Install Prestashop

https://www.youtube.com/embed/Fe0mHxMG71Q

Prestashop is a self hosted, open source shopping / eCommerce site you can run yourself. It has a lot of great tooling, and is really powerful, but also easy enough to get used to using fairly quickly.

Today, we'll install Prestashop using Docker and Docker Compose, and I'll show you a few things that will help you get the system up and running in a more useful way than it is "out of the box".

What You'll Need

- a server or VM to run your application on (can be your local desktop if you want)
- Docker and Docker compose installed on the server or VM
- (optional, but needed for any public store) A fully Qualified Domain Name (FQDN)
- (optional) Reverse Proxy
- About 25 minutes of your time

Installing Docker CE and Docker Compose

You can install Docker-CE (Community Edition) and Docker Compose with a single line command on most Linux distributions.

```
curl http://get.docker.com | sh
```

You'll be prompted for your super user password, and once supplied, Docker-CE and Docker Compose will be installed.

Next, add your user to the 'docker' group with the command:

```
sudo usermod -aG docker <username>
```

Once done, simply log out and back in to the system, and you'll be able to run `docker` and `docker compose` commands without needing `sudo` in front of them.

Installing Prestashop with Docker

Prestashop requires a few things to be setup. Their documentation provides a very full explanation of what you need, but I had to learn the hard way that the first docker compose example they provide on the page is not the full compose file you'll want in the end. So, if you refer to their docs make sure to scroll down, and read each section. THey build on their initial example as the page goes on. The document I'm providing should have you setup the way you need to be.

First, let's create a folder structure to support our install:

`mkdir -p docker/prestashop`

Next, we'll move into the new directory with

`cd docker/prestashop`

Now, we need to create a new file called "compose.yaml" in this directory.

`nano compose.yaml`

Copy the Docker Compose file below, and paste it into the file you just created.

```
---
services:
  mysql:
    container_name: presta-mysql
    image: mysql:5.7
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: <a-l0n6-57r0NG-p4ssworD>
      MYSQL_DATABASE: prestashop
    volumes:
      - ./dbdata:/var/lib/mysql
    networks:
      - prestashop_network
  prestashop:
    container_name: prestashop
    image: prestashop/prestashop:latest
    restart: unless-stopped
    depends_on:
      - mysql
    ports:
      - 8080:80
```

```
    environment:
      DB_SERVER: presta-mysql
      DB_NAME: prestashop
      DB_USER: root
      DB_PASSWD: <a-l0n6-57r0NG-p4ssworD>
      PS_INSTALL_AUTO: 0
      PS_DOMAIN: <shop.mygreatdomain.org>
    volumes:
      - ./psdata:/var/www/html
    networks:
      - prestashop_network
networks:
    prestashop_network:
```

In the file above, you will want to change the values surrounded with less than "<" and greater than ">" signs. Make sure to create a long, strong password for your MySQL instance, and make sure the password matches in both sections of the compose file.

For the PS_DOMAIN, if you have created a domain / sub-domain for the application to run at, fill that in, otherwise enter the IP address where you'll access that application. In my case I entered "shop.opensourceisawesome.com".

Finally, for the port mapping, if you already have port 8080 in use on the host machine you are setting this up on, you can change the left side of the mapping to a port number that isn't in use already (e.g. `- 8082:80`).

Pull the Images

Now, we can pull the images for our Prestashop containers to run from with

`docker compose pull`

Allow this to run, and pull down the mysql and prestashop images. Once complete, we can bring up our containers with the commands:

`docker compose up -d && docker compose logs -f`

These two commands will bring up the containers and run them detached (in the background), and will display the live logs as they are being created and started (thus the `-f` for follow the logs).

If you see no errors in the logs while the containers are starting up, you should be able to go to your IP address or URL to start the configuration wizard for Prestashop.

Configuring Our New Site

First Run Wizard

1. Set your preferred Primary Language.
2. Read and Accept their Open Source License.
3. Enter your store name, type of sales, Country, store Timezone, and enable SSL set to 'Yes'. Also enter your First and Last name, an email address for you as the admin of the site, and enter a password for your admin user, and re-enter it to confirm it.
4. I suggest allowing the system to install the demo products, and all available modules to begin. You can always remove them when you're ready to start setting up your own.
5. Enter your database information. If you followed my 'compose.yaml' file above, you'll enter the following

Database server: presta-mysql
Database name: prestashop
Database login: root
Database password: <whatever you set as your database password>
Table Prefix: ps_
Drop existing: <checked>

If you changed any of the values in step 5, then you should adjust the above values accordingly.

