Размещение ASP.NET Core в Windows со службами IIS

01.11.2018 • 41 minutes to read • Edit Online

Автор Люк Латэм (Luke Latham)

Установка пакета размещения .NET Core

Поддерживаемые операционные системы

Поддерживаются следующие операционные системы:

- Windows 7 и более поздние версии
- Windows Server 2008 R2 и более поздние версии

Сервер HTTP.sys (ранее назывался WebListener) не работает в конфигурации обратного прокси-сервера со службами IIS. Используйте сервер Kestrel.

Сведения о размещении в Azure см. в статье Развертывание приложений ASP.NET Core в Службе приложений Azure.

Поддержка HTTP/2

HTTP/2 поддерживается в ASP.NET Core для следующих сценариев развертывания IIS:

- Внутрипроцессно
  - Windows Server 2016 / Windows 10 или более поздних версий; IIS 10 или более поздней версии
  - Требуемая версия .NET Framework: .NET Core версии 2.2 или более поздней
  - Подключение TLS 1.2 или более поздней версии
- Внепроцессно
  - Windows Server 2016 / Windows 10 или более поздних версий; IIS 10 или более поздней версии
  - Подключения к пограничным серверам, открытых для общего доступа, выполняются по протоколу HTTP/2, а подключения к серверу Kestrel через обратный прокси-сервер — по HTTP/1.1.
  - Целевая платформа: неприменимо к развертываниям вне процесса, так как IIS полностью обрабатывает подключение HTTP/2.
  - Подключение TLS 1.2 или более поздней версии

При внутрипроцессном развертывании и установленном подключении HTTP/2 HttpRequest.Protocol возвращает `HTTP/2`. При внепроцессном развертывании и установленном подключении HTTP/2 HttpRequest.Protocol возвращает `HTTP/1.1`.

Дополнительные сведения о моделях размещения внутри и вне процесса см. в статьях Модуль ASP.NET Core и Справочник по конфигурации модуля ASP.NET Core.

HTTP/2 поддерживается для внепроцессных развертываний, которые удовлетворяют следующим базовым требованиям:

- Windows Server 2016 / Windows 10 или более поздних версий; IIS 10 или более поздней версии
- Подключения к пограничным серверам, открытых для общего доступа, выполняются по протоколу HTTP/2, а подключения к серверу Kestrel через обратный прокси-сервер — по HTTP/1.1.

- Целевая платформа: неприменимо к развертываниям вне процесса, так как IIS полностью обрабатывает подключение HTTP/2.
- Подключение TLS 1.2 или более поздней версии

Если установлено подключение HTTP/2, HttpRequest.Protocol возвращает `HTTP/1.1`.

Протокол HTTP/2 по умолчанию включен. Если не удается установить подключение HTTP/2, применяется резервный вариант HTTP/1.1. Дополнительные сведения о настройке HTTP/2 для развертываний IIS см. в статье об HTTP/2 в IIS.

Настройка приложения

Включение компонентов IISIntegration

Модель внутрипроцессного размещения

Обычно *Program.cs* вызывает CreateDefaultBuilder, чтобы начать настройку узла. `CreateDefaultBuilder` вызывает метод `UseIIS` для загрузки CoreCLR и размещения приложения внутри рабочего процесса IIS (`w3wp.exe`). Результаты тестов производительности показывают, что размещение приложения .NET Core в процессе позволяет обработать больше запросов по сравнению с размещением приложения вне процесса с перенаправлением запросов к Kestrel.

Модель размещения вне процесса

Обычно *Program.cs* вызывает CreateDefaultBuilder, чтобы начать настройку узла. Для размещения вне процесса с IIS `CreateDefaultBuilder` настраивает Kestrel в качестве веб-сервера и активирует интеграцию с IIS, задавая базовый путь и порт для модуля ASP.NET Core:

```
public static IWebHost BuildWebHost(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
        ...
```

Модуль ASP.NET Core создает динамический порт для назначения серверному процессу. `CreateDefaultBuilder` вызывает метод UseIISIntegration. `UseIISIntegration` настраивает Kestrel для прослушивания динамического порта по IP-адресу localhost (`127.0.0.1`). Если динамический порт — 1234, Kestrel прослушивает `127.0.0.1:1234`. Эта конфигурация заменяет другие конфигурации URL-адресов, предоставляемые:

- `UseUrls`
- API прослушивания Kestrel;
- Конфигурацией (или параметром командной строки--urls).

Вызовы `UseUrls` или API `Listen` Kestrel при работе с этим модулем не требуются. При вызове `UseUrls` или `Listen` Kestrel ожидает передачи данных на порты, указанные только при выполнении приложения без IIS.

Дополнительные сведения о моделях размещения внутри и вне процесса см. в статьях Модуль ASP.NET Core и Справочник по конфигурации модуля ASP.NET Core.

Обычно *Program.cs* вызывает CreateDefaultBuilder, чтобы начать настройку узла. `CreateDefaultBuilder` настраивает Kestrel в качестве веб-сервера и активирует интеграцию IIS, задавая базовый путь и порт для модуля ASP.NET Core:

```
public static IWebHost BuildWebHost(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
        ...
```

Модуль ASP.NET Core создает динамический порт для назначения серверному процессу. `CreateDefaultBuilder` вызывает метод UseIISIntegration. `UseIISIntegration` настраивает Kestrel для прослушивания динамического порта по IP-адресу localhost (`127.0.0.1`). Если динамический порт — 1234, Kestrel прослушивает `127.0.0.1:1234`. Эта конфигурация заменяет другие конфигурации URL-адресов, предоставляемые:

- `UseUrls`
- API прослушивания Kestrel;
- Конфигурацией (или параметром командной строки--urls).

Вызовы `UseUrls` или API `Listen` Kestrel при работе с этим модулем не требуются. При вызове `UseUrls` или `Listen` Kestrel ожидает передачи данных на порт, указанный только при выполнении приложения без IIS.

Обычно *Program.cs* вызывает CreateDefaultBuilder, чтобы начать настройку узла. `CreateDefaultBuilder` настраивает Kestrel в качестве веб-сервера и активирует интеграцию IIS, задавая базовый путь и порт для модуля ASP.NET Core:

```
public static IWebHost BuildWebHost(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
        ...
```

Модуль ASP.NET Core создает динамический порт для назначения серверному процессу. `CreateDefaultBuilder` вызывает метод UseIISIntegration. `UseIISIntegration` настраивает Kestrel для прослушивания динамического порта по IP-адресу localhost (`localhost`). Если динамический порт — 1234, Kestrel прослушивает `localhost:1234`. Эта конфигурация заменяет другие конфигурации URL-адресов, предоставляемые:

- `UseUrls`
- API прослушивания Kestrel;
- Конфигурацией (или параметром командной строки--urls).

Вызовы `UseUrls` или API `Listen` Kestrel при работе с этим модулем не требуются. При вызове `UseUrls` или `Listen` Kestrel ожидает передачи данных на порт, указанный только при выполнении приложения без IIS.

