Kentico 8.1

Running Kentico on Microsoft Azure

Microsoft Azure is a cloud platform for hosting and managing applications and services. This section provides information on the specifics of developing Kentico on Microsoft Azure and offers instructions on the installation, configuration and deployment of Microsoft Azure web projects.

Basic information about running Kentico in Microsoft Azure environment

- Architecture of Microsoft Azure environment
- Requirements and limitations for running Kentico on Microsoft Azure
- Microsoft Azure project development lifecycle

Installation and deployment

To install and deploy a new Azure project:

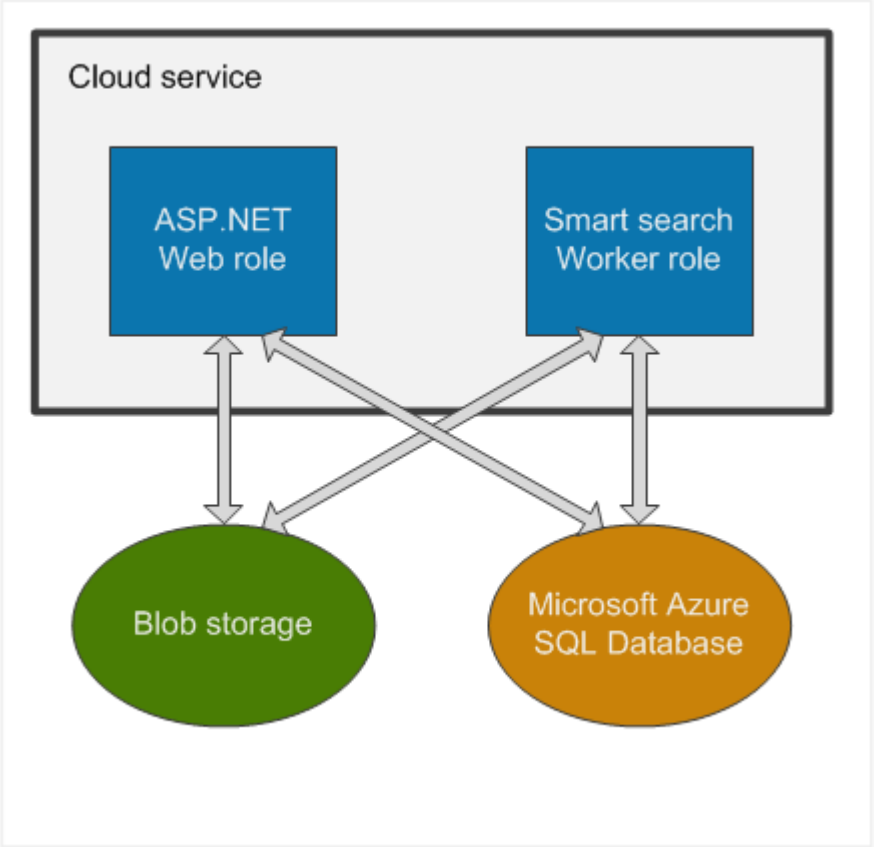
1. Prepare the cloud environment
2. Install Kentico as an Azure project
3. Configure the Azure project
4. Deploy the project to the cloud environment
5. Install the database for your project

Additional information

- Developing Azure projects locally - learn how to set up Azure projects to prepare them for the development on your local machines.
- Microsoft Azure Web Sites - Microsoft Azure Web Sites is an alternative option to the mainly used Cloud Services environment offering easier set up, but fewer customization options.
- Configuring Azure storage - in case you do not need to host your whole application in the cloud, you can utilize this hybrid scenario, which uses Microsoft Azure Storage as a file storage for your project.
- Azure web.config settings - see this reference for a list of available web.config setting keys for your Azure project.

Architecture of Microsoft Azure environment

This topic describes how Kentico works within the Microsoft Azure environment using Azure Cloud Services, which features of the service are utilized and how the application stores and manages its data in the cloud.



Application

If you choose to install Kentico to Microsoft Azure, all files will be grouped into a solution based on Visual Studio's Microsoft Azure template. The solution contains several projects. One of them is a web application, which encompasses almost all the functions of Kentico and is designed to run as an Azure **ASP.NET Web role**.

The **Smart search** worker, is separated from the web application in another project because it cannot run together with the application as the Web role. To index content of websites correctly and effectively, the Smart search worker runs as an Azure **Worker role**.

Because the application is divided into these two services, you also need to configure them separately. See Configuring an Azure project.

Database

Kentico on Microsoft Azure uses an **Azure SQL** relational database. This database engine is almost identical to the standard SQL Server engine, with only a few limitations. These limitations are taken into account in Kentico, and no additional configuration or customization is required. If you're interested in which SQL Server features are not available in Azure SQL, refer to SQL Server Feature Limitations (Azure SQL Database) on MSDN.

File storage

Microsoft Azure does not offer a persistent file system similar to the file systems in Windows that you are used to. Data stored within Azure cannot be hierarchically organized into folders. However, Kentico provides an abstract layer called **CMS.IO**, which enables the system to operate on different types of file storages. See Working with physical files using the API for more information.

The CMS.IO namespace acts as an intermediary between the Kentico business layer and various file storages, including Azure **blob storage** . On a standard non-Azure installation, CMS.IO only overrides the System.IO namespace. On Microsoft Azure, the namespace uses a provider which works with the blob storage, creating an imitation of the regular Windows file system. The CMS.IO namespace can be extended to support any other type of storage, e.g. the Amazon cloud drive.

Additionally, you can make use of the Azure storage provider and store files in the cloud even if you're running a non-Azure installation. You can find more information about this approach in Configuring Azure storage.

The file storage is shared across multiple web role instances, therefore, no file synchronization is needed.

> **Case sensitivity**
>
> Unlike Windows file systems, the blob storage is case-sensitive. As a result, names of all files processed by Kentico are **converted to lower case** before saving.

Multiple web role instances

Kentico application can run in multiple instances of one web role on Microsoft Azure. Therefore, the data must be synchronized between these instances. Kentico handles the synchronization by considering each instance a web farm server. When you set the number of instances, the web farm servers are configured automatically. Also, web farm tasks are created and executed automatically so no manual configuration is needed.

When you need to optimize cache performance, you can create a Cache service on the Azure Management Portal and configure it to synchronize cache. See Storing session state information in Azure Cache Service for details.

Storing session state

Every complex web application needs to store information about its state, especially user session data. Since the Azure environment is dynamic and the application does not reside constantly in one place, its state has to be stored separately. When your application uses only one web role instance, the session state data can be stored on that instance. If your application uses two or more web role instances, you need to configure your project to store the session state data elsewhere. We recommend these options for storing the session state data:

- In Microsoft Azure SQL Database - easy to set up, suitable for small projects or projects with mostly read access to web pages – see Storing session state information in Azure SQL database.
- In Microsoft Azure Cache - more suitable for larger projects than the Azure SQL database option, but requires configuration of Azure Cache – see Storing session state information in Azure Cache Service.

Alternative approaches

Kentico+

If you are looking for a complete solution, where you do not need to configure your project and manage the hosting cloud environment, you can use our Kentico+ platform. See Kentico+ documentation for more information.

Microsoft Azure Web Sites

Web Sites are quick to deploy and easy to set up and manage. However, they do not offer as much customization options as cloud services. For a more detailed comparison, see Azure Web Sites, Cloud Services and Virtual Machines comparison on MSDN.

See Microsoft Azure Web Sites for information on how to set up Web Sites.

Requirements and limitations for running Kentico on Microsoft Azure

If you wish to run Kentico in the Microsoft Azure environment, make sure that you are familiar with the requirements.

Development tools

Install the following software on the machines where you plan to develop your Azure project:

- Microsoft Visual Studio 2012/2013 (including Microsoft Visual Studio 2012/2013 Express Edition)
- Microsoft .NET Framework 4.5/4.5.1 (it will be installed automatically on your computer during the installation)
- **Azure SDK 2.2** including Microsoft Azure Tools for the version of Microsoft Visual Studio you are using
- Microsoft Internet Information Services (IIS) 7.0 or newer - required if you wish to run the application locally on the Microsoft Azure Compute Emulator.

You can download the Azure SDKs with Tools for Visual Studio from http://azure.microsoft.com/en-us/downloads/archive-net-downloads/.

Microsoft Azure requirements

The exact hosting requirements depend on the resource consumption and traffic load of your site and its components. You need to prepare the following services for your Azure subscription:

- **Azure cloud service** - for production (live) deployments of Kentico websites, it is highly recommended to use at least the **Small** Compute Instance Size.
- **Azure storage account**
- **Azure SQL server and database** - required if you wish to host your application database on Microsoft Azure. The smallest available database (1GB) is sufficient for the default Kentico installation.

Kentico licensing

You need a Website license (CMS or EMS) with web farm server licenses for the number of Microsoft Azure instances you are using. The number of required web farm licenses depends on the number of instances, not the number of actual physical servers involved. We recommend that you use two instances to qualify for Microsoft Cloud service SLA.

For licensing information, refer to Cloud licensing and Price list.

Limitations

If the application database is hosted on Azure SQL Database (formerly known as SQL Azure), it is important to keep in mind that certain statements and expressions are restricted when writing queries (e.g., in custom code, web part properties or for reporting objects). For a comprehensive list of all SQL Database features and requirements, see Azure SQL Database Guidelines and Limitations on MSDN.

A shortcoming of any application running on Microsoft Azure is the need to redeploy (or at least update your deployment) before any changes made to the project structure or code are reflected. This may make it more difficult and time consuming to perform website maintenance, customization and some development tasks. However, you can use the Web deploy functionality for development and testing purposes.

In addition, the following features of Kentico are currently limited:

- **Full site import -** importing sites is possible, but files stored in the web project, such as custom user controls, ASPX page templates or design files, will not be included in the import process. To work around this limitation, you can copy such files into the target Azure project and include them in the solution before you deploy the application to Microsoft Azure.
- **CSS Theme tabs** - management of files contained in stylesheet Theme folders directly through the Kentico administration interface is not available when running on Microsoft Azure.
- **JavaScript** minification does not work on Microsoft Azure.

Microsoft Azure project development lifecycle

When you develop projects on Microsoft Azure, in typical cases you want to begin with a small size of the project, which uses the least resources possible. Then, as your project grows, you configure your project to utilize more resources to accommodate the performance and size requirements of the project. This topic presents main **levels of development** on Microsoft Azure and provides links to related configuration tasks, which you need to perform when ascending to a higher level.

Level 0 - Local development

You can choose to begin developing your Azure project locally in an emulator before deploying it to the cloud environment. In this case, set up a database and Azure Storage service and configure the web role's settings. See Developing Azure projects locally.

