**«ОБЛАЧНЫЕ ТЕХНОЛОГИИ»**

**ЛАБОРАТОРНАЯ РАБОТА № 9**

**"Создание приложения ASP.NET в Azure с подключением к базе данных SQL"**

***Цель*** – создать приложение ASP.NET в Azure с подключением к службе "База данных SQL"".

При выполнении лабораторной работы изучаются приемы выполнения следующих задач:

1. Создание базы данных SQL в Azure.
2. Подключение приложения ASP.NET к базе данных SQL.
3. Развертывание приложения в Azure
4. Обновление модели данных и повторное развертывание приложения.
5. Потоковая передача журналов из Azure в окно терминала.
6. Управление приложением на портале Azure.

## Краткие теоретические сведения

[Служба приложений Azure](https://docs.microsoft.com/ru-ru/azure/app-service/overview) — это служба веб-размещения с самостоятельной установкой исправлений и высоким уровнем масштабируемости. В этой практической работе показано, как развернуть управляемое данными приложение ASP.NET в Службе приложений, а затем подключить его к службе [База данных SQL Azure](https://docs.microsoft.com/ru-ru/azure/sql-database/sql-database-technical-overview).

**Служба приложений Azure** — это полностью управляемая вычислительная платформа в среде Azure, которая оптимизирована для размещения веб-приложений, REST API и серверных частей мобильных приложений. Этот сервис Microsoft Azure уровня PaaS (платформа как услуга) позволяет разработчикам полностью сосредоточиться на сборке приложений. Настройка инфраструктуры для запуска и масштабирования приложений выполняется в Azure автоматически.

**Группа ресурсов** — это метод группирования взаимозависимых ресурсов и служб, таких как виртуальные машины, веб-приложения, базы данных и пр., используемых конкретным приложением в конкретной среде. Можно рассматривать группу ресурсов как **папку**, в которой размещаются элементы приложения.

Группы ресурсов позволяют легко управлять ресурсами и удалять их. Они также упрощают мониторинг, управление доступом, подготовку и управление выставлением счетов для коллекций ресурсов, которые используются для запуска приложения или потребляются клиентом.

**План службы приложений** — это набор физических ресурсов и доступной емкости, используемых для развертывания приложений в Службе приложений.

Портал Azure предлагает шаблон для создания плана службы приложений. В этом шаблоне нужно указать следующие базовые сведения:

* регион (Западная часть США, Центральная часть США, Северная Европа и т. д.);
* уровень масштабирования (один, два, три экземпляра и т. д.);
* размер экземпляра (небольшой, средний или крупный);
* номер SKU или ценовая категория ("Бесплатный", "Общий", "Базовый", "Стандартный", "Премиум", "Премиум V2" и "Изолированный").

В рамках плана службы приложений Azure выполняются веб-приложения, мобильные приложения, приложения API в составе Службы приложений Azure и Функции Azure. В любом плане службы приложений можно развернуть неограниченное число приложений, но реально используемое число обычно зависит от типов развернутых приложений, требуемых для них ресурсов и уровня потребления ЦП.

Для плана службы приложений на портале Azure всегда можно отобразить уровни использования ЦП и памяти, чтобы оценить потребности в масштабировании или перемещении приложений в другой план службы приложений.

**Служба "База данных SQL" Azure.** База данных SQL — это реляционная база данных общего назначения (управляемая служба) в Microsoft Azure, которая поддерживает такие структуры, как реляционные данные, JSON, пространственные данные и XML. База данных SQL предоставляет возможности динамического масштабирования производительности в двух разных моделях приобретения: на основе виртуальных ядер и DTU. База данных SQL также предоставляет такие компоненты, как [индексы columnstore](https://docs.microsoft.com/sql/relational-databases/indexes/columnstore-indexes-overview) для экстремального аналитического анализа и отчетности и [выполняющуюся в памяти OLTP](https://docs.microsoft.com/ru-ru/azure/sql-database/sql-database-in-memory) для экстремальной обработки транзакций. Майкрософт быстро и эффективно обрабатывает исправления и обновления базы кода SQL, а также упрощает обслуживание базовой инфраструктуры.

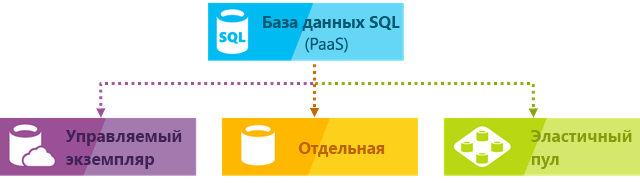
**Возможности службы База данных SQL Azure**. Данная служба предоставляет следующие возможности развертывания для Базы данных SQL Azure:

[Отдельная база данных](https://docs.microsoft.com/ru-ru/azure/sql-database/sql-database-single-database) с собственным набором ресурсов, управляемая с помощью сервера Базы данных SQL. Отдельная база данных аналогична [автономным базам данных](https://docs.microsoft.com/sql/relational-databases/databases/contained-databases) на SQL Server.

[Эластичный пул](https://docs.microsoft.com/ru-ru/azure/sql-database/sql-database-elastic-pool) — это коллекция баз данных с общим набором ресурсов, управляемая с помощью сервера Базы данных SQL. Отдельные базы данных можно перемещать в эластичный пул и обратно.

[Управляемый экземпляр](https://docs.microsoft.com/ru-ru/azure/sql-database/sql-database-managed-instance), который представляет коллекцию системных и пользовательских баз данных с общим набором ресурсов. Управляемый экземпляр аналогичен экземпляру [ядра СУБД Microsoft SQL Server](https://docs.microsoft.com/sql/sql-server/sql-server-technical-documentation).

Эти варианты развертывания показаны на рисунке ниже:



База данных SQL использует свою базу кода совместно с [ядром СУБД Microsoft SQL Server](https://docs.microsoft.com/sql/sql-server/sql-server-technical-documentation). С облачной стратегией Майкрософт новейшие возможности SQL Server сначала выпускаются для базы данных SQL, а затем для самой платформы SQL Server. Такой подход позволяет использовать новейшие возможности SQL Server без лишних расходов на внесение исправлений или обновлений. При этом эти функции проверены на миллионах баз данных.

## Порядок выполнения

### Замечания по назначению имен и других параметров используемых при выполнении работы.

При выполнении данной работы используйте следующие рекомендации по назначению имен различных ресурсов и других параметров (имена приложений, групп ресурсов, серверов, баз данных и т.п.):

1. При назначении имен, которые должны быть уникальными, используйте имена указанные в тексте или их модификации и обязательно включайте индивидуальный префикс: - xxyysssss

Где: xx – номер учебной группы;

yy – номер по журналу;

sssss – сокращенное наименование специальности (например, poit, isit, и т.п.).

