# Sleutelconnector implementeren

# Sleutelconnector implementeren

Dit artikel leidt je door de procedure voor het inschakelen en configureren van Key Connector in een bestaande zelfgehoste omgeving. Lees **voordat u verder gaat** het artikel over Key Connector grondig door, zodat u volledig begrijpt wat Key Connector is, hoe het werkt en wat de gevolgen van implementatie zijn.

Bitwarden ondersteunt de implementatie van één Key Connector voor gebruik door één organisatie voor een zelf gehoste instantie.

## Vereisten

> ⚠ **Warning**
>
> Management of cryptographic keys is incredibly sensitive and is **only recommended for enterprises with a team and infrastructure** that can securely support deploying and managing a key server.

Om Key Connector te gebruiken moet je:

- Een Enterprise-organisatie hebben.
- Een zelf gehoste Bitwarden-server hebben.
- Een actieve SSO-implementatie hebben.
- Activeer de beleidsregels Eén organisatie en Eén aanmelding vereisen.

Als jouw organisatie aan deze eisen voldoet of kan voldoen, inclusief een team en infrastructuur die het beheer van een keyserver kan ondersteunen, neem dan contact met ons op en dan activeren we Key Connector.

## Key Connector instellen en implementeren

**Zodra je contact met ons hebt opgenomen over Key Connector**, nemen we contact met je op om een Key Connector-gesprek te starten. De stappen die volgen in dit artikel moeten worden uitgevoerd in samenwerking met de Bitwarden customer success & implementatie specialisten.

### Nieuw licentiebestand verkrijgen

Zodra u contact met ons heeft opgenomen over Key Connector, zal een lid van het customer success & implementatie team een Key Connector licentiebestand voor uw organisatie genereren. Wanneer uw Bitwarden-medewerker aangeeft dat het klaar is, voert u de volgende stappen uit om de nieuwe licentie te verkrijgen:

1. Open de Bitwarden cloud-webapp en navigeer naar het scherm **Facturering** → **Abonnement** van uw organisatie in de beheerconsole.
2. Scroll naar beneden en selecteer de knop **Licentie downloaden**.
3. Voer wanneer daarom wordt gevraagd de installatie-ID in die is gebruikt om uw zelf gehoste server te installeren en selecteer **Submit**. Als u uw installatie-ID niet direct weet, kunt u deze ophalen uit `./bwdata/env/global.override.env`.

Je hebt je licentiebestand niet meteen nodig, maar je zult het in een latere stap moeten uploaden naar je zelf gehoste server.

### Sleutelconnector initialiseren

Om uw Bitwarden-server voor te bereiden op Key Connector:

1. Sla een back-up op van ten minste `.bwdata/mssql`. Zodra Key Connector in gebruik is, is het aan te raden om toegang te hebben tot een pre-Key Connector back-up image in geval van een probleem.

   > **Note**
   >
   > Als je een externe MSSQL-database gebruikt, maak dan een back-up van je database op een manier die past bij jouw implementatie.

2. Werk uw zelf gehoste Bitwarden-installatie bij om de laatste wijzigingen op te halen:

   ```bash
   ./bitwarden.sh update
   ```

3. Bewerk het bestand `.bwdata/config.yml` en schakel Key Connector in door `enable_key_connector` op `true` te zetten.

   ```bash
   nano bwdata/config.yml
   ```

4. Herbouw uw zelf gehoste Bitwarden-installatie:

   ```bash
   ./bitwarden.sh rebuild
   ```

5. Update uw zelf gehoste Bitwarden-installatie opnieuw om de wijzigingen toe te passen:

   ```bash
   ./bitwarden.sh update
   ```

## Sleutelconnector configureren

Om Key Connector te configureren:

1. Bewerk het `.bwdata/env/key-connector.override.env` bestand dat is gedownload met de `./bitwarden.sh update`.

   ```bash
   nano bwdata/env/key-connector.override.env
   ```

> ⚠ **Warning**
>
> This file will be pre-populated with default values that will spin up a functional local Key Connector setup, however the **default values are not recommended for production environments**.

2. In `key-connector.override.env` moet je waarden opgeven voor het volgende:
   - Eindpunten: Met welke Bitwarden eindpunten Key Connector kan communiceren.
   - Database: Waar Key Connector gebruikerssleutels opslaat en ophaalt.
   - RSA sleutelpaar: Hoe Key Connector toegang krijgt tot een RSA sleutelpaar om gebruikerssleutels in rust te beschermen.

