For projects hosted in **Microsoft Azure Cloud Services**, you need to **perform the entire upgrade on a local machine** and then redeploy the project after the upgrade is successfully completed.

Completing the upgrade procedure may take a significant amount of time, particularly for heavily customized projects. We recommend that you keep your production site running and first perform a local upgrade of your project to upgrade your code files and try out the database upgrade in advance. Then you need to take your website offline and repeat the process for a copy of the production database.

Choose the upgrade procedure according to your Azure environment:

- Upgrading Azure Cloud Services projects through the local emulator
- Upgrading Azure Web Apps projects

> ⚠ We do NOT recommend upgrading your database on an Azure SQL server directly, as this may prevent the upgrade script from performing certain changes in the database.

See also: Running Kentico on Microsoft Azure

Upgrading Azure Cloud Services projects through the local emulator

We recommend that you perform the first site request and testing after the upgrade of an Azure project through the local emulator.

Part 1 - Project upgrade with a test database upgrade

First upgrade your project code files in a local environment and try out the upgrade of your database.

Database export

1. Create a copy of your Azure database on the same Azure SQL server. Creating a copy of a running database ensures that the copied database stays consistent, unlike database export performed in the next step, which may result in a faulty export.
    a. Open the Azure management portal and select your **SQL database**.
    b. Click **Copy** on the **Overview** tab.
    c. Copy the database to the same Azure SQL server.
2. Export the copied Azure database and import it to your on premise server:
    a. Open **Microsoft SQL Server Management Studio** and connect to your Azure SQL server (it is recommended to always use the latest version of Management Studio to remain compatible with Azure SQL Database updates).
        - You can find the name of your database server in the Azure management portal on the **Overview** tab. Enter the username and password that you used to create the Azure SQL server.
        - To connect to the Azure SQL server, you need to configure the firewall rules for the server in the Azure management portal – select your SQL database, click **Set server firewall** on the **Overview** tab, and add firewall rules for your local machine.
    b. Right-click the copied database and select **Tasks -> Export Data-tier Application**.
    c. Save the database export to the disk.
    d. Connect to your own SQL server.
    e. Right-click the **Databases** node and select **Import Data-tier Application**.
    f. Import the exported database to your server.

Project upgrade

1. Set up your project in a local development environment (emulated) and set the connection strings for the **CMSApp** and **SmartSearchWorker** roles to connect to the imported database on your server.
    - You need to set the connection string for all configurations (both Local and Cloud). The upgrade procedure upgrades the database specified in the connection string of the Cloud configuration.
    - See Developing Kentico Azure projects locally for details.
    - You do not need to start the emulator right now.

2. Increase the *Connection Timeout* value in the connection strings of the CMSApp and SmartSearchWorker roles. We recommend at least 600 seconds.
3. Perform the required Steps before the upgrade.
4. Run the upgrade utility for your local project (see Applying the upgrade).
5. The upgrade attempts to automatically update the project's Microsoft Azure configuration files (*ServiceConfiguration.cscfg*, *ServiceDefinition.csdef*). This process may fail if your configuration files contain certain types of customizations. In this case, new files are created with the *.new* extension. You need to manually transfer your configuration to the new files and then replace the original ones.
6. The upgrade creates a new version of the *SmartSearchWorker* project's *app.config* file with the *.new* extension. You need to manually transfer your configuration to the new file and then replace the original one.
7. If your CMSAzure project is customized, the upgrade creates a new version of *CMSAzure.ccproj* in the *CMSAzure* folder (with the *.new* extension). Transfer any customizations from your original CMSAzure.ccproj file and remove the .new extension from the new one.
8. Open the project in Visual Studio (using CMSAzure.sln) and start the site in the emulator. Wait for the upgrade procedure to finish.

   > **Important**: You always need to perform the first request on the upgraded site in a local development environment. Processing of the first request may take a longer than usual (up to an hour).

9. Perform the required Steps after the upgrade and test your site.
   - If you need to make any database changes, track these changes so that you can replicate them again later on the production database.
10. Remove the connection string from the Cloud configurations of both the CMSApp and SmartSearchWorker roles and deploy the project to your Microsoft Azure staging environment. See Deploying Azure projects to Cloud Services for more information.

    - It is essential NOT to include the connection string to ensure that the application does not start after the deployment, which could cause problems with the web farm server settings.

Part 2 - Production database upgrade

When you have upgraded your code files and uploaded your project to the staging environment, you can now shut down your production website and upgrade the production database.

Database export

1. Take your website offline. We recommend uploading the App_offline.htm file to your site using remote desktop:
   a. Open the Azure management portal and select your Cloud service.
   b. Select the **Remote Desktop** tab and configure access for All roles.
   c. Switch to the **Roles and Instances** tab.
   d. Select an instance and click **Connect**. The system downloads a file for remote connection.
   e. Run the file and connect to the cloud server.
   f. Locate your application and upload the **App_offline.htm** file (available in the upgrade installation folder) to its root. You can simply use copy & paste.
   g. Repeat steps *e - g* for all instances.
2. Export your production database from the Azure SQL server and import it to your on premise server.

Database upgrade

1. Run the SQL upgrade script on the imported database on your server:
   - If you have a standard Kentico database, run the ***upgrade.sql*** script located in the **SQL** folder of your upgrade installation directory.
   - If your project uses a separated on-line marketing database, expand the **SQL\Separation** folder in your upgrade installation directory and run the scripts *in the following order:*
     a. ***separated.sql*** (against the separated on-line marketing database)
     b. ***default.sql*** (against the main Kentico database)
2. Set the connection strings for the **CMSApp** and **SmartSearchWorker** roles to connect to the newly imported database on your server.

