

Laboratorium Storage

Wojciech Tobiś

Wymagania

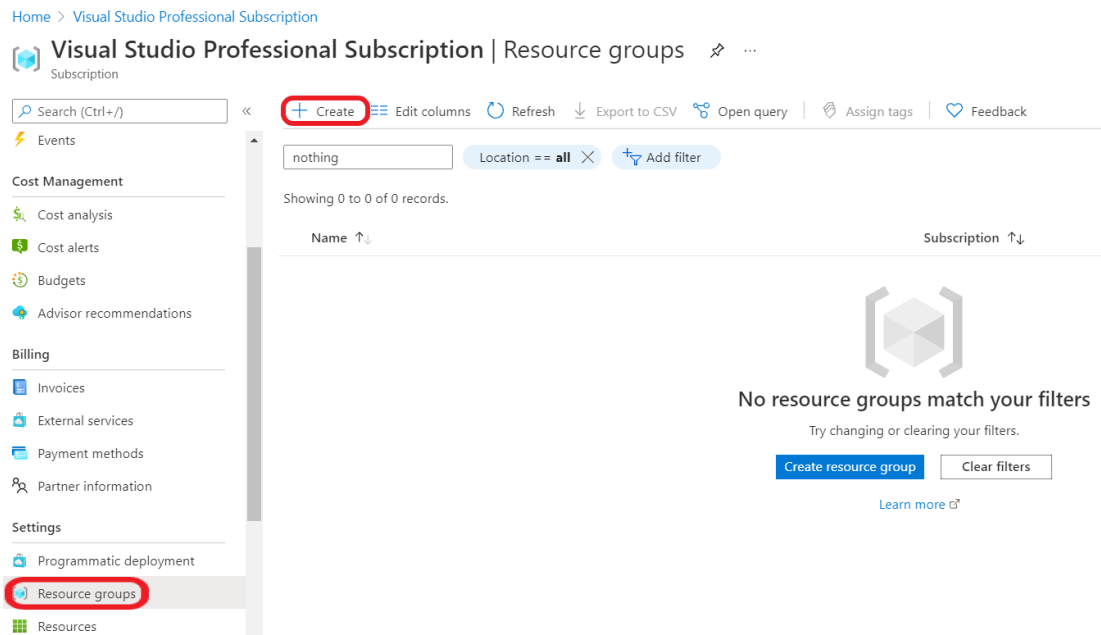
- Konto Azure
- Visual Studio lub Visual Studio Code
- Konto na GitHubie
- Dotnet 5 SDK

Scenariusz

1. Utworzenie nowej bazy danych

Celem tego ćwiczenia jest stworzenie bazodanowego serwera oraz bazy danych w Azure.

- Zaczynamy od utworzenia *Resource Group*. Możemy to zrobić z poziomu subskrypcji, wybierając **Resource groups -> Create**.



- Wprowadzamy nazwę grupy a następnie wybieramy region (najlepiej najbliższy naszej lokalizacji). Tworzymy *Resource Group*.
- Otwieramy utworzoną grupę i tworzymy nowy zasób klikając **Create**.
- Wybieramy z listy *SQL Database* i klikamy **Create**.
- W sekcji *Project details* wybieramy odpowiednią subskrypcję i utworzoną grupę zasobów.
- W sekcji *Database details* wpisujemy nazwę naszej bazy danych, a następnie tworzymy nowy serwer poprzez wybranie **Create new**.

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name * ✓

Server * ⓘ ✓

[Create new](#)

- Po otwarciu formularza do tworzenia serwera bazy danych wprowadzamy jego unikalną nazwę i lokalizację (najlepiej tę samą, którą wybraliśmy przy tworzeniu grupy).
- Jako metodę uwierzytelniania wybieramy **Use both SQL and Azure AD Authentication** a następnie ustawiamy swoje konto jako *Azure AD admina* oraz wprowadzamy login i hasło dla *Server admina*. Klikamy **Ok**.

[Home](#) > [Visual Studio Professional Subscription](#) > [AGH_LAB_STORAGE](#) > [Create a resource](#) > [SQL Database](#) > [Create SQL Database](#) >

Create SQL Database Server

Microsoft

Server details

Enter required settings for this server, including providing a name and location. This server will be created in the same subscription and resource group as your database.

Server name * ✓
 .database.windows.net

Location * ✓

Authentication

Select your preferred authentication methods for accessing this server. Create a server admin login and password to access your server with SQL authentication, select only Azure AD authentication [Learn more](#) using an existing Azure AD user, group, or application as Azure AD admin [Learn more](#), or select both SQL and Azure AD authentication.

