**HRM CODE TUTORIAL COLLECTION**

**Collection By:** R&D Team

Contents

[IIS Documentation 2](#_Toc62722921)

[Introduction to IIS Architectures 2](#_Toc62722922)

[Introduction 2](#_Toc62722923)

[Component trong IIS 2](#_Toc62722924)

[Protocal Listeners 2](#_Toc62722925)

[Hypertext Transfer Protocol Stack (HTTP.sys) 2](#_Toc62722926)

[World Wide Web Publishing Service (WWW Service) 3](#_Toc62722927)

[How WWW Service works in IIS 6.0 3](#_Toc62722928)

[How the WWW Service works in IIS 3](#_Toc62722929)

[Windows Process Activation Service (WAS) 4](#_Toc62722930)

[Modules in IIS 4](#_Toc62722931)

[Request Processing in IIS 6](#_Toc62722932)

[Application Pools in IIS 6](#_Toc62722933)

[HTTP Request Procesing in IIS 6](#_Toc62722934)

[Host and deploy ASP.NET Core 8](#_Toc62722935)

[Host ASP.NET Core trên Windows với IIS 8](#_Toc62722936)

[Overview 8](#_Toc62722937)

[Publish một ASP.NET Core app lên IIS 9](#_Toc62722938)

[In-processs Hosting với IIS và ASP.NET Core 15](#_Toc62722939)

[Out-of-process Hosting 15](#_Toc62722940)

[Hosting Bundle 15](#_Toc62722941)

[web.config file 15](#_Toc62722942)

# IIS Documentation

## Introduction to IIS Architectures

### Introduction

Internet Information Service (IIS) 7 và các phiên bản mới hơn cung cấp Request-Processing Architecture bao gồm

Windows Process Activation Service (WAS), nó cho phép Website sử dụng Protocols ngoài HTTP và HTTPS

Một Web Server Engine có thể Customize bằng cách thêm hoặc xóa các Module.

Tích hợp Request-Processing Pipelines từ IIS và ASP.NET

### Component trong IIS

IIS bao gồm vài Components có trách nhiệm thực thi các chức năng quan trọng cho Application và Web Server Roles trong Windows Server 2008 (IIS 7) và Windows Server 2008 R2 (IIS 7.5). Mỗi Component có các trách nhiệm riêng, như lắng nghe Request được tạo bởi Server, quản lý Proceses, và đọc files cấu hình. Những Components bao gồm Protocol Listeners, như HTTP.sys, và Services như World Wide Web Publishing Service (WWW Service) và Window Process Activation Service (WAS).

### Protocal Listeners

Protocal Listeners nhận các Protocol-Specific Requests, sau đó gửi chúng cho IIS xử lý, và trả về Responses cho Requestors. Ví dụ, khi một Client Browser gửi một Request tới Web Page từ Internet, HTTP Listener, HTTP.sys, nhận Request và tửi chúng cho IIS xử lý. Một khi IIS xử lý Request, HTTP.sys trả về Response cho Client Browser.

Mặc định, IIS cung cấp HTTP.sys như một giao thức Listener để lắng nghe HTTP và HTTPS Request. HTTP.sys được giới thiệu tại phiên bản IIS 6.0 như một HTTP-Specific Protocol Listener cho HTTP Request. HTTP.sys giữa lại HTTP listener trong IIS 7 và các phiên bản sau này, nhưng bao gồm hỗ trợ cho Secure Sockers Layers (SSL).

Để hỗ trợ Services và Application sử dụng Protocols ngoài HTTP và HTTPS, chúng ta có thể sử dụng các công nghệ như Windows Communication Foundation (WCF), WCF có Listener Adaper cung cấp chức năng của cả Protocol Listener và một Listener Adaper

### Hypertext Transfer Protocol Stack (HTTP.sys)

HTTP Listener là một bộ phận của mạng lưới Subsystem của Windows OS, và nó được triển khai như một Kernel-mode Device Driver được gọi là HTTP Stack (HTTP.sys). HTTP.sys lắng nghe HTTP Requests từ Network, sau đó truyền tới IIS để xử lý. Và cuối cùng trả về Response cho Client Browser.

IIS 6.0, HTTP.sys thay thế Windows Sockets API (Winsock), thứ trước đây là một User-mode Component sử dụng bởi các phiên bản cũ để nhận HTTP Requests và gửi HTTP Response. IIS 7 và phiên bản mới hơn tiếp tục dựa trên HTTP.sys cho HTTP Requests.

HTTP.sys cung cấp các lợi ích

* Kernel-mode caching. Request cho Cached Responses được xử lý mà không phải chuyển qua User-mode
* Kernel-mode sử dụng hàng đợi Request. Request phát sinh ít chi phí hơn trong chuyển đổi Context vì Kernel chuyển Request trực tiếp tới Worker Process. Nếu không có Worker Processs nào khả dụng để nhận Request, Kernel-mode Request sẽ được đưa vào hàng đợi cho tới khi Worker Process tiếp nhận chúng.
* Tiền xử lý Request và Security Filtering

### World Wide Web Publishing Service (WWW Service)

Trong IIS 7 và phiên bản sau này, chức năng trước đây dùng để Handle WWW Service độc lập bây giờ được chia ra thành hai Services: WWW Service và một Service mới, Windows Process Activation Service (WAS). Cả hai Service cùng chạy LocalSystem trong cùng một Svchost.exe Process, và chia sẻ Binaries

### How WWW Service works in IIS 6.0

Trong IIS 6.0, WWW Service quản lý những vấn đề chính trong IIS

* HTTP Administration and Configuration
* Process Management
* Performance Monitoring

#### HTTP Administration and Configuration

WWW Service đọc thông tin cấu hình từ IIS Metabase và sử dụng những thông tin đó để cấu hình còn Update HTTP Listener, HTTP.sys. Thêm nữa, WWW service Start, Stop, Monitor và Manage Worker Process thực thi HTTP Requests

#### Performance Monitoring

WWW Service giám sát vấn đề hiệu suất và cung cấp bộ đếm hiệu suất cho Web sites and cho IIS cache

#### Process Management

WWW Service quản lý Application Pool và Worker Processes, như Starting, Stoping, và Recycling Worker Process. Thêm nữa, WWW Service giám sát sức khỏe của Worker Process, và gọi lên nhanh chóng các lỗi được phát hiện để dừng các Processes mới từ khi bắt đầu khi một vài Worker Processes thất bại một số lần nhất định được cấu hình trước đó.