2. При назначении паролей следуйте подсказкам, выдаваемым при редактировании соответствующих полей.

## Задание 1. Подготовка к выполнению работы

### Предварительные требования

Для выполнения работы с требуется:

1. Установить [Visual Studio 2017](https://www.visualstudio.com/downloads/) с рабочей нагрузкой **ASP.NET и веб-разработка**.

Если вы уже установили Visual Studio, добавьте в него рабочие нагрузки. Для этого последовательно выберите **Инструменты** > **Get Tools and Features (Получить инструменты и функции)**.

2. Если у вас еще нет [подписки Azure](https://docs.microsoft.com/azure/guides/developer/azure-developer-guide#understanding-accounts-subscriptions-and-billing), создайте [бесплатную учетную запись Azure](https://azure.microsoft.com/free/?ref=microsoft.com&utm_source=microsoft.com&utm_medium=docs&utm_campaign=visualstudio), прежде чем начать работу.

### Шаг 1. Скачивание примера приложения

1.[Загрузите пример проекта](https://github.com/Azure-Samples/dotnet-sqldb-tutorial/archive/master.zip).

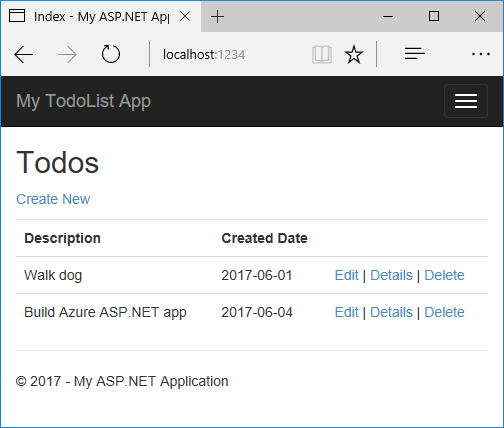
2. Извлеките (распакуйте) файл *dotnet-sqldb-tutorial-master.zip*.

Этот пример проекта содержит простое CRUD-приложение [ASP.NET MVC](https://www.asp.net/mvc), созданное на основе [Entity Framework Code First](https://docs.microsoft.com/ru-ru/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application).

### Шаг 2. Запуск приложения

1. Откройте в Visual Studio файл решения dotnet-sqldb-tutorial-master/DotNetAppSqlDb.sln.

2. Введите Ctrl+F5, чтобы запустить приложение без отладки. Это приложение откроется в браузере по умолчанию. Щелкните ссылку **Создать (Create New)**, чтобы создать несколько элементов *списка дел*.



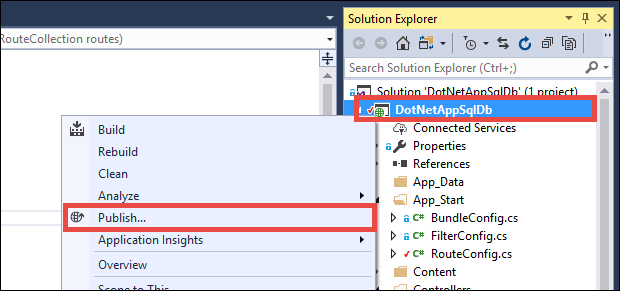
3. Проверьте работу приложения при переходе по ссылкам **Изменить (Edit)**, **Сведения (Details)** и **Удалить (Delete)**.

Чтобы подключиться к базе данных, приложение использует контекст базы данных. В этом примере контекст базы данных использует строку подключения MyDbConnection. Строка подключения определена в файле *Web.config*. Ссылка на нее также имеется в файле *Models/MyDatabaseContext.cs*. Имя строки подключения потребуется далее в этом руководстве при подключении приложения Azure к Базе данных SQL.

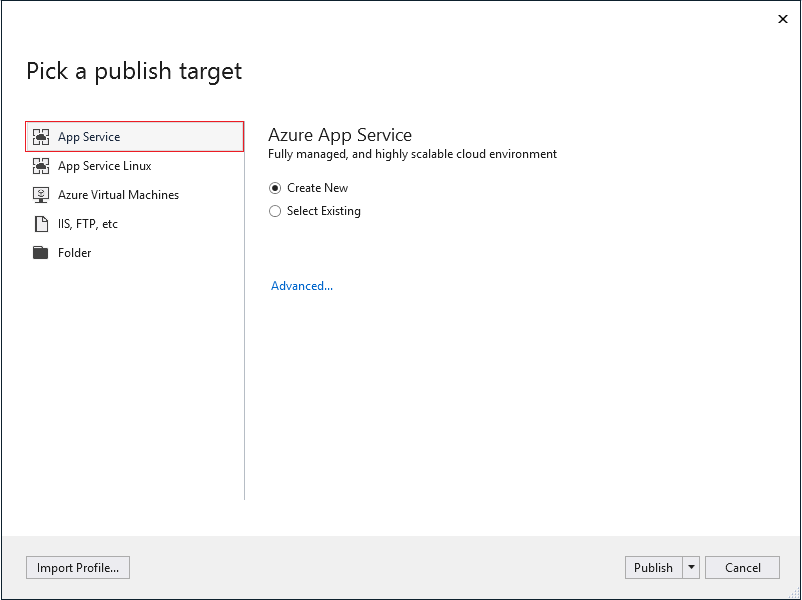
## Задание 2. Публикация приложения с базой данных SQL в Azure

### Шаг 1. Опубликовать проект в Azure

1. В **обозревателе решений** щелкните правой кнопкой мыши проект **DotNetAppSqlDb** и выберите **Опубликовать**.