3. Open the Azure project in Visual Studio (using CMSAzure.sln) and start the site in the emulator. Wait for the upgrade procedure to finish.
4. Perform any database changes that you tracked previously during the test database upgrade.
5. Export your upgraded database from your on premise SQL server and import it to the Azure SQL server into a new database:
    a. Export the upgraded database to a BACPAC file.
    b. Import the BACPAC file to your Azure SQL Database.

    > We recommend that you do not delete the original database immediately and wait after you have tested the upgraded database.

6. Set the connection string of your previously deployed project in the staging environment to connect to the new upgraded database. Set the connection string for both the CMSApp and SmartSearchWorker roles.
    a. Open the Azure management portal and select your Cloud service.
    b. Select the **Configuration** tab.
    c. Switch to the **Staging** slot.
    d. Set the **CMSConnectionString** to connect to the upgraded database.
7. Swap the staging deployment with your production deployment.
8. Open your website in the administration interface and open the **Event log** application and check if any errors occurred.

Your project files and database are now upgraded. When you have adequately tested your website and you are sure it is running as expected, you can delete your original Azure SQL database, the copied database and the staging deployment. You can also add the connection string to the Cloud configuration file of your local project to have it prepared for future deployments.

> **Upgrading Azure projects through a web application**
>
> If you are developing Azure projects locally as web applications, you can also upgrade your projects this way, without setting up the local emulator. The difference is that you set the connection string in the web.config file instead of the configuration files.

Upgrading Azure Web Apps projects

If you want to upgrade an Azure Web Apps project, the procedure is similar to upgrading a standard web site or web application project with the difference that you must export the database from the Azure SQL server and import it to your on premise server.

Part 1 - Project upgrade with a test database upgrade

Database export

1. Create a copy of your Azure database on the same Azure SQL server. Creating a copy of a running database ensures that the copied database stays consistent, unlike database export performed in the next step, which may result in a faulty export.
    a. Open the Azure management portal and select your **SQL database**.
    b. Click **Copy** on the **Overview** tab.
    c. Copy the database to the same Azure SQL server.
2. Export the copied Azure database and import it to your on premise server:
    a. Open **Microsoft SQL Server Management Studio** and connect to your Azure SQL server (it is recommended to always use the latest version of Management Studio to remain compatible with Azure SQL Database updates).
        - You can find the name of your database server in the Azure management portal on the **Overview** tab. Enter the username and password that you used to create the Azure SQL server.
        - To connect to the Azure SQL server, you need to configure the firewall rules for the server in the Azure management portal – select your SQL database, click **Set server firewall** on the **Overview** tab, and add firewall rules for your local machine.
    b. Right-click the copied database and select **Tasks -> Export Data-tier Application**.
    c. Save the database export to the disk.
    d. Connect to your own SQL server.
    e. Right-click the **Databases** node and select **Import Data-tier Application**.

f. Import the exported database to your server.

Project upgrade

1. Set the connection string in the web.config file of your project to the imported database on your server.
2. Perform the required Steps before the upgrade.
3. Run the upgrade utility for your local project (see Applying the upgrade).
4. Be sure to **open the upgraded website** in a browser and wait for the upgrade procedure to finish.
   - When handling the first request, the system performs certain tasks required to finalize the upgrade. Processing of the first request may take longer than usual.
5. Perform the required Steps after the upgrade and test your site.
   - If you need to make any database changes, track these changes so that you can replicate them again later on the production database.
6. Remove the connection string from the web.config file and deploy the project to your Azure Web App. You can use a staging environment.

   - It is essential not to include the connection string to ensure that the application does not start after the deployment, which could cause problems with the web farm server settings.

Part 2 - Production database upgrade

When you have upgraded your code files and uploaded your project to a staging environment, you can now shut down your production website and upgrade the production database.

Database export

1. Take your website offline. You can upload the App_offline.htm file to your site through FTP.
2. Export your production database from the Azure SQL server and import it to your on premise server.

Database upgrade

1. Run the SQL upgrade script on the newly imported database on your server:
   - If you have a standard Kentico database, run the ***upgrade.sql*** script located in the **SQL** folder of your upgrade installation directory.
   - If your project uses a separated on-line marketing database, expand the **SQL\Separation** folder in your upgrade installation directory and run the scripts *in the following order:*
     a. ***separated.sql*** (against the separated on-line marketing database)
     b. ***default.sql*** (against the main Kentico database)
2. Set the connection string in the web.config file of your project to the new database on your server.
3. Open the project in a browser and wait for the upgrade procedure to finish.
4. Perform any database changes that you tracked previously during the test database upgrade.
5. Export your upgraded database from your on premise SQL server and import it to the Azure SQL server into a new database:
   a. Export the upgraded database to a BACPAC file.
   b. Import the BACPAC file to your Azure SQL Database.

   > We recommend that you do not delete the original database immediately and wait after you have tested the upgraded database.

6. Set the connection string of your previously deployed project (in the staging environment) to connect to the newly upgraded database.
7. Swap the staging deployment with your production deployment.
8. Open your website in the administration interface and open the **Event log** application and check if any errors occurred.

Your project files and database are now upgraded. When you have adequately tested your website and you are sure it is running as expected, you can delete your original Azure SQL database, the copied database and the staging deployment. You can also add the connection string in the web.config file of your local project to have it prepared for future deployments.