Contents

[Create SSL endpoint in Azure 2](#_Toc395522834)

[Creating certificates. 2](#_Toc395522835)

[Create DevRoot Certificate 2](#_Toc395522836)

[Create certificate for Azure endpoint 2](#_Toc395522837)

[Import the certificate into Azure 6](#_Toc395522838)

[Configure the project to use SSL in Visual Studio 8](#_Toc395522839)

[Configure SSL for a worker role endpoint 8](#_Toc395522840)

# Create SSL endpoint in Azure

## Creating certificates.

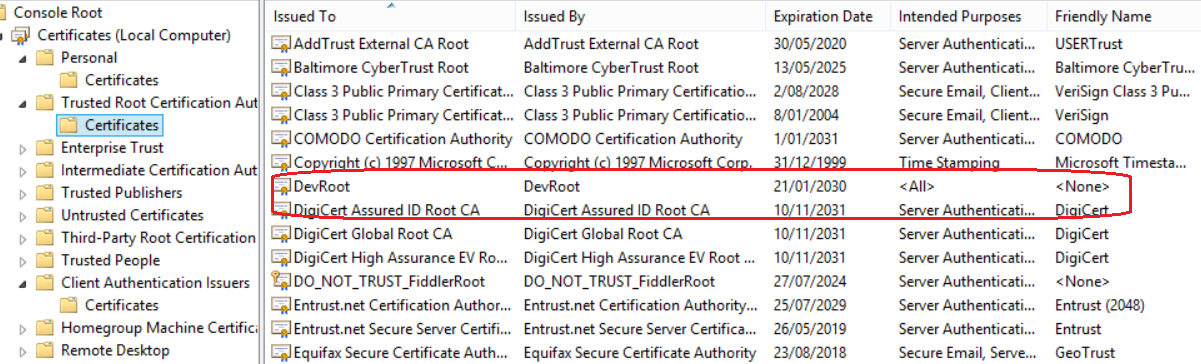
### Create DevRoot Certificate

* 1. Firstly create a root certificate which will then be used to sign other certificates. This will be installed in LocalMachine - Trusted authority. Thus all certificates signed using this will be valid and you need to have only one certificate under trusted root.
  2. Command: We use makecert to create the certificate and use Pvk2pfx to package the private key and the certificate together into a pfx file.

Command that does this: run command prompt and type CreateDevRoot.bat



* 1. When you run this, you will be prompted for a password. You can provide the password here, or ignore it. We will not be uploading this to cloud, so password is not required.
  2. Import DevRoot.cer into the trusted root authority in the certificate store under local machine.



### Create certificate for Azure endpoint

1. Create a certificate for the cloud service signed by the DevRoot certificate. You will need to provide a password here, as you will be uploading this certificate in cloud.

Command: run command prompt and type

CreateSSLCertificate.cmd “[name of the domain of the cloud service]”

Ex: **CreateSSLCertificate.cmd navigatorskeleton.cloudapp.net**



Password: Install02

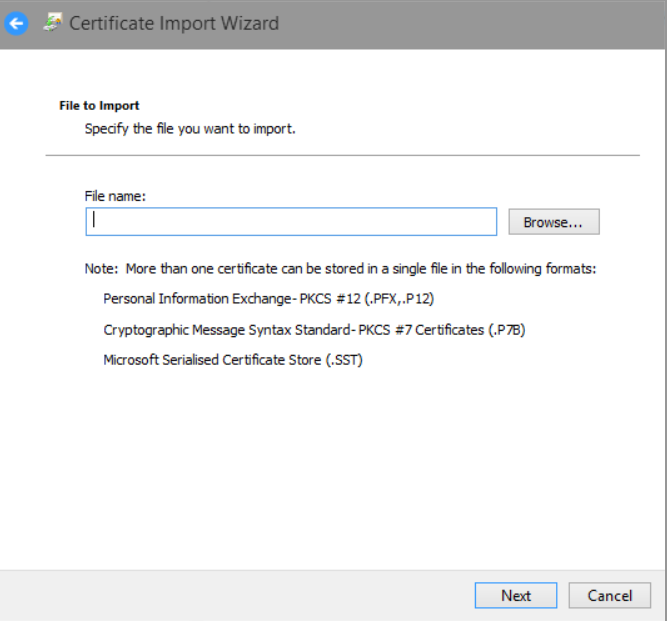
1. Now package this certificate and private key into a single package file. Provide the same password as you used in the previous command.