Level 1 - Development

For the duration of project development, **one instance** of CMSApp web role is usually enough. To configure a project to use one web role instance, perform the basic configuration tasks.

Level 2 - Production with SLA

When you deploy your project and switch to the production environment, you may want to qualify for Microsoft SLA. In such case, your project must use at least **two instances** of the CMSApp web role. When you increase the number of used web role instances, you need to adjust your project to synchronize the data between the instances and to store session state information.

For data synchronization, you can use the default **web farm synchronization** tasks, in which case you do not need to configure anything.

For storing session state information, you can use **Microsoft Azure SQL Database** – see Storing session state information in Azure SQL Database.

Level 3 - Performance

When the performance of the level 2 environment is not sufficient, you can configure the **Azure Cache Service** to store session state information – see Storing session state information in Azure Cache Service.

Level 4 - Scalability

When you need even more power, you can further scale your project using the following approaches:

- Utilize larger cloud services
- Use more web role instances
- Configure autoscaling

Preparing the cloud environment

Before you start installing a Kentico project for Microsoft Azure, it can be convenient to configure the cloud environment using the Microsoft Azure Management Portal and create parts of the system you will need. It is not necessary to create all the parts in advance, but doing so all at once can save you time later.

To run a Kentico project on Microsoft Azure, you will need to create:

- Azure Storage
- Azure Cloud Service

An Azure project can use either a standard hosted SQL Server or a cloud server. If you want to use the database service on Microsoft Azure

platform, create also:

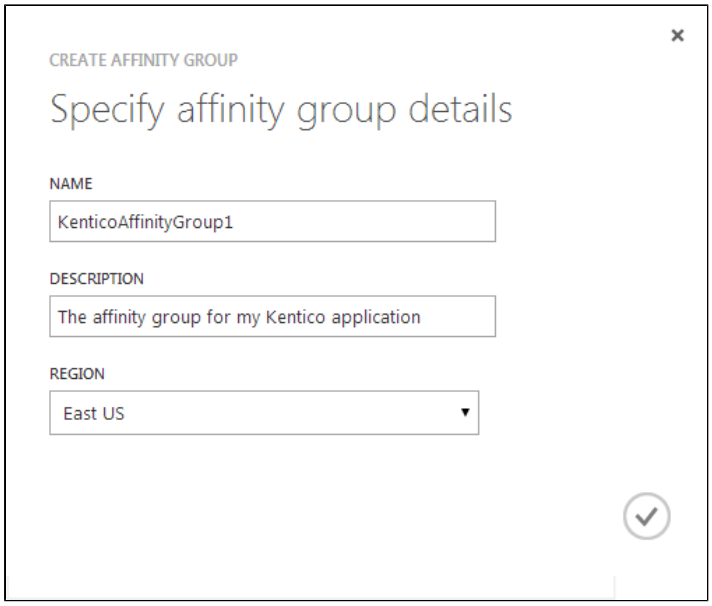
- Azure SQL Server
- Azure SQL Database

Service location and affinity groups

We recommend that you configure an **affinity group** and use it for all data and cloud services you create. An affinity group ensures that the services assigned to it will be located in one datacenter and even close together within the particular datacenter.

To create an affinity group:

1. Click **Settings** in the left toolbar.
2. Select the **Affinity groups** tab.
3. Click **Add**.
4. Type the **Name** and **Description** for the affinity group.
5. Select the ideal **Region**.



6. Finish the creation.

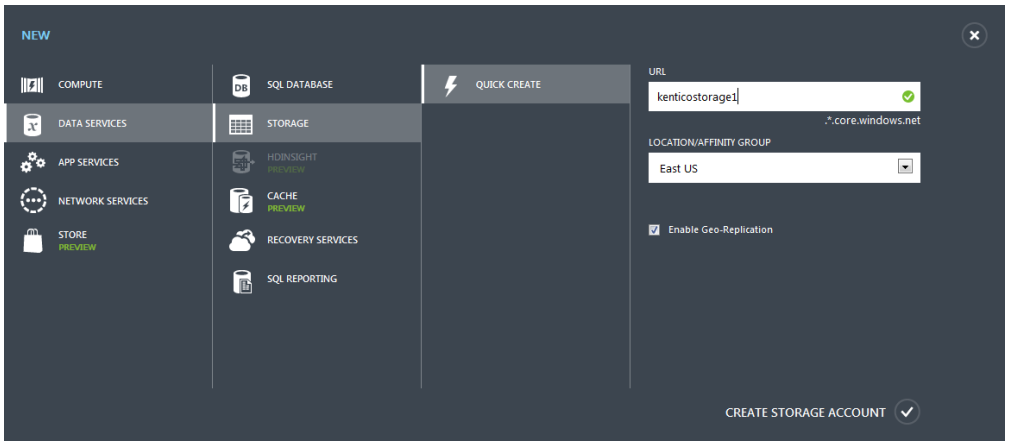
When creating new services, you can now assign them to the same affinity group to ensure their best performance.

> Alternatively, you can also select the **same location** (for example, East US) for all created services.

Creating a Microsoft Azure Storage

The Microsoft Azure Storage account will serve as a file system for your cloud based project. It provides access to the blob, queue and table services.

1. Click **New -> Data services -> Storage -> Quick create**.
2. Type a URL (name) of your new Azure storage, which is not currently in use.
3. Select the location for your storage. Use an affinity group or the same location for all created services.



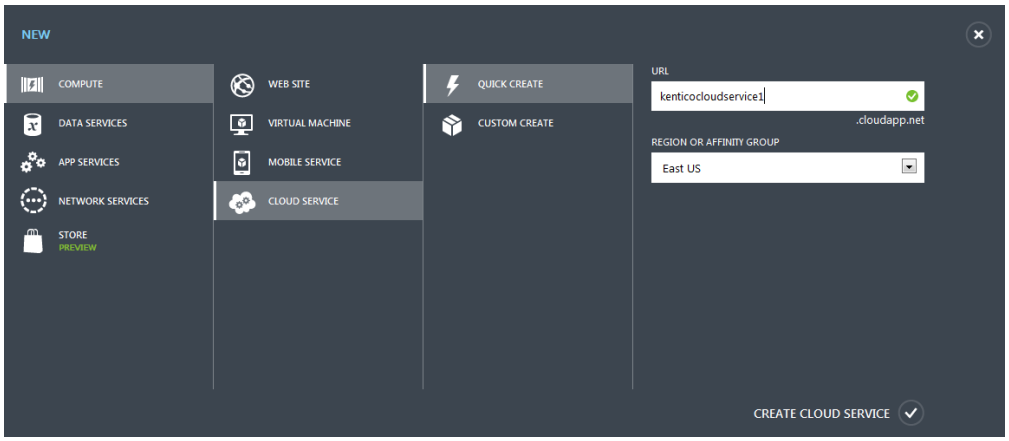
4. Click **Create storage account**.

The portal creates the storage and you can see it in the list of storages.

Creating a Microsoft Azure Cloud service

The Microsoft Azure Cloud Service provides an infrastructure for the code and configuration of your project.

1. Click **New -> Compute -> Cloud service -> Quick create**.
2. Type a URL (name) of your new Azure cloud service, which is not currently in use.
3. Select the location for your storage. Use an affinity group or the same location for all created services.



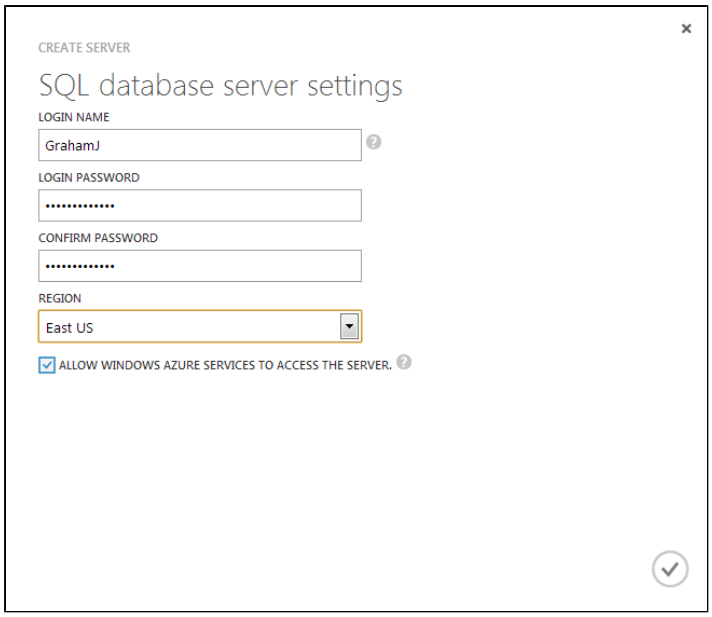
4. Click **Create cloud service**.

The portal creates the cloud service.

Creating a Microsoft Azure SQL Server

The Azure SQL Server is a virtual server you can create in the cloud. You will need the server to create databases in it.

1. Select **SQL Databases** in the panel on the left.
2. Click **Servers**.
3. Click **Add**.
4. Specify a login name and password for the server.
5. Choose a region for the server's location. Use an affinity group or the same location for all created services..
6. Ensure, that the option **Allow Windows Azure services to access the server** is selected.
    - This option adds special firewall rules to allow your deployed Kentico project to access the server and its databases.



7. Confirm the creation of the server.

The portal creates a new server. If you want to manage this server locally or use a local emulator, add firewall rules for IP ranges of your development and administration machines:

Adding firewall rules for a server

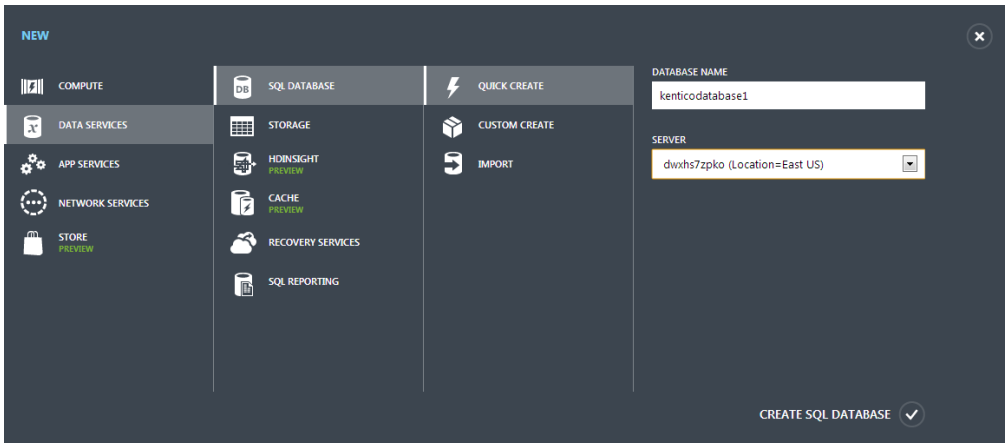
1. Select the server.
2. Switch to the **Configure** tab.
3. Click the **Add to the allowed IP addresses** link or type the name of the rule and the IP address range of your development and administration machines.
4. Click **Save** in the bottom panel.

