Open Source Invoicing Software

- Solid Invoice
  - Installing Solid Invoice
- Invoice Ninja
  - Install Invoice Ninja with Docker
- InvoiceShelf
  - Install and Configure Invoice Shelf

Solid Invoice

Installing Solid Invoice

https://www.youtube.com/embed/trviH2WRJ1w

I've been asked several times to install InvoiceNinja, and no matter how I tried, it never seemed to work, but gave errors in the log every single time. So, I started looking for other open source invoicing solutions. I tried a bunch of them. Invoice Lion, Crater, Invoice Ninja, Invoice Plane, and several others, and none of them would install correctly, or were intuitive enough to get running, or had decent documentation.

Then, I found Solid Invoice. It took a little bit of effort to create a Docker Compose file that would setup both the DB and the Solid Invoice image together, but minimal effort at best. So, today, we are going to go through installing Solid Invoice, an open source, self hosted invoicing system.

What you'll need

- A server to run it on (this can be a VPS like on Digital Ocean, or a personal computer).
- Docker-CE (Community Edition) installed
- Docker-Compose installed
- A Domain / Sub-domain for the solid invoice install with an A Record pointed at your server's public IP address (optional, but recommended)
- NGinX Proxy Manager (optional, but recommended)
- About 30 minutes to an hour of time.

Installing Docker-CE and Docker-Compose

I have several script sfor various Linux distros on my github page. If you'd like to use one of these scripts, I highly recommend it.

Just do a git pull, or click on the version you want, and highlight the entire script code.

Now on your server where you'll run Dolibarr, create a new file called `docker-install.sh`.

`nano docker-install.sh`

Now pasted the contents into this file (for Linux you can use CTRL + Shift + V to paste).

Save the file with CTRL + O, then Enter, and exit nano with CTRL + X.

Now change the file permissions to make it executable.

`chmod +x docker-install.sh`

Now you can run the script with the command

`./docker-install.sh`

Allow the script to finish. Once complete, you can log out and back in, or reboot the server. This will allow your user to run docker commands without having to use the `sudo` keyword each time.

Install NGinX Proxy Manager

I have covered this on several other videos and posts, so I'll link the post here if you need to install this application.

Install Solid Invoice using Docker-CE and Docker-Compose

On your server create a new folder called "solidinvoice", then move into that folder:

`mkdir solidinvoice`

`cd solidinvoice`

Now create a new file called "docker-compose.yml" and open it for editing:

`nano docker-compose.yml`

Now, copy the script below, and paste it into the editor. If you are editing in nano as I suggest, then you can paste with the hotkey combination CTRL + Shift + V.

Now, save that file with CTRL + O, then Enter, then CTRL + X.

Next, create the folder "db_data" inside your solidinvoice folder:

`mkdir db_data`

Finally, you can run `docker-compose up -d` to start pulling down the Solid Invoice, and MySQL images and get them up and running.

Once they've completed, and you see the "done" message for each in the terminal, give it a few minutes, then see if you can reach your site at it's IP address, or domain / subdomain name and port 8000.

For me, I used:

`http://invoice.opensourceisawesome.com:8000`

Test it out at `https://invoice.opensourceisawesome.com` , and use the following credentials

Once I saw that it loaded the system checks page, I moved on to setup my NGinX Proxy Manager entry and my Lets Encrypt certificates for SSL.

Follow along in the video for setting up a domain, NGinX Proxy Manager host entry, and LetsEncrypt.

Also, check out the video for the basic setup of Solid Invoice.

Support my Efforts at Patreon

Support my Channel and ongoing efforts through Patreon:
https://www.patreon.com/bePatron?u=234177

Invoice Ninja

Install Invoice Ninja with Docker

https://www.youtube.com/embed/Uv11VPP4XP4

Invoice Ninja is a power open source offering that has been around for years now. It's only gotten better and better as time has gone by. I have been attempting to install it for a while now, but for whatever reason, just kept hitting little blocks that I couldn't quite get around. I have now gotten to the point that the install went very smoothly, and I wanted to share my steps and setup with all of you.

What You'll Need

- A server or machine to install Invoice Ninja on.
- Docker and Docker Compose installed on that machine
- (optional) A Domain Name you own and can create A or CNAME records for.
- (optional) a reverse proxy (I use NGinX Proxy Manager, but you can use any you like)
- (optional) SMTP email server for sending invoices through
- About 30 minutes of your time.