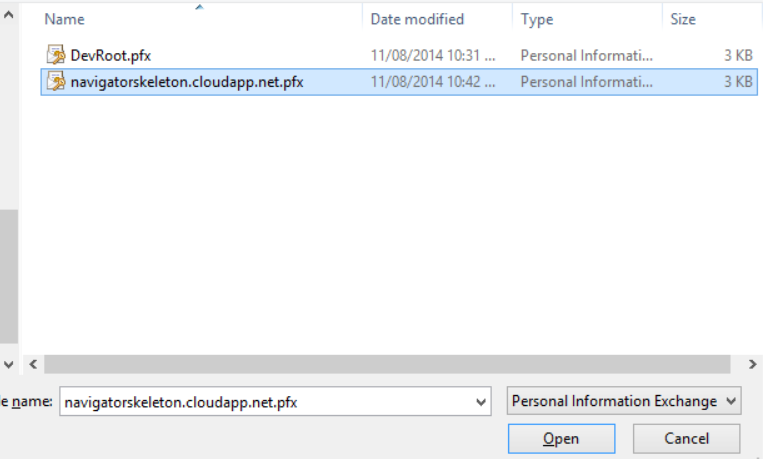
Command: CreatePfx.cmd “[name of the domain of the cloud service]” “[password]”

Ex: **CreatePfx.cmd navigatorskeleton.cloudapp.net Install02**

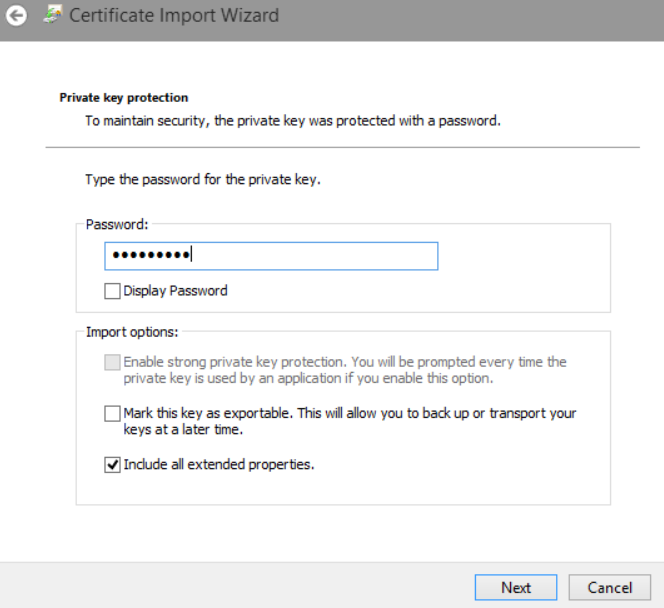


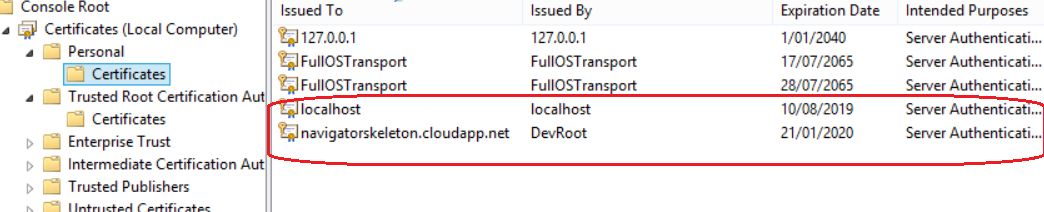
1. Import the certificate under LocalMachine – Personal. Import the .pfx file this time. i.e navigatorskeleton.cloudapp.net.pfx





