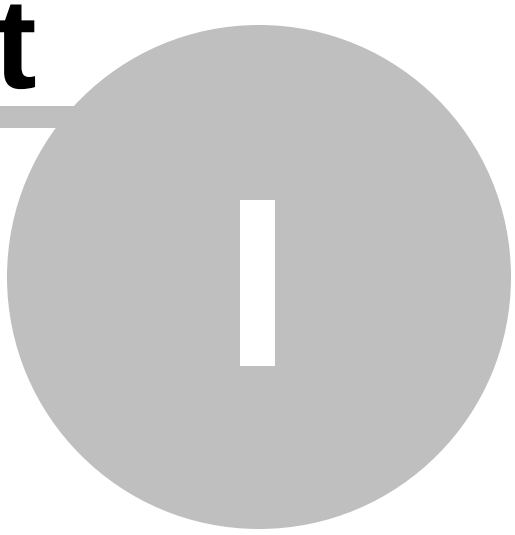
Kentico CMS 7.0 Windows Azure Deployment Guide

Kentico
unlimited website possibilities

www.kentico.com

Table of Contents

Part



Introduction

1 Introduction

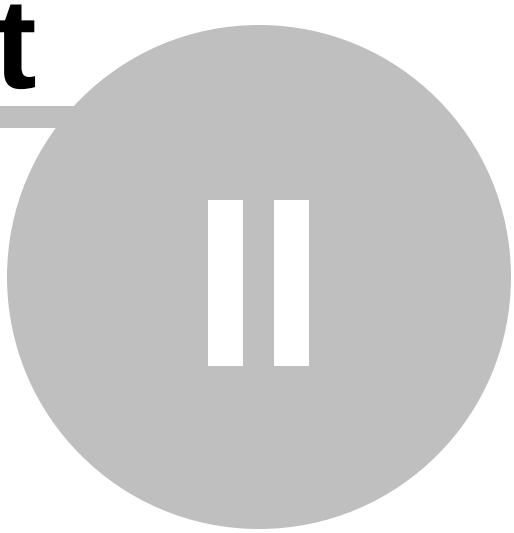
1.1 About this guide

The main purpose of this guide is to describe the steps that need to be taken to successfully deploy a Kentico CMS website to the Windows Azure cloud computing platform. It also provides information about the differences between an application deployed on Windows Azure and one that is installed and set up for on-premise hosting. You can also learn about the additional configuration options available in Kentico CMS that are related to Windows Azure and its various services and components.

To follow the instructions in this guide, you will need to have general knowledge of the Windows Azure platform and a basic overview of the development principles that apply to Azure applications. Another requirement is access to an active Windows Azure subscription. If you are new to Windows Azure and need to create a subscription or are looking for information, please visit http://www.microsoft.com/windowsazure/.

If you are not familiar with Kentico CMS, we recommend that you get to know the system and its features before attempting to deploy a website to Windows Azure. First-time users can go through the Kentico CMS Tutorial, which describes the standard installation in a local environment and the basics of its administration interface. If you need additional information about any parts of the system, please refer to the Kentico CMS Developer's Guide.

Part



Installation and deployment

2 Installation and deployment

2.1 Overview

The installation process for Kentico CMS and the deployment of a website to Windows Azure consists of the following steps:

- Setup (kenticocms.exe) - installs the basic files required to create Kentico CMS web projects and the documentation. This setup is intended for your development machines, since it allows you to run the web installer.
- Web Installer - can be used to create new Kentico CMS projects specifically designed for deployment to the Windows Azure platform.
- Application structure and configuration - once you have an Azure project installed, you can configure it as necessary and start developing your website. At this stage, you can run the application locally on the Windows Azure emulator.
- Deployment to the cloud - there are several ways to deploy a website to Windows Azure. The guide describes the simplest approach, which is to create a deployment package directly in Visual Studio, and then upload it through the Windows Azure Management Portal.
- Database setup - when the website is opened in a browser for the first time, it starts the database installer, which creates the required system tables and other data on a specified database. The target database server can either be hosted on-premise or on SQL Azure.

If you wish to move an existing Kentico CMS website to Windows Azure, please follow the instructions given in the Deploying an existing website to Windows Azure topic.

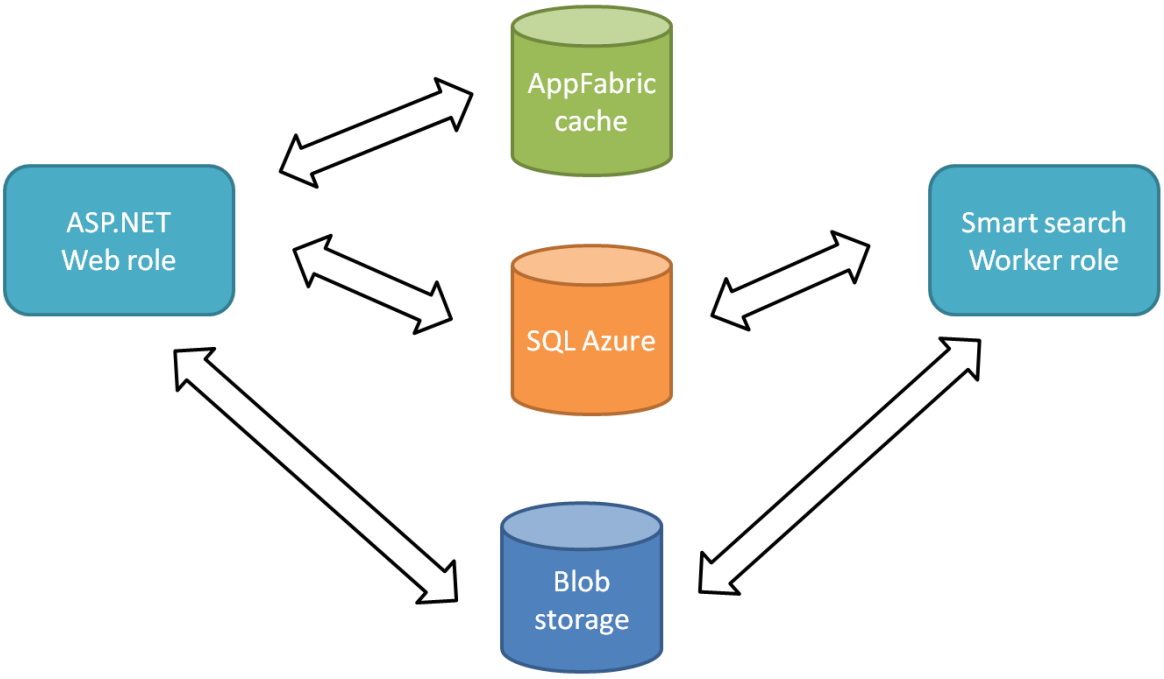
Before you start the installation and deployment, please see the Prerequisites & Limitations topic and make any necessary preparations. It is recommended to read through the limitations and confirm that your website requirements are compatible with the environment.

If you wish to learn the details about Kentico CMS operation in the cloud, refer to the Architecture topic.

2.2 Architecture

This chapter describes how Kentico CMS works within the Windows Azure environment, which features of the service are utilized and how the application stores and manages its data in the cloud environment.

The following diagram illustrates the relations between the individual elements that together form Kentico CMS in the cloud environment. The different components depicted in the diagram are described in the text that follows.



Application

If you choose to install Kentico CMS to Windows Azure, all files will be grouped into a solution based on Visual Studio's Windows Azure template. The solution contains several projects. One of them is a web application, which encompasses almost all the functions of Kentico CMS and is designed to run as a Windows 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. In order to index content of websites correctly and effectively, the Smart search worker runs as a Windows Azure **Worker role**.

Because the application is divided into these two entities (called **services** in the Azure terminology), you will also need to configure them separately. More information on how to configure the application can be found in Application structure and configuration.

Database

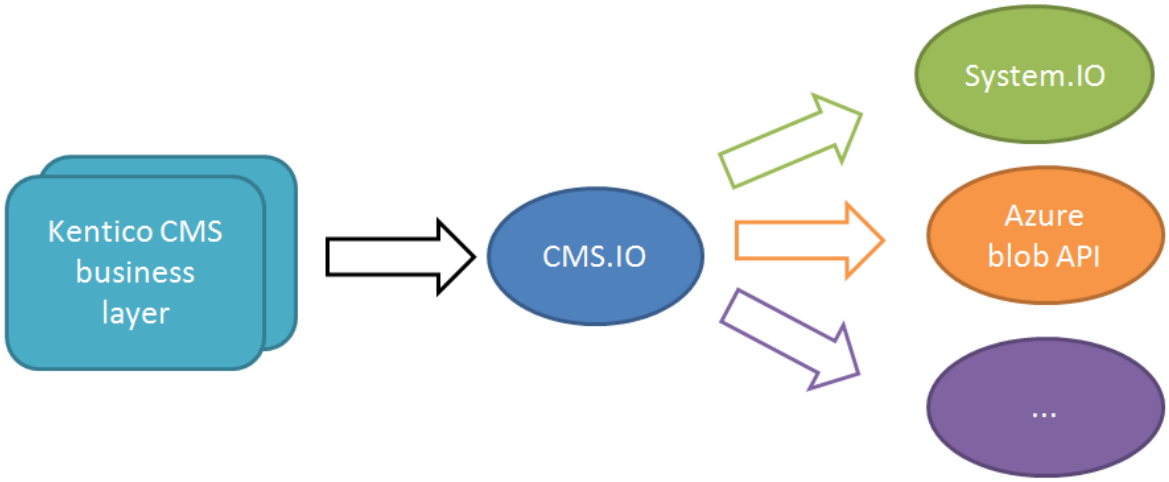
Kentico CMS on Windows Azure uses the **SQL Azure** relational database. This database engine is almost identical to the standard SQL Server engine, with only a few limitations. These limitations are taken into account in Kentico CMS, hence no additional configuration or customization is required. If you're interested in which SQL Server features are not available in SQL Azure, refer to the following MSDN article.

File storage

Windows Azure doesn't offer a persistent file system similar to the file systems in Windows that you're used to. Data stored within Azure can't be hierarchically organized into folders. However, Kentico CMS provides an abstract layer called **CMS.IO**, which enables the system to operate on different types of file storages.

The CMS.IO namespace acts as an intermediary between the Kentico CMS business layer and various file storages, including Azure's **blob storage**. On a standard non-Azure installation, CMS.IO only overrides the System.IO namespace. On Windows 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 the Hybrid storage scenarios topic.

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

Important!

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

This won't normally affect functioning of the system, although there are a few scenarios when you will have to take additional actions to prevent unexpected behavior.

Running on multiple instances

The fact that Kentico CMS can run in multiple instances on Windows Azure creates the need for synchronization of data between the instances. The CMS handles this by considering each instance a web farm server. Unlike the traditional web farm comprising multiple physical servers, you don't have to configure the web farm servers manually in the multiple-instance Azure environment. The only configuration that needs to be done is setting the number of instances in the service configuration file. Data is synchronized via web farm tasks, which are created and executed automatically.

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 doesn't reside constantly in one place, its state has to be stored separately.

Kentico CMS is pre-configured to use the **Windows Azure AppFabric** caching service to take care of session information. This way, all instances of the application have access to their data, which keeps them synchronized.

Please note

Using the AppFabric cache is recommended, however, you can choose any other means of storing session state information.

2.3 Prerequisites & Limitations

If you wish to run Kentico CMS in Windows Azure, ensure that the requirements listed in this topic are fulfilled.

Development tools

The following software must be installed on the machines where you plan to develop and deploy your Windows Azure application:

- Microsoft Visual Studio 2010/2012 (including Microsoft Visual Studio 2010/2012 Express Edition)
- Microsoft .NET Framework 4.0
- Windows Azure SDK 1.7 or 1.8
- Windows Azure Tools for Microsoft Visual Studio
- Windows Azure AppFabric SDK
- Microsoft Internet Information Services (IIS) 7.0 or newer - required if you wish to run the application locally on the Windows Azure Compute Emulator.

The Windows Azure SDKs and Tools for Visual Studio may be downloaded from http://www.microsoft.com/windowsazure/sdk/.

Windows Azure requirements

The exact hosting requirements depend on the resource consumption and traffic load of your site and its components. For live production deployments of Kentico CMS websites, it is highly recommended to use at least the **Small** *Compute Instance Size*.

In addition, you will need to prepare the following services for your Windows Azure subscription:

- **Azure Storage account**
- **AppFabric Caching service** (128 MB or more) - required if you wish to run the application on multiple instances.
- **SQL Azure server and database** - required if you wish to host your application's database on Windows Azure. The smallest available database (1GB) is more than sufficient for the default Kentico CMS installation.

Licensing

If you wish to run Kentico CMS in multiple Windows Azure instances, you will need a licence which allows the corresponding number of web farm servers, regardless of the number of actual physical servers involved. For more information, refer to our website.

Limitations

There are some issues and limitations related to the Windows Azure environment, that affect the functionality of Kentico CMS websites:

If the application's database is hosted on 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 Azure features and requirements, please see the Guidelines and Limitations (SQL Azure Database) articles on MSDN.

A shortcoming of any application running on Windows 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.

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

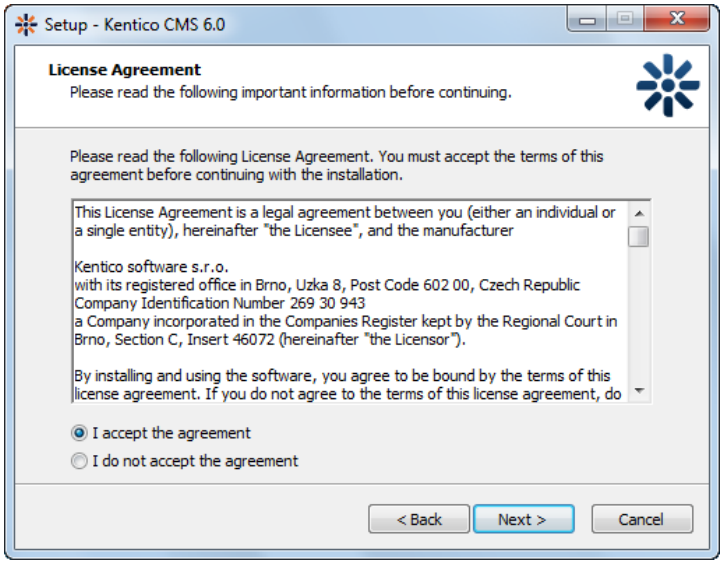
- **Full site import -** importing sites is possible, but files stored in the web project, such as custom user controls, ASPX page templates or design files, will not be included in the import process. To work around this limitation, you can copy such files into the target Azure project and include them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

2.4 Setup (KenticoCMS.exe)

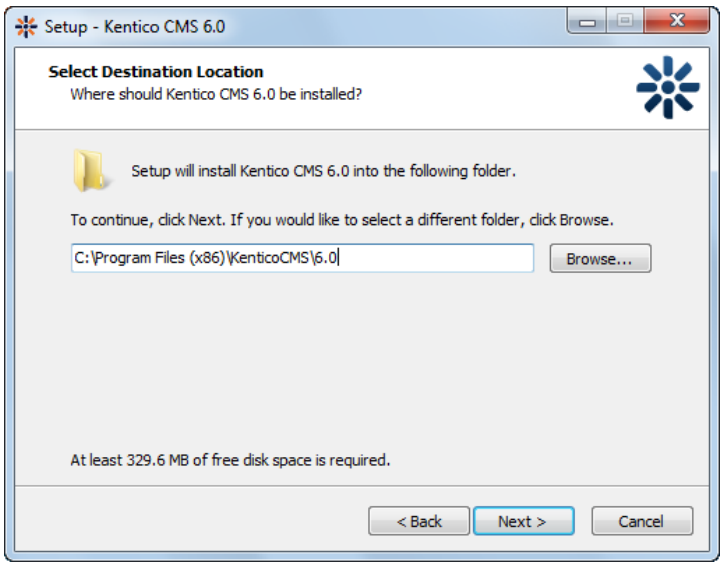
If you do not have Kentico CMS installed yet, it is necessary to do so before you can continue. The initial setup is the same for both standard installations and Azure deployment scenarios:

1. Run *KenticoCMS_<version>.exe*. You will see the welcome screen. Click **Next**.

2. Read the license agreement, accept it and click **Next** if you want to continue.

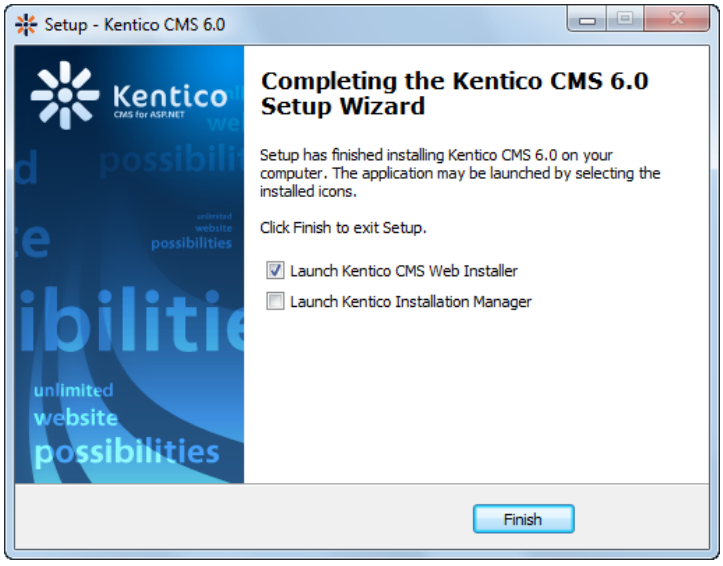


3. Choose the location where the Kentico CMS web installer, documentation and other related files will be deployed. Click **Next** and then **Install**.



