Configure Microsoft Entra Id Service Principal with Access to Fabric SQL Database

Published: December 14, 2024

Microsoft Fabric is an exciting all-in-one platform to build and manage automation and data-centric solutions. Recently a new feature was announced bringing the power and flexibility of Azure SQL Server into Fabric – Fabric SQL database. Previously, we had to link to or mirror an Azure SQL Server database into Fabric for usage and analytics. Now our traditional OLTP database, which supports column stores for analytics, is baked into Fabric.

While still a preview feature, it is functional at a minimum level for production while providing automatic backups, disaster recovery and replication, and full security integration with Microsoft Entra ID.

How to connect to and use the Fabric SQL database

The article below walks through the steps to configure a Fabric tenant for Fabric SQL database, configure a Microsoft Entra ID Service Principal and applying to a Fabric SQL database, and finally testing the connectivity from data tools and Python.

The following topics are covered:

1. Install or Verify the Required Tools and Drivers
2. Create a Service Principal and Security Group in Microsoft Entra ID
3. Verify or Change Required Fabric Tenant Settings
4. Create and Configure Fabric SQL Database
5. Verify Connectivity with SQL Server Management Studio
6. Connect to Fabric SQL Database from Python

Install or Verify the Required Tools and Drivers

1. Install ODBC Driver 18 for SQL Server
2. Install preferred database tool that supports authentication with Entra ID Service Principals:

- SQL Server Management Studio (SSMS) - Recommend version 20.X
- Azure Data Studio
- Visual Studio Code with SQL extensions

3. Install required Python Library:

- pyodbc

Create a Service Principal and Security Group in Microsoft Entra ID

1. Entra Id App Registration:

    - Azure Portal > Entra ID > Manage > App Registrations
    - New Registration
    - Provide "Name"
    - Leave default "Supported Account Types" or change per enterprise policies

2. For the newly created App Registration:

    - Manage > Certificates & Secrets
    - Client Secrets > New Client Secret
    - Provide "Description"
    - Set "Expires" to the recommended value or change per enterprise policies
    - Immediately after the secret is created, copy and save the Value
    - For production, this should be saved in a Key Vault and accessed programmatically

3. Gather required values and save for later steps:

    - Application (client) ID
    - Directory (tenant) ID
    - Secret Value

4. Create Security Group for the new App Registration:

    - Azure Portal > Entra ID > Manage > Groups
    - New Group
    - Group Type: Security
    - Provide “Group Name”

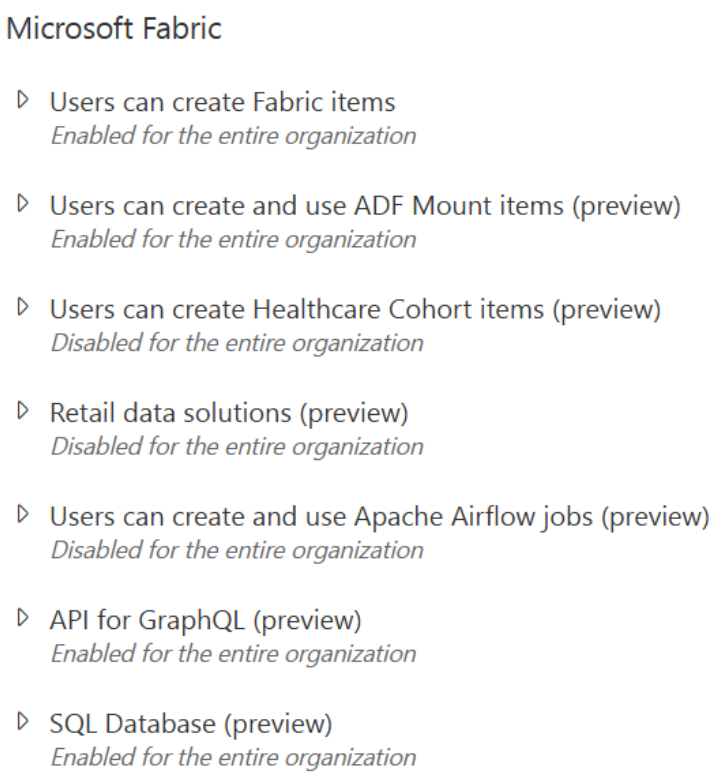
- Membership Type: Assigned

5. Add Members to New Group:

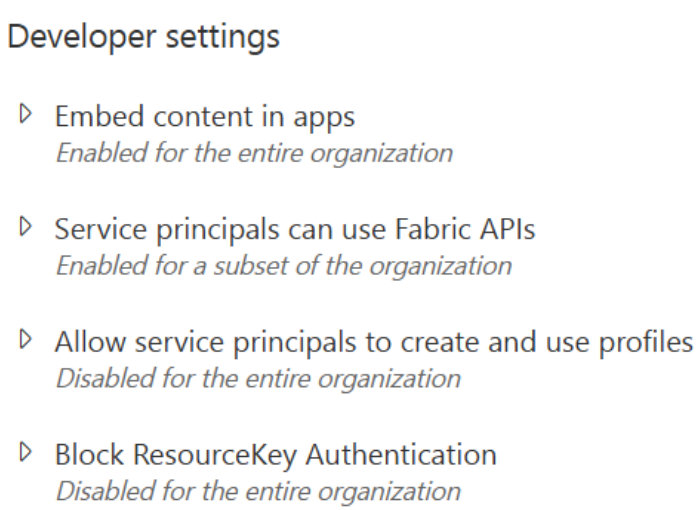
   - Add Member
   - Select the App Registration created earlier which will list as type "Service Principal"

