Deploying ASP.NET Core Website on a Debian Linux Server

This document is created as part of a group project implemented by Group 19 in Software Engineering Practice at the University of Canberra.

In the following pages, we will walk you through the steps to deploy a website built in ASP.NET Core on a Linux server instance on Amazon Web Services (AWS).

The high level steps include:

1. Setting up the server
2. Transferring website project to the server
3. Building and configuring the project

Note: This document assumes that you already have an Amazon EC2 (Elastic Cloud Compute) instance running Debian Stretch on your AWS account. It also assumes familiarity with shell commands like ssh, scp and Debian package management commands.

Note: For the shell commands, command line arguments should be replaced with appropriate items specific to your environment. For example, in the command *"cd <project_directory>"*, replace *<project_directory>* with the name of your project directory.

Setting up the server

The server setup consists of installing the Apache Webserver and .NET core framework.

Install Apache webserver

1. Log onto your server instance using ssh:
   ```
   $ ssh –i <server_private_key.pem> admin@<server_IP_Address>
   ```

2. Install Apache2 package:
   ```
   $ sudo apt-get update
   $ sudo apt-get install apache2
   ```

Configure Apache to work as reverse proxy

1. Enable modproxy
   ```
   $ a2enmod proxy proxy_http proxy_html
   ```

2. Create apache configuration file for reverse proxy
   ```
   $ sudo vim /etc/apache2/conf-enabled/aspnetcoredemo.conf
   ```

3. Copy following text into the apache reverse proxy configuration file
   ```
   <VirtualHost *:80>
   ```

```
ProxyPreserveHost On
ProxyPass / http://127.0.0.1:5000/
ProxyPassReverse / http://127.0.0.1:5000/
ErrorLog /var/log/apache2/aspnetcoredemo-error.log
CustomLog /var/log/apache2/aspnetcodedemo-access.log common
</VirtualHost>
```

4. Restart apache to load the configuration file
   ```
   $ sudo /etc/ini.d/apache2 restart
   ```

<u>Install .NET Core framework 2.0</u>

1. Install dependencies:
   ```
   $ sudo apt-get install curl libunwind8 gettext apt-transport-https
   ```

2. Register the trusted Microsoft Product key:
   ```
   $ curl https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > microsoft.gpg
   $ sudo mv microsoft.gpg /etc/apt/trusted.gpg.d/microsoft.gpg
   ```

3. Install .NET core framework
   ```
   $ sudo sh -c 'echo "deb [arch=amd64] https://packages.microsoft.com/repos/microsoft-
   debian-stretch-prod stretch main" > /etc/apt/sources.list.d/dotnetdev.list'
   $ sudo apt-get update
   $ sudo apt-get install dotnet-sdk-2.0.0
   ```

Transferring website project to the server

Once the server setup is done, transfer your project to the server. You can transfer the project folder using *scp* command on your local development model as follows:

```
$ scp –i <server_private_key.pem> -r <path_of_project_file>
admin@<server_IP_Address>:<destination_path_on_server>
```

The above command will transfer your project files to the destination folder on the server

Building and configuring the project

After transferring your project file to the server machine, ssh into the server and execute following steps:

1. Change directory to your project directory
   ```
   $ cd <project_directory>
   ```

2. Build and publish the project
   ```
   $ dotnet publish
   ```

3. The above command will build and publish ASP.NET web files under the directory *bin/Debug/netcoreapp2.0/publish* in your project folder.

4. Copy the publish folder to apache root document directory. In Debian, it is /var/www

```
$ sudo cp bin/Debug/netcoreapp2.0/publish /var/www/
```

5. If you want, you can rename the publish folder under /var/www to something appropriate.
   ```
   $ sudo mv /var/www/publish /var/www/<name>
   ```

6. Kestrel is the ASP.NET core webserver. While, Apache is not set up to manage the Kestrel process. We will use systemd and create a service file to start and monitor the underlying web app. systemd is an init system that provides many powerful features for starting, stopping and managing processes. Create a system configuration file using the following command:
   ```
   $ sudo vim /etc/systemd/system/kestrel-aspnetcoredemo.service
   ```

7. Copy following into the configuration file. The configuration details should be changed according to your own settings:
   ```
   [Unit]
   Description=Example ASP .NET Web Application
   [Service]
   WorkingDirectory=/var/www/project
   ExecStart=/usr/bin/dotnet /var/www/project/projectname.dll
   Restart=always
   RestartSec=10
   SyslogIdentifier=dotnet-demo
   User=www-data
   Environment=ASPNETCORE_ENVIRONMENT=Production
   [Install]
   WantedBy=multi-user.target
   ```

8. Enable and start the systemd service configuration created above:
   ```
   $ sudo systemctl enable kestrel-aspnetcoredemo.service
   $ sudo systemctl start kestrel-aspnetcoredemo.service
   ```

Once all the configurations are completed and services started without error, you can access your website or web application from the browser by entering your domain name or the Public IP address of server.

Reference

1. Microsoft .NET support website: https://www.microsoft.com/net/core#linuxdebian
2. https://blog.todotnet.com/2017/07/publishing-and-running-your-asp-net-core-project-on-linux/