

SELF-HOST > DEPLOY & CONFIGURE > CONFIGURATION OPTIONS

Connect to an External MSSQL Database

View in the help center:

<https://bitwarden.com/help/external-db/>

Connect to an External MSSQL Database

By default, self-hosted instances of Bitwarden will use a Microsoft SQL Server (MSSQL) database image created as a normal part of [installation setup](#), however you configure Bitwarden to use an external MSSQL database.

Note

Bitwarden only **supports and recommends SQL Server 2022**. Mainstream support for Server 2017 and server 2019 have ended. Deprecation of support for a specific SQL server version will be noted here and in the [release notes](#) for a given release if Bitwarden implements features that are not available on a specific version of SQL Server.

Learn about the system requirements for SQL Server on [Windows](#) and [Linux](#).

Setup external database

To setup your self-hosted instance with an external database:

⇒ Docker

1. Create a new MSSQL database.
2. **(Recommended)** Create a dedicated DBO for your database.
3. In the `global.override.env` file for your server, edit the `globalSettings__sqlServer__connectionString=` value for the following information:
 - Replace `"Data Source=tcp:mssql,1433";` with your MSSQL server name, for example `"Data Source=protocol:server_url,port"`.
 - Replace the `vault` in `Initial Catalog=vault`; with your database name.
 - Replace the `sa` in `User ID=sa;` with your DBO User ID.
 - Replace the `<default_pw>` in `Password=<default_pw>;` with your DBO password.
4. Save your changes to `global.override.env`.
5. Start Bitwarden (`./bitwarden.sh start`).

Once the above steps are complete, you can test the connection by creating a new user through the web vault and querying the external `vault` database for creation of the new user.

⇒ Helm

1. Create a new MSSQL database.
2. **(Recommended)** Create a dedicated DBO for your database.

3. In your `my-values.yaml` configuration file, set the value `database.enabled: false` to stop the included SQL pod from being deployed.
4. In the Kubernetes secrets object used for deployment, set a `globalSettings__sqlServer__connectionString=` value with the following information:

Note

The method you use to configure your secrets object may depend on your deployment, for example [AWS deployments](#) and [Azure deployments](#) may use a CSI SecretProviderClass to do so.

- `Data Source=tcp:<SERVERNAME>,1433` where `<SERVERNAME>` is your MSSQL server's name.
- `Initial Catalog=<VAULT>` where `<VAULT>` is your database name.
- `Persist Security Info=False`.
- `User ID=<USER>` where `<USER>` is your DBO user ID.
- `Password=<PASSWORD>` where `<PASSWORD>` is your DBO password.
- `Multiple Active Result Sets=False`.
- `Connect Timeout=30`.
- `Encrypt=True`.
- `Trust Server Certificate=true`. This value can be set to `false` if you require that the Bitwarden server validates your MSSQL server's certificate.

Validate a server certificate

To configure Bitwarden to validate your MSSQL database server's certificate:

⇒ Docker

1. Copy your root CA certificate into `./bwdata/ca-certificates`.
2. Run the `./bitwarden.sh restart` command to apply the certificate to your containers and restart your server.

⇒ Helm

1. In your `my-values.yaml` configuration file, set the value `caCertificate.enabled: true`.
2. Create a ConfigMap object that contains your certificate file. The simplest way to do this would be to add a `preInstall` [RawManifest](#) to your `my-values.yaml` file, as in the following example:

Bash

```
rawManifests:
  preInstall:
    - kind: ConfigMap
      apiVersion: v1
      metadata:
        name: cacert
      data:
        rootca.crt: |
          -----BEGIN CERTIFICATE-----
          ...
          -----END CERTIFICATE-----
  postInstall:
```