

**STUDY GUIDE FOR MODULE NO. LAB 10****Password Manager****MODULE OVERVIEW**

The Password Managers module is crafted to educate individuals on the significance of password managers and offers a thorough grasp of the effective utilization of these tools. A password manager like BitWarden is a software application or a cloud-based service designed to store, generate, and manage passwords securely. It offers a safe and quick way to organize and retrieve login credentials for various online services, including email, social networking, banking, and others.

By aiding users in creating strong, unique passwords for each account, automatically filling in login details, and bolstering overall online security, password managers emerge as indispensable tools for safeguarding and managing online accounts. This module will give learners the knowledge and skills to elevate their online safety.

At its core, a password manager's primary purpose is to streamline password management, heighten security, and furnish users with the means to uphold robust, distinct passwords for all their online accounts. In the context of this module, BitWarden is employed as the password manager to proficiently store and generate secure passwords. The module explores advanced password manager features, including two-factor authentication and secure note storage, while emphasizing the importance of routinely auditing and updating stored passwords.

**MODULE LEARNING OUTCOMES**

After finishing this Password Managers module, participants should achieve the following:

- Acquire a fundamental comprehension of password managers as a cybersecurity tool.
- Comprehend the sequential procedures involved in implementing password managers.
- Develop a foundational understanding of the primary features and capabilities of password managers.
- Investigate the realm of portable cybersecurity evaluation tools.
- Demonstrate competence in establishing and configuring a secure environment within a controlled setting.
- Cultivate the expertise to monitor password manager activities and discern potential risks.





## LEARNING CONTENT

Name: Cerujano Erman Ace M.Due date: April 18, 2024

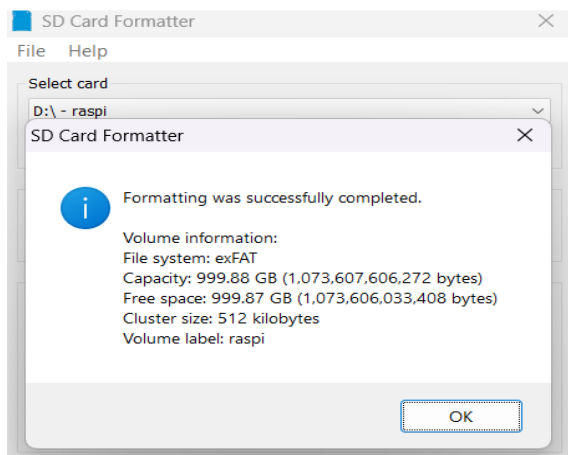
This lab will guide you in utilizing the equipment.

Acquire the necessary hardware and software components essential for configuring the password manager.

- Raspberry Pi (with compatible model and hardware)
- MicroSD card (4GB or larger recommended)
- MicroSD card reader
- Computer with an SD card reader
- Power supply
- Pocket Wi-Fi with Internet

The steps are as follows:

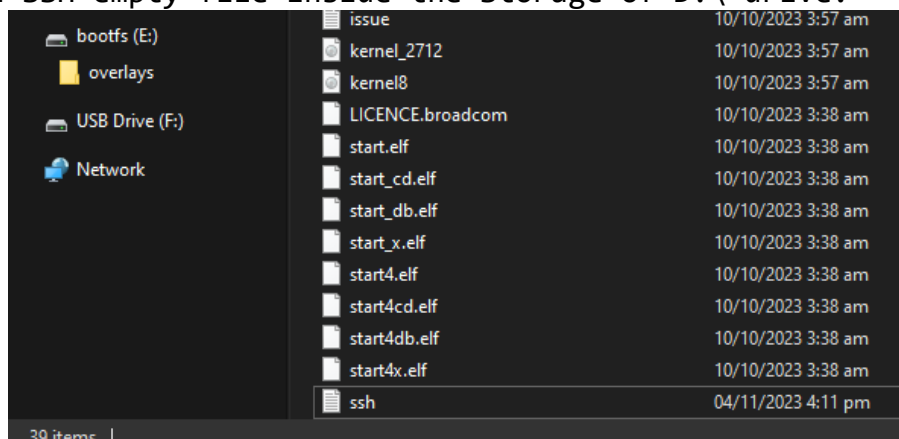
**Step 1.** Format an SD card with an SD card formatter.



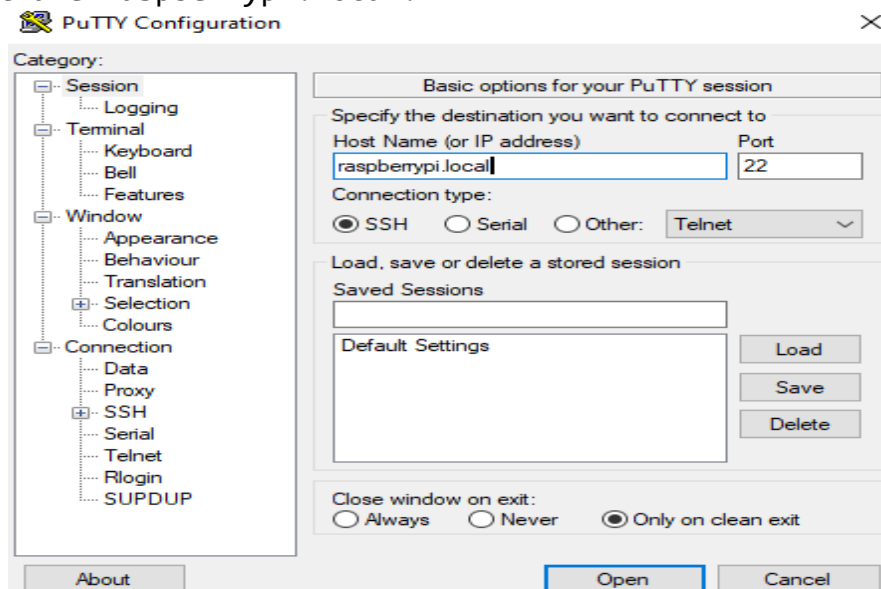
**Step 2.** On the Raspberry Pi Imager, install the Raspbian operating system.



**Step 3.** Add SSH empty file inside the Storage of D:\ drive.

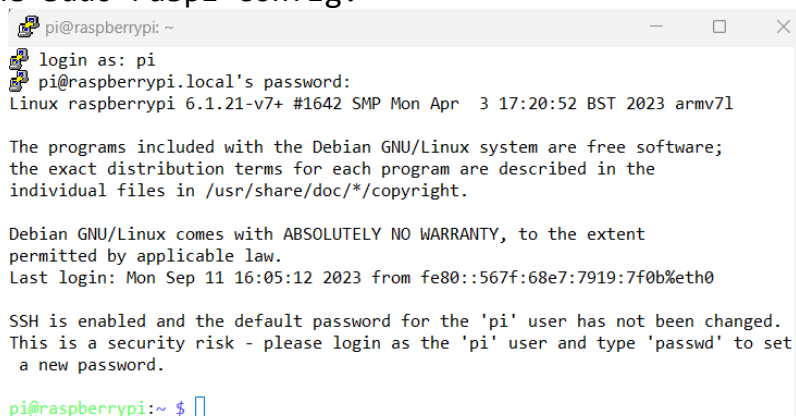


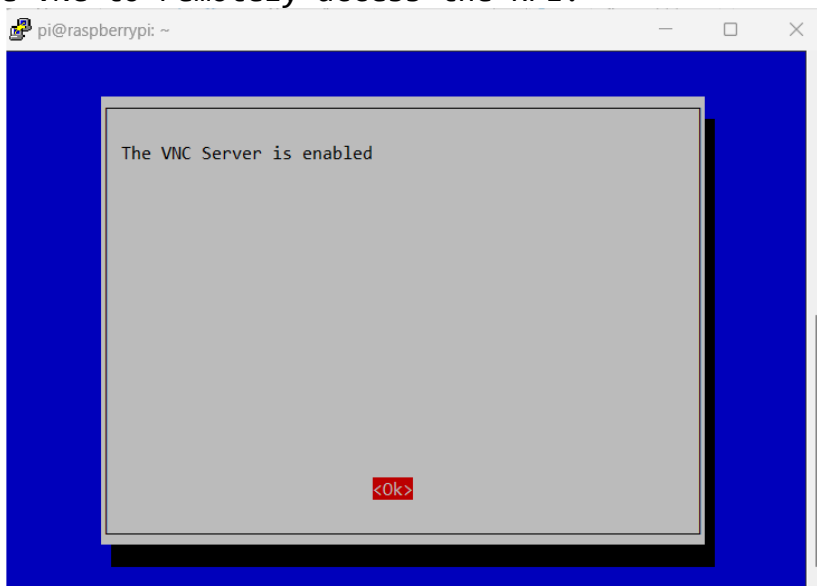
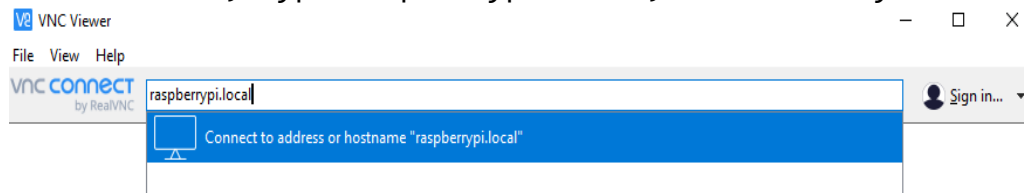
**Step 4.** Log in to puTTY using the credentials we specified earlier: the default hostname raspberrypi.local.



**Step 5.** Click accept and log in with your credentials.

**Step 6.** Use the sudo raspi-config.



**Step 7. Enable VNC to remotely access the RPI.****Step 8. Reboot now.****Step 9. Launch event, type raspberrypi.local, then enter your credentials.****Step 10. Upgrade and update.**

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~$ sudo apt update && sudo apt upgrade -y
Hit:1 https://download.docker.com/linux/raspbian bullseye InRelease
Get:2 http://raspbian.raspberrypi.org/raspbian bullseye InRelease [15.0 kB]
Hit:3 http://archive.raspberrypi.org/debian bullseye InRelease
Fetched 15.0 kB in 11s (1,370 B/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
92 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra libfuse2
Use 'sudo apt autoremove' to remove them.
The following packages will be upgraded:
  base-files bind9-host bind9-libs chromium-codecs-ffmpeg-extra cups-browsed
  cups-filters cups-filters-core-drivers ffmpeg file firmware-atheros
  firmware-brcm80211 firmware-libertas firmware-misc-nonfree firmware-realtek
  ghostscript gstreamer1.0-alsa gstreamer1.0-plugins-bad gstreamer1.0-plugins-base
  gstreamer1.0-plugins-good gstreamer1.0-x libaom0 libavcodec58 libavdevice58
  libavfilter7 libavformat58 libavresample4 libavutil56 libcamera-apps
  libcamera-tools libcamera0 libcupsfilters1 libfontembed1 libgs9 libgs9-common
  libgstreamer-gli1.0-0 libgstreamer-plugins-bad1.0-0 libgstreamer-plugins-base1.0-0

```

**Step 11.** Preparing the Raspberry Pi for BitWarden involves installing Docker since Bitwarden operates within a Docker container. A detailed guide is available to assist you in configuring Docker on the Raspberry Pi.

**Step 12:** Now that the Raspberry Pi is fully updated, install Docker. Fortunately, Docker simplifies this procedure by offering a bash script that expeditiously installs all the necessary components. Execute the following command to download and run the official Docker setup script:

- `curl -sSL https://get.docker.com | sh`

```

pi@raspberrypi:~$ curl -sSL https://get.docker.com | sh
# Executing docker install script, commit: e5543d473431b782227f8908005543bb4389b
8de
Warning: the "docker" command appears to already exist on this system.

If you already have Docker installed, this script can cause trouble, which is
why we're displaying this warning and provide the opportunity to cancel the
installation.

If you installed the current Docker package using this script and are using it
again to update Docker, you can safely ignore this message.

You may press Ctrl+C now to abort this script.
+ sleep 20
+ sudo -E sh -c apt-get update -qq >/dev/null
+ sudo -E sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq apt-transp
ort-https ca-certificates curl >/dev/null
+ sudo -E sh -c install -m 0755 -d /etc/apt/keyrings
+ sudo -E sh -c curl -fsSL "https://download.docker.com/linux/raspbian/gpg" | gp
g --dearmor --yes -o /etc/apt/keyrings/docker.gpg
+ sudo -E sh -c chmod a+r /etc/apt/keyrings/docker.gpg
+ sudo -E sh -c echo "deb [arch=armhf signed-by=/etc/apt/keyrings/docker.gpg] ht

```

Before seamlessly using Docker, a minor adjustment to our user configuration is necessary. This is related to how the Linux permission system interacts with Docker. By default, only the Docker user can engage with Docker, but a workaround is available.

A few additional tasks remain after Docker completes its installation on your Raspberry Pi. To enable another user to interact with Docker, it must

be included in the Docker group. Therefore, our subsequent action involves adding the end user to the docker group by using the usermod Command, as illustrated below. Utilizing "\$USER," we incorporate the environment variable storing the current user's name.

```
pi@raspberrypi:~ $ sudo usermod -aG docker $USER
```

Failure to include our user in the group will result in an inability to engage with Docker without assuming the role of the root user. For a more in-depth understanding of permissions and groups in Linux, refer to our guide on file permissions in Linux.

**Step 14:** As adjustments have been made to our user, logging out and back in is necessary for the changes to take effect. Execute the following Command in the terminal to log out.

- **Exit**

**Step 15.** After logging back in, you can confirm the successful addition of the docker group to your user by executing the provided Command.

- **groups**

```
pi@raspberrypi:~ $ groups
pi adm dialout cdrom sudo audio video plugdev games users input render netdev lp
admin docker gpio i2c spi
```

This Command shall display a list of all the groups to which the present user belongs. If the process unfolded correctly, the group "docker" should be visible in this list.

**Step 16.** Verifying Docker Installation on Raspberry Pi

To confirm the functionality of Docker, proceed by executing the provided Command on your Pi. This Command instructs Docker to download, set up, and run a container named "hello-world."