Installation of Docker and Docker Compose via a Simple Script

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

```
wget https://gitlab.com/bmcgonag/docker_installs/-/raw/main/install_docker_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install_docker_nproxyman.sh
```

and then run the script with the command:

```
./install_docker_nproxyman.sh
```

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

In this case, you need to install Docker-CE and Docker Compose at a minimum, but you may also want to install NGinX Proxy Manager if you don't already have a reverse proxy on your network.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of sudo in front of them.

The Domain Name

Domain names are a huge topic all their own. The short of it is, you can purchase a domain name fairly cheap these days. If you are invoicing people, you should definitely be doing it from a domain name that identifies your business well. Spend a few bucks and get one from anyone who'll give you access to set an A record. The A Record is a special DNS record that maps your domain (e.g. mygreatbusiness.com) or sub-domain (e.g. billing.mygreatbusiness.com) to your public IP address. Thus when someone tries to go to https://mygreatbusiness.com or https://billing.mygreatbusiness.com they get directed properly to your public IP address.

Installing Invoice Ninja

To begin with, make sure to check out the official documentation at the Invoice Ninja Dockerfiles site if you have any issues from this article. As these articles and videos age, the steps can sometimes be replaced with other processes that need to be done, or potentially can have processes removed that are no longer needed.

First, we need to make sure you have git, curl, and wget installed on your system. Use your systems package manager to install these.

For Debian / Ubuntu distros you can use:

```
sudo apt install git curl wget -y
```

For Fedora / Red Hat / CentOS, etc distros you can use:

```
sudo dnf install git curl wget -y
```

Once you have those installed, we'll pull down the git repository for Invoice Ninja Dockerfiles. I suggest putting your docker applications into a parent folder name "docker". To create that folder, use this command:

`mkdir docker`

Now, move into your 'docker' folder with

`cd docker`

and clone the repository:

```
git clone https://github.com/invoiceninja/dockerfiles.git
```

Once cloned, you'll have a new directory called 'dockerfiles'. I personally don't like this name, so let's change it to something more identifiable with

`mv dockerfiles invoiceninja`

There, now we have a new folder called 'invoiceninja'. Let's move into it.

`cd invoiceninja`

In this folder, you'll find several files. We are interested in the files name 'env' and 'docker-compose.yml'. We'll be editing the contents of these files in order to get Invoice Ninja setup for our use. First, however, let's create a backup copy of each of the original files. This gives us the opportunity to make a mistake, and still have something we can revert back to if needed.

`cp env env.bak`

`cp docker-compose.yml docker-compose.yml.bak`

The two 'cp' commands copy the existing file to the backup file with the .bak extension. Now we can start editing the two original files as we see fit.

First, let's edit the 'env' file.

`nano env`

Feel free to make the changes manually based on my example file below, or delete all the contents from your file, then copy and paste my example below, and just change the items I have identified.

```
# IN application vars
APP_URL=http://in.localhost:8003    # <-- change this to your domain name, or the ip address
of your server
APP_KEY=<insert your generated key in here>  # <-- we'll generate this key with a command line
command
APP_DEBUG=false
REQUIRE_HTTPS=false
PHANTOMJS_PDF_GENERATION=false
PDF_GENERATOR=snappdf
```

```
TRUSTED_PROXIES='*'

QUEUE_CONNECTION=database

# DB connection
DB_HOST=db
DB_PORT=3306
DB_DATABASE=ninja
DB_USERNAME=ninja  # <-- you should change this to a user you like
DB_PASSWORD=ninja  # <-- you should make this a very long strong password

# Create initial user
# these credentials are what you'll use to login to the web interface
# DO NOT LEAVE THIS EMPTY!!!
IN_USER_EMAIL=      # <-- put the email of your admin user here
IN_PASSWORD=        # <-- put a very long strong password here for your login

# Mail options
# If you want email of invoices to work from the web interface
# you need to fill out this section. It's important that you
# know what details need to go here.
MAIL_MAILER=log
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS='user@example.com'
MAIL_FROM_NAME='Self Hosted User'

# MySQL
MYSQL_ROOT_PASSWORD=ninjaAdm1nPassword  # <-- You need to change this to a really long strong
password
MYSQL_USER=ninja # <-- this needs to match the mysql user you created in the section near the
top
MYSQL_PASSWORD=ninja # <-- this needs to match the strong password from the mysql section near
the top
MYSQL_DATABASE=ninja

# V4 env vars
```

```
# DB_STRICT=false