Откроется окно "Pick a publish target"



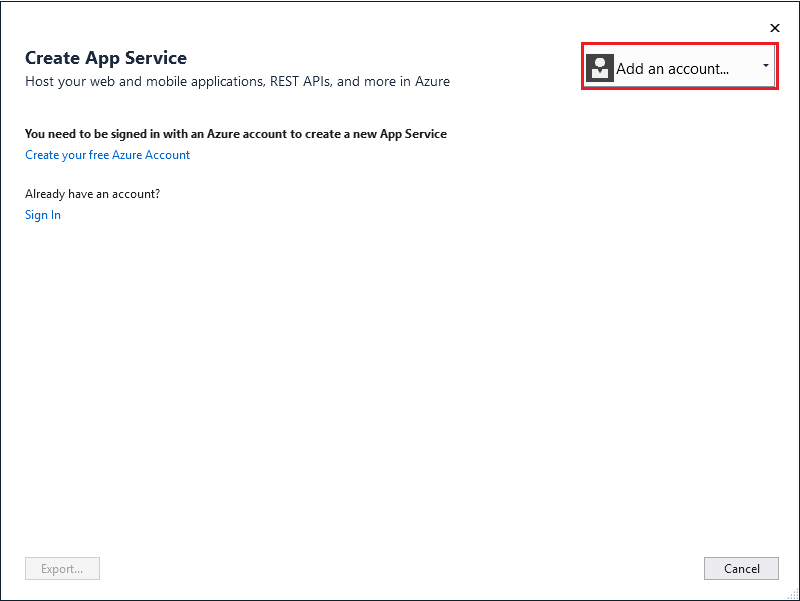
2. Выберите **Служба приложений Microsoft Azure (App Service)** и нажмите кнопку **Опубликовать**.

Во время публикации откроется диалоговое окно **Создать службу приложений**, с помощью которого вы можете создать все ресурсы Azure, необходимые для запуска приложения ASP.NET в Azure.

### Шаг 2. Вход в Azure

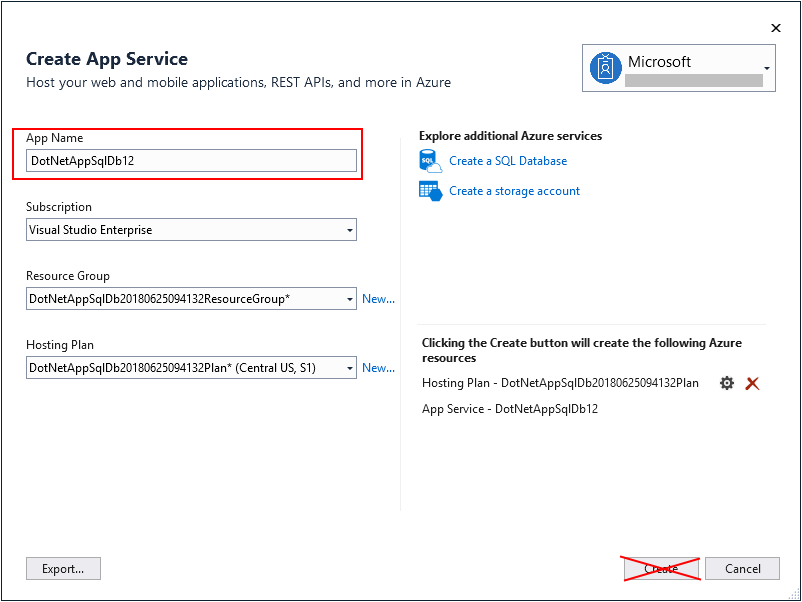
В диалоговом окне **Создать службу приложений** щелкните **Добавить новую учетную запись**, а затем выполните вход в подписку Azure. Если вы уже вошли в учетную запись Майкрософт, проверьте, содержит ли она подписку Azure. Если подписки нет, щелкните ее, чтобы добавить правильную учетную запись.

Примечание Если вы уже выполнили вход, пока не нажимайте кнопку **Создать**.



### Шаг 3. Настройка имени веб-приложения

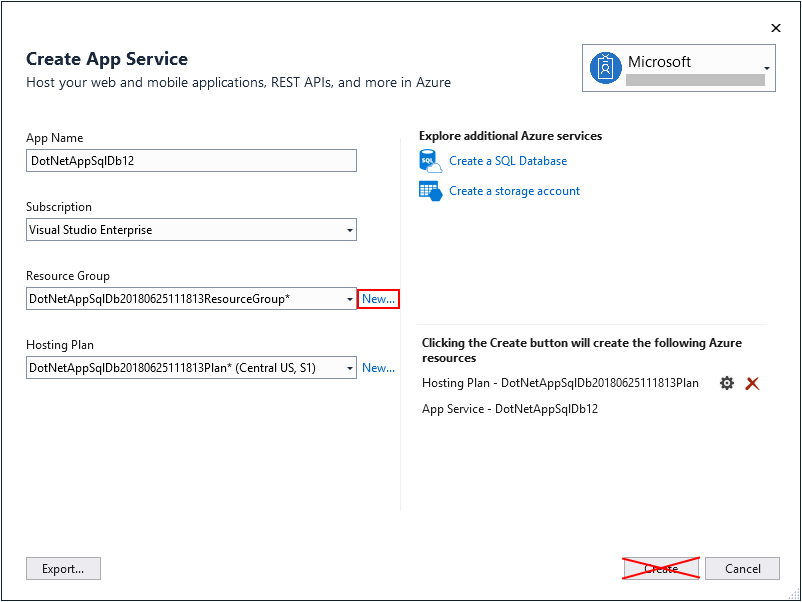
Вы можете использовать созданное имя веб-приложения или присвоить ему уникальное имя (допустимые символы: a-z, 0-9 и -). Это имя используется как часть URL-адреса по умолчанию для приложения (<app\_name>.azurewebsites.net, где <app\_name> — имя вашего веб-приложения). Оно должно быть глобально уникальным среди всех приложений Azure. См. замечания по назначению имен указанные выше в начале раздела Порядок выполнения.



### Шаг 4. Создание группы ресурсов

[Группа ресурсов](https://docs.microsoft.com/ru-ru/azure/azure-resource-manager/resource-group-overview#terminology) — это логический контейнер, в котором происходит развертывание ресурсов Azure (веб-приложений, баз данных и учетных записей хранения) и управление ими. Например, в дальнейшем можно удалить всю группу ресурсов при помощи одного простого действия.

Рядом с **группой ресурсов** щелкните **Создать**.



Присвойте группе ресурсов имя **myResourceGroup**.

### Шаг 5. Создание плана службы приложений

[План службы приложений](https://docs.microsoft.com/ru-ru/azure/app-service/overview-hosting-plans) указывает расположение, размер и функции фермы веб-серверов, в которой размещается приложение. Вы можете сэкономить средства при размещении нескольких приложений, настроив совместное использование одного плана службы приложений в веб-приложениях.