Authentication method

☐ Use SQL Authentication

☐ Use only Azure Active Directory (Azure AD) authentication

☒ Use both SQL and Azure AD Authentication

Set Azure AD admin

Wojciech Tobisz
Admin Object/App ID: 0249c939-13db-422b-b388-5b3c161b4d32
[Set admin](#)

Server admin login * ✓

Password * ✓

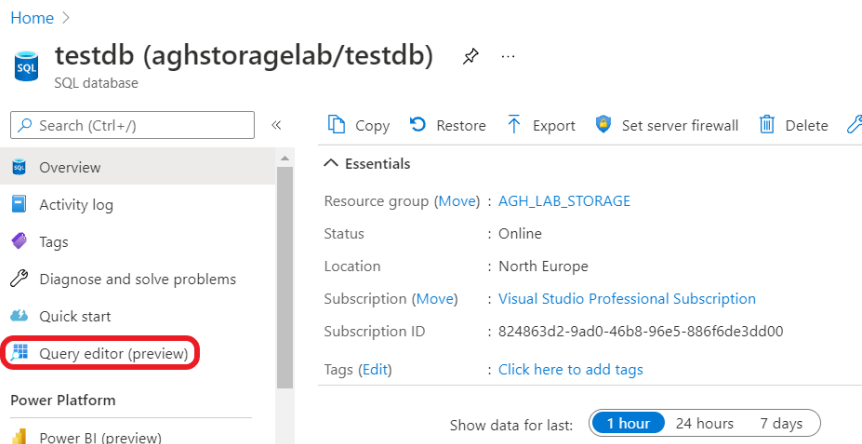
Confirm password * ✓

- Ustawiamy **Want to use SQL elastic pool?** na **No**.
- Ustawiamy **Workload environment** na **Development**.
- W **Compute + storage** klikamy **Configure database** i wybieramy najtańszy plan, jaki uda się znaleźć.
- Zmieniamy **Backup storage redundancy** na **Locally-redundant backup storage**.
- Klikamy **Review + create** i później **Create**.

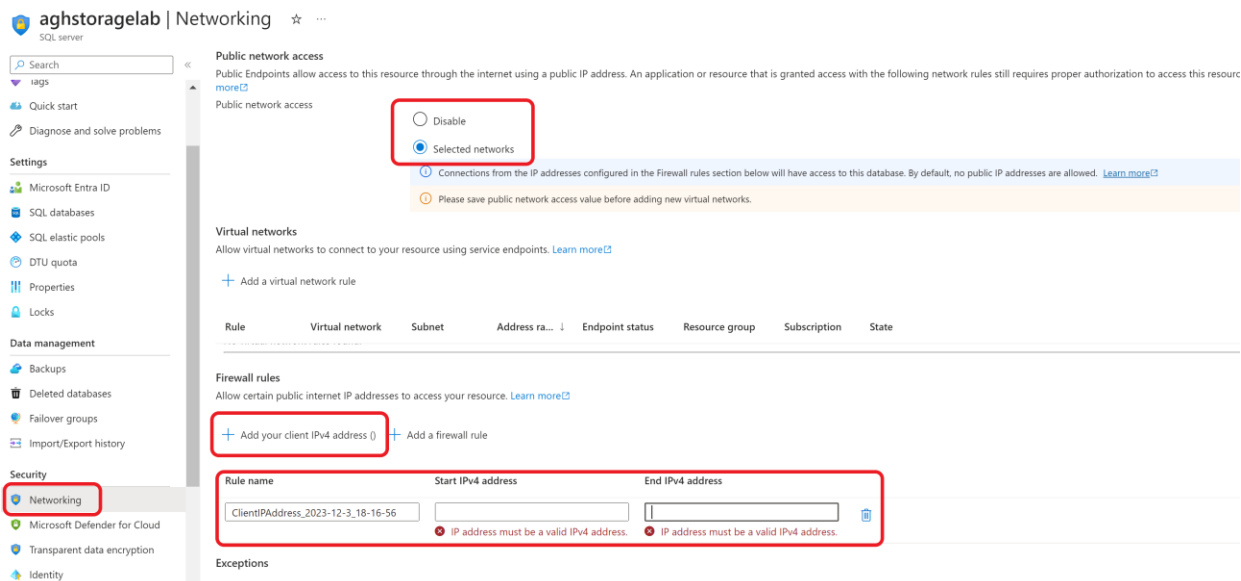
2. Podstawowe operacje na bazie danych

W tym ćwiczeniu przetestujemy podstawowe operacje z wykorzystaniem nowoutworzonej bazy danych.

- Wyszukujemy i otwieramy w *Azure Portal* bazę danych utworzoną w poprzednim ćwiczeniu.
- Z bocznego menu wybieramy **Query editor (preview)**



- Aby dostać się do edytora (z wykorzystaniem uwierzytelniania *SQL server* lub *Active Directory*) musimy dodać nasz lokalny adres IP do listy dopuszczonych adresów (ustawienia *Firewall* dla serwera bazy danych). Możemy to zrobić poprzez kliknięcie **Add your client IPv4 address** (reguły Firewall dla SQL servera – nie bazy danych!).



