Windows Azure Support in Kentico CMS 5.5 R2

Kentico
unlimited website possibilities

www.kentico.com

Table of Contents

Introduction and prerequisites

This guide provides information about how Kentico CMS 5.5 R2 websites can be manually configured and deployed to the Windows Azure cloud platform. An automatic installer is unfortunately not available at this time, but is planned to be included in the next version along with other improvements to Windows Azure support.

To follow the instructions in this guide, basic knowledge of the Windows Azure Platform and access to a working Azure account is required. If you are new to Windows Azure and need to create an account or are looking for information, please visit http://www.microsoft.com/windowsazure/.

A further requirement is to have the following installed:

- Microsoft Visual Studio 2010 (or Microsoft Visual Studio 2010 Express Edition)
- Windows Azure SDK **version 1.3 or 1.4** and Windows Azure Tools for Microsoft Visual Studio (these may be downloaded at http://msdn.microsoft.com/en-us/windowsazure/cc974146)
- Microsoft SQL Server 2008 or newer (the Express version is sufficient)

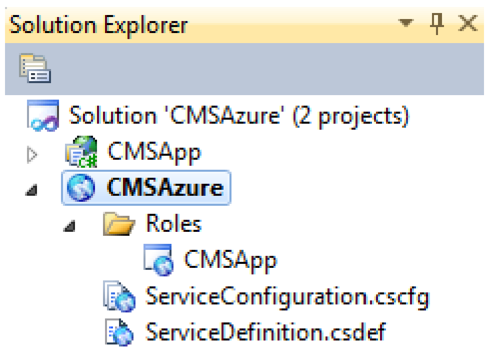
Deployment to Windows Azure (using a prepared package)

Along with the release of Kentico CMS 5.5 R2, we offer a way to try out Kentico CMS on the Windows Azure Platform. This can be achieved by using a prepared solution that is ready for deployment to Azure. The package containing it is available for download at http://www.kentico.com/downloads/Kentico_CMS_55_R2_Windows_Azure.zip and includes all Kentico CMS sample sites.

The following steps describe the deployment process:

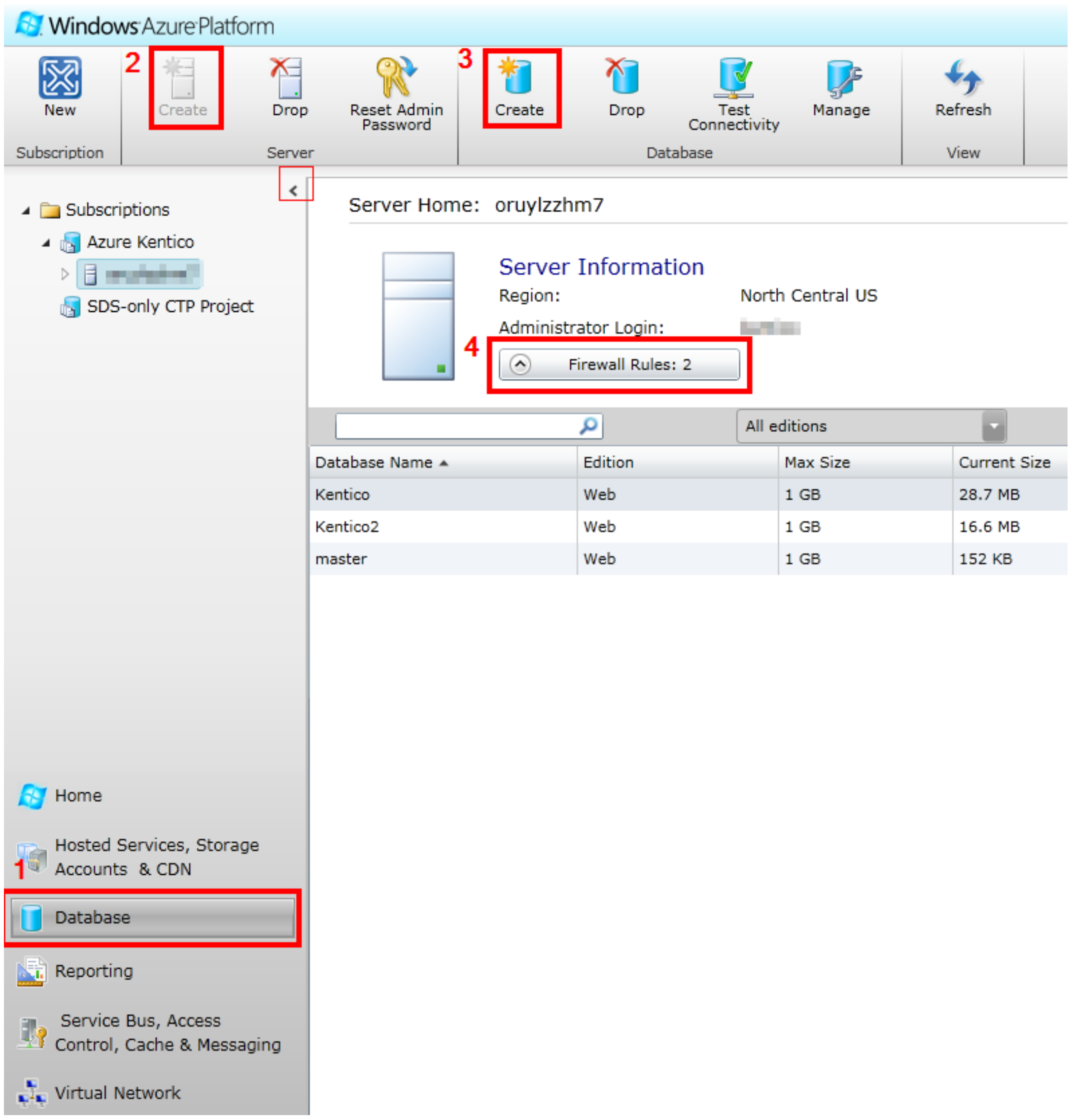
1. Download the .zip file from the link above and extract its content if you have not yet done so.

