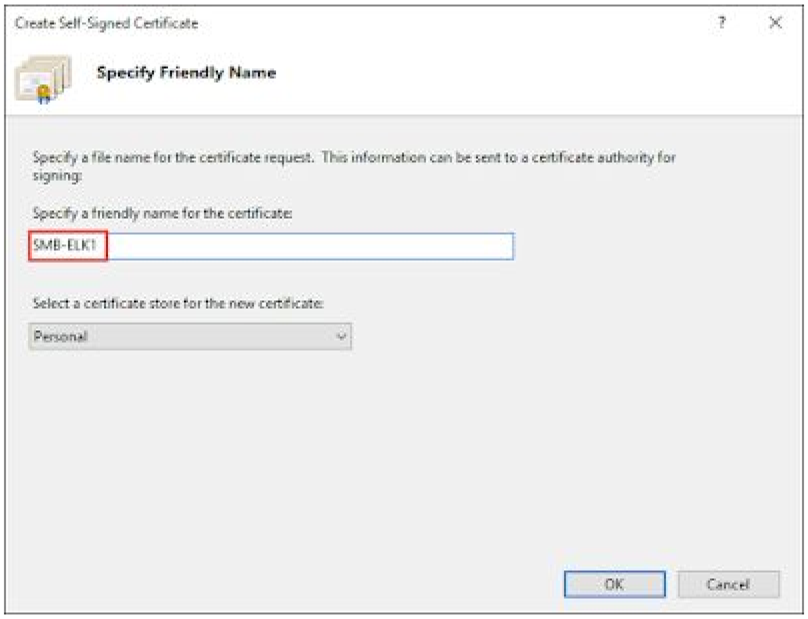
1. Select the server name in the left-hand panel, and then choose the Server Certificates option.



2. We then choose Create Self-Signed Certificate… from the Actions pane.


Actions
Import...
Create Certificate Request...
Complete Certificate Request...
Create Domain Certificate...
Create Self-Signed Certificate...
Enable Automatic Rebind of Renewed Certificate
Help

3. Type the name you want to use for referencing this certificate. I just used the server name. Click OK.


Create Self-Signed Certificate

Specify Friendly Name

Specify a file name for the certificate request. This information can be sent to a certificate authority for signing:

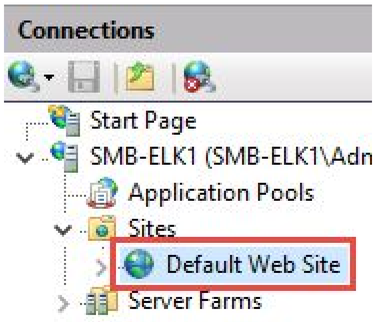
Specify a friendly name for the certificate:

SMB-ELK1

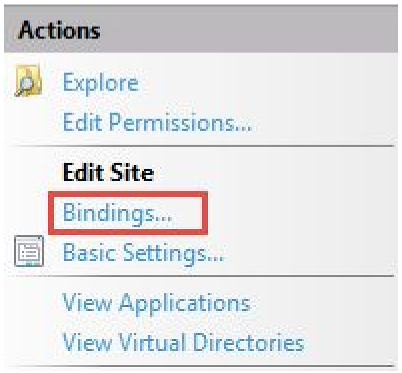
Select a certificate store for the new certificate:

Personal

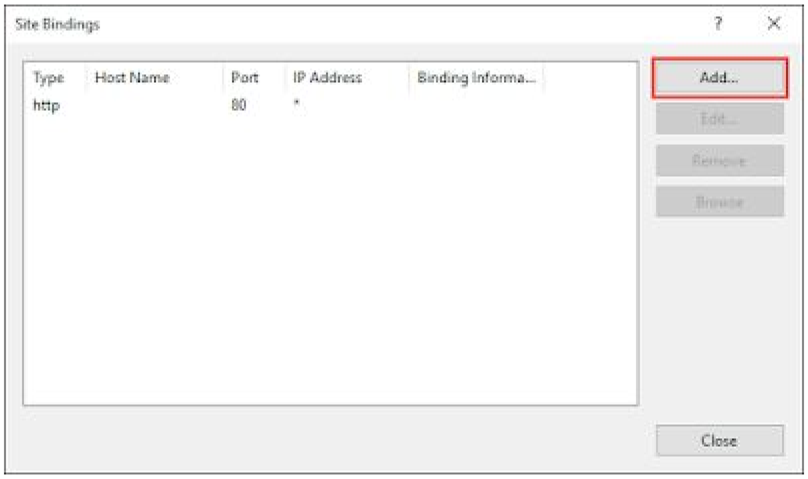
4. With the certificate created, we can go ahead and bind it to our website. To do that, expand the server in IIS and select the website.