Планы службы приложений определяют такие компоненты:

Регион (например Северная Европа, восточная часть США, Юго-Восточная Азия)

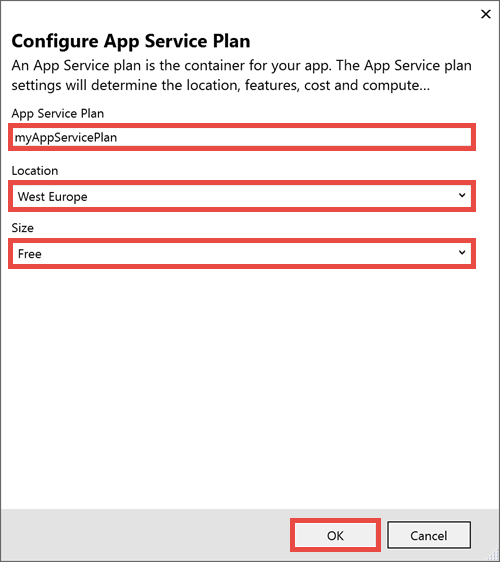
Размер экземпляра (небольшой, средний, крупный)

Число экземпляров при масштабировании (от 1 до 20)

SKU ("Бесплатный", "Общий", "Базовый", "Стандартный" или "Премиум").

1. Рядом с **планом службы приложений** щелкните **Создать**.

В диалоговом окне **Настроить план службы приложений** настройте новый план службы приложений, задав приведенные ниже параметры.

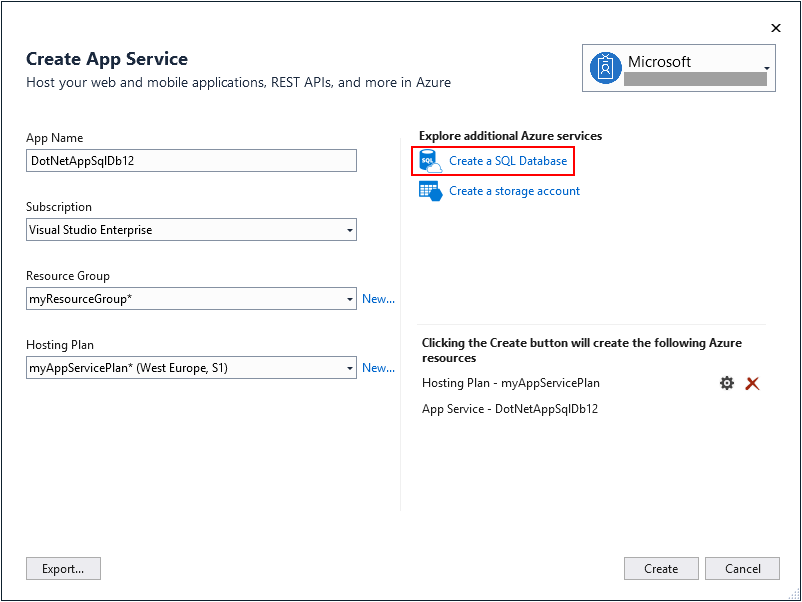


| **Параметр** | **Рекомендуемое значение** | **Дополнительные сведения** |
| --- | --- | --- |
| **План службы приложений** | myAppServicePlan | [Планы службы приложений](https://docs.microsoft.com/ru-ru/azure/app-service/overview-hosting-plans) |
| **Местоположение.** | Западная Европа | [Регионы Azure](https://azure.microsoft.com/regions/?ref=microsoft.com&utm_source=microsoft.com&utm_medium=docs&utm_campaign=visualstudio) |
| **Размер** | Free | [Ценовые категории](https://azure.microsoft.com/pricing/details/app-service/?ref=microsoft.com&utm_source=microsoft.com&utm_medium=docs&utm_campaign=visualstudio) |

### Шаг 6. Создание экземпляра SQL Server

Перед созданием базы данных необходимо сначала создать [логический сервер базы данных SQL Azure](https://docs.microsoft.com/ru-ru/azure/sql-database/sql-database-features). Логический сервер содержит группу баз данных, которыми можно управлять как группой.

1. Щелкните **Создать базу данных SQL**.



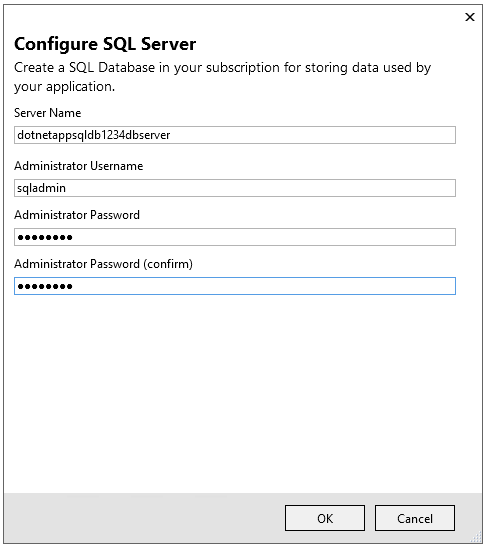
2. В окне **Настройка базы данных SQL** нажмите кнопку **Создать** рядом с **SQL Server**.

Создается уникальное имя сервера. Это имя используется как часть URL-адрес по умолчанию для логического сервера (<server\_name>.database.windows.net). Оно должно быть уникальным для всех экземпляров логических серверов в Azure. Имя сервера можно изменить, но в рамках этого руководства используйте созданное значение.

3. Добавьте имя пользователя и пароль администратора. Требования к сложности пароля см. в статье [Политика паролей](https://docs.microsoft.com/ru-ru/sql/relational-databases/security/password-policy).

Запомните это имя пользователя и пароль. Они потребуются позже для управления экземпляром логического сервера.

Важно! Хотя пароль в строке подключения скрывается (в Visual Studio и в службе приложений), он все равно где-то сохраняется, что делает приложение уязвимым перед атаками. В таком случае служба приложений может использовать [управляемые удостоверения службы](https://docs.microsoft.com/ru-ru/azure/app-service/overview-managed-identity), полностью устраняя необходимость хранить секреты в коде или конфигурации приложения.



4. Нажмите **ОК**. Но диалоговое окно **Настроить базу данных SQL** пока не закрывайте.