```
pi@raspberrypi:~ $ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (arm32v7)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

If Docker has been installed successfully on your Raspberry Pi, you should observe a message containing the following text.

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

**Step 17:** You have the option to decide whether to utilize Portainer for Docker container management. You can execute it directly through the command line or integrate it with Portainer. By doing so, Portainer provides a user-friendly web interface for managing your Bitwarden container. For any detailed instructions, refer to the guide on installing Portainer on the Raspberry Pi.

#### **Step 18. Getting Your Raspberry Pi Ready for Portainer**

We need to do a few things to get your Raspberry Pi ready to run Portainer. While the program is simple to install and use, it cannot be launched on a fresh, clean installation.

The first thing you need to do is install Docker on your Raspberry Pi. The post demonstrates how to install Docker and set it properly for the Raspberry Pi.

You can safely go to the following step once Docker has been installed. If you already have Docker installed, please let us know so that our Pi is entirely up to date.

#### **Step 19. Portainer installation on the Raspberry Pi**

We can install the Portainer software now that we have set up our Raspberry Pi. Fortunately, this is a relatively straightforward operation because Portainer operates within a Docker container.



After installing and configuring Docker, we can install Portainer on our Raspberry Pi. Because Portainer is accessible as a Docker container on the official Docker hub, we can use the following command to get the most recent version.

- **sudo Docker pull portainer/portainer-ce: latest**

```
pi@raspberrypi:~ $ sudo docker pull portainer/portainer-ce:latest
latest: Pulling from portainer/portainer-ce
Digest: sha256:eedfaee91bcb43025c072278ada094a548e2ce27ecd64db83a2ce6cf172489e9
Status: Image is up to date for portainer/portainer-ce:latest
docker.io/portainer/portainer-ce:latest
```

This Command downloads the Docker image to your device, allowing us to run it. We expressly suggest that it obtain the newest version of the container by including "latest" at the end of the pull request.

**Step 20.** We may now launch the Portainer image after Docker has finished downloading it to your Raspberry Pi. We must provide a few additional arguments to tell Docker to execute this container.

To launch Portainer, enter the following command into your Pi's terminal.

```
pi@raspberrypi:~ $ sudo docker run -d -p 9000:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest
07a987541f4910c39f3a0f3948725571965567d3799a756eb13469574fae8c05
```

One of the first things we do is designate which ports we want Portainer to have access to. This will be port 9000 in our instance. The docker container was given the name "portainer" so we could immediately identify it if necessary.

In addition, we informed the Docker management that we want it to restart this Docker if it ever goes offline accidentally.

**Step 21.**

## Using Portainer's Web User Interface

This section will show you how you can access the Portainer web interface that is running on your Raspberry Pi.

We will also show you how to complete the initial setup experience of the software.



## Accessing the Web Interface

Before we can do anything with the software, you will need to connect to its web interface.

To do this, you will need a web browser of your choice and know your Raspberry Pi's IP address.

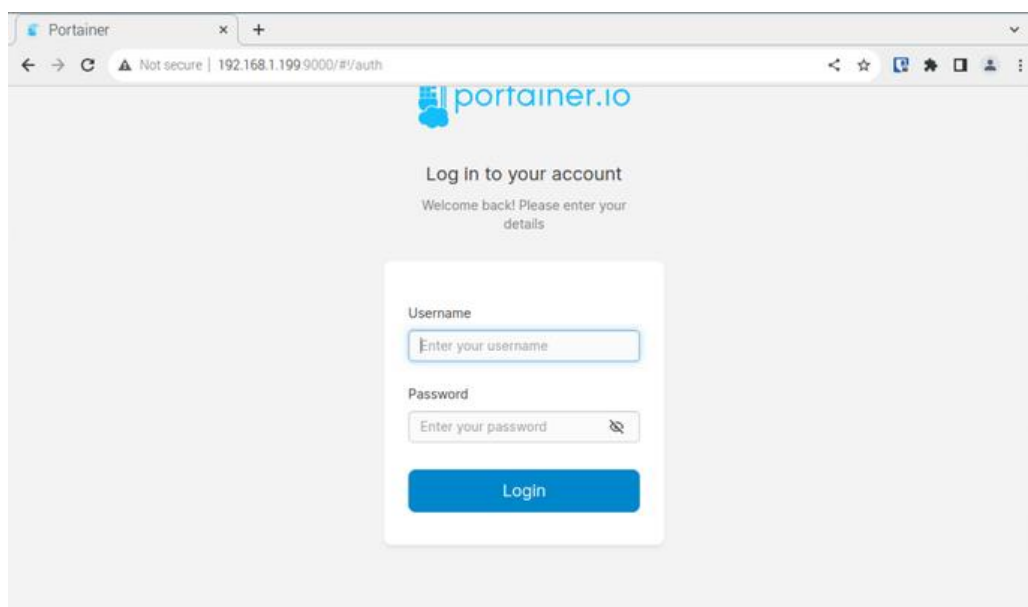
Figure 12.17

You can find it relatively easily if you need to know your Raspberry Pi's local IP address. We may have the hostname command report out the local IP address allocated to our Pi.

```
pi@raspberrypi:~ $ hostname -I  
192.168.1.199 172.17.0.1 169.254.10.137
```

When executing this Command, be sure to use a capital I.

**Step 22.** To visit Portainer, navigate to the following URL in your browser. As you can see, we mentioned the port "9000" at the end of this address. This is the location where Portainer may be found.



Make careful to change "[PIIPADDRESS]" with your Raspberry Pi's local IP address. This IP should have been obtained in the previous stage.

**Step 23.**

## Configuring Portainer on your Raspberry Pi

When you first connect to the Portainer web interface, you will need to do some initial setup steps.

Don't worry, as these are incredibly straightforward to follow and helps ensure it will run as we want it to.

When you initially start Portainer's online interface, you must create an admin account. It would help if you gave this admin account a username to make it.

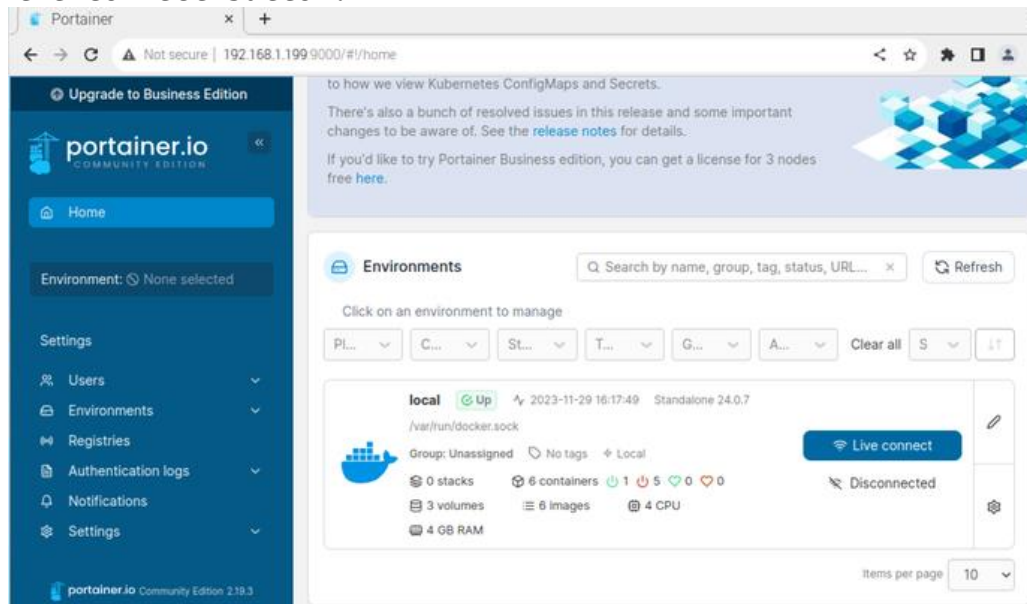
It would help if you created a password for this new account. Passwords must be at least 12 characters long to be accepted by Portainer. Once you've decided on a username and password, click the "Create user" button to complete the process.

- **username:** admin
- **password:** raymundpabaira2023
- **confirm password:** raymundpabaira2023

The screenshot shows the Portainer web interface with the title 'Please create the initial administrator user:'. The form has three input fields: 'Username' with the value 'pimyadmin', 'Password' with masked characters, and 'Confirm password' with masked characters and a green checkmark. Below the fields, a message states '✓ The password must be at least 8 characters long'. At the bottom, there is a 'Create user' button with a red arrow pointing to it labeled '3.'. A red arrow labeled '1.' points to the 'Username' field, and a red arrow labeled '2.' points to the 'Confirm password' field. A checkbox at the bottom is checked and labeled 'Allow collection of anonymous statistics. You can find more information about this in our privacy policy.'

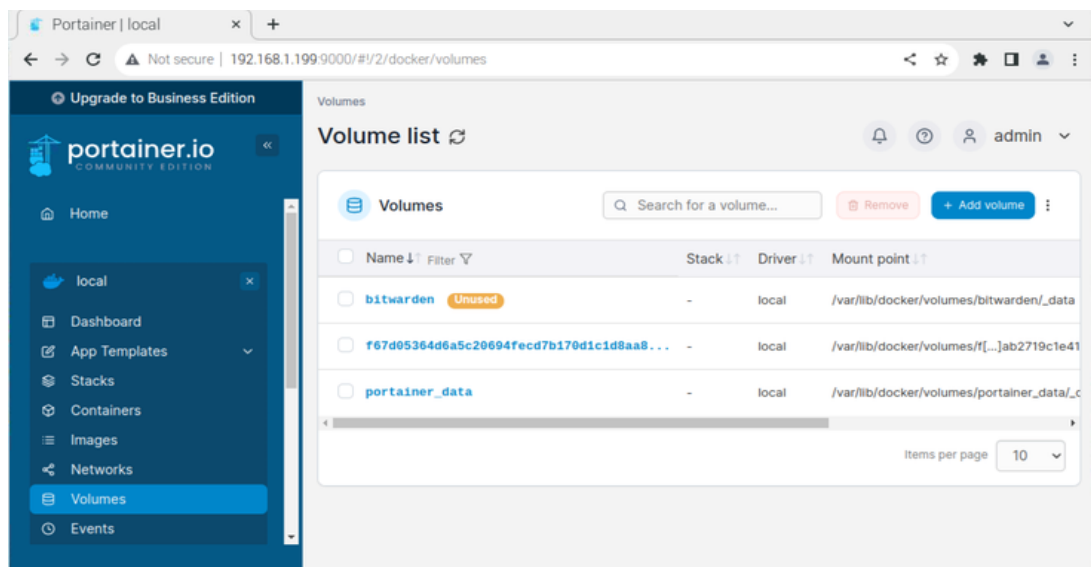
**Step 24.** We must now decide what container environment we want Portainer to handle.

- click the **Connect** button.

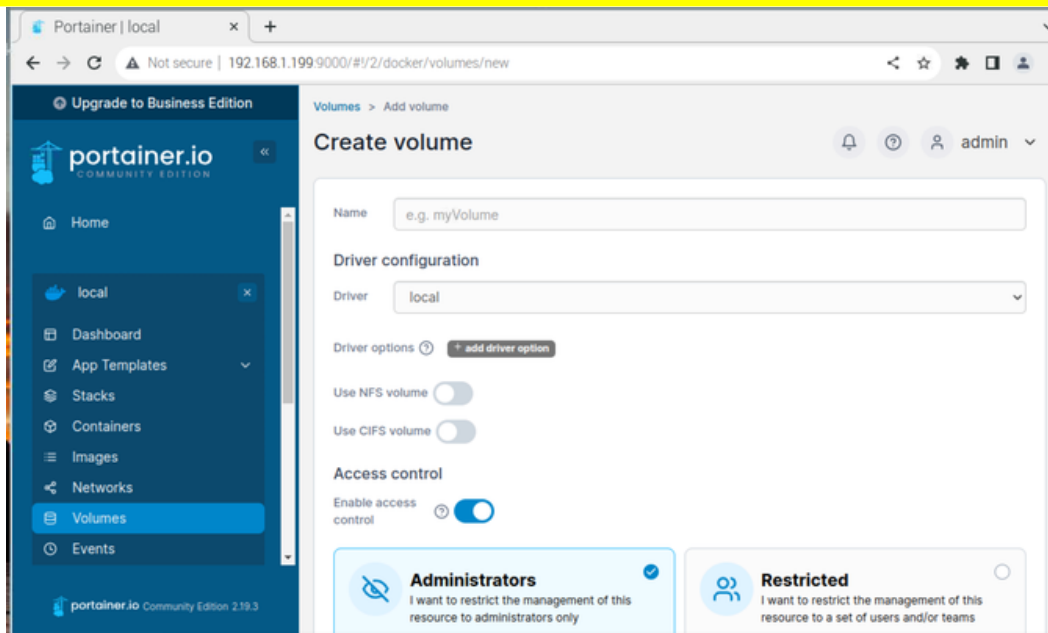


**Step 25.** You should now be able to access Portainer on your Raspberry Pi. It may now be used to manage the Docker containers running on your device.