Включите в зависимости приложения зависимость от пакета Microsoft.AspNetCore.Server.IISIntegration. Используйте ПО промежуточного слоя для интеграции IIS, добавив метод расширения UseIISIntegration в WebHostBuilder.

```
var host = new WebHostBuilder()
    .UseKestrel()
    .UseIISIntegration()
    ...
```

Оба метода, UseKestrel и UseIISIntegration, — обязательные. Код, вызывающий `UseIISIntegration`, не влияет на переносимость кода. Если приложение запускается не в IIS (например, запускается непосредственно в Kestrel), `UseIISIntegration` не работает.

Модуль ASP.NET Core создает динамический порт для назначения серверному процессу. `UseIISIntegration` настраивает Kestrel для прослушивания динамического порта по IP-адресу localhost (`localhost`). Если динамический порт — 1234, Kestrel прослушивает `localhost:1234`. Эта конфигурация заменяет другие конфигурации URL-адресов, предоставляемые:

- `UseUrls`

- Конфигурацией (или параметром командной строки--urls).

При использовании этого модуля вызов `UseUrls` не требуется. При вызове `UseUrls` Kestrel ожидает передачи данных на порт, указанный только при выполнении приложения без IIS.

При вызове `UseUrls` в приложении ASP.NET Core 1.0 следует выполнять вызов **до** вызова `UseIISIntegration`, чтобы исключить перезапись порта, настроенного в модуле. В ASP.NET Core 1.1 соблюдать этот порядок вызовов не требуется, так как параметр модуля переопределяет `UseUrls`.

Дополнительные сведения о размещении см. в разделе Размещение в ASP.NET Core.

Параметры служб IIS

Чтобы настроить параметры IIS, включите конфигурацию службы для IISOptions в ConfigureServices. В примере ниже пересылка сертификатов клиентов в приложение с целью заполнения `HttpContext.Connection.ClientCertificate` отключена.

```
services.Configure<IISOptions>(options =>
{
    options.ForwardClientCertificate = false;
});
```

ПАРАМЕТР	ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ	ПАРАМЕТР
`AutomaticAuthentication`	`true`	Если значение — `true`, ПО промежуточного слоя для интеграции IIS задает свойство `HttpContext.User`, которое прошло проверку подлинности Windows. Если значение — `false`, ПО промежуточного слоя только предоставляет идентификатор для `HttpContext.User` и отвечает на явные запросы защиты от `AuthenticationScheme`. Для работы `AutomaticAuthentication` необходимо включить в службах IIS проверку подлинности Windows. Дополнительные сведения см. в статье о проверке подлинности Windows.
`AuthenticationDisplayName`	`null`	Задает отображаемое имя для пользователей на страницах входа.
`ForwardClientCertificate`	`true`	Если значение — `true` и если присутствует заголовок запроса `MS-ASPNETCORE-CLIENTCERT`, происходит заполнение `HttpContext.Connection.ClientCertificate`.

Сценарии использования прокси-сервера и подсистемы балансировки нагрузки

ПО промежуточного слоя для интеграции IIS, которое настраивает ПО промежуточного слоя переадресации заголовков, и модуль ASP.NET Core настраиваются на пересылку схемы (HTTP/HTTPS) и удаленного IP-адреса расположения, где был сформирован запрос. Для приложений, размещенных за дополнительными прокси-серверами и подсистемами балансировки нагрузки, может потребоваться дополнительная настройка. Дополнительные сведения см. в разделе Настройка ASP.NET Core для работы

с прокси-серверами и подсистемами балансировки нагрузки.

Файл web.config

В файле *web.config* содержится конфигурация модуля ASP.NET Core. Создание, преобразование и публикация файла *web.config* обрабатываются целевым объектом MSBuild (`_TransformWebConfig`) при публикации проекта. Этот целевой объект присутствует в целевых веб-пакетах SDK (`Microsoft.NET.Sdk.Web`). Пакет SDK задается в начале файла проекта:

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

Если в проекте нет файла *web.config*, он создается с соответствующими аргументами *processPath* и *arguments* для настройки модуля ASP.NET Core и переносится в опубликованные выходные данные.

Если в проекте есть файл *web.config*, он преобразуется с соответствующими аргументами *processPath* и *arguments* для настройки модуля ASP.NET Core и переносится в опубликованные выходные данные. Преобразование не изменяет параметры конфигурации служб IIS, включенные в файл.

Файл *web.config* может содержать дополнительные параметры конфигурации IIS, управляющие активными модулями IIS. Сведения о модулях IIS, которые могут обрабатывать запросы к приложениям ASP.NET Core, см. в статье Модули IIS.

Чтобы пакет SDK не преобразовывал файл *web.config*, добавьте в файл проекта свойство **<IsTransformWebConfigDisabled>**.

```
<PropertyGroup>
  <IsTransformWebConfigDisabled>true</IsTransformWebConfigDisabled>
</PropertyGroup>
```

Если пакет SDK не преобразует файл, аргументы *processPath* и *arguments* нужно задать вручную. Дополнительные сведения см. в разделе Справочник по конфигурации модуля ASP.NET Core.

Расположение файла web.config

Для создания обратного прокси-сервера между и сервером Kestrel и IIS файл *web.config* должен находиться в корневой папке с содержимым развернутого приложения (как правило, это основной путь приложения). Это расположение соответствует физическому пути веб-сайта, указанному в службах IIS. Файл *web.config* требуется в корне приложения, чтобы разрешить публикацию нескольких приложений с помощью веб-развертывания.

По физическому пути приложения находятся файлы с конфиденциальной информацией: *<имя_сборки>.runtimeconfig.json*, *<имя_сборки>.xml* (комментарии к XML-документации) и *<имя_сборки>.deps.json*. Когда файл *web.config* присутствует и сайт запускается нормально, службы IIS не обрабатывают запросы к этим файлам. Если файл *web.config* отсутствует, неправильно именован или не может настроить нормальный запуск сайта, службы IIS могут свободно передавать содержимое этих конфиденциальных файлов.

Файл *web.config* должен постоянно находиться в развертывании, у этого файла должно быть правильное имя, и файл должен быть в состоянии настроить нормальный запуск сайта. Никогда не удаляйте файл *web.config* из развертывания в рабочей среде.