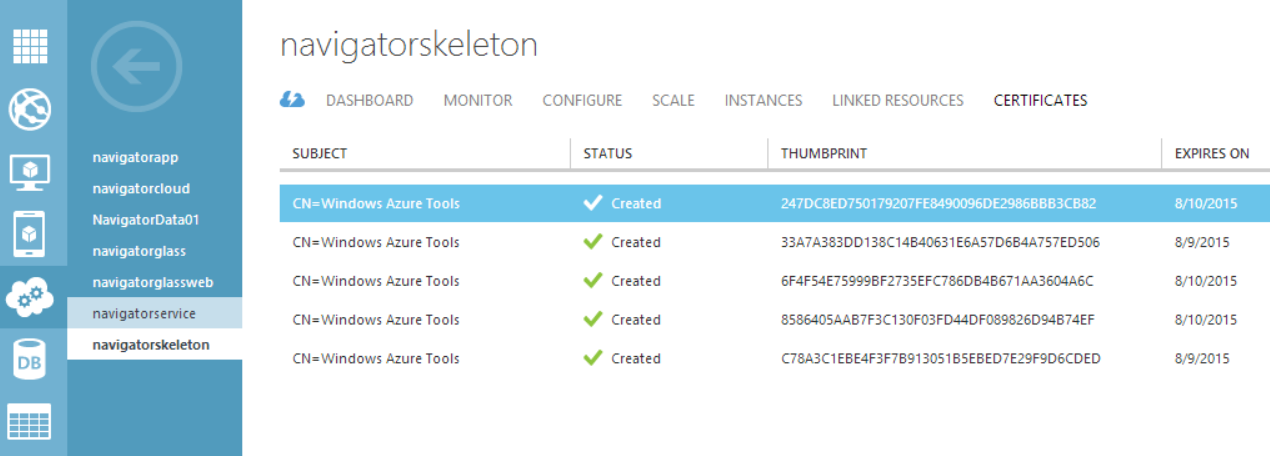
Provide password : Install02



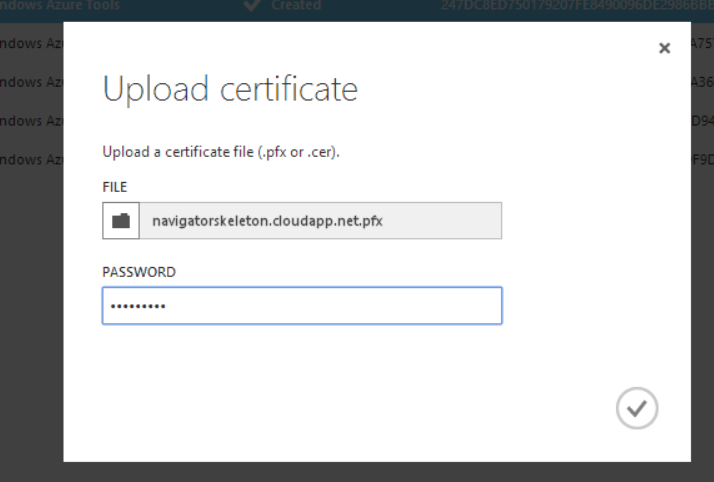


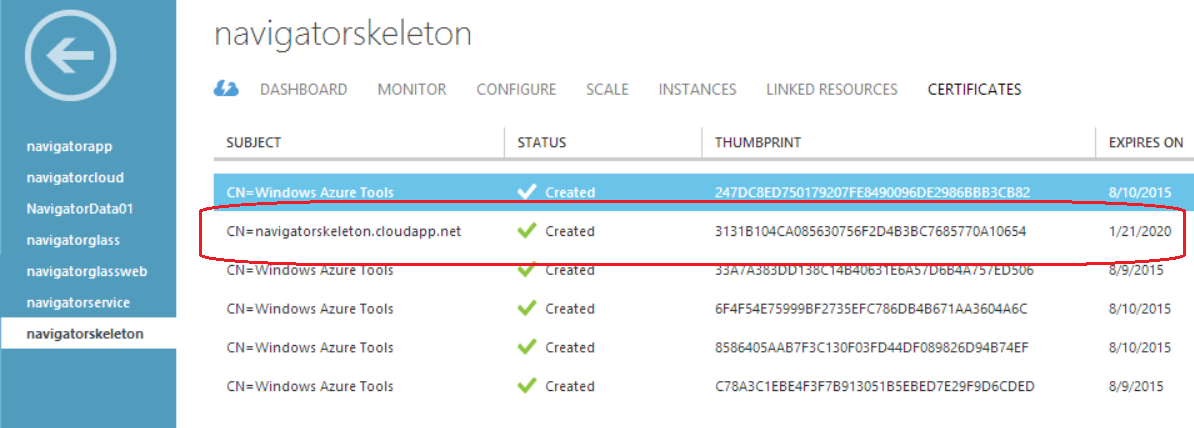
## Import the certificate into Azure

1. Navigate to the cloud service in windows azure portal, under certificates import the domain certificate (pfx). Ex: navigatorskeleton.cloudapp.net.pfx