Please note: this is not the folder where your website will be placed, it's only a location for Kentico CMS program and help files.

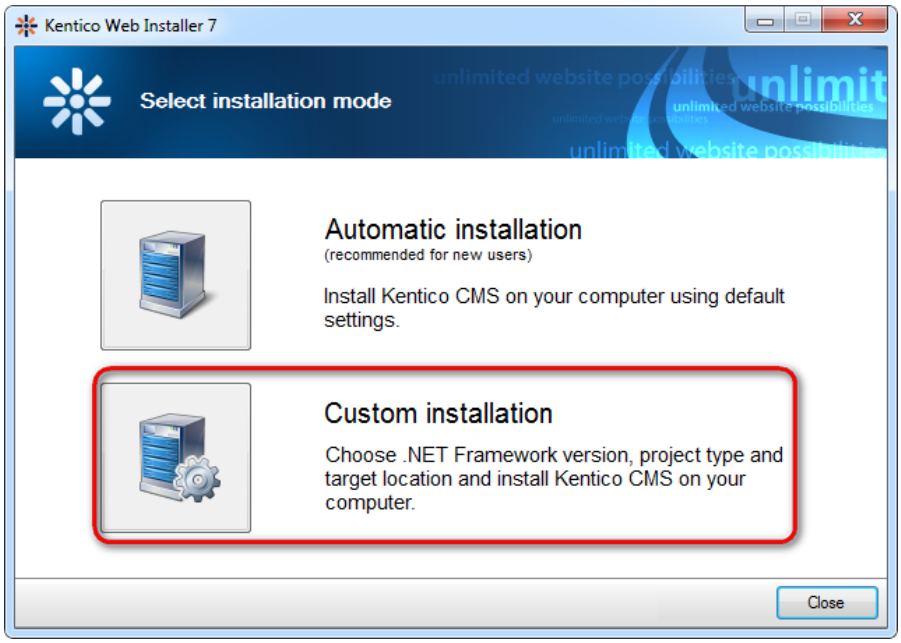
4. After the installation is finished, check the **Launch Kentico CMS Web Installer** option and click **Finish**. Then continue with the Web installer.



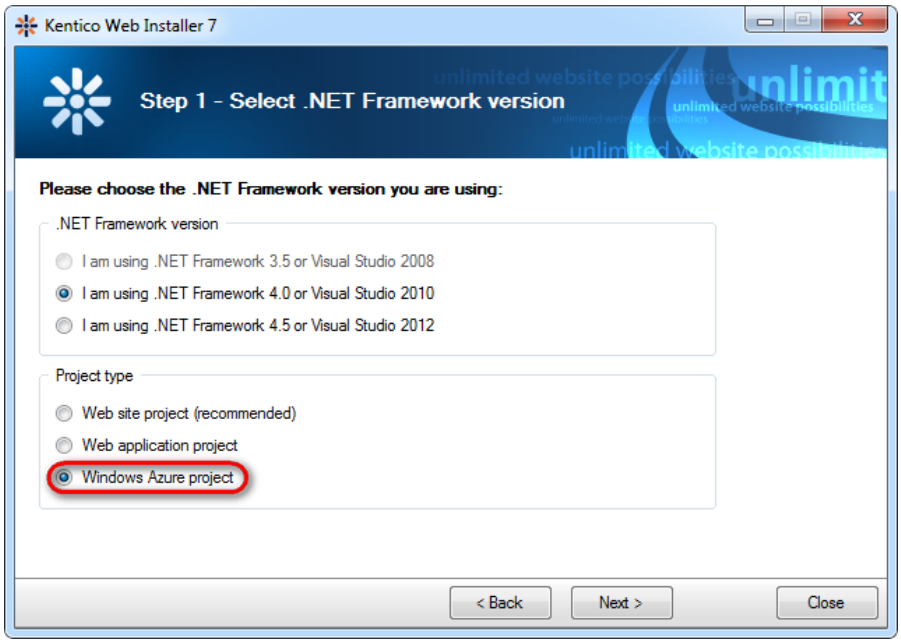
2.5 Web installer

To create a new Azure web project, run the Kentico CMS Web Installer. You can either do that directly after completing the initial setup wizard or by running *WebInstaller.exe* from the **Bin** folder of your installation directory.

1. Once the installer opens, choose **Custom installation** as the **Installation mode**.



2. In Step 1 of the installer, choose the **Windows Azure project** option, which will create a web application suitable for deployment to the Windows Azure platform. You are allowed to choose .NET Framework 4.0 or 4.5 for the installation of Windows Azure project.



Click **Next**.

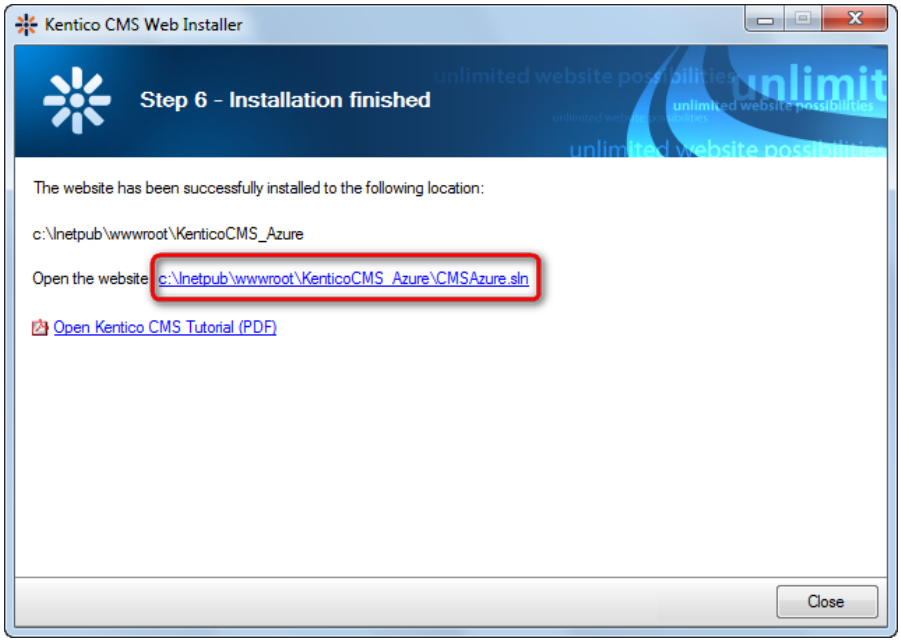
3. You can then select from the following two installation types:

- **I want to use built-in web server in Visual Studio** - creates a new clean installation of the Kentico CMS Azure project.
- **I want to modify an existing Kentico CMS installation** - you can use this option to change an existing Azure project installation on the local machine (i.e. add or remove its components).

4. Next, you will need to enter a target location for the project on your local disk.

5. In Step 4, you can choose whether you want to perform a **Full installation** including all optional components or a **Custom installation** where you can select which components should be installed.

6. The installer will then copy the project files to the specified folder. Once this is complete, open the solution in Visual Studio using the displayed link.

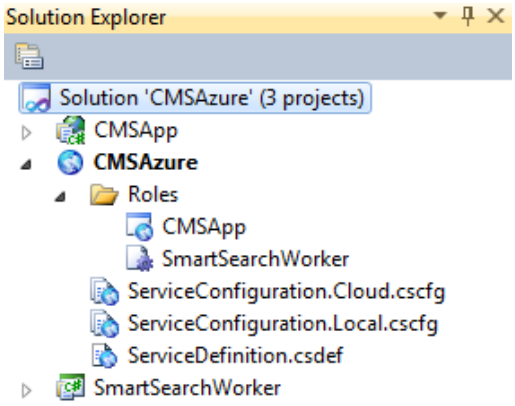


Continue on to Application structure and configuration, where the Kentico CMS Azure solution is described.

2.6 Application structure and configuration

After you finish the installation of the Kentico CMS web application as an Azure project, open the solution in Visual Studio. You can either follow the link displayed in the final step of the Web Installer, or use the **CMSAzure.sln** file created in the project directory at any time.

The structure of the solution is slightly different than that of a standard Kentico CMS installation and some additional projects are included:



The first project (**CMSApp**) is Kentico CMS in the web application format. The second (**CMSAzure**) uses the *Windows Azure Project* template and is necessary to ensure that the application can be deployed as a hosted service on Windows Azure. It contains the service definition and service configuration files and has the other projects defined as roles. The CMSApp project is added as a Web role and the **SmartSearchWorker**, used to carry out tasks required by the Smart search module, is set up as a Worker role.

If you are certain that you will not need the Smart search module, you can remove the SmartSearchWorker project (and role) from the solution. This reduces the number of roles that need to be hosted, so the costs of running your application on Windows Azure will be lower. In this case, you can ignore any further instructions related to this project and role.

Running the project on the local emulator

The application can be started on the local compute and storage emulator provided by the Azure SDK. This can be used to test the website during its development without actually hosting it in the cloud.

To run the website on the emulator you will need to have the appropriate service configuration file set up in a specific way. The following settings must be present in the <*ConfigurationSettings*> section of both roles:

```
<Setting name="CMSUrlPort" value="81" />
<Setting name="CMSAzureAccountName" value="devstoreaccount1"/>
<Setting name="CMSAzureSharedKey"
value="Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1SZFPTOtr/
KBHBeksoGMGw=="/>
<Setting name="CMSAzureQueueEndPoint" value="http://127.0.0.1:10001/
devstoreaccount1"/>
<Setting name="CMSAzureTableEndPoint" value="http://127.0.0.1:10002/
devstoreaccount1"/>
<Setting name="CMSAzureBlobEndPoint" value="http://127.0.0.1:10000/
devstoreaccount1"/>
```

If you wish to use the Windows Azure diagnostics module and have it transfer data to the local development storage, you also need to use the following settings:

CMSApp role:

```
<Setting name="DiagnosticsConnectionString" value="UseDevelopmentStorage=true" />
```

SmartSearchWorker role:

```
<Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
```

These settings are configured by default in the **ServiceConfiguration.Local.cscfg** file after a clean installation of the Azure project. The *Local* service configuration is used when running on the emulator. If necessary, you can specify a different configuration file by right clicking the **CMSAzure** project, selecting **Properties** and switching to the **Development** tab of the displayed dialog.

You can start the emulator from Visual Studio by selecting **Debug -> Start Debugging** (*F5*) or by pressing *CTRL + F5* to run the application on the emulator without debugging. Once the project is successfully built and everything starts up, the website will be opened in your browser and you can start working with it as usual. The first step that the website will require is to perform the database installation (unless you entered a connection string manually). Please see the Database setup topic to learn more about the database installer and connection strings in a Windows Azure environment.

If you wish to debug the application on the emulator, it is first necessary to make a small modification in the **web.config** file of the **CMSApp** project. Find the *<compilation>* element under the *<system.web>* section and change the value of its **debug** attribute to *true*.

```
<system.web>
    ...
    <compilation debug="true" numRecompilesBeforeAppRestart="100"
targetFramework="4.0">
        ...
    </compilation>
    ...
</system.web>
```

Remember to set the **debug** attribute back to *false* when you are done debugging to make sure that it is disabled on your production deployments (for security and performance reasons).

Please note that you will have to restart the emulator deployment every time you make any changes to the configuration or perform customizations of the application's code. This ensures that the changes are reflected on the website.

Emulator performance and requirements

You may experience relatively slow loading times when running Kentico CMS on the local emulator. This is caused by the limited resources of the emulated environment. The performance of the website will be significantly better when it is actually deployed to Windows Azure.

Additionally, you may encounter memory-related errors during the emulator startup when running on a low-end machine. This is caused by the relatively high requirements of the emulator, not by a problem with Kentico CMS.

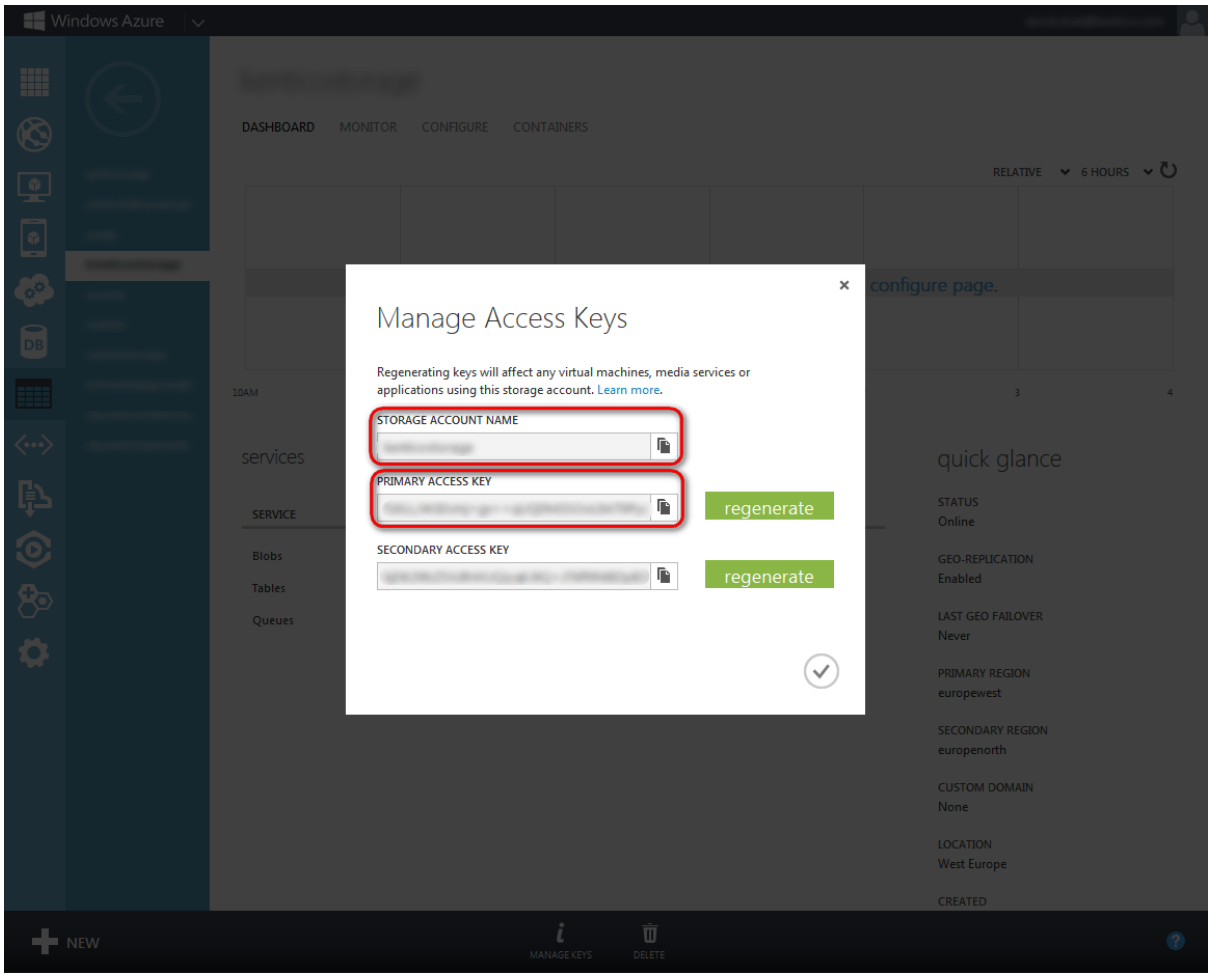
Configuration for cloud deployment

If you want to move the application to the Azure cloud, you will first have to ensure that your Azure subscription contains the necessary accounts and services. The application must then be configured according to the specific details of your subscription. You can set up your subscription and also find the required information on the Windows Azure Management Portal, which you can access at https://manage.windowsazure.com/. Use your Windows Live ID credentials to sign in.

The following steps are required:

1. You will need an active Azure Blob Storage account to serve as the file system of your cloud-based application. To create a new storage account, select **Storage** in the management portal, then click **New** and continue according to the instructions in the portal.

Once you have an existing storage account, click it in the list of accounts and click Manage keys.



Copy your **Storage account name** and **Primary access key**, and enter them as the values of the **CMSAzureAccountName** and **CMSAzureSharedKey** settings in the *<ConfigurationSettings>* section of both the **CMSApp** and **SmartSearchWorker** roles in your application's service configuration file. You can use the default **ServiceConfiguration.Cloud.cscfg** file for this purpose, so you do not have to overwrite the local configuration for the emulator.

```
<Setting name="CMSAzureAccountName" value="AccountName"/>
<Setting name="CMSAzureSharedKey" value="PrimaryAccessKey"/>
```

Additionally, if you wish to use the Windows Azure diagnostics module and log its output on the storage account, you must set the diagnostics connection string of both roles to use https and enter the name and access key of the given account:

```
<Setting name="DiagnosticsConnectionString" value="DefaultEndpointsProtocol=https;
AccountName=AccountName;AccountKey=PrimaryAccessKey" />
```

2. If you want the storage account connection to use the default endpoints for its services (blob, table or queue), leave empty values for the **CMSAzureQueueEndPoint**, **CMSAzureTableEndPoint**, **CMSAzureBlobEndPoint** settings for both roles in the service configuration file. Otherwise, if you use a specific endpoint for any of the services (e.g. if you have registered a custom domain name for the blob service or wish to access a specific container), enter the given endpoint as the value of the appropriate setting.