You have configured the server to be locally accessible.

Creating a Microsoft Azure SQL Database

Create a new database on the virtual server, so that you can use it for Kentico database installation:

1. Click **New -> Data services -> SQL database -> Quick create**.
2. Type the name of the database.
3. Choose the Azure SQL database server you have already created.



4. Click **Create SQL database**.

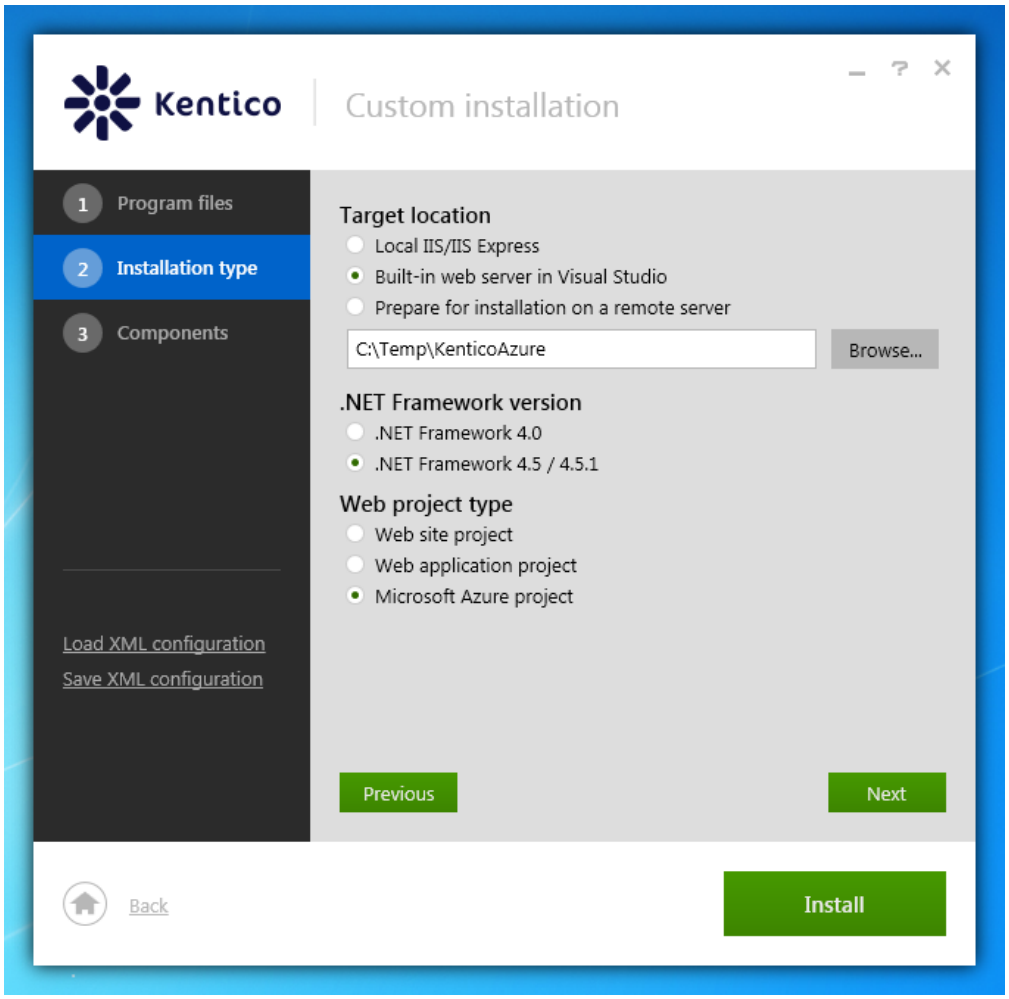
The portal creates your SQL database and you can now use it to install the Kentico database in it.

Installing an Azure project

If you want to install Kentico on a Microsoft Azure server, you must first install Kentico on your local computer:

1. Run the Kentico Installer on your **local** development computer.
2. Check the license agreement and select the **Custom installation** option.

3. Switch to the **Installation type** tab.
4. Select the **Built-in web server in Visual Studio**.
5. Choose a folder for the project files (for example, C:\Temp\KenticoAzure).
    - The files will be only copied to this location.
6. Select **.NET Framework 4.5 / 4.5.1**. Other versions are currently not supported for Microsoft Azure.
7. Select the **Microsoft Azure project** as a Web project type.



8. Click **Install**.

The Installer copies the project files into the specified folder. When the installation is finished, click the displayed link to open the web project in Visual Studio.

Proceed with Configuring an Azure project.

Configuring an Azure project

After you install an Azure project and prepare the cloud environment, you need to configure your project before the actual deployment.

- Basic configuration - perform these configuration tasks for every Azure project.
- Advanced configuration - perform these configurations when you upgrade your project to use two or more web role instances.
- Additional configurations - you can perform these configurations in any phase of your project.

Adding application settings in an Azure project

Generally, you can add settings for your Azure application either in the **web.config file** or in the **ServiceConfiguration.Cloud.cscfg** file.

However, when you need to modify setting values in the web.config file, you have to deploy the whole project again. When you need to modify setting values configured in the ServiceConfiguration.Cloud.cscfg file, you can modify them on the Azure Management Portal in **Cloud services -> Configure** tab. Therefore, we recommend that you configure your application mainly using the ServiceConfiguration.Cloud.cscfg file.

To add new settings to the configuration file:

1. Open your Azure project in Visual Studio.
2. Double-click **CMSApp** role in **CMSAzure/Roles**.
3. Switch to the **Settings** tab.
4. Click **Add Setting**.

When you add or remove settings this way, Visual Studio ensures that all necessary files (ServiceConfiguration.Cloud.cscfg, ServiceConfiguration.Local.cscfg and ServiceDefinition.csdef) are modified according to your changes.

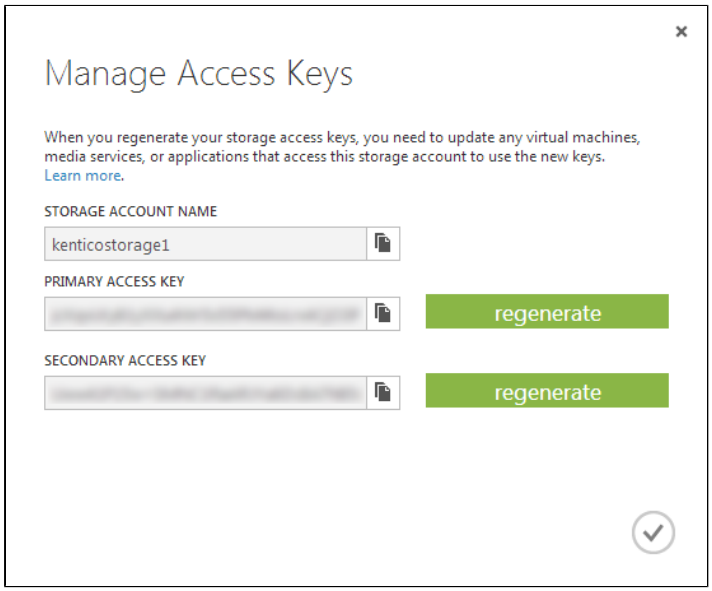
> ⚠ Whenever you add or remove settings, you have to **deploy your project** to the cloud. Therefore, you should decide in advance which functionality you need to configure in your Azure project.

Basic configuration

These configuration tasks are necessary to perform for every Microsoft Azure project.

Setting the Azure blob storage access keys

1. Open your Azure project in Visual Studio.
2. Open the **ServiceConfiguration.Cloud.cscfg** file.
3. Access the Azure Management Portal in a browser and log in.
4. Click **Storage**.
5. Select your storage.
6. Click **Manage access keys** on the bottom panel.



7. Copy the **Storage account name** and enter it as a value of **CMSAzureAccountName** setting in **CMSApp** role section in the ServiceConfiguration.Cloud.cscfg file.
8. Copy the **Primary access key** and enter it as a value of **CMSAzureSharedKey** setting in **CMSApp** role section in the ServiceConfiguration.Cloud.cscfg file.

```
<Role name="CMSApp">
 <ConfigurationSettings>
  <Setting name="CMSAzureAccountName" value="YourStorageName"/>
  <Setting name="CMSAzureSharedKey" value="YourPrimaryAccessKey"/>
 </ConfigurationSettings>
</Role>
```

   Replace **YourStorageName** and **YourPrimaryAccessKey** with your own values.

9. Save the configuration file.

You have connected your Azure project with the Azure blob storage.

> **Setting the keys after the deployment**
>
> If you do not set the storage access keys before you deploy the Azure project to the cloud, you can do it after the deployment as well in the Azure Management Portal. Navigate to **Cloud services ->** select your service **-> Configure**, where you can copy the **Storage account name** and **Primary access key** as values of the **CMSAzureAccountName** and **CMSAzureSharedKey** settings for the roles.

Configuring smart search

Before you deploy your Azure project to the cloud, you must decide whether you want to utilize the smart search functionality or not. You have the following options:

- Configure the SmartSearchWorker role (recommended)
- Remove the SmartSearchWorker role and configure processing of smart search tasks in the CMSApp role
- Remove the SmartSearchWorker role and disable smart search functionality

Configuring the SmartSearchWorker role

If you want to use the SmartSearchWorker role and utilize the smart search functionality, configure the **Storage account name** and **Primary access key** for this role.

1. Open your Azure project in Visual Studio.
2. Open the **ServiceConfiguration.Cloud.cscfg** file.
3. Copy the **CMSAzureAccountName** and **CMSAzureSharedKey** keys (which you set when configuring the storage access keys) with their values from the CMSApp role section to the **SmartSearchWorker** role section:

```
<Role name="SmartSearchWorker">
 <ConfigurationSettings>
  <Setting name="CMSAzureAccountName" value="YourStorageName"/>
  <Setting name="CMSAzureSharedKey" value="YourPrimaryAccessKey"/>
 </ConfigurationSettings>
</Role>
```

   Replace **YourStorageName** and **YourPrimaryAccessKey** with your own values.

4. **Save** the configuration file.