- Kontynuujemy poprzez wprowadzenie loginu i hasła (podanego przy tworzeniu serwera) lub przy pomocy *Active Directory*.
- Mamy do dyspozycji edytor, w którym możemy wykonywać bazodanowe zapytania. Przykładowe zapytania:

```
-- Utworzenie nowej tabeli
CREATE TABLE [User] (
    [ID] INTEGER NOT NULL IDENTITY,
    [FirstName] TEXT NULL,
    [LastName] TEXT NOT NULL
```

```
);

-- Wstawianie nowych użytkowników
INSERT INTO [dbo].[User](FirstName, LastName)
VALUES ('Robert', 'Lewandowski'), ('Arkadiusz', 'Milik')

-- Przeglądanie zawartości tabeli
SELECT * FROM [dbo].[User]
```

- Powyższe zapytania oraz inne operacje związane z zarządzaniem bazą danych można wykonać również w innych aplikacjach, takich jak:
 - Microsoft SQL Server Management Studio (MS SSMS)
 - Visual Studio
 - Visual Studio Code (z odpowiednimi wtyczkami, np. *SQL Server*)

3. Lokalna aplikacja

W tej części wykorzystamy gotową aplikację pochodzącą z dokumentacji Microsoft, która została zmodyfikowana na potrzeby laboratorium. Następnie połączymy ją z utworzoną bazą danych i uruchomimy lokalnie.

- Otwieramy wiersz poleceń i nawigujemy się do naszego folderu roboczego. Wywołujemy komendę:

```
git clone https://github.com/wojciechtobis/AGHStorageLab.git
```

- Otwieramy pobrany projekt w Visual Studio lub Visual Studio Code.
- W pliku *appsettings.json* podmieniamy *Connection string*. Docelowy *Connection string* możemy znaleźć w *Azure Portal*. Należy pamiętać o uzupełnieniu hasła (ustawionego przy tworzeniu serwera)

Home > testdb (aghstoragelab/testdb)

testdb (aghstoragelab/testdb) | Connection strings ...
SQL database

Search (Ctrl+J) <<

Overview
Activity log
Tags
Diagnose and solve problems
Quick start
Query editor (preview)

Power Platform

Power BI (preview)
Power Apps (preview)
Power Automate (preview)

Settings

Compute + storage

Connection strings

Properties

ADO.NET JDBC ODBC PHP Go

ADO.NET (SQL authentication)

Server=tcp:aghstoragelab.database.windows.net,1433;Initial Catalog=testdb;Persist Security Info=False;User ID=dbadmin;Password=(your_password);MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;

ADO.NET (Active Directory password authentication)

Server=tcp:aghstoragelab.database.windows.net,1433;Initial Catalog=testdb;Persist Security Info=False;User ID=(your_username);Password=(your_password);MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Authentication="Active Directory Password";

ADO.NET (Active Directory integrated authentication)

Server=tcp:aghstoragelab.database.windows.net,1433;Initial Catalog=testdb;Persist Security Info=False;User ID=(your_username);MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Authentication="Active Directory Integrated";

- W celu utworzenia niezbędnych do działania aplikacji obiektów bazodanowych uruchamiamy migracje (w celu utworzenia bazy danych zawierającej jakieś tabele) wykonując poniższe polecenia:

```
dotnet tool install -g dotnet-ef
```

```
dotnet ef database update
```

- Uwaga! W przypadku niepowodzenia należy utworzyć tabelę ręcznie (jak w poprzednim punkcie) za pomocą poniższego skryptu i przejść do kolejnego ćwiczenia.

```
CREATE TABLE [Todo] (  
    [ID] INTEGER NOT NULL IDENTITY,  
    [Description] TEXT NULL,  
    [CreatedDate] DATETIME NOT NULL,  
    CONSTRAINT [PK_Todo] PRIMARY KEY ([ID])  
);
```

- Uruchamiamy aplikację:

```
dotnet run
```

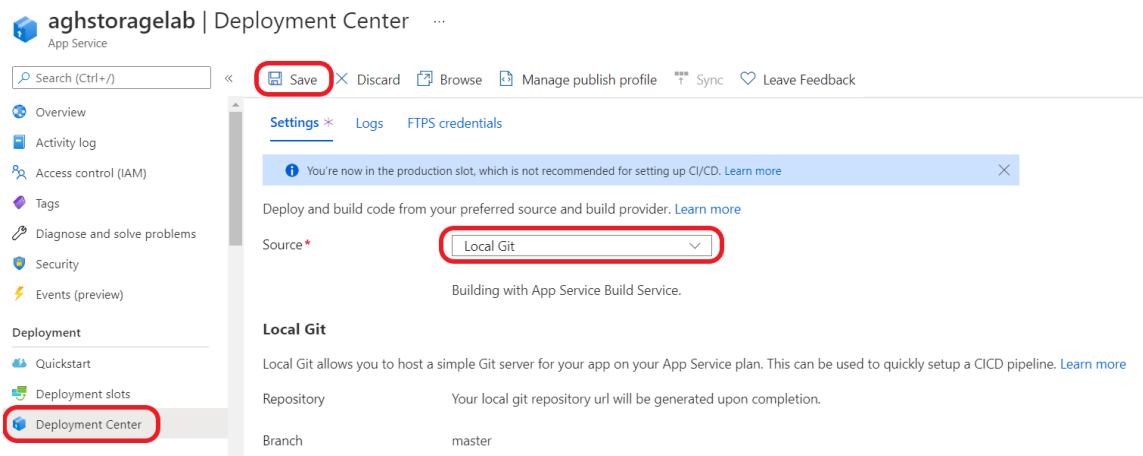
- Uruchamiamy przeglądarkę i pod adresem <https://localhost:<port>> (dokładny adres znajduje się w pliku *launchSettings.json*) powinniśmy mieć dostęp do naszej aplikacji.
- Po wprowadzeniu kilku wierszy, możemy sprawdzić zawartość tabeli w *Query editor* w Azure Portalu

```
SELECT * FROM [dbo].[Todo]
```