1. Click on upload, provide the password and import the certificate.



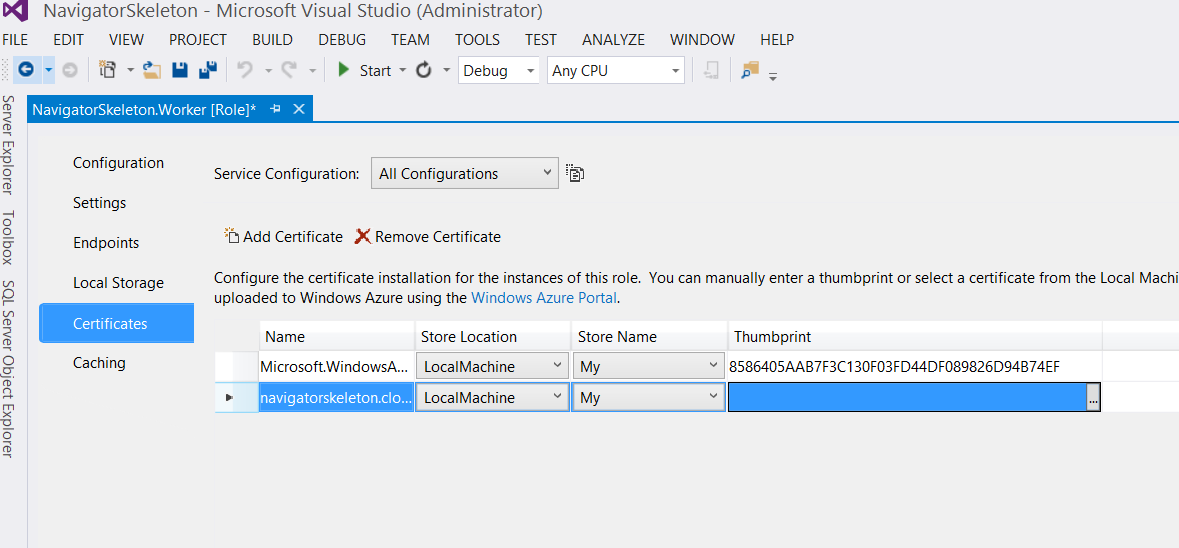


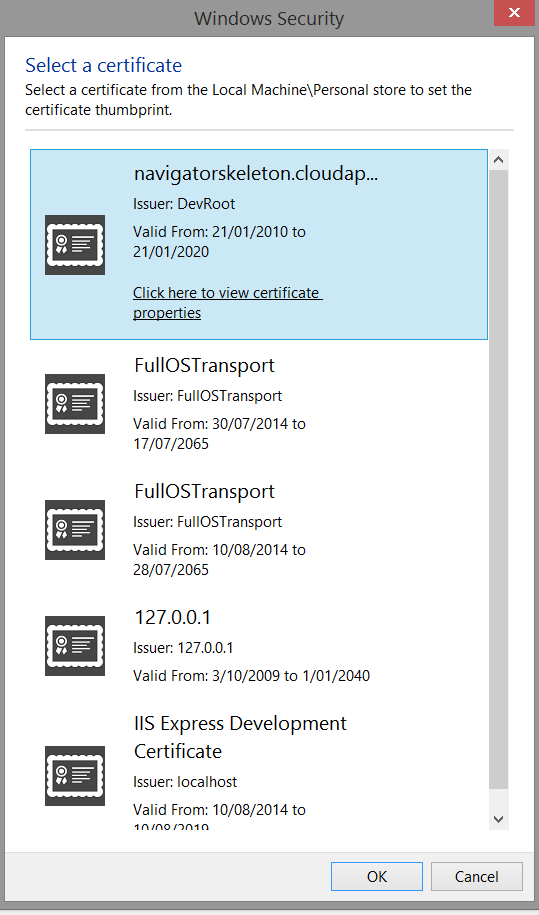
## Configure the project to use SSL in Visual Studio