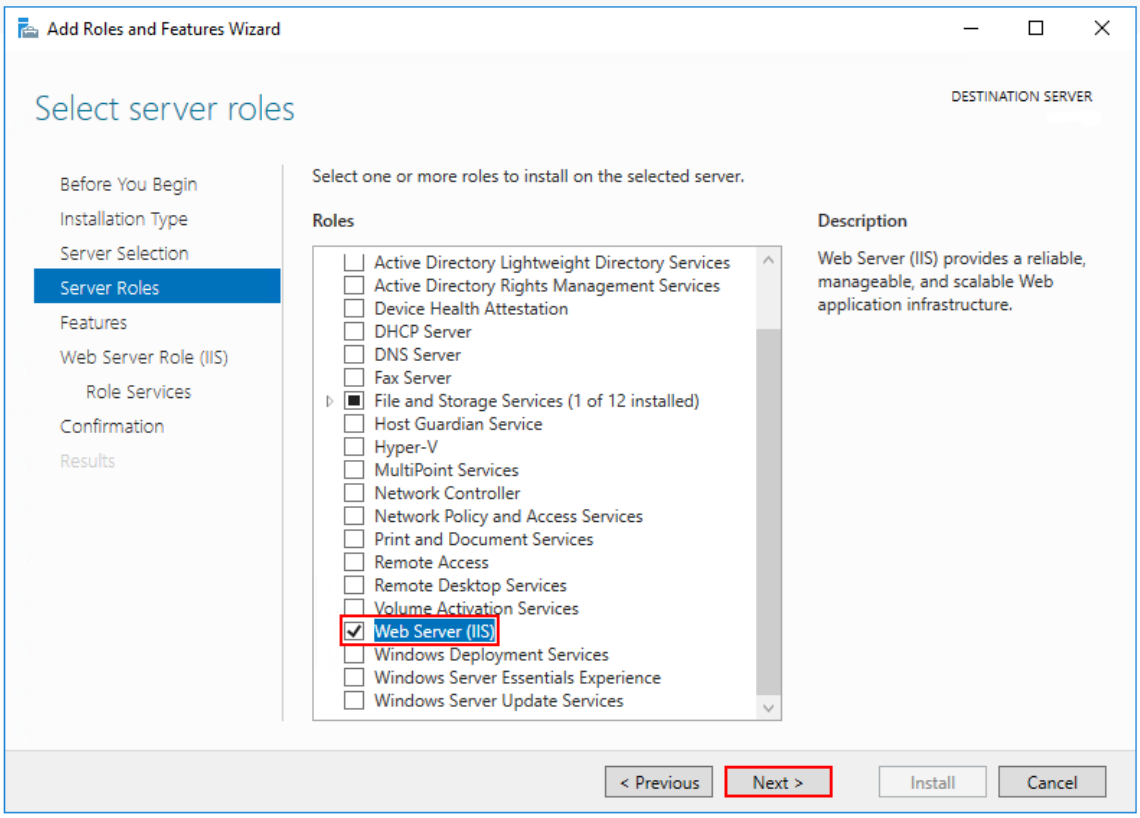
Конфигурация IIS

Операционные системы семейства Windows Server

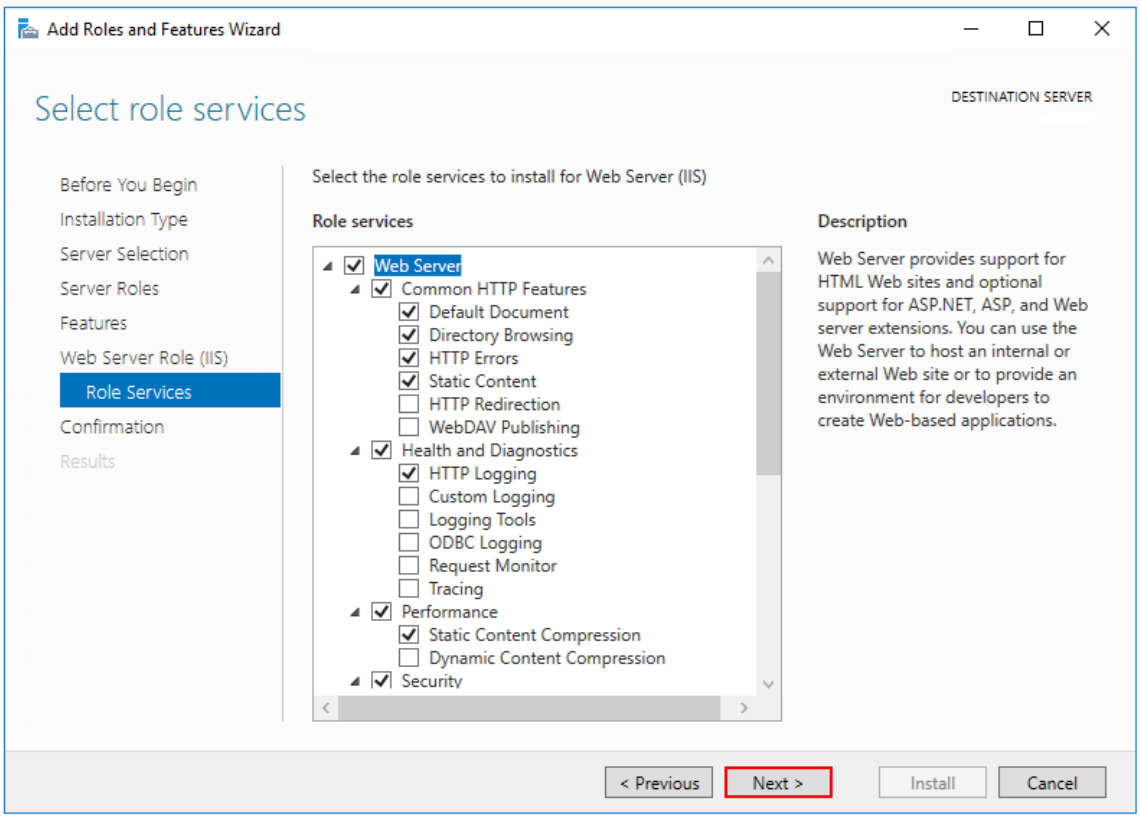
Включите роль сервера **Веб-сервер (IIS)** и настройте службы роли.

1. В меню **Управление** запустите мастер **Добавить роли и компоненты** или в окне **Диспетчер**

серверов щелкните соответствующую ссылку. На этапе **Роли сервера** установите флажок **Веб-сервер (IIS)**.



2. После этапа выбора **компонентов** запускается этап выбора **служб роли** для веб-сервера (IIS). Выберите нужные службы роли IIS или оставьте службы по умолчанию.



Проверка подлинности Windows (необязательный компонент)

Чтобы включить проверку подлинности Windows, разверните такие узлы: **Веб-сервер** > **Безопасность**. Выберите компонент **Проверка подлинности Windows**. Дополнительные сведения см. в статьях Windows Authentication <windowsAuthentication> (Проверка подлинности

Windows) и Configure Windows authentication in an ASP.NET Core app (Настройка проверки подлинности Windows в приложении ASP.NET Core).

Протокол WebSocket (необязательный компонент)
Протокол WebSocket поддерживается в ASP.NET Core начиная с версии 1.1. Чтобы включить протокол WebSocket, разверните такие узлы: **Веб-сервер** > **Разработка приложений**. Выберите компонент **Протокол WebSocket**. Дополнительные сведения см. в разделе Протокол WebSocket.

3. Пройдите этап **Подтверждение**, чтобы установить роль и службы веб-сервера. После установки роли **Веб-сервер (IIS)** перезагружать сервер или службы IIS не требуется.