2. Unzip the **KenticoCMS55R2Azure.zip** file and open *KenticoCMS55R2Azure\CMSAzure\CMSAzure.sln* in Visual studio. The structure of the solution should be similar to the following:



The first project (**CMSApp**) is Kentico CMS converted to a web application and the second (**CMSAzure**) uses the Windows Azure Project template with the first project added as a Web role.

3. The next step is to create a database using the Windows Azure Management Portal. You can sign in using your Windows Live ID credentials at https://windows.azure.com/.



Server Home: oruylzzhm7

Server Information

Region: North Central US

Administrator Login: [redacted]

Firewall Rules: 2

Database Name ▲	Edition	Max Size	Current Size
Kentico	Web	1 GB	28.7 MB
Kentico2	Web	1 GB	16.6 MB
master	Web	1 GB	152 KB

When the management portal opens, click **Database** (1) and select your subscription. Next, **Create** a new database server (2), followed by a new database (3) on that server. Then configure the **Firewall Rules** (4) of the new database. Enable **Allow other Windows Azure services to access this service** and add the range of IP addresses that you use to connect to the internet in order to allow a connection to the database from your computer.

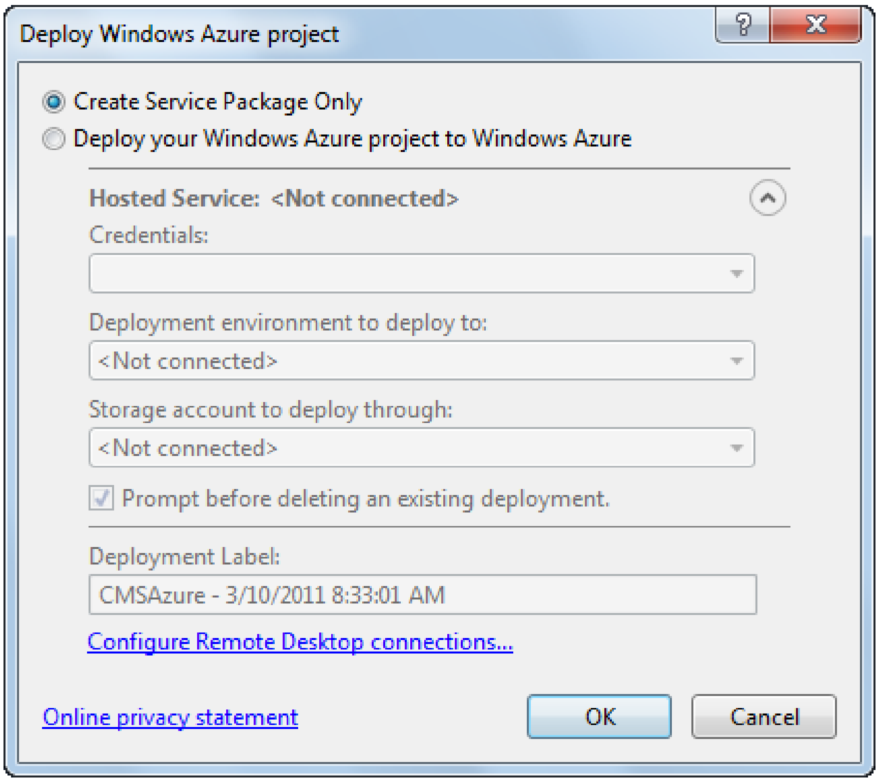
4. Next, it is necessary to import the Kentico database. There are two scripts included in the downloaded package for this purpose: **schema.sql** and **data.sql**. You can either use Microsoft SQL Management Studio 2008 R2 to connect to the cloud database server, open the script files and execute them, or run the scripts via the Database Manager for SQL Azure. The second option can be accessed from the Azure Management Portal by selecting your database and clicking the **Manage** button on the right of the ribbon at the top of the screen.