### How the WWW Service works in IIS

Trong IIS, WWW Service không còn quản lý Worker Processes. Thay vào đó WWW Service là Listener Adaper cho HTTP Listener, HTTP.sys. Như là một Listener Adaper, WWW Service chịu trách nhiệm cơ bản cho việc cấu hình HTTP.sys, cập nhật HTTP.sys khi cấu hình bị thay đổi, và thông báo cho WAS khi một Request đi vào Request Queue.

Thêm nữa, WWW Service tiếp tục thu thập những bộ đếm cho Web sites. Bởi vì Performance Counters giữ lại một phần của WWW Service, chúng là HTTP-Specific và không apply vào WAS

### Windows Process Activation Service (WAS)

Trong IIS 7 và các phiên bản sau này, WAS quản lý cấu hình Application Pool và Worker Process thay vì WWW Service. Điều này cho phép chúng ta sử dụng cùng một cấu hình và Process Model cho HTTP và non-HTTP sites.

Thêm nữa, chúng ta có thể chạy WAS mà không cần WWW Service nếu chúng ta không cần chức năng HTTP. Ví dụ, chúng ta có thê quản lý một Web Service thông qua WCF Listener Adapter, như NetTcpActivator mà không cần chạy WWW Service nếu chúng ta không cần lắng nghe HTTP Request tại HTTP.sys.

#### Configuration Management in WAS

Tại thời điểm bắt đầu, WAS đọc những thông tin cụ thể từ ApplicationHost.config file, và tuyền những thông tin đó tới Listener Adapter tại Server. Listener Adapters là những Components thực hiện kết nối giữa WAS và Protocol Listener, như HTTP.sys. Một khi Listener Adapters nhận được thông tin cấu hình, chúng sẽ cấu hình những Protocol Listener liên quan và chuẩn bị Listeners để lắng nghe Requests.

Trong trường hợp WAS, một Listener Adapter bao gồm chức năng của một Protocol Listener. Vì vậy, một WCF Listener Adapter, như NetTcpActivator sẽ được cấu hình dựa trên thông tin từ WAS, một khi NetTcpActivator được cấu hình, nó sẽ lắng nghe Request sử dụng net.tcp Protocol.

Danh sách dưới đây mô tả những định dạng thông tin WAS sẽ đọc từ cấu hình

* Thông tin Gloabal Configuration
* Thông tin Protocol Configuration cho cả HTTP và non-HTTP Protocols
* Application Pool Configuration, như là Process Account Information
* Site Configuration, như Bidings và Applications
* Application Configuration, như Enabled Protocol và Application Pools mà Application thuộc về

Nếu ApplicationHost.config thay đổi, WAS sẽ nhận một thông báo và cập nhật Listener Adapter với những thông tin mới này

#### Process Management

WAS quản lý Application Pool và Worker Processes cho cả HTTP và non-HTTP Request. Khi một Protocol Listener tiếp nhận một Client Request, WAS sẽ xác định nếu có một Worker Process nào đang chạy hay không. Nếu một Application Pool đã có một Worker Process đang xử lý Request, Listener Adapter sẽ truyền Request cho Worker Process để xử lý. Nếu không có một Worker Process nào trong Application Pool, WAS sẽ Start một Worker Process để Listener Adapter có thể truyền Request cho nó để tiến hành xử lý.

##### Note

Bởi vì WAS quản lý Processes cho cả HTTP và non-HTTP Protocols nên chúng ta có thể chạy Application với các Protocols khác nhau trong cùng một Application Pool. Ví dụ, chúng ta có thể phát triển một Application, như một XML Service, và Host cả HTTP và net.tcp

### Modules in IIS

IIS cung cấp một kiến trúc mới khác biệt với các phiên bản IIS trước đây. Thay vì giữ những chức năng chính trong IIS Server, giờ đây IIS sẽ bao gồm một Web Server Engine, bên trong Web Server Engine này chúng ta có thể thêm hoặc xóa các Component, gợi Modules dựa theo từng nhu cầu cụ thể.

Modules là một thuộc tính riêng biệt mà Server sử dụng để xử lý Request. Ví dụ, IIS sử dụng Authentication Modules để xác thực Client Credentials, và Cache Modules sẽ quản lý các hoạt động caching.

Kiến trúc mới cung cấp các lợi thế sau so với các phiên bản IIS trước

* Chúng ta có thể kiểm soát những Modules nào chúng ta muốn có trên Server.
* Chúng ta có thể Customize một Server theo từng Role cụ thể trong môi trường của mình.
* Chúng ta có thể sử dụng Custom Modules thay thế các Modules có sẵn hoặc tạo ra các Feature mới.

Kiến trúc mới này cũng cải thiện Security và đơn giản hóa công việc quản trị bằng cách loại bỏ đi các Modules không cần thiết .

#### Native Modules

Danh sách các Native Modules đi kèm đầy đủ với các phiên bản cài đặt IIS 7 và các phiên bản sau này. Chúng ta có thể thay thế chúng với các Custom Modules tùy vào nhu cầu

##### HTTP Modules

Một vài Modules trong IIS 7 và sau này thực hiện các tác vụ dành riêng cho Hypertxt Transfer Protocol (HTTP) trong Request-Processing Pipiline. HTTP Modules bao gồm các modules Response cho Client Requests, Response HTTP Errors, Redirect Requests ,v.v…

* CustomErrorModule: Gửi thông báo lỗi HTTP Error mặc định và được cấu hình khi một Error Status được set vào Response
* HttpRedirectionModule: Hỗ trợ cấu hình redirect HTTP Requests
* PrototolSupportModule: Thực thiện các hành động liên quan tới Protocol, như Setting Response Header và Redirecting Header dựa trên cấu hình
* RequestFilteringModule: Được thêm vào tại IIS 7.5, có chức năng Filter Requests như cấu hình tới kiểm soát Protocol và Content Behavior.
* WebDAVModule: Được thêm vào tại IIS 7.5. Cho phép việc Publish Content thêm an toán sử dụng HTTP thông qua SSL

##### Security Modules

Một vài Modules trong IIS thực hiện các tác vụ liên quan đến Request-Processing Pipepline. Thêm nữa, có những Modules tách biệt cho mỗi Authentication Schemes, việc đó cho phép chúng ta lựa chọn Modules cho các kiểu Authentication mong muốn trên Server.