### Eindpunten

De geautomatiseerde installatie zal de eindpuntwaarden invullen op basis van uw installatieconfiguratie, maar het wordt aanbevolen om te controleren of de volgende waarden in `key-connector.override.env` juist zijn voor uw installatie:

```bash
keyConnectorSettings__webVaultUri=https://your.bitwarden.domain.com
keyConnectorSettings__identityServerUri=http://identity:5000
```

### Database

Key Connector moet toegang krijgen tot een database waarin gecodeerde gebruikerssleutels voor de leden van je organisatie worden opgeslagen. Maak een beveiligde database aan om versleutelde gebruikerssleutels in op te slaan en vervang de standaardwaarden van `keyConnectorSettings__database__` in `key-connector.override.env` door de waarden in de kolom **Required Values** voor de gekozen database:

> ⚠ **Warning**
>
> Migration from one database to another is **not supported** at this time. Regardless of which provider you choose, **implement a frequent automated backup schedule** for the database.

| Database | Vereiste waarden |
|---|---|
| Lokale JSON(**standaard**) | **Niet aanbevolen buiten testen.**<br>`keyConnectorSettings__database__provider=json`<br>`keyConnectorSettings__database__jsonFilePath={File_Path}` |

| Database | Vereiste waarden |
| --- | --- |
| Microsoft SQL Server | `keyConnectorSettings__database__provider=sqlserver`<br>`keyConnectorSettings__database__sqlServerConnectionString={Connection_String}`<br>Leren hoe je MSSQL-connectiestrings opmaakt |
| PostgreSQL | `keyConnectorSettings__database__provider=postgresql`<br>`keyConnectorSettings__database__postgreSqlConnectionString={Connection_String}`<br>Leer hoe je PostgreSQL-connectiestrings opmaakt |
| MySQL/MariaDB | `keyConnectorSettings__database__provider=mysql`<br>`keyConnectorSettings__database__mySqlConnectionString={Connection_String}`<br>Leer hoe u MySQL-connectiestrings formatteert |
| MongoDB | `keyConnectorSettings__database__provider=mongo`<br>`keyConnectorSettings__database__mongoConnectionString={Connection_String}`<br>`keyConnectorSettings__database__mongoDatabaseName={DatabaseName}`<br>Leren hoe je MongoDB-connectiestrings opmaakt |

### RSA sleutelpaar

Key Connector gebruikt een RSA sleutelpaar om gebruikerssleutels in rust te beschermen. Maak een sleutelpaar aan en vervang de standaardwaarden voor `keyConnectorSettings__rsaKey__` en `keyConnectorSettings__certificate__` in `key-connector.override.env` door de waarden die vereist zijn voor de door u gekozen implementatie.

> **Tip**
> The RSA key pair must be **at a minimum** 2048 bits in length.

Over het algemeen kunt u Key Connector toegang geven tot een X509 **Certificaat** dat het sleutelpaar bevat of Key Connector rechtstreeks toegang geven tot het **sleutelpaar**:

## ⇒Certificaat

Om een X509-certificaat te gebruiken dat een RSA-sleutelpaar bevat, moet u de vereiste waarden opgeven, afhankelijk van de locatie waar uw certificaat is opgeslagen (zie **Bestandssysteem**, **OS Certificate Store**, enzovoort):

**Tip**

The certificate **must** be made available as a PKCS12 `.pfx` file, for example:

```
Bash

openssl req -x509 -newkey rsa:4096 -sha256 -nodes -keyout bwkc.key -out bwkc.crt -subj "/CN=Bi
twarden Key Connector" -days 36500

openssl pkcs12 -export -out ./bwkc.pfx -inkey bwkc.key -in bwkc.crt -passout pass:{Password}
```

In all certificate implementations, you'll need the `CN` value shown in this example.

### Bestandssysteem (standaard)

Als het certificaat is opgeslagen op het bestandssysteem van de machine waarop Key Connector draait, geef dan de volgende waarden op:

**Note**

By default, Key Connector will be configured to create a `.pfx` file located at `etc/bitwarden/key-connector/bwkc.pfx` with a generated password. **It is not recommended** for enterprise implementations to use these defaults.

```
Bash

keyConnectorSettings__rsaKey__provider=certificate
keyConnectorSettings__certificate__provider=filesystem
keyConnectorSettings__certificate__filesystemPath={Certificate_Path}
keyConnectorSettings__certificate__filesystemPassword={Certificate_Password}
```