5. Then, select Bindings… from the Actions pane.

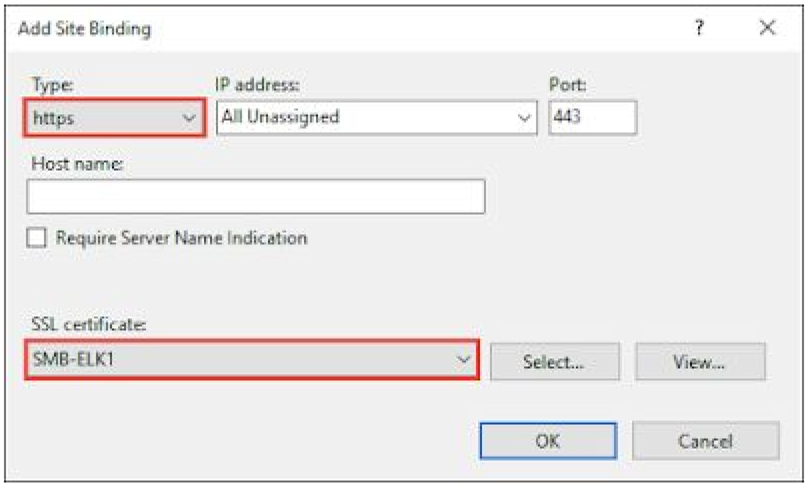


6. On the Site Bindings screen, choose Add…



Type	Host Name	Port	IP Address	Binding Informa...
http		80	*	

7. On the Add Site Binding screen, choose HTTPS as the type and select your certificate from the SSL certificate: dropdown menu.

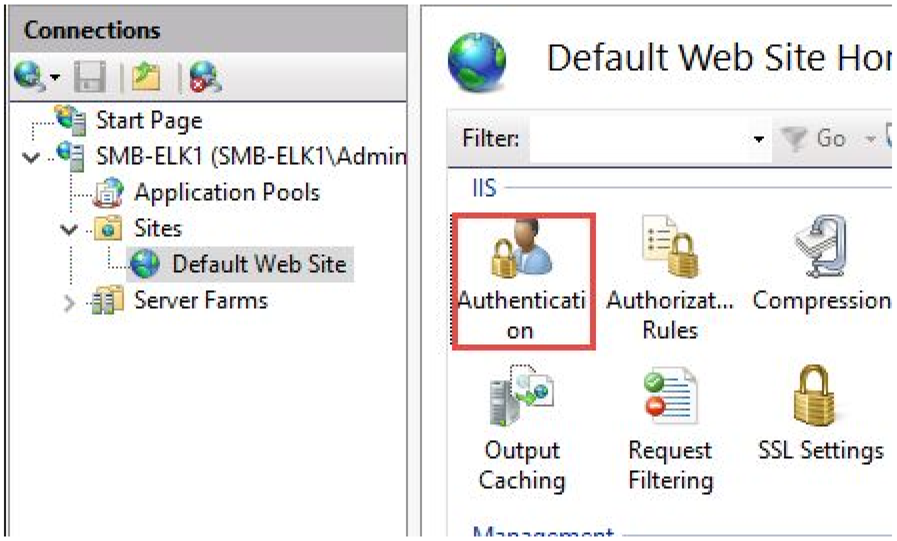


Type: https
IP address: All Unassigned
Port: 443
Host name:
☐ Require Server Name Indication
SSL certificate: SMB-ELK1

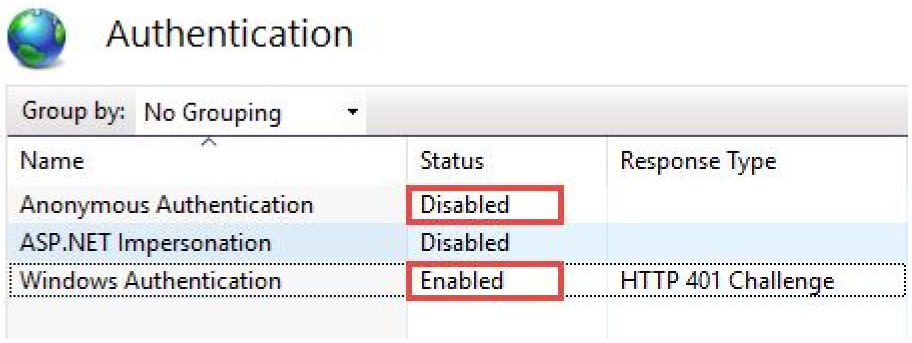
8. Click OK again to add the site binding, and then click Close to close the Site Bindings screen. Now we'll be able to access our website over HTTPS.