* AnonymousAuthenticationModule
* BasicAuthenticationModule
* CertificateMappingAuthenticationModule
* DigestAuthenticationModule
* RequestFilteringModule
* UrlAuthorizationModule
* WindowsAuthenticationModule
* IpRestrictionModule

##### Content Module

Một vài Modules trong IIS thực hiện tác vụ liên quan đến nội dung trong Request-Processing Pipeline. Content Modules bao gồm các Modules xử lý các Request cho Static Files để trả về Default Page khi Client không chỉ ra Resource trong Request

* CgiModule
* DefaultDocumentModule
* DirectoryListingModule
* IsapiModule
* ServerSideIncludeModule
* StaticFileModule
* FastCgiModule

##### Compression Modules

##### Caching Modules

##### Logging and Diagnostic Modules

##### Managed Support Modules

##### Managed Modules

### Request Processing in IIS

Trong IIS, IIS và ASP.NET Request Pipelines kết hợp với nhau để xử lý Requests với một phương pháp tích hợp. Kiến trúc Request-Processing mới bao gồm một danh sách được sắp xếp của các Native và Managed Modules thực thi các tác vụ cụ thể trong Response của Request.

Kiểu thiết kế này cung cấp một vài lợi ích so với trước đây. Thứ nhất, tất cả các kiểu File có thể sử dụng những tính năng trước đây chỉ khả dụng để quản lý code. Ví dụ, chúng ta có thể sử dụng ASP.NET Forms Authentication và Uniform Resource Locator (URL) Authorization cho Static Files. Active Server Page (ASP) Files, và tất cả cá kiểu Files khác trong Sites hoặc Application của chúng ta.

Thứ hai, kiểu thiết kế này loại bỏ việc trùng lặp một vài thuộc tính của IIS và ASP.NET. Ví dụ, khi một Client Request một Managed File, Server sẽ gọi Authentication Modules thích hợp để tích hợp Pipeline để xác thực Client.

Thứ ba, chúng ta có thể quản lý tất cả Modules tỏng một nơi duy nhất, thay vì phải quản lý một vài tính năng của IIS, hoặc một và nơi khác tại ASP.NET Configuration. Điều này đơn giản hóa việc quản trị các Sites và Application trên Server.

### Application Pools in IIS

Application Pools tách biệt Application bằng Process Boundaries để tránh tình trạng các Application bị ảnh hưởng bởi các Application khác trên Server. Đối với IIS 7 và các phiên bản mới hơn, Application Pools tiếp tục sử dụng IIS 6.0 Worker Process Isolation Mode. Và hơn thế, chúng ta có thể chỉ định một Setting trong đó chỉ ra bằng cách nào để xử lý Requests liên quan tới Managed Resource: Intergrated Mode và Classic Mode

#### Intergrated Application Pool mode

Khi một Application Pool trong Intergrated Mode, chúng ta có thể sử dụng các lợi thế của kiến trúc Intergrated Request-Processing của IIS và ASP.NET. Khi một Worker Process trong một Application Pool nhận được một Request, Request sẽ được chuyển thông qua một danh sách được sắp xếp các sự kiện. Mỗi sự kiện gọi những Native và Managed Modules cần thiết để xử lý từng phần của Request và tạo ra Response.

#### Classic Application Pool mode

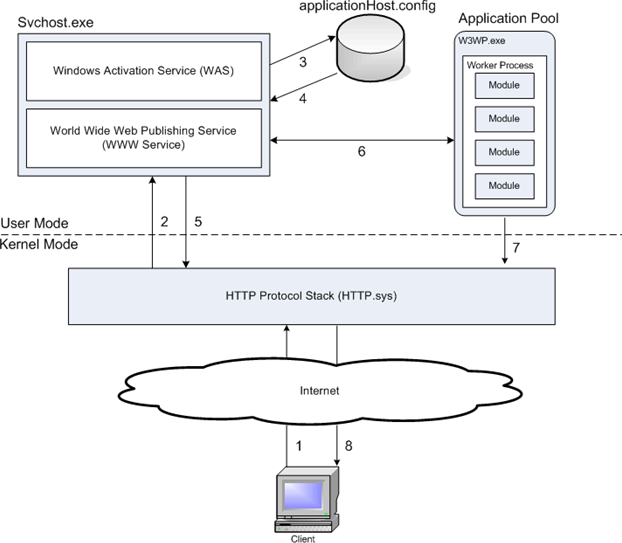
…

### HTTP Request Procesing in IIS

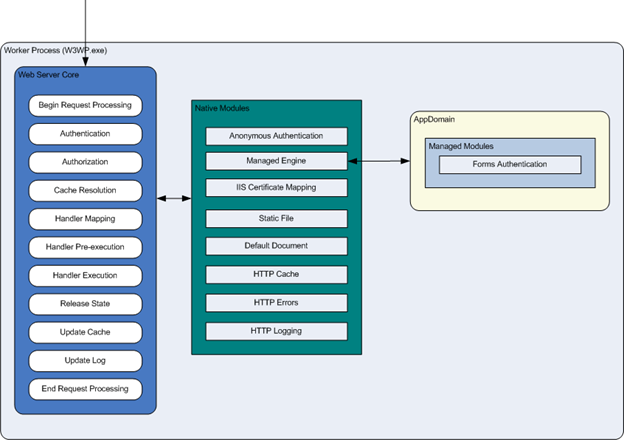
IIS 7 và các phiên bản sau này có chugn HTTP Request Processing với IIS 6.0. Biểu đồ trong phần này sẽ cung cấp một cái nhìn tổng thể về một HTTP Request được xử lý ra sao.