5. Once complete, the correct connection string for the database must be entered. Open the **web.config** file located in the main directory of the **CMSApp** project in Visual Studio and find the connection string (inside the **<connectionStrings>** section). It should be in the following format:

```
<add name="CMSConnectionString"
connectionString="Server=tcp:ServerName.database.windows.net;D
atabase=DBName;User ID=UserName@ServerName;
Password=Password;Trusted_Connection=False;Encrypt=True;"/>
```

Replace the *ServerName*, *DBName*, *UserName and Password* strings with the actual database server name, database name, user name and password for your Azure database and save the changes.

6. The next step is the deployment of the application. Right-click the **CMSAzure** project in the Solution Explorer and select **Publish**.


Deploy Windows Azure project

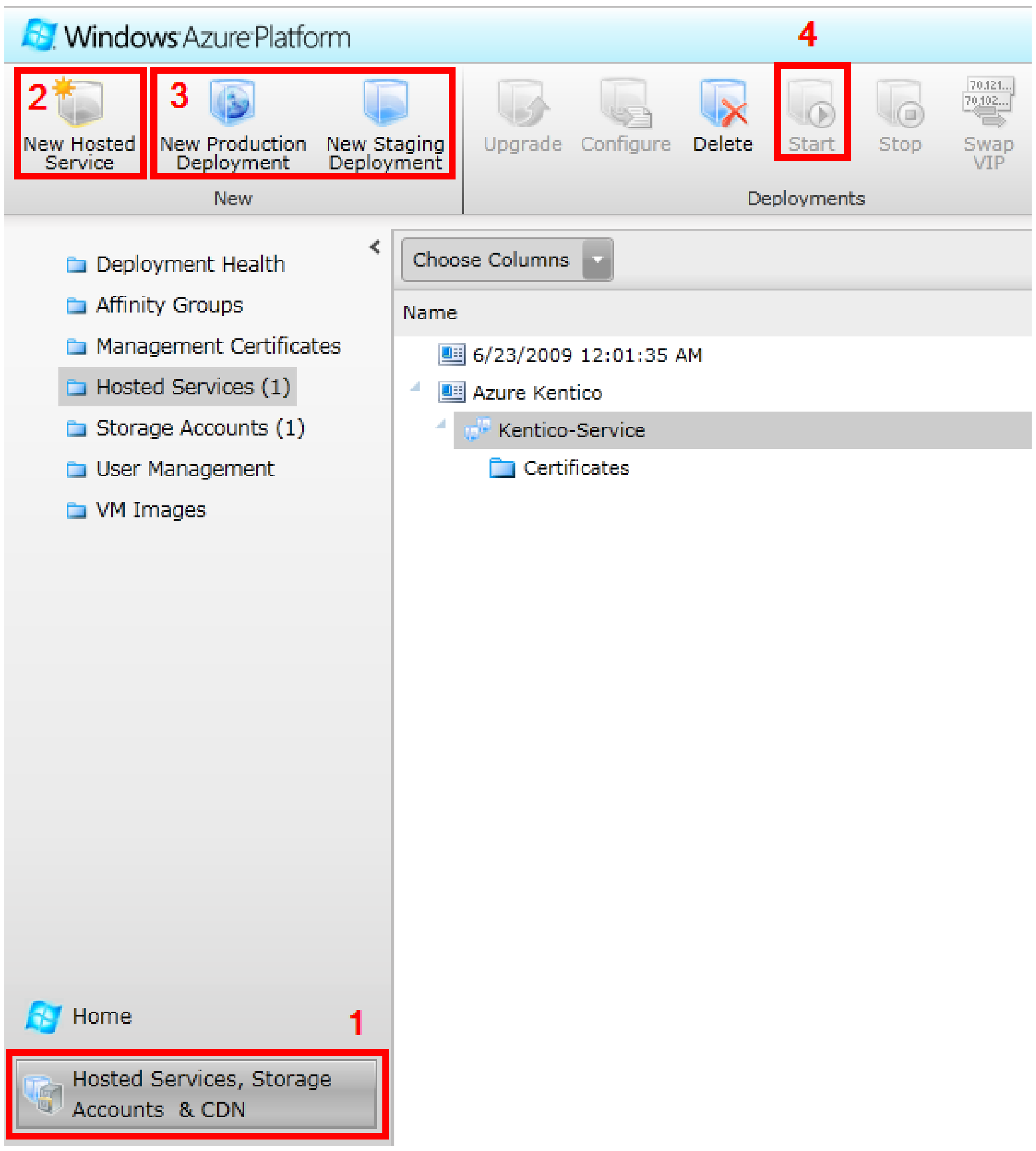
(●) Create Service Package Only
( ) Deploy your Windows Azure project to Windows Azure