You have configured the SmartSearchWorker role to work on Microsoft Azure. If you do not need to perform any other configuration tasks, continue to Deploying an Azure project.

Configuring the processing of smart search tasks in CMSApp web role

If you would not utilize the whole worker role, but you do not want to lose the smart search functionality entirely, you can set that the search tasks will be processed by the main CMSApp web role. This solution can be used by small projects only, as the smart search tasks affect the performance of the web role.

1. Open the CMSAzure**CMSAzure.ccproj** file in a text editor.
2. **Delete** the following block of code:

```
<AzureRoleContent Include="$(SolutionDir)Lib\">
 <RoleName>SmartSearchWorker</RoleName>
     <Destination>
     </Destination>
</AzureRoleContent>
<AzureRoleContent Include="$(SolutionDir)Lib\Azure\">
 <RoleName>SmartSearchWorker</RoleName>
 <Destination>
 </Destination>
</AzureRoleContent>
```

3. **Save** and close the file.
4. Open your Azure project in Visual Studio.
5. Remove the **SmartSearchWorker** role from CMSAzure/Roles.
6. Open the **web.config** file from the CMSApp project.
7. Add the **CMSProcessSearchTasksByScheduler** key to the <appSettings> section:

```
<add key="CMSProcessSearchTasksByScheduler" value="true" />
```

8. **Save** the web.config file.

The smart search tasks will now be processed by the CMSApp web role. The SmartSearchWorker role will not be deployed to the hosting environment, so the costs of running your application on Microsoft Azure will be lower. If you do not need to perform any other configuration tasks, continue to Deploying an Azure project.

Disabling smart search functionality

If you are certain that you will not need the Smart search module in your project:

1. Open the CMSAzure**CMSAzure.ccproj** file in a text editor.
2. **Delete** the following block of code:

```
<AzureRoleContent Include="$(SolutionDir)Lib\">
 <RoleName>SmartSearchWorker</RoleName>
     <Destination>
     </Destination>
</AzureRoleContent>
<AzureRoleContent Include="$(SolutionDir)Lib\Azure\">
 <RoleName>SmartSearchWorker</RoleName>
 <Destination>
 </Destination>
</AzureRoleContent>
```

3. **Save** and close the file.
4. Open your **Azure project** in Visual Studio.
5. Remove the **SmartSearchWorker** role from CMSAzure/Roles.

Disabling the smart search completely reduces the number of roles that need to be hosted, so the costs of running your application on Microsoft Azure will be lower. If you do not need to perform any other configuration tasks, continue to Deploying an Azure project.

Advanced configuration (more web roles)

When you use more than one instance of the CMSApp web role, the system considers these instances as web farm servers. Therefore, you need to configure your project according to the instructions in this section.

Configuring the number of instances

You can set up the number of instances used for the **CMSApp** role, which represents the Kentico application. This determines the number of virtual machines dedicated to the website. The number of instances influences the performance and load handling capacity of the application.

To set the number of instances used for the CMSApp role, change the value of the **count** attribute of the role's *<Instances>* element:

1. Open your Azure project in Visual Studio.
2. Open the **ServiceConfiguration.Cloud.cscfg** file.
3. Change the **<Instances count="1" />** setting to the required number of instances:

```
<Role name="CMSApp">
  <Instances count="2" />
  <ConfigurationSettings>
    ...
  </ConfigurationSettings>
</Role>
```

4. Save the configuration file.

Each instance is represented by a separate web farm server within the Kentico system. The creation and management of the servers is handled automatically, and you do not have to perform any further configuration.

You can also change the number of used instances on the Azure Management Portal in **Cloud services -> Scale** tab.

SmartSearchWorker role

Do NOT increase the number of instances for the **SmartSearchWorker** role. Due to the way smart search indexes are processed, the required tasks must be performed by a single instance.

Instance licensing

The Kentico license used for your domain must allow at least as many web farm servers as the amount of instances set for the role. See http://www.kentico.com for pricing information.

Configuring cache and session state data

If you want your Azure application to use two or more web role instances, choose where to store session state information. Synchronizing data between instances is facilitated through web farm tasks. The synchronization tasks are created automatically, so not further configuration is needed in this case.

Storing session state information:

- In Microsoft Azure SQL Database - easy to set up, suitable for small projects or projects with read access to web pages.
- In Microsoft Azure Cache Service - create the Azure Cache Service and configure it to store session state information.

Storing session state information in Azure SQL Database

1. Open your Azure project in Visual Studio.
2. Right-click the **CMSApp** project and select **Manage NuGet** packages.
3. Install the **Microsoft ASP.NET Universal Providers** package.
4. Open the **web.config** file.
5. Follow the instructions in the code comments of the **sessionState** section.

After this, your project is configured to store session state information in the Microsoft Azure SQL Database.

Storing session state information in Azure Cache Service

Previously, Microsoft Azure used **AppFabric** caching for synchronizing data and storing session state information. This solution is no longer supported. Instead, we recommend that you use the **Microsoft Azure Cache Service**.

Follow the instruction in How to Use Azure Managed Cache Service on MSDN to create and configure a new cache. Continue to How To: Store ASP.NET Session State in the Cache to configure your project to store session state information in the cache.

Additional configurations

You can perform the configurations in this section in any phase of project development.

Configuring sizes of the CMSApp web role

The size of a web role determines the number of CPU cores, the memory capacity, and the local file system size that is allocated to a running instance. You can change the size of the web role anytime, however, note that **full redeployment** is required after the change.

1. Open your Azure project in Visual Studio.
2. Open the **ServiceDefinition.csdef** file.
3. Set the **vmsize** attribute of the **WebRole** element to the size that you desire.
    - For more information about the available size options, see Virtual Machine and Cloud Service Sizes for Azure.

```
<WebRole name="CMSApp" vmsize="Large">
```

Configuring external Windows services

By default, external Windows services (Scheduler and Health monitor) that come with Kentico do not run in the Azure environment. However, you can make a few adjustments to the Visual Studio project to make the services work. After performing the steps described in this section, the Scheduler service will run as part of the **SmartSearchWorker** role and the Health monitoring service will run as part of the **CMSApp** role.

To enable external services in your Azure project:

1. Open your Azure project in Visual Studio.
2. Open the CMSAzure/**ServiceDefinition.csdef** file and uncomment the following code:

Scheduler

```
<Startup>
 <Task commandLine="InstallSchedulerService.cmd" executionContext="elevated"
taskType="simple" />
</Startup>
```

Health monitoring

```
<Startup>
      <Task commandLine="InstallHealthMonitoringService.cmd"
executionContext="elevated" taskType="simple" />
</Startup>
```

3. Open the **web.config** file in the CMSApp project and copy the value of **CMSApplicationName** key.
4. Open the SmartSearchWorker/**InstallSchedulerService.cmd** file (CMSApp_AppCode/**InstallHealthMonitoringService.cmd**) and replace <*ApplicationName*> with the value of the **CMSApplicationName** key.
    - For example, if the value of the CMSApplicationName key is:

      ```
      <add key="CMSApplicationName" value="My Web Site/Kentico8" />
      ```

    - then the appropriate line would be:

      ```
      SET _applicationIdentifier=My Web Site/Kentico8
      ```

    - You can also use the value of the CMSApplicationGuid key, but note that the services will use this value in their names.
5. Choose a password for a new administrator account, which will be created on your Microsoft Azure machine by the InstallSchedulerService.cmd (InstallHealthMonitoringService.cmd) script. Replace <*YourPassword*> with the chosen password:

   ```
   SET _adminPassword=QyCZ5HDj
   ```

6. Open the Visual Studio's **Properties Window** (by selecting **View -> Properties Window** in the main menu or by pressing **F4**).
7. Set the **Copy to Output Directory** property to *Copy always* for the following files:
    - CMSApp/App_Data/CMSModules/WinServices/services.xml
    - CMSApp/Web.config

Once the application is deployed and starts for the first time, the InstallSchedulerService.cmd and InstallHealthMonitoringService.cmd scripts register the services into the system and start them. You will then be able to manage them via remote desktop.

Deploying an Azure project

To deploy a Microsoft Azure project to the cloud, publish the project directly from Visual Studio to Microsoft Azure.

Deploying the Azure project from Visual Studio to Microsoft Azure

Deploying an Azure project from Visual Studio is a recommended practice. This procedure consists of two parts.

Importing a credentials file

1. Open your Azure project in Visual Studio.
2. Right-click the **CMSAzure** project in the Solution Explorer and select **Publish**.
3. Click the **Sign in** button.
4. Sign in to the Azure Management Portal and download the credentials file (the download should start automatically).
5. Click **Import...** and select the downloaded file.
6. Click **Next**.

Configuring the deployment

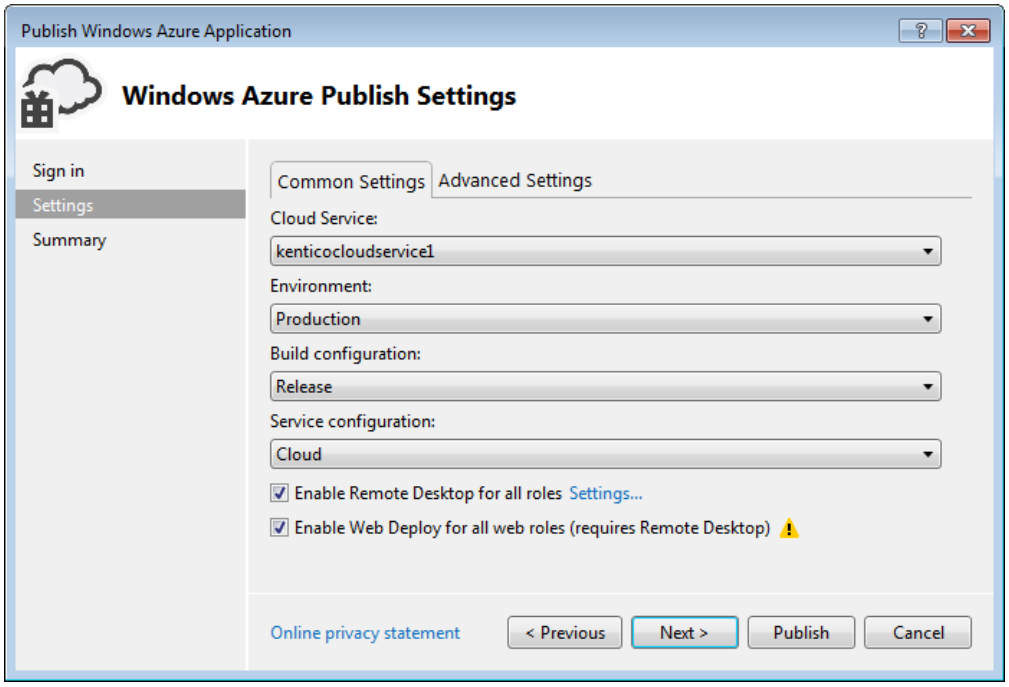
1. Select the **cloud service** you have created (according to the instructions in Preparing the cloud environment).
2. Choose the **Environment**:
    - Production - live environment.
    - Staging - environment designed for testing the deployed project before promoting it to the production environment. You can swap the production and staging environments easily.