Verify or Change Required Fabric Tenant Settings

1. Log into Fabric Portal - https://app.fabric.microsoft.com/home
2. Click on "Settings" > "Admin Portal"
3. Review / update: Tenant Settings > Microsoft Fabric > SQL Database (preview) > Enabled
4. Review / update: Tenant Settings > Developer Settings > Service Principals Can Use Fabric APIs > Enabled



Enable Fabric SQL Database Feature



Allow Service Principals to Use Fabric API

Create and Configure Fabric SQL Database

1. While logged into the Fabric Portal, Switch to the "Databases" persona
2. Choose an existing Fabric workspace or create a new workspace for the Fabric SQL database and assign to your Fabric Capacity
3. Create a new Fabric SQL database or configure an existing
4. Add Item Level permissions to the Fabric SQL database:
    - In the workspace view where resources are listed, find the Fabric SQL database and click "More Options" (...), choose "Manage Permissions"
    - In "Direct Access", click "Add User"
    - Search for the Security group created earlier that has the Service Principal as a member
    - Choose the desired additional permissions – read to the database is required at a minimum: Read all data using SQL database and Read all data using SQL analytics end point
    - Click "Grant"

5. Configure Fabric SQL database permissions:
    - Click on the Fabric database name which opens the Explorer
    - Click on "Security" tab and "Manage SQL Security"
    - Create a new role such as "appuser_rw"
    - Check the allowed schemas and operations (Select, Insert, Update, etc.)
    - Click "Manage Access" and search for the Security group created earlier that has the Service Principal as a member
    - Click "Save"
6. Review the generated ODBC connection string:
    - Click "Settings" > "Connection Strings" > "ODBC":
    - Example Connection String: Driver={ODBC Driver 18 for SQL Server}; Server=<fabric sql server>,1433; Database=<fabric sql database>; Encrypt=yes; TrustServerCertificate=no; Authentication=ActiveDirectoryServicePrincipal; UID=<service principal client/app ID>; PWD=<service principal secret value>
    - Note the following values:
        - Driver
        - Server
        - Database
    - Save these values for the next steps to connect from SSMS and Python

---

Verify Connectivity with SQL Server Management Studio (SSMS)

1. Connect to Server:
    - Server type: Database Engine
    - Server name: Server value from Fabric SQL database connection string
    - Authentication: Microsoft Entra Service Principal

- Username: The App Registration Application (client) ID
- Password: The App Registration Secret Value
- In "Connection Properties" set the database to value from Fabric SQL database connection string
- If this is not set, the default connection is to master database and the login attempt will fail

2. Verify configuration:
   - In the Object Explorer, expand the Fabric SQL database and "Security"
   - The configured security group should be listed
   - Expand "Roles"
   - The configured database role should be listed

Connect to Fabric SQL Database from Python

1. To connect from Python to the Fabric SQL Database, we will use the PyODBC library and the ODBC 18.0 driver for Windows.
2. Below is a sample function to open a database connection and a function to run a query.

```
# Azure SQL Database details
'''
Connection String
'''
DB_SERVER = '<azure sql server>'
DB_NAME = '<azure sql database>'
DB_USER = '<local database user>'
DB_TOKEN = '<password>'
DB_DRIVER = '{ODBC Driver 18 for SQL Server}'
DB_AUTH_TYPE = ''
```

```
# Fabric SQL Database details
'''
Connection String
'''
DB_SERVER_F = '<fabric sql server>'
DB_NAME_F = '<fabric sql database>'
DB_USER_F = '<service principal client/app ID>'
DB_TOKEN_F = '<service principal secrect value>'
DB_DRIVER_F = "{ODBC Driver 18 for SQL Server}"
DB_AUTH_TYPE_F = 'ActiveDirectoryServicePrincipal'

# Database - Open Connection
def f_open_db_connection(azure_env):
    '''
    Open a database connection
    Paramters:
    - azure_env: Azure or Fabric environment - possible values: 'sql' or 'fabric'
    '''
    if (pf.system() =='Windows' and azure_env == 'sql'):
        l_db_driver = DB_DRIVER
        l_db_server = DB_SERVER
        l_db_name = DB_NAME
        l_db_uid = DB_USER
        l_db_pwd = DB_TOKEN
        l_db_auth_type = DB_AUTH_TYPE
        l_db_conn_enc = 'Yes'
    elif (pf.system() =='Windows' and azure_env == 'fabric'):
        l_db_driver = DB_DRIVER_F
        l_db_server = DB_SERVER_F
        l_db_name = DB_NAME_F
        l_db_uid = DB_USER_F
        l_db_pwd = DB_TOKEN_F
```

```
        l_db_auth_type = DB_AUTH_TYPE_F
        l_db_conn_enc = 'Yes'
    db_conn = db.connect('DRIVER=' + l_db_driver + ';'
                         'SERVER=' + l_db_server + ';'
                         'DATABASE=' + l_db_name + ';'
                         'UID=' + l_db_uid + ';PWD=' + l_db_pwd +';'
                         'Encrypt=' + l_db_conn_enc +';'
                         'TrustServerCertificate=no;'
                         'Authentication=' + l_db_auth_type + ';'
    return db_conn
# Database - Read Record
def f_get_record(sql_statement, params):
    # Get records from Log table
    l_conn = f_open_db_connection(azure_env='fabric')
    cursor = l_conn.cursor()
    sql_statement = 'select * from [dbo].[log_table] order by log_date desc'
    sql_command = (sql_statement)
    cursor.execute(sql_command, params)
    for row in cursor:
        print('row = %r' % (row,))
    f_close_db_connection(l_conn)
    return
```

William Ricci is a serial entrepreneur with over twenty-five years of experience in IT and system architecture, development, and data platforms. As founder of Willow River Solutions, he provides fully-integrated and AI solutions to customers across many industries.