#### Mô tả

1. Khi một Client Browser bắt đầu thực hiện một HTTP Request để lấy Resource từ Web Server, HTTP.sys can thiệp vào Request.
2. HTTP.sys tương tác với WAS để nhận các thông tin cấu hình từ Configuration Store.
3. WAS Requests thông tin cấu hình từ Configuration Store, applicationHost.config.
4. WWW Service nhận thông tin cấu hình, ví dụ như các thông tin về Application Pool và Site Configuration
5. WWW Service sử dụng thông tin cấu hình này để cấu hình HTTP.sys.
6. WAS khởi động một Worker Process cho Application Pool mà Request được tạo ra.
7. Worker Process xử lý Request và trả về Response cho HTTP.sys.
8. Client nhận Response



Trong một Worker Process, một HTTP Request phải đi qua một vài bước xác định được gọi là Events, trong Web Server Core. Tại một Event, một Native Module xử lý một phần của Request, như là chứng thực người dùng hoặc thêm thông tin vào Event Log. Nếu một Request đòi hỏi Managed Module, Native ManagedEngine Module sẽ tạo ra một AppDomain, nơi mà Managed Module có thể xử lý những công việc cần thiết như chứng thực người dùng với Forms Authentication. Khi Request đi qua tất cả Events tại Web Server Core, Response được trả về cho HTTP.sys



# Host and deploy ASP.NET Core

## Host ASP.NET Core trên Windows với IIS

### Overview

#### Support Platform

Hỗ trợ các OS sau

* Windows 7 trở lên
* Windows Server 2012 R2 hoặc mới hơn

Apps Published cho 32-bit (x86) hoặc 64bit (x64) Deployment được hộ trợ. Deploy một 32-bit app với 32-bit (x86) .NET Core SDK ngoại trừ các apps

Đòi hỏi khổi lượng địa chỉ bộ nhớ ảo lớn hơn cho một 64-bit app.

Đòi hỏi size lớn hơn ISS Stack Size

Có 64-bit Native Dependencies

#### Install ASP.NET Core Module/Hosting Bundle

<https://dotnet.microsoft.com/download/dotnet-core/thank-you/runtime-aspnetcore-5.0.1-windows-hosting-bundle-installer>

### Publish một ASP.NET Core app lên IIS

#### Prerequisites

Cần cài đặt .NET Core SDK trong Development Machine.

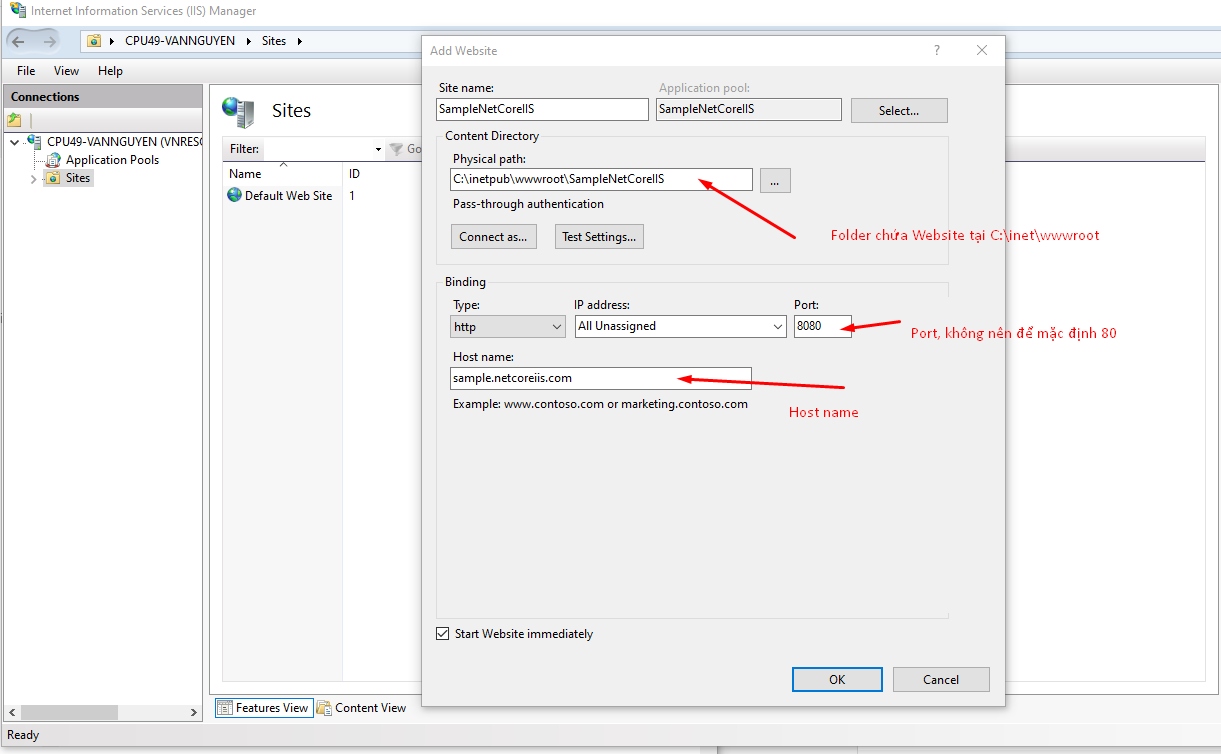
Windows Server được cấu hình với Web Server (IIS) Server Role

#### Cài đặt .NET Core Hosting Bundle

Cài đặt .NET Core Hosting Bundle cho IIS Server. .NET Core Hosting Bundle cài đặt .NET Core Runtime, .NET Core Library và ASP.NET Core Module. Module này cho phép ASP.NET Core Apps chạy trên IIS.

