# Creating a .NET Core MVC Project to Azure With Azure SQL

## Create a .NET Core MVC Project

* File, New Project, Web, ASP.NET Core Web Application (or search in VS2019)
* Select MVC
* Deselect HTTPS so we have to buy/create a certificate for use in Azure.

## Add your Models and configure Setup.cs

(same as for a local setup… use a local SQL Connection String for initial local testing)

## Add your Controllers and Views

(same as for a local setup…)

## Build and test your application locally

(don’t publish to Azure until everything is working locally!)

## Create Your Azure SQL Database

1. Log into Azure
2. In the left navigation click SQL Databases.
3. Click **“+ Add**”.
4. Select or create your preferred Resource Group (probably the same one you are using for your web application)
5. Enter a Database Name.
6. If you have not previously created a Server, click **Create**.
   1. Enter a server name (must be unique in Azure!)
   2. Enter a name for an admin user (Could use “admin” or “Susan”, etc. as this is not an AD account. It is a “SQL Account”.)
   3. Enter a password.
   4. Select a data center. This is almost always the same as where you will be hosting your web application.
   5. Checkmark “Allow Azure services to access server” so you can browse and edit your SQL data right in the Azure Portal.
   6. Click **OK**.
7. Leave **Elastic Pool** as “No”.
8. Click **Configure Database** and click “**Looking for basic, standard, premium?**”.
9. Select **Basic** and click **Apply**. (only $5 a month!)
10. Click **Review and Create**.
11. Click **Create**.
12. Wait until the database is ready.
13. Click **Go to resource**. (Or click **SQL databases** on the left and click the new database.)

Here you can:

* Monitor or delete the database.
* Configure firewall rules to permit testing from your local PC.
* Use the browser based Query Editor to create tables and data, and to run queries.
* Configure backups, database replication, and dozens of other options.

Notes:

* Your “Server” is a virtual SQL Server and can host multiple databases.
* You create different Azure SQL Servers to group databases by billing needs or performance levels.

## Configure the Azure SQL Server firewall to allow local testing and administration

1. Navigate to your database in Azure.
2. In the top toolbar click **Set server firewall**.
3. In the top toolbar click **Add client IP** and click **Save** and **OK**. (that’s all you need for testing!)
4. Optionally, add ranges of addresses for your corporate internet ID or your business partners.
5. Close the Firewall settings blade.

Note:

* You can confirm that Azure found the right IP address for your PC by opening a CMD window and typing IPCONFIG and pressing Enter.
* You do not need to modify firewall settings when an Azure app accesses Azure SQL.

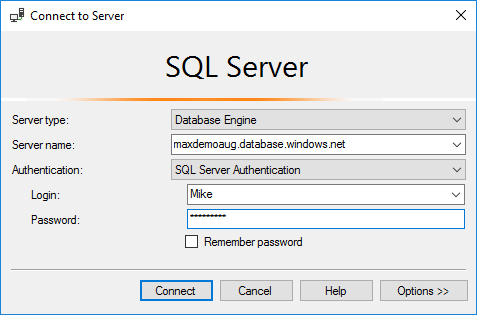
## Find the Azure SQL Connection String for your database.

1. In the Overview blade for your database click the “Connection Strings” link. (blue text: “Show database connection strings”)
2. Make sure the ADO.NET tab is selected. (or pick one of the others if you are writing a Java app!)
3. Copy the connection string or click the Copy button on the far right.
4. Paste the connection string into Notepad or other error and update the user name and password you created when you created the database.