3. Another thing that can be set up in the service configuration file is the number of instances used for the **CMSApp** role, which represents the Kentico CMS application. This determines how many virtual machines will be dedicated to the website, so its performance and load handling capacity will increase with each instance. Simply enter the required number of instances as the value of the **count** attribute of the role's <*Instances*> element.

```
<Role name="CMSApp">
  <Instances count="4" />
  <ConfigurationSettings>
    ...
  </ConfigurationSettings>
</Role>
```

Each instance will be represented by a separate web farm server within the Kentico CMS system. The creation and management of the servers is handled automatically, and it is not necessary to perform any further configuration.

Instance licensing

In order for the website to run, the Kentico CMS license used for your domain must allow at least as many servers as the amount of instances set for the role.

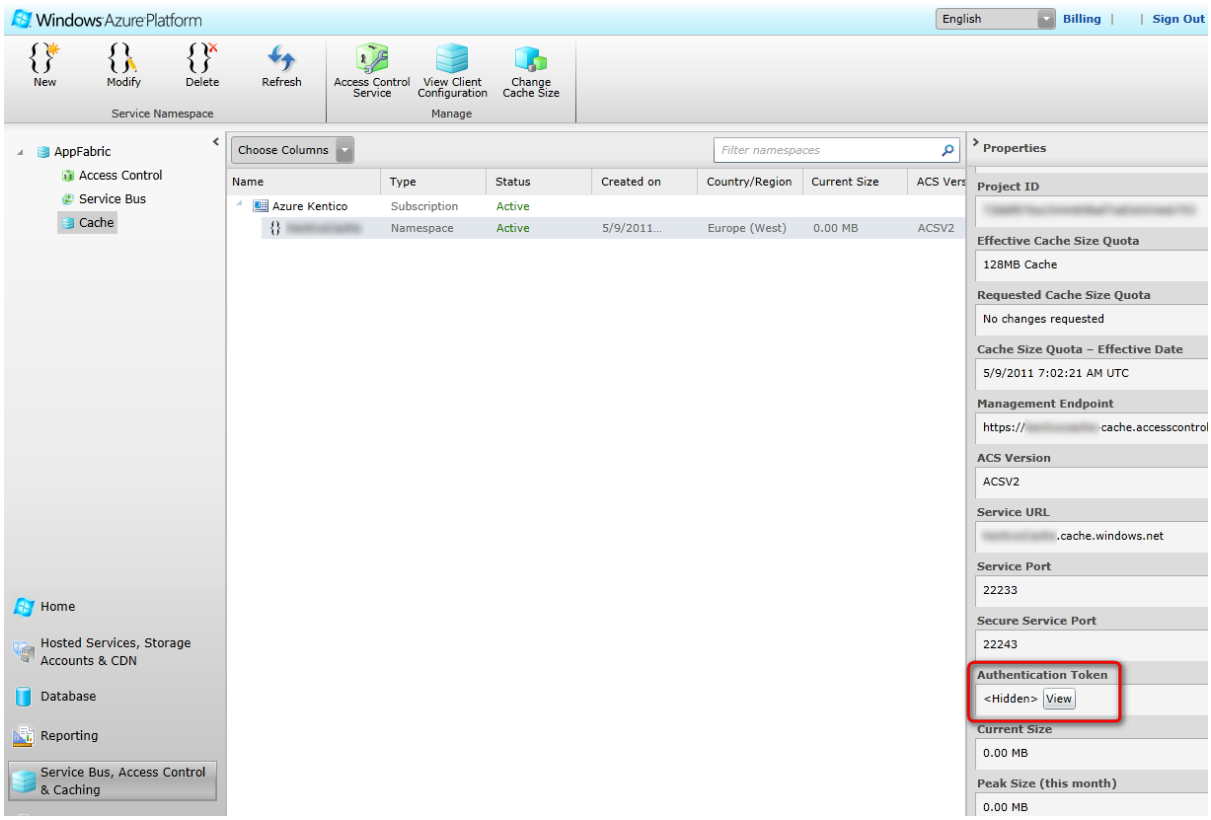
Do not increase the number of instances for the **SmartSearchWorker** role. Due to the way smart search indexes are processed, the required tasks must be performed by a single instance.

4. If you wish to use more than one instance for the application, you will also need a Windows Azure AppFabric Cache service namespace as part of your subscription. It is used to share the session state

between multiple instances.

The Windows Azure portal does not support managing AppFabric services. It is possible only in the old version of the management portal. To switch to the old version, click your account name in the top right corner of the portal and click **Previous portal**.

In the old portal, you can manage AppFabric services for your Azure subscription in **Service Bus, Access Control & Caching -> AppFabric -> Cache**. Select your cache namespace and view its properties on the right.



Scroll down to the **Authentication Token** property, view it and copy it to your clipboard. Next, open the **web.config** file of the **CMSApp** project and locate its *<dataCacheClients>* section. Here, enter the Service URL of your AppFabric cache namespace into the **name** attribute of the *<host>* elements and copy your authentication token into the *<messageSecurity>* **authorizationInfo** attributes.

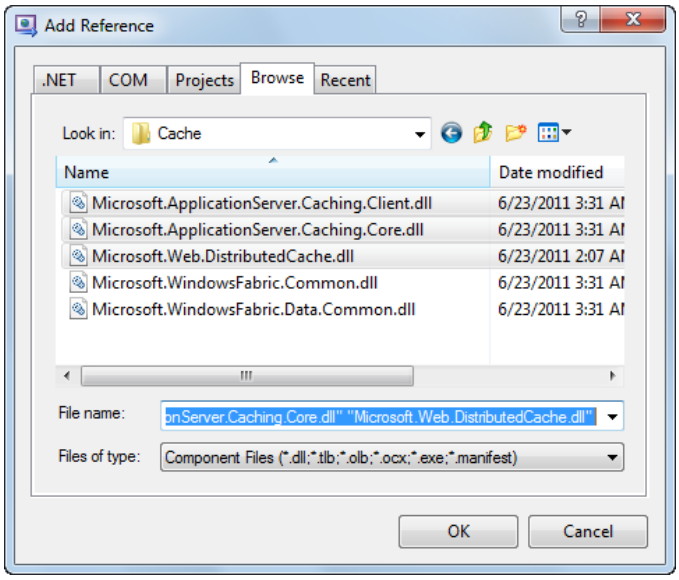
```
<!-- Azure AppFabric cache BEGIN -->
<dataCacheClients>
  <dataCacheClient name="default">
    <hosts>
      <host name="CacheNamespaceName.cache.windows.net" cachePort="22233" />
    </hosts>
    <securityProperties mode="Message">
      <messageSecurity authorizationInfo="AuthenticationToken">
      </messageSecurity>
    </securityProperties>
  </dataCacheClient>
```

```
  <dataCacheClient name="SslEndpoint">
    <hosts>
      <host name="CacheNamespaceName.cache.windows.net" cachePort="22243" />
    </hosts>
    <securityProperties mode="Message" sslEnabled="true">
      <messageSecurity authorizationInfo="AuthenticationToken">
      </messageSecurity>
    </securityProperties>
  </dataCacheClient>
</dataCacheClients>
<!-- Azure AppFabric cache END -->
```

Next, find the **<sessionState>** element under the <*system.web*> section, delete (or comment out) the default line of code and replace it with the version that uses the AppFabric cache that is included as a comment.

```
<system.web>

...

  <!-- Azure AppFabric cache BEGIN -->
  <sessionState mode="Custom" customProvider="AppFabricCacheSessionStoreProvider">
      <providers>
        <add name="AppFabricCacheSessionStoreProvider"
             type
="Microsoft.Web.DistributedCache.DistributedCacheSessionStateStoreProvider,
Microsoft.Web.DistributedCache"
             cacheName="default"
             useBlobMode="true"
             dataCacheClientName="default" />
      </providers>
  </sessionState>
  <!--  Azure AppFabric cache END -->

...

</system.web>
```

Then, right click the **CMSApp** project, select **Add reference**, switch to the **Browse** tab and navigate to the dlls in the AppFabric SDK installation directory (typically in *Program Files\Windows Azure AppFabric SDK\V1.0\Assemblies\NET4.0\Cache*).



Choose the *Microsoft.ApplicationServer.Caching.Client.dll*, *Microsoft.ApplicationServer.Caching.Core.dll* and *Microsoft.Web.DistributedCache.dll* files and click **OK**.

Other types of session state providers

Alternatively, you can set up a Blob/table or SQL Azure provider to maintain the session state if you do not wish to use AppFabric cache.

Please see the article about Session State on Windows Azure published on our DevNet portal for more information.

5. Set the value of the **CMSURLPort** setting in the service configuration file to *80*.

```
<Setting name="CMSUrlPort" value="80" />
```

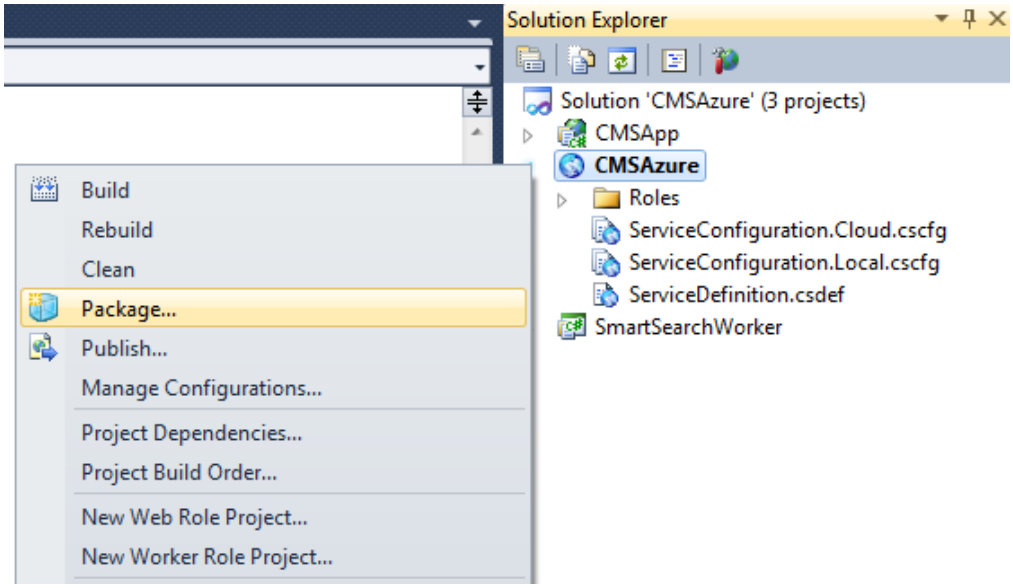
The value of this setting should always match the **port** attribute of the <*InputEndpoint*> element under the **CMSApp** role in your service definition file (**ServiceDefinition.csdef**). If you need to use a different port for some reason, adjust the values in both the service definition and configuration.

6. At this point, you can also choose if you wish to run the application in *Full IIS* mode or under *Hosted Web Core* (legacy mode). Full IIS mode is set by default and recommended, since it allows you to use the full range of IIS features, such as Windows Authentication. If you wish to use Hosted Web Core, you can do so by deleting or commenting out the <*Sites*> section of the **CMSApp** role in the service definition file. Also, edit the **AzureInit.cs** file in the **Old_App_code/CMSModules/WindowsAzure** folder and comment out the content of its *ApplicationStartInit()* and *BeginRequestInit()* methods.

Once everything is configured appropriately, save the changes to all files and you can perform the actual deployment of the application. Please continue reading in the Deployment to the cloud topic to learn how this can be done.

2.7 Deployment to the cloud

When you are ready to deploy your website to Windows Azure, you can do so from Visual Studio using the Windows Azure Tools. Right-click the **CMSAzure** project in the Solution Explorer and select either the **Package** or **Publish** option.



Packaging creates a pair of deployment files representing the application and its configuration, which you can upload manually through the Azure Management Portal. The **Publish** option allows you to directly upload the application as a hosted service deployment. With this approach, you will need to enter the credentials for your Windows Azure subscription into Visual Studio, so that it can authenticate itself when deploying the application.

You can find additional information about publishing options in the Publishing the Windows Azure Application using the Windows Azure Tools MSDN article.

Before you perform the deployment, please make sure that your application is configured correctly as described in the Application structure and configuration topic. Also, make sure that the correct service configuration intended for cloud deployment is selected in the publishing options.

Deploying the application as a service package

If you choose the **Package** option, Visual Studio will build the solution and create two files: **CMSAzure.cspkg** and **ServiceConfiguration.cscfg**. The packaging process may take some time, so please be patient. When complete, the directory containing the files will automatically be opened (by default, the target folder is *~/app.publish*). You can then upload the service package via the Azure Management Portal (https://windows.azure.com/).

Accessing the Management Portal

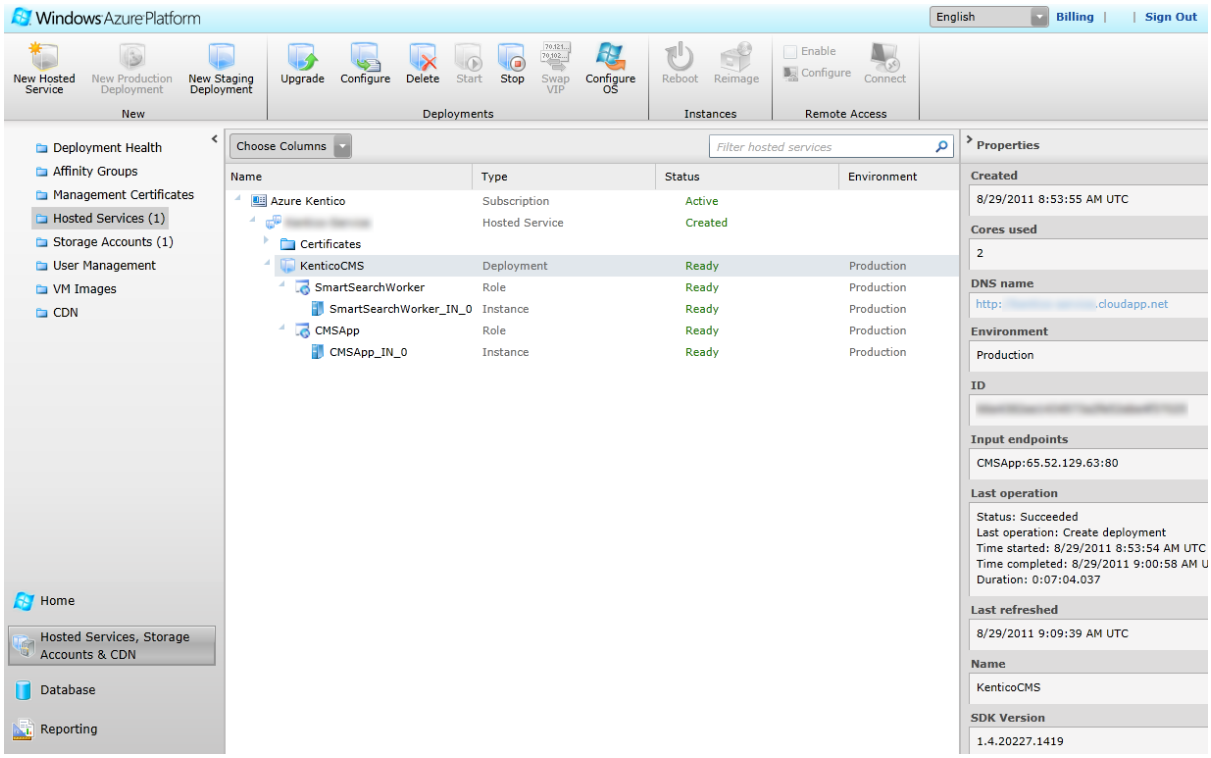
This chapter describes the old version of the Windows Azure Management Portal. To access the old portal from the new portal, click your account name in the top right

corner of the new portal and click **Previous portal**.

Go to the old portal directly

Navigate to **Hosted services, Storage Accounts & CDN** and select **Hosted Services**. If you do not yet have a hosted service for the website, click **New Hosted Service** to create one and configure it as necessary. Select the appropriate service and click **New Production Deployment** or **New Staging Deployment** in the top ribbon menu. The choice between these two determines under which address the application will be available. The production deployment uses a URL in format *<hosted service URL prefix>.cloudapp.net*, while staging deployments are available at *<Deployment ID>.cloudapp.net*. The Deployment ID of a staging deployment is generated automatically upon creation. Staging deployments are meant for development or testing purposes and the production deployment for running the live version of your application.