Test your connection. If you get a green checkmark, you're set. Now click 'Next', and your system will be setup. Be patient while this process completes.

Remove the Install Directory

After configuration is complete, it's important to remove the 'Install' directory, as this could be used by a nefarious person to re-run the install configuration and mess up your site, or take it over. To do this, we can run the command:

```
docker compose exec -i prestashop rm -rf install
```

Optionally, you can go into the `psdata` folder inside your `docker/prestashop` folder, and remove the "install" directory from there with

```
sudo rm -rf ./install
```

as well.

Change the Back End Folder Name

As a secondary security precaution, it's useful to also change the back-end folder name.

Prestashop is built with a front-end (intended for shoppers in your site), and a back-end intended for you and any admin users to setup the site, add new products, setup sales, etc. You don't want just anyone getting to the back-end too easily, so you can adjust the path to the back-end by changing the folder name it runs from.

Use this command to change the folder name:

```
docker exec -i mv admin admin_1357
```

You can change the number "1357" to any number you like. it's just something to make it harder for someone to just guess the admin path.

Once changed, you'll get to the back-end by going to your domain / IP and the path 'admin_1357' (or the number you set).

I go to https://shop.opensourceisawesome.com/admin_1357

The front-end will remain at the main domain name.

Payment Options in the Base System

So, the smelly turd in the water here... I was really disappointed to find that there were only three payment options in the base system.

1. by Check
2. by Wire Transfer (because everyone sets this up for a single shop...ugh)
3. Cash on Delivery...yep...

Anyways, I did a lot of digging to find out that there are other payment modules you can add on, but it's definitely a bit of a process.

Now, if you're like me, you may be asking yourself...if it's e-commerce why in the world wouldn't they just bake in things like Stripe, Square, PayPal, and so on? Great question! If anyone gets the answer, let me know.

Anyway, we need to go get the modules we want and add them. Good news! There is a module market place! Yay! But, it's a module that is also not included in the base system! OMG! Why?!!!!

Deep breath! If you have found your way to my channel because you too found only the vague references to adding modules for other payment options, and references stating to "just search the

marketplace...", then you aren't alone.

We can get the module from Prestashop for the Marketplace, but, of course, we have to sign up to get it. Yes, if I had any hair, it would be getting all pulled out by now.

As of this writing, you can get to this module's downlaod page at Prestashop Marketplace in Your Back Office. If it's changed, apologies, but hopefully you'll find it again.

Now, you'll notice that it says "Free Download", but if you have to sign up for something, the it wasn't really "free" now was it? Anyway, just go sign up, then verify, login, and so on. Go back to that page linked above, and get this zip file add-on.

Now, in your Prestashop Back Office, go to the left menu, and click on 'Modules' under the 'Improve' heading. Next, click on 'Module Manager'. On this page, you can now click on the 'Upload a Module' button in the upper right.

Select the .zip file you just downloaded from your file system, or drag and drop the .zip file onto the upload view. Click 'Submit'.

The module will be installed.

This does 2 things:

1. It installs the market place which will make it easier to add other modules to your Prestashop install.
2. It also creates a user with admin privileges. This user is needed in order to add other modules and install them from the shop. So it's normal for this user to be there.

After install completes, you can refresh your page, and you should now see a new option under 'Improve > Modules' called 'Marketplace'. You can click on it, and you'll find you can search for modules, or browse them.

Adding a New Option for Payments

Search for 'PayPal', and select the PayPal module you like the best. I selected the 'PayPal Official' add-on module, and clicked the 'Install' button on it's page. Within a few seconds I got a toast message saying it had been installed.

I navigated to 'Improve > Payments > Payment Methods' and there was a new option for PayPal! Oh glorious day! Something that should have been there by default is finally installed and available! Whew!

Well, I hope this helped you out. If so, make sure to let me know about your online shop, what you're selling, and how it's going!

Revision #2
Created 2025-05-13 14:01:52 UTC by Brian McGonagill
Updated 2025-05-13 16:04:52 UTC by Brian McGonagill