DOCS › SELF-HOST

Networking Requirements

Ansicht im Hilfezentrum https://bitwarden.com/help/networking-requirements/

Networking Requirements

Deploying Bitwarden in a self-hosted environment requires that your network infrastructure supports specific protocols, ports, and communication patterns. This article outlines the networking requirements and compatibility considerations for self-hosted Bitwarden deployments. Some client-side requirements may also apply to organizations using Bitwarden Cloud, these are noted inline with each applicable requirement.

Protocol requirements

The following sections describe strict protocol requirements when deploying a self-hosted Bitwarden server:

Ports

Bitwarden **requires** two open ports for traffic, one for HTTP (by default, `80`) and one for HTTPS (by default, `443`). Bitwarden does not support operating with only one port available, however HTTP and HTTPS ports can be changed from the default during installation:

Deployment	How to configure non-default ports
Docker, Standard (Linux & Windows)	Edit `http_port=` and `https_port=` in `./bwdata/config.yml` and subsequently run the rebuild command.
Docker, Manual (Linux)	Create a copy of `docker-compose.yml` in the same directory and rename it `docker-compose.override.yml`. In the override file, edit the `nginx` port mappings. These changes will be merged at runtime.
Docker, Offline (Linux & Windows)	Create a copy of `docker-compose.yml` in the same directory and rename it `docker-compose.override.yml`. In the override file, edit the `nginx` port mappings. These changes will be merged at runtime.
Helm	Port exposure is controlled by your Kubernetes Ingress Controller rather than by Bitwarden services.
Lite	Set the ports to use at runtime in the `docker run` command or `docker-compose.yml` file.

HTTP Verbs

> **Note**
>
> This requirement also applies to cloud customers.

Bitwarden **does not support** operating in environments where HTTP verbs are restricted or whitelisted. Bitwarden uses multiple HTTP verbs throughout the product. The exact verbs in use depend on the functionality being invoked, and may change to support existing or future features.

WebSocket connections

> **Note**
>
> This requirement also applies to cloud customers.

Bitwarden **does not support** operating in environments where WebSocket connections are blocked. WebSocket (`wss://`) connectivity is required between Bitwarden clients and the self-hosted server.

Intermediary network devices

The following sections describe how intermediary network devices must be configured for a self-hosted deployment:

Reverse proxies

Bitwarden **supports** operating behind a reverse proxy, however using a reverse proxy **requires** that all headers, including `Host:`, be passed unmodified to the Bitwarden `nginx` container.

Forward proxies

Bitwarden **supports** operating behind a forward proxy, however in such an environment one of the following two deployment strategies is highly recommended:

- Follow the instructions for offline deployment on Linux or Windows.
- Follow the instructions for deploying using Docker Compose with a forward proxy.

Whitelist firewalls

Bitwarden **supports** operating behind a whitelist firewall when configured to allow necessary traffic, however in such an environment one of the following two deployment strategies is highly recommended:

- Follow the instructions for offline deployment on Linux or Windows.
- Follow the instructions for deploying using Docker Compose with a forward proxy.

Web application firewalls & intrusion prevention systems

> **Note**
>
> This requirement also applies to cloud customers.

Bitwarden **is compatible** with operation behind a web application firewall (WAF) or intrusion prevention system (IPS), however in such an environment the WAF or IPS should be set to learning (also called "training" or "simulation") mode prior production rollout of the Bitwarden server. This will allow administrators the opportunity to review rulesets while the Bitwarden server is in test, reducing the likelihood that transmission of encrypted payloads, access tokens, or other cryptographic data is blocked when the server is in production and the WAF or IPS is actively blocking.

Other network platforms

The following sections describe how other devices on your network may interact with a self-hosted Bitwarden server:

Endpoint detection, mail scanning, & anti-virus

> **Note**
>
> This requirement also applies to cloud customers.

Endpoint detection & response, mail scanning, and anti-virus platforms has been reported to interfere with the normal operation of Bitwarden containers; specifically, reports include Windows anti-virus platforms flagging Bitwarden containers for being Linux-based, and mail-scanning platforms redacting or reformatting hyperlinks in invitation emails.

Do not disable these solutions unless recommended by Bitwarden support during troubleshooting, but if you experience these issues check your endpoint detection, mail scanning, and anti-virus platforms for false positives related to the normal operation of Bitwarden.

Man-in-the-middle proxies

Bitwarden may be dangerous to operate in environments with man-in-the-middle (MITM) proxies deployed (for example, ZScaler) depending on your organization's configuration. **Logging full data from Bitwarden traffic should be disabled** on this type of network connection in order to preserve critical security technologies, such as transit-layer encryption on Bitwarden traffic.

Environmental factors

The following sections describe host environment configurations that will affect a self-hosted deployment:

Air-gapped environments

Bitwarden **supports** operating in an air-gapped or offline environment, however migration between a standard "online" deployment and offline deployment is not recommended. Use one of the following documents to deploy in an offline environment:

- Linux Offline Deployment Guide
- Windows Offline Deployment Guide

Docker firewall backend (nftables)

Bitwarden **supports** Docker-based deployments running with either the `iptables` backend or the **experimental** `nftables` backend. If you run Docker with the experimental `nftables` backend, the following daemon configuration steps are required on the host before deploying Bitwarden:

1. **Enable IP forwarding**. Docker with nftables does not automatically enable IP forwarding, so enable it manually by adding the following lines to `/etc/sysctl.d/99-docker-forward.conf`:

   ```
   Plain Text

   net.ipv4.ip_forward=1
   net.ipv6.conf.all.forwarding=1
   ```

   Then apply the configuration change using `sudo sysctl --system`.

2. **Configure the Docker daemon**. Create or edit `/etc/docker/daemon.json` to specify the firewall backend:

```
Plain Text
```

```
{
  "firewall-backend": "nftables"
}
```

Then apply the configuration change by restarting Docker with `sudo systemctl restart docker`.

3. **Verify the configuration**. Confirm that your changes were correctly applied using `sudo nft list ruleset`. A table of ip docker-bridges or similar Docker-managed chains should be printed to your console.

   You can also check the daemon logs for any backend initialization errors with `sudo journalctl -u docker`.

Note

Compatibility with firewalld. `firewalld` is the default frontend on many Linux distributions and typically uses `nftables` as its backend. Docker and `firewalld` can conflict when both manage rules. If container networking fails after enabling the `nftables` backend, ensure that the `docker` zone has masquerading enabled as this is needed for container internet access:

```
Plain Text
```

```
sudo firewall-cmd --zone=docker --add-masquerade --permanent
sudo firewall-cmd --reload
```