#### Tạo IIS Site

1. Truy cập IIS Server, tạo một Folder để chứa App’s Published Folders và Files theo như trong hình phía dưới.



#### Publish và Deploy ASP.NET Core App

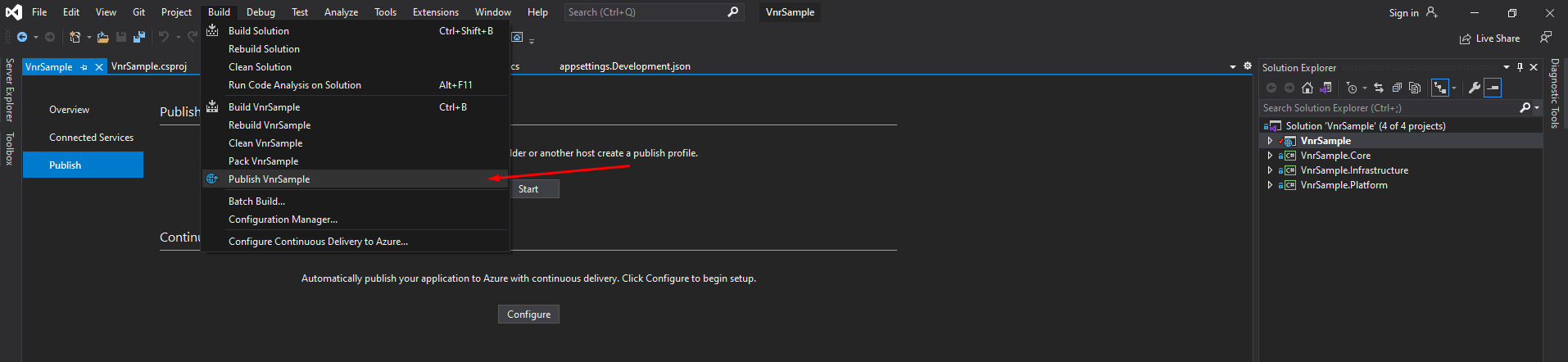