Hosted Service: <Not connected>
Credentials:
Deployment environment to deploy to: <Not connected>
Storage account to deploy through: <Not connected>
☑ Prompt before deleting an existing deployment.
Deployment Label: CMSAzure - 3/10/2011 8:33:01 AM

Here you can select how the Windows Azure project will be published. If you already have your certificate installed in the cloud, you may select the **Deploy your Windows Azure project to Windows Azure** option, enter your credentials and deploy directly. Otherwise use **Create Service Package Only**.

Alternatively, you may run the application on the local emulator by pressing F5. Please be aware that running the emulator from Visual Studio causes an error when using Full IIS (configured by default). Please read the Running in a Full IIS hosting environment section of this guide to learn how to work around the issue.

7. If you selected **Create Service Package Only** in the previous step, two files will be created in the *~/CMSAzure/bin/Debug/Publish* directory: **CMSAzure.cspkg** and **ServiceConfiguration.cscfg**. When the files are published, open the Azure Management Portal again to upload them.



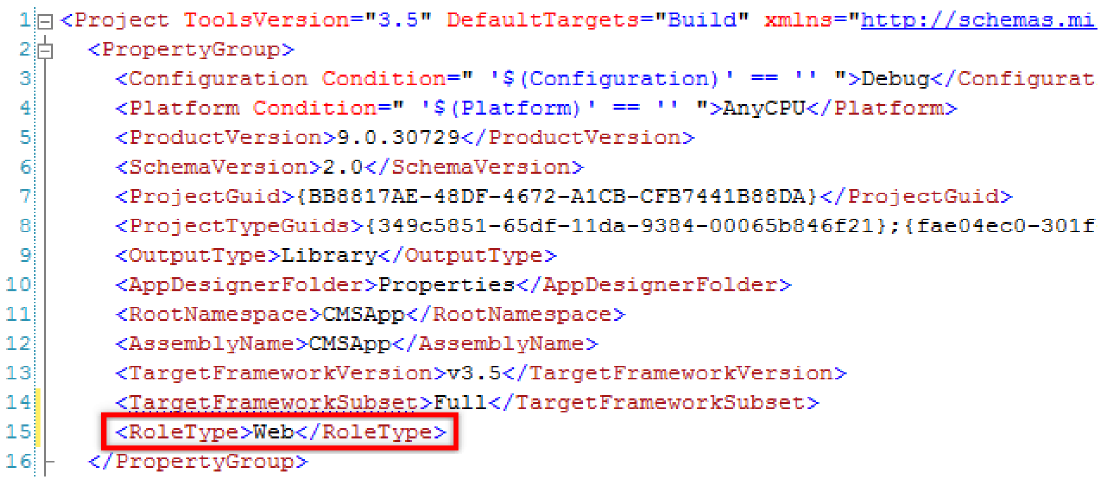
Here, click **Hosted services, Storage Accounts & CDN** (1), choose **Hosted Services** and select your subscription. If you do not yet have a Hosted Service, click **New Hosted Service** (2) to create a new one. Select your service from the tree on the right and click **New Production Deployment** or **New Staging Deployment** (3). The choice between these two determines which address the application will be available under. The production deployment uses a URL in format *<Selected name>.cloudapp.net*, while the staging deployment is available at *<Deployment ID>.cloudapp.net*. The *Deployment ID* of a staging deployment is generated automatically upon creation. Staging is meant for development and testing purposes and production 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 published service package (**.cspkg**) and configuration (**.cscfg**) files. When done, click **OK** to begin uploading the package to the cloud. Once the package is loaded, simply press **Start** (4) to run the application.

Conversion of an existing Kentico website to an Azure application

If you wish to deploy your existing Kentico CMS application to Azure, it is necessary to either develop it using version 5.5 R2 or perform the upgrade procedure to 5.5 R2 before you start. Further steps are described below:

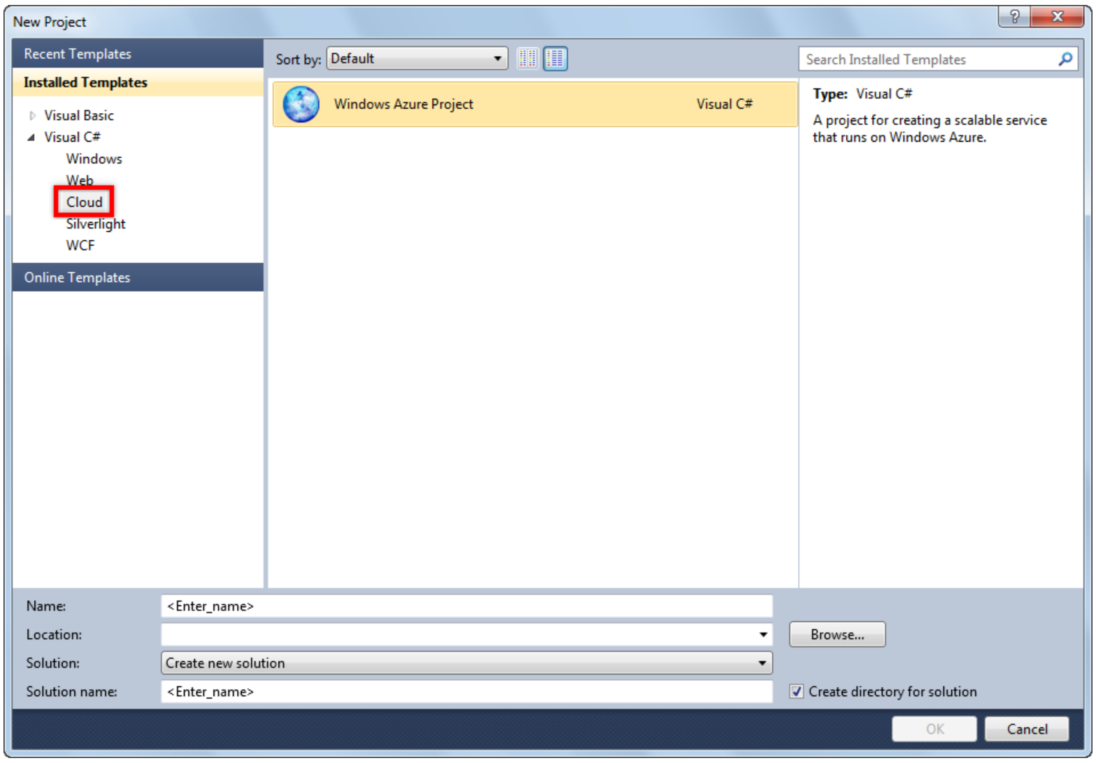
1. Convert your Kentico CMS website to a web application using the guide published on our DevNet portal at http://devnet.kentico.com/Blogs/Martin-Hejtmanek/November-2010/How-to-convert-Kentico-CMS-web-site-to-Web-Applica.aspx.

2. Open your Kentico CMS solution in Visual Studio, right-click the project file in the Solution Explorer and select **Unload Project**. Now right-click the project again and select **Edit**. Enter `<RoleType>Web</RoleType>` into the **<PropertyGroup>** element as shown in the following screenshot:



```
<Project ToolsVersion="3.5" DefaultTargets="Build" xmlns="http://schemas.mi
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configurat
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProductVersion>9.0.30729</ProductVersion>
    <SchemaVersion>2.0</SchemaVersion>
    <ProjectGuid>{BB8817AE-48DF-4672-A1CB-CFB7441B88DA}</ProjectGuid>
    <ProjectTypeGuids>{349c5851-65df-11da-9384-00065b846f21};{fae04ec0-301f
    <OutputType>Library</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>CMSApp</RootNamespace>
    <AssemblyName>CMSApp</AssemblyName>
    <TargetFrameworkVersion>v3.5</TargetFrameworkVersion>
    <TargetFrameworkSubset>Full</TargetFrameworkSubset>
    <RoleType>Web</RoleType>
  </PropertyGroup>
```

3. Next, create a **New Project** in Visual Studio. Select the **Windows Azure Project** template from the **Cloud** category.



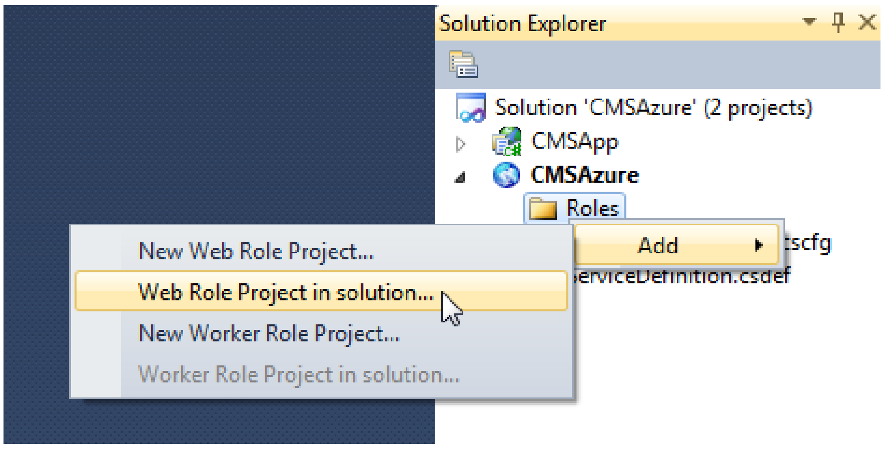
Type: Visual C#
A project for creating a scalable service that runs on Windows Azure.