Configuring User Authentication

1. Back on the Visualizer server in IIS, we need to select our website and choose the Authentication option.

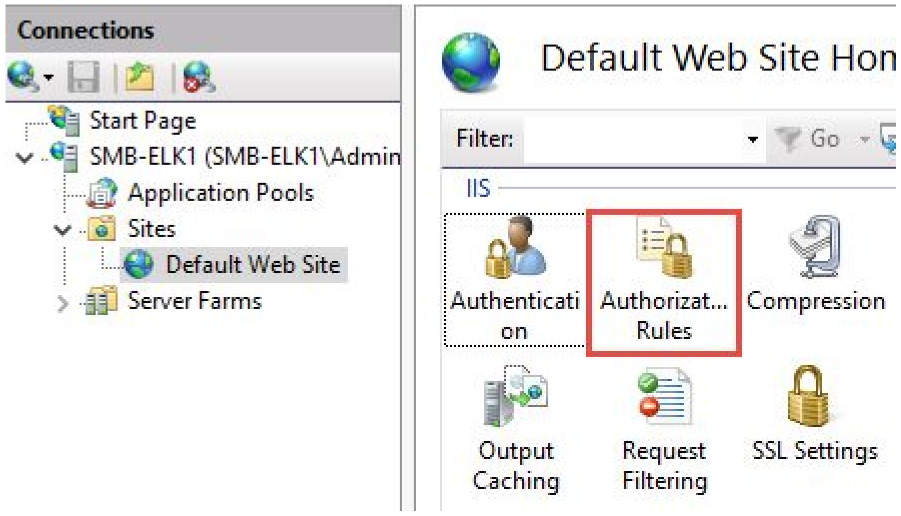


2. Within Authentication, we need to set Anonymous Authentication to Disabled, and set Windows Authentication to Enabled.

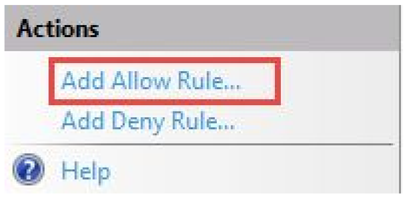


Name	Status	Response Type
Anonymous Authentication	Disabled	
ASP.NET Impersonation	Disabled	
Windows Authentication	Enabled	HTTP 401 Challenge

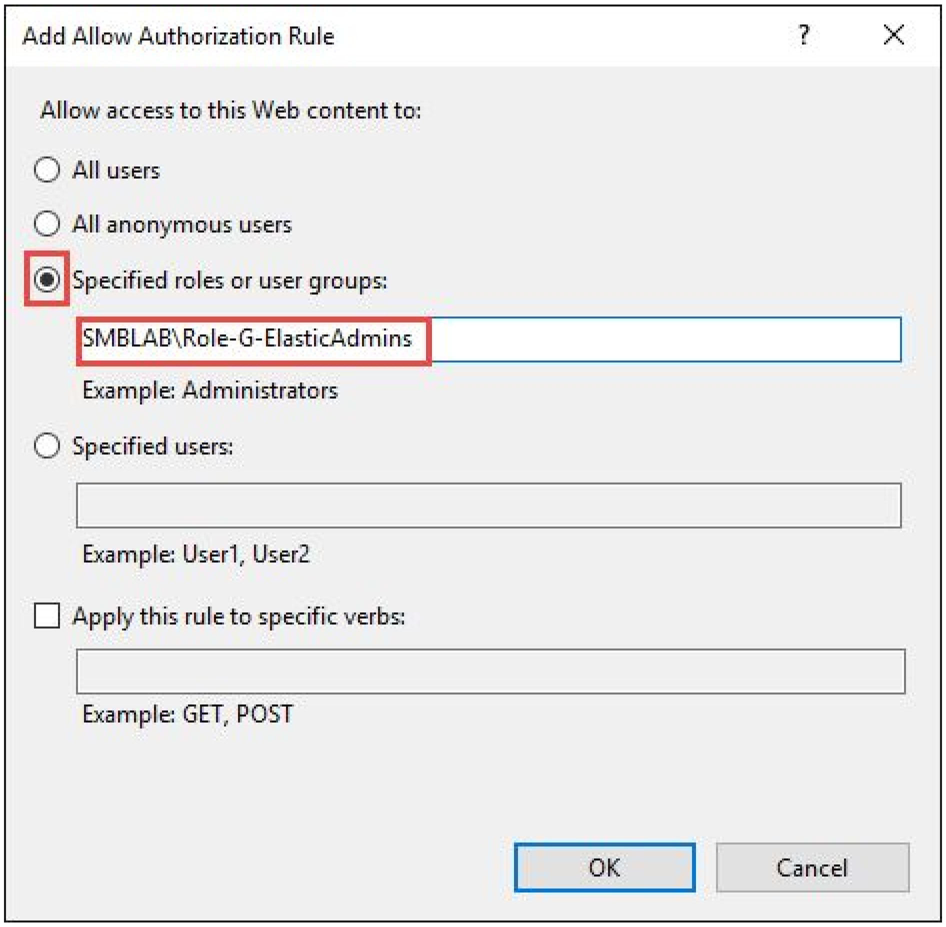
3. Back to the main IIS screen, we'll now select Authorization Rules.



4. We need to delete the Allow -> All Users rule that is created by default. Then, click Add Allow Rule… in the Actions pane.



5. On the Add Allow Authorization Rule dialogue, we want to select the radio button for Specified roles or user groups, and type the name of the group for which we're allowing access. Then, click OK.


Add Allow Authorization Rule

Allow access to this Web content to:

( ) All users
( ) All anonymous users
(●) Specified roles or user groups:
SMBLAB\Role-G-ElasticAdmins
Example: Administrators
( ) Specified users:
Example: User1, User2
☐ Apply this rule to specific verbs:
Example: GET, POST

OK Cancel

Guide adapted from:
https://www.smbadmin.com/2017/07/securing-kibana-with-iis-reverse-proxy.html