3. Select the **Build configuration**:
    - Debug - includes debugging information.
    - Release - includes code optimizations and is better suited for live environment.

4. Select **Cloud** in the Service configuration drop-down list.
5. Check **Enable Remote Desktop for all roles**.
6. Specify the credentials for connecting to the remote desktop.
7. [optional] Check **Enable Web Deploy for all web roles (requires Remote Desktop)**.
    - After you publish your project, this option will allow you to publish changes for the web role directly without having to package or publish the whole project again. See Web Deploy for instructions.
    - You can find detailed information in the Update a Web Role MSDN article.

    > Use Web Deploy only for **development and testing purposes** and only if you use **one web role**.

    .



8. Click **Next** and check the summary.
9. Click **Publish**.

Visual Studio packages your project and uploads the package to the cloud. This may take some time (about half an hour).

Continue to Installing the database for an Azure project.

Installing the database for an Azure project

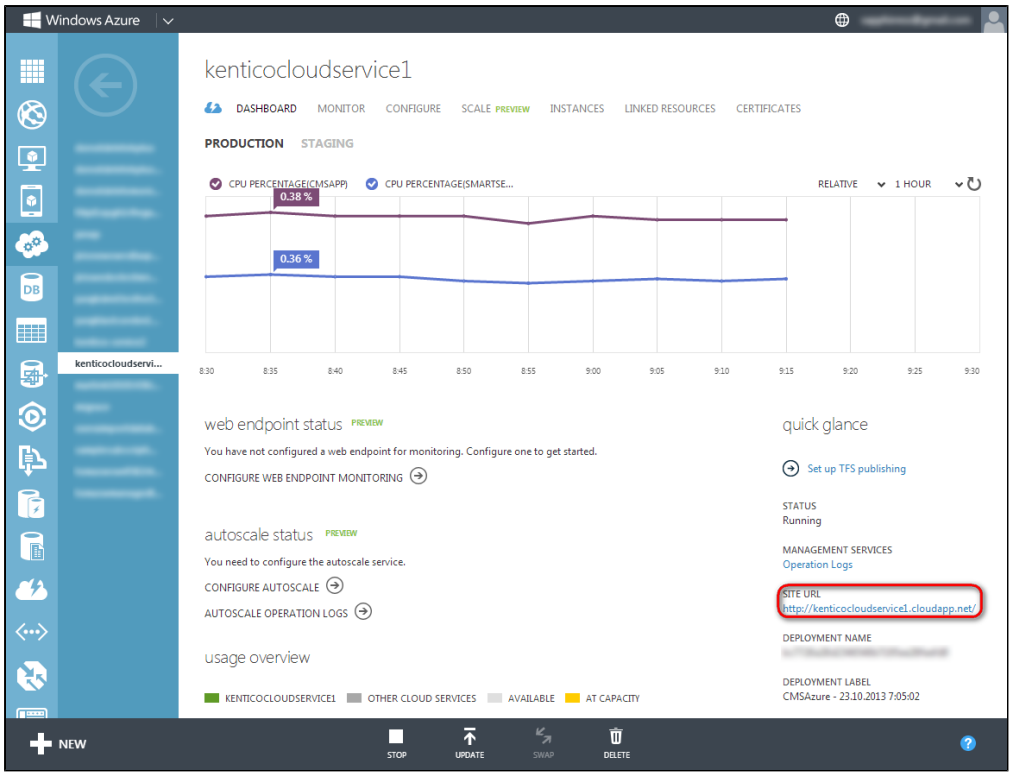
After you have deployed your Azure project to the cloud, continue to installing the database. The Microsoft Azure project can use any type of SQL database, hosted either on a standard SQL Server or in the cloud. This topic describes the installation on an SQL Azure server.

Installing a database on an Azure SQL server

To install the database on an Azure SQL server, use the system to generate the connection string, paste the connection string as a value to your cloud service and let the system install the database.

Generating the connection string

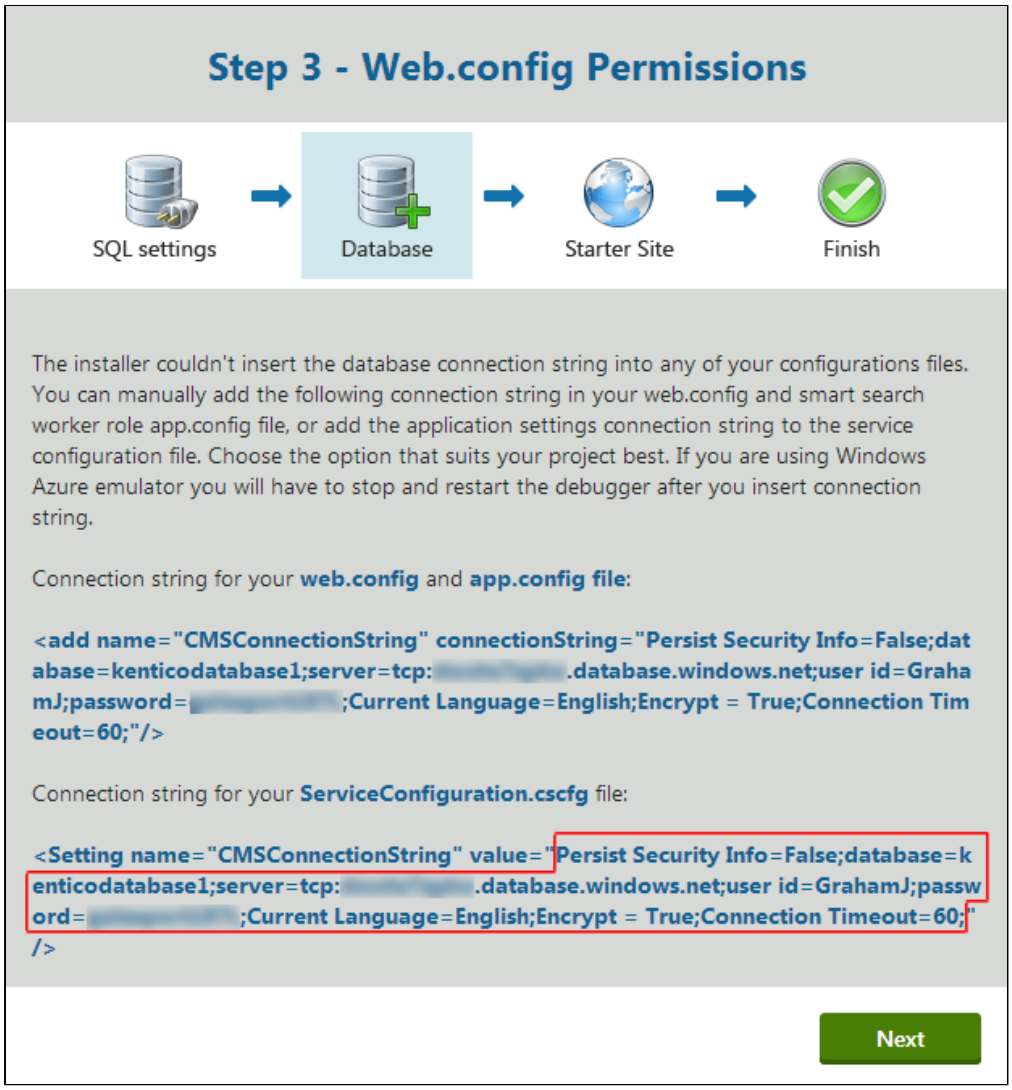
1. Access the website of your deployed Azure project.
    - You can find the URL if you choose **Cloud services ->** select your service **-> Dashboard** under Site URL.



2. Specify the target Azure SQL server (which you have created according to the instructions in Preparing the cloud environment) and supply the credentials for the server:
    - **SQL Server name or IP address** – enter the SQL Azure server name, which you can find at **SQL Databases ->** select your database **-> Dashboard** under **Server name**. The standard format is *<servername>.database.windows.net*.

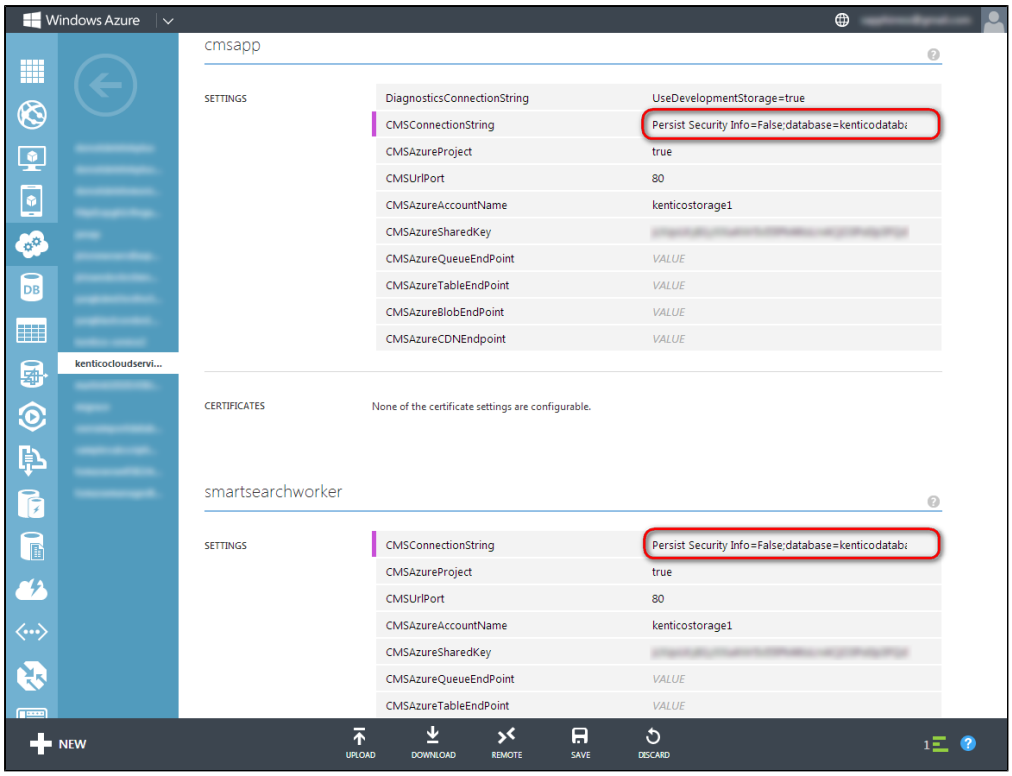
3. Click **Next**.