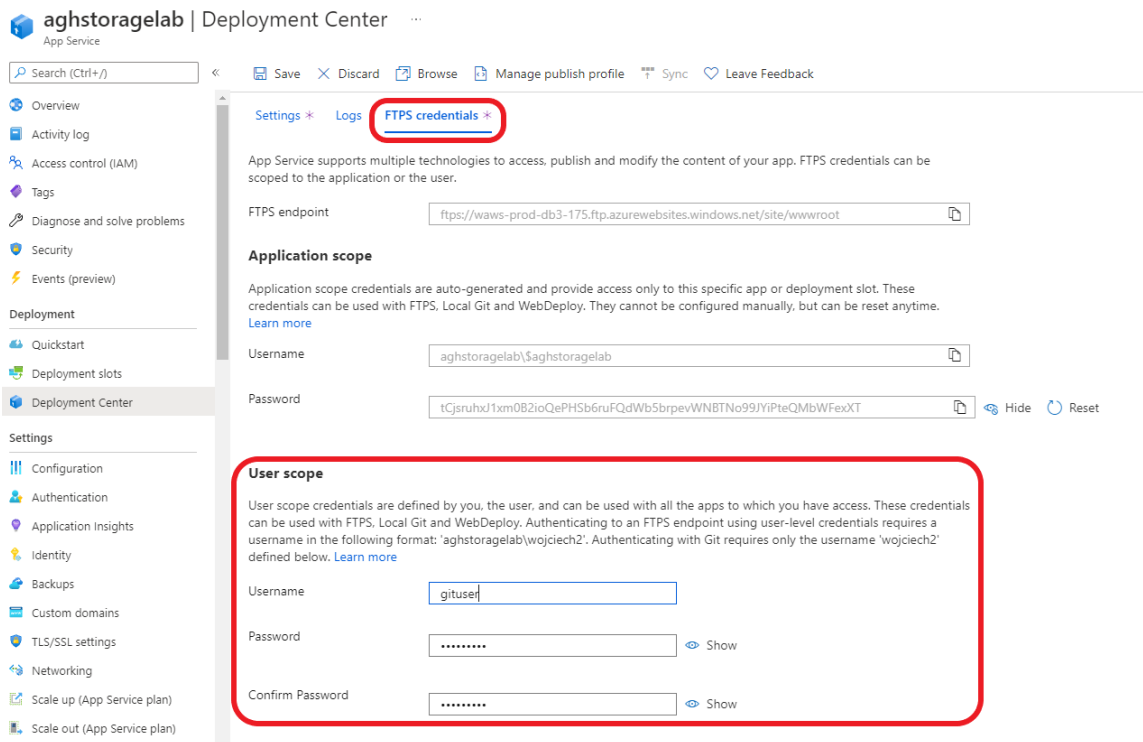
4. Aplikacja w Azure

Kolejne ćwiczenie ma na celu umieszczenie powstałej aplikacji w chmurze.

- Przenosimy się do grupy zasobów (utworzonej w ćwiczeniu 1) i tworzymy nowy zasób klikając **Create**.
- Wybieramy z listy *Web App* i klikamy **Create**.
- W sekcji *Project details* wybieramy odpowiednią subskrypcję i utworzoną grupę zasobów.
- W sekcji *Instance details* wypełniamy:
 - Unikalną nazwę naszej aplikacji,
 - **Code** jako metodę publikacji (**Publish**),
 - **.NET 6** jako framework wykorzystany do budowania aplikacji (**Runtime stack**),
 - **Windows** jako system operacyjny,
 - Region (najlepiej najbliższy naszej lokalizacji).
- Zmieniamy **Sku and size** na **F1**. (Stock-keeping-Unit, od tego zależą koszty)
- Klikamy **Review + create** i później **Create**.
- Przechodzimy do utworzonej aplikacji w *Azure Portal*.
- W *Deployment Center* wybieramy **Local Git** jako preferowane źródło, skąd dostarczane będą dane do wdrożenia aplikacji. Zapisujemy klikając **Save**.



- Pozostając w *Deployment Center*, przechodzimy do zakładki *Local Git/FTPS credentials*. Wprowadzamy login i hasło użytkownika, które będą wykorzystane do przesłania kodu na zdalne repozytorium w Azure. Zapisujemy klikając **Save**.