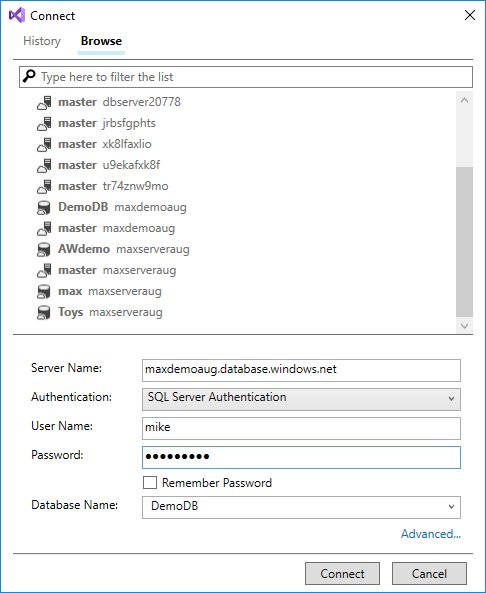
Server=tcp:***yourDatabaseServerName***.database.windows.net,1433;Initial Catalog=DemoDB;Persist Security Info=False;**User ID={your\_username};Password={your\_password};**MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;

You can now use the SQL Server Management Studio or the Visual Studio tools on your local PC to maintain the new database. You can also create a connection string to run your local application with the Azure SQL database.

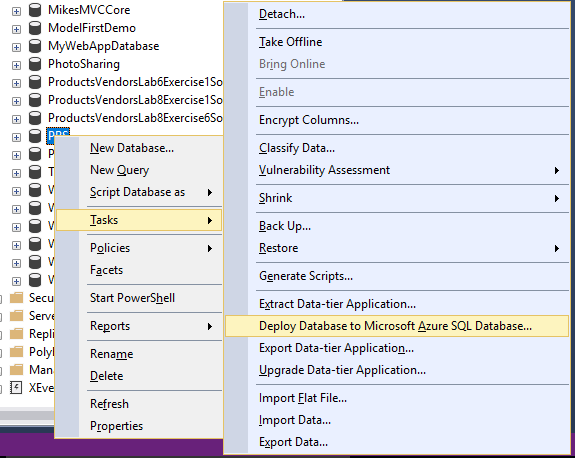
## Access your database from SSMS (optional)

1. Launch SSMS.
2. When asked for a server name, enter the URL-like name from azure.
3. Select SQL Server Authentication.
4. Enter the user name and password you created when you created the database.  
    
5. Click Connect.
6. In the Object Explorer expand your server, database and tables.

## Access your database from Visual Studio (optional)

1. Display the SQL Server Object Explorer.
2. Click the Add SQL Server icon.
3. Expand the Azure branch.
4. Click your new database. (Not the “master” database of your new server.)
5. Enter the user name and password you created when you created the database.  
    
6. Click Connect.
7. In the SQL Server Object Explorer expand your server, database and tables.

## Test Your Local Application with the Azure SQL Database

1. In your Startup.cs or your appsettings.json file, replace the local connection string with the one you copied and updated from Azure.
2. Then:
   1. If this is a Code First EF project, run Update-Database.
   2. If you are moving an existing database from an on-prem server or development PC, use one of the available data migration tools.
      1. SQL Server database migration to Azure SQL Database: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-single-database-migrate
      2. Using SSMS: Right-click the database, click Tasks, click Deploy Database to Microsoft Azure SQL Database and follow the steps. 😉  
         
      3. Azure Data Factory - https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/move-sql-azure-adf
      4. Large databases can be shipped on disk drives to Microsoft and processed for a fee.
      5. And many other tools… search for “Migrate SQL Server data to Azure”

## Publish the Project to Azure

1. Change your database connection string to the Azure connection string.
2. Right-click the project and click Publish (or click the Build menu and then Publish).
3. Click App Service (or App Service Linux) and select Create New.
4. Click Publish.
5. Login and/or select the correct account for Azure from the top right of the dialog box.
6. Review the name, and change it to better one (and one that no one has ever used before!).
7. Select the Resource Group you want to use (or create a new one for this project).
8. Click New next to Hosting Plan and review the options. In Size select Free!
9. Leave Application Insights as None.
10. Click Create.
11. Wait until the publish has completed and a browser has opened. (The status is in the bottom left corner of the dialog box.)

Notes:

* You can first create the Web App in Azure and then publish your Visual Studio project to it.
* The Free hosting plan is good for testing and hobby sites, but may be useful for a production site.