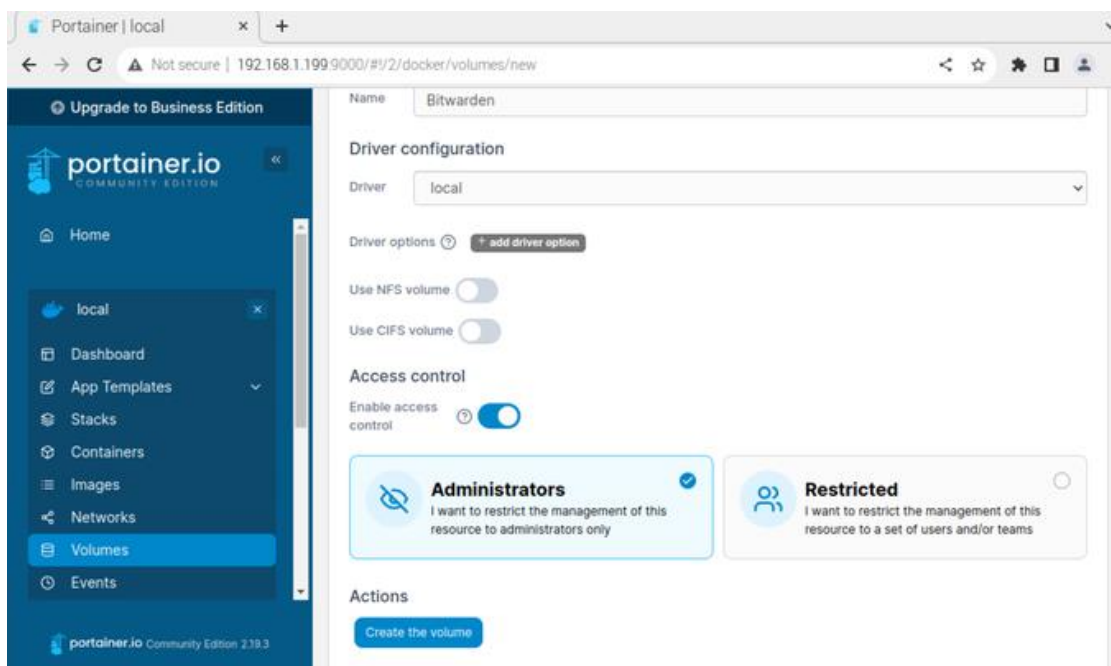
**Step 26.** We must establish a volume for the Bitwarden container before proceeding. You should notice the "Volumes" option in the sidebar; click it.



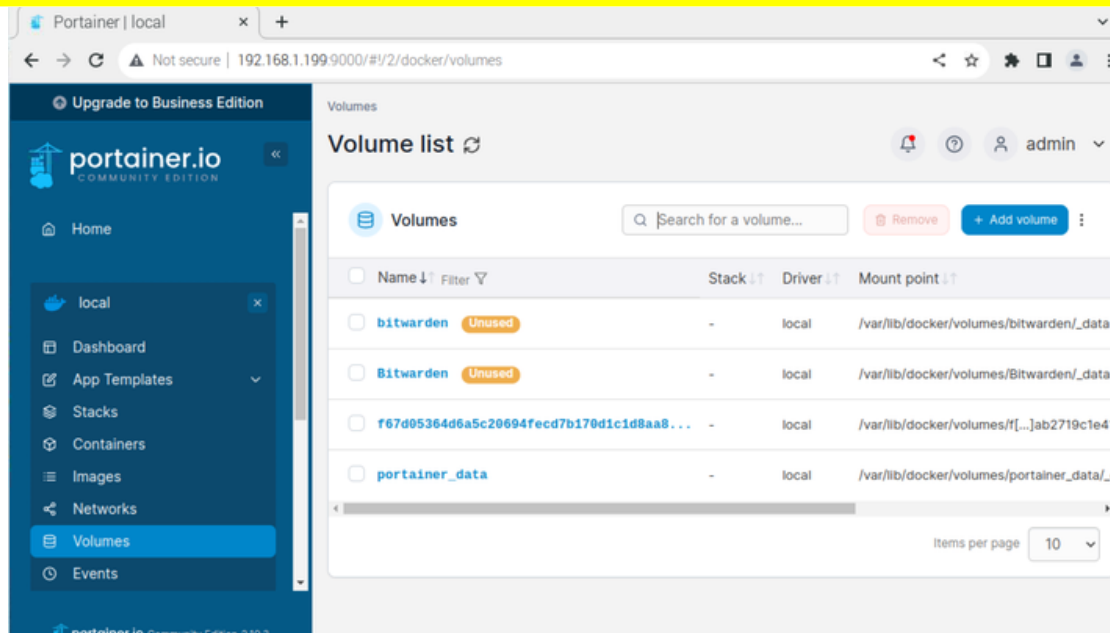
**Step 27.** You should see a list of volumes you've already created in this menu. There should be an "Add volume" button at the top of this list. To access the volume controls, click the button.



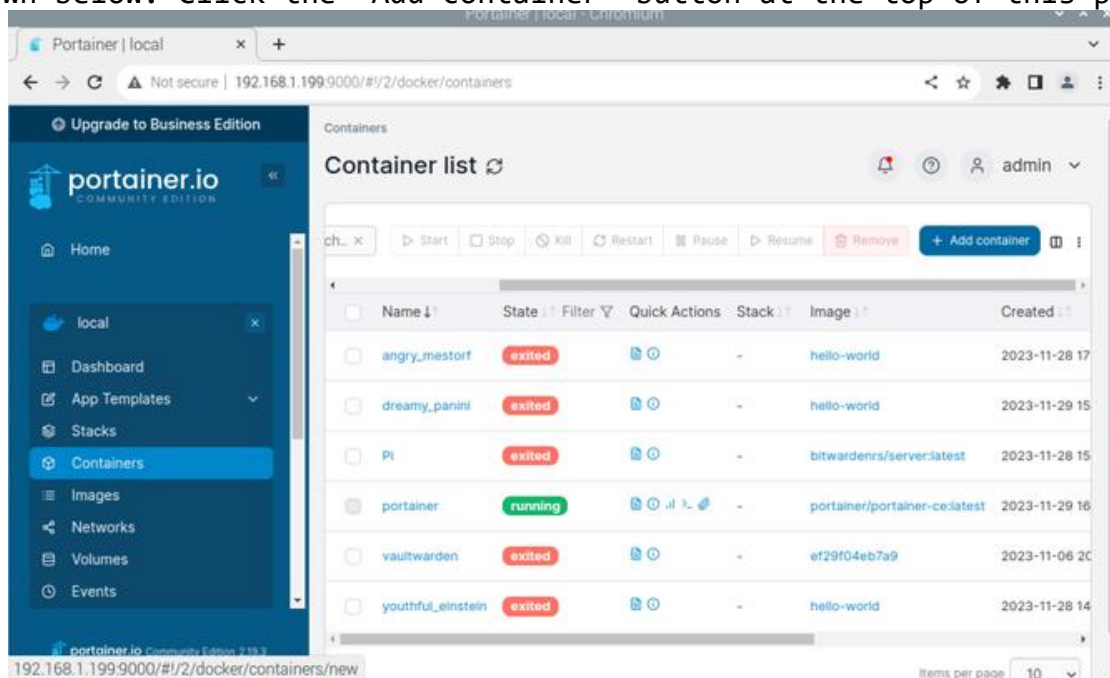
**Step 28.** We need to give this new volume a title. We'll continue using "Bitwarden" for our lesson. After you've given your Raspberry Pi Bitwarden's volume a name, click the "Create the volume" button.



**Step 29.** After we've created the volume, we'll need to navigate to the "Containers" menu. You may switch to this mode by selecting "Containers" from the sidebar.

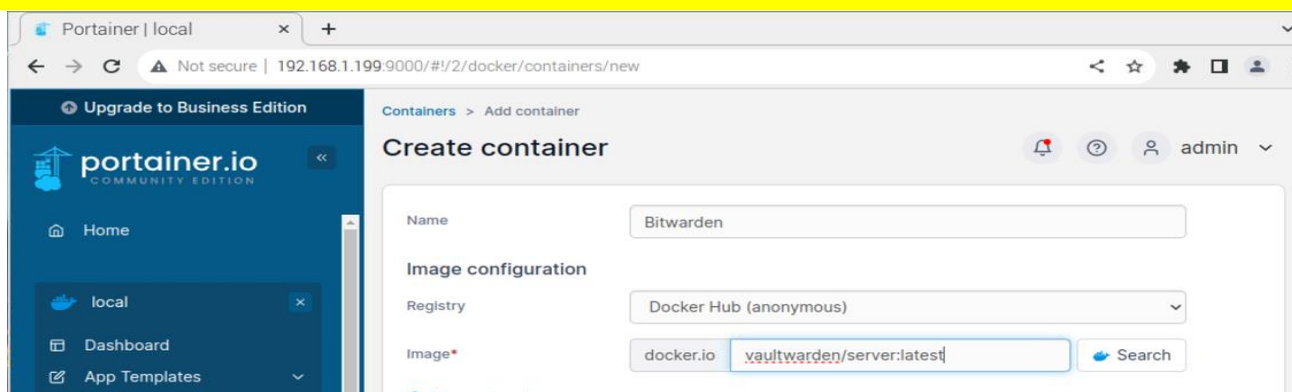


**Step 30.** You should see a list of containers accessible on your Raspberry Pi down below. Click the "Add container" button at the top of this page.



**Step 31.** Making the Bitwarden RS Container for the Raspberry Pi  
After completing the preliminary work, We may install the Bitwarden container onto our Raspberry Pi. Give your container a name first. "Bitwarden" will be the name of our container.

The image we wish to pull from the Docker hub must be specified. In our example, this will be the Bitwarden container for the Raspberry Pi. Type "vaultwarden/server:latest" into the text box next to "Image."



**Step 32.** Then, step is to configure the network settings for our container. Find the "Network ports configuration" header. You must click the "publish a new network port" button twice beneath this header.

You should now see two boxes on the screen that will enable us to specify the ports we want Bitwarden to expose. In the first box, enter "127.0.0.1:8080" for both the host and the port. You must also set the container port to 80. This is the port through which Bitwarden's web interface will be available. The second step will require changing the host to "127.0.0.1:3012".

The port "container" should be set to 3012. Bitwarden uses this port for web socket connections. It is worth noting that we have tied both host ports to the local device. This is because we do not require any additional equipment to access these. Instead, an NGINX reverse proxy will be used to expose these.

#### Network ports configuration

Publish all exposed network ports to random host ports ☐

Manual network port publishing ☐ [+ publish a new network port](#)

host	127.0.0.1:8080	→	container	80	TCP	UDP	
host	127.0.0.1:3012	→	container	3012	TCP	UDP	

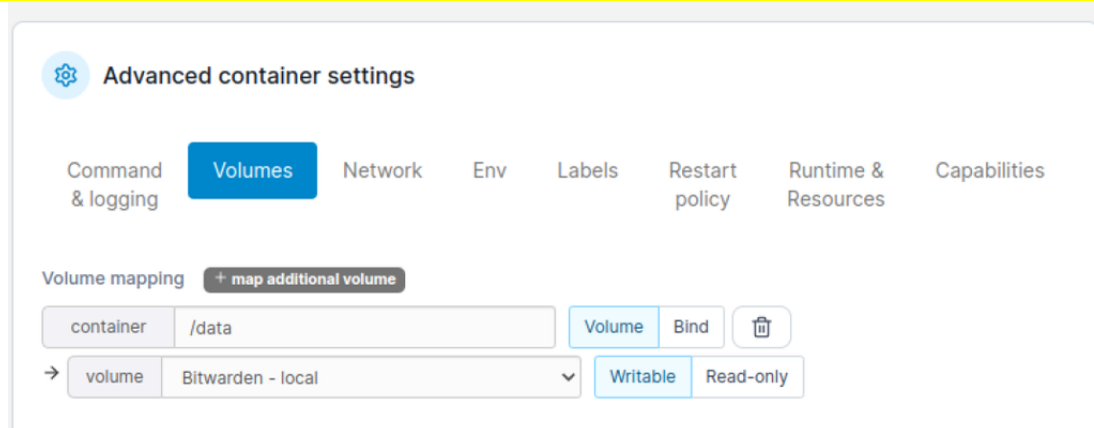
#### Access control

Enable access control ☒

**Step 33.** Scroll down to the "Advanced container settings" heading at the bottom of the page. Then, by clicking it, switch to the "Volumes" tab. Then, select the "map additional volume" option to add the Raspberry Pi's volume to the container.

Choose "/data" for the "container" option. Finally, we must select the volume we established in the previous stage. The term "Bitwarden-local" should be used.





**Step 34.** Next, we'll make sure Bitwarden remains online on our Raspberry Pi. We can accomplish this by modifying the restart policy. After that, click the "Restart policy" button to switch tabs. We want to alter the restart policy from "Always" to "Always." When the container fails, Docker will try to keep it online.

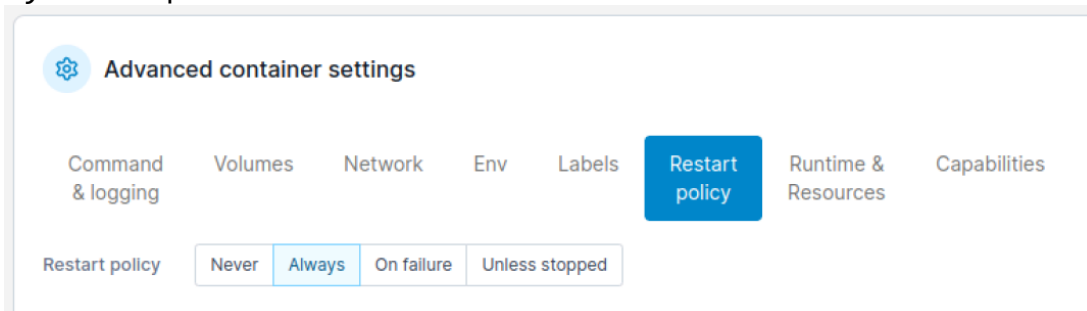


Figure 12.31

**Step 35.** We can now deploy the Bitwarden container when we have done configuring it. You must click the "Deploy the container" button under this page's "actions" header.

## Actions

Auto remove ?



Deploy the container

**Step 36.** Once Portainer has downloaded the Bitwarden docker image to your Raspberry Pi, it will appear in the containers list. We've included a screenshot of our container list after Bitwarden was successfully installed.