Операционные системы Windows для настольных компьютеров

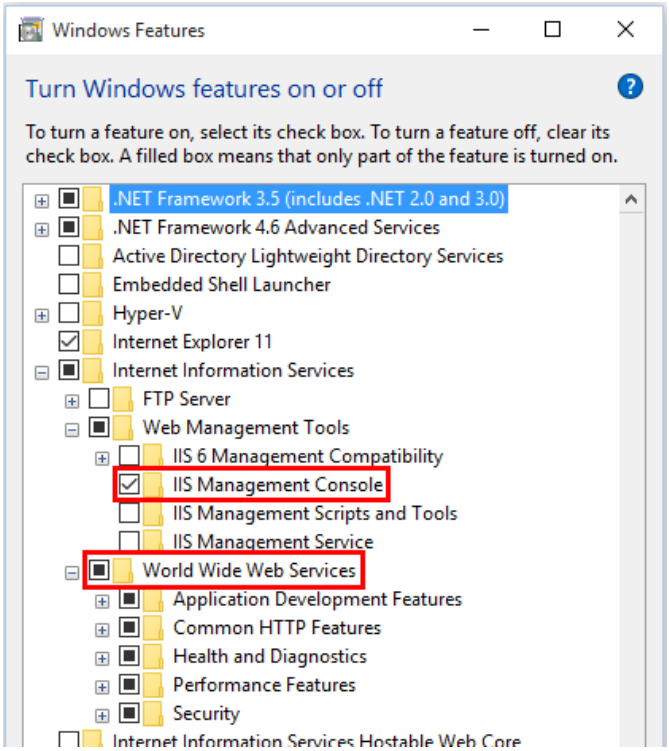
Включите компоненты **Консоль управления IIS** и **Службы Интернета**.

1. Последовательно выберите **Панель управления** > **Программы** > **Программы и компоненты** > **Включение или отключение компонентов Windows** (в левой части экрана).
2. Разверните узел **Службы IIS**. Разверните узел **Средства управления веб-сайтом**.
3. Установите флажок **Консоль управления IIS**.
4. Установите флажок **Службы Интернета**.
5. В группе **Службы Интернета** оставьте компоненты по умолчанию или настройте компоненты IIS.

   Проверка подлинности Windows (необязательный компонент)
   Чтобы включить проверку подлинности Windows, разверните такие узлы: **Службы Интернета** > **Безопасность**. Выберите компонент **Проверка подлинности Windows**. Дополнительные сведения см. в статьях Windows Authentication <windowsAuthentication> (Проверка подлинности Windows) и Configure Windows authentication in an ASP.NET Core app (Настройка проверки подлинности Windows в приложении ASP.NET Core).

   Протокол WebSocket (необязательный компонент)
   Протокол WebSocket поддерживается в ASP.NET Core начиная с версии 1.1. Чтобы включить протокол WebSocket, разверните такие узлы: **Службы Интернета** > **Компоненты разработки приложений**. Выберите компонент **Протокол WebSocket**. Дополнительные сведения см. в разделе Протокол WebSocket.

6. Если во время установки IIS потребуется перезагрузка, перезагрузите систему.



Установка пакета размещения .NET Core

Установите *пакет размещения .NET Core* в размещающей системе. В составе пакета устанавливаются среда выполнения .NET Core, библиотека .NET Core и модуль ASP.NET Core. Модуль создает обратный прокси-сервер между службами IIS и сервером Kestrel. Если система не подключена к Интернету, перед установкой пакета размещения .NET Core получите и установите Распространяемый компонент Microsoft Visual C++ 2015.

> **IMPORTANT**
> Если пакет размещения устанавливается до установки служб IIS, его нужно восстановить. После установки служб IIS запустите установщик пакета размещения еще раз.

Прямая загрузка (текущая версия)

Скачайте установщик по следующей ссылке:

Текущий установщик пакета размещения .NET Core (прямая загрузка)

Более ранние версии установщика

Получение более ранней версии установщика:

1. Перейдите в архивы загрузок .NET.
2. В разделе **.NET Core** выберите версию .NET Core.
3. В столбце **Запуск приложений — среда выполнения** найдите строку, содержащую нужную версию среды выполнения .NET Core.
4. Скачайте установщик по ссылке **Среда выполнения и пакет размещения**.

> **WARNING**
> Некоторые установщики содержат версии выпусков, которые достигли конца своего жизненного цикла и больше не поддерживаются корпорацией Майкрософт. Дополнительные сведения см. в разделе Политика поддержки.

Установка пакета размещения .NET Core

1. Запустите установщик на сервере. При запуске установщика из командной строки администратора доступны следующие параметры:

   - `OPT_NO_ANCM=1` — пропуск установки модуля ASP.NET Core;
   - `OPT_NO_RUNTIME=1` — пропуск установки среды выполнения .NET Core;
   - `OPT_NO_SHAREDFX=1` — пропуск установки общей платформы ASP.NET (среды выполнения ASP.NET);
   - `OPT_NO_X86=1` — пропуск установки 32-разрядных сред выполнения. Этот параметр следует использовать, если вы наверняка не будете размещать 32-разрядные приложения. Если есть хоть малейшая возможность, что в будущем придется размещать и 32-разрядные, и 64-разрядные приложения, не указывайте этот параметр и установите обе среды выполнения.
2. Перезагрузите систему или в командой строке выполните команду **net stop was /y**, а затем — команду **net start w3svc**. Перезапуск служб IIS позволит обнаружить внесенные установщиком изменения в системном пути, который является переменной среды.

Если установщик пакета размещения Windows обнаруживает, что для завершения установки требуется сброс IIS, установщик выполняет этот сброс. Если установщик применяет сброс IIS, перезапускаются все пулы приложений и веб-сайты IIS.

> **NOTE**
> Сведения об общей конфигурации IIS см. в разделе Модуль ASP.NET Core с общей конфигурацией IIS.

Установка веб-развертывания при публикации с помощью Visual Studio

При развертывании приложений на серверах с помощью веб-развертывания установите на сервере последнюю версию веб-развертывания. Чтобы установить веб-развертывание, можно использовать установщик веб-платформы (WebPI) или получить установщик непосредственно в Центре загрузки Майкрософт. Предпочтительный метод — использовать WebPI. WebPI обеспечивает автономную установку и настройку поставщиков размещения.