- W celu dodania *git remote* wykonujemy polecenie:

```
git remote add azure https://<app-name>.scm.azurewebsites.net/<app-name>.git
```

Gdzie <app-name> jest nazwą aplikacji, którą przed chwilą stworzyliśmy.

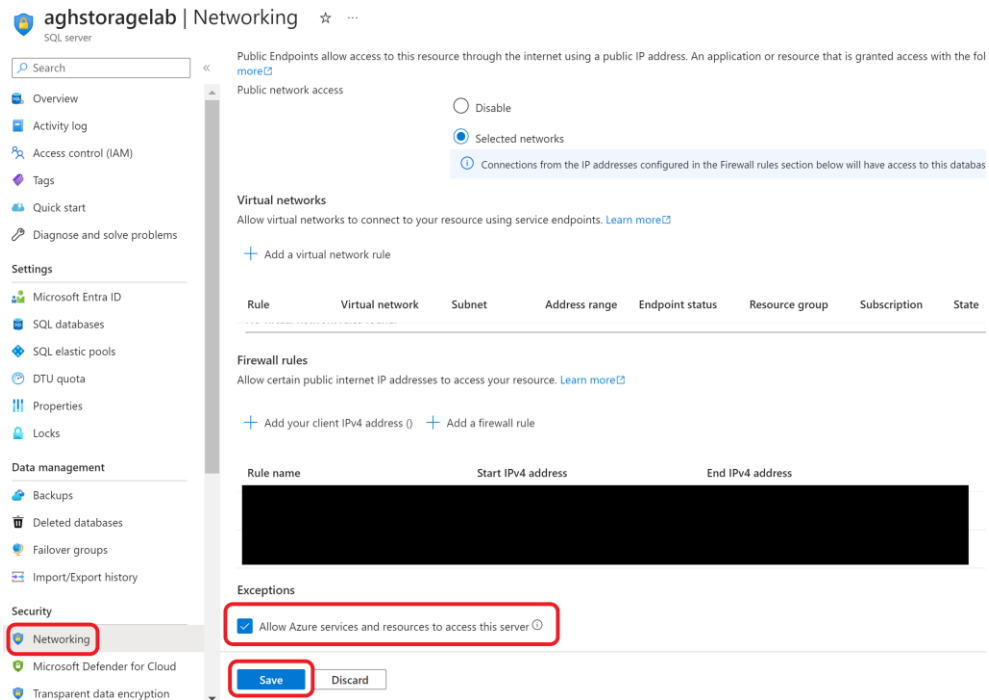
- Upewniamy się, że jesteśmy na branchu *master* i wykonujemy polecenie:

```
git add .
git commit -m '<your commit message>'
git push azure master
```

- Gdy będziemy poproszeni o podanie loginu i hasła, wpisujemy te wartości, które ustawiliśmy w *User Scope* (w *Deployment Center*, zakładka *FTPS credentials*).
- Sprawdzamy stan naszej aplikacji naszą wchodząc w odpowiedni adres URL:

`https://<app-name>.azurewebsites.net`

- Aplikacja powinna zwracać błąd. Jest to związane z zabezpieczeniami sieciowymi naszej bazy danych. Aby naprawić aplikację, z poziomu utworzonego w zadaniu 1 *SQL Servera* wchodzimy do **Networking**, a następnie zaznaczamy **Allow Azure services and resources to access this server**. Zapisujemy klikając **Save**.

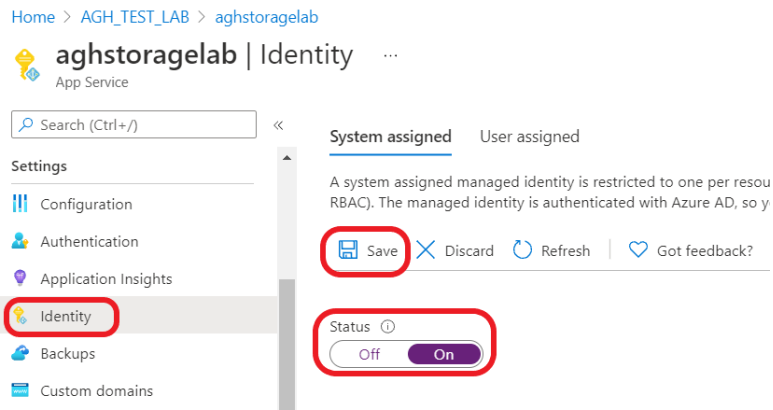


- Ponownie sprawdzamy działanie aplikacji. Tym razem wszystko powinno działać.

5. Aplikacja w Azure wykorzystująca Managed Identity

W poprzednim ćwiczeniu do połączenia z bazą danych wykorzystaliśmy *Connection string* zawierający nazwę i hasło użytkownika. Nie jest to bezpieczne rozwiązanie, ponieważ można dostać się do tych danych z poziomu chmury (np. poprzez *App Service Editor (preview)* -> *appsettings.json*). W tym ćwiczeniu spróbujemy temu zapobiec i wykorzystamy inny sposób autentykacji.