Once you select one of the options, a form will be displayed where you can specify the service package (**.cspkg**) and configuration (**.cscfg**) files that were published on your local machine. When done, click **OK** to begin uploading the application to the hosted service.



When your deployment is ready, press **Start** to run the application (if the hosted service is not automatically started after a successful deployment).

The website will now be available under the URL displayed in the **DNS name** property of the deployment. If you open the site in a new tab of your browser, the application's database installer will start up. Please continue in the Database setup topic

Domain names of Windows Azure websites

In most cases, you will probably want to use a different domain name than the default one set for the production deployment. To achieve this, use your domain name registrar to create a CNAME record that maps a custom domain name to the **DNS name** of your application on Windows Azure.

The same applies if you are running multiple sites on your Kentico CMS instance. Define a CNAME record for each site's domain name and map it to your production deployment's DNS name. The appropriate site will be loaded automatically.

2.8 Database setup

A Kentico CMS website deployed to Windows Azure can use any type of SQL database, either hosted on a standard SQL Server or in the cloud. This topic describes how the database installation can be performed on an SQL Azure server, which is the database service of the Windows Azure platform.

Accessing the Management Portal

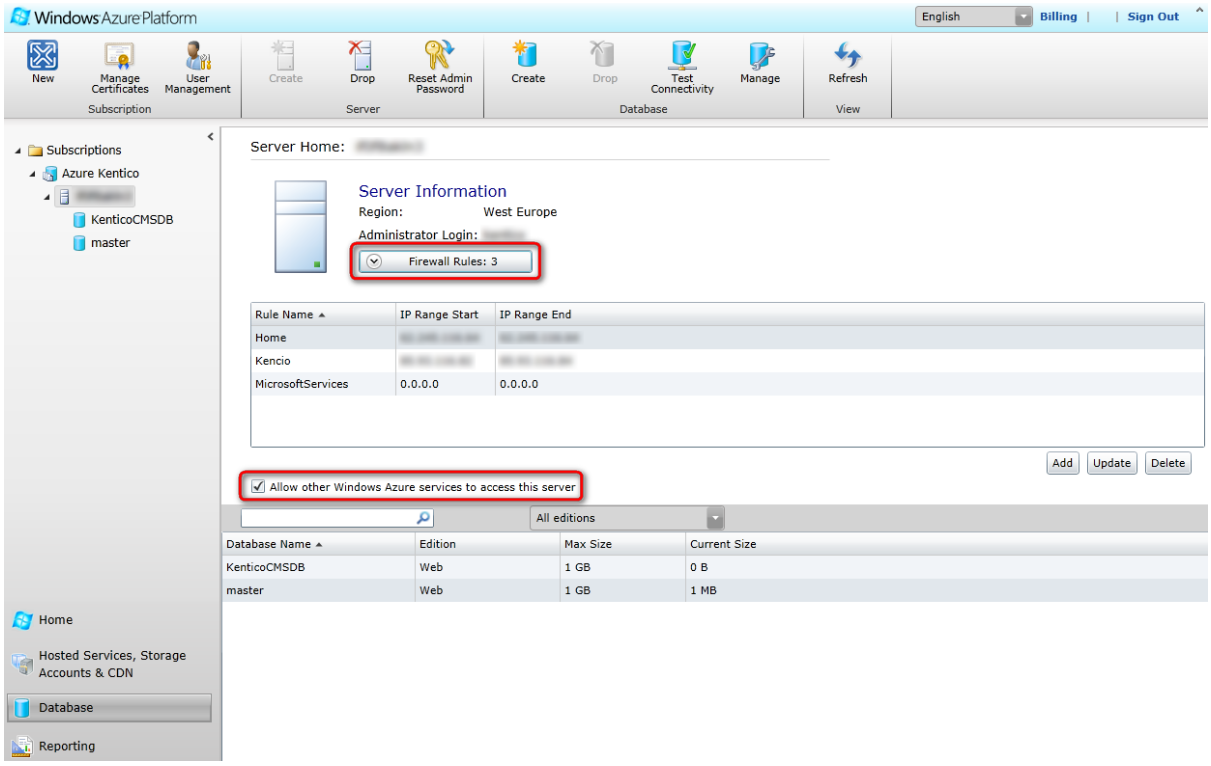
This chapter describes the old version of the Windows Azure Management Portal. To access the old portal from the new portal, click your account name in the top right corner of the new portal and click **Previous portal**.

Go to the old portal directly

Setting up an SQL Azure server and database

You can view and manage your SQL Azure database servers through the Windows Azure Management Portal (https://windows.azure.com/) by clicking **Database** in the main navigation menu. If you do not have a server yet, click the **Create** button in the *Server* section of the ribbon menu at the top of the page, then select a region for the server and set up its credentials.

Then select the server and configure its **Firewall Rules**. First, enable **Allow other Windows Azure services to access this service**, which means that the server will be accessible by your Kentico CMS application when it is deployed as a hosted service. Next, add rules for any IP ranges that include your development or administration machines, so that you can connect to the server when using the local emulator or manually (e.g. with SQL Server Management Studio).



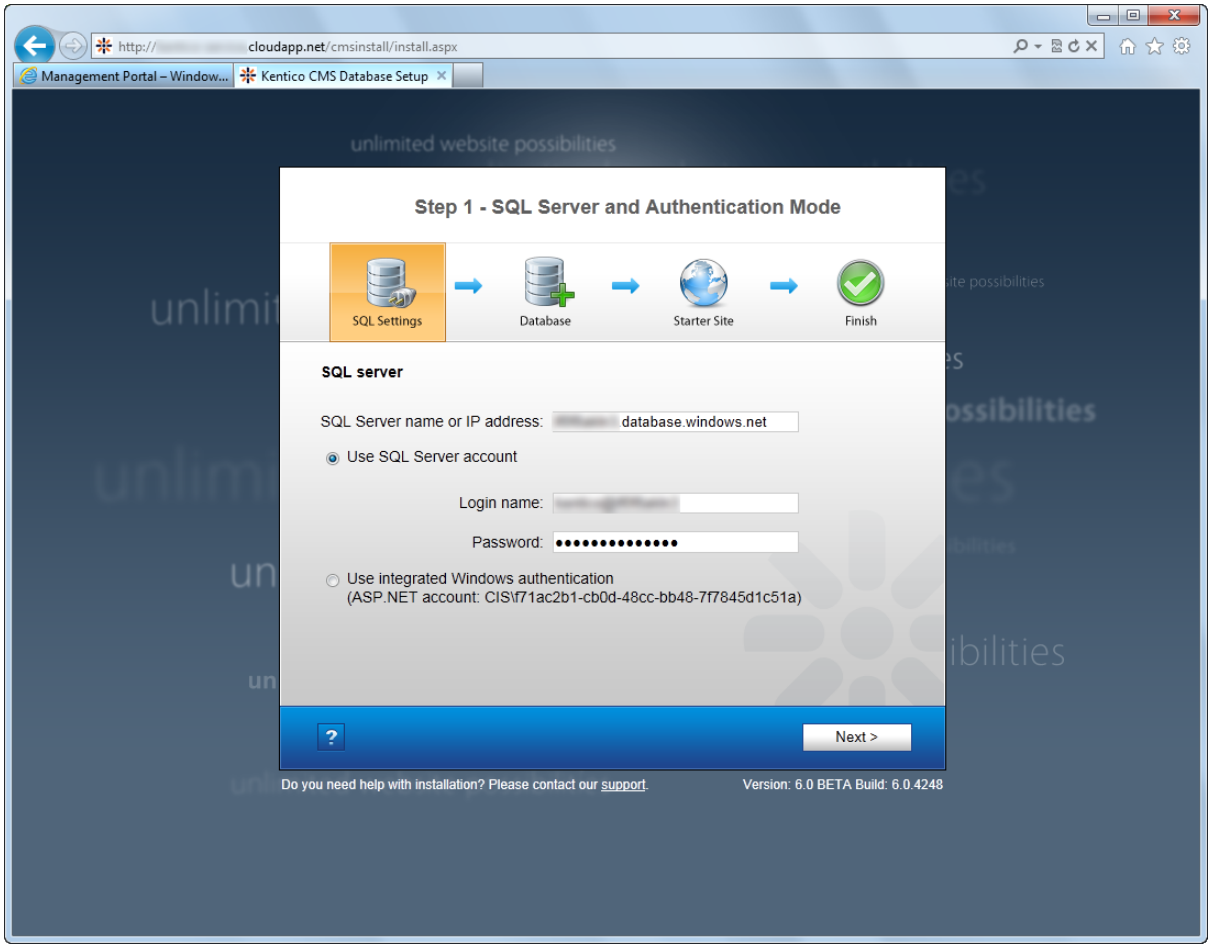
Finally, click **Create** in the *Database* section of the menu to add a database under the server that will be used for the application. Choose an appropriate edition and size, depending on the requirements of your website.

Database installer

Once your application is successfully deployed to Windows Azure (or started up on the local emulator) you can open it in your browser, where the database installer will automatically begin. This process will create the database tables required by Kentico CMS on the specified SQL database. The following steps must be taken to complete the installation:

1. In the first step, you need to specify the target server and enter the credentials used to access it.

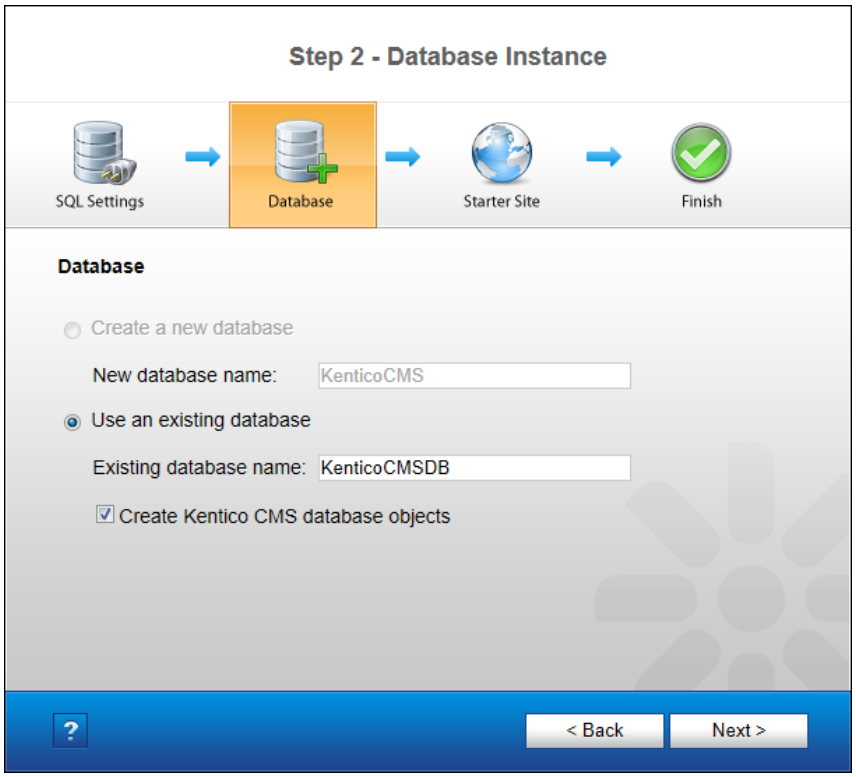
- **SQL Server name or IP address** - enter the fully qualified server name of your SQL Azure server. The standard format is: *<server name>.database.windows.net*
- **Use SQL Server account** - check this option and type in a valid *Login name* and *Password* for the server. The name is entered using the format *Login@ServerName.*



Click the **Next** button.

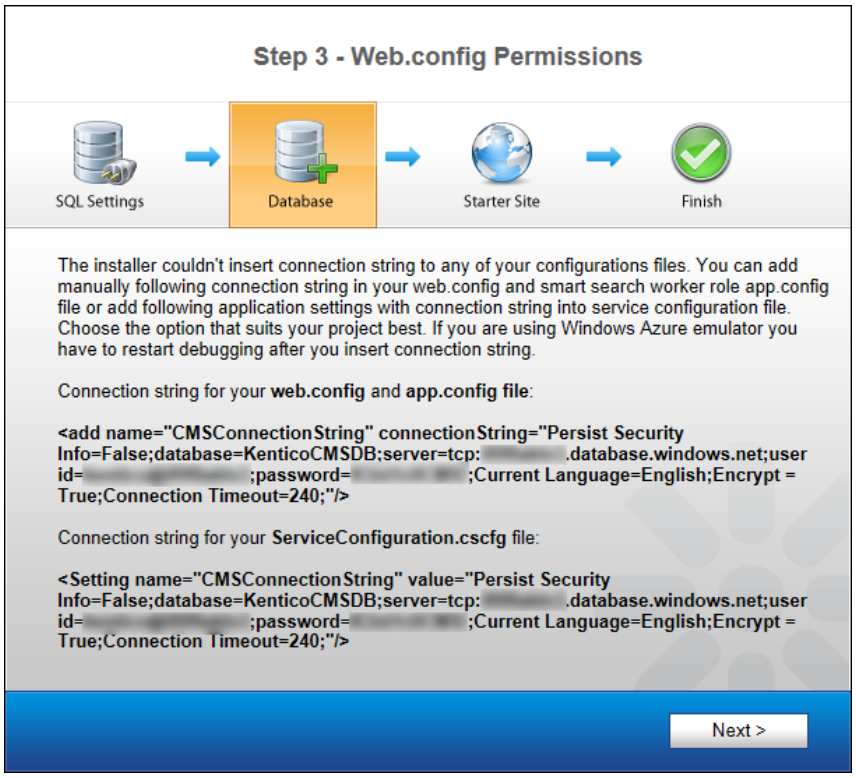
2. In the next step, enter the name of the database into the **Existing database name** field. Kentico CMS cannot create a new database on an SQL Azure server, so you will have to use a predefined database (you can define an empty one for a clean installation as described in the section above).

You can also choose whether you want to **Create Kentico CMS database objects**. If the existing database already contains Kentico CMS objects (tables, stored procedures, views), you can uncheck the box. If the database does not contain these objects, i.e. when you are installing into an empty database, you need to leave the option enabled.

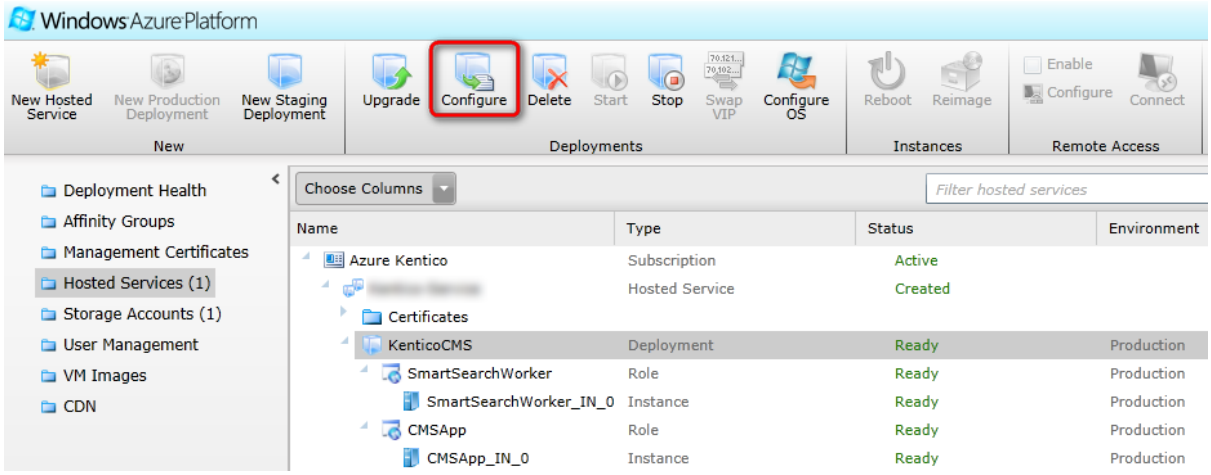


Click **Next.**

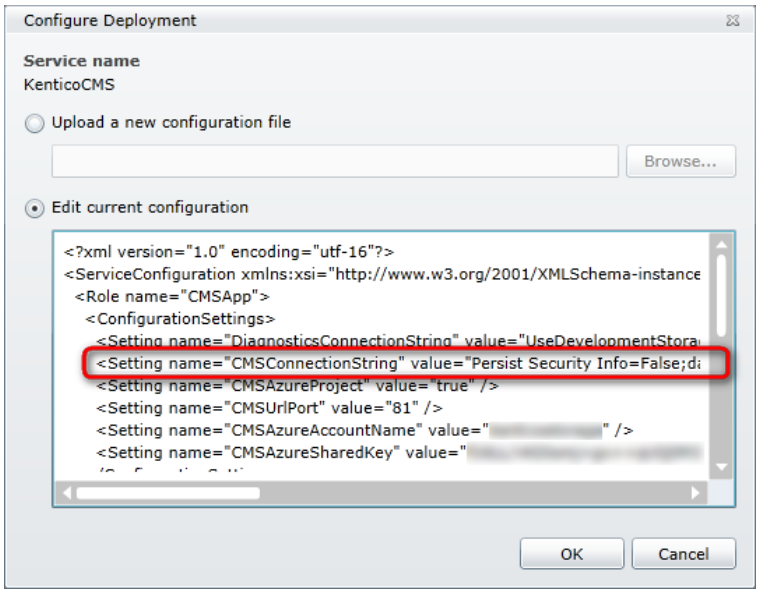
3. The installer will now generate connection strings for the specified database and server. There are two possible ways to set the connection string for the application. It may either be added as a key into the <*connectionStrings*> section of the **web.config** and **app.config** files of the **CMSApp** and **SmartSearchWorker** projects respectively, or it can be included as a setting in the application's service configuration file.