Создание сайта IIS

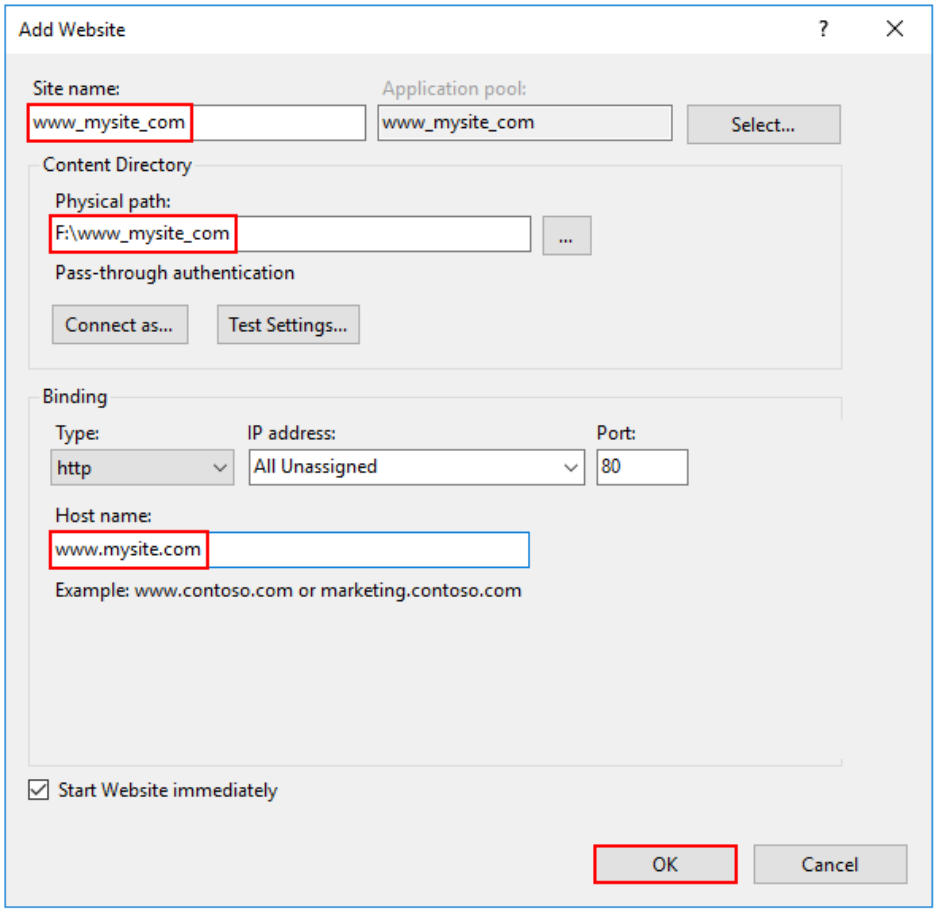
1. В размещающей системе создайте папку, в которой будут храниться файлы и папки опубликованного приложения. Макет развертывания приложения описан в статье Directory structure of published ASP.NET Core apps (Структура каталогов опубликованных приложений ASP.NET Core).

2. В созданной только что папке создайте папку *logs*, в которой будут храниться журналы StdOut модуля ASP.NET Core (если включено ведение таких журналов). Если приложение развертывается с папкой *logs* в полезных данных, пропустите этот шаг. Сведения о том, как в MSBuild настроить автоматическое создание папки *logs* при создании проекта локально, см. в статье Directory structure of published ASP.NET Core apps (Структура каталогов опубликованных приложений ASP.NET Core).

> **IMPORTANT**
>
> Журнал StdOut следует использовать только для устранения ошибок, возникающих при запуске приложения. Никогда не используйте журнал StdOut как журнал повседневных операций приложения. Ни размер файла журнала, ни количество создаваемых файлов журналов ничем не ограничены. Пул приложений должен иметь доступ на запись к папке, куда записываются журналы. Все папки в пути к папке журнала должны существовать. Дополнительные сведения о журнале StdOut см. в разделе о создании и перенаправлении журналов. Сведения о ведении журнала приложения ASP.NET Core см. в статье Общие сведения о ведении журналов в ASP.NET Core.

3. В окне **Диспетчер служб IIS** в области **Подключения** разверните узел сервера. Щелкните правой кнопкой мыши папку **Сайты**. В контекстном меню выберите пункт **Добавить веб-сайт**.

4. Укажите имя в поле **Имя сайта** и задайте значение в поле **Физический путь** для созданной папки развертывания приложения. Укажите конфигурацию **привязки** и нажмите кнопку **OK**, чтобы создать веб-сайт.



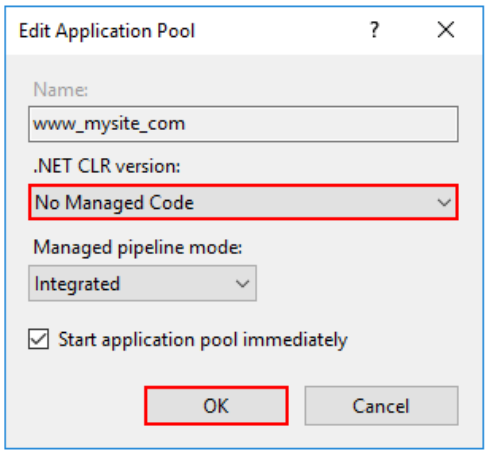
> **WARNING**
>
> **Не используйте** привязки с подстановочными знаками (`http://*:80/` и `http://+:80`) на верхнем уровне. Это может создать уязвимость и поставить ваше приложение под угрозу. Сюда относятся и строгие, и нестрогие подстановочные знаки. Вместо этого используйте имена узлов в явном виде. Привязки с подстановочными знаками на уровне дочерних доменов (например `*.mysub.com`) не создают таких угроз безопасности, если вы полностью контролируете родительский домен (в отличие от варианта `*.com` , создающего уязвимость). Дополнительные сведения см. в документе rfc7230, раздел 5.4.

5. Разверните узел сервера и выберите **Пулы приложений**.

6. Щелкните правой кнопкой мыши пул приложений сайта и в контекстном меню выберите пункт

Основные параметры.

7. В окне **Изменение пула приложений** задайте для параметра **Версия среды CLR .NET** значение **Без управляемого кода**.



   ASP.NET Core выполняется в отдельном процессе и управляет средой выполнения. Для ASP.NET Core не требуется загрузка классической среды CLR. Задавать для параметра **Версия среды CLR .NET** значение **Без управляемого кода** необязательно.

8. Убедитесь в том, что удостоверение модели процесса имеет соответствующие разрешения.

   При изменении удостоверения по умолчанию для пула приложений (**Модель процесса** > **Удостоверение**) с **ApplicationPoolIdentity** на другое, убедитесь, что у нового удостоверения есть соответствующие разрешения для доступа к папке приложения, базе данных и другим необходимым ресурсам. Например, пулу приложений требуются права на чтение и запись в папках, в которых приложение считывает и записывает файлы.

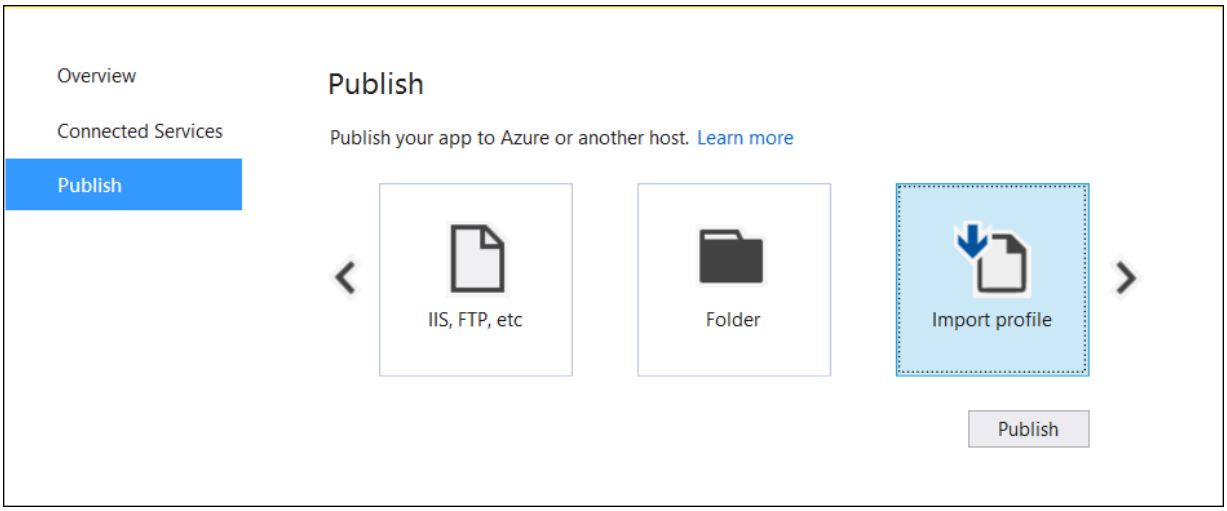
Настройка проверки подлинности Windows (необязательный этап)
См. статью Configure Windows authentication in an ASP.NET Core app (Настройка проверки подлинности Windows в приложении ASP.NET Core).