- Z poziomu utworzonego *App Service* wybieramy *Identity*, a następnie zmieniamy **Status** na **Yes**. Zapisujemy klikając **Save**.



- Następnie, w celu dodania odpowiednich ról bazodanowych dla utworzonego *identity* klikamy w **Azure role assignments**, a następnie **Add role assignment (Preview)**.
- Wybieramy odpowiedni **Scope (SQL)**, interesującą nas subskrypcję, zasób (nasza baza danych) i odpowiednią rolę (**Contributor**). Klikamy **Save**.

Add role assignment (Preview)

Scope ⓘ
SQL

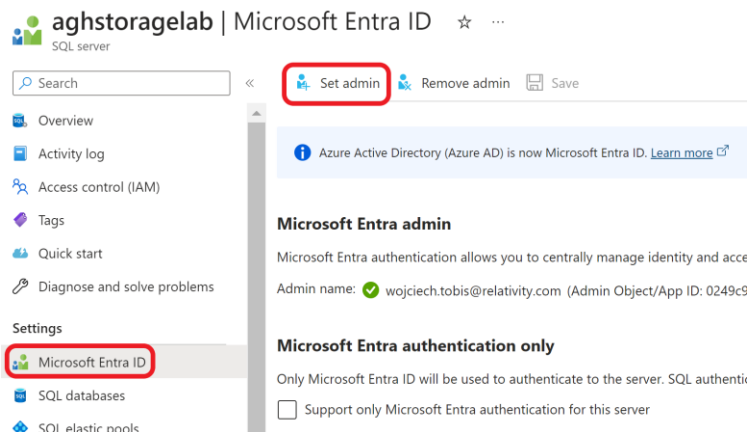
Subscription
Visual Studio Professional Subscription

Resource ⓘ
wttest ⓘ

Role ⓘ
Contributor ⓘ

[Learn more about RBAC](#)

- Następnie w serwerze bazy danych musimy zmienić *Azure Active Directory* admina z tego, którego wybraliśmy przy tworzeniu serwera, na nowoutworzone *identity* związane z aplikacją. W tym celu z poziomu serwera wybieramy **Microsoft Entra ID -> Set admin**. Po wybraniu odpowiedniego admina, zapisujemy klikając **Save**.



- Zmieniamy *Connection string* w pliku *appsettings.json* na:

```
"Server=tcp:<server-name>.database.windows.net;Authentication=Active Directory Device Code Flow; Database=<database-name>;"
```

- W pliku *Startup.cs* musimy zakomentować fragment:

```
//<Zadania 3 i 4>

    services.AddDbContext<MyDatabaseContext>(options =>

        options.UseSqlServer(Configuration.GetConnectionString("MyDbConnection")));

//</Zadania 3 i 4>
```

I odkomentować fragment:

```
//<Zadanie 5>

    services.AddDbContext<MyDatabaseContext>(options =>

    {

        SqlAuthenticationProvider.SetProvider(

            SqlAuthenticationMethod.ActiveDirectoryDeviceCodeFlow,

            new CustomAzureSQLAuthProvider());

        var sqlConnection = new

        SqlConnection(Configuration.GetConnectionString("MyDbConnection"));

        options.UseSqlServer(sqlConnection);

    });

//</Zadanie 5>
```

- Upewniamy się, że jesteśmy na branchu *master* i wykonujemy polecenie:

```
git add .
git commit -m '<your commit message>'
git push azure master
```

6. Utworzenie Storage Account

W tym ćwiczeniu utworzymy zasób Storage Account. W tym celu:

- Przenosimy się do grupy zasobów (utworzonej w ćwiczeniu 1) i tworzymy nowy zasób klikając **Create**.
- Wybieramy z listy *Storage account* i klikamy **Create**.
- Wybieramy globalnie unikalną nazwę Storage account i zmieniamy **Redundancy** na **Locally-redundant storage (LRS)**.
- Klikamy **Review** i później **Create**.

Create a storage account ...

Basics Advanced Networking Data protection Encryption Tags Review

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription *

Resource group *
[Create new](#)

Instance details

Storage account name ⓘ *

Region ⓘ *
[Deploy to an edge zone](#)

Performance ⓘ * ☒ **Standard:** Recommended for most scenarios (general-purpose v2 account)
☐ **Premium:** Recommended for scenarios that require low latency.

Redundancy ⓘ *

- Przechodzimy do utworzonego zasobu i dodajemy pierwszy plik do *Blob Storage*. Zanim do tego dojdzie, musimy utworzyć kontener. W tym celu klikamy **Containers** -> **+ Container**, wprowadzamy nazwę i zatwierdzamy przez **Create**.

Home > aghstoragelab_1701627703349 | Overview > aghstoragelab

aghstoragelab | Containers

Search containers by prefix

Name	Last modified	Anonymous access level
<input type="checkbox"/> \$logs	12/3/2023, 7:22:14 PM	Private

New container

Name *

Anonymous access level ⓘ
Private (no anonymous access)

The access level is set to private because anonymous access is disabled on this storage account.