**Step 37.**



## Installing Bitwarden using the Docker CLI

If you would prefer not to install and utilize Portainer to use Bitwarden on your Raspberry Pi, using the CLI is easy.

Following the steps below, we will get you to pull the Bitwarden image to your device then run it.

The first step is to use Docker to fetch the most recent version of Bitwarden RS. These procedures will download the most current version of the server and make it ready to use.

```
pi@raspberrypi:~ $ docker pull vaultwarden/server:latest
latest: Pulling from vaultwarden/server
Digest: sha256:ab9fe547277245533a28d8e0a0c4a1e1120daf469f983fd683fc13556927d4fe
Status: Image is up to date for vaultwarden/server:latest
docker.io/vaultwarden/server:latest
```

**Step 38.** You may proceed once Docker downloads Bitwarden RS to your Raspberry Pi. The picture will be run in the following step. We may accomplish this by using the following Command.

```
pi@raspberrypi:~ $ sudo docker run -d --name bitwarden_new \
  --restart=always \
  -v /bw-data:/data/ \
  -p 127.0.0.1:8081:80 \
  -p 127.0.0.1:3013:3012 \
  vaultwarden/server:latest
```

This command will start the Bitwarden RS server from which we obtained the image. Then, we specify the ports we want Docker to forward from the Bitwarden image.

In the example, we use port "8080" to expose the webserver. Then it tells port "3012," which is the port on which Bitwarden's web sockets connect. Both ports will be bound to the localhost (127.0.0.1) of the Raspberry Pi. The proxy server we put up in the next part will allow outside access to Bitwarden by enabling HTTPS.

**Step 39.**



## Setting up an NGINX Proxy for Bitwarden

Even though we have Bitwarden up and running now, it isn't possible to use it until we set up HTTPS.

This is because Bitwarden's web interface uses certain JavaScript functions that browsers only allow when running on an HTTPS connection.

To achieve support for HTTPS, we are going to have to set up a proxy using NGINX.

NGINX will sit in front of our Raspberry Pi's Bitwarden server and proxy the requests.

## Preparing NGINX on the Pi

Before we can use Bitwarden, we are required to set up NGINX. .

To do that, you need to install the webserver software then generate an SSL certificate for us to use for the HTTPS connection.

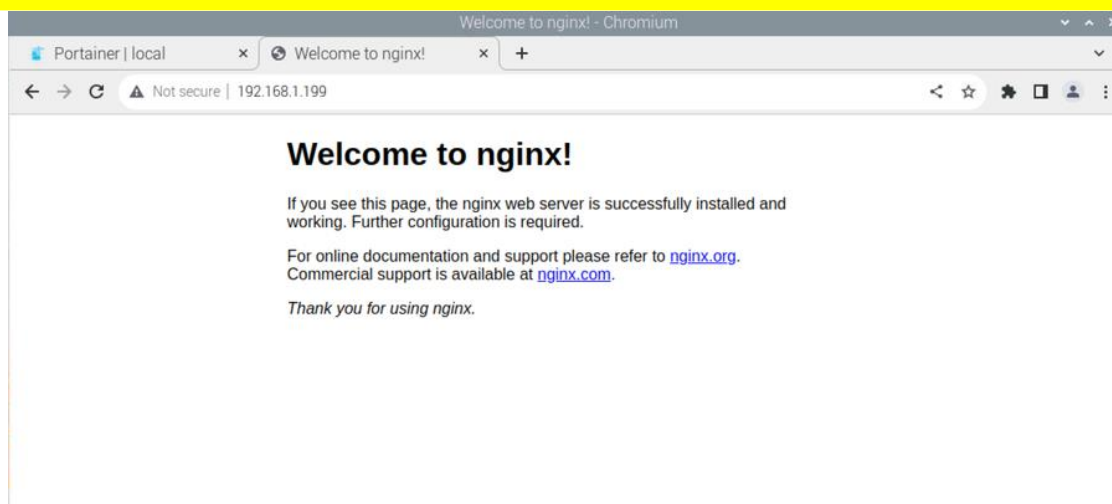
**Step 40.** With the packages updated and Apache 2 gone, we can continue with the lesson. Finally, let's install NGINX on our Raspberry Pi by typing the Command below on your Raspberry Pi.

```
pi@raspberrypi:~ $ sudo apt install nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nginx is already the newest version (1.18.0-6.1+deb11u3).
The following packages were automatically installed and are no longer required:
  libfuse2 libintl-perl libintl-xs-perl libmodule-find-perl
  libmodule-scandeps-perl libproc-processtable-perl libsort-naturally-perl
  libterm-readkey-perl needrestart tini
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

**Step 41.** We can now run the software after installing NGINX. To start the web server on your Raspberry Pi, enter the following command into the terminal.

```
pi@raspberrypi:~ $ sudo systemctl start nginx
```

**Step 42.** Now that we have the local IP address of our Raspberry Pi, we can open it in any web browser. Navigate to the local IP address you acquired by running `hostname -I`.



When you navigate to the address, you should see something like the Above. Refrain from being concerned if this displays an Apache page; NGINX does not always override the Apache default index page.

#### Step 43. Setting Up NGINX for PHP

Unlike Apache, NGINX will not be configured to work with PHP automatically. To get it to load in, we need to make adjustments to its configuration files.

Because of the way NGINX works, we will also have to use PHP-FPM rather than regular PHP. Before we begin setting PHP for NGINX, we must first install PHP 7.4 and several suggested PHP modules that will make working with more complex PHP scripts simpler. You may install everything by using the following Command.

```
pi@raspberrypi:~$ sudo apt install php-fpm php-mbstring php-mysql php-curl php-gd php-zip php-xml -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
php-curl is already the newest version (2:7.4+76).
php-fpm is already the newest version (2:7.4+76).
php-gd is already the newest version (2:7.4+76).
php-mbstring is already the newest version (2:7.4+76).
php-mysql is already the newest version (2:7.4+76).
php-xml is already the newest version (2:7.4+76).
php-zip is already the newest version (2:7.4+76).
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data apache2-utils augeas-lenses libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libaugeas0 libfuse2 libintl-perl libintl-xs-perl liblua5.3-0 libmodule-find-perl
  libmodule-scandeps-perl libproc-processtable-perl libsort-naturally-perl libterm-readkey-perl
  needrestart python3-augeas tini
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

**Step 44.** Now that PHP-FPM is installed, we can make the necessary changes to the default NGINX configuration file. Run the following Command on Raspberry Pi to alter the default configuration file.

- **sudo nano /etc/nginx/sites-enabled/default**

**Find**

Find &gt;

 Copy

```
index index.html index.htm;
```

**Replace With**

Replace With &gt;

 Copy

```
index index.php index.html index.htm;
```

Here we need to add `index.php` to the index line, and this tells NGINX to recognize the `index.php` file as a possible index, adding it first in the list means it will be selected over an `index.html` file.

**Find**

Find &gt;

 Copy

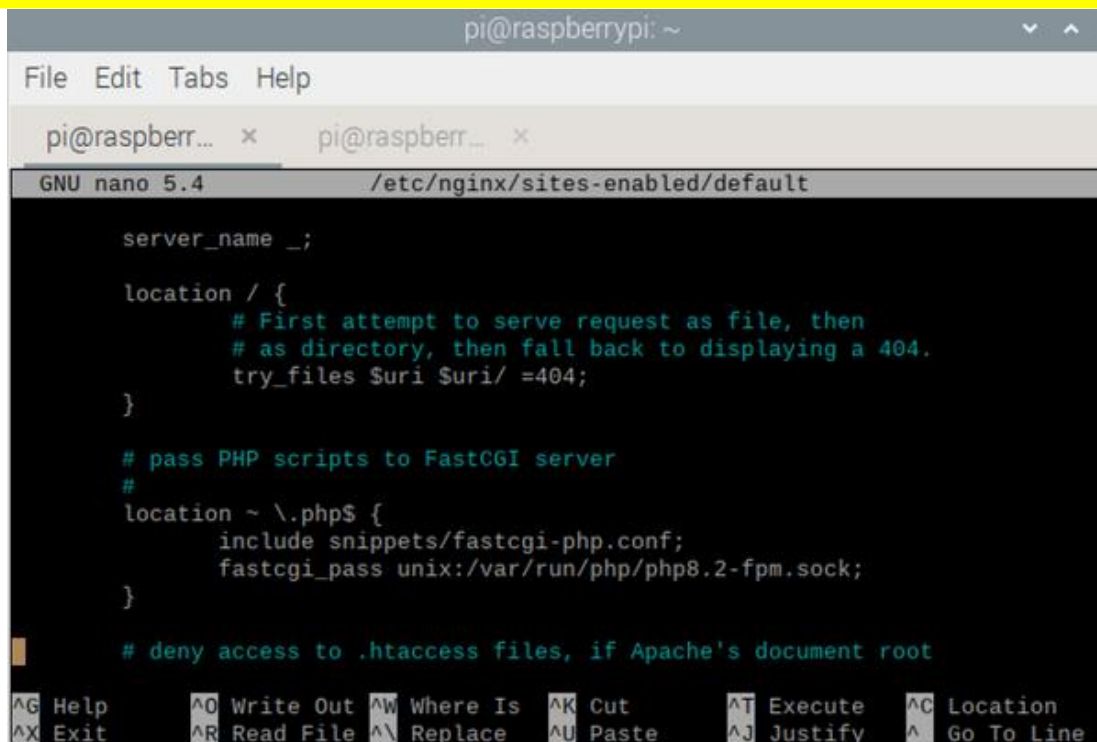
```
#location ~ \.php$ {  
    #       include snippets/fastcgi-php.conf;  
    #  
    #   # With php5-cgi alone:  
    #   fastcgi_pass 127.0.0.1:9000;  
    #   # With php5-fpm:  
    #   fastcgi_pass unix:/var/run/php5-fpm.sock;  
    #}
```

**Replace With**

Replace With &gt;

 Copy

```
location ~ \.php$ {  
    include snippets/fastcgi-php.conf;  
    fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;  
}
```



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberr... x pi@raspberr... x
GNU nano 5.4 /etc/nginx/sites-enabled/default

server_name _;

location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ =404;
}

# pass PHP scripts to FastCGI server
#
location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
}

# deny access to .htaccess files, if Apache's document root

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^N Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

**Step 45.** Following that, in this Raspberry Pi Nginx server tutorial, we will instruct NGINX to refresh its settings by issuing the following Command.

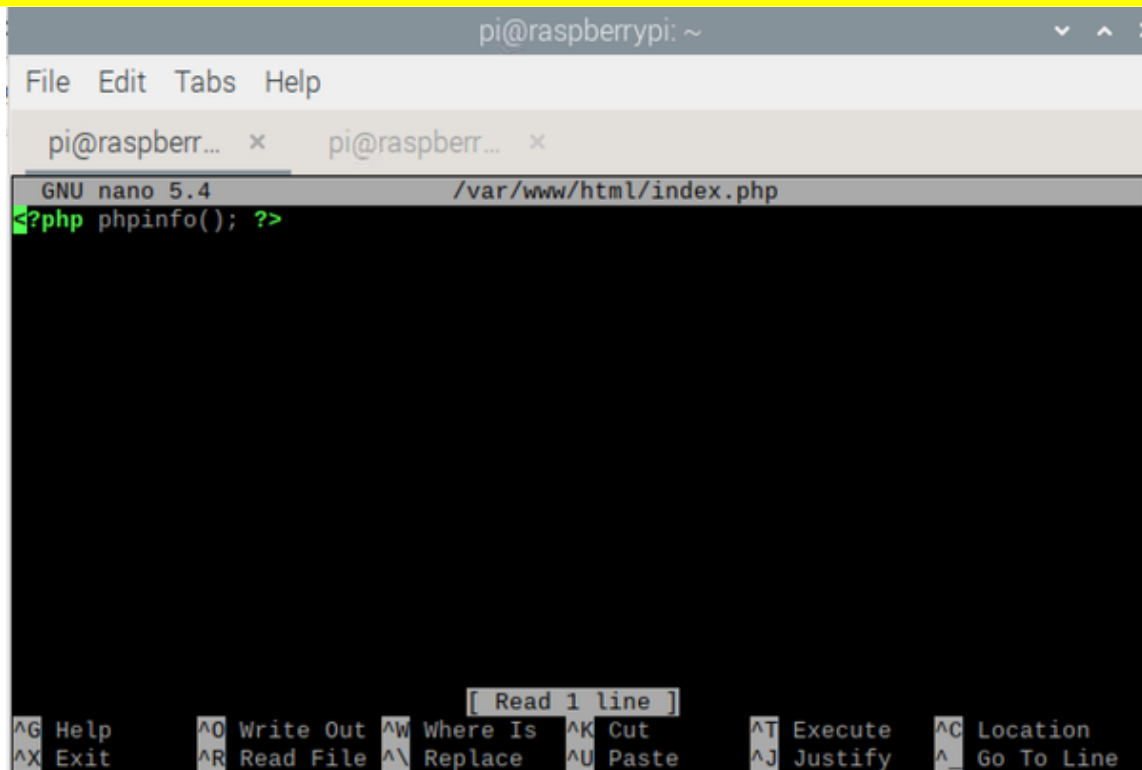
```
pi@raspberrypi:~ $ sudo systemctl reload nginx
```

**Step 46.** Finally, let's put the PHP configuration to the test by creating a basic index.php file in our /var/www/html directory. To create and begin altering our index.php file, run the following Command.

```
pi@raspberrypi:~ $ sudo nano /var/www/html/index.php
```

**Step 47.** Add the following line of code to this file.





```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberr... x pi@raspberr... x
GNU nano 5.4 /var/www/html/index.php
<?php phpinfo(); ?>
[ Read 1 line ]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^_ Go To Line
```

**Step 48.** On your Raspberry Pi, you may generate a certificate in two methods.

Another option is to produce a self-signed certificate. While they are still secure, the browser will notify you that it cannot verify its legitimacy from a certificate authority. It can produce this certificate on the Raspberry Pi with a single command, making it easy to use.