Развертывание приложения

Разверните приложение в папке, созданной в размещающей системе. Рекомендуемый механизм развертывания — веб-развертывание.

Веб-развертывание с помощью Visual Studio

Сведения о создании профиля публикации для веб-развертывания см. в статье Профили публикации Visual Studio для развертывания приложений ASP.NET Core. Если поставщик услуг размещения предоставляет профиль публикации или позволяет его создать, скачайте этот профиль и импортируйте его с помощью диалогового окна **Публикация** в Visual Studio.



Веб-развертывание вне Visual Studio

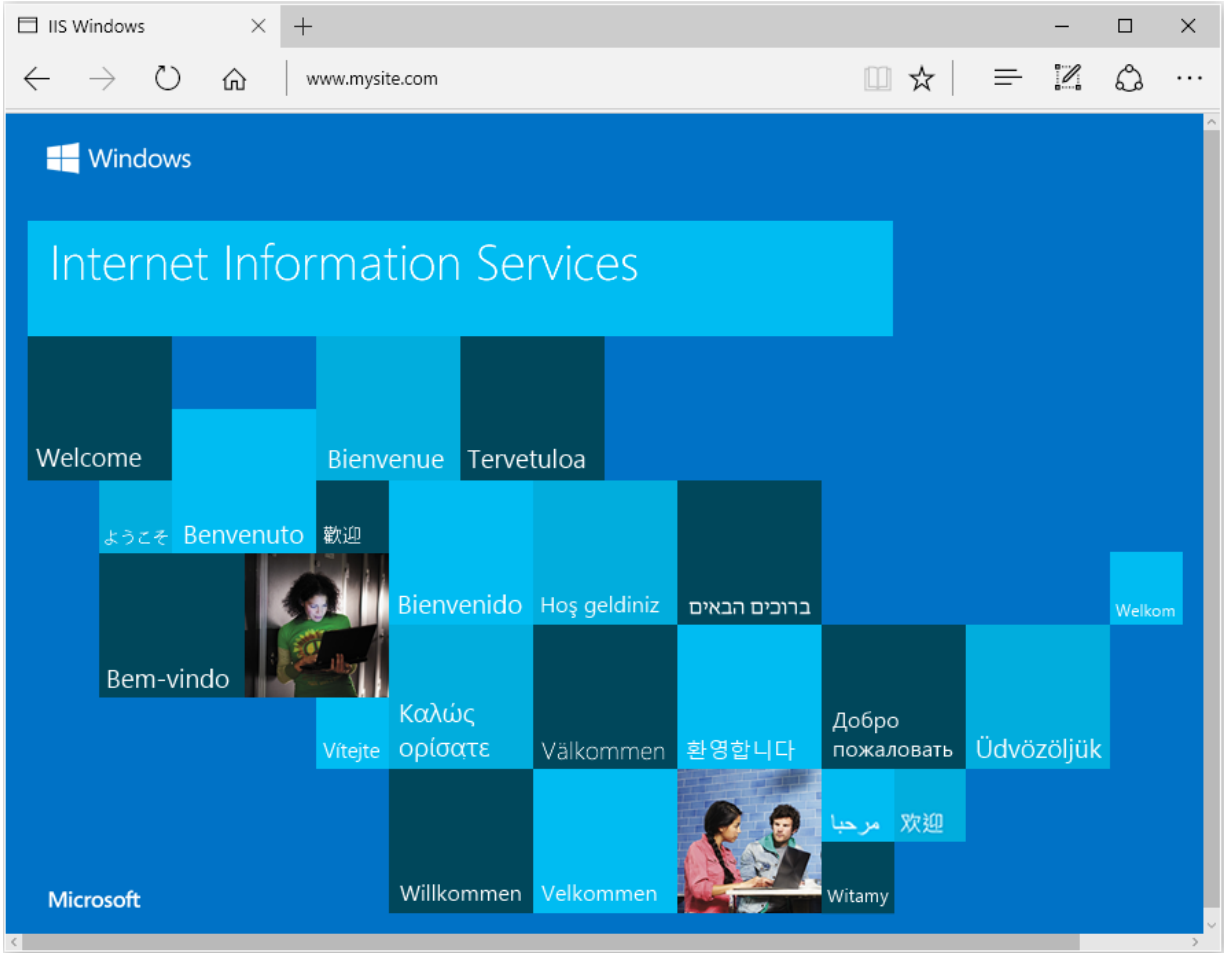
Веб-развертывание можно также использовать вне Visual Studio с помощью командной строки. Дополнительные сведения см. в статье Средство веб-развертывания.

Альтернативы веб-развертыванию

Переместить приложение в размещающую систему можно несколькими способами: с помощью копирования вручную, Xcopy, Robocopy или PowerShell.

Дополнительные сведения о развертывании ASP.NET Core в службах IIS см. в разделе Ресурсы развертывания для администраторов IIS.

Обзор веб-сайта



Заблокированные файлы развертывания

Во время выполнения приложения файлы в папке развертывания блокируются. Заблокированные файлы невозможно перезаписать во время развертывания. Чтобы снять блокировку с файлов в развертывании, остановите пул приложений с помощью **одного** из следующих методов:

- Запустите веб-развертывание и добавьте ссылку на `Microsoft.NET.Sdk.Web` в файл проекта. Файл *app_offline.htm* помещается в корень каталога веб-приложения. Если файл присутствует, модуль ASP.NET Core корректно завершает работу приложения и обслуживает файл *app_offline.htm* во время развертывания. Дополнительные сведения см. в разделе Справочник по конфигурации модуля ASP.NET Core.
- Вручную остановите пул приложений в диспетчере служб IIS на сервере.
- Используйте PowerShell для остановки и перезапуска пула приложений (требуется PowerShell 5 или более поздняя версия).

```
$webAppPoolName = 'APP_POOL_NAME'

# Stop the AppPool
if((Get-WebAppPoolState $webAppPoolName).Value -ne 'Stopped') {
    Stop-WebAppPool -Name $webAppPoolName
    while((Get-WebAppPoolState $webAppPoolName).Value -ne 'Stopped') {
        Start-Sleep -s 1
    }
    Write-Host `-AppPool Stopped
}

# Provide script commands here to deploy the app

# Restart the AppPool
if((Get-WebAppPoolState $webAppPoolName).Value -ne 'Started') {
    Start-WebAppPool -Name $webAppPoolName
    while((Get-WebAppPoolState $webAppPoolName).Value -ne 'Started') {
        Start-Sleep -s 1
    }
    Write-Host `-AppPool Started
}
```

Защита данных

Стек защиты данных ASP.NET Core используют несколько ПО промежуточного слоя ASP.NET Core, включая ПО, которое применяется для аутентификации. Даже если API-интерфейсы защиты данных не вызываются из пользовательского кода, защиту данных следует настроить для создания постоянного хранилища криптографических ключей. Это можно сделать с помощью скрипта развертывания или в пользовательском коде. Если защита данных не настроена, ключи хранятся в памяти и удаляются при перезапуске приложения.

