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

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

However, we recommend that you use a Microsoft Azure Storage account as a file system for your project.

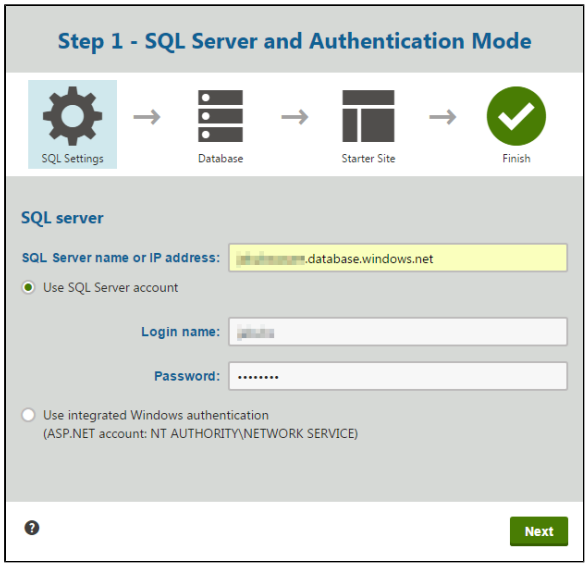
Developing Azure projects locally in the emulator

1. Prepare an SQL database or prepare a database on Microsoft Azure:
    a. Create a new Azure SQL Server and a database.
    b. Add firewall rules for the server.
2. Create an Azure Storage account through the Microsoft Azure Management Portal.
3. Open your Azure project in Visual Studio.
4. Right-click the **CMSAzure\Roles\CMSApp** role and select **Properties**.
5. Switch to the **Settings** tab.
6. Select **Local** as the Service Configuration.
7. Set the value of **CMSConnectionString** key to the connection string for your database.
    - If you are connecting to an Azure SQL Database, you can find the connection string on the Azure Management Portal in **SQL Databases ->** select your database **-> Overview** tab **-> Show database connection strings** link.

      Use 'username@servername' for the User ID connection string parameter.

8. Based on what storage you want to use, choose one of the following options - either the Azure storage or the Storage emulator:
    - Azure storage - set the values of **CMSAzureAccountName** and **CMSAzureSharedKey** keys for your Azure Storage.
        - You can find these values on the Azure Management Portal in **Your storage account -> Access keys** tab. Use the **Storage account name** and one of the generated access key values (Azure provides two access keys so that you can maintain connections using one key while regenerating the other).
    - Storage emulator - set ***devstoreaccount1*** as the **CMSAzureAccountName** and ***Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1SZFPTOtr/KBHBeksoGMGw==*** as the **CMSAzureSharedKey**.
        - See Use the Azure Storage Emulator for Development and Testing for more information.
9. Repeat the same steps for the **SmartSearchWorker** role.
10. Right-click the **CMSAzure** project in the Solution Explorer and select **Set as StartUp Project**.
11. Run the project in the debug mode.
12. Go through the Kentico database installation when the system automatically opens the database installation in a browser.

- If developing on the Azure SQL Server, enter the SQL server name in the following format: *servername.database.windows.net*.



Your web project is now prepared for local development.

> **CMSHashStringSalt key**
>
> We recommend that you make sure that the **CMSHashStringSalt** key is set to a single static value in your web.config file. See The CMSHashStringSalt key is identical in all environments.

Developing Azure projects locally as web applications

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

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

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

   - Replace **YourServerName**, **YourDatabaseName**, **YourUsername** and **YourPassword** with your own values.
   - You can find the connection string of Azure SQL Database on the Azure Management Portal in **SQL Databases ->** select the database **-> Overview** tab **-> Show database connection strings** link.
6. Add the **CMSAzureAccountName** and **CMSAzureSharedKey** keys to the <appSettings> section of the web.config file.

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

- Replace **YourStorageAccountName** and **YourPrimaryAccessKey** with your own values.
- You can find these values on the Azure Management Portal in **Your storage account -> Access keys** tab. Use the **Storage account name** and one of the generated access key values (Azure provides two access keys so that you can maintain connections using one key while regenerating the other).

7. Right-click the **CMSApp** project in the Solution Explorer and select **Set as StartUp Project**.
8. Run the project in **debug** mode.
9. Go through the Kentico database installation when the system automatically opens the database installation in a browser.
    - If developing on Azure SQL Server, enter the SQL server name in the following format: *servername.database.windows.net*.

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

CMSHashStringSalt key

We recommend that you make sure that the **CMSHashStringSalt** key is set to a single static value in your web.config file. See The CMSHashStringSalt key is identical in all environments.