To generate a self-signed certificate for BirWarden, use the command below.

```
• sudo openssl req -x509 -nodes -days 365 -newkey rsa:4096 -keyout /etc/ssl/private/nginx-bitwarden.key -out /etc/ssl/certs/nginx-bitwarden.crt
```

This self-signed certificate will be saved in the "/etc/ssl/private/" and "/etc/ssl/certs/" folders for 365 days. You will be requested to supply some more information throughout the SSL creation procedure. These details will be written to the certificate by OpenSSL to assist you in determining whether or not it is a counterfeit.

```
pi@raspberrypi:~ $ sudo openssl req -x509 -nodes -days 365 -newkey rsa:4096 -key
out /etc/ssl/private/nginx-bitwarden.key -out /etc/ssl/certs/nginx-bitwarden.crt
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/etc/ssl/private/nginx-bitwarden.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PH
State or Province Name (full name) [Some-State]:PANGASINAN
Locality Name (eg, city) []:URDANETA
Organization Name (eg, company) [Internet Widgits Pty Ltd]:PSU
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:192.168.1.199
Email Address []:
apalaganas_20ur0511@psu.edu.phpi@raspberrypi:~ $
pi@raspberrypi:~ $
```

**Step 49.** Our final duty in preparation is to form a robust Diffie-Hellman group. This is used to assist in strengthening the security of SSL connections on your device.

```
pi@raspberrypi:~ $ sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.....+.....+.....
```

### Step 50.

## Configuring NGINX to act as Proxy

We can now create a virtual host for NGINX that will allow it to proxy connections to Bitwarden.

Luckily for us, configuring NGINX to act as a proxy is a very straightforward process.

First, we'll delete NGINX's default configuration file. You do not need to do this if you intend to use Bitwarden in conjunction with a domain name.

```

+
+
+
+
+*++*
pi@raspberrypi:~ $ sudo rm /etc/nginx/sites-enabled/default
pi@raspberrypi:~ $

```

**Step 51.** We need to build a new NGINX configuration file. Begin writing this new configuration file with the nano text editor by executing the

Command which is shown below.

```
pi@raspberrypi:~$ sudo nano /etc/nginx/sites-enabled/bitwarden.conf
```

Fill in the blanks with the text shown below. This initial block that we are implementing will redirect any traffic from http (port 80) to https (port 443). Following that, we must include our server block, which will handle the proxying and HTTPS connection.

Furthermore, if you generated the certificate with Lets Encrypt, you must change the paths for "ssl\_certificate" and "ssl\_certificate\_key." Depending on how you intend to use your Raspberry Pi Bitwarden server, you may need to make a few changes.

To include a domain name, replace server name \_with it. For instance, "server\_name bitwarden.pimylifeup.com" would be our server name. When finished, your file should look like the one shown below.

The Command that were used are shown below:

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberr... x pi@raspberr... x
GNU nano 5.4 /etc/nginx/sites-enabled/bitwarden.conf
server {
    listen 80;
    listen [::]:80;
    server_name _; #Change this to your domain name
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    server_name _; #Change this to your domain name

    ssl_certificate /etc/ssl/certs/nginx-bitwarden.crt; #Swap these out wi
    ssl_certificate_key /etc/ssl/private/nginx-bitwarden.key; #Swap these out wi
    ssl_dhparam /etc/ssl/certs/dhparam.pem;

    # Allow large attachments
    client_max_body_size 128M;

    location / {
        [ Read 37 lines ]
    }
}
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^_ Go To Line

```

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberr... x pi@raspberr... x
GNU nano 5.4 /etc/nginx/sites-enabled/bitwarden.conf

location / {
    proxy_pass http://0.0.0.0:8080;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

location /notifications/hub {
    proxy_pass http://0.0.0.0:3012;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}

location /notifications/hub/negotiate {
    proxy_pass http://0.0.0.0:8080;
}

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location

```

```

server {
    listen 80;
    listen [::]:80;
    server_name _; #Change this to your domain name
    return 301 https://$host$request_uri;
}

```

```

server {
    listen 443 ssl http2;
    server_name _; #Change this to your domain name

    ssl_certificate      /etc/ssl/certs/nginx-bitwarden.crt;    #Swap these
out with Lets Encrypt Path if using signed cert
    ssl_certificate_key  /etc/ssl/private/nginx-bitwarden.key;  #Swap these
out with Lets Encrypt Path if using signed cert

    ssl_dhparam /etc/ssl/certs/dhparam.pem;

    # Allow large attachments
    client_max_body_size 128M;

    location / {
        proxy_pass http://0.0.0.0:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /notifications/hub {
        proxy_pass http://0.0.0.0:3012;
    }
}

```

```
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}

location /notifications/hub/negotiate {
    proxy_pass http://0.0.0.0:8080;
}
}
```

**Step 52.** All that remains is to restart the NGINX service. It will be unaware of any recent modifications unless the service is restarted. Run the following command on your Raspberry Pi to restart NGINX.

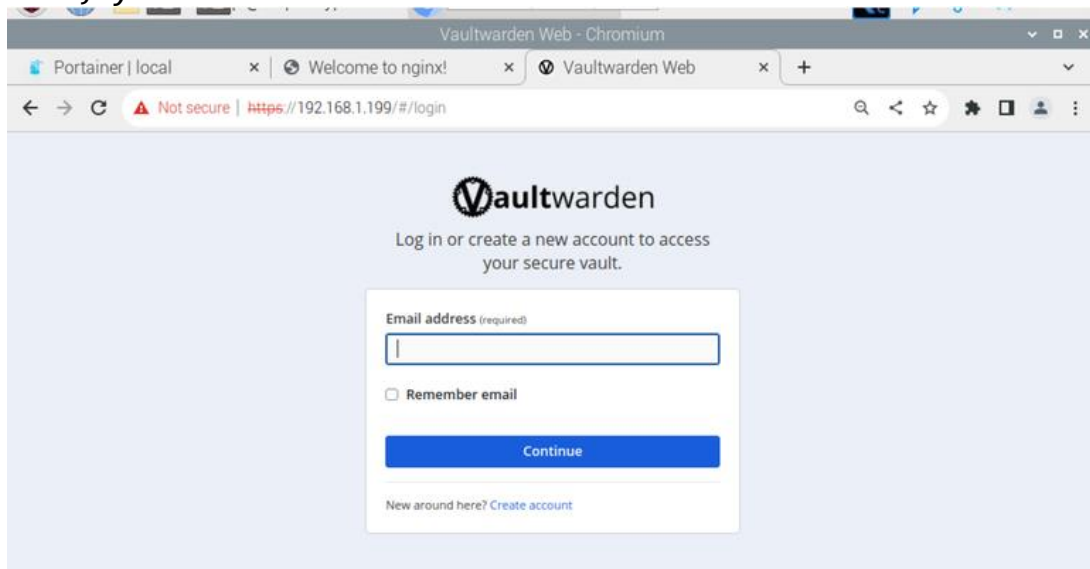
```
pi@raspberrypi:~ $ sudo systemctl restart nginx
pi@raspberrypi:~ $
```

**Step 53.**

## Accessing the Bitwarden Web Interface

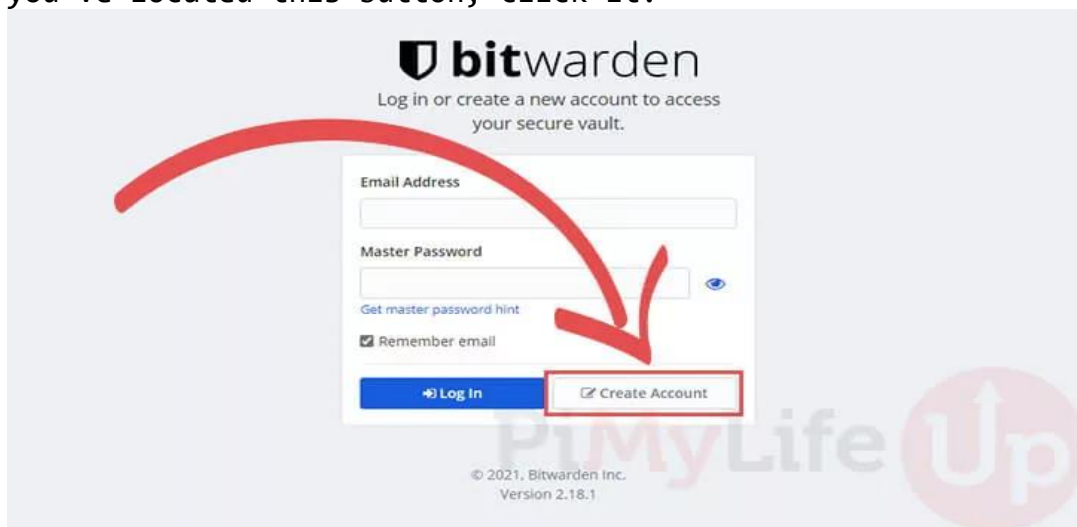
With a proxy server setup, we can now access and use the Bitwarden web interface running on our Raspberry Pi.

To access your Bitwarden interface, navigate to the following address in your preferred web browser. Make careful to change "YOURPIIPADDRESS" with the IP address of your Raspberry Pi. Alternatively, if you have a domain name, you must use it instead.



A warning notice will be displayed if you are using a self-signed certificate. You may disregard this warning because you know you created this certificate and that you are connected to your Raspberry Pi.

**Step 54.** You must register an account before you can begin using Bitwarden. The "Create Account" button should be visible on the login screen. Once you've located this button, click it.



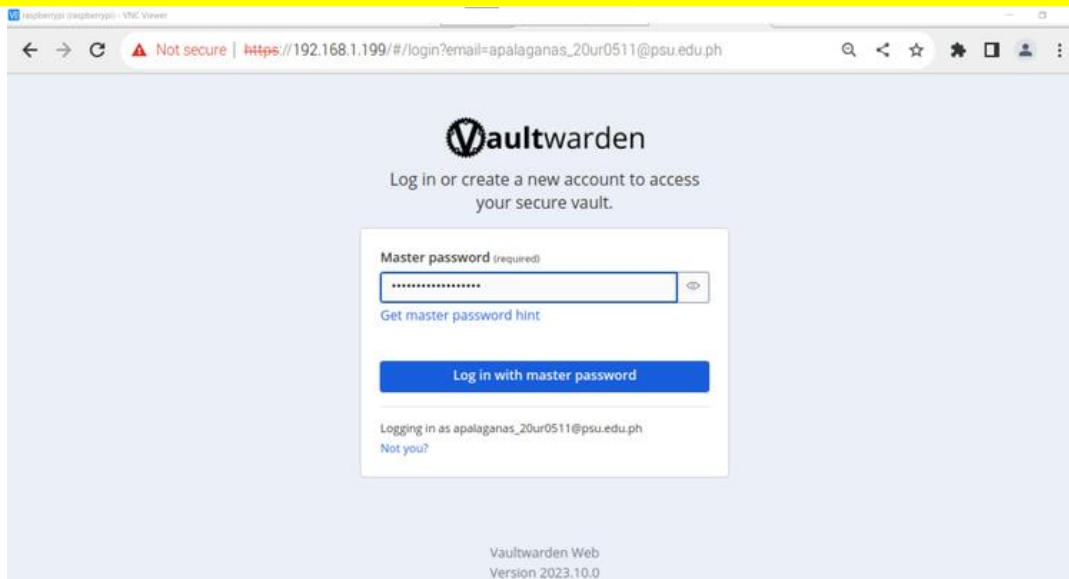
**Step 55.** You can now begin entering details for your new account. The first step is to input your Bitwarden account's email address. This is the email address you will use to log in.

Following that, you must provide a name for Bitwarden to use in its interfaces. Then, it would help if you created a password for this account. Make sure this is set to something secure and difficult to guess. Before completing the account creation process, you must agree to Bitwarden's terms of service and privacy policy.

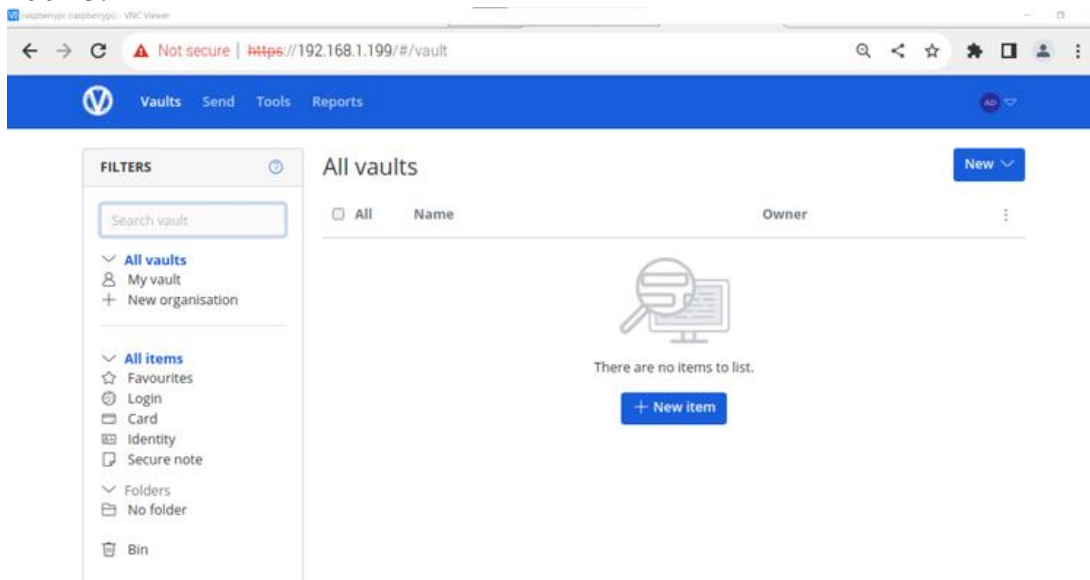
When you are satisfied with your options, click the "Submit" button to create the account.