The easiest way is to copy the service configuration version of the connection string (the second one) and switch back to the Azure Management Portal. Go to **Hosted Services, Storage Accounts & CDN -> Hosted Services**, select your Kentico CMS deployment and click the **Configure** button.

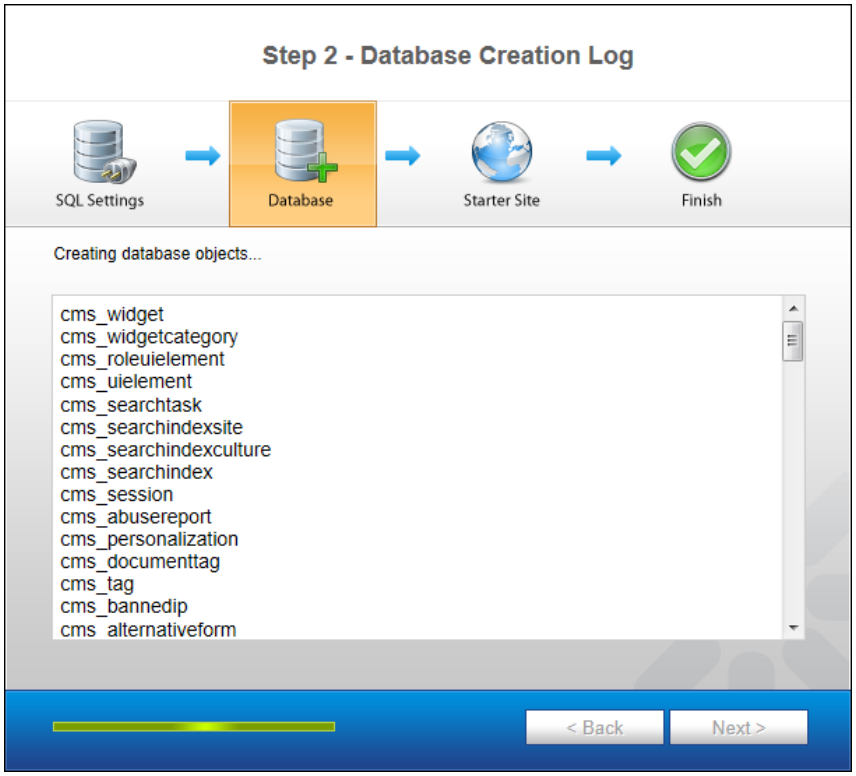


Select **Edit current configuration** in the displayed dialog box and insert the connection string setting from the database installer instead of the default **CMSConnectionString** setting. Do this in the <*ConfigurationSettings*> section of both the **CMSApp** and **SmartSearchWorker** roles.



Click **OK** and wait until the deployment is updated.

4. Close the original tab with the database installer and open the website again once it is ready (click on the link in the **DNS name** property of your deployment). The installer will automatically continue the previous installation. Confirm the target database by clicking **Next** and the installation will start. A log will be displayed, showing the progress of the database creation. After it finishes, you will be moved forward to the next step.



5. If you are installing on a production deployment, you will be asked to insert a license key for the domain. Staging deployments are automatically covered by any other valid license (including the temporary trial license), so you do not have to request a separate license for each unique deployment. The same applies to the local emulator, since it runs on the *localhost* (127.0.0.1) domain, which may be used with any license, including the trial.

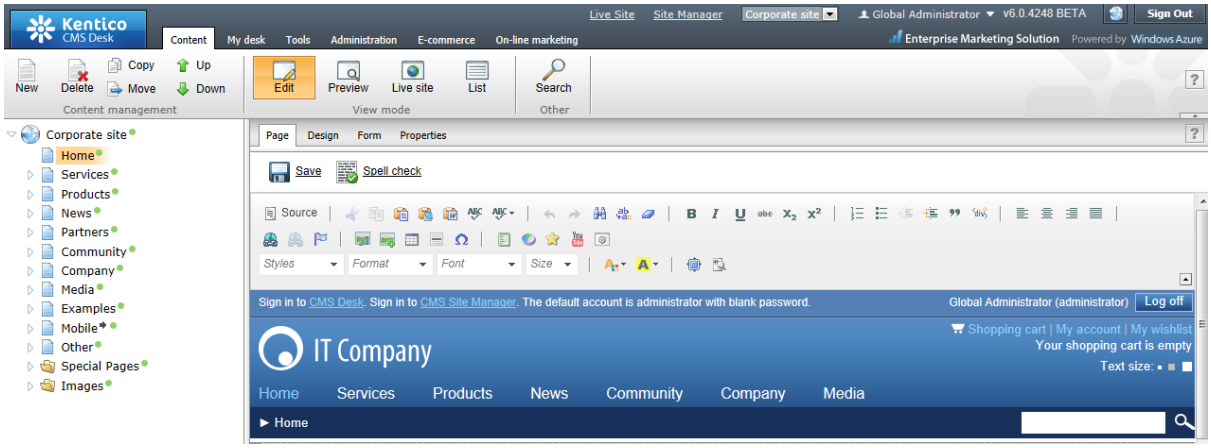
If necessary, enter a valid license key and click the **Next** button.

6. On the next screen, you will be offered the following options to set up an initial website:

- **Choose a starter site -** you can choose one of the sample sites provided with Kentico CMS to try out the features or use it as a base for the development of your own website.
- **Continue to the New site wizard** - this option is recommended if you wish to develop a new site from scratch.
- **Import an existing Kentico CMS website** - use this option if you have already created a website with Kentico CMS and need to import it into the new installation.

If you are new to Kentico CMS, it is highly recommended to start with the sample *Corporate Site*. Select an option and click the **Next** button. A log will be displayed showing the progress of the website creation.

7. Once the website is created, a confirmation will be displayed and you can follow the link to the live website. From here, you can log into the **CMS Desk** or **Site Manage** administration interface (the sample sites use the *administrator* user name and a blank password by default) and start working with Kentico CMS.



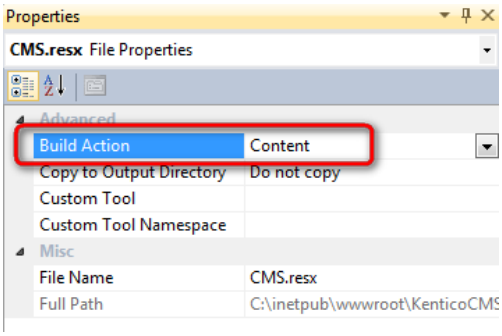
2.9 Deploying an existing website to Windows Azure

If you already have a Kentico CMS website hosted on a traditional web server and have decided to move it to the Windows Azure platform, there are several tasks that need to be performed in order to prepare the site for deployment.

The recommended method is to install a new instance of Kentico CMS as an Azure project using the Web installer. Then, carry over all customizations from your original project into the **CMSApp** project of the new application. Remember to check the configuration options specified in your site's **web.config** file and transfer them into the web.config of the new project. It contains some additional sections needed to run on Windows Azure, so simply copying the file is not an option.

You can enter the connection string for your site's database either into the web.config file as usual, or the service configuration file under the application's **CMSAzure** project. Please note that your database must use the same version of Kentico CMS as the installed Azure project. If this is not the case, you will have to perform the appropriate upgrade procedure for your original website, which also upgrades the database. You can download the upgrade package from the Support section of the *Kentico.com* website. If you also wish to have your database hosted on Windows Azure, please see the section below to learn how this can be done.

You will also need to copy any additional files required by the site (such as custom user controls or stylesheet design files). Once the files are present in the new application's directory, include them in the project. To do this in Visual Studio, enable the **Show all files** button at the top of the Solution Explorer, then right click the given files and select **Include In Project**. Also check the properties (**Right click -> Properties**) of all files that were included into the project and make sure that the value of their **Build Action** property is set to *Content*.



This is necessary to ensure that the files will be part of the application once it is deployed to Windows Azure.

Once all customizations and required files are included in the project, you can configure it as described in the Application structure and configuration topic and then follow the instructions in Deployment to the cloud to complete the deployment of the application.

Another possible approach is to manually convert your existing website to a Windows Azure application as described in the Converting a web site project to an Azure application topic. This is a relatively complicated process that should only be considered if your project is heavily customized and carrying over the changes is not a suitable option.

Deploying media libraries

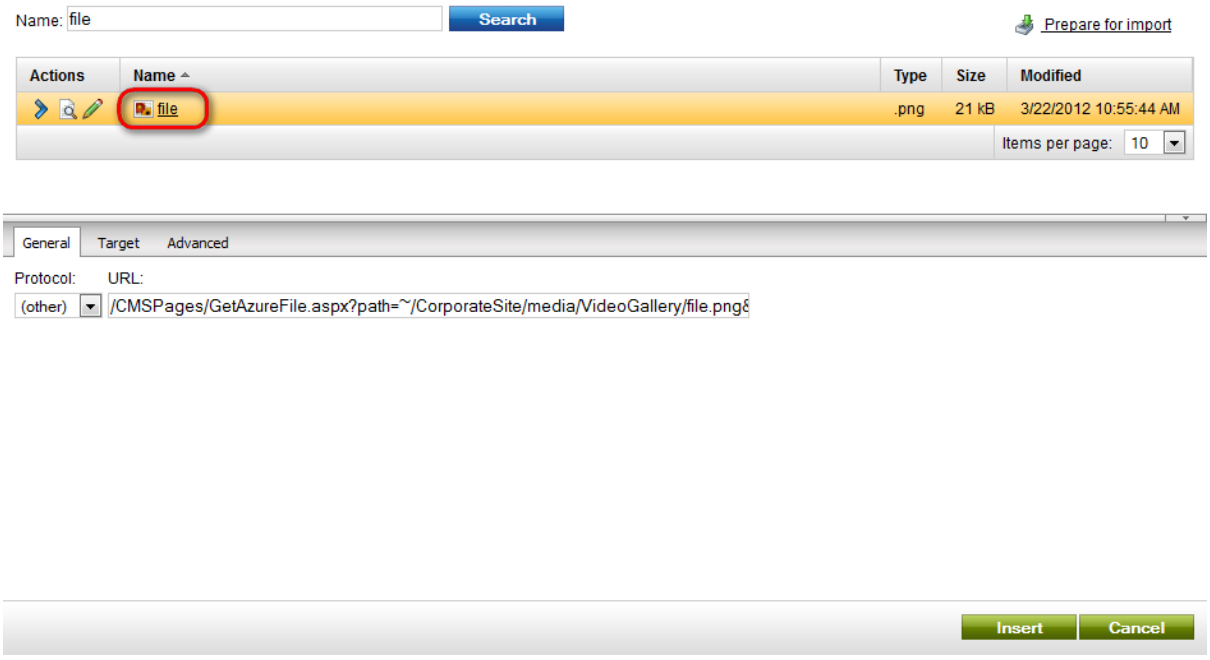
Media files are organized into folders in the file system. If you have existing libraries on your website, you will have to include those media folders into the project, as mentioned previously. This approach ensures that your existing media libraries will be included in the package that will be deployed to the cloud. However, files that will be added while the website runs in Azure, will automatically be uploaded to the Blob storage. For this reason, it is recommended to move media files to the Blob storage at the time of deployment, so that both the existing files and the files that will be added in the future will share a common storage.

The move to Blob storage will, however, cause the files' URLs to change. You will have to manually go through your website's content and change links pointing to media files to the new, correct URLs. To do this, follow these steps:

1. Right-click the particular link.

2. Click **Link properties**.

3. In the dialog that opens, search for the file you're linking and click it. This will regenerate the file's URL.



4. Save the changes by clicking **Insert**.

Migrating your database to SQL Azure

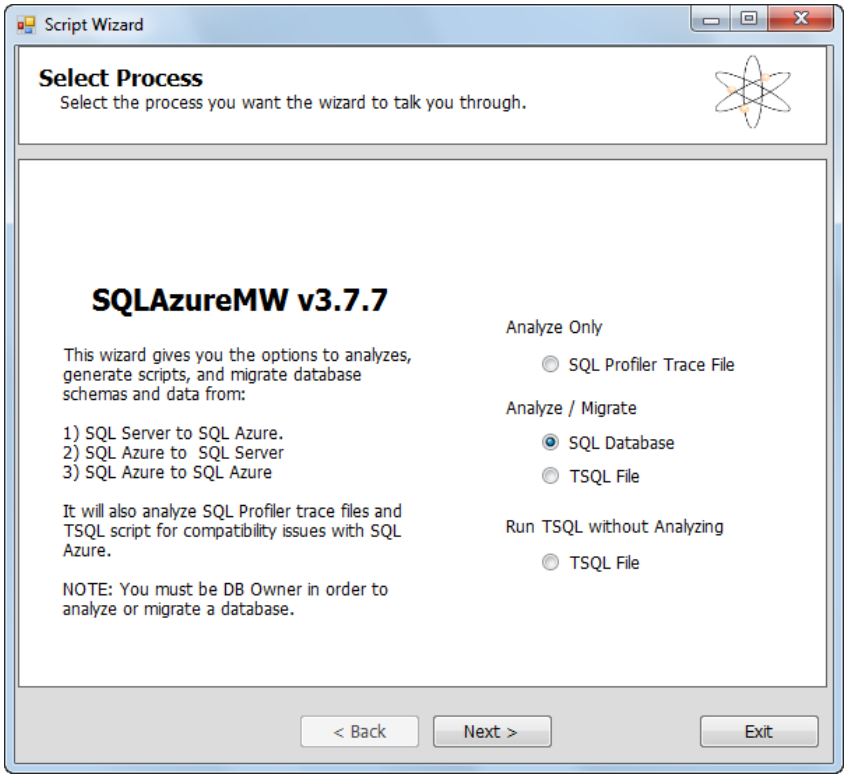
The SQL Azure service may be utilized to host your website's database on the Windows Azure platform along with the application. Transferring your database to SQL Azure is a possibility even if you wish to keep your Kentico CMS website hosted on a standard server.

The procedure requires your database to run on **Microsoft SQL Server 2008 R2 SP1** or later. The Express edition is sufficient.

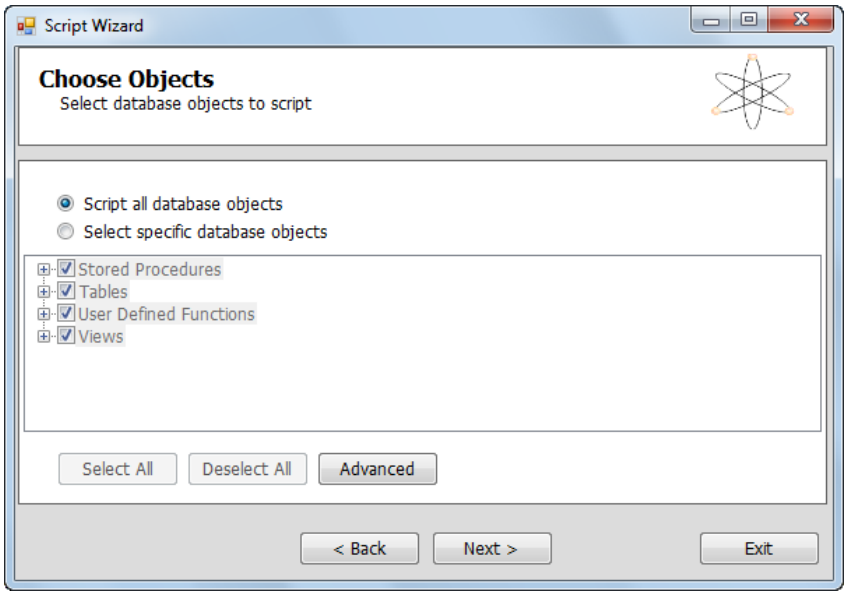
1. First, it is necessary to prepare the target SQL Azure database. If you do not have an available server and database yet, please see the Database setup topic to learn how you can set these up through the Windows Azure Management Portal.

2. You can either migrate your database manually by executing a script on the SQL Azure database, or use an appropriate tool or integration service to make this process easier. We recommend the SQL Azure Migration Wizard, which is available free of charge at http://sqlazuremw.codeplex.com/.

3. Download the SQL Azure Migration Wizard tool and run it.

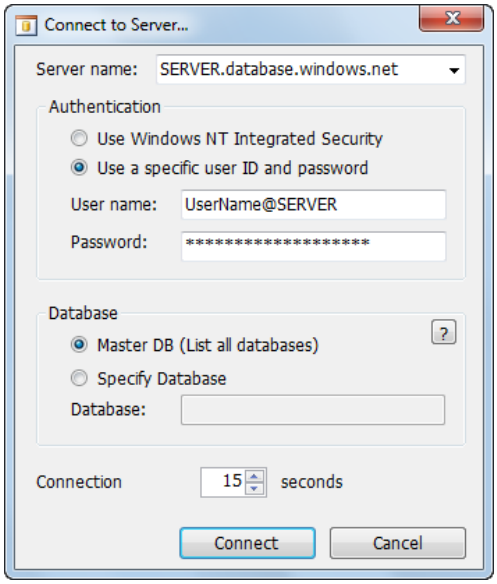


Select the **Analyze / Migrate -> SQL Database** option, click **Next** and choose the server where the database of your Kentico CMS website is located as the source. Select the specific database by double-clicking on it and then the **Script all database objects** option.