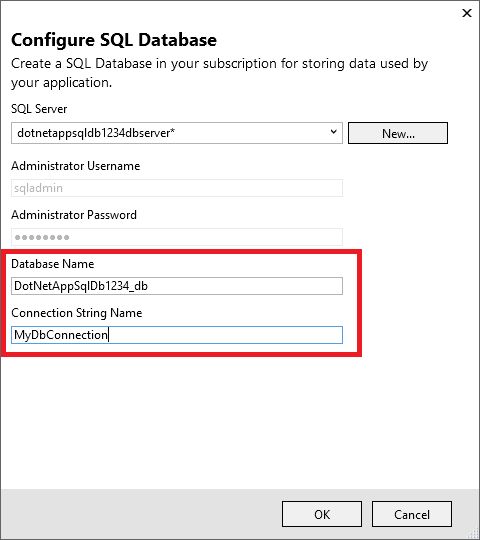
### Шаг 7. Создание базы данных SQL

В диалоговом окне **Настроить базу данных SQL выполнить**:

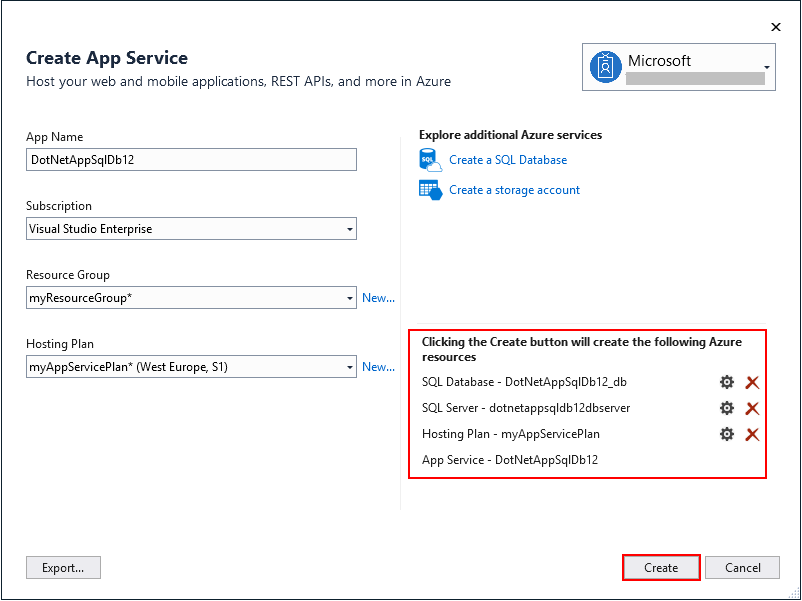
1. В поле **Имя базы данных** не изменяйте созданное по умолчанию имя.

2. В поле **Имя строки подключения** введите *MyDbConnection*. Это имя должно совпадать с именем строки подключения, указанном в *Models\MyDatabaseContext.cs*.

3. Нажмите кнопку **ОК**.



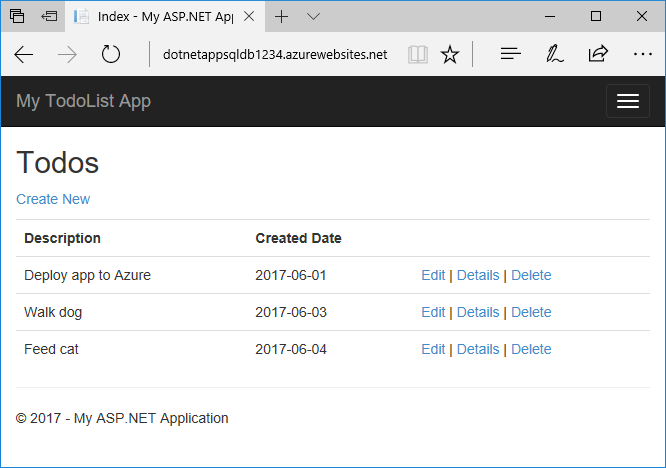
4. В диалоговом окне **Создать службу приложений** отобразятся настроенные ресурсы. Нажмите кнопку **Создать**.



Когда мастер завершит создание ресурсов Azure, он опубликует приложение ASP.NET в Azure. Откроется браузер по умолчанию с URL-адресом развернутого приложения.

Шаг 7. Проверка работоспособности развернутого приложения.

1. Добавьте несколько элементов списка дел.



Результат. Вы запустили управляемое данными приложение ASP.NET в службе приложений Azure.

## Задание 2. Создание доступа к базе данных SQL развернутой в Azure c локальной машины

В **обозревателе объектов SQL Server** Visual Studio вы сможете легко просматривать свою базу данных SQL и управлять ею.

### Шаг 1. Создание подключения к базе данных

В меню **Представление** выберите **Обозреватель объектов SQL Server**.

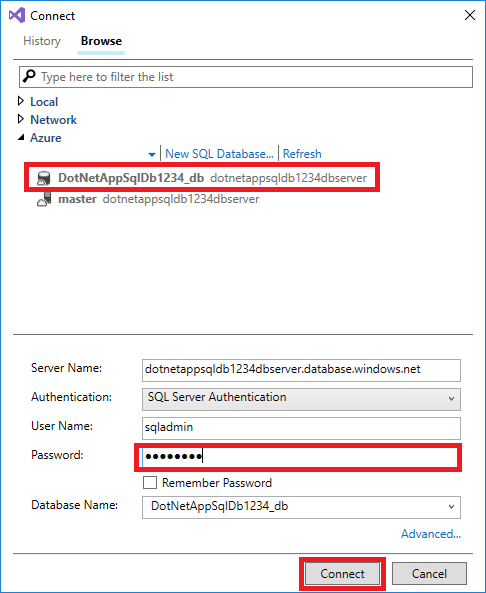
1. В верхней части **обозревателя объектов SQL Server** нажмите кнопку **Добавить SQL Server**.

### Шаг 2. Настройка подключения к базе данных

1. В диалоговом окне **Подключение** разверните узел **Azure**. Здесь перечислены все экземпляры базы данных SQL в Azure.

2. Выберите ранее созданную базу данных SQL. В нижней части автоматически появится созданное ранее подключение.

3. Введите созданный ранее пароль администратора базы данных и нажмите кнопку **Подключиться**.

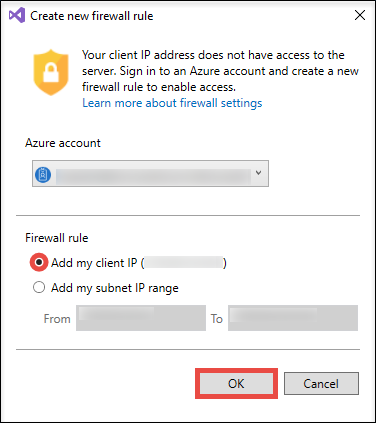


### Шаг 3. Разрешение клиентских подключений с локального компьютера

Откроется диалоговое окно **Создать новое правило брандмауэра**. По умолчанию к экземпляру Базы данных SQL могут подключаться только службы Azure, такие как ваше приложение Azure. Чтобы подключиться к базе данных, создайте правило брандмауэра в экземпляре базы данных SQL. Это правило разрешает подключения с общедоступного IP-адреса вашего локального компьютера.

