

# Linode Stacks

- [NGINX Proxy Manager](#)
- [Portainer](#)
- [Vaultwarden Password Management](#)
- [Vaultwarden Password Management \(draft\)](#)
- [HumHub on Linode](#)

# NGINX Proxy Manager

```
version: '3'

networks:
  nginx_proxy_manager:

services:
  app:
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    ports:
      - '80:80'
      - '81:81'
      - '443:443'
    volumes:
      - /home/docker/nginxproxymanager/data:/data
      - /home/docker/nginxproxymanager/letsencrypt:/etc/letsencrypt
    networks:
      nginx_proxy_manager:
```

# Portainer

```
version: '3.3'

services:
  portainer-ce:
    ports:
      - '9000:9000'
      - '8000:8000'
    container_name: portainer

networks:
  - nginx_proxy_manager

restart: always

volumes:
  - '/var/run/docker.sock:/var/run/docker.sock'
  - '/home/docker/portainer:/data'

image: 'portainer/portainer-ce:latest'

networks:
  nginx_proxy_manager:
    external: true
```

# Vaultwarden Password Management

The first thing I self-hosted on my Docker server (after a media server, of course) and made public on a domain was a password manager. At the time, the container was called BitWarden RS, but it has since been renamed to VaultWarden.

The reason I wanted to self-host a password manager is that I believe 2 factor authentication and strong passwords are a great first step in securing our accounts online. I had been using a paid password manager for years and had even convinced others to do the same. Unfortunately, the company that I recommended to everyone switched their pricing structure to only allow for 1 device to be connected to their account on the free tier.

I regularly use 2-4 devices per day and so using a service that only allows for 1 device wasn't going to cut it.

## Prerequisites for VaultWarden in Docker.

First things first, you'll need a Docker server set up. Linode has made that process very simple and you can set one up for just a few bucks a month and can add a private IP address (for free) and backups for just a couple bucks more per month.

Another thing you'll need is a domain name, which you can buy from almost anywhere online for a wide range of prices depending on where you make your purchase. Be sure to point the domains DNS settings to point to Linode. You can find more information about that here:

<https://www.linode.com/docs/guides/dns-manager/>

You'll also want a reverse proxy set up on your Docker Server so that you can do things like route traffic and manage SSLs on your server. I made a video about the process of setting up a Docker server with Portainer and a reverse proxy called Nginx Proxy Manager that you can check out here: INSERT LINK TO THAT VIDEO HERE.

Once you've got your Docker server set up, you can begin the process of setting up your VaultWarden password manager on that server.

There are 2 primary ways you can do this:

1. In the command line via SSH.
2. In Portainer via the Portainer dashboard.

We're going to take a look at how to do this in Portainer so that we can have a user interface to work with.

Head over to <http://your-server-ip-address:9000> and get logged into Portainer with the credentials we setup in our previous post/video.

On the left side of the screen, we're going to click the "Stacks" link and then, on the next page, click the "+ Add stack" button.

This will bring up a page where you'll enter the name of the stack. Below that that you can then copy and paste the following:

```
version: "2"
services:
  vaultwarden:
    image: vaultwarden/server:latest
    container_name: vaultwarden
    networks:
      - nginxproxymanager_default
    volumes:
      - /home/docker/vaultwarden:/data/
    ports:
      - 90:80
    restart: unless-stopped

networks:
  nginxproxymanager_default:
    external: true
```

This Docker compose file / stack tells the system to download the latest vaultwarden/server image to the Docker server, name the container "vaultwarden", attach it to the "nginxproxymanager\_default" network, mount it to the directory "home/docker/vaultwarden", and make it available on the docker server's port 90.

You can change the path of where you'd like to store the vaultwarden data if you'd like. Just make sure that the folder has the correct permissions to be written to.

Once you're happy with the settings here, we can deploy the container by clicking the button that says "Deploy the stack".

This will start the process of downloading and deploying VaultWarden on your server. You'll know that the container has been deployed once the page reloads and you see all your running containers.

<input type="checkbox"/>	Name	State <small>Filter</small>	Quick Actions	Stack	Image	Created	IP Address	Published Ports	Ownership
<input type="checkbox"/>	vaultwarden	healthy		vaultwarden	vaultwarden/server:latest	2022-03-04 18:11:44	172.18.0.4	90.80	administrators
<input type="checkbox"/>	portainer	running		portainer	portainer/portainer-ce:latest	2022-02-15 13:34:58	172.18.0.3	8000:8000  9000:8000	administrators
<input type="checkbox"/>	nginxproxymanager_app_1	running		nginxproxymanager	jc21/nginx-proxy-manager:latest	2022-02-15 13:20:08	172.18.0.2	443:443  80:80  81:81	administrators

Now you can go to <http://your-server-ip-address:90> and you should be able to see the login screen for VaultWarden.

STOP.

In order for you to actually be able to use VaultWarden, it needs a domain name and SSL.

## Setting up a domain and SSL

Head back over to your Linode dashboard and go to "Domains". Then find the domain that you added to your account. Click it and then look for the CNAME section of the domain management.

Add a CNAME to your domain by entering a hostname entry for what you'd like your subdomain to be. In the example video, I entered "pw" (without the quotes). Below that, I entered the @ symbol in the "Alias to" box and then clicked "Save".

Now that you have that done, we can head over to Nginx Proxy Manager and setup our SSL and domain.

Now you can go to <http://your-server-ip-address:81> and you should be able to see the login screen for Nginx Proxy Manager.

The default credentials for Nginx Proxy Manager are:

Email: admin@example.com  
Password: changeme

Enter those cred

# Vaultwarden Password Management (draft)

The first thing I self-hosted on my Docker server (after a media server, of course) and made public on a domain was a password manager. At the time, the container was called BitWarden RS, but it has since been renamed to VaultWarden.