You may now access your new Bitwarden vault after registering an account. To begin, enter the email address you provided to your account. Then, enter the password you created for your new account. Finally, click the "Log In" button to log in.





**Step 56.** You may now begin saving data in your own Raspberry Pi Bitwarden vault.<sup>[19]</sup>



**Step 57.**

## Enabling the Bitwarden Admin Panel

Now that you have created an account, we can now generate the admin token.

The admin token is what you will be using to access the Bitwarden admin panel. This will require us to make changes to our Docker containers configuration.

Within the admin panel, you will be able to view all registered users and delete them. You can even generate invites for new users even if you have disabled the functionality.

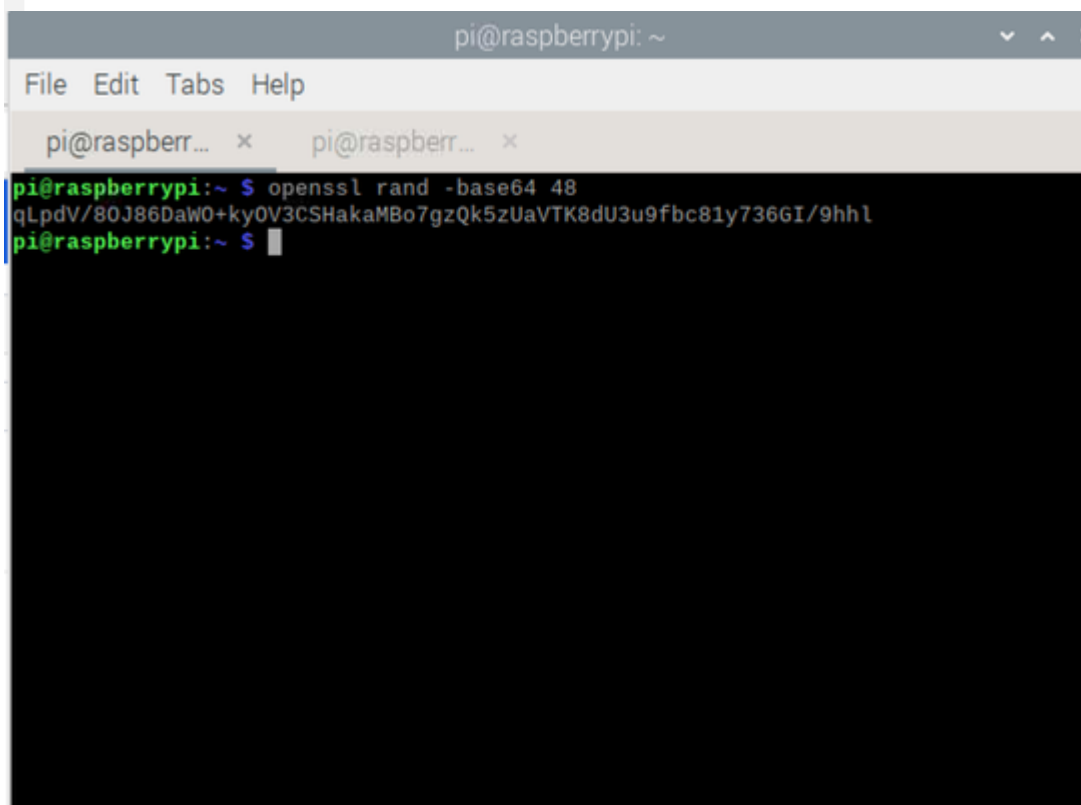
This interface is also used to configure the numerous Bitwarden options. Such as whether you want people to be able to sign up.

## Generate the Admin Token

Our first step is to generate a new admin token for Bitwarden to utilize.

As this token needs to be a relatively long string of strong randomly generated characters, we will be using `openssl`.

To generate this secure string, you can run the following command on your Raspberry Pi.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberr... x pi@raspberr... x  
pi@raspberrypi:~ $ openssl rand -base64 48  
qLpdV/80J86DaW0+ky0V3CSHakaMBo7gzQk5zUaVTK8dU3u9fbc81y736GI/9hh1  
pi@raspberrypi:~ $
```

Keep this token private, since it will grant anybody complete access to the Bitwarden RS server.

## Step 58.

### Accessing the Admin Page

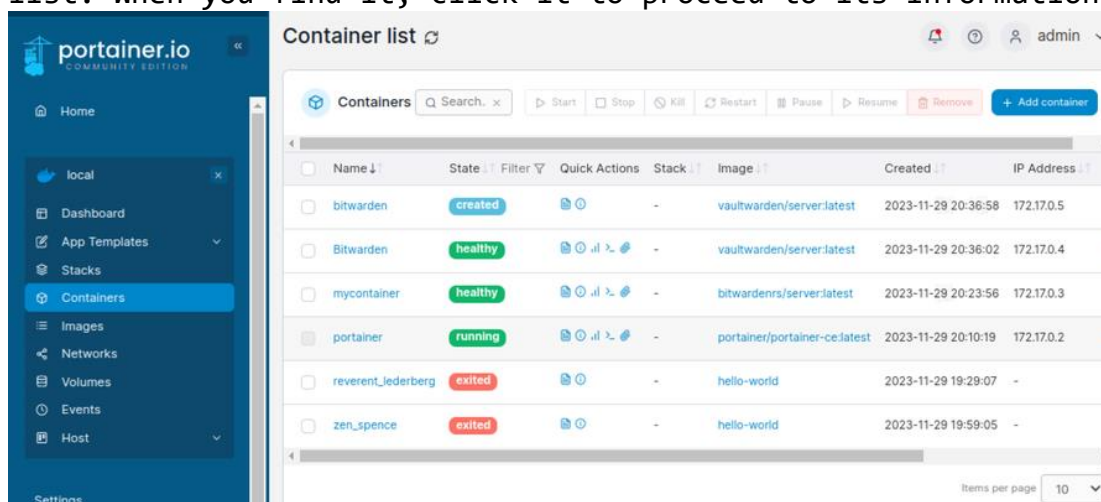
We now need to feed this new admin token into our Raspberry Pi's Bitwarden Docker container.

It is possible to do this using the Portainer web interface or by modifying the command line to feed it in.

### Setting the Admin Token using Portainer

To set the admin token, you will need to fire up the Portainer interface and return to the container list.

It would be best if you located the Bitwarden container inside the container list. When you find it, click it to proceed to its information page.



**Step 59.** A "Duplicate/Edit" button must be at the top of the next page. To begin altering the container's settings, click this button.



**Step 60.** Scroll down until you reach the "Advanced container settings" section. It would be best if you changed the tabs beneath this heading by choosing the "ENV" option. To add the admin token, you must click the "add environment variable" button. When you click this button, two additional text boxes shall appear at the bottom of the page.

You must enter "ADMIN\_TOKEN" in the "name" text field. You must enter the admin token you wish to use in the "value" field. Finally, after the admin token has been generated, you may click the "Deploy the container" button.

Advanced container settings

Command & logging Volumes Network **Env** Labels Restart policy Runtime & Resources Capabilities

Environment variables

These values will be applied to the container when deployed

☒ Advanced mode  
 Switch to advanced mode to copy & paste multiple variables

name	value
PATH	/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
ROCKET_PROFILE	release
ROCKET_ADDRESS	0.0.0.0
ROCKET_PORT	80
DEBIAN_FRONTEND	noninteractive
ADMIN_TOKEN	EXAMPLEPIMYLIFEUPADMINTOKEN

+ Add an environment variable Load variables from .env file

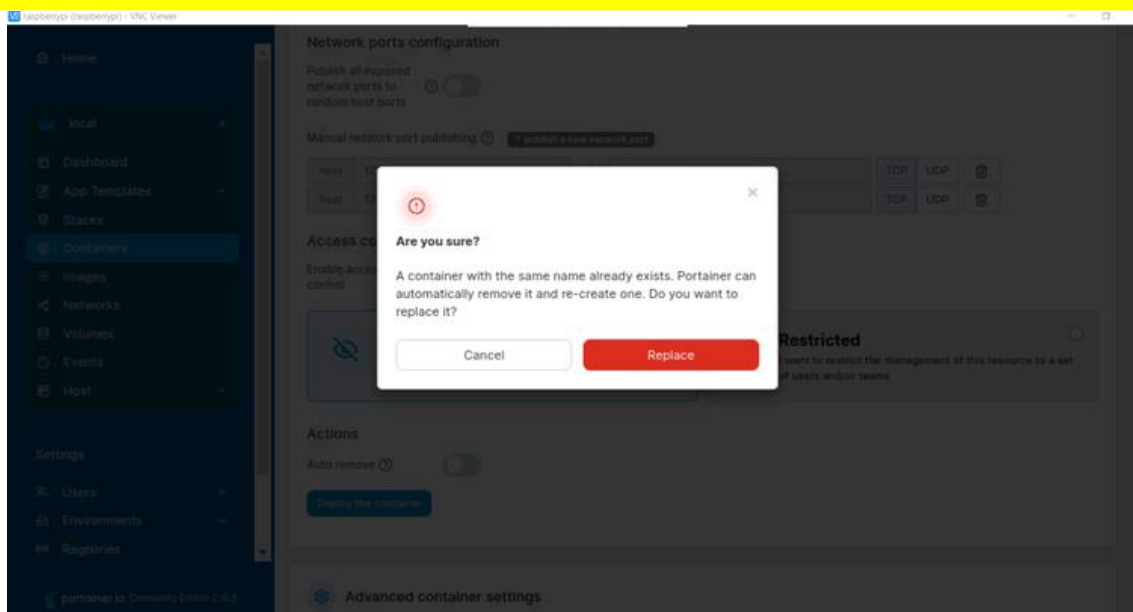
## Actions

Auto remove ?



Deploy the container

**Step 61.** The notification below will appear before Portainer updates your Raspberry Pi's Bitwarden container. This notification informs you that the existing container will be replaced since the names match. You must select the "Replace" option.

**Step 62.****Setting the Admin Token within the Command Line**

Updating a container is slightly more complicated when using the command line, as you have to delete the existing container manually.

We must first take our Raspberry Pi's Bitwarden container offline before we can remove it. The following Command will terminate a currently operating container.

```
pi@raspberrypi:~ $ sudo docker stop bitwarden
bitwarden
```

**Step 63.** The current container must then be removed. If a container with identical ports and name already exists, Docker will not allow us to reproduce it. Run the Command below to remove the existing Bitwarden container.

```
pi@raspberrypi:~ $ sudo docker rm bitwarden
bitwarden
```

**Step 64.** Finally, we must re-execute the docker command. This time, we'll use the admin token that we created before.