В диалоговом окне уже указан общедоступный IP-адрес компьютера.

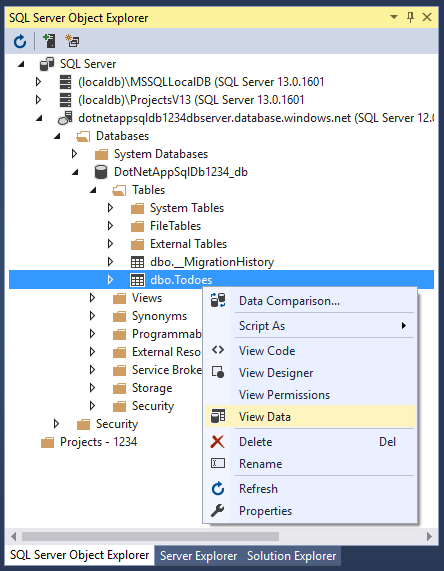
1. Убедитесь, что флажок **Добавить IP-адрес моего клиента** установлен, и щелкните **ОК**.



После окончания настройки брандмауэра для экземпляра базы данных SQL ваше подключение появится в **обозревателе объектов SQL Server**.

В нем вы можете выполнять большинство распространенных операций с базой данных: выполнять запросы, создавать представления и хранимые процедуры и многое другое.

2. Разверните узел подключения, выберите **Базы данных** > **<ваша база данных >** > **Таблицы**. Щелкните таблицу Todoes правой кнопкой мыши и выберите **Просмотреть данные**.



## Задание 3. Изменение приложения с помощью Code First Migrations

Обновить базу данных и приложение в Azure можно с помощью знакомых инструментов Visual Studio. На этом шаге вы измените схему базы данных с помощью Code First Migrations в Entity Framework и опубликуете ее в Azure.

Дополнительные сведения об использовании Entity Framework Code First Migrations см. в статье [Getting Started with Entity Framework 6 Code First using MVC 5](https://docs.microsoft.com/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application) (Начало работы с Entity Framework 6 Code First с помощью MVC 5).

### Шаг 1. Обновление модели данных

Откройте файл *Models\Todo.cs* в редакторе кода. Добавьте в класс ToDo следующее свойство:

C#

public bool Done { get; set; }

### Шаг 2. Локальный запуск Code First Migrations

Выполните несколько команд, чтобы обновить локальную базу данных.

1. В меню **Инструменты** выберите **Диспетчер пакетов NuGet** > **Консоль диспетчера пакетов**.

2. В окне консоли диспетчера пакетов включите Code First Migrations:

PowerShell

Enable-Migrations

3. Добавьте миграцию:

PowerShell

Add-Migration AddProperty

4. Обновите локальную базу данных:

PowerShell

Update-Database

5. Введите Ctrl+F5, чтобы запустить приложение. Проверьте ссылки "Изменить", "Сведения" и "Создать".

Если приложение загружается без ошибок, то Code First Migrations успешно включен. Однако ваша страница не изменилась, так как новое свойство все еще не используется в логике приложения.

### Шаг 3. Использование нового свойства

Внесите некоторые изменения в код, чтобы использовалось свойство Done. Для простоты мы изменим только представления Index и Create, чтобы просмотреть свойство в действии.

1. Откройте файл *Controllers\TodosController.cs*.

2. Найдите метод Create() в строке 52 и добавьте Done в список свойств атрибута Bind. Когда все будет готово, сигнатура метода Create() должна выглядеть следующим образом:

C#

public ActionResult Create([Bind(Include = "Description,CreatedDate,Done")] Todo todo)

3. Откройте файл *Views\Todos\Create.cshtml*.

В коде Razor вы должны увидеть элемент <div class="form-group">, который использует model.Description, и еще один элемент <div class="form-group">, который использует model.CreatedDate. Сразу после этих двух элементов добавьте еще один элемент <div class="form-group">, который использует model.Done:

C#

<div class="form-group">

@Html.LabelFor(model => model.Done, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

<div class="checkbox">

@Html.EditorFor(model => model.Done)

@Html.ValidationMessageFor(model => model.Done, "", new { @class = "text-danger" })

</div>

</div>

</div>

4. Откройте файл *Views\Todos\Index.cshtml*.

Найдите пустой элемент <th></th>. Добавьте следующий код Razor над этим элементом:

C#

<th>

@Html.DisplayNameFor(model => model.Done)

</th>

Найдите элемент <td>, который содержит вспомогательные методы Html.ActionLink(). *Над* элементом <td> добавьте еще один элемент <td> со следующим кодом Razor:

C#

<td>

@Html.DisplayFor(modelItem => item.Done)

</td>

Это все, что нужно сделать, чтобы увидеть изменения в представлениях Index и Create.

5. Введите Ctrl+F5, чтобы запустить приложение.

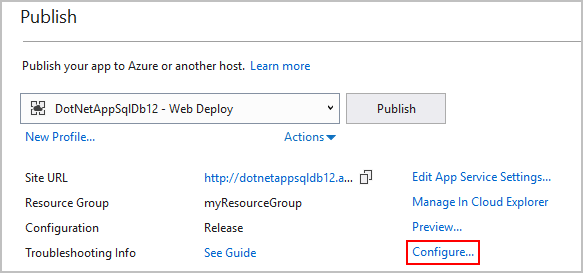
Теперь вы сможете добавить элемент списка дел и установить флажок **Готово**. После этого задание должно появиться на главной странице как выполненное. Помните, что в представлении Edit не отображается поле Done, так как вы не изменили представление Edit.

### Шаг 4. Включение Code First Migrations в Azure

Теперь, когда изменения в коде, включая миграцию базы данных, выполнены успешно, можно опубликовать изменения в приложение Azure, а также обновить Базу данных SQL для использования Code First Migrations.

1. Как и прежде, щелкните правой кнопкой мыши свой проект и выберите **Опубликовать**.

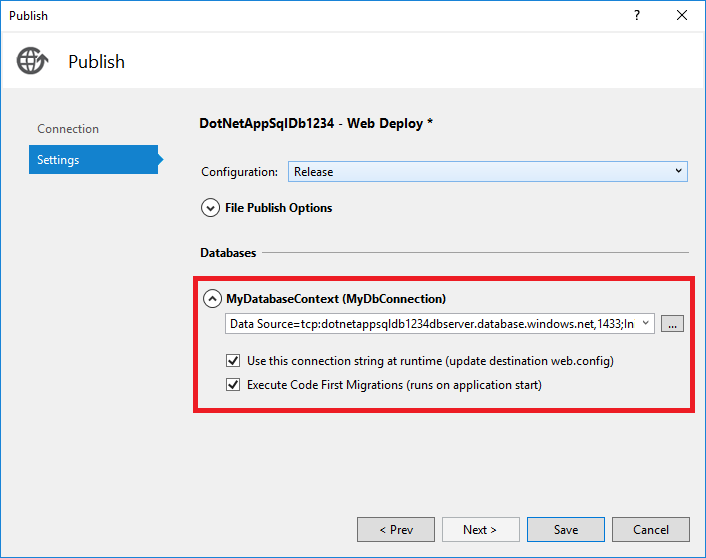
2. Щелкните **Настройка**, чтобы открыть параметры публикации.