Name: <Enter_name>
Location:
Solution: Create new solution
Solution name: <Enter_name>
✓ Create directory for solution

Click **OK** in the role selection dialog that appears next without adding any roles to the Windows Azure solution.

4. Now select **File -> Add -> Existing Project** and add the project file from your converted Kentico CMS web application.

5. The next step is to connect the Windows Azure project and the web application. Right-click the **Roles** folder under the Azure project, select **Add -> Web Role Project in solution** and choose your web application project.


Solution Explorer
Solution 'CMSAzure' (2 projects)
CMSApp
CMSAzure
Roles
Add
scfg
ServiceDefinition.csdef
New Web Role Project...
Web Role Project in solution...
New Worker Role Project...
Worker Role Project in solution...

6. Next, it is necessary to migrate your Kentico database into SQL Azure. We recommend using the SQL Azure Migration Wizard tool, which is available free of charge at http://sqlazuremw.codeplex.com/. Please note that the tool requires your database to run on Microsoft SQL Server 2008 or newer. Older versions are not supported. Kentico CMS also supports the Express version and it will be sufficient if you do not own a license for Microsoft SQL Server 2008 (or newer).

7. Before you continue, it is necessary to have a database created and ready on SQL Azure. If this is not the case, please see step 3 in the previous section of this guide (Deployment to Windows Azure - step 3) for information on how this can be achieved.

8. Install the SQL Azure Migration Wizard tool and run it. Select the **Analyze and Migrate -> SQL Database** option and choose the server where your Kentico database is located as the source. Select the specific database and then the **Script all database objects** option. The tool will now generate the required SQL script.

9. In the following step, enter the required connection information for your SQL Azure server and specify the target database. Click **Next** and the tool will execute the generated script. If successful, your database will be migrated to the cloud.

10. To complete the deployment of your application, follow steps 5, 6 and 7 from the previous section of this guide (Deployment to Windows Azure).

Kentico CMS 5.5 R2 limitations on Azure

The current version only supports a **single instance**.

Also, modules that require access to a local file system are not supported in the default version. This includes:

- Import/Export
- Smart search*
- Web analytics*
- Media libraries*
- File import
- WebDAV

* – the modules marked by an asterisk may be enabled by utilizing Windows Azure Drive as described in the Configuration for Windows Azure Drive section of this guide.

Additionally, the following specific functionality is not currently possible due to the implementation of the Windows Azure Platform:

- **Check out to file** – the option of saving large text objects such as page layouts, CSS stylesheets and transformations to a file for external editing is not available, because the local disk cannot be accessed.
- **BizForm data export to Excel** – this function, usually available on the *Data* tab of the BizForm editing interface, is not supported when running on Azure.

Recommended settings for Kentico CMS on Azure

Because of the limitations of Kentico CMS version 5.5 R2 when running on the Windows Azure Platform, we recommend using the following configuration options and settings:

1. Keys in the **<appSettings>** section of the web.config:

```
<add key="CMSLogEventsToFile" value="false"/>
```

This ensures that the Event log does not attempt to write to the file system. The default value for the **CMSLogEventsToFile** key is *false*, so no changes are necessary unless you already have this key set to *true*.

2. Application settings in the following categories of the **CMS Site Manager -> Settings** interface:

Web site:

- **Enable smart search indexing***: *false*; the Smart search module requires the use of a file system, so it is not supported by default.

Files:

- **Store files in file system**: *false*; the file system cannot be accessed.
- **Store files in database**: *true*; files must be stored in the SQL Azure database.
- **Generate thumbnails**: *false*; thumbnails cannot be generated, since files are not stored in the file system.

Web analytics:

- **Enable web analytics***: *false*; the Web analytics module requires the use of a file system, so it is not supported by default.

WebDAV:

- **Enable WebDAV support**: *false*; WebDAV editing is not supported on Azure.

* – the settings marked by an asterisk are not necessary if the respective modules are enabled by utilizing Windows Azure Drive as described in the following section of this guide.

Configuration for Windows Azure Drive

Windows Azure Drive may be used to provide a file system that allows the **Smart search**, **Web analytics** and **Media libraries** modules to function when running on Azure. This is achieved by operating a durable NTFS-formatted virtual drive under a Windows Azure Storage Account.

The following steps must be performed in order to configure the three mentioned modules to use Azure Drive. It is assumed that your website is already converted to an Azure application and that your database is migrated to SQL Azure as described in the Conversion of an existing Kentico website to an Azure application section of this guide.