```
pi@raspberrypi:~ $ sudo docker run -d --name bitwarden \
-e ADMIN_TOKEN=EXAMPLEPIMYLIFEUPADMIN_TOKEN \
--restart=always \
-v /bw-data:/data/ \
-p 127.0.0.1:8080:80 \
-p 127.0.0.1:3012:3012 \
vaultwarden/server:latest
432bf13ef6ea49a9b6bcd92d92dbf062085fbb1fde18cc46f5956b1a23c7da0e
```

Make sure to substitute "EXAMPLEPIMYLIFEUPADMIN\_TOKEN" with the token you

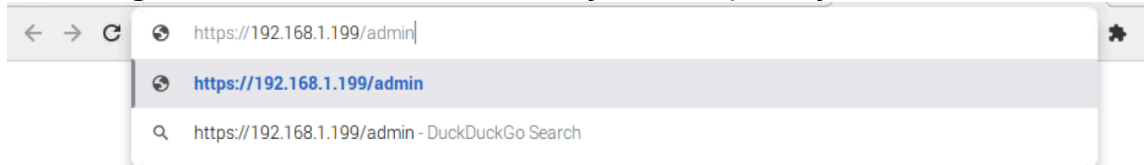
produced previously when you execute this new command.

### Step 65.

## Accessing the Bitwarden Admin Panel

Once you have finally gotten your Raspberry Pi's Bitwarden installation to accept your newly generated admin token, you can now access the admin page.

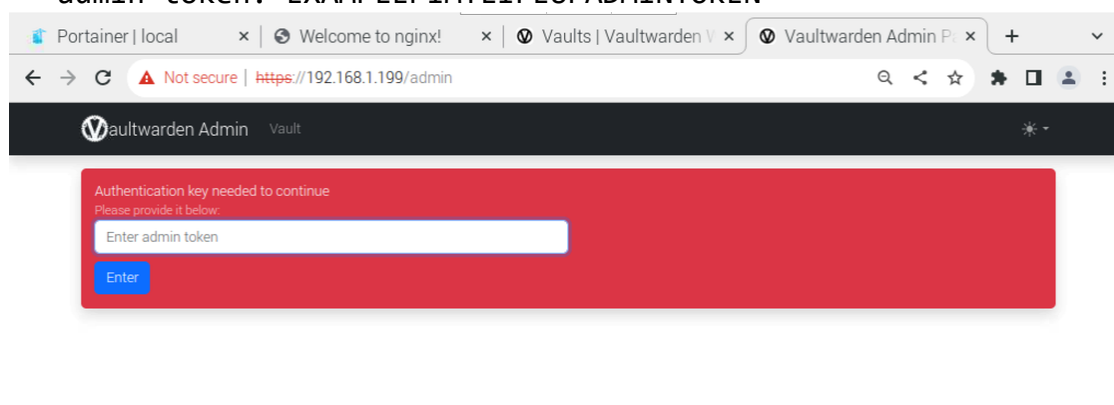
Navigate to the following location in your computer browser. Make careful to change "YOURPIIPADDRESS" with your Raspberry Pi's IP address.



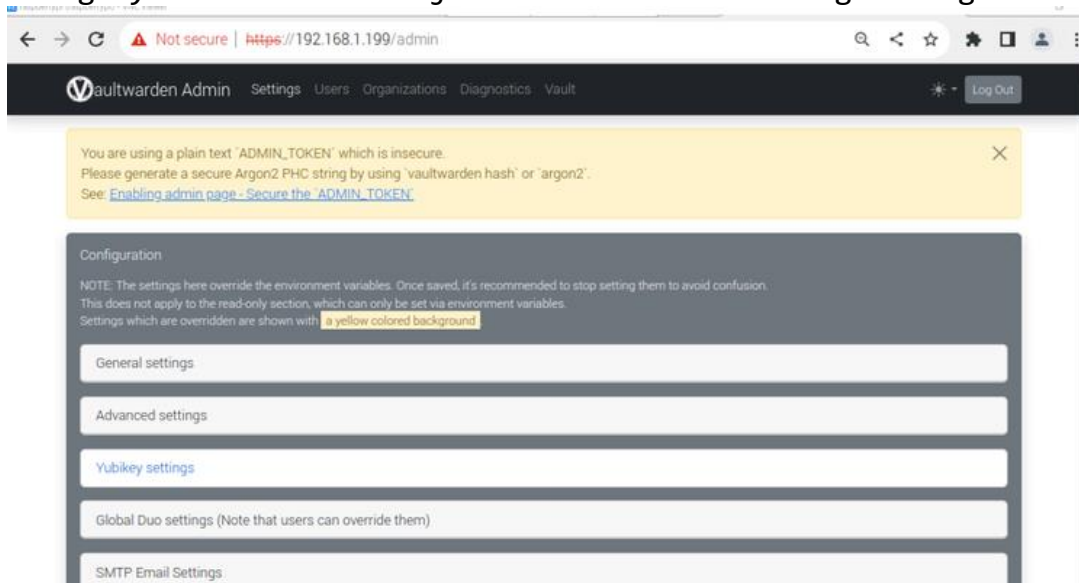
Going to this URL will take you to the Bitwarden administration panel.

**Step 66.** It would be best if you input the admin token you generated on this page. Once you've input the token, click the "Enter" button to log in.

- admin token: `EXAMPLEPIMYLIFEUPADMINTOKEN`



**Step 67.** You now gain entry to the Bitwarden administration panel. You may manage your users and adjust Bitwarden's settings using these sites.





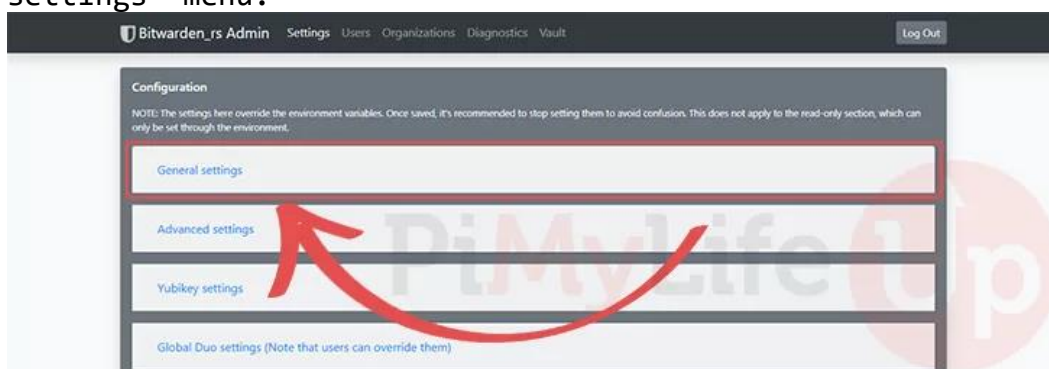
**Step 68.**

## Disable New User Creation

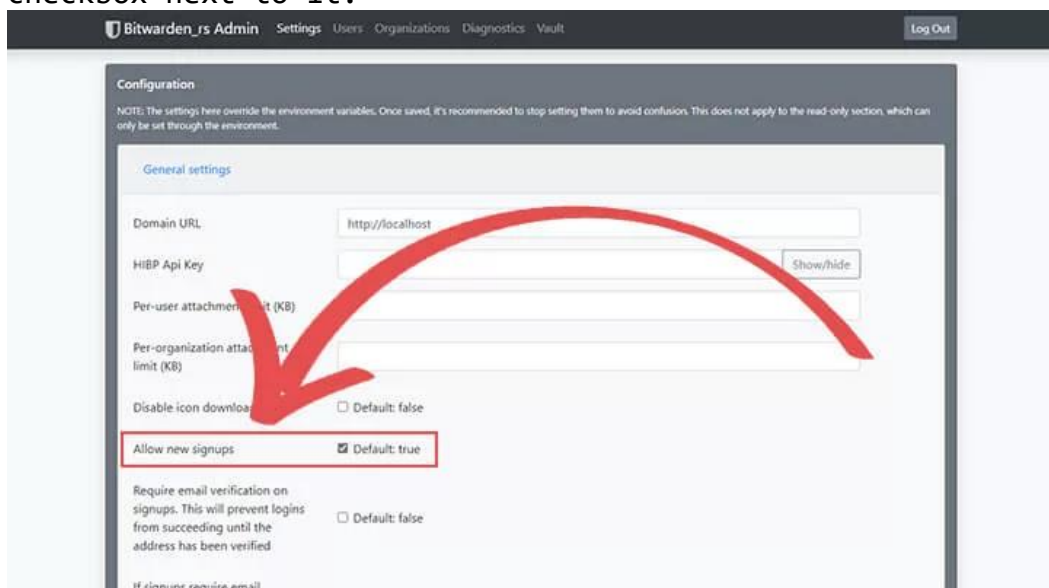
Once you have created an account and set up access to the admin panel, you can choose to disable the user registration menu.

This means that only users you invite personally will be able to create users on your Bitwarden vault.

To block new user creation on your Raspberry Pi's Bitwarden menu, open the admin panel. You can proceed once you are in the admin panel. The option to turn off new user signups is accessible on the general settings page. To see all the options buried inside that panel, click the "General settings" menu.



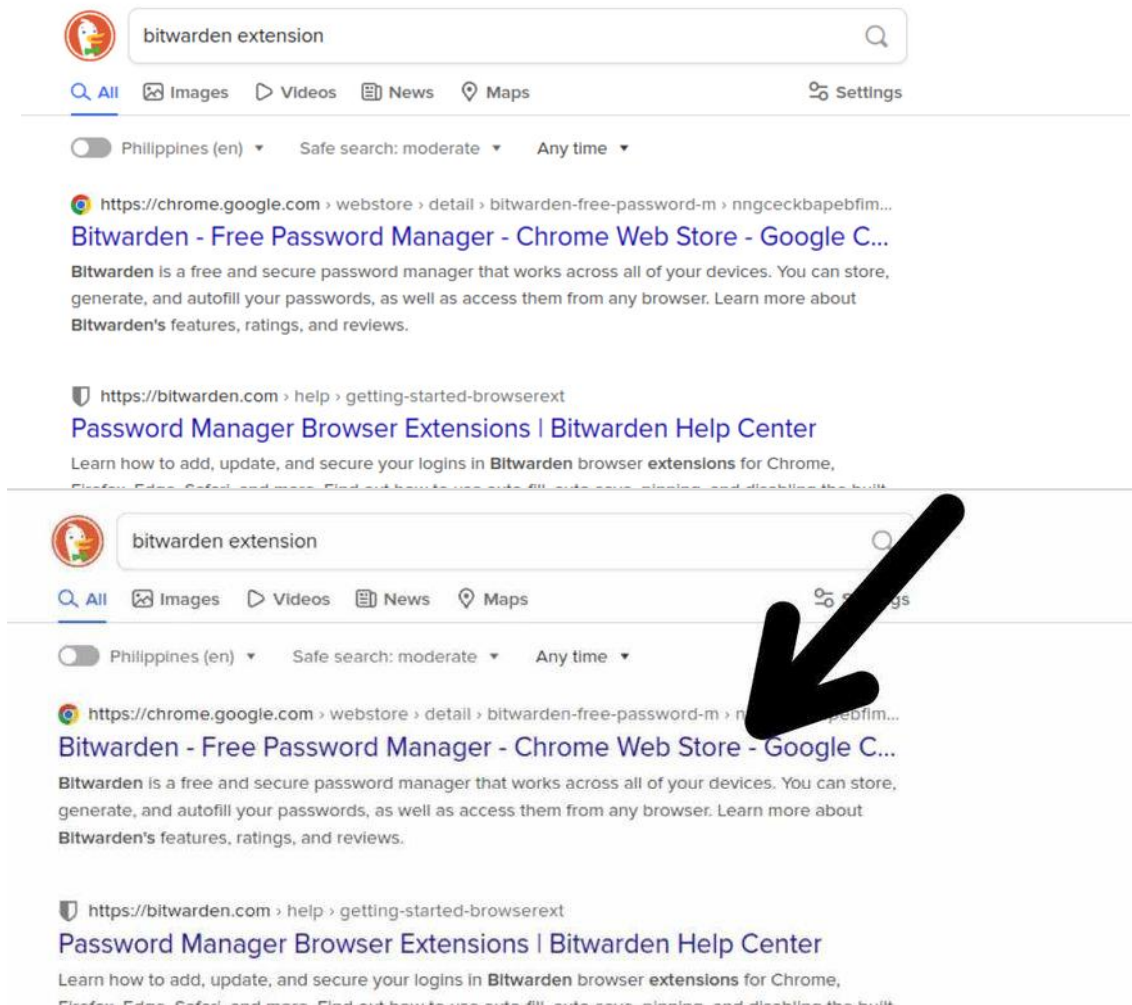
**Step 69.** You should find an option called "Allow new signups" in the "General settings" list of choices. To turn off this feature, click the checkbox next to it.



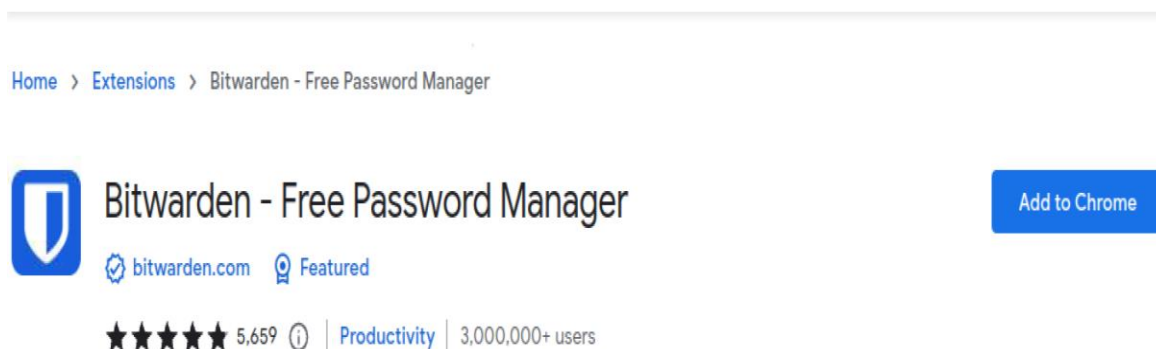
Scroll to the bottom of the page to confirm the configuration changes. You should notice a blue button at the bottom with the phrase "Save" on it. To save these changes, click this button.



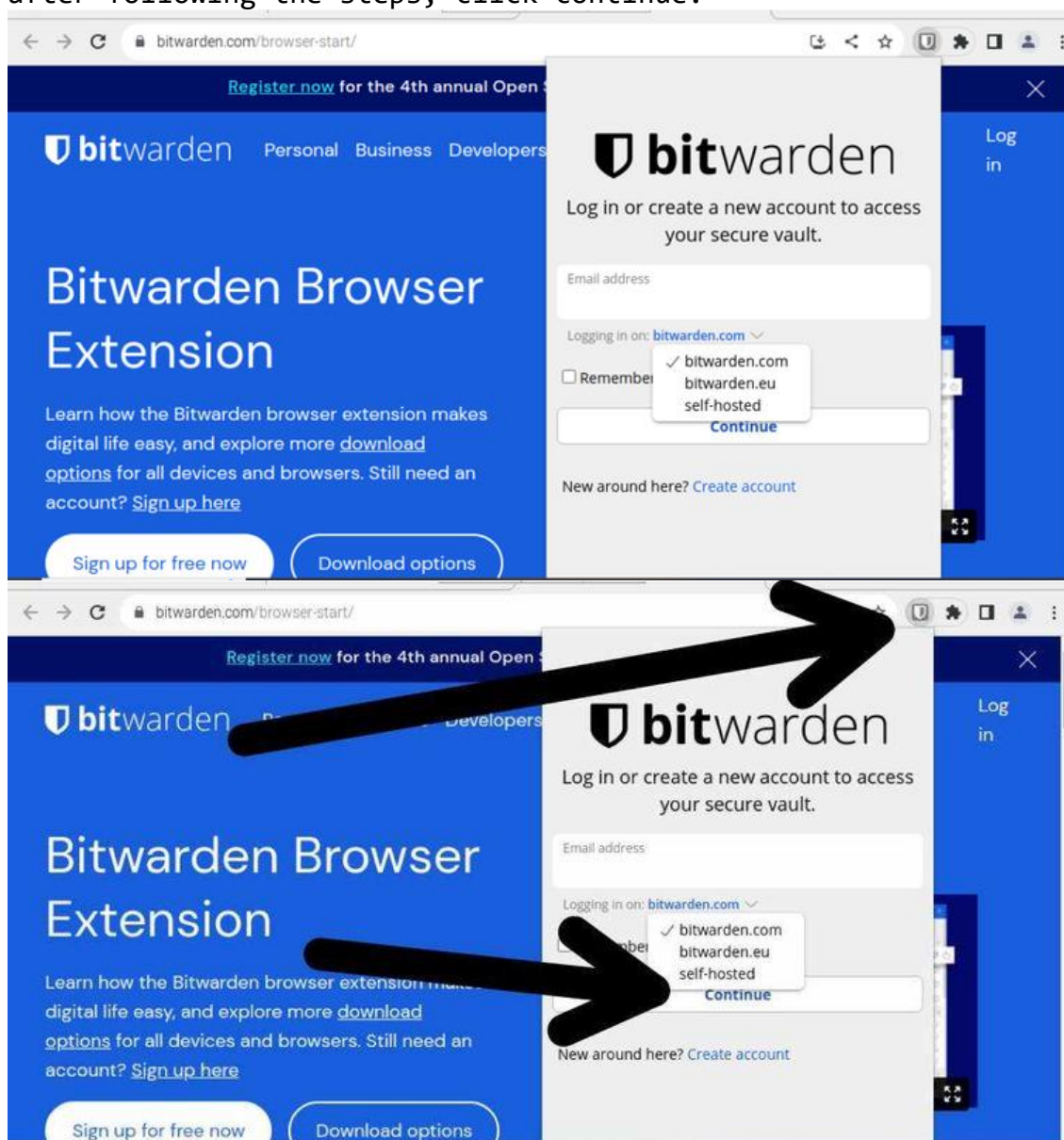
**Step 70.** Next, open a browser and type in the terms BitWarden extension. Then, you've spotted the first link that has shown and clicked on it.



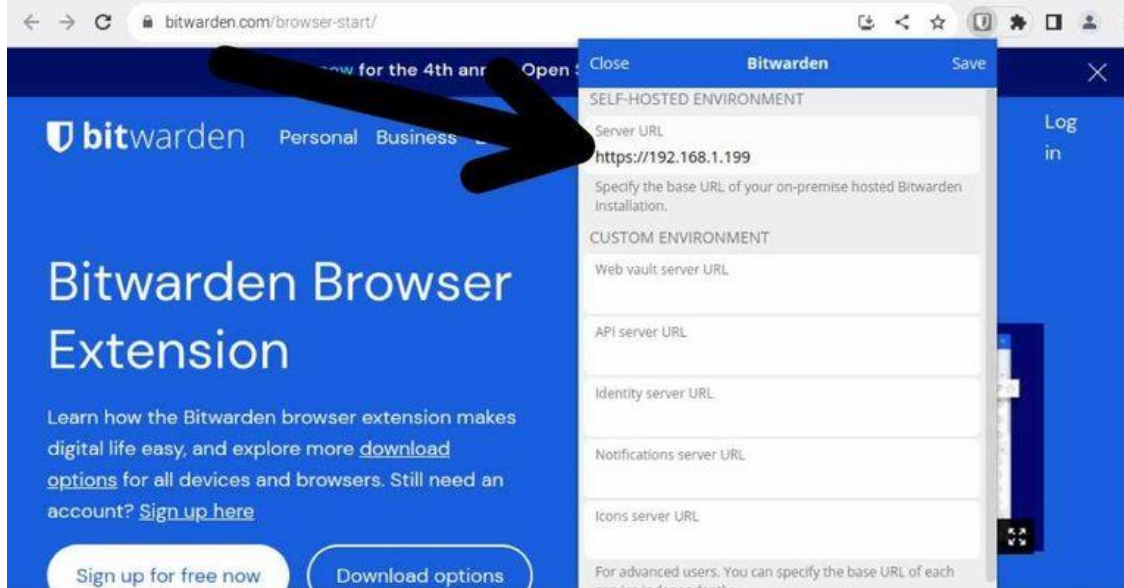
It will also take you directly to the extensions, where you may click the Add to Chrome button.



