bitwarden | Help Center Article

SELF-HOSTING > INSTALL & DEPLOY GUIDES >

Linux Offline Deployment

View in the help center:
https://bitwarden.com/help/install-and-deploy-offline/

Linux Offline Deployment

This article will walk you through the procedure to install and deploy Bitwarden to your own server in an **offline or air-gapped environment**. Please review Bitwarden software release support documentation.

> ⚠ Warning
>
> **Manual installations should be conducted by advanced users only.** Only proceed if you are very familiar with Docker technologies and desire more control over your Bitwarden installation.
>
> Manual installations lack the ability to automatically update certain dependencies of the Bitwarden installation. As you upgrade from one version of Bitwarden to the next you will be responsible for changes to required environment variables, changes to nginx `default.conf`, changes to `docker-compose.yml`, and so on.
>
> We will try to highlight these in the release notes on GitHub. You can also monitor changes to the dependency templates used by the Bitwarden installation script on GitHub.

Requirements

	Minimum	Recommended
Processor	x64, 1.4GHz	x64, 2GHz dual core
Memory	2GB RAM	4GB RAM
Storage	12GB	25GB
Docker Version	Engine 26+ and Compose[a]	Engine 26+ and Compose[a]

[a] – Docker Compose is automatically installed as a plugin when you download Docker Engine. Download Docker Engine for Linux.

Additionally, ensure the following requirements are met:

- Using a machine with internet access, you have downloaded the latest `docker-stub-US.zip` or `docker-stub-EU.zip` file from the Bitwarden Server repository's releases page and transferred this file to your server.
- An offline SMTP Server is setup and active in your environment.

The server your Bitwarden deployment runs on will not be required to allow outbound traffic to any addresses outside of your network, however client applications must be configured to access the server's fully qualified domain name (FQDN) on, by default, ports `80` and `443`. You may opt to choose different ports during installation, but whichever ports you choose these must be opened for client access.

Installation procedure

Configure your domain

By default, Bitwarden will be served through ports 80 (`http`) and 443 (`https`) on the host machine. Open these ports so that Bitwarden can be accessed from within and/or outside of the network. You may opt to choose different ports during installation.

We recommend configuring a domain name with DNS records that point to your host machine (for example, `bitwarden.example.com`), especially if you are serving Bitwarden over the internet.

Create Bitwarden local user & directory

We recommend configuring your server with a dedicated bitwarden service account from which to install and run Bitwarden. Doing so will isolate your Bitwarden instance from other applications running on your server.

These steps are Bitwarden-recommended best practices, but are not required. For more information, see Docker's post-installation steps for Linux documentation.

1. Create a bitwarden user:

```Bash
sudo adduser bitwarden
```

2. Set a password for the bitwarden user:

```Bash
sudo passwd bitwarden
```

3. Create a docker group (if it doesn't already exist):

```Bash
sudo groupadd docker
```

4. Add the bitwarden user to the docker group:

```Bash
sudo usermod -aG docker bitwarden
```

5. Create a bitwarden directory:

```Bash
sudo mkdir /opt/bitwarden
```

6. Set permissions for the `/opt/bitwarden` directory:

```
Bash

sudo chmod -R 700 /opt/bitwarden
```

7. Set the bitwarden user ownership of the `/opt/bitwarden` directory:

```
Bash

sudo chown -R bitwarden:bitwarden /opt/bitwarden
```

Configure your machine

> ⚠ **Warning**
>
> If you have created a Bitwarden user & directory, complete the following as the `bitwarden` user from the `/opt/bitwarden` directory. **Do not install Bitwarden as root**, as you will encounter issues during installation.

To configure your machine with the assets required for your Bitwarden server:

1. Create a new directory named `bwdata` and extract `docker-stub-US.zip` (or `docker-stub-EU.zip`) to it, for example:

```
Bash

unzip docker-stub-US.zip -d bwdata
```

   Once unzipped, the `bwdata` directory will match what the `docker-compose.yml` file's volume mapping expects. You may, if you wish, change the location of these mappings on the host machine.

2. In `./bwdata/env/global.override.env`, edit the following environment variables:

   - `globalSettings__baseServiceUri__vault=`: Enter the domain of your Bitwarden instance.
   - `globalSettings__sqlServer__ConnectionString=`: Replace the `RANDOM_DATABASE_PASSWORD` with a secure password for use in a later step.
   - `globalSettings__identityServer__certificatePassword`: Set a secure certificate password for use in a later step.
   - `globalSettings__internalIdentityKey=`: Replace `RANDOM_IDENTITY_KEY` with a random alphanumeric string.
   - `globalSettings__oidcIdentityClientKey=`: Replace `RANDOM_IDENTITY_KEY` with a random alphanumeric string.
   - `globalSettings__duo__aKey=`: Replace `RANDOM_DUO_AKEY` with a random alphanumeric string.
   - `globalSettings__installation__id=`: Enter an installation id retrieved from https://bitwarden.com/host.

- `globalSettings__installation__key=`: Enter an installation key retrieved from https://bitwarden.com/host.
- `globalSettings__pushRelayBaseUri=`: This variable should be blank. See Configure Push Relay for more information.

> **Tip**
>
> At this time, consider also setting values for all `globalSettings__mail__smtp__` variables and for `adminSettings__admins`. Doing so will configure the SMTP mail server used to send invitations to new organization members and provision access to the System Administrator Portal.
>
> Learn more about environment variables.