4. Enter the name of the SQL database (which you have created according to the instructions in Preparing the cloud environment).
5. Choose whether you want to **Create Kentico CMS database objects**.
    - Select this option if you are installing into an empty database.
    - Clear this option if the existing database already contains Kentico objects (tables, stored procedures, views).
6. Click **Next**.
7. Copy the second connection string value into the clipboard.



Setting the connection strings

1. Open the Azure Management Portal in another browser tab.
2. Navigate to **Cloud services ->** select your service **-> Configure**.
3. Paste the copied connection string value into the fields of the **CMSConnectionString** setting for the CMSApp role and also the SmartSearchWorker role if you are using it.



4. Click **Save**.
5. Close the original window with the database installation.

You have configured the connection strings for your database.

Finishing the database installation

1. Open the website of your project anew (*<your cloud service name>.cloudapp.net*).
   - The database installation automatically resumes.

2. Confirm the target database by clicking **Next**.
   - The installer now installs the database for your project.

3. Enter a license key for the domain.
   - Staging deployments are automatically covered by any other valid license, so you do not have to request a separate license for staging deployments. However, this applies only if you use the recommended configuration, which is to use separate databases for live and staging environment.

4. Click **Next**.
5. Choose an initial website to install:
   - **Choose a starter site** – you can choose from the supplied sample sites to try out the functionalities of the system or use it as a base for custom development. We recommend the **Corporate site** or the **E-commerce site**.
   - **Continue to the New site wizard** – choose this option if you want to to develop a new site from scratch. See Creating new sites using the New site wizard for more information.
   - **Import existing Kentico CMS website** – choose this option if you have already created a website using Kentico and you want to import it to the new installation. See Importing a site or objects for more information.

6. Click **Next**.

The installer creates your chosen website and finishes the installation.

If you continue to your new website, you can log in to the administration interface.

> **Login to our sample starter sites:**
>
> User name: *administrator*
> Password: (blank password)

Developing Azure projects locally

You can start developing the application in the local computing emulator provided by Azure SDK. You can use the emulator to test the website during its development without actually hosting it in the cloud.

You can use either a standard SQL database or Microsoft Azure SQL Database (note that when developing locally with a database in the cloud, you may experience slower performance based on the location of the server and database services). However, we recommend that

you use a Microsoft Azure Storage account as a file system for your project.

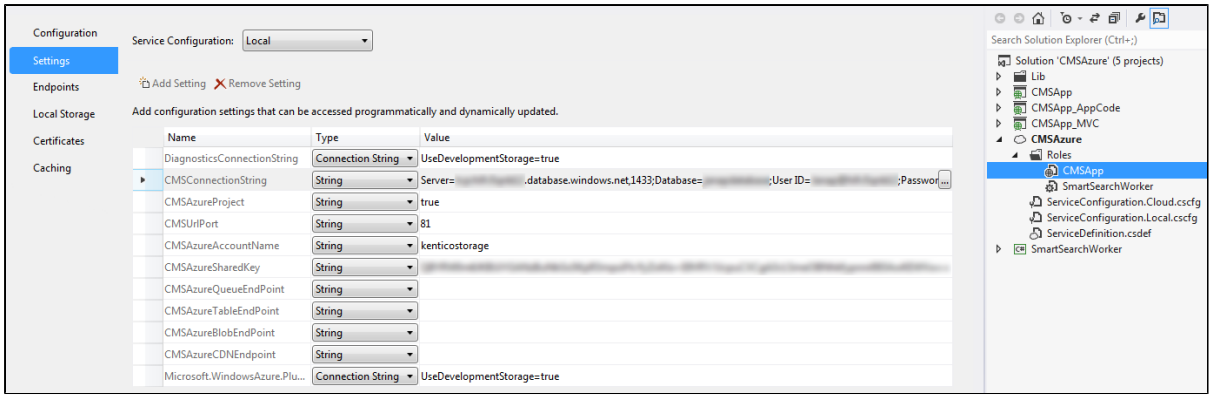
Developing Azure projects locally in the emulator

1. Prepare an SQL database or prepare a database on Microsoft Azure:
    a. Create a new Azure SQL Server.
    b. Add firewall rules for the server.
    c. Create an SQL Database.

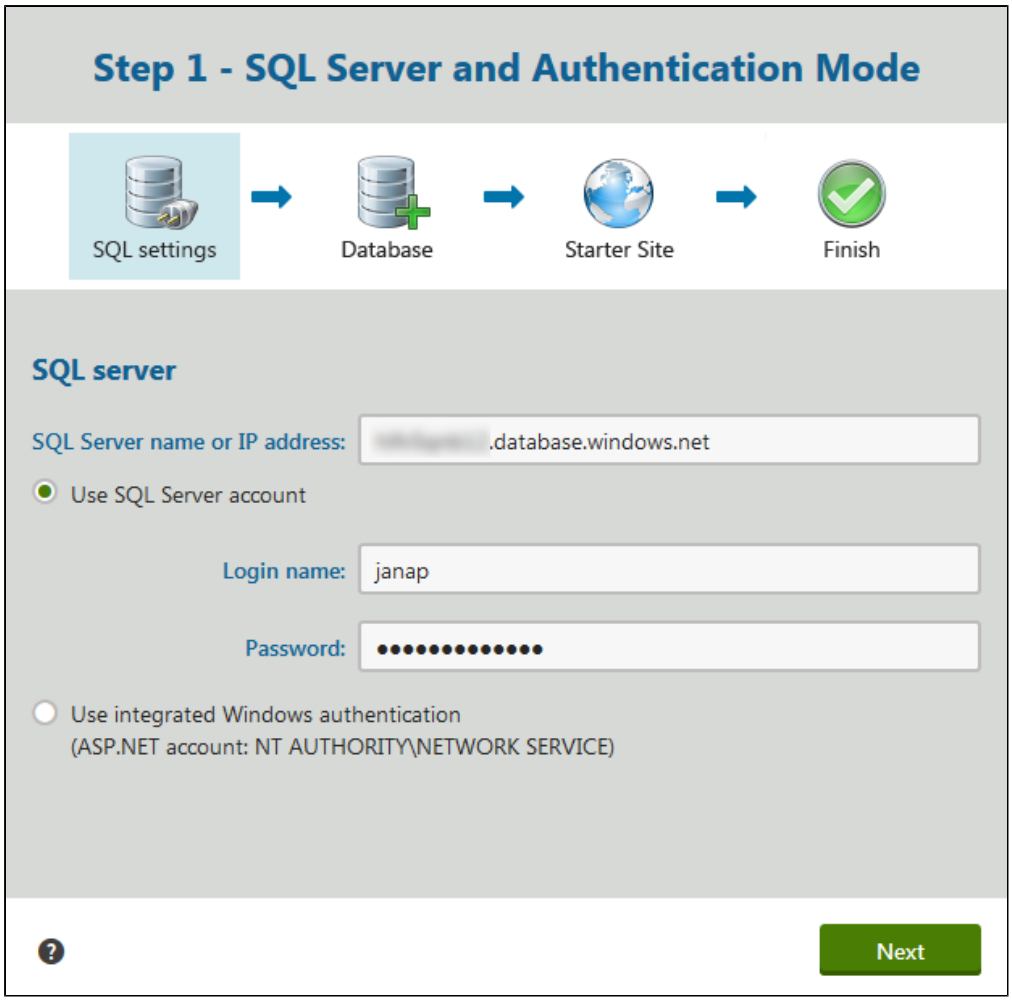
2. Create an Azure Storage account through the Microsoft Azure Management Portal.
3. Open your Azure project in Visual Studio.
4. Right-click the **CMSAzure/Roles/CMSApp** role and select **Properties**.
5. Switch to the **Settings** tab.
6. Select **Local** as the Service Configuration.
7. Set the value of **CMSConnectionString** key to the connection string for your database.
    - You can find the connection string of Azure SQL Database on the Azure Management Portal in **SQL Databases ->** select the database **-> Dashboard** tab **-> Show connection strings** link.

8. Set the values of **CMSAzureAccountName** and **CMSAzureSharedKey** keys for your Azure Storage.
    - You can find these values on the Azure Management Portal in **Storage ->** select your storage **-> Manage access keys** (in the bottom panel) as **Storage account name** and **Primary access key**.

9. Clear the values of **CMSAzureQueueEndPoint**, **CMSAzureTableEndPoint** and **CMSAzureBlobEndPoint** keys.



10. Repeat steps 4 - 9 for the **SmartSearchWorker** role.
11. Right-click the **CMSAzure** project in the Solution Explorer and select **Set as StartUp Project**.
12. Run the project in the debug mode.
13. Go through the Kentico database installation when the system automatically opens the database installation in a browser.
    - In case of Azure SQL Server, use the following format of the server name: *servername.database.windows.net*.



Your web project is now prepared to be developed on your local computer.

Developing Azure projects locally as web applications

If you are having trouble running your project in the local emulator, you can develop your Azure project as a normal web application project. We recommend this approach only if you do not plan to perform any customizations specific to Microsoft Azure environment.

1. Prepare an SQL database or create a new Azure SQL Server and SQL Database.
2. Create an Azure Storage account through the Azure Management Portal.
3. Open your Azure project in Visual Studio using the **WebApp.sln** file.
4. Open the **CMSApp\Web.config** file.
5. Add the **CMSConnectionString** key and value to the <connectionStrings> section of the web.config file.

```
<connectionStrings>
    <add name="CMSConnectionString" connectionString="Data
Source=tcp:YourServerName.database.windows.net,1433;Initial
Catalog=YourDatabaseName;User
Id=YourUsername@YourServerName;Password=YourPassword;" />
</connectionStrings>
```

   - Replace **YourServerName**, **YourDatabaseName**, **YourUsername** and **YourPassword** with your own values.
   - You can find the connection string of Azure SQL Database on the Azure Management Portal in **SQL Databases ->** select the database **-> Dashboard** tab **-> Show connection strings** link.

6. Add the **CMSAzureAccountName** and **CMSAzureSharedKey** keys to the <appSettings> section of the web.config file.

```
<appSettings>
 ...
 <add key="CMSExternalStorageName" value="azure" />
 <add key="CMSAzureAccountName" value="YourStorageAccountName" />
 <add key="CMSAzureSharedKey" value="YourPrimaryAccessKey" />
</appSettings>
```

- Replace **YourStorageAccountName** and **YourPrimaryAccessKey** with your own values.
- You can find these values on the Azure Management Portal in **Storage ->** select your storage **-> Manage access keys** (in the bottom panel) as **Storage account name** and **Primary access key**.