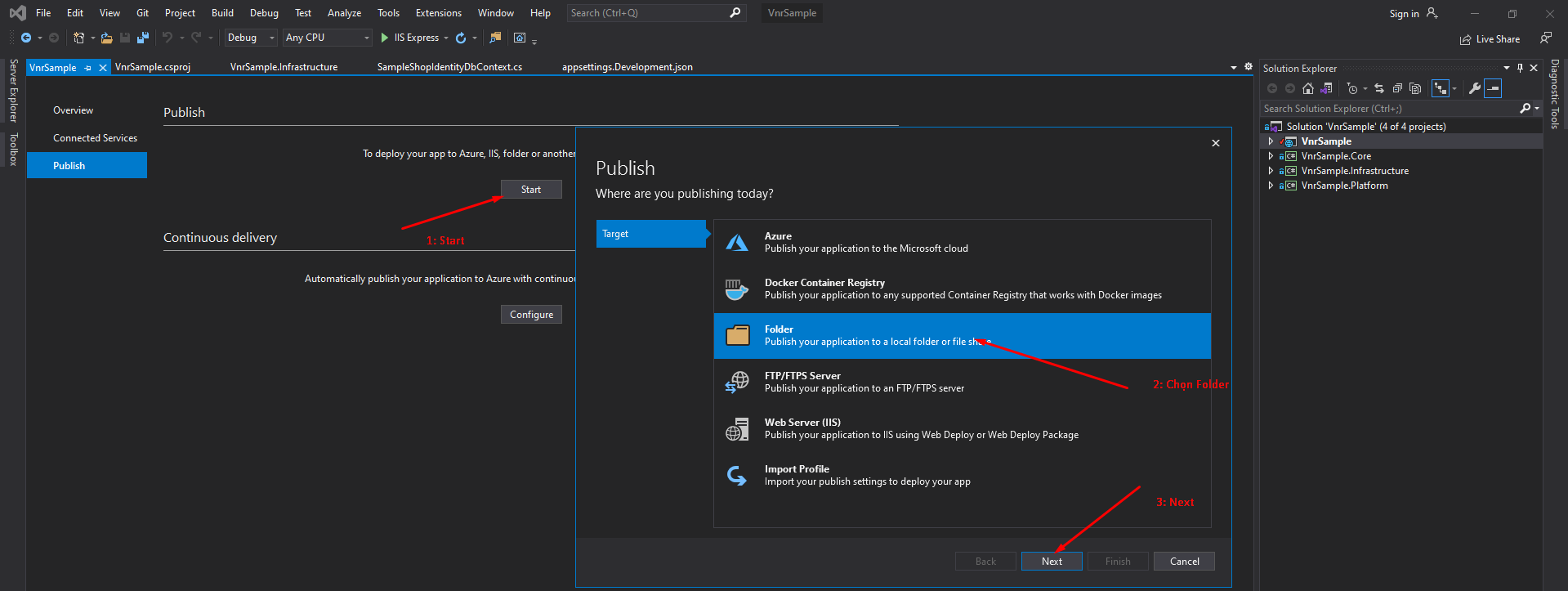
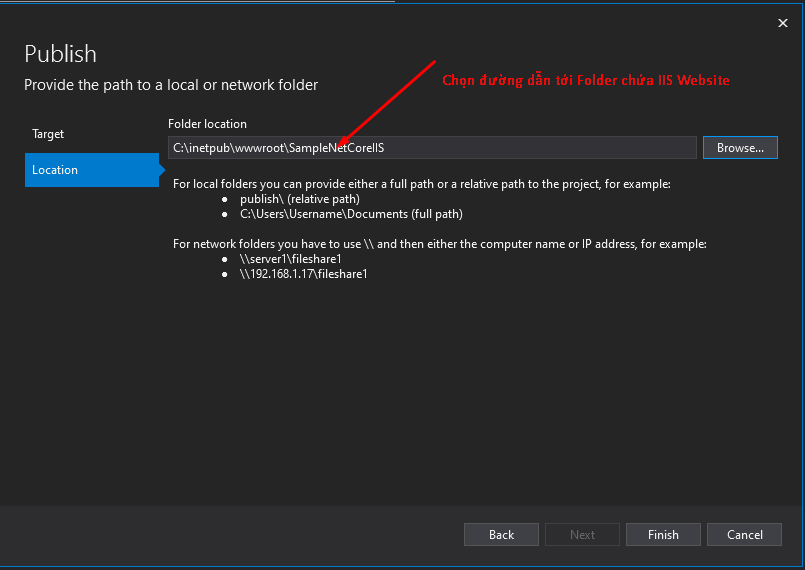
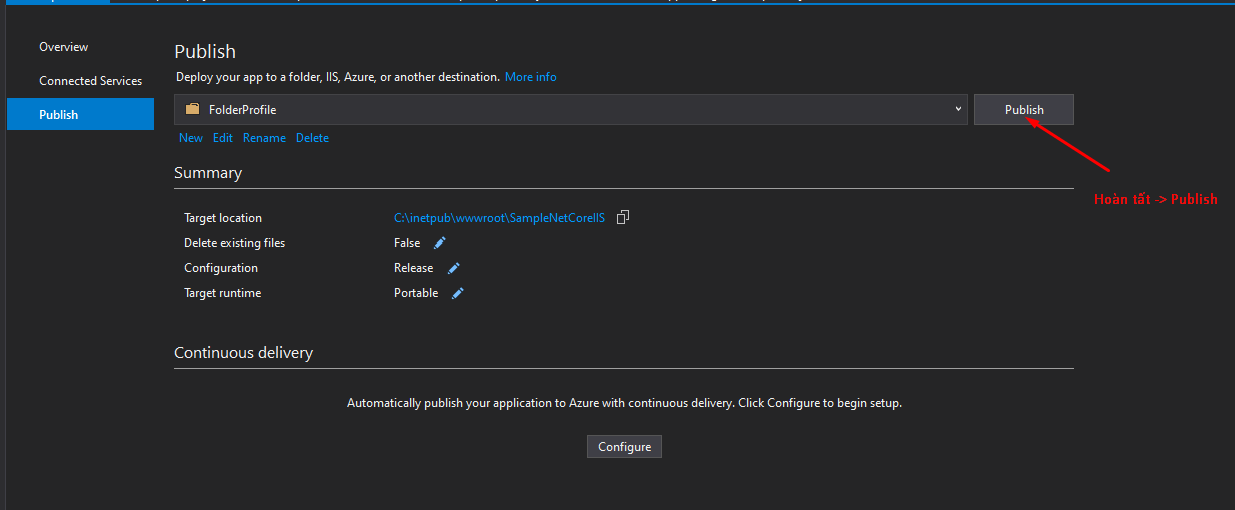
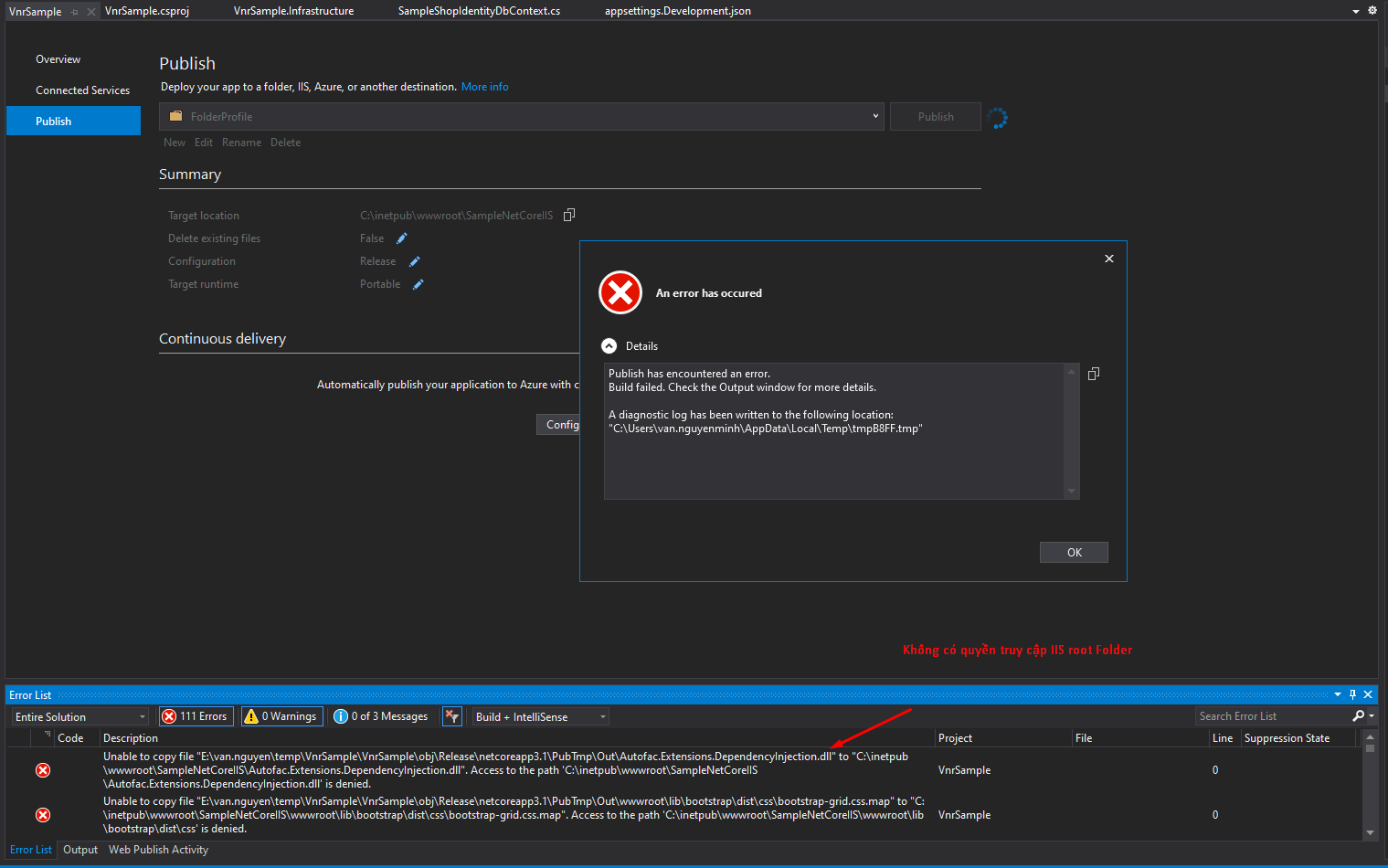
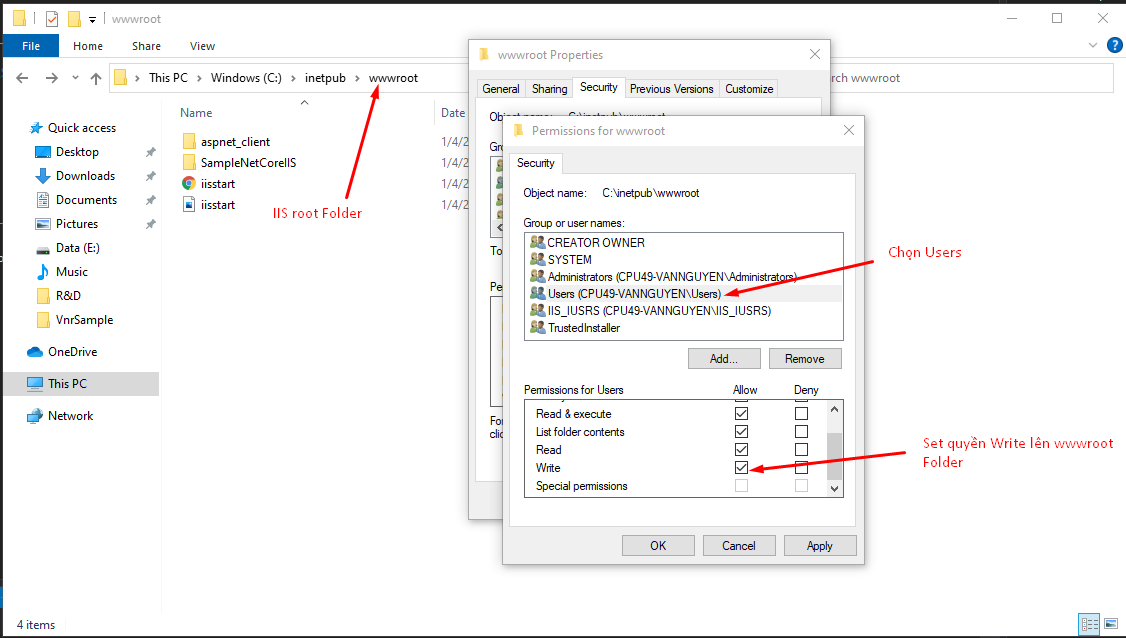