3. From `./bwdata`, generate a `.pfx` certificate file for the identity container and move it to the mapped volume directory (by default, `./bwdata/identity/`). For example, run the following commands:

```
Bash

openssl req -x509 -newkey rsa:4096 -sha256 -nodes -keyout identity.key -out identity.crt -subj
"/CN=Bitwarden IdentityServer" -days 10950
```

and

```
Bash

openssl pkcs12 -export -out ./identity/identity.pfx -inkey identity.key -in identity.crt -passou
t pass:IDENTITY_CERT_PASSWORD
```

In the above command, replace `IDENTITY_CERT_PASSWORD` with the certificate password created and used in **Step 2**.

4. Create a subdirectory in `./bwdata/ssl` named for your domain, for example:

```
Bash

mkdir ./ssl/bitwarden.example.com
```

5. Provide a trusted SSL certificate and private key in the newly created `./bwdata/ssl/bitwarden.example.com` subdirectory.

> **Note**
>
> This directory is mapped to the NGINX container at `/etc/ssl`. If you can't provide a trusted SSL certificate, front the installation with a proxy that provides an HTTPS endpoint to Bitwarden client applications.

6. In `./bwdata/nginx/default.conf`:
    1. Replace all instances of `bitwarden.example.com` with your domain, including in the `Content-Security-Policy` header.

2. Set the `ssl_certificate` and `ssl_certificate_key` variables to the paths of the certificate and private key provided in **Step 6**.

3. Take one of the following actions, depending on your certificate setup:

   - If using a trusted SSL certificate, set the `ssl_trusted_certificate` variable to the path to your certificate.
   - If using a self-signed certificate, comment out the `ssl_trusted_certificate` variable.

7. In `./bwdata/env/mssql.override.env`, replace `RANDOM_DATABASE_PASSWORD` with the password created in **Step 2**.

8. In `./bwdata/web/app-id.json`, replace `bitwarden.example.com` with your domain.

9. In `./bwdata/env/uid.env`, set the UID and GID of the `bitwarden` users and group you created earlier so the containers run under them, for example:

```
Bash

LOCAL_UID=1001
LOCAL_GID=1001
```

Download & transfer images

To get docker images for use on your offline machine:

1. From an internet-connected machine, download all `bitwarden/xxx:latest` docker images, as listed in the `docker-compose.yml` file in `docker-stub.zip`.

2. Save each image to a `.img` file, for example:

```
Bash

docker image save -o mssql.img bitwarden/mssql:version
```

3. Transfer all `.img` files to your offline machine.

4. On your offline machine, load each `.img` file to create your local docker images, for example:

```
Bash

docker image load -i mssql.img
```

Start your server

Start your Bitwarden server with the following command:

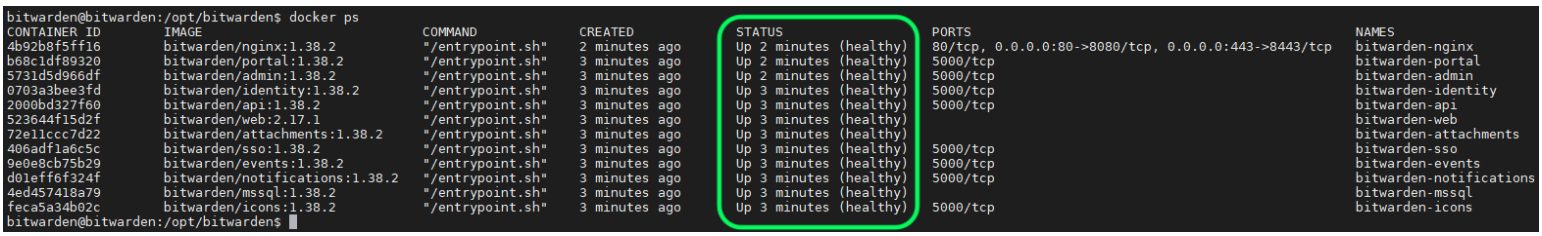
```
Bash

docker compose -f ./docker/docker-compose.yml up -d
```

Verify that all containers are running correctly:

```
Bash

docker ps
```



docker-healthy.png

Congratulations! Bitwarden is now up and running at `https://your.domain.com`. Visit the web vault in your browser to confirm that it's working.

You may now register a new account and log in. Your will need to have configured SMTP environment variables (see environment variables) in order to verify the email for your new account.

Next Steps:

- If you are planning to self-host a Bitwarden organization, see self-host an organization to get started.
- For additional information see self hosting FAQs.

Update your server

Updating a self-hosted server that has been installed and deployed manually is different from the standard update procedure. To update your manually-installed server:

1. Download the latest `docker-stub.zip` archive from the releases pages on GitHub.
2. Unzip the new `docker-stub.zip` archive and compare its contents with what's currently in your `bwdata` directory, copying anything new to the pre-existing files in `bwdata`.
   Do not overwrite your pre-existing `bwdata` directory with the contents of the newer `docker-stub.zip` archive, as this would overwrite any custom configuration work you've done.
3. Download the latest container images and transfer them to your offline machine as documented above.
4. Run the following command to restart your server with your updated configuration and the latest containers:

```Bash
docker compose -f ./docker/docker-compose.yml down && docker compose -f ./docker/docker-compose.yml up -d
```