Click **Next** and confirm that you wish to generate the required SQL script.

4. When the script is completed, continue to the next step where you can enter the required connection information for your SQL Azure server.



Connect to the server and specify the target database. The tool will then execute the generated script. If successful, your database will be migrated to the cloud.

5. Now that your database is hosted on SQL Azure, change your application's connection string to match the new database. If you have Kentico CMS installed (or converted) as a Windows Azure project, you can add the connection string as a setting into the <*ConfigurationSettings*> section of both roles in the appropriate service configuration file. With this approach, use the following format:

```
<Setting name="CMSConnectionString" value="Persist Security Info=False;
database=DatabaseName;server=tcp:ServerName.database.windows.net;user
id=Login@ServerName;password=Password;Current Language=English;Encrypt=True;
Connection Timeout=240;" />
```

Alternatively, you may specify the database through the standard <*connectionStrings*> section of the application's **web.config** file:

```
<add name="CMSConnectionString" connectionString="Persist Security Info=False;
database=DatabaseName;server=tcp:ServerName.database.windows.net;user
id=Login@ServerName;password=Password;Current Language=English;Encrypt=True;
Connection Timeout=240;" />
```

2.10 Converting a web site project to an Azure application

In cases where your application is heavily modified or contains large amounts of custom code, you may find it easier to manually convert it to a Windows Azure application, rather than using the recommended approach described in Deploying an existing website to Windows Azure. This way, you can be sure that nothing is left out, since you will only be adding extra items to your project.

You can use the following steps to perform the conversion process:

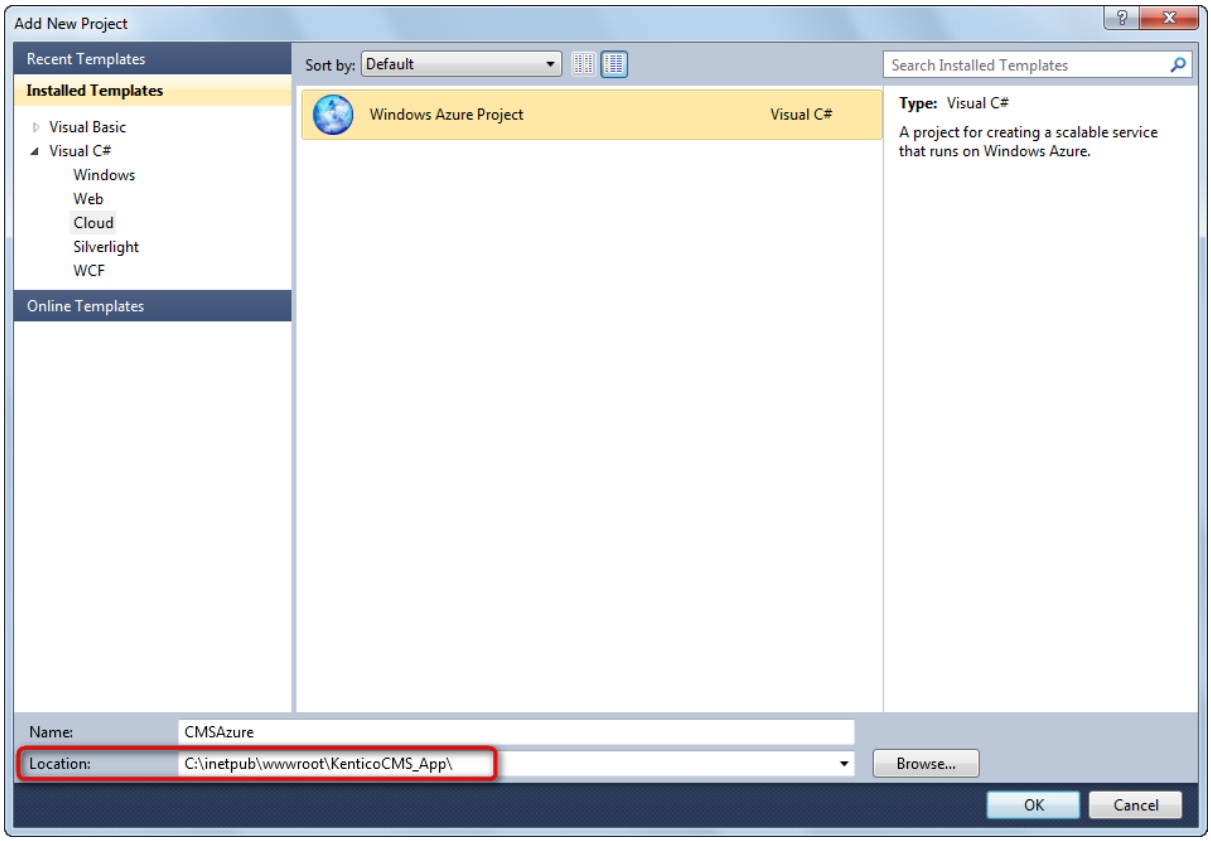
1. If needed, perform the upgrade procedure to the latest version of Kentico CMS. You can download the upgrade package from the Support section of the *Kentico.com* website.

2. Before you can continue, it is necessary to have your Kentico CMS project in the web application format. If your project was not installed this way, you can convert it according to the instructions in the How to convert a Kentico CMS web site to a Web Application article published on our DevNet portal.

3. Open your solution in Visual Studio, right-click the project file in the Solution Explorer and select **Unload Project**. If the solution only contains a single project, you may have to go to **Tools -> Options -> Projects and Solutions -> General** and check **Always show solution** to make this option available. Now right-click the project again and select **Edit <project name>**. Add `<RoleType>Web</RoleType>` into the first *<PropertyGroup>* element of the project file.

Save the change, right click the project in the Solution Explorer and choose **Reload Project**.

4. Next, **Add** a **New Project** to the solution and select the **Windows Azure Project** template from the **Cloud** category. Specify the path of the directory containing your project files as the **Location** and click **OK**.



Confirm the role selection dialog that appears next without adding any roles to the Windows Azure solution. The necessary roles will be added manually later. When managing your application, always make sure that this is the start up project of your solution (right click it and click **Set as StartUp Project**).

5. Now install a new instance of Kentico CMS as an Azure project to another directory using the Web installer. This will be used as a source for the files required to run on Windows Azure.

6. Copy the **SmartSearchWorker** folder into your project directory, return back to Visual Studio, right click your solution, select **Add -> Existing project** and choose the **SmartSearchWorker.csproj** file in the given folder. You can skip this step if you are certain that you will not need the Smart search module. This reduces the number of roles that need to be hosted, so the costs of running your application on Windows Azure will be lower. In this case, you can ignore any further instructions related to this project (and worker role).

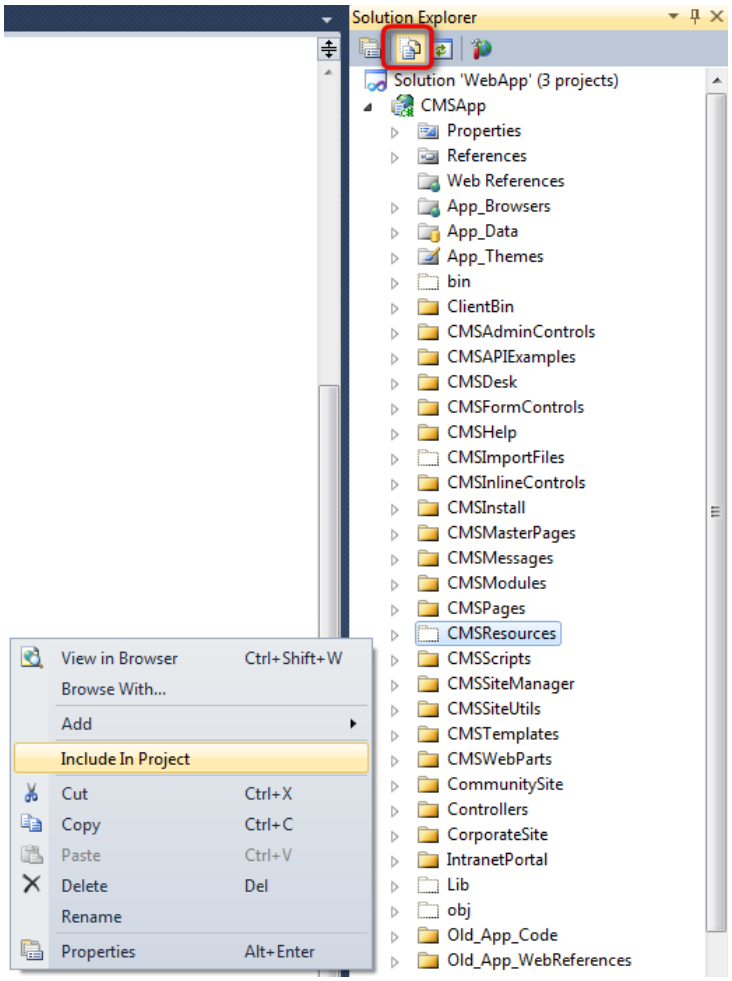
7. Next, copy the content of the Azure project's **bin** folder (dlls and other files) into the **bin** folder of your web application and overwrite the previous versions of the files (if you are using the source code version of Kentico CMS, you only need to add the **CMS.AzureStorage** assembly to your solution).

8. You also have to add the appropriate references to your original Kentico CMS project. Right click it in the Solution Explorer, select **Add Reference** and perform the following actions:

- **Projects** tab - add a reference to the **SmartSearchWorker** project.
- **Browse** tab - expand your application's *bin* folder and select **CMS.AzureStorage.dll**.
- **.NET** tab - add references to the items listed below:
  - Microsoft.WindowsAzure.CloudDrive
  - Microsoft.WindowsAzure.Diagnostics
  - Microsoft.WindowsAzure.ServiceRuntime
  - Microsoft.WindowsAzure.StorageClient

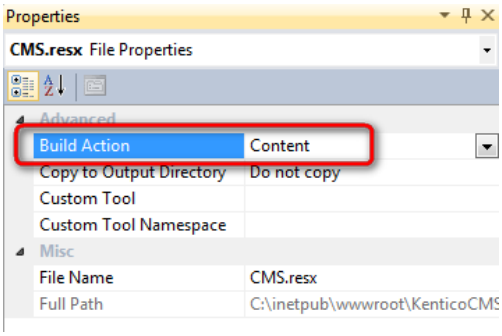
9. There are also some additional files that must be included in your project to ensure that it works correctly on Windows Azure. Copy **WebRole.cs** from the root of the **CMSApp** project of the new Azure installation and the **GetAzureFile.aspx** page from its **CMSPages** directory and insert them to the corresponding location under your application using **Add -> Existing item**. Also, replace your existing **~/Old_App_code/CMSModules/WindowsAzure/AzureInit.cs** file with the one from the Azure project.

Then enable the **Show all files** button at the top of the Solution Explorer, find the **Global.asax** file under the root of the project and make sure that it is included. If this is not the case, right click it and select **Include In Project**.



Do the same for the **CMSResources** and **CMSHelp** folders, and the content of the **App_Data/Templates** folder (even if you do not need the sample website templates, it is necessary to include at least the **BlankSite** sub-folder).

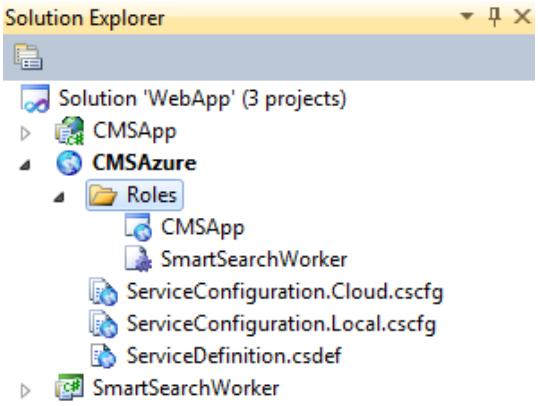
Also check the Properties window (**Right click -> Properties**) of all files that were included into the project and make sure that the value of their **Build Action** property is set to *Content*.



Keep in mind that any other custom files or folders that are used by your website must be included in the project in this way to ensure that they are part of the application once it is deployed to Windows Azure.

10. Create a copy of the web.config file from the instance installed as an Azure project and carry over any changes that you made to the web.config of your existing application. Once this new file contains both the configuration for your application and the sections required to run on Windows Azure, use it to replace your original web.config.

11. The next step is to define your projects as roles in the Windows Azure application. To do this, right-click the **Roles** folder under the Azure project, select **Add -> Web Role Project in solution** and choose your Kentico CMS web application project. Next, add another role, but select **Add -> Worker Role Project in solution** instead and choose the **SmartSearchWorker** project. Your solution should now look like the one in the image below.



12. Edit the service definition file (**ServiceDefinition.csdef**) of the Azure project and modify the definition of your web role (the name must match the name of your Kentico CMS project):

```
<WebRole name="CMSApp">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="HttpIn" endpointName="HttpIn" />
        </Bindings>
      </Site>
    </Sites>
    <ConfigurationSettings>
      <Setting name="DiagnosticsConnectionString" />
      <Setting name="CMSConnectionString" />
      <Setting name="CMSAzureProject" />
      <Setting name="CMSUrlPort" />
      <Setting name="CMSAzureAccountName" />
      <Setting name="CMSAzureSharedKey" />
      <Setting name="CMSAzureQueueEndPoint" />
      <Setting name="CMSAzureTableEndPoint" />
      <Setting name="CMSAzureBlobEndPoint" />
    </ConfigurationSettings>
    <Endpoints>
      <InputEndpoint name="HttpIn" protocol="http" port="80" />
      <InternalEndpoint name="InternalHttpIn" protocol="http" port="8080" />
    </Endpoints>
    <LocalResources>
      <LocalStorage name="CMSTemp" cleanOnRoleRecycle="true" />
      <LocalStorage name="CMSCache" cleanOnRoleRecycle="false" />
    </LocalResources>
</WebRole>
```

The <*Sites*> section ensures that the application is deployed in *Full IIS* mode (recommended). You can delete or comment it out if you wish to run under *Hosted Web Core*. The <*ConfigurationSettings*> section declares the settings that will be available in the service configuration file.

Next, modify the definition of the **SmartSearchWorker** role as shown below:

```
<WorkerRole name="SmartSearchWorker">
    <ConfigurationSettings>
      <Setting name="CMSConnectionString" />
      <Setting name="CMSAzureProject" />
      <Setting name="CMSUrlPort" />
      <Setting name="CMSAzureAccountName" />
      <Setting name="CMSAzureSharedKey" />
      <Setting name="CMSAzureQueueEndPoint" />
      <Setting name="CMSAzureTableEndPoint" />
      <Setting name="CMSAzureBlobEndPoint" />
    </ConfigurationSettings>
    <LocalResources>
      <LocalStorage name="CMSTemp" cleanOnRoleRecycle="true" />
      <LocalStorage name="CMSCache" cleanOnRoleRecycle="false" />
    </LocalResources>
    <Imports>
      <Import moduleName="Diagnostics" />
```

```
    </Imports>
</WorkerRole>
```

13. Now edit the service configuration files (**.cscfg**) and add the following settings for the web role:

```
<ConfigurationSettings>
    <Setting name="DiagnosticsConnectionString" value="UseDevelopmentStorage=true" /
    <Setting name="CMSConnectionString" value="" />
    <Setting name="CMSAzureProject" value="true"/>
    <Setting name="CMSUrlPort" value="80" />
    <Setting name="CMSAzureAccountName" value=""/>
    <Setting name="CMSAzureSharedKey" value=""/>
    <Setting name="CMSAzureQueueEndPoint" value=""/>
    <Setting name="CMSAzureTableEndPoint" value=""/>
    <Setting name="CMSAzureBlobEndPoint" value=""/>
</ConfigurationSettings>
```

Also add the same settings under the **SmartSearchWorker** role, but without the **DiagnosticsConnectionString** (it should already contain the default setting for this purpose).

You may have multiple service configuration files for your application, which can be used for different deployment types. These files may contain different values, but all of them must include the settings declared in the service definition.

14. To complete the deployment of your application, configure it as described in Application structure and configuration and then follow the instructions in Deployment to the cloud.

2.11 Running multiple instances in one web role

Kentico CMS allows you to run more than one web project instance in a single Azure Web role. To enable this functionality, you only need to make a few adjustments to the service definition file. This topic describes the steps you need to take in order to add another website project to your Azure application.