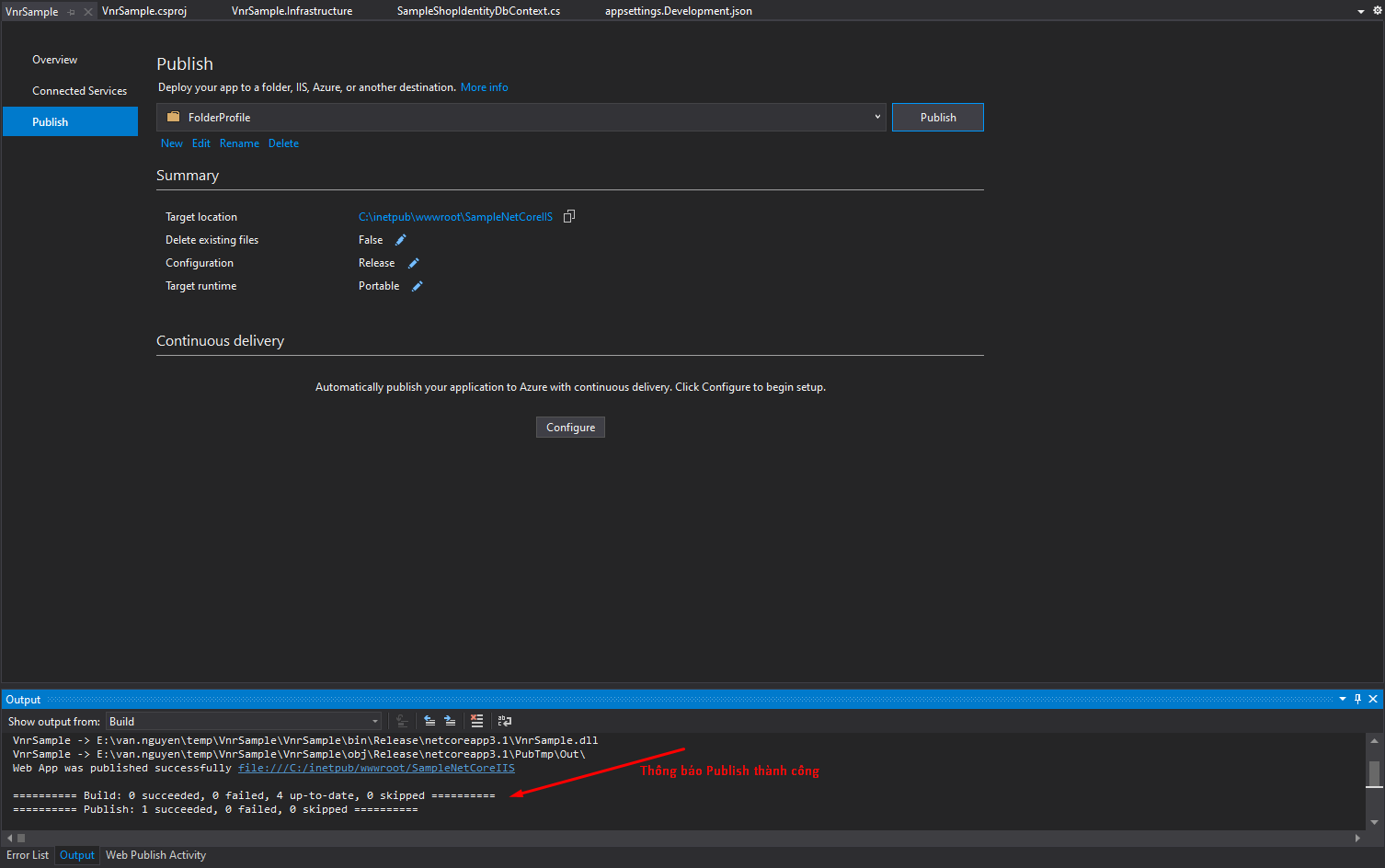
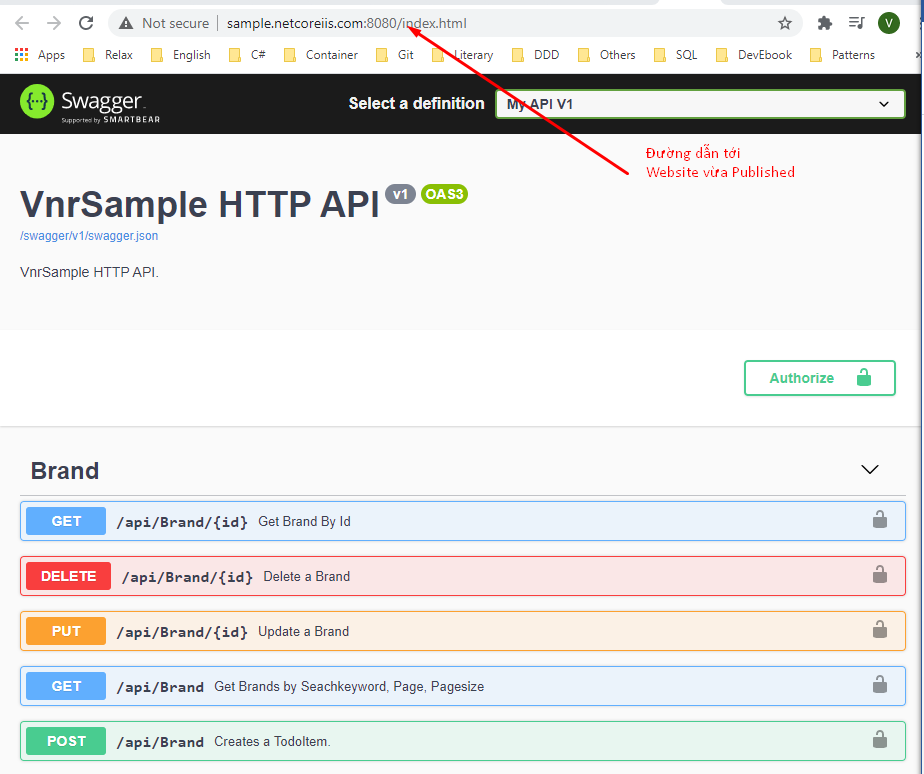
Publish App nghĩa là tạo ra một Compilied App có thể được Hosted bởi một Server. Deploy App nghĩa là chuyển Published App tới một Hosting System.