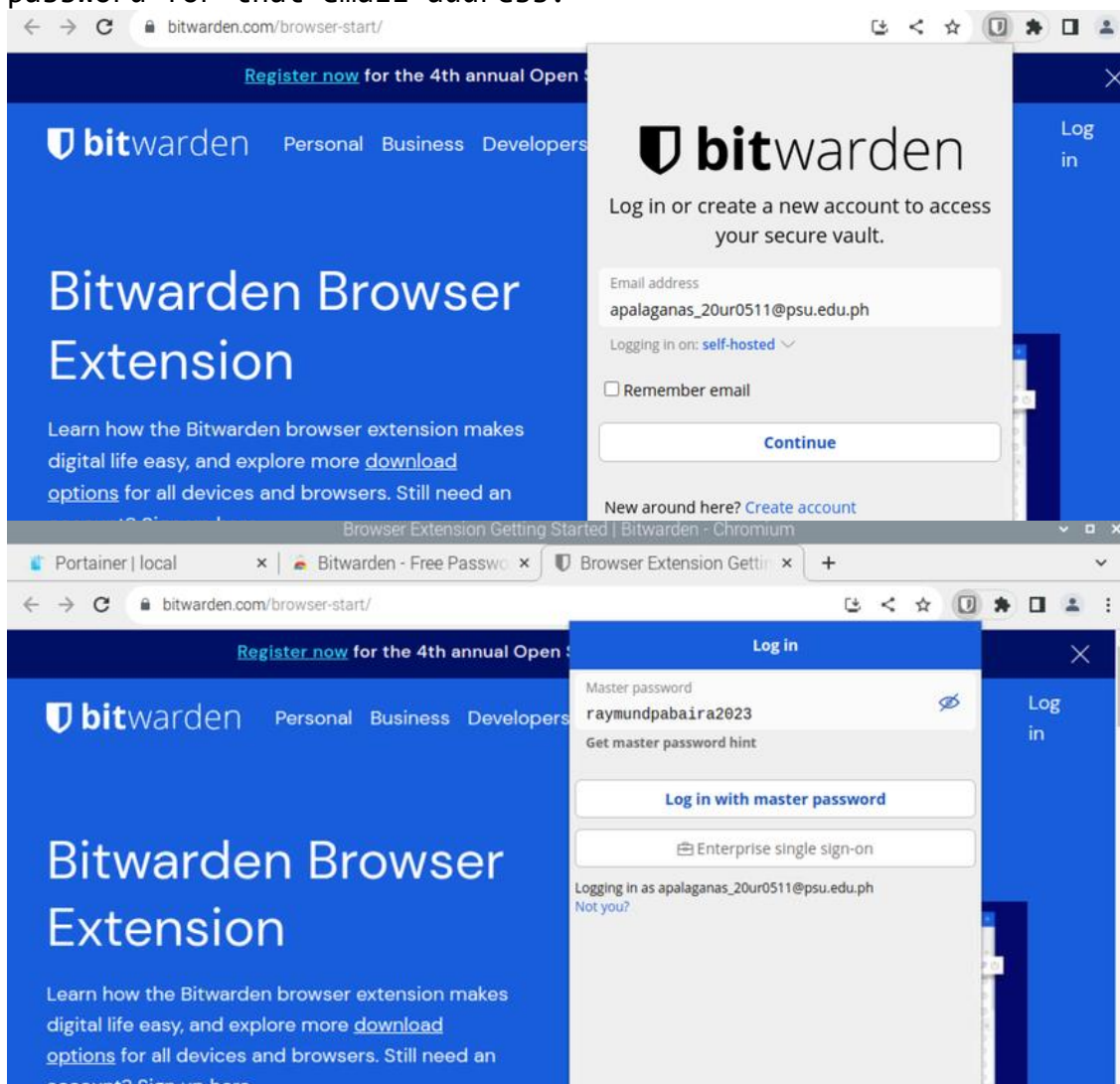
**Step 71.** Next, open the link that has been displayed below, or click the extension part of the browser, and it will show the BitWarden's logo; you noticed that it is highlighted in the Logging in section; click that, then change it to self-hosted, which shows the following pictures below, and after following the steps, click continue.



**Step 72.** Then, as shown in the images below, enter the server URL of the self-hosted environment link and click save.

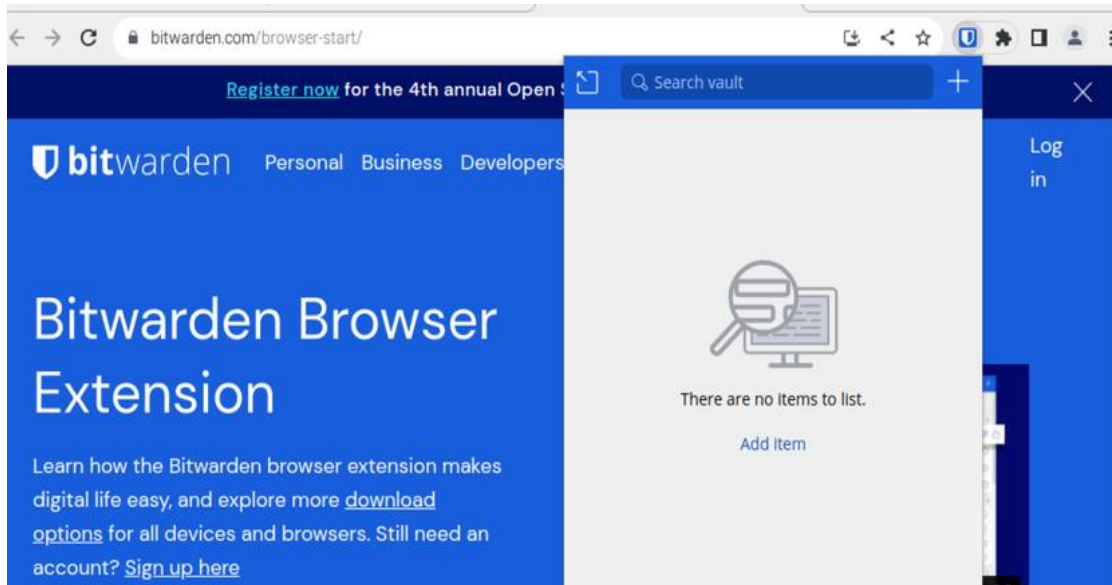


**Step 73.** Return to the browser and type `bitwarden.com/browser-start/` again. Then, provide the registered account's email address as well as the master password for that email address.

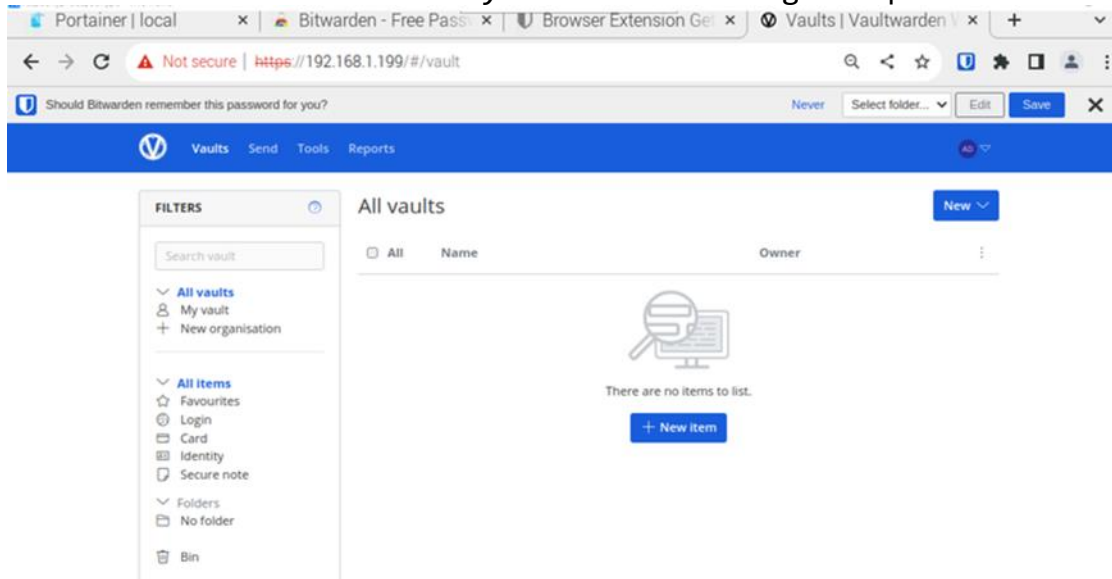




**Step 74.** Following that, you will be led to the identical first image that is seen below on the bitwarden website.



Then, in a new tab, enter the server URL link, and it will display the self-hosted bitwarden that you created during the procedure.<sup>[2]</sup>



That's all; you now have your own self-hosted password manager (BitWarden).

**SUMMARY / CONCLUSION**

Building a password manager like Bitwarden on a Raspberry Pi can offer several advantages, including increased privacy, control over your data, and the satisfaction of building a DIY solution. Throughout the process, you've learned valuable skills in setting up and configuring software on a Raspberry Pi, managing security protocols, and gaining a deeper understanding of password management systems.

By using open-source software like Bitwarden, you ensure transparency and security in managing your sensitive information. Additionally, hosting your password manager on a Raspberry Pi gives you full control over your data, reducing reliance on third-party services and minimizing the risk of data breaches or leaks.

In conclusion, the project to build a password manager using Bitwarden on a Raspberry Pi is not only a practical exercise but also an empowering journey towards digital security and self-sufficiency. It equips you with the knowledge and tools to protect your online identity and personal information effectively, reinforcing the importance of privacy and security in today's digital landscape.

**REFERENCES**

- [1] Emmet. (2023, April 6). Self hosting Bitwarden on the Raspberry Pi. Pi My Life Up.  
[https://pimylifeup.com/raspberry-pi-bitwarden/?fbclid=IwAR3aael\\_XhIUxllfZwjK7lV-gG\\_bY7GvTrMUMFpiQQsT5R-JUPSss3hvJ3I#preparebitwarden](https://pimylifeup.com/raspberry-pi-bitwarden/?fbclid=IwAR3aael_XhIUxllfZwjK7lV-gG_bY7GvTrMUMFpiQQsT5R-JUPSss3hvJ3I#preparebitwarden)
- [2] The password manager trusted by millions | Bitwarden. (n.d.).  
Bitwarden. <https://bitwarden.com/>