The reason I wanted to self-host a password manager is that I believe 2 factor authentication and strong passwords are a great first step in securing our accounts online. I had been using a paid password manager for years and had even convinced others to do the same. Unfortunately, the company that I recommended to everyone switched their pricing structure to only allow for 1 device to be connected to their account on the free tier.

I regularly use 2-4 devices per day and so using a service that only allows for 1 device wasn't going to cut it.

## Prerequisites for VaultWarden in Docker.

First things first, you'll need a Docker server set up. Linode has made that process very simple and you can set one up for just a few bucks a month and can add a private IP address (for free) and backups for just a couple bucks more per month.

Another thing you'll need is a domain name, which you can buy from almost anywhere online for a wide range of prices depending on where you make your purchase. Be sure to point the domains DNS settings to point to Linode. You can find more information about that here:

<https://www.linode.com/docs/guides/dns-manager/> (or link to the previous article)

You'll also want a reverse proxy set up on your Docker Server so that you can do things like route traffic and manage SSLs on your server. I made a video about the process of setting up a Docker server with Portainer and a reverse proxy called Nginx Proxy Manager that you can check out here: INSERT LINK TO THAT VIDEO/POST HERE.

Once you've got your Docker server set up, you can begin the process of setting up your VaultWarden password manager on that server.

There are 2 primary ways you can do this:

1. In the command line via SSH.
2. In Portainer via the Portainer dashboard.

We're going to take a look at how to do this in Portainer so that we can have a user interface to work with.

Head over to <http://your-server-ip-address:9000> and get logged into Portainer with the credentials we setup in our previous post/video.

On the left side of the screen, we're going to click the "Stacks" link and then, on the next page, click the "+ Add stack" button.

This will bring up a page where you'll enter the name of the stack. Below that that you can then copy and paste the following:

```
version: "2"
services:
  vaultwarden:
    image: vaultwarden/server:latest
    container_name: vaultwarden
    networks:
      - nginxproxymanager_default
    volumes:
      - /home/docker/vaultwarden:/data/
    ports:
      - 90:80
    restart: unless-stopped

networks:
  nginxproxymanager_default:
    external: true
```

This Docker compose file / stack tells the system to download the latest vaultwarden/server image to the Docker server, name the container "vaultwarden", attach it to the "nginxproxymanager\_default" network, mount it to the directory "home/docker/vaultwarden", and make it available on the docker server's port 90.

You can change the path of where you'd like to store the vaultwarden data if you'd like. Just make sure that the folder has the correct permissions to be written to.

Once you're happy with the settings here, we can deploy the container by clicking the button that says "Deploy the stack".

This will start the process of downloading and deploying VaultWarden on your server. You'll know that the container has been deployed once the page reloads and you see all your running containers.

[image-1646867721920.png](#)



Now you can go to <http://your-server-ip-address:90> and you should be able to see the login screen for VaultWarden.

STOP.

In order for you to actually be able to use VaultWarden, it needs a domain name and SSL.

## Setting up a domain and SSL

Head back over to your Linode dashboard and go to "Domains". Then find the domain that you added to your account. Click it and then look for the CNAME section of the domain management.

Add a CNAME to your domain by entering a hostname entry for what you'd like your subdomain to be. In the example video, I entered "pw" (without the quotes). Below that, I entered the @ symbol in the "Alias to" box and then clicked "Save".

Now that you have that done, we can head over to Nginx Proxy Manager and setup our SSL and domain.

Now you can go to <http://your-server-ip-address:81> and you should be able to see the login screen for Nginx Proxy Manager.



Log into your account.

# HumHub on Linode

```
version: '3'

networks:
  nginx_proxy_manager:

services:
  nginxproxymanager:
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    container_name: nginx-proxy-manager
    ports:
      - '80:80'
      - '81:81'
      - '443:443'
    volumes:
      - /home/docker/nginxproxymanager/data:/data
      - /home/docker/nginxproxymanager/letsencrypt:/etc/letsencrypt
    networks:
      nginx_proxy_manager:

  portainer-ce:
    ports:
      - '9000:9000'
      - '8000:8000'
    container_name: portainer
    restart: always
    volumes:
      - '/var/run/docker.sock:/var/run/docker.sock'
      - 'portainer_data:/data'
    image: 'portainer/portainer-ce:latest'
    networks:
      nginx_proxy_manager:

  db:
    image: mariadb:10.2
```

```
container_name: humhubdb
environment:
  MYSQL_ROOT_PASSWORD: root
  MYSQL_DATABASE: humhub
  MYSQL_USER: humhub
  MYSQL_PASSWORD: humhub
volumes:
  - humhubdb3:/var/lib/mysql
networks:
  nginx_proxy_manager:
```

humhub:

```
image: mriedmann/humhub:latest
container_name: humhub
links:
  - db:db
ports:
  - 8080:80
volumes:
  - /home/docker/humhub/config:/var/www/localhost/htdocs/protected/config
  - /home/docker/humhub/uploads:/var/www/localhost/htdocs/uploads
  - /home/docker/humhub/modules:/var/www/localhost/htdocs/protected/modules
environment:
  HUMHUB_DB_USER: humhub
  HUMHUB_DB_PASSWORD: humhub
networks:
  nginx_proxy_manager:
```

volumes:

```
humhubdb3:
portainer_data:
```

To avoid issues during the setup process, you'll want to create the following file structure:

- /home/docker/humhub
- /home/docker/humhub/config
- /home/docker/humhub/modules
- /home/docker/humhub/uploads
- /home/docker/humhub/uploads/profile\_image

I placed my docker-compose.yml file in /home/docker and then ran `docker-compose up -d` to spin up the containers.