Quá trình Publish được Handled bởi .NET Core SDK, trong khi Deployment được Handled bởi nhiều phương pháp khác nhau.

Dưới đây chúng ta sẽ sử dụng phương pháp Folder Deployment, nơi mà

* App sẽ được Published tới một Folder
* Folder’s Contents được di chuyển tới IIS Site’s Folder (Physical Path ở hình phía trên)

##### Sử dụng Visual Studio

1. Tạo một ASP.NET Core bất kỳ, trong ví dụ ở đây sử dụng một ASP.NET Core API thông thường có thể chạy được
2. Nhấn chuột phải vào Solution Explorer chọn Publish, hoặc Build -> Publish  
   
3. Chọn Publish Folder  
   
4. Thay đổi đường dẫn tới Folder chứa IIS Site  
   
5. Publish Website  
   
6. Lỗi có thể phát sinh  
   
7. Khắc phục lỗi  
   
8. Thực hiện lại việc Publish  
   
9. Truy cập Website  
   

### In-processs Hosting với IIS và ASP.NET Core

### Out-of-process Hosting

### Hosting Bundle

### web.config file

web.config là một file được đọc bởi IIS và ASP.NET Core Module để cấu hình một App với IIS

#### web.config file location

Để có thể Set Up ASP.NET Core Module một cách chính xác, web.config phải được đặt tại Content Root Path (thông thường là Base Path của App) của Deployed App. Và điều này giúp nó có cùng Location với Website Physical Path cung cấp bởi IIS. web.config file được đặt tại Root của App cho phép Publish Multiple Apps sử dụng Web Deploy.

Các Sensitive Files xuất hiện tại Physical Path của App, như là {ASSEMBLY}.runtimeconfig.json, {ASSEMBLY}.xml (XML Documentation Comments) và {ASSEMBLY}.deps.json, và Placeholder {ASSEMBLY} là tên của Assembly.  
Khi web.config file xuất hiện và Site chạy bình thường, IIS không quan tâm tới các Sensitive Files trên nếu chúng có Request. Nếu web.config không có, hoặc sai tên, hoặc không thể cấu hình để Site có thể chạy bình thường, IIS khi này sẽ xử lý các Sensitive Files.

**web.config file phải xuất hiện tại Deployment tất cả các thời điểm, tên được đặt chính xác, và có thể cấu hình cho Site Start Up một cách bình thường. Không bao giờ được xóa web.config file từ một Production Deployment.**

Nếu một web.config file không xuất hiện trong Project, file này sẽ được tạo với processPath và arguments chính xác để cấu hình ASP.NET Core Module và chuyển tới published output

Nếu web.config file xuất hiện trong Project, file này sẽ được Transform với processPath và arguments chính xác để cấu hình ASP.NET Core Module và chuyển tới publish output

web.config file có thể cung cấp thêm các cấu hình cài đặt IIS có thể điều khiển các Active IIS Modules.

Create, Transforming, và Publishing web.config file được Handled bởi MSBuild Target (\_TransformWebConfig) trong khi Project được Published. Target này xuất hiện trong Web SDK Targets (Microsoft.NET.Sdk.Web). SDK được đặt trên Top của Project File:

XML

<Project Sdk="Microsoft.NET.Sdk.Web">

Để ngăn chặn Web SDK Transforming web.config file, sử dụng <IsTransformWebConfigDisabled> property trong Project file:

XML

<PropertyGroup>

<IsTransformWebConfigDisabled>true</IsTransformWebConfigDisabled>

</PropertyGroup>

Khi Disable việc Web SDK Tranforming web.config file, processPath và arguments nên được cài đặt tay bởi Developer

#### Cấu hình ASP.NET Core Module với web.config

ASP.NET Core Module được cấu hình với aspNetCore Section của system.webServer Node trên web.config file

web.config file dưới đây được Published cho một framework-dependent deployement và cấu hình ASP.NET Core Module để Handle Site Requests

XML

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<location path="." inheritInChildApplications="false">

<system.webServer>

<handlers>

<add name="aspNetCore" path="\*" verb="\*" modules="AspNetCoreModuleV2" resourceType="Unspecified" />

</handlers>

<aspNetCore processPath="dotnet"

arguments=".\MyApp.dll"

stdoutLogEnabled="false"

stdoutLogFile=".\logs\stdout"

hostingModel="inprocess" />

</system.webServer>

</location>

</configuration>

web.config được Published cho self-contained deployment

XML

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<location path="." inheritInChildApplications="false">

<system.webServer>

<handlers>

<add name="aspNetCore" path="\*" verb="\*" modules="AspNetCoreModuleV2" resourceType="Unspecified" />

</handlers>

<aspNetCore processPath=".\MyApp.exe"

stdoutLogEnabled="false"

stdoutLogFile=".\logs\stdout"

hostingModel="inprocess" />

</system.webServer>

</location>

</configuration>

…

#### Set environment variables

Environment Variables có thể được chỉ định cho Process trong processPath Attribute.

Chỉ định một Environment Variable với <environmentVariable> Child Element của một <environmentVariables> Collection Element. Environment Variables được Set trong Section này được ưu tiên hơn các Environment Variable của System

Ví dụ sau mô tả việc Set hai Environment Variables tại web.config. ASPNETCORE\_ENVIRONMENT cấu hình App’s Environment thành Development. Một Developer có thể tmj thời Set giá trị này tại web.config file để có thể Load Developer Exception Page trong quá trình Debug. CONFIG\_DIR là một ví dụ của User-defined Environment Variable, nơi mà Developer viết Code để đọc giá trị tại Startup tới hình thành một Path để Load Configuration File của App.

XML

<aspNetCore processPath="dotnet"

arguments=".\MyApp.dll"

stdoutLogEnabled="false"

stdoutLogFile=".\logs\stdout"

hostingModel="inprocess">

<environmentVariables>

<environmentVariable name="ASPNETCORE\_ENVIRONMENT" value="Development" />

<environmentVariable name="CONFIG\_DIR" value="f:\application\_config" />

</environmentVariables>

</aspNetCore>