7. Right-click the **CMSApp** project in the Solution Explorer and select **Set as StartUp Project**.
8. Right-click the **CMSApp_AppCode\Old_App_Code\CMSModules\WindowsAzure** folder and select **Exclude From Project**.
9. Run the project in the **debug** mode.
10. Go through the Kentico database installation when the system automatically opens the database installation in a browser.
    - In case of Azure SQL Server, use the following format of the server name: *servername.database.windows.net*.

You can now develop your project as a standard web application project.

Deploying locally developed Azure projects

When you are ready, you can move your locally developed Azure project to the production environment while still being able to make changes to the project locally. For this scenario to work, you must use only **one instance** of the CMSApp web role.

WindowsAzure folder

If you are developing your project locally as a web application, you need to return the WindowsAzure folder to your project before the deployment. The folder contains the AzureInit.cs file, which is essential for running your project on Microsoft Azure.

1. Open your Azure project in Visual Studio using the **CMSAzure.sln** file.
2. Select **Show all files** on the Solution Explorer panel.
    - This option displays folders and files which are excluded from the project.
3. Open the **CMSApp_AppCode\Old_App_Code\CMSModules** folder.
4. Right-click the **WindowsAzure** folder and select **Include in project**.

The system includes the AzureInit.cs file back into the project.

Deployment with migration to a new database

This procedure presumes that you used a local SQL database and you want to migrate the database to the Azure SQL Database.

1. Export your website from Kentico database in **Sites** application **-> Export site**.
2. Create Cloud Service.
3. Create a new Azure SQL Server.
4. Create SQL Database.
5. Open your Azure project in Visual Studio using the **CMSAzure.sln** file.
6. In **Properties** of the **CMSApp** web role, on the **Settings** tab, in the **Cloud** service configuration, fill the **CMSAzureAccountName** a nd **CMSAzureSharedKey** keys with appropriate values for your Azure Storage (the same as for the Local configuration).
7. Set the **CMSConnectionString** value with the connection string of the new database.
8. Configure the SmartSearchWorker role.
9. Deploy your project to the created Cloud Service.
    - Make sure that the **Enable Remote Desktop for all roles** and **Enable Web Deploy for all web roles (requires Remote Desktop)** are selected.
10. Install a new database for your project.
11. Import the previously exported website in **Sites** application **-> Import site or objects**.

Now you can deploy changes to the project files quickly using the Web deploy functionality.

Deployment when using the same database

This procedure presumes that you are using the Azure SQL Database from the beginning and you do not need to migrate the database.

1. Create Cloud Service.
2. Open your Azure project in Visual Studio using the **CMSAzure.sln** file.
3. In **Properties** of the **CMSApp** web role, on the **Settings** tab, in the **Cloud** service configuration, fill the **CMSAzureAccountName** a nd **CMSAzureSharedKey** keys with appropriate values for your Azure Storage (the same as for the Local configuration).
4. Set the **CMSConnectionString** value with the connection string of your database (the same as for the Local configuration).
5. Configure the SmartSearchWorker role.
6. Deploy your project to the created Cloud Service.
    - Make sure that the **Enable Remote Desktop for all roles** and **Enable Web Deploy for all web roles (requires Remote**

Desktop) options are selected.

Now you can deploy changes to the project files quickly using the Web deploy functionality.

Web Deploy

If you are performing web deploy from the **same computer** from which you performed the initial deployment, this procedure is straightforward as you can use an already created publish profile:

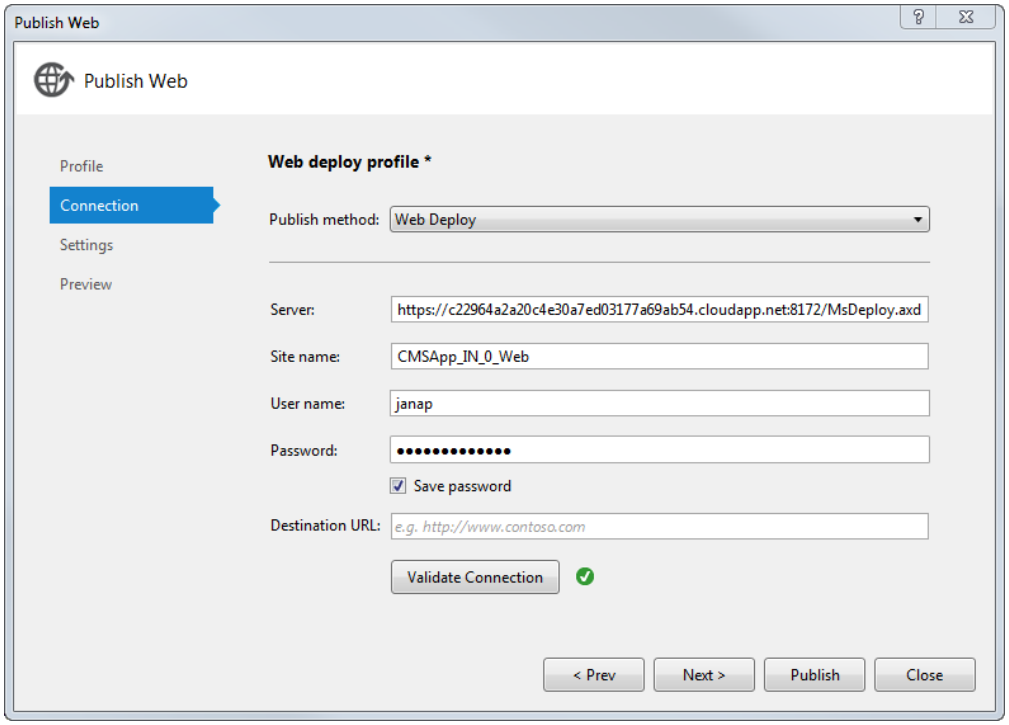
1. Right-click the **CMSApp** project and select **Publish**.
2. Select the existing publish profile from the drop-down list.
3. Switch to the **Connection** tab.
4. Provide the password for remote desktop connection.
5. Click **Publish**.

If you are performing web deploy from a **different computer** than the one from which you performed the initial deployment, you need to create a new publish profile:

1. Right-click the **CMSApp** project and select **Publish**.
2. Click the drop-down list and select <New profile...>.
3. Type the name of the profile and click **OK**.
4. On the **Connection** tab, select **Web Deploy** as the Publish method.
5. Provide the **Server** address:
    a. Open the Azure Management Portal in a browser.
    b. Select **Cloud services ->** your cloud service **-> Dashboard** tab.
    c. Copy the **Site URL** without the protocol (http://) and trailing slash (/) to the Server field of the web deploy profile in the following format:

```
https://YourSiteURL:8172/MsDeploy.axd
```

6. Provide the **Site name**:
    - This corresponds to the name of your website registered in IIS of the server.
    - You can find the name by remotely connecting to the server (**Cloud services ->** your cloud service **-> Instances** tab -> Click **Connect** in the bottom panel and run the downloaded file).
    - The default value is **CMSApp_IN_0_Web**.
7. Type the username and password for remote desktop connection.



8. Switch to the **Settings** tab.
9. Choose the build Configuration.
10. Click **Publish**.

Final deploy

When your project is fully developed and ready to go live, Deploy your project again, with the **Enable Remote Desktop for all roles** and **Enable Web Deploy for all web roles (requires Remote Desktop)** options CLEARED. After that, your CMSApp web role can use more than one instance.

Microsoft Azure Web Sites

Microsoft Azure Web Sites are quick to deploy and easy to set up and manage. However, they do not offer as many customization options as cloud services. For a more detailed comparison, see Azure Web Sites, Cloud Services and Virtual Machines comparison on MSDN.

You can create Kentico websites using the Azure Web Sites service:

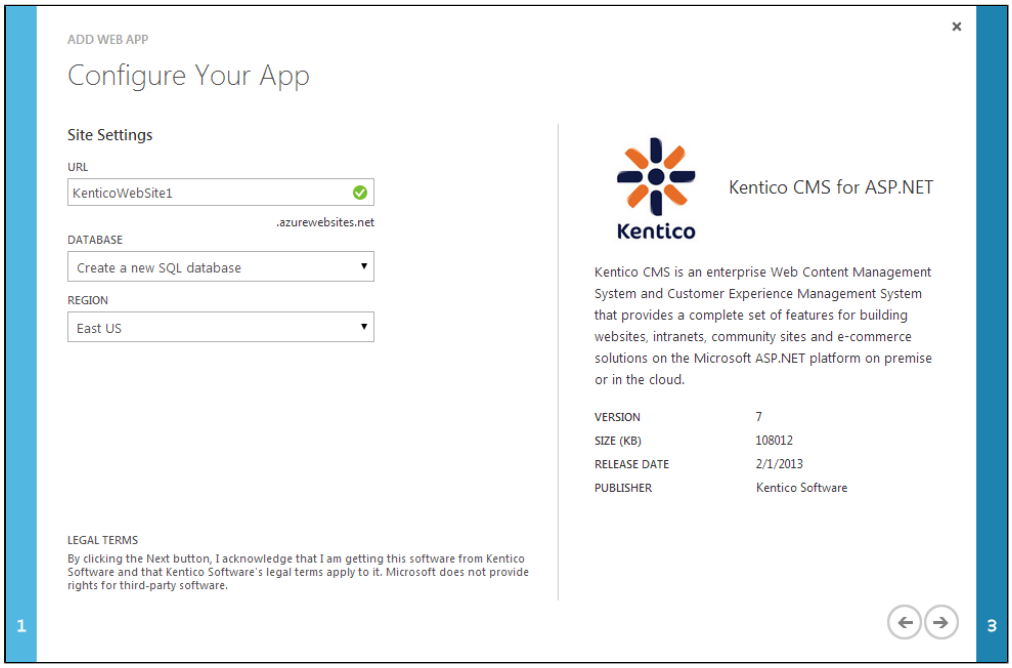
- From the Web Application Gallery
- From Visual Studio

When you have a website created, you can manage its files over an FTP connection.

Creating Web Sites from Web Application Gallery

You can find a Kentico package in the application gallery. Creating new websites using the provided template is quick and easy.

1. Open the Azure Management Portal.
2. Click **New -> Compute -> Web Site -> From gallery**.
3. Select **Kentico CMS for ASP.NET**.
4. Click Next.
5. Type the URL (name) of your website.
6. Choose to create a new SQL database.
7. Choose a region (location) for your website.