### Configure SSL for a worker role endpoint

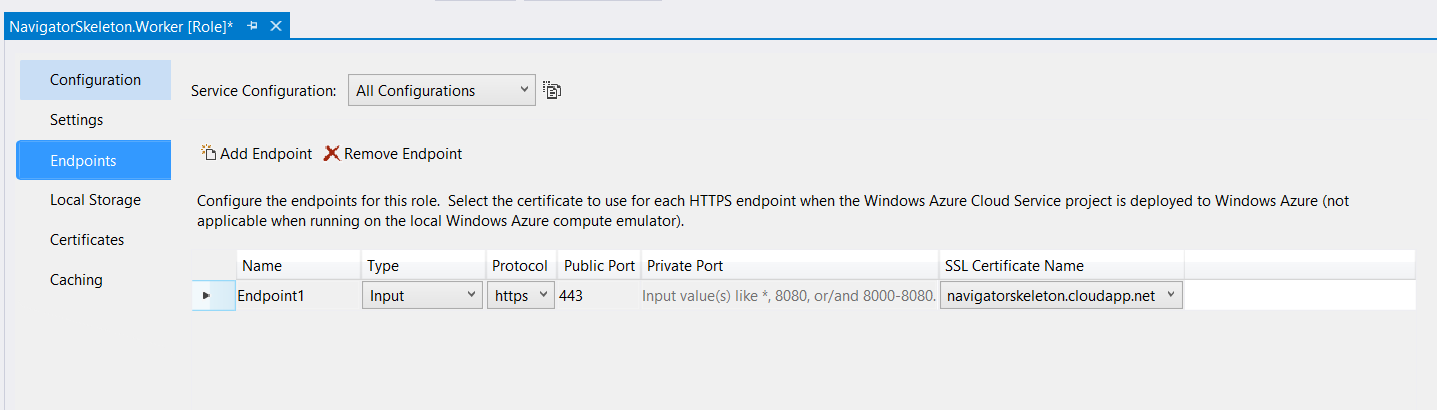
1. Go to properties for the worker role, and then click on certificates. Import the certificate uploaded to azure cloud here.

Click on Add certificate and enter details

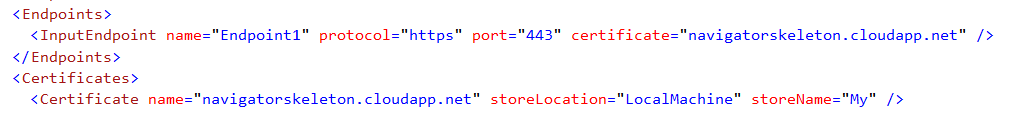




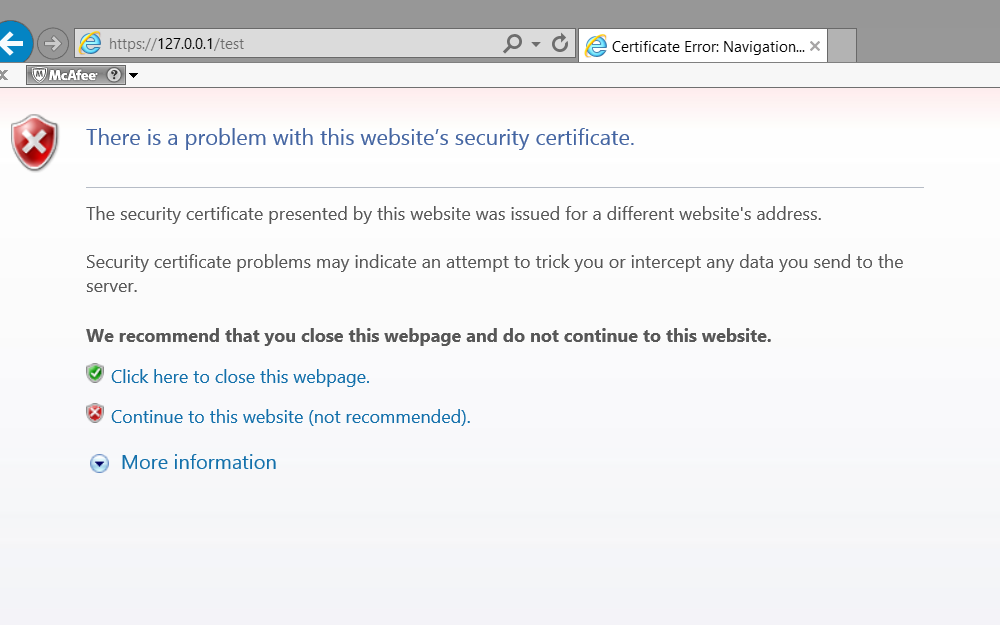
1. Click ok and assign the thumbprint
2. Click on EndPoints under project settings and add https input and associate the certificate from the certificates in the certificate tab.