Storing media files, metafiles and attachments when developing Azure projects as web applications

If you store media files, metafiles and attachments in the file system (the **Settings -> System -> Files -> Store files in the file system** setting is selected) when you develop your Azure project as a web application, these files are stored downright in your project folders as media files. This may cause problems after you deploy your project to the Microsoft Azure platform, as these files are regarded as read-only and deleting such files from the administration interface may not be allowed.

Therefore, we recommend that you configure your project to either:

- Store files only in the database - clear the **Store files in the file system** setting and select the **Store files in the database** setting in **Settings -> System -> Files**.
  or
- Store such files in the Azure blob storage - select the **Store files in the file system** setting and configure the **CMSAzureAccountName** and **CMSAzureSharedKey** keys in your web.config file as specified in this section.

You can find more information about the file storing settings in Storing files.

Error associated with this problem:

- "Failed to publish document: File.Delete: File <filename> cannot be deleted because it exists in application file system."

Deploying locally developed Azure projects

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

Deployment with migration to a new database

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

1. Create a Cloud Service.
2. Create a new Azure SQL Database Server and a database and configure the server's firewall to accept your IP address.
3. Deploy your SQL server, for example according to instructions in the Migrate SQL Server database to SQL Database article.
4. Open your Azure project in Visual Studio using the **CMSAzure.sln** file.

5. In **Properties** of the **CMSApp** web role, on the **Settings** tab, in the **Cloud** service configuration, fill the **CMSAzureAccount Name** and **CMSAzureSharedKey** keys with appropriate values for your Azure Storage (the same as for the Local configuration).
6. Set the **CMSConnectionString** value to the connection string of the new Azure SQL database.
7. Configure the SmartSearchWorker role.
8. Deploy your project to the created Cloud Service.
    - Make sure that the **Enable Remote Desktop for all roles** and **Enable Web Deploy for all web roles (requires Remote Desktop)** options are selected.

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

Tip: You can also use other approaches to deploy an on-premise database to Azure SQL Database. See the SQL Server database migration to SQL Database in the cloud article for more information.

Alternative database deployment

If the database deployment process does not suit you, you can export and import your Kentico website instead. This way you can choose particular object types which you want to deploy.

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

Deployment when using the same database

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

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

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

Web Deploy

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

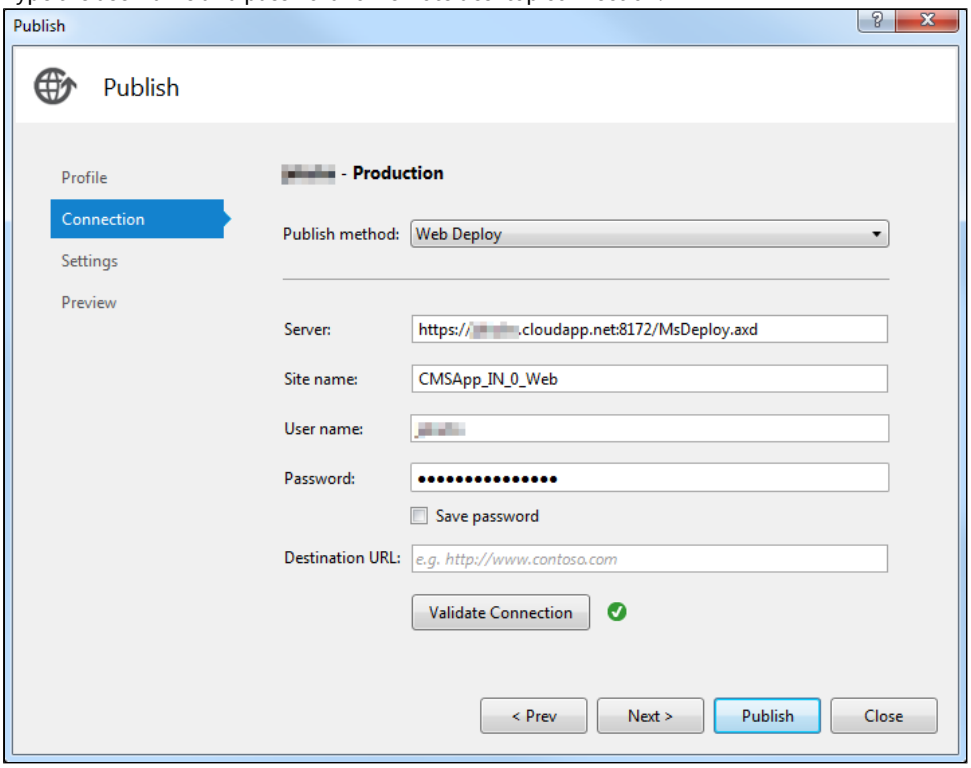
1. Right-click the **CMSApp** project and select **Publish**.
2. Select the existing publish profile from the drop-down list.
3. Switch to the **Connection** tab.
4. Provide the password for remote desktop connection.
5. Click **Publish**.