### Azure Blob Storage

Als het certificaat wordt geüpload naar Azure Blob Storage, geef dan de volgende waarden op:

```
Bash

keyConnectorSettings__rsaKey__provider=certificate
keyConnectorSettings__certificate__provider=azurestorage
keyConnectorSettings__certificate__azureStorageConnectionString={Connection_String}
keyConnectorSettings__certificate__azureStorageContainer={Container_Name}
keyConnectorSettings__certificate__azureStorageFileName={File_Name}
keyConnectorSettings__certificate__azureStorageFilePassword={File_Password}
```

Stel `azureStorageConnectionString` in op een **verbindingsstring** die je kunt genereren in je Azure portaal vanaf de **Gedeelde toegang handtekening** (SAS) pagina van je opslagaccount. De SAS moet beschikken over:

- Toegestane services: Blob en Bestand
- Toegestane brontypes: Service, Container en Object
- Toegestane rechten: Lezen, schrijven en lijst
- Toegestane blob index rechten: Lezen/schrijven en filteren

### Azure-sleutelkluis

Als het certificaat is opgeslagen in Azure Key Vault, specificeer dan de volgende waarden:

> **Note**
>
> To use Azure Key Vault to store your `.pfx` certificate, you'll need to create an Active Directory **App Registration**. This App Registration must:
>
> - Give delegated API permissions to access Azure Key Vault
> - Have a client secret generated to allow access by Key Connector

```bash
keyConnectorSettings__certificate__provider=azurekv
keyConnectorSettings__certificate__azureKeyvaultUri={Vault_URI}
keyConnectorSettings__certificate__azureKeyvaultCertificateName={Certificate_Name}
keyConnectorSettings__certificate__azureKeyvaultAdTenantId={ActiveDirectory_TenantId}
keyConnectorSettings__certificate__azureKeyvaultAdAppId={AppRegistration_ApplicationId}
keyConnectorSettings__certificate__azureKeyvaultAdSecret={AppRegistration_ClientSecretValue}
```

### Hashicorp Kluis

Als het certificaat is opgeslagen in Hashicorp Vault, geef dan de volgende waarden op:

> **Note**
>
> Key Connector integrates with the Hashicorp Vault KV2 Storage Engine. As per the top of this tab, the certificate file should be in PKCS12 format and stored base64-encoded as the value to a named key in your Vault. If following a Vault tutorial for the KV2 Storage Engine, the key name may be `file` unless otherwise specified.

```Bash
keyConnectorSettings__rsaKey__provider=certificate
keyConnectorSettings__certificate__provider=vault
keyConnectorSettings__certificate__vaultServerUri={Server_URI}
keyConnectorSettings__certificate__vaultToken={Token}
keyConnectorSettings__certificate__vaultSecretMountPoint={Secret_MountPoint}
keyConnectorSettings__certificate__vaultSecretPath={Secret_Path}
keyConnectorSettings__certificate__vaultSecretDataKey={Secret_DataKey}
keyConnectorSettings__certificate__vaultSecretFilePassword={Secret_FilePassword}
```

## ⇒Sleutelpaar

Als je een cloudprovider of fysiek apparaat wilt gebruiken om een RSA 2048 sleutelpaar op te slaan, moet je de vereiste waarden opgeven, afhankelijk van de gekozen implementatie (zie **Azure Key Vault**, **Google Cloud Key Management**, enzovoort):

### Azure-sleutelkluis

Als je Azure Key Vault gebruikt om een RSA 2048 sleutelpaar op te slaan, geef dan de volgende waarden op:

> **Note**
>
> To use Azure Key Vault to store your RSA 2048 key, you'll need to create an Active Directory **App Registration**. This App Registration must:
>
> - Give delegated API permissions to access Azure Key Vault
> - Have a client secret generated to allow access by Key Connector

```Bash
keyConnectorSettings__rsaKey__provider=azurekv
keyConnectorSettings__rsaKey__azureKeyvaultUri={Vault_URI}
keyConnectorSettings__rsaKey__azureKeyvaultKeyName={Key_Name}
keyConnectorSettings__rsaKey__azureKeyvaultAdTenantId={ActiveDirectory_TenantId}
keyConnectorSettings__rsaKey__azureKeyvaultAdAppId={AppRegistration_ApplicationId}
keyConnectorSettings__rsaKey__azureKeyvaultAdSecret={AppRegistration_ClientSecretValue}
```