1. Add another site to the **<Sites>** section of your service definition file, as highlighted in the following code example.

```
<Sites>
  <Site name="Web">
    <Bindings>
      <Binding name="HttpIn" endPointName="HttpIn" />
    </Bindings>
  </Site>

  <Site name="Web2" physicalDirectory="c:\inetpub\wwwroot\KenticoCMS_SecondSite">
    <Bindings>
      <Binding name="HttpIn" endPointName="HttpIn" hostHeader="www.example.com" /
>
    </Bindings>
```

```
  </Site>
</Sites>
```

2. Set the **physicalDirectory** parameter to the path to the directory where you have your second website project installed. You don't need to provide this parameter to the first site, because its default name "Web" instructs Visual Studio that the site definition is referring to the Web role project.

3. Specify a **hostHeader** for the second site. Similarly to IIS, two sites cannot listen on the same port without having a host header specified.

Running on an emulator

If you want the two sites to run on your local machine using the emulator, you will need to take the following additional steps:

1. Edit the **hosts** file from the following directory: *<%SYSTEMROOT%>\System32\drivers\etc*.

2. Add mappings of your local IP address to host names, which you specified for sites in the service definition file. In the case of the provided example code, you would add the following line at the end of the file:

```
127.0.0.1 www.example.com
```

Deploying to the cloud

There's no difference between deploying a standard application and deploying an application that runs more Kentico CMS instances in a Web role. To deploy your application, follow the steps described in the Deployment to the cloud topic.

2.12 Configuring external services

By default, external Windows services that come with Kentico CMS 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 topic, the services - Scheduler and Health monitor - will run as part of the **Smart search worker role**.

To enable external services on your Azure application, follow these steps:

1. Open the **ServiceDefinition.csdef** file and uncomment the following code:

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

2. Open the **InstallService.cmd** file from the **SmartSearchWorker** project and on the following line

replace <*YourAppGuid*> with the value of the **CMSApplicationGuid** key. You can find the key in the **appSettings** section of the **Web.config** file in the **CMSApp** project.

```
SET _guid=<YourAppGuid>
```

For example, if the **CMSApplicationGuid** key looks like this:

```
<add key="CMSApplicationGuid" value="77b0671d-ed31-4679-9bd3-fce7e245d591" />
```

... then the appropriate line in the **InstallService.cmd** would look like this:

```
SET _guid=77b0671d-ed31-4679-9bd3-fce7e245d591
```

3. The InstallService.cmd script creates a user with administrator rights on your Windows Azure machine, which will then register the services. Therefore, you need to choose a password for the new user and insert it into the script file.

```
SET _adminPassword=<YourPassword>
```

4. Open the Visual Studio's **Properties Window** (by selecting **View -> Properties Window** in the main menu or by pressing **F4**) and set the **Copy to Output Directory** property to *Copy always* for the following two files (both are located within the **CMSApp** project):

- App_Data/CMSModules/WinServices/services.xml
- Web.config

The solution is now ready to be deployed to the cloud. Once the application is deployed and starts for the first time, the InstallService.cmd script will register the services into the system. You will then be able to manage them via remote desktop.

2.13 Configuring Azure CDN

A Content Delivery Network (CDN) can significantly improve performance of your websites running in the Azure environment. By replicating the contents of your Blob storage across multiple data centers around the world, the CDN integrated into Windows Azure can speed up delivery of static content, such as images or stylesheets, as well as streaming media.

Kentico CMS supports the **Azure Blob storage CDN**, provided you perform a few configuration tasks, which are described in the following text.

Accessing the Management Portal

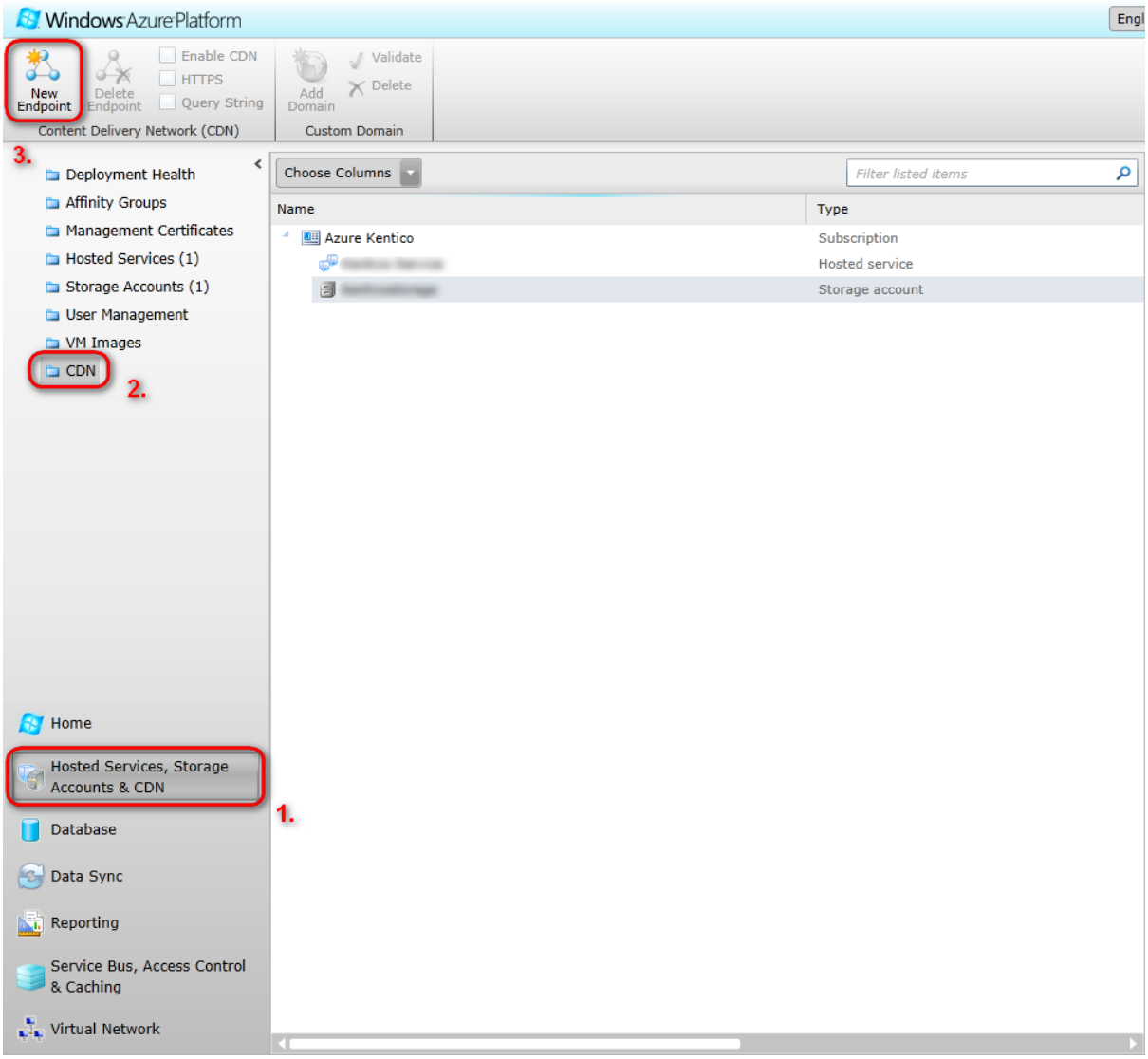
This chapter describes the old version of the Windows Azure Management Portal. To access the old portal from the new portal, click your account name in the top right corner of the new portal and click **Previous portal**.

Go to the old portal directly

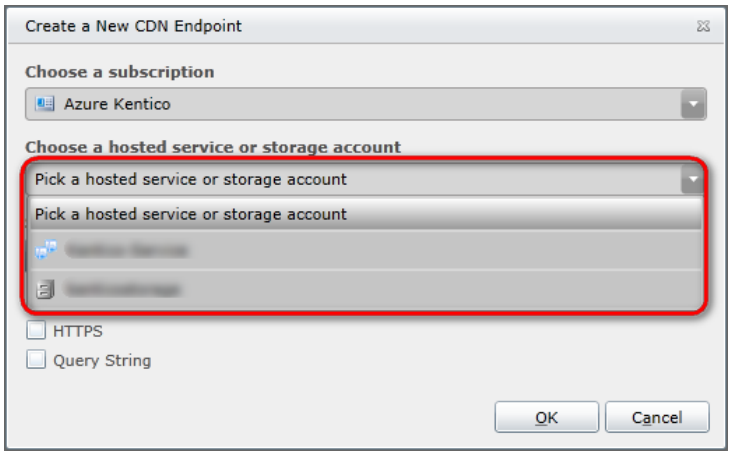
Enabling Azure Blob storage CDN

To enable CDN for your Windows Azure subscription, follow these steps:

1. Log in to the Windows Azure management portal.
2. Navigate to **Hosted Services, Storage Accounts & CDN -> CDN**.
3. Click **New endpoint**.

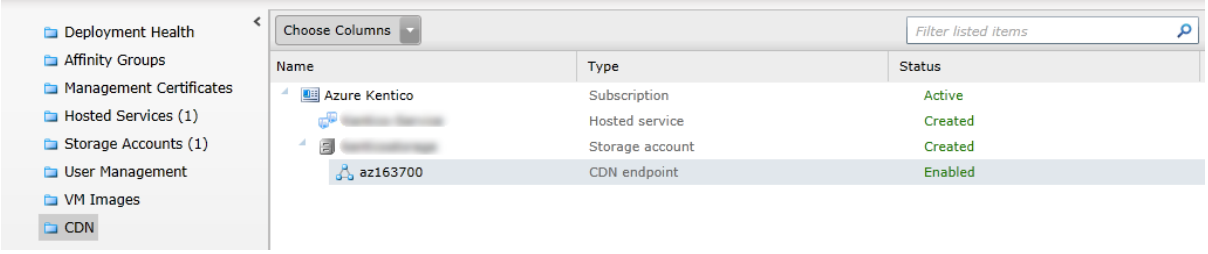


4. From the drop-down list in the window that pops up, select the source for the CDN. The source can be either a hosted service or a particular storage account.



5. Click OK to start creating the CDN. This action may take up to an hour.

When the CDN is created, it will appear in the list under the hosted service or storage account that you selected as a source.



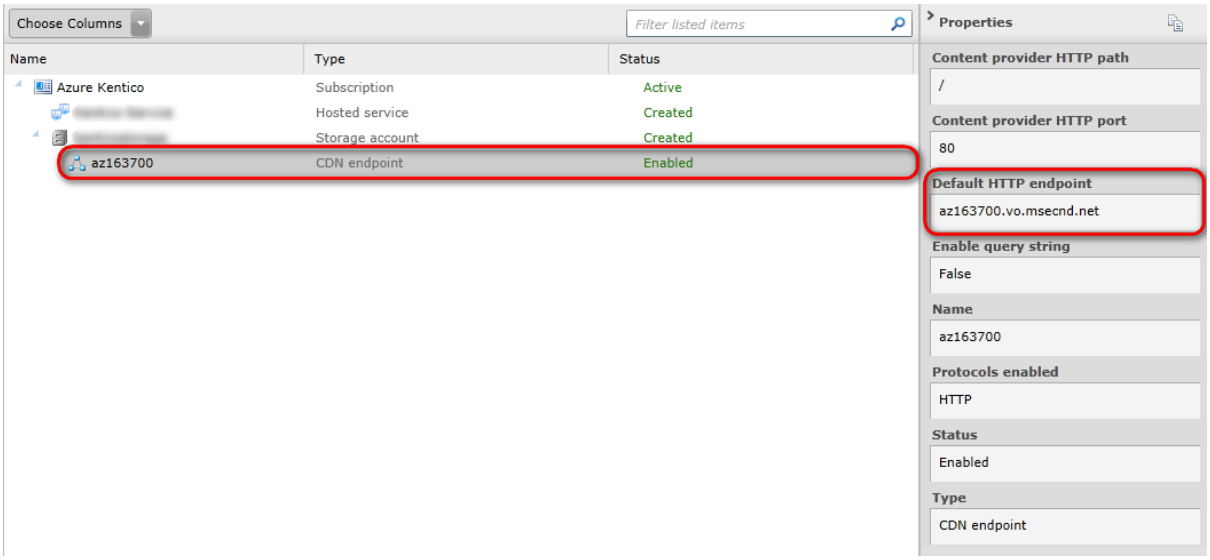
Configuring Kentico CMS to use CDN

To start fetching files from the CDN instead of the Blob storage, follow these steps:

1. Add the following pair of settings keys in the appSettings section of your web.config file:

```
<add key="CMSAzurePublicContainer" value="true" />
<add key="CMSAzureCDNEndpoint" value="Endpoint URL" />
```

2. Replace *Endpoint URL* with the URL of your CDN endpoint. To find out what the URL is, select the CDN endpoint in the Management portal. The URL is displayed in the **Properties** column, under **Default HTTP endpoint**.



2.14 Troubleshooting

This topic lists errors that are most commonly encountered when running Kentico CMS in the Azure environment, and describes solutions to them.

Poor performance on local machine

When running Kentico CMS in a local emulated environment, you may experience slow performance of the system, e.g. pages take long to load.

This issue is caused by insufficient system resources. We recommend deploying the application to the cloud or running the emulator on a machine which provides enough performance.

Role discovery data is unavailable

When running Kentico CMS in a local emulated environment, you may encounter the following exception:

System.InvalidOperationException: role discovery data is unavailable

To resolve the issue, you can try some of the following solutions:

- Verify that you have set the correct start-up project for the Kentico CMS solution. Open the solution in Visual Studio, right click the **CMSAzure** project in the Solution Explorer and choose **Set as StartUp project** from the pop-up menu.
- Verify that the Windows Azure Compute Emulator is running and that it is configured correctly. See the Application structure and configuration topic to learn how to configure the application to run in the emulated environment.
- The occurrence of the problem may be limited to machines which do not have enough resources to handle the load of the SDK. We recommend deploying the application to the cloud or running the emulator on a machine with sufficient performance.

Running without an emulator

It is not possible to run Kentico CMS in the Azure format without making changes in the application's code. If you still wish to run the Azure project as a standard web application, open the */Old_App_Code/CMSModules/WindowsAzure/AzureInit.cs* file and comment out the contents of the following methods:

- ApplicationStartInit
- BeginRequestInit

Remember to uncomment the code before deploying the application to the cloud.

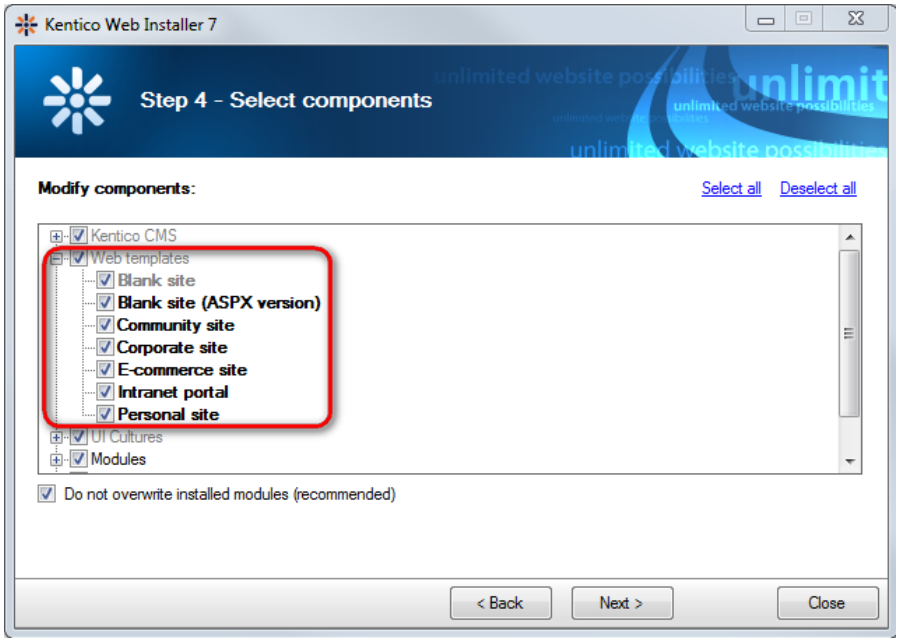
The specified path, file name, or both are too long

When packaging or publishing the application, you may get the following error while building the project in Visual Studio:

The specified path, file name, or both are too long. The fully qualified file name must be less than 260 characters, and the directory name must be less than 248 characters.

This error is caused by paths to some files, particularly web template files, being too long. You can solve the issue with one of the following procedures:

- Rename the project folder to a shorter name.
- Move the project higher in the folder tree (e.g. from *C:\inetpub\wwwroot* to *C:*).
- Delete sample website templates:
  1. Run the **Web installer**.
  2. In Step 2, choose to **modify an existing installation**.
  3. In Step 3, select the path to your project.
  4. In Step 4, expand the **Web templates** node of the component tree.
  5. **Deselect all templates** and click Next.



Dynamically generated web farm servers don't get deleted after stopping emulator

If you're running your application in multiple Azure instances, the system creates a web farm server for each of the instances. If you're testing the application on the emulator and stop the emulator, the servers don't get deleted automatically.

To solve this, restart your application from **Site Manager -> Administration -> System**.