Advanced

- Wchodzimy do nowoutworzonego kontenera i dodajemy uprawnienia dla nas i naszej aplikacji webowej poprzez wybranie **Access Control (IAM)** -> **+ Add** -> **Add role assignment**.
- Wybieramy *Storage Blob Data Contributor* jako rolę.
- Na kolejnym ekranie wybieramy członków – nas samych oraz utworzoną wcześniej web aplikację.



Add role assignment ...

[Role](#) **[Members](#)** [Conditions \(optional\)](#) [Review + assign](#)

Selected role Storage Blob Data Contributor

Assign access to
☐ User, group, or service principal
☒ Managed identity

Members [+ Select members](#)

Name	Object ID	Type	
Wojciech Tobis		User	
aghstoragelab		App Service ⓘ	

Description Optional

- Klikamy **Review + assign**.

7. Podstawowe operacje na plikach

W tym ćwiczeniu wykonamy podstawowe operacje na plikach w Blob storage.

- Wchodzimy do nowoutworzonego kontenera i klikamy **Upload**.
- Wybieramy 2 pliki (jeden png, drugi tekstowy). Można wykorzystać pliki z pobranego repozytorium – *example.txt* i *relativity-logo.png*. Przed ostateczną akceptacją, przeglądamy dostępne opcje, takie jak *Access tier* czy *Block size*.
- Klikamy plik tekstowy (np. *example.txt*) i przeglądamy dostępne metadane w zakładce *Overview*.
- Przechodzimy do zakładki *Edit* i modyfikujemy plik. Następnie klikamy **Save** oraz pobieramy zmodyfikowany plik klikając **Download**.
- Ostatnim zadaniem w tej części jest wygenerowanie SAS tokenu. W tym celu przechodzimy do zakładki *Generate SAS* i przeglądamy dostępne opcje. Pozostawiamy je bez zmian i klikamy **Generate SAS token and URL**.

example.txt ...

Blob

Save Discard Download Refresh Delete

Overview Versions Snapshots Edit **Generate SAS**

A shared access signature (SAS) is a URI that grants restricted access to an Azure Storage blob. Use it when you want range without sharing your storage account key. [Learn more about creating an account SAS](#)

Signing method

☒ Account key ☐ User delegation key

Signing key ⓘ

Key 1 ▼

Stored access policy

None ▼

Permissions * ⓘ

Read ▼

Start and expiry date/time ⓘ

Start

12/03/2023 8:12:54 PM

(UTC+01:00) Sarajevo, Skopje, Warsaw, Zagreb ▼

Expiry

12/04/2023 4:12:54 AM

(UTC+01:00) Sarajevo, Skopje, Warsaw, Zagreb ▼

Allowed IP addresses ⓘ

for example, 168.1.5.65 or 168.1.5.65-168.1....

Allowed protocols ⓘ

☒ HTTPS only ☐ HTTPS and HTTP

Generate SAS token and URL

- Kopiuujemy wygenerowany token i próbujemy dostać się do naszego pliku z poziomu przeglądarki.

8. Wykorzystanie plików w aplikacji webowej

W tej części wykorzystamy utworzony wcześniej obrazek w naszej aplikacji webowej.

- Wybieramy nasz obrazek z poziomu kontenera i przechodzimy do *Generate SAS*, a następnie klikamy **Generate SAS token and URL**. **Ważne!** Musimy upewnić się, że *Signing method* jest ustawione na **Account key** a *Signing key* na **Key 1**.
- Kopiujemy wygenerowany *Blob SAS URL*.
- Wklejamy do pliku *Index.cshtml* (*Views/Todos/Index.cshtml*, pomiędzy 7 a 8 liniijkę) poniższy kod:

```
<div></div>
```

- Upewniamy się, że jesteśmy na branchu *master* i wykonujemy polecenie:

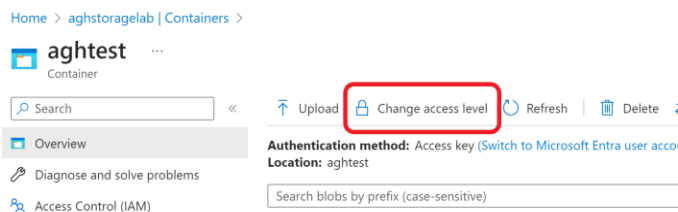
```
git add .
git commit -m '<your commit message>'
git push azure master
```

- Sprawdzamy, czy nasza aplikacja działa poprawnie i obrazek pojawił się w odpowiednim miejscu.