Leer hoe je Azure Key Vault gebruikt om een sleutelpaar aan te maken

### Google Cloud-sleutelbeheer

Als je Google Cloud Key Management gebruikt om een RSA 2048 sleutelpaar op te slaan, geef dan de volgende waarden op:

```Bash
keyConnectorSettings__rsaKey__provider=gcpkms
keyConnectorSettings__rsaKey__googleCloudProjectId={Project_Id}
keyConnectorSettings__rsaKey__googleCloudLocationId={Location_Id}
keyConnectorSettings__rsaKey__googleCloudKeyringId={Keyring_Id}
keyConnectorSettings__rsaKey__googleCloudKeyId={Key_Id}
keyConnectorSettings__rsaKey__googleCloudKeyVersionId={KeyVersionId}
```

Leer hoe u Google Cloud Key Management Service kunt gebruiken om sleutelringen en asymmetrische sleutels te maken

### AWS sleutelbeheerservice

Als je AWS Key Management Service (KMS) gebruikt om een RSA 2048 sleutelpaar op te slaan, geef dan de volgende waarden op:

```Bash
keyConnectorSettings__rsaKey__provider=awskms
keyConnectorSettings__rsaKey__awsAccessKeyId={AccessKey_Id}
keyConnectorSettings__rsaKey__awsAccessKeySecret={AccessKey_Secret}
keyConnectorSettings__rsaKey__awsRegion={Region_Name}
keyConnectorSettings__rsaKey__awsKeyId={Key_Id}
```

Leer hoe je AWS KMS gebruikt om asymmetrische sleutels te maken

### PKCS11 Fysieke HSM

Als u een fysiek HSM-apparaat gebruikt met de PKCS11-provider, geef dan de volgende waarden op:

```bash
keyConnectorSettings__rsaKey__provider=pkcs11
keyConnectorSettings__rsaKey__pkcs11Provider={Provider}
keyConnectorSettings__rsaKey__pkcs11SlotTokenSerialNumber={Token_SerialNumber}
keyConnectorSettings__rsaKey__pkcs11LoginUserType={Login_UserType}
keyConnectorSettings__rsaKey__pkcs11LoginPin={Login_PIN}

ONE OF THE FOLLOWING TWO:
keyConnectorSettings__rsaKey__pkcs11PrivateKeyLabel={PrivateKeyLabel}
keyConnectorSettings__rsaKey__pkcs11PrivateKeyId={PrivateKeyId}

OPTIONALLY:
keyConnectorSettings__rsaKey__pkcsLibraryPath={path/to/library/file}
```

Waar:

- `{Provider}` kan `yubihsm` of `opensc` zijn
- `{Login_UserType}` kan `gebruiker-`, `dus-` of `contextspecifiek` zijn

> **Note**
> If you are using the PKCS11 provider to store your private key on an HSM device, the associated public key must be made available and configured as a certificate using any of the options found in the **Certificates** tab.

## Sleutelconnector activeren

Nu Key Connector volledig is geconfigureerd en je een Key Connector-licentie hebt, kun je de volgende stappen uitvoeren:

1. Start uw zelf gehoste Bitwarden-installatie opnieuw op om de configuratiewijzigingen toe te passen:

   ```bash
   ./bitwarden.sh restart
   ```

2. Log in op uw zelf gehoste Bitwarden als **eigenaar van** de organisatie en navigeer naar het scherm **Facturering** → **Abonnement** in de beheerconsole.

3. Selecteer de knop **Licentie bijwerken** en upload de Key Connector-licentie die je in een eerdere stap hebt opgehaald.

4. Als je dat nog niet hebt gedaan, ga dan naar het scherm **Instellingen** → **Beleid** en schakel de beleidsregels Eenmalige organisatie en Eenmalige aanmeldingsverificatie vereisen in. **Beide zijn vereist om Key Connector te gebruiken**.

5. Navigeer naar het scherm **Instellingen** → **Eenmalige aanmelding**.

> **Tip**
>
> The next few steps assume that you already have an active login with SSO implementation using SAML 2.0 or OIDC. **If you don't**, please implement and test login with SSO before proceeding.

6. Selecteer **Key Connector** in de sectie **Ontcijferingsopties voor leden**.

7. Voer in de **Key Connector URL-invoer** het adres in waarop Key Connector wordt uitgevoerd (standaard `https://your.domain/key-connector`) en selecteer de knop **Test** om ervoor te zorgen dat je Key Connector kunt bereiken.

8. Scroll naar de onderkant van het scherm en selecteer **Opslaan**.