Part

Storing files on Windows Azure

3 Storing files on Windows Azure

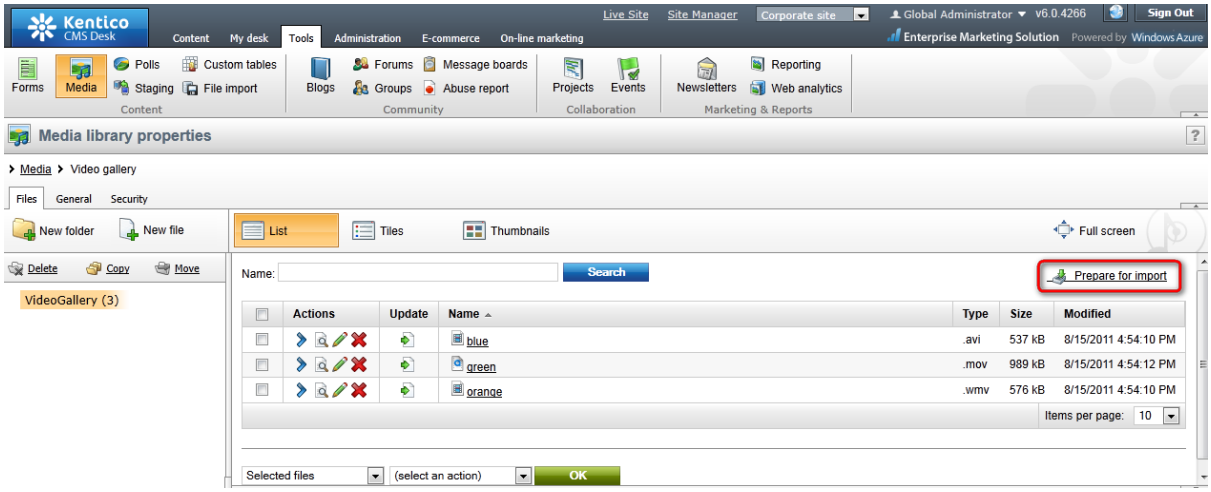
3.1 Azure Storage overview

When your application is deployed on Windows Azure, it is not possible to store files locally on individual instances (virtual machines in the cloud). The environment is dynamic and the application will be moved between different instances as necessary. For this reason, Kentico CMS stores its file system using the Blob service of a Windows Azure storage account, which allows it to be persistent and durable.

The entire file system of an application is stored within a single *container*. Individual files are then saved within the container as *blobs* (binary large objects). Containers cannot be nested within each other, but the folder hierarchy is simulated using blob prefixes. As a result, the application's files use the same structure as a standard Kentico CMS website.

Even though blob names are case sensitive, all file paths are converted to lower case before being processed, so path resolving behaviour should be just like that of a local file system. The only scenario that requires additional attention is when you need to import existing files into the CMS that were copied to the storage account manually. The names of such files will usually not be completely in lower case, which may prevent the system from reading them correctly.

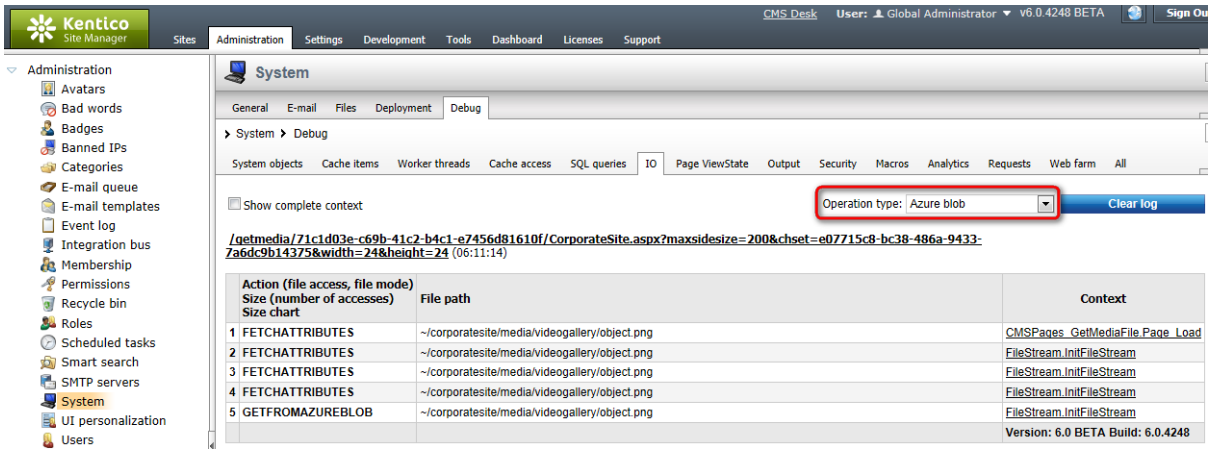
This situation will typically be encountered when importing files into a Media library, when using the File import or with other features that utilize a file system dialog. To solve the issue, you can select the given folder in the appropriate part of the interface and click the **Prepare for import button** above the file list.



The names of all files contained in the folder will be converted to lower case, and you can then import them into the system as needed.

If you encounter any file-related problems when using a Windows Azure storage account, you can monitor its input/output operations through the debugging features of Kentico CMS. To enable the logging of such operations, turn on the appropriate settings in the **Site Manager -> Settings -> System -> Debug -> IO** category (or add the **CMSDebugFiles** key to your web.config and set its value to *true*).

If you then go to **Site Manager -> Administration -> System -> Debug**, select the **IO** tab and choose the *Azure blob* option from the **Operation type** drop-down list, you can view recent input and output operations related to files stored on the given storage account.



Exploring your Windows Azure storage account

We recommend downloading and using CloudBerry Explorer for Azure Blob Storage, which allows you to connect to a storage account and browse the file content of its blob service.

This way, you can view and manage the files stored by your application just like you would with a file system hosted on-premise.

Storage configuration

You can specify several settings that affect your application's file storage:

The **CMSAzureRootContainer** setting specifies the name of the blob container that will serve as the root of the application's file system. The default value is *cmsroot*. You can set a different value in cases where you need to separate the files of multiple websites that use the same storage account.

You can use the **CMSAzurePublicContainer** setting to indicate whether the blob container used to store the application's file system should be public. If set to *true*, anyone will be allowed to access files directly through the URL of the appropriate blob service, for example:

http://<AccountName>.blob.core.windows.net/cmsroot/corporatesite/media/imagelibrary/logo.png

The two settings described above can be entered into the <*ConfigurationSettings*> section of the roles in the application's service configuration file (they must also be declared in the service definition):

```
<Setting name="CMSAzureRootContainer" value="Site2Root"/>
<Setting name="CMSAzurePublicContainer" value="true"/>
```

Additionally, if you open the website and go to **Site Manager -> Settings -> Cloud services**, you can set the value of the **Redirect files to Windows Azure storage** setting. If enabled, requests for files that are saved on a Windows Azure storage account will be redirected to the appropriate URL of the given account's blob service.

3.2 Hybrid storage scenarios

In some situations, it may be beneficial to store the files of an on-premise website on a Windows Azure storage account rather than on a local disk. For example, if your server has a limited storage capacity and you need to save large amounts of file data, using Windows Azure may be more convenient than upgrading your server, especially if the increased requirements are only temporary.

To set up this type of hybrid scenario with a Kentico CMS website, you need to configure the project to connect to the appropriate storage account:

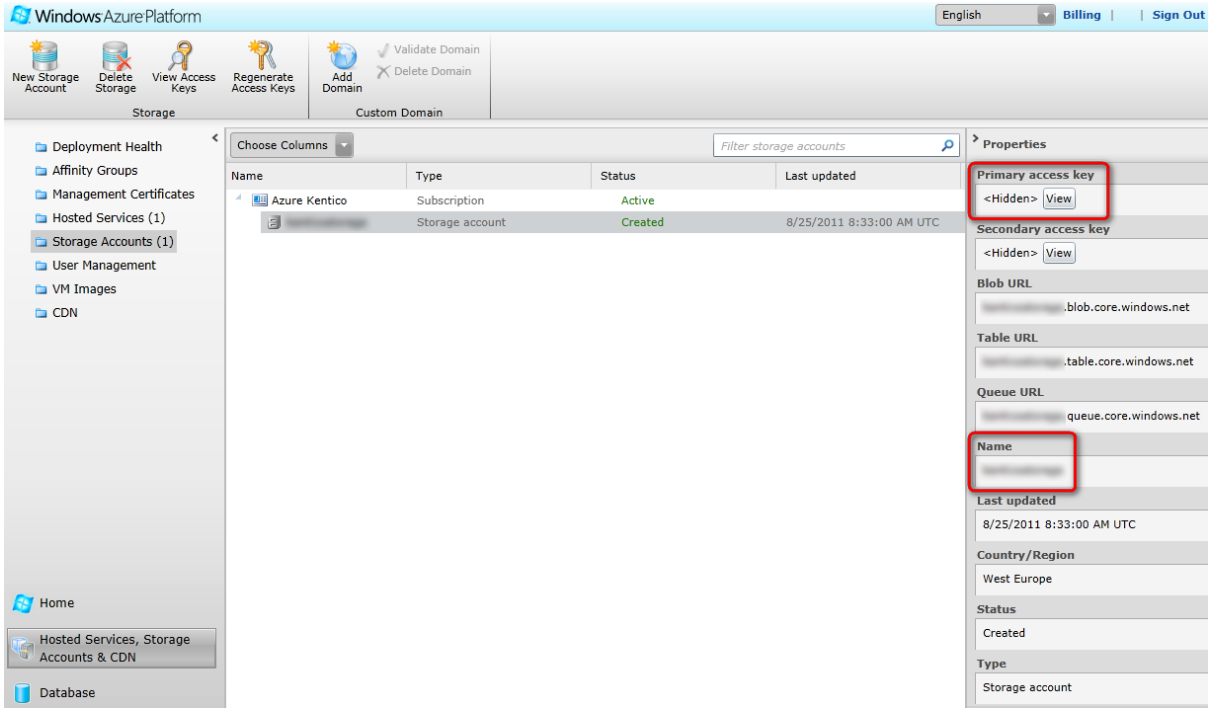
1. Edit your website's **web.config** file and add the following key into the <*appSettings*> section:

```
<add key="CMSExternalStorageName" value="Azure" />
```

This configures the application to use a Windows Azure Storage account for its file system. It is not necessary to set this key if the application itself is hosted on Windows Azure. Optionally, you may also set the two keys below:

Key	Description	Sample Value
CMSAzureTempPath	The folder specified by this key will be used to store temporary files on a local disk, e.g. when transferring large files to or from the storage account.	`<add key="CMSAzureTempPath" value="C:\AzureTemp" />`
CMSAzureCachePath	Specifies a folder on a local disk where files requested from the storage account will be cached. This helps minimize the amount of blob storage operations, which saves time and resources.	`<add key="CMSAzureCachePath" value="C:\AzureCache" />`

2. The next step is to specify the target storage account and ensure that your application can authenticate itself against it. You can create or view your storage account on the Windows Azure Management Portal (https://windows.azure.com/). Go to **Hosted Services, Storage Accounts & CDN -> Storage Accounts**, select the account that you wish to use. Its properties will be displayed on the right.



Copy your storage account's **Name** and **Primary access key**, and enter them as the values of the **CMSAzureAccountName** and **CMSAzureSharedKey** web.config keys in the <*appSettings*> section:

```
<add key="CMSAzureAccountName" value="AccountName"/>
<add key="CMSAzureSharedKey" value="PrimaryAccessKey"/>
```

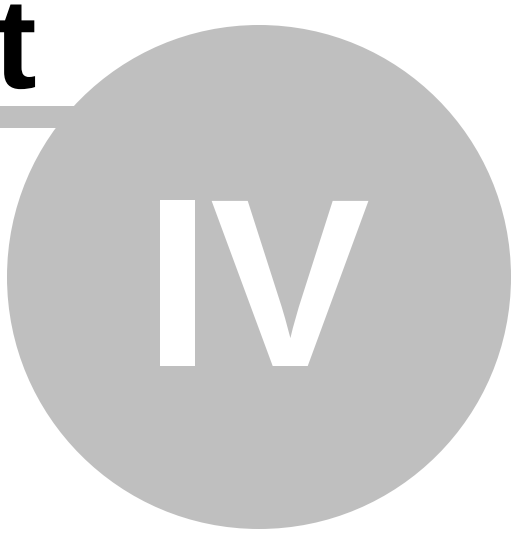
3. Save the changes to the web.config file and **Build** your project. All new files should now be stored in the cloud on the given storage account.

Additional website settings

When configuring this type of scenario, keep in mind that the website itself must be configured to store files in the file system rather than in the database only. Open the administration interface of your website, go to **Site Manager -> Settings -> System -> Files** and make sure that **Store files in file system** is checked.

It is also recommended to enable **Redirect files to disk** in the **System -> Performance** settings category. This means that files will be requested from the Azure Storage account rather than from the database (if possible).

Part



Application settings

4 Application settings

The settings in the table below may be used to set up the deployment of your Windows Azure application and configure its behaviour:

Key	Description	Sample Value
CMSAzureProject	Must be set to *true* if you wish to run the application on Windows Azure. *True* by default if the application is installed as a Windows Azure project, *false* in standard installations.	`<Setting name="CMSAzureProject" value="true"/>`
CMSAzureAccountName	Specifies the name of the storage account that the application will use for its file system. The account must belong to a valid Windows Azure subscription. If you wish to run the application on the local emulator, enter *devstoreaccount1* as the value.	`<Setting name="CMSAzureAccountName" value="devstoreaccount1"/>`
CMSAzureSharedKey	Must contain the primary access key of the storage account specified in the **CMSAzureAccountName** setting. You can find the appropriate value for your storage account on the Windows Azure Management Portal.	`<Setting name="CMSAzureSharedKey" value = "Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1SZFPTOtr/KBHBeksoGMGw==" />`
CMSAzureBlobEndPoint	Sets the endpoint used for the connection to the blob service of the specified storage account. If you wish to use the default endpoint, remove the setting completely from the appropriate files.	`<Setting name="CMSAzureBlobEndPoint" value="http://127.0.0.1:10000/devstoreaccount1"/>`
CMSAzureQueueEndPoint	Sets the endpoint used for the connection to the queue service of the specified storage account. If you wish to use the default endpoint,	`<Setting name="CMSAzureQueueEndPoint" value="http://127.0.0.1:10001/devstoreaccount1"/>`

	remove the setting completely from the appropriate files.	
CMSAzureTableEndPoint	Sets the endpoint used for the connection to the table service of the specified storage account. If you wish to use the default endpoint, remove the setting completely from the appropriate files.	`<Setting name="CMSAzureTableEndPoint" value="http://127.0.0.1:10002/devstoreaccount1"/>`
CMSAzureRootContainer	Specifies the name of the blob container that will serve as the root of the application's file system on the Windows Azure storage account. This can be useful in scenarios where multiple applications use the same storage account. The default value is *cmsroot*.	`<Setting name="CMSAzureRootContainer" value="CustomRoot"/>`
CMSAzurePublicContainer	Indicates if the blob container used to store the application's file system should be public. If set to *true*, it will be possible to access files directly through the URL of the appropriate blob service, for example: *http://<StorageAccountName>.blob.core.windows.net/cmsroot/corporatesite/media/imagelibrary/logo.png*	`<Setting name="CMSAzurePublicContainer" value="true"/>`
CMSConnectionString	This setting can be used to enter a database connection string for the application through the service configuration file.	`<Setting name="CMSConnectionString" value="Persist Security Info=False;database=DBName; server=tcp: serverName.database.windows.net; user id=login@serverName; password=password;Current Language=English;Encrypt=True; Connection Timeout=240;" />`

You can enter these settings directly into the <*ConfigurationSettings*> section of the roles in the application's service configuration file (**ServiceConfiguration.<name>.cscfg** under the **CMSAzure** project).

Alternatively, you can add these settings to the <*appSettings*> section of the **CMSApp** project's *web.config* file and the **SmartSearchWorker** project's *app.config*. In this case, the names are the same, but the standard key format must be used instead, for example:

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

Both options are valid, but placing your settings into the service configuration provides a higher degree of flexibility, since the service configuration file may be edited through the Azure Management Portal without the need to upload the application again as a new deployment.

Any of the web.config keys that you would normally use to set up the behaviour of your Kentico CMS website may also be defined as settings in the service configuration file. All you need to do is declare the given settings in the application's service definition (**ServiceDefinition.csdef**) as shown below:

```
<WebRole name="CMSApp">
...
  <ConfigurationSettings>

  ...

    <Setting name="CMSDefaultUICulture" />

  </ConfigurationSettings>
...
</WebRole>
```

Once the setting is declared, you can simply add a value for the setting in the service configuration file:

```
<Setting name="CMSDefaultUICulture" value="de-DE" />
```

When adding settings via the service configuration file, it is recommended to include them for both the **CMSApp** and the **SmartSearchWorker** roles. In the case of some settings, it may be sufficient to add them only for the CMSApp role, but only do this if you are sure that the given setting does not affect the smart search module at all.

Please note

You can edit your service definition and configuration files through a more user friendly interface in Visual Studio by right clicking the appropriate role under the **Roles** folder of the **CMSAzure** project and selecting **Properties**.

Unfortunately, any modifications made to the service definition require you to redeploy the application. To work around this, you can declare any settings that you may want to work with in the future and leave empty or default values in the service configuration until you need to change them.

Index

- A -

- B -

- C -

- D -

- F -

- I -

- L -

- P -

- S -

- U -

- W -