3. В мастере нажмите кнопку **Далее**.

4. Убедитесь, что строка подключения для базы данных SQL в контексте **MyDatabaseContext (MyDbConnection)** заполнена. Возможно, вам потребуется выбрать базу данных **myToDoAppDb** из раскрывающегося списка.

5. Установите флажок **Выполнять Code First Migrations (при запуске приложения)** и нажмите кнопку **Сохранить**.

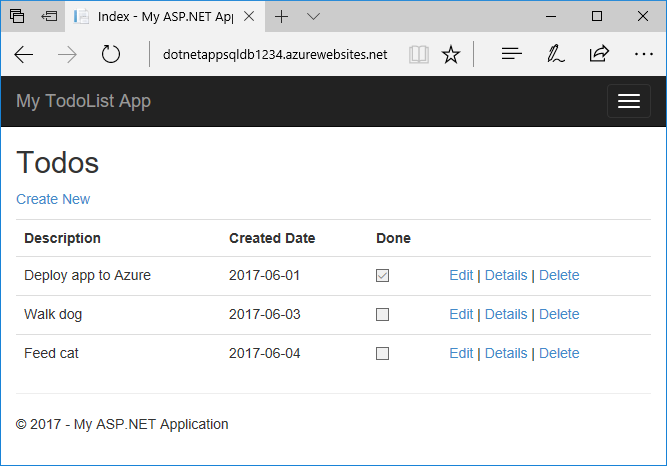


### Шаг 5. Публикация изменений

Включив Code First Migrations в приложении Azure, опубликуйте изменения кода.

1. На странице публикации щелкните **Опубликовать**.

2. Попробуйте добавить новые задачи, устанавливая флажок **Готово**. Эти задачи должны появляться на главной странице как выполненные.



Все имеющиеся элементы списка дел по-прежнему отображаются. При повторной публикации приложения ASP.NET существующие данные в базе данных SQL не теряются. Кроме того, Code First Migrations изменяет только схему данных, оставляя существующие данные нетронутыми.

## Задание 4. Настройка потоковой передачи журналов приложений

Сообщения трассировки можно передавать прямо из приложения Azure в Visual Studio.

Откройте файл *Controllers\TodosController.cs*.

Каждое действие начинается с метода Trace.WriteLine(). Этот код добавлен, чтобы показать, как добавить сообщения трассировки в приложение Azure.

### Шаг 1. Откройте обозреватель сервера

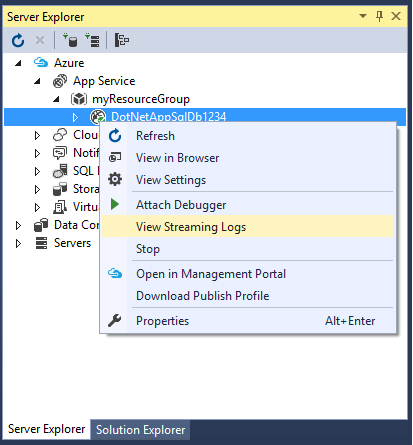
1. В меню **Представление** выберите **Обозреватель серверов**. Ведение журнала для приложения Azure можно включить в **обозревателе сервера**.

### Шаг 2. Включение потоковой передачи журналов

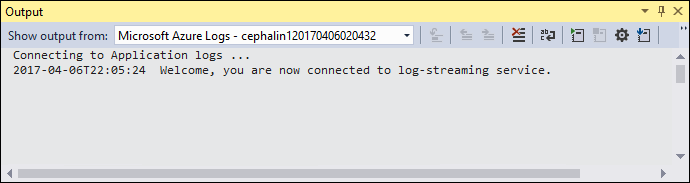
1. В **обозревателе сервера** выберите **Azure** > **Служба приложений**.

2. Разверните группу ресурсов **myResourceGroup**, которая была создана при создании приложения Azure.

3. Щелкните приложение правой кнопкой мыши и выберите **Просмотреть журналы потоковой передачи**.



Теперь журналы передаются в окно **Выходные данные**.



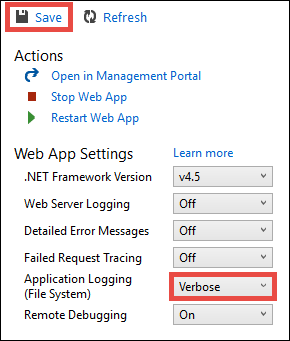
Однако сообщения трассировки пока не отображаются. Это объясняется тем, что когда вы в первый раз выбираете пункт меню **Просмотреть журналы потоковой передачи**, приложение устанавливает уровень трассировки в Error, при котором в журнал записываются только события ошибок (с помощью метода Trace.TraceError()).

### Шаг 3. Изменение уровней трассировки

Чтобы изменить уровень трассировки для вывода других сообщений трассировки, вернитесь в **обозреватель сервера**.

1. Снова щелкните приложение правой кнопкой мыши и выберите **Просмотр параметров**.

2. В раскрывающемся списке **Ведение журнала приложения (файловая система)** выберите **Подробно**. Выберите команду **Сохранить**.



Совет Вы можете поэкспериментировать с различными уровнями трассировки, чтобы посмотреть, какие типы сообщений отображаются для каждого уровня. Например, уровень **Информация** включает все журналы, созданные Trace.TraceInformation(), Trace.TraceWarning(), и Trace.TraceError(), но не включает журналы, созданные Trace.WriteLine().