8. Click Next.
9. Type the name of the database
10. Create a new server or choose an existing server for the database and provide login credentials.
11. Complete the creation.

Database installation

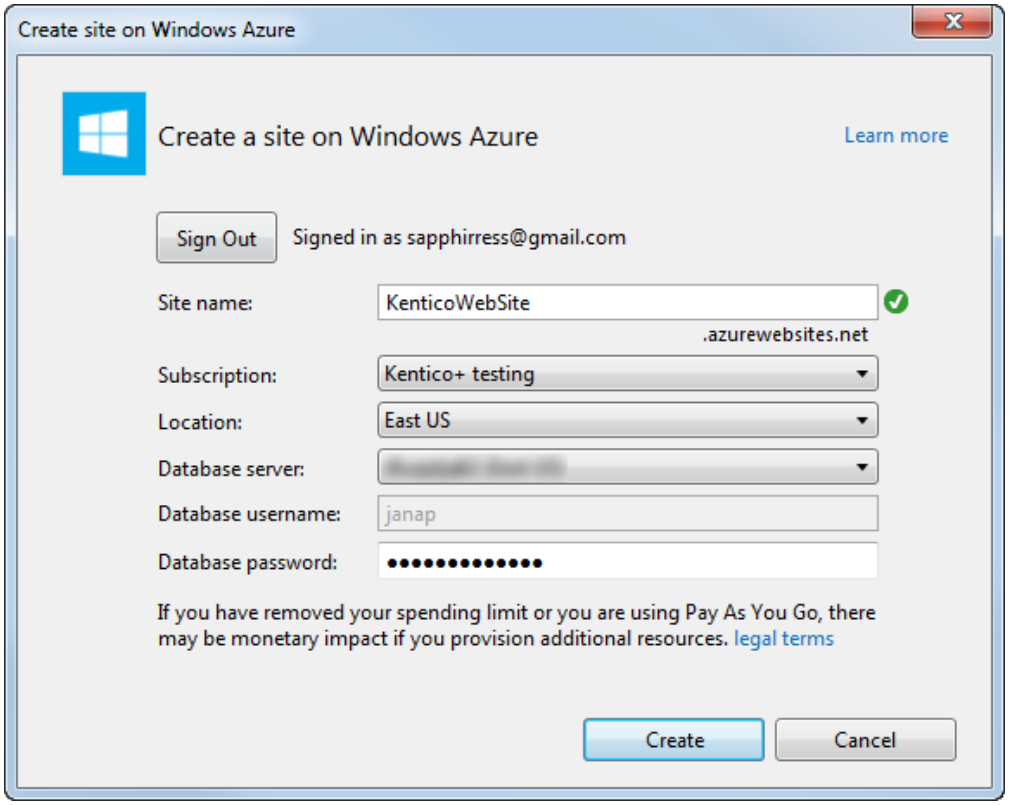
1. Access the created website.
2. Provide the server name and credentials for the server used to create the database in.
3. Click **Next**.
4. Select **Use an existing database** and type the name of the database created for the web site.
5. Click **Next**.
6. Click **Next** without filling any field.
7. Select **Import existing Kentico CMS website** and click **Next**.
    - The installer redirects you to the administration interface.
8. Switch to the **Licenses** tab and add the license for the website domain.

Creating Web Sites from Visual Studio

This scenario presumes that you already have an installed Kentico web project on a local computer and you want to deploy it as Microsoft

Azure Web Site.

1. Open your web project in Visual Studio.
2. Open the **Server Explorer** tab (or Database Explorer tab in Express editions of Visual Studio).
3. Right-click Windows Azure/**Web Sites** and select **Add New Site...**
   - A window **Create site on Windows Azure** opens.
4. Type the web site name (URL).
5. Select the location for your website.
6. Select an existing database server or create a new one.
7. Provide the credentials for the server.



8. Click **Create**.

Visual Studio creates the Web Site. Continue with adjusting the configuration of the web.config file and deploying your project.

1. Right-click the created web site and select **View settings**.
2. Copy the connection string value and paste it to the CMSApp/**web.config** file of your web project.

```
<connectionStrings>
    <add name="CMSConnectionString" connectionString="Data
Source=tcp:YourServerName.database.windows.net,1433;Initial
Catalog=YourDatabaseName;User
Id=YourUsername@YourServerName;Password=YourPassword;" />
</connectionStrings>
```

Replace **YourServerName**, **YourDatabaseName**, **YourUsername** and **YourPassword** with your own values.

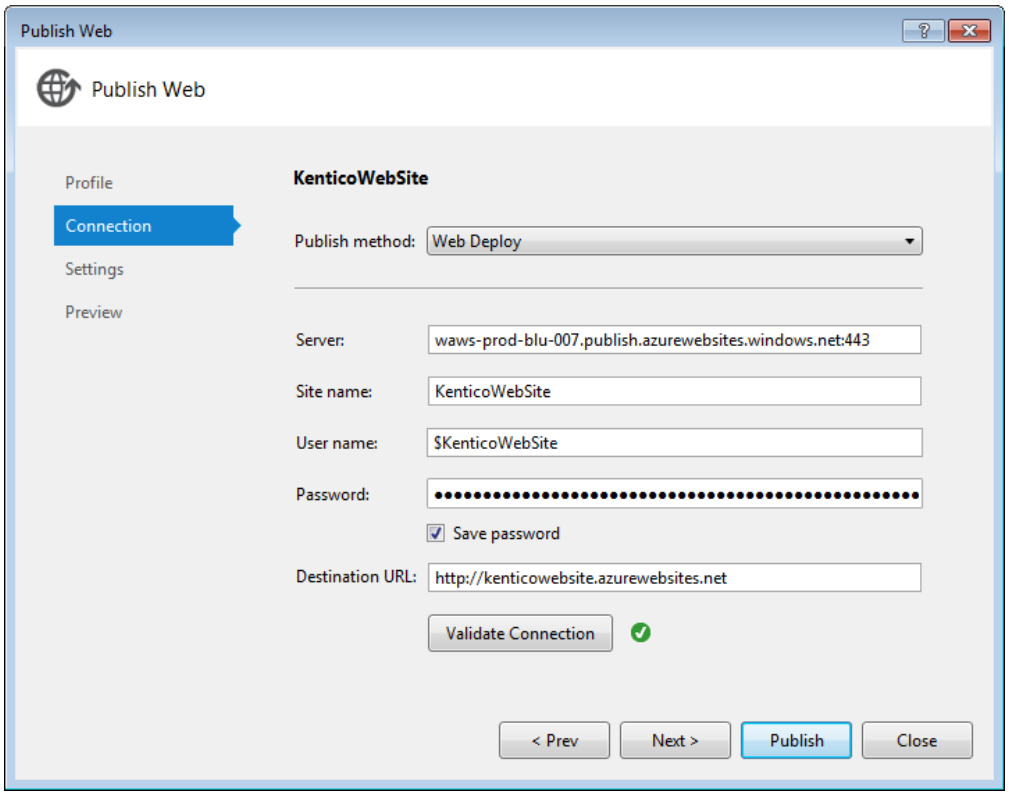
Configuring the connection string after the deployment

You can also configure the connection string through the Azure Management Portal in **Web Sites ->** select a website **-> Configure** tab **-> connection strings** section. To set the connection string for the web site, change the *DefaultConnection* name to *CMSConnectionString*.

3. Make sure that the sessionState mode is set to **InProc** in the web.config file.

```
<sessionState mode="InProc" />
```

4. Right-click the project folder (CMSApp or CMS) in the Solution Explorer and select **Publish** (or **Publish Web Site**).
5. Click **Import**.
6. Select the created website in the drop-down list.
7. Click **OK**.
8. Leave the settings as they are.

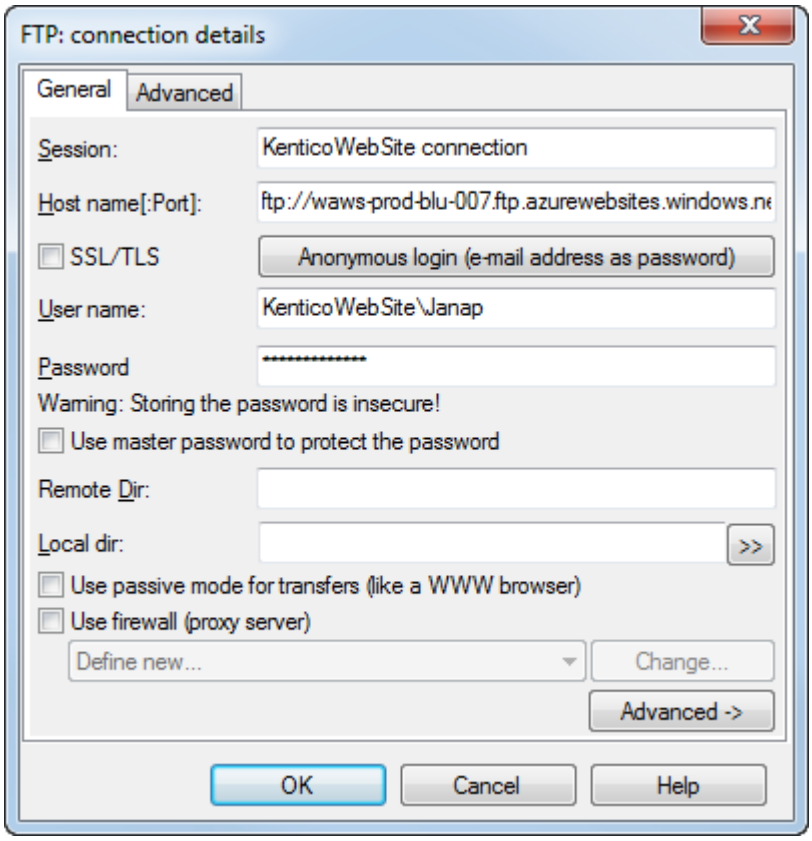


9. Click **Publish**.
10. When the project is published and the website of your project is automatically opened in a browser, complete the database installation.

Uploading files to Microsoft Azure Web Sites over FTP

When you have created Azure Web Sites and deployed your web project, you can manage its files over an FTP connection:

1. Open the Azure Management Portal.
2. Select your web site on the **Web Sites** tab.
3. Switch to the **Dashboard** tab.
4. Click **Set up deployment credentials**.
5. Type a user name and password and click OK.
6. Use the **FTP host name** and **Deployment / FTP User** information to configure your FTP client.



When you connect to your website project, you can see the following folders:

- Logfiles - contains website specific diagnostics log files.
- site - contains website files.