1. Before you start, it is necessary to apply Kentico CMS hotfix **5.5R2.4** or newer if you have not already done so. You may download the appropriate hotfix package from http://devnet.kentico.com/Bugtracker/Hotfixes.aspx. This is not necessary if you are using the sample solution from the Kentico CMS 5.5 R2 Azure package, since it already contains the proper hotfix.

2. Next, download the Kentico CMS 5.5 R2 Azure package from http://www.kentico.com/downloads/Kentico_CMS_55_R2_Windows_Azure.zip if you have not yet done so, extract it and copy the **WebRole.cs** file to the directory containing your Kentico web project. Open your Kentico CMS solution in Visual Studio, right-click the project file of the CMS application, select **Add -> Existing item** and choose the **WebRole.cs** file. This class provides the functionality required to use Azure Drive.

If you are using the sample solution, this file is already present under the **CMSApp** project, but its functionality is disabled. In this case, edit the file and enable it by removing the comments surrounding to code.

3. The drive requires a working Windows Azure Storage Account. The credentials that allow access to the Storage Account you wish to use must be specified. This can be achieved by adding the following two keys into the **<ConfigurationSettings>** section of your Azure project's **ServiceConfiguration.cscfg** file:

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

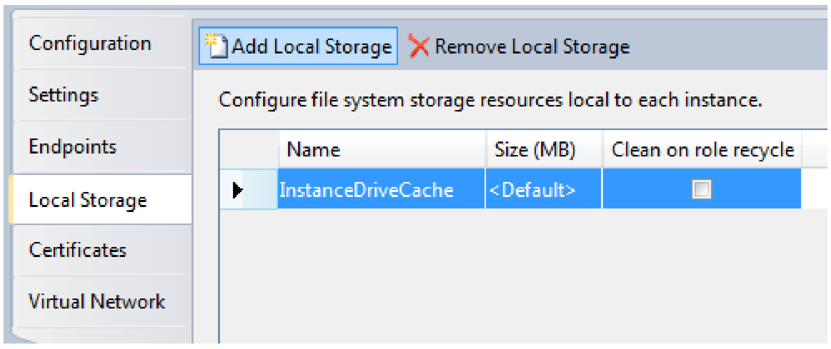
Replace the values of these keys with the actual name and primary access key of your Storage Account. To find this information, sign in to the Windows Azure Management Portal, navigate to **Hosted Services, Storage Accounts & CDN -> Storage Accounts**, select the given account and view its **Properties** displayed on the right.

The settings must also be registered in the **ServiceDefinition.csdef** file by adding a **<ConfigurationSettings>** section into the **<WebRole>** element as shown below:

```
<ConfigurationSettings>
      <Setting name="CMSAzureAccountName" />
      <Setting name="CMSAzureSharedKey" />
</ConfigurationSettings>
```

This section is already present in the sample solution.

4. Next, right-click the Web role representing your CMS application, located in the **Roles** folder under the Azure project, and select **Properties**. Switch to the **Local Storage** tab of the properties dialog, click **Add Local Storage** and name it *InstanceDriveCache*.



Configure file system storage resources local to each instance.

Name	Size (MB)	Clean on role recycle
InstanceDriveCache	<Default>	☐

5. Once this configuration is done, the application must be deployed to the cloud again as described in steps 5, 6 and 7 of the Deployment to Windows Azure section of this guide.

6. When the Azure deployment is complete, the three mentioned modules should be functional. However, the following tasks must be performed to ensure that existing files related to the modules are recreated on the drive:

- **Smart search** – all smart search indexes must be rebuilt (using the *Rebuild* action) via the administration interface at *CMS Site Manager -> Administration -> Smart search* before they become functional. Please note that using Azure Drive will not enable searching through file attachments, since smart search indexes are not used for this purpose and SQL Azure does not yet support full-text search.
- **Web analytics** – web analytics data is stored in the database regularly (every minute by default), so it should already be transferred if your database was migrated to SQL Azure. Recreating the *.log* files from the local file system should not be necessary.
- **Media libraries** – existing media files will not be transferred and must be added manually through the CMS interface of the deployment in order to be recreated on the drive.

Session state provider configuration for Azure

Information about storing session state using SQL Azure and performing the required configuration may be found in the following article:
http://blogs.msdn.com/b/sqlazure/archive/2010/08/04/10046103.aspx

Running in a Full IIS hosting environment

Windows Azure SDK 1.3 brings several new features, such as for example the possibility of connecting to a service instance directly using a Remote Desktop client. Notable among these improvements is the option of running Windows Azure web roles, and as a result Kentico CMS applications, in a Full IIS hosting environment (as opposed to using Hosted web core). More information about Full IIS capabilities can be found in the following blog: http://blogs.msdn.com/b/windowsazure/archive/2010/12/02/new-full-iis-capabilities-differences-from-hosted-web-core.aspx

Unfortunately, due to an error in the SDK, the Kentico CMS application cannot be run on the local Windows Azure compute emulator under Full IIS directly from Visual Studio. If you attempt to do this, you will get the following exception:

Windows Azure SDK 1.3

"The communication object, System.ServiceModel.Channels.ServiceChannel, cannot be used for communication because it is in the Faulted state."