Если набор ключей хранится в памяти, при перезапуске приложения происходит следующее:

- Все токены аутентификации, использующие файлы cookie, становятся недействительными.
- При выполнении следующего запроса пользователю требуется выполнить вход снова.
- Все данные, защищенные с помощью набора ключей, больше не могут быть расшифрованы. Это могут быть токены CSRF и файлы cookie временных данных ASP.NET Core MVC.

Чтобы настроить защиту данных в службах IIS для хранения набора ключей, воспользуйтесь **одним** из следующих методов:

- **Создание раздела реестра для защиты данных**.

  Ключи защиты данных, используемые приложениями ASP.NET Core, хранятся во внешнем для приложений реестре. Чтобы хранить эти ключи для определенного приложения, нужно создать

разделы реестра для пула приложений.

При автономной установке служб IIS, не поддерживающей веб-ферму, можно выполнить скрипт PowerShell Provision-AutoGenKeys.ps1 для защиты данных применительно к каждому пулу приложений, в который входит приложение ASP.NET Core. Этот скрипт создает раздел в реестре HKLM, который будет доступен только для учетной записи рабочего процесса пула приложений, к которому относится соответствующее приложение. Неактивные ключи шифруются с помощью API защиты данных с ключом компьютера.

В случаях, когда используется веб-ферма, в приложении можно настроить UNC-путь, по которому это приложение будет хранить набор ключей для защиты данных. По умолчанию ключи защиты данных не шифруются. Разрешения на доступ к файлам в сетевой папке должны быть предоставлены только учетной записи Windows, с помощью которой выполняется приложение. Для защиты неактивных ключей можно использовать сертификат X509. Рассмотрите возможность реализации механизма, позволяющего пользователям отправлять сертификаты: поместить сертификаты в хранилище доверенных сертификатов пользователя и обеспечивать их доступность на всех компьютерах, где выполняется приложение. Дополнительные сведения см. в разделе Настройка защиты данных в ASP.NET Core.

- **Настройка пула приложений IIS для загрузки профиля пользователя**.

  Этот параметр находится на странице **Дополнительные параметры** пула приложений в разделе **Модель процесса**. Задайте для параметра "Загрузить профиль пользователя" значение `True`. В результате ключи будут храниться в каталоге профиля пользователя, а их безопасность будет обеспечиваться за счет применения API защиты данных и ключа на уровне учетной записи пользователя, которую использует пул приложений.

- **Использование файловой системы в качестве хранилища набора ключей**.

  Измените код приложения так, чтобы в качестве хранилища набора ключей использовалась файловая система. Для защиты набора ключей используйте доверенный сертификат X509. Если сертификат — самозаверяющий, поместите его в доверенное корневое хранилище.

  При использовании служб IIS в веб-ферме:

  - используйте общую папку, которая доступна всем компьютерам;
  - разверните сертификат X509 на каждом компьютере; настройте защиту данных в коде.

- **Настройка политики защиты данных на уровне компьютера**.

  Система защиты данных обеспечивает ограниченную поддержку задания политики по умолчанию на уровне компьютера для всех приложений, использующих интерфейсы API защиты данных. Дополнительные сведения см. в разделе Защита данных в ASP.NET Core.

Конфигурация дочерних приложений

Дочерние приложения, добавленные в корневое приложение, не должны включать в себя модуль ASP.NET Core в качестве обработчика. Если модуль добавляется в качестве обработчика в файл *web.config* дочернего приложения, при попытке работы с этим приложением выводится ошибка *500.19* (внутренняя ошибка сервера) с указанием некорректного файла конфигурации.

Вот пример опубликованного файла *web.config* для дочернего приложения ASP.NET Core:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.webServer>
    <aspNetCore processPath="dotnet"
      arguments=".\<assembly_name>.dll"
      stdoutLogEnabled="false"
      stdoutLogFile=".\logs\stdout" />
  </system.webServer>
</configuration>
```

При размещении дочернего приложения не на основе ASP.NET Core в приложении ASP.NET Core, нужно явным образом удалить унаследованный обработчик из файла *web.config* дочернего приложения.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.webServer>
    <handlers>
      <remove name="aspNetCore" />
    </handlers>
    <aspNetCore processPath="dotnet"
      arguments=".\<assembly_name>.dll"
      stdoutLogEnabled="false"
      stdoutLogFile=".\logs\stdout" />
  </system.webServer>
</configuration>
```

Дополнительные сведения о настройке модуля ASP.NET Core см. в статье Введение в модуль ASP.NET Core Module и в справочнике по настройке модуля ASP.NET Core.

Настройка служб IIS с помощью файла web.config

Раздел **<system.webServer>** файла *web.config* действует для тех компонентов IIS, которые относятся к конфигурации прокси-сервера. Если в службах IIS на уровне сервера настроено динамическое сжатие, элемент **<urlCompression>** в файле *web.config* приложения может отключить это сжатие.

Дополнительные сведения см. в справочнике по настройке <system.webServer>, справочнике по настройке модуля ASP.NET Core и статье Модули IIS с ASP.NET Core. Сведения о настройке переменных среды для отдельных приложений, выполняющихся в изолированных пулах приложений (такая возможность поддерживается в службах IIS начиная с версии 10.0), см. в справочной документации по службам IIS, в частности в разделе *AppCmd.exe* статьи Environment Variables <environmentVariables> (Переменные среды).

Разделы конфигурации файла web.config

Разделы конфигурации приложений ASP.NET 4.x в файле *web.config* не используются для конфигурации приложений ASP.NET Core.

