Controlling and granting database access

03/16/2018 • 11 minutes to read • Contributors

In this article

Unrestricted administrative accounts
Additional server-level administrative roles
Non-administrator users
Groups and roles
Permissions
Next steps

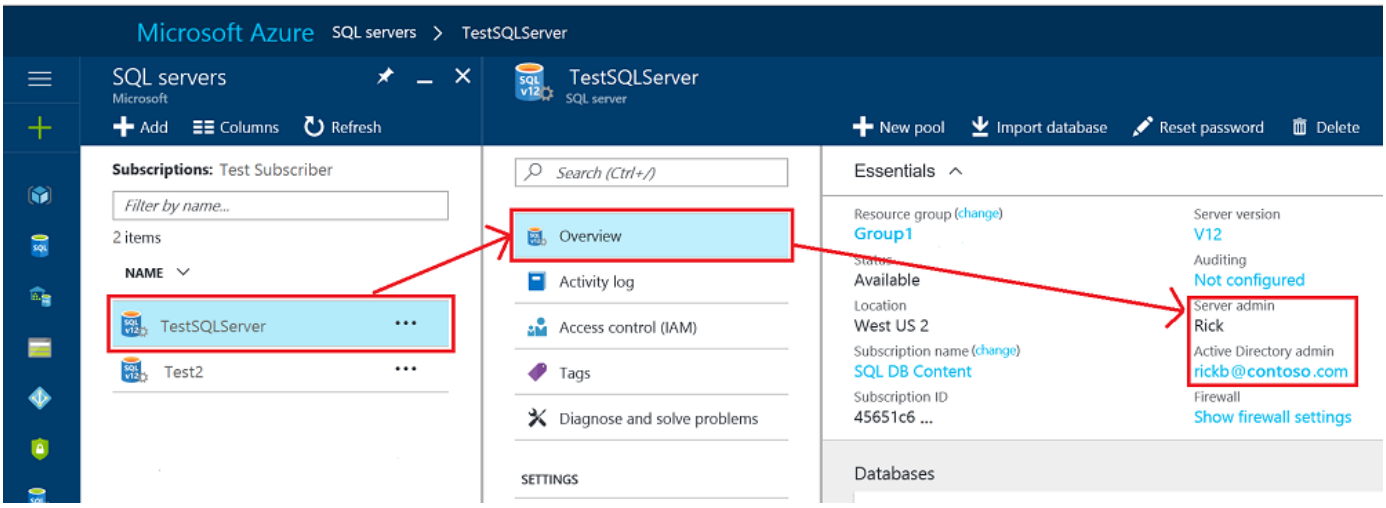
After firewall rules have been configured, people can connect to a SQL Database as one of the administrator accounts, as the database owner, or as a database user in the database.

> **Note**
>
> This topic applies to Azure SQL server, and to both SQL Database and SQL Data Warehouse databases that are created on the Azure SQL server. For simplicity, SQL Database is used when referring to both SQL Database and SQL Data Warehouse.

> **Tip**
>
> For a tutorial, see Secure your Azure SQL Database.

Unrestricted administrative accounts

There are two administrative accounts (**Server admin** and **Active Directory admin**) that act as administrators. To identify these administrator accounts for your SQL server, open the Azure portal, and navigate to the properties of your SQL server.



- **Server admin**
  When you create an Azure SQL server, you must designate a **Server admin login**. SQL server creates that account as a login in the master database. This account connects using SQL Server authentication (user name and password). Only one of these accounts can exist.
- **Azure Active Directory admin**
  One Azure Active Directory account, either an individual or security group account, can also be configured as an administrator. It is optional to configure an Azure AD administrator, but an Azure AD administrator must be configured if you want to use Azure AD accounts to connect to SQL Database. For more information about configuring Azure Active Directory access, see Connecting to SQL Database or SQL Data Warehouse By Using Azure Active Directory Authentication and SSMS support for Azure AD MFA with SQL Database and SQL Data Warehouse.