Windows Azure SDK 1.4

“This request operation sent to net.pipe://localhost/iisconfigurator did not receive a reply within the configured timeout (00:01:00). The time allotted to this operation may have been a portion of a longer timeout. This may be because the service is still processing the operation or because the service was unable to send a reply message. “

As described in the blog, a part of the application runs under the **WAIISHost.exe** process, which communicates with the **IISConfigurator** process through a named WCF pipe. This pipe has a 60 second timeout, which cannot be changed through any configuration. At the start of the emulation, the permissions of all files in the project are modified to ensure that IIS can access them. However, Kentico CMS contains too many files and the 60 second timeout is reached before all permissions can be set. This is the cause of the mentioned exception.

This problem does **not** occur when running Kentico CMS under Full IIS in the actual cloud.

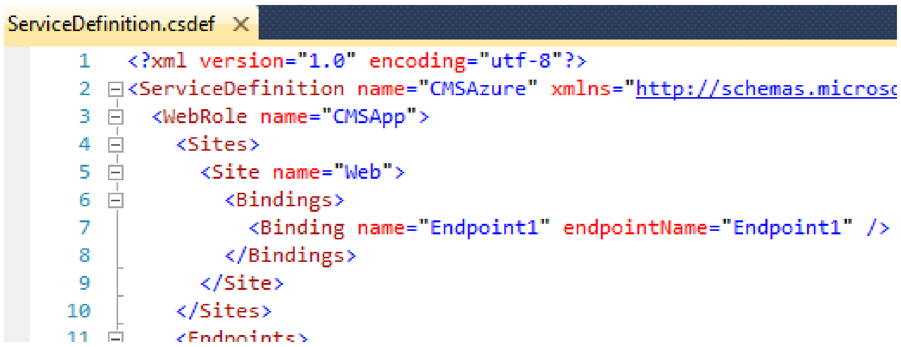
If you do not wish to run your Kentico CMS application under Full ISS and prefer to use Hosted Web Core, simply comment out the **<Sites>** section in the **ServiceDefinition.csdef** file of your Azure project.

There is also an alternative way of running Kentico CMS on the local emulator under full IIS. To do this, you must utilize the **CSPack** and **CSRun** command line tools as described below.

1. Run the **Windows Azure SDK Command Prompt** as administrator from the Start menu. Now prepare your application for deployment to the Windows Azure compute emulator using the CSPack tool. Enter a command according to the following syntax:

```
cspack <ServiceDefinitionPath> /role:<WebRoleName>;<KenticoCMSProjectPath>
/sitePhysicalDirectories:<WebRoleName>;<VirtualSitePath>;<KenticoCMSProjectPath>
/out:<OutputDirectory> /copyOnly
```

- **<ServiceDefinitionPath>** - enter the absolute physical path to the definition file of your Azure project, for example: *C:\...\KenticoCMS55R2Azure\CMSAzure\ServiceDefinition.csdef*
- **<WebRoleName>** - enter the name of the web role under which your Kentico CMS application is running, for example: *CMSApp*
- **<KenticoCMSProjectPath>** - enter the physical path of the directory containing your Kentico CMS application, for example: *C:\...\KenticoCMS55R2Azure\CMSApp*
- **<VirtualSitePath>** - enter the path of your site as defined in the *<Sites>* section of your Azure project's service definition file. If you are using the default definition with no virtual directories, simply enter: *Web*



```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="CMSAzure" xmlns="http://schemas.microsc
  <WebRole name="CMSApp">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="Endpoint1" endpointName="Endpoint1" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
```

- **<OutputDirectory>** - enter the physical path of a directory where the processed files of your application will be stored, for example: *C:\Deployment\KenticoCMS*

2. When the output directory is created, deploy the site to the emulator using the CSRun tool. Enter a command according to the following syntax:

```
csrun <ApplicationDirectory> <ServiceConfigurationPath> /launchbrowser
```

- **<ApplicationDirectory>** - enter the physical path of the output directory created by the CSPack tool (*C:\Deployment\KenticoCMS* in this example).
- **<ServiceConfigurationPath>** - enter the absolute physical path to the configuration file of your Azure project, for example: *C:\...\KenticoCMS55R2Azure\CMSAzure\ServiceConfiguration.cscfg*

3. The emulator should now start and your application will be opened in a browser.