- **<system.web>**
- **<appSettings>**
- **<connectionStrings >**
- **<location>**

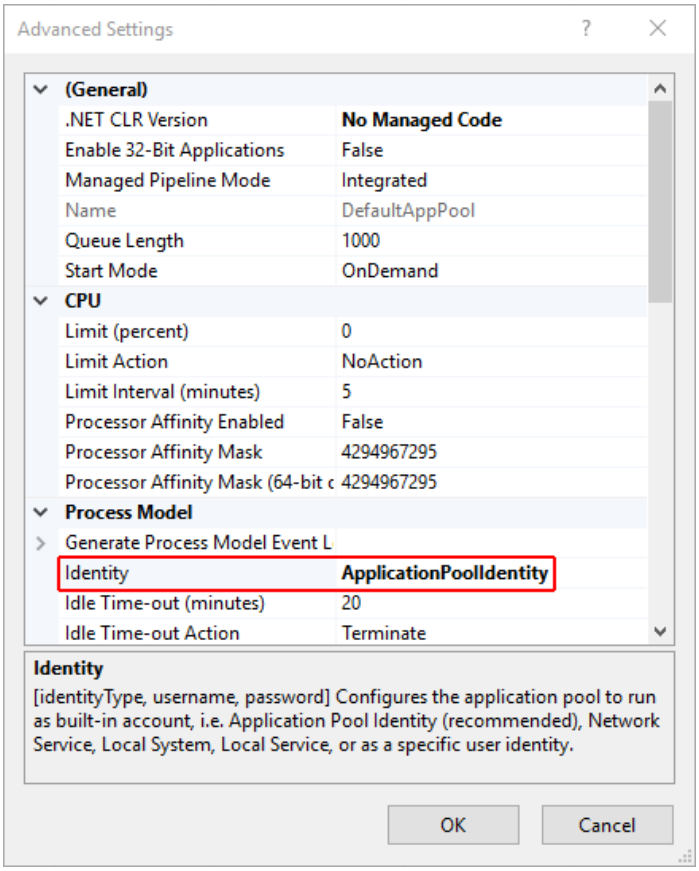
Для настройки приложений ASP.NET Core используются другие поставщики конфигураций. Дополнительные сведения см. в разделе Конфигурация.

Пулы приложений

При размещении на сервере множества веб-сайтов рекомендуем изолировать приложения друг от друга, запуская каждое из них в собственном пуле. В диалоговом окне **Добавить веб-сайт** служб IIS такая конфигурация задана по умолчанию. Если указано **имя сайта**, оно автоматически переносится в текстовое поле **Пул приложений**. При добавлении веб-сайта создается пул приложений с именем сайта.

Удостоверение пула приложений

Учетная запись удостоверения пула приложений позволяет запускать приложение от имени уникальной учетной записи, не создавая домены или локальные учетные записи и не управляя ими. В службах IIS начиная с версии 8.0 рабочий процесс администратора IIS (WAS) создает виртуальную учетную запись с именем нового пула приложений и по умолчанию выполняет рабочие процессы пула приложений с помощью этой учетной записи. В консоли управления IIS в окне **Дополнительные параметры** для пула приложений должно быть выбрано **Удостоверение ApplicationPoolIdentity**.

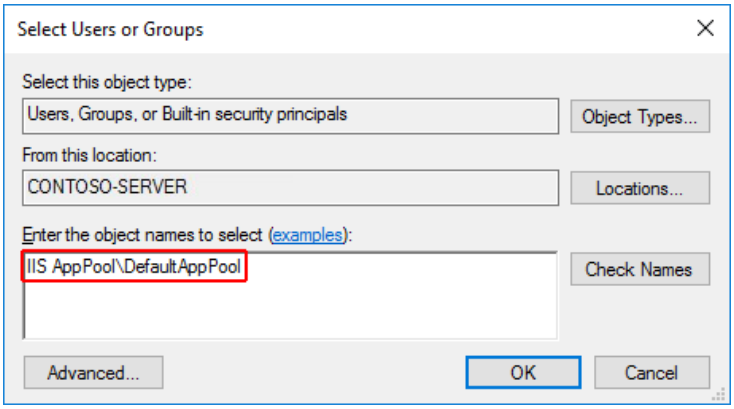


Процесс управления IIS создает в системе безопасности Windows безопасный идентификатор с именем пула приложений. С помощью этого идентификатора можно защищать ресурсы, но он не является реальной учетной записью и не отображается в консоли управления пользователями Windows.

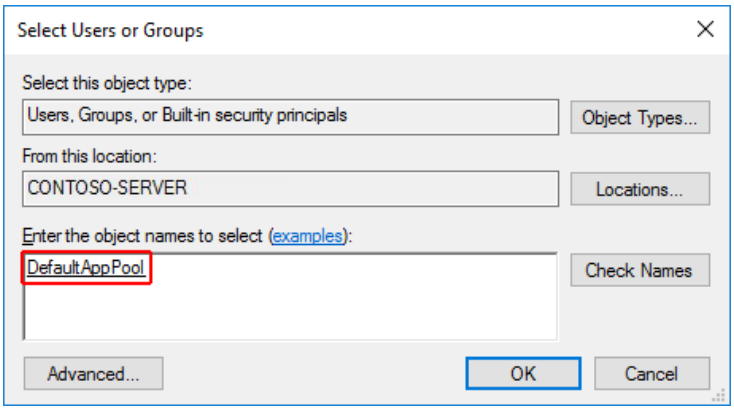
Если рабочему процессу IIS требуется предоставить доступ к приложению с повышенными правами, измените список управления доступом (ACL) для каталога, содержащего приложение.

1. Откройте проводник и перейдите к каталогу.
2. Щелкните каталог правой кнопкой мыши и выберите пункт **Свойства**.
3. На вкладке **Безопасность** нажмите кнопку **Изменить**, а затем — кнопку **Добавить**.
4. Нажмите кнопку **Размещение** и выберите систему.
5. В поле **Введите имена выбираемых объектов** введите **IIS AppPool\ <имя_пула_приложений>**. Нажмите кнопку **Проверить имена**. В случае с *DefaultAppPool* проверьте имена с помощью **IIS AppPool\DefaultAppPool**. После нажатия кнопки **Проверить имена** в области имен объектов отобразится значение **DefaultAppPool**. Вручную ввести имя пула

приложений в области имен объектов нельзя. При поиске имени объекта используйте формат **IIS AppPool\<имя_пула_приложений>**.



6. Нажмите кнопку **OK**.



7. Разрешения на чтение и выполнение предоставляются по умолчанию. При необходимости предоставьте дополнительные разрешения.

Кроме того, доступ можно предоставить через командную строку, используя средство **ICACLS**. В случае с *DefaultAppPool* выполняется такая команда:

```
ICACLS C:\sites\MyWebApp /grant "IIS AppPool\DefaultAppPool":F
```

Дополнительные сведения об ICACLS см. здесь.

Ресурсы развертывания для администраторов IIS

Дополнительные сведения о службах IIS см. в документации по ним.
Документация по службам IIS

Дополнительные сведения о моделях развертывания приложения .NET Core.
Развертывание приложений .NET Core

Сведения о том, как модуль ASP.NET Core позволяет веб-серверу Kestrel использовать IIS или IIS Express в качестве обратного прокси-сервера.
Модуль ASP.NET Core

Сведения о настройке модуля ASP.NET Core для размещения приложений ASP.NET Core.
Справочник по конфигурации модуля ASP.NET Core

Сведения о структуре каталогов опубликованных приложений ASP.NET Core.
Структура каталогов

Сведения об обнаружении активных и неактивных модулей IIS для приложения ASP.NET Core и

управлении модулями IIS.
Модули IIS

Сведения о диагностике проблем с развертываниями IIS приложений ASP.NET Core.
Устранение неполадок

Распознавание распространенных ошибок при размещении приложений ASP.NET Core в IIS.
Справочник по общим ошибкам для службы приложений Azure и служб IIS

Дополнительные ресурсы

- Введение в ASP.NET Core
- Официальный веб-сайт Microsoft IIS
- Библиотека технического содержимого по Windows Server
- HTTP/2 в IIS