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

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

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

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



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

Final deploy

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

Setting up post-production development of Azure projects

If you realize that you need to make changes to your Azure project already in the production, you can utilize the content staging functionality to synchronize the database changes between your local development and production projects. However, you will still need to deploy your project files.

Moreover, you may want to set up a new blob storage to separate the content files of your local and production projects. The synchronization of these files will be handled by the content staging functionality.

Although there are other ways to develop your projects after you promote them to the production, we recommend the following configuration:

1. Create a new Azure Storage.
2. Set the local configuration of your project to connect to a local database and the newly created blob storage:
    a. Open your Azure project in Visual Studio using the **CMSAzure.sln** file.
    b. Double-click the **CMSAzure\Roles\CMSApp** role.
    c. Switch to the **Settings** tab and select **Local** as the Service Configuration.
    d. Make sure that the value of **CMSConnectionString** key is set to the connection string for your local database (you can use your existing development database or copy your production database to a local server).
    e. Set the values of **CMSAzureAccountName** and **CMSAzureSharedKey** keys for the newly created Azure Storage.
        - You can find these values on the Azure Management Portal in **Your storage account -> Access keys** tab. Use the **Storage account name** and one of the generated access key values (Azure provides two access keys so that you can maintain connections using one key while regenerating the other).
    f. Set these settings also for the **SmartSearchWorker** role.
3. Configure the content staging between your projects. Configure your local project as the source server and production project as the target server.

> **Staging large files (larger than 2 GB)**
>
> By default, the system will not stage files that are larger than 2 GB. If you need to stage larger files, change the maximum content and request size values in your projects.

> **CMSHashStringSalt key**
>
> The **CMSHashStringSalt** key defines the salt value that the system uses in hash functions, for example in macro signatures and media library links. For the content to be synchronized properly through Content staging, the value of this key must be identical in all environments you use. Otherwise, the hash values of objects will be different in different environments.
>
> Therefore, make sure that the value of this web.config key is set to a single static value, for example a random GUID. If the key is not present in the web.config file, it means that its default value is the connection string of your project. In this case, we recommend that you add this key to your local project with the connection string of your production project as a value.
>
> **Note:** We do not recommend changing the **CMSHashStringSalt** key value for the production project, as it will break macro signatures and media library links.

Your local project is now configured to be developed separately from the production project.

Deploying updated production projects

After you have configured the content staging functionality for your projects, you can propagate object changes to the production project through the content staging functionality (see Synchronizing the content). If you develop custom code, you also need to redeploy your project to the Staging environment and swap staging with production.

1. Make sure that the cloud configuration in your project is set to connect to the production database and production Azure storage.
    a. Open your Azure project in Visual Studio using the **CMSAzure.sln** file.
    b. Double-click the **CMSAzure\Roles\CMSApp** role.
    c. Switch to the **Settings** tab and select **Cloud** as the Service Configuration.
    d. Make sure that the value of **CMSConnectionString** key is set to the connection string for your production database.
    e. Set the values of **CMSAzureAccountName** and **CMSAzureSharedKey** keys for the production Azure Storage.
    f. Set these settings also for the **SmartSearchWorker** role.
2. Deploy your project with the **Enable Remote Desktop for all roles** and **Enable Web Deploy for all web roles (requires Remote Desktop)** options CLEARED.
3. Disable Smart search functionality in the staging deployment (Smart search cannot run simultaneously on two projects connected to the same database):
    a. Open the Kentico administration interface and the **Settings** application.
    b. Select the **System -> Search** category.
    c. Clear the **Enable smart search indexing** setting.
    d. Click **Save**.
4. Test that the staging website works as expected.

    > Do not make any changes to the website at this moment. The staging website is connected to the production database and blob storage, which could cause inconsistencies if objects are modified through the staging project.

5. Swap production deployment with staging deployment:
    a. Open the Azure Management Portal.
    b. Select your Cloud service and switch to the **Overview** tab.
    c. Click **Swap** in the top panel. Azure instantly swaps the production and staging deployments.
6. Stop or delete the Staging deployment (so that you can enable Smart search again).
7. Enable Smart search functionality in the production deployment.
8. Open the administration interface of your **local** project and synchronize the content changes.

You have redeployed your locally developed changes to the production.