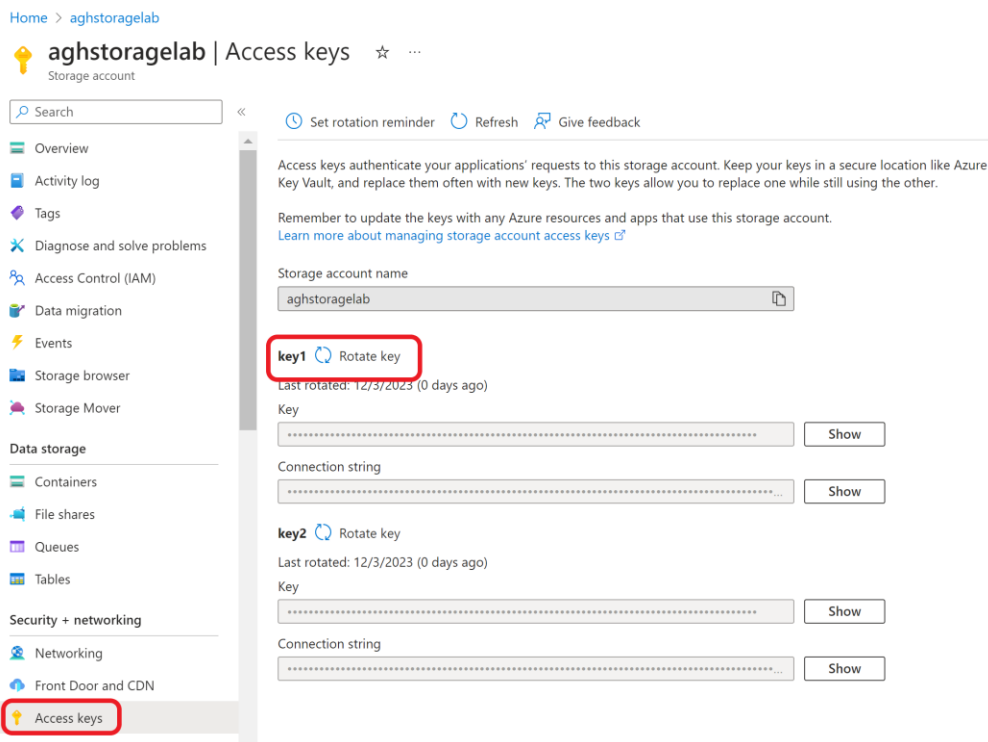
9. Rotacja Access Keys

Ostatnim zadaniem jest rotacja Access Keys. Operację tę wykonujemy w momencie, kiedy chcemy zabrać użytkownikom posługującym się tymi kluczami (lub wygenerowanymi na ich podstawie SAS tokenami – jak w naszym przypadku) dostęp do plików, lub cyklicznie, jeśli jest to wymagane przez wymogi bezpieczeństwa.

- Zmieniamy poziom dostępu naszego kontenera na prywatny klikając na **Change access level** z poziomu kontenera.



- Upewniamy się, że w naszej aplikacji w dalszym ciągu mamy dostęp do obrazka (dzięki SAS tokenowi).
- Przechodzimy do naszego *Storage account*, a następnie do *Access keys*.
- Klikamy **Rotate key** przy kluczu *key1* (tym, dla którego tworzyliśmy SAS token w poprzednim punkcie). Potwierdzamy chęć ponownego wygenerowania tego klucza.



- Odświeżamy aplikację. Po rotacji *Access Key* nasz *SAS token* powinien przestać działać i powinniśmy widzieć brakujący obrazek oraz błąd 403 w konsoli przeglądarki.

Index

[Create New](#)

Description	Created Date	
My desc	2023-12-12	Edit Details Delete
My desc 2	2023-12-22	Edit Details Delete
My desc 3v2	2023-12-05	Edit Details Delete

10. Usuwanie zasobów

Na koniec musimy usunąć utworzone zasoby.

- Z poziomu grupy zasobów wybieramy **Delete resource group**.

[Home](#) > [wtest](#) > [wt_lab_1 \(wtest/wt_lab_1\)](#) >

AGH_TEST_LAB

Resource group

[+ Create](#) [Edit columns](#) [Delete resource group](#) [Refresh](#) [Export to CSV](#)

Overview

Activity log

Access control (IAM)

Tags

Resource visualizer

Events

Settings

Deployments

Security

Policies

Properties

Locks

Essentials

Subscription (Move) : [Visual Studio Professional Subscription](#)

Subscription ID : 824863d2-9ad0-46b8-96e5-886f6de3dd00

Tags (Edit) : [Click here to add tags](#)

Resources

Recommendations

[Type == all](#) [Location == all](#) [Add filter](#)

Showing 1 to 5 of 5 records. ☐ Show hidden types

☐ Name ↑↓

☐ aghstoragelab

☐ aghstoragelab

- Potwierdzamy zamiar usunięcia i klikamy **Delete**.

Wykorzystane zasoby

- <https://docs.microsoft.com/en-us/azure/app-service/tutorial-dotnetcore-sqldb-app>
- <https://docs.microsoft.com/en-us/sql/connect/ado-net/sql/azure-active-directory-authentication>
- <https://docs.microsoft.com/en-us/azure/app-service/app-service-web-tutorial-connect-msi>
- <https://docs.microsoft.com/en-us/azure/app-service/overview-managed-identity>
- <https://docs.microsoft.com/en-us/azure/app-service/scenario-secure-app-authentication-app-service>
- <https://docs.microsoft.com/en-us/azure/app-service/deploy-local-git>
- <https://docs.microsoft.com/en-us/azure/app-service/deploy-configure-credentials>