3. В браузере снова перейдите к приложению по адресу *http://<имя вашего приложения>.azurewebsites.net*, а затем щелкните правой кнопкой приложение списка дел в Azure. Теперь сообщения трассировки передаются в окно **Выходные данные** в Visual Studio.

console

Application: 2017-04-06T23:30:41 PID[8132] Verbose GET /Todos/Index

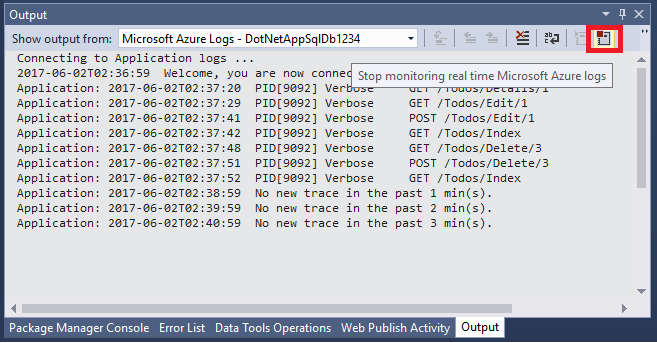
Application: 2017-04-06T23:30:43 PID[8132] Verbose GET /Todos/Create

Application: 2017-04-06T23:30:53 PID[8132] Verbose POST /Todos/Create

Application: 2017-04-06T23:30:54 PID[8132] Verbose GET /Todos/Index

### Шаг 4. Выключение потоковой передачи журналов

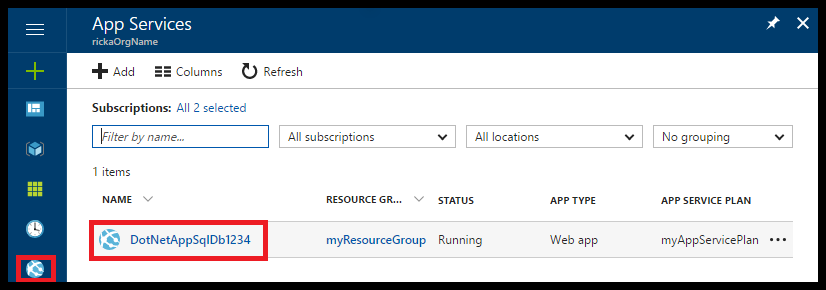
1. Чтобы остановить службу потоковой передачи журналов, нажмите кнопку **Остановить наблюдение** в окне **Выходные данные**.



## Задание 5. Управление приложением Azure

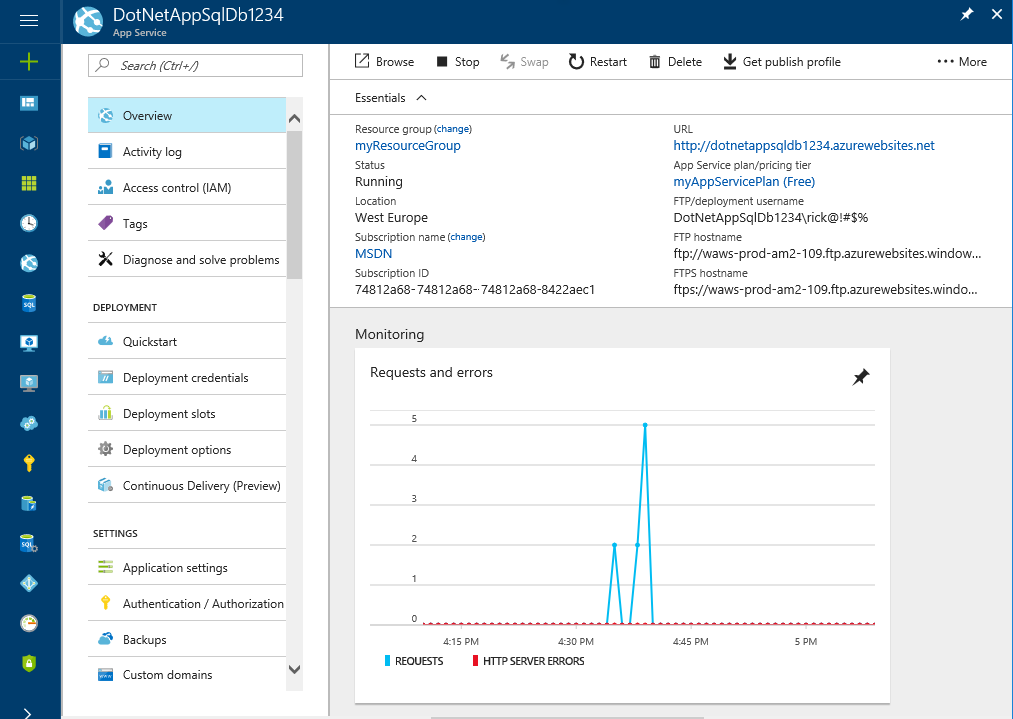
### Шаг 1. Перейдите на [портал Azure](https://portal.azure.com), чтобы увидеть созданное приложение.

В меню слева выберите **Служба приложений**, а затем щелкните имя своего приложения Azure.



Вы попадете на страницу приложения.

По умолчанию на портале отображается страница **Обзор**. Здесь вы можете наблюдать за работой приложения. Вы также можете выполнять базовые задачи управления: обзор, завершение, запуск, перезагрузку и удаление. На вкладках в левой части страницы отображаются различные страницы конфигурации, которые можно открыть.



## Задание 6. Очистка ресурсов

### Шаг 1. Выбрать группу ресурсов

На предыдущем шаге вы создали ресурсы Azure в группе ресурсов. Если вы считаете, что в будущем эти ресурсы вам не понадобятся, их можно удалить, удалив группу ресурсов.

1. На странице **Обзор** веб-приложения на портале Azure щелкните ссылку **myResourceGroup** в разделе **Группы ресурсов**.

### Шаг 2. Удаление ресурсов

1. На странице группы ресурсов убедитесь, что перечислены ресурсы, которые нужно удалить.

2. Щелкните **Удалить**, введите в текстовое поле **myResourceGroup** и снова щелкните **Удалить**.

## Заключение

В процессе выполнения данной лабораторной работы вы узнали, как выполнять следующие задачи:

Создание базы данных SQL в Azure.

Подключение приложения ASP.NET к базе данных SQL.

Развертывание приложения в Azure

Обновление модели данных и повторное развертывание приложения.

Потоковая передача журналов из Azure в окно терминала.

Управление приложением на портале Azure.

## Задание 6. Подготовить отчет о выполнении лабораторной работы

В отчете необходимо кратко описать все выполненные действия, привести скриншоты с результатами выполнения каждого из заданий. В отчет также должны быть включены ответы на контрольные вопросы.

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Какие возможности предоставляет служба облачного сервиса в MS Azure.

2. Что такое группа ресурсов в MS Azure.

3. Приведите понятие ресурсов для развертывания облачного сервиса в MS Azure и дайте характеристику плана ресурсов.

4. Какие возможности предоставляет служба "База данных SQL" Azure.