The **Server admin** and **Azure AD admin** accounts has the following characteristics:

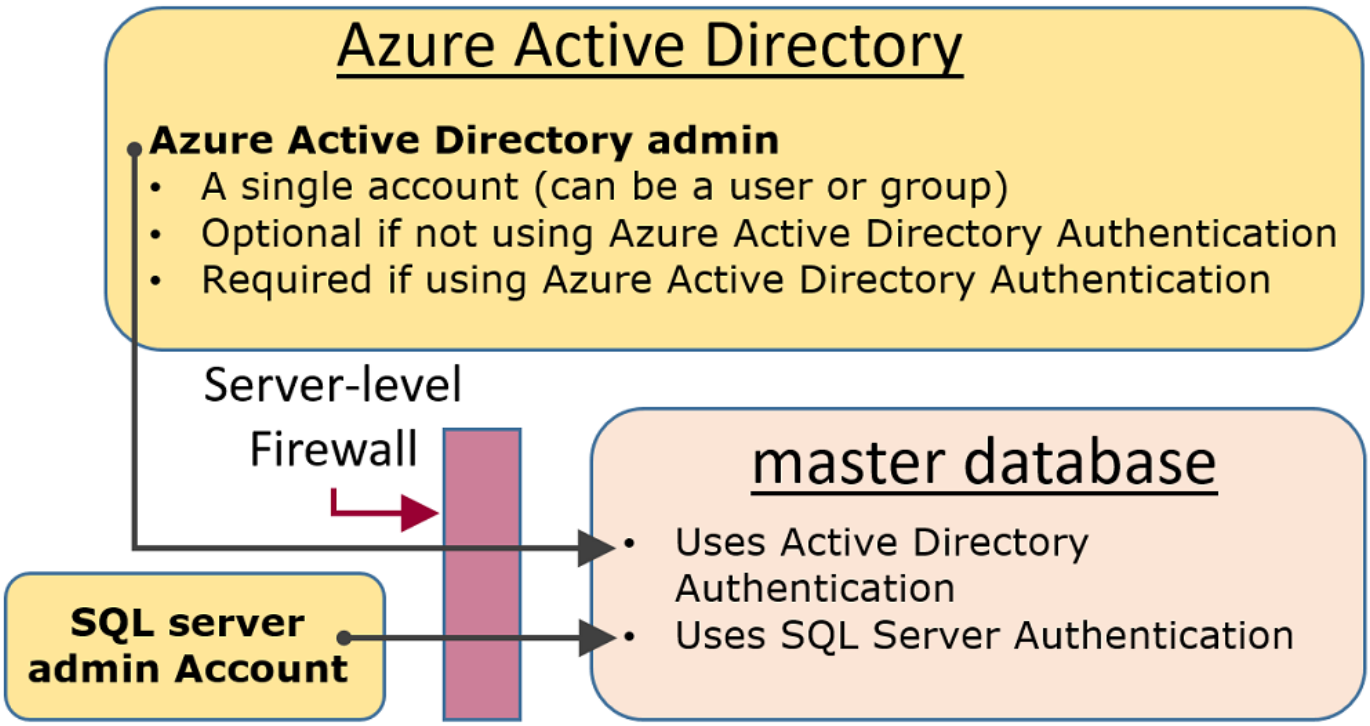
- These are the only accounts that can automatically connect to any SQL Database on the server. (To connect to a user database, other accounts must either be the owner of the database, or have a user account in the user database.)
- These accounts enter user databases as the `dbo` user and they have all the permissions in the user databases. (The owner of a user database also enters the database as the `dbo` user.)
- These accounts do not enter the `master` database as the `dbo` user and they have limited permissions in master.
- These accounts are not members of the standard SQL Server `sysadmin` fixed server role, which is not available in SQL database.
- These accounts can create, alter, and drop databases, logins, users in master, and server-level firewall rules.
- These accounts can add and remove members to the `dbmanager` and `loginmanager` roles.
- These accounts can view the `sys.sql_logins` system table.

Configuring the firewall

When the server-level firewall is configured for an individual IP address or range, the **SQL server admin** and the **Azure Active Directory admin** can connect to the master database and all the user databases. The initial server-level firewall can be configured through the Azure portal, using PowerShell or using the REST API. Once a connection is made, additional server-level firewall rules can also be configured by using Transact-SQL.

Administrator access path

When the server-level firewall is properly configured, the **SQL server admin** and the **Azure Active Directory admin** can connect using client tools such as SQL Server Management Studio or SQL Server Data Tools. Only the latest tools provide all the features and capabilities. The following diagram shows a typical configuration for the two administrator accounts.