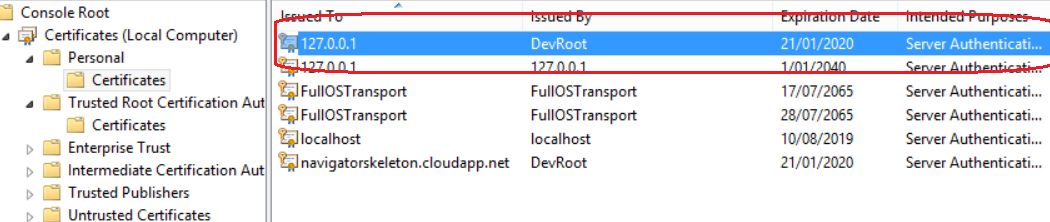
1. This basically adds the following entry in the config files

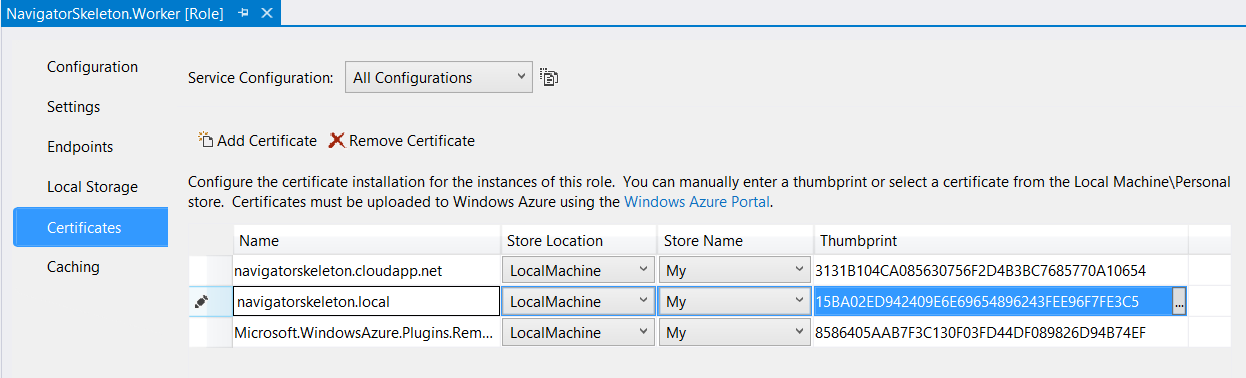


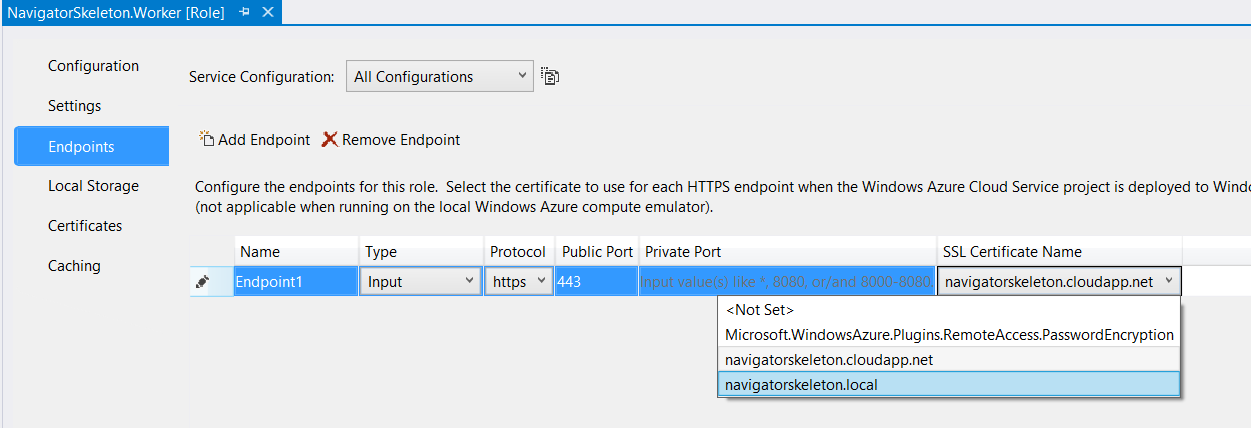
1. Save all the setting and deploy the project. You should be able to navigate to your SSL endpoint. Ex: <https://navigatorskeleton.cloudapp.net>
2. To work with Azure emulator, when you navigate to the site locally, you will get a certificate error. As the certificate used is issued for navigatorskeleton.cloudapp.net and not for local domain.



1. To prevent this, you can create another certificate by running all the steps under (Create certificate for Azure endpoint) but this time give the domain name: 127.0.0.1 and import it into certificate store under personal. When running the local emulator, reset the certificate to point to the local certificate instead.







1. Run the emulator. You will now be able to view this without the error.

Notes: If you face SSL connection error, when running locally, good to check if there is any service that is using the port. And if it is using it, you can kill the process using the PID. You can check this using the following command:

netstat -aon | find ":443"

## Google Mirror and OAuth Configuration



OAuth Application Configuration