When using an open port in the server-level firewall, administrators can connect to any SQL Database.

Connecting to a database by using SQL Server Management Studio

For a walk-through of creating a server, a database, server-level firewall rules, and using SQL Server Management Studio to query a database, see Get started with Azure SQL Database servers, databases, and firewall rules by using the Azure portal and SQL Server Management Studio.

> **Important**
>
> It is recommended that you always use the latest version of Management Studio to remain synchronized with updates to Microsoft Azure and SQL Database. Update SQL Server Management Studio.

Additional server-level administrative roles

In addition to the server-level administrative roles discussed previously, SQL Database provides two restricted administrative roles in the master database to which user accounts can be added that grant permissions to either create databases or manage logins.

Database creators

One of these administrative roles is the **dbmanager** role. Members of this role can create new databases. To use this role, you create a user in the `master` database and then add the user to the **dbmanager** database role. To create a database, the user must be a user based on a SQL Server login in the master database or contained database user based on an Azure Active Directory user.

1. Using an administrator account, connect to the master database.

2. Optional step: Create a SQL Server authentication login, using the CREATE LOGIN statement. Sample statement:

   Copy

   ```
   CREATE LOGIN Mary WITH PASSWORD = '<strong_password>';
   ```

   > **Note**
   >
   > Use a strong password when creating a login or contained database user. For more information, see **Strong Passwords**.

   To improve performance, logins (server-level principals) are temporarily cached at the database level. To refresh the authentication cache, see DBCC FLUSHAUTHCACHE.

3. In the master database, create a user by using the CREATE USER statement. The user can be an Azure Active Directory authentication contained database user (if you have configured your environment for Azure AD authentication), or a SQL Server authentication contained database user, or a SQL Server authentication user based on a SQL Server authentication login (created in the previous step.) Sample statements:

   Copy

   ```
   CREATE USER [mike@contoso.com] FROM EXTERNAL PROVIDER;
   CREATE USER Tran WITH PASSWORD = '<strong_password>';
   CREATE USER Mary FROM LOGIN Mary;
   ```

4. Add the new user, to the **dbmanager** database role by using the ALTER ROLE statement. Sample statements:

```
ALTER ROLE dbmanager ADD MEMBER Mary;
ALTER ROLE dbmanager ADD MEMBER [mike@contoso.com];
```

> **Note**
>
> The dbmanager is a database role in master database so you can only add a database user to the dbmanager role. You cannot add a server-level login to database-level role.

5. If necessary, configure a firewall rule to allow the new user to connect. (The new user might be covered by an existing firewall rule.)

Now the user can connect to the master database and can create new databases. The account creating the database becomes the owner of the database.

Login managers

The other administrative role is the login manager role. Members of this role can create new logins in the master database. If you wish, you can complete the same steps (create a login and user, and add a user to the **loginmanager** role) to enable a user to create new logins in the master. Usually logins are not necessary as Microsoft recommends using contained database users, which authenticate at the database-level instead of using users based on logins. For more information, see Contained Database Users - Making Your Database Portable.

Non-administrator users

Generally, non-administrator accounts do not need access to the master database. Create contained database users at the database level using the CREATE USER (Transact-SQL) statement. The user can be an Azure Active Directory authentication contained database user (if you have configured your environment for Azure AD authentication), or a SQL Server authentication contained database user, or a SQL Server authentication user based on a SQL Server authentication login (created in the previous step.) For more information, see Contained Database Users - Making Your Database Portable.

To create users, connect to the database, and execute statements similar to the following examples:

```
CREATE USER Mary FROM LOGIN Mary;
CREATE USER [mike@contoso.com] FROM EXTERNAL PROVIDER;
```

Initially, only one of the administrators or the owner of the database can create users. To authorize additional users to create new users, grant that selected user the `ALTER ANY USER` permission, by using a statement such as:

```
GRANT ALTER ANY USER TO Mary;
```

To give additional users full control of the database, make them a member of the **db_owner** fixed database role using the `ALTER ROLE` statement.

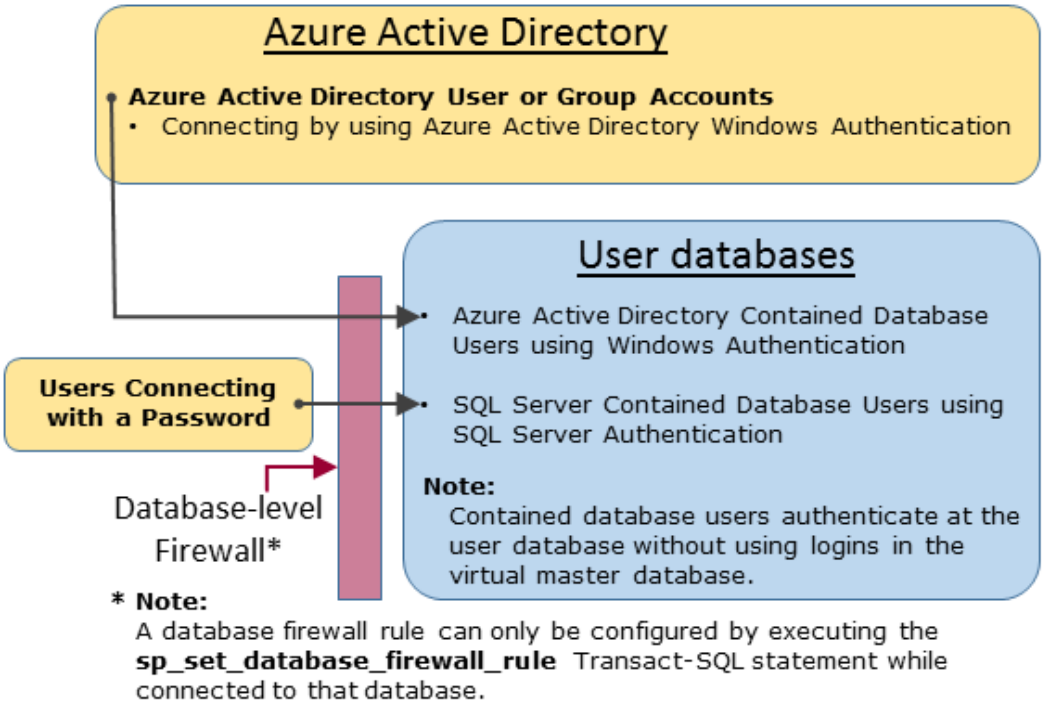
> **Note**
>
> The most common reason to create database users based on logins, is when you have SQL Server authentication users that need access to multiple databases. Users based on logins are tied to the login, and only one password that is maintained for that login. Contained database users in individual databases are each individual entities and each maintains its own password. This can confuse contained database users if they do not maintain their passwords as identical.

Configuring the database-level firewall

As a best practice, non-administrator users should only have access through the firewall to the databases that they use. Instead of authorizing their IP addresses through the server-level firewall and giving them access to all databases, use the sp_set_database_firewall_rule statement to configure the database-level firewall. The database-level firewall cannot be configured by using the portal.

Non-administrator access path

When the database-level firewall is properly configured, the database users can connect using client tools such as SQL Server Management Studio or SQL Server Data Tools. Only the latest tools provide all the features and capabilities. The following diagram shows a typical non-administrator access path.



Groups and roles

Efficient access management uses permissions assigned to groups and roles instead of individual users.

- When using Azure Active Directory authentication, put Azure Active Directory users into an Azure Active Directory group. Create a contained database user for the group. Place one or more database users into a database role and then assign permissions to the database role.
- When using SQL Server authentication, create contained database users in the database. Place one or more database users into a database role and then assign permissions to the database role.

The database roles can be the built-in roles such as **db_owner**, **db_ddladmin**, **db_datawriter**, **db_datareader**, **db_denydatawriter**, and **db_denydatareader**. **db_owner** is commonly used to grant full permission to only a few users. The other fixed database roles are useful for getting a simple database in development quickly, but are not recommended for most production databases. For example, the **db_datareader** fixed database role grants read access to every table in the database, which is usually more than is strictly necessary. It is far better to use the CREATE ROLE statement to create your own user-defined database roles and carefully grant each role the least permissions necessary for the business need. When a user is a member of multiple roles, they aggregate the permissions of them all.

Permissions

There are over 100 permissions that can be individually granted or denied in SQL Database. Many of these permissions are nested. For example, the `UPDATE` permission on a schema includes the `UPDATE` permission on each table within that schema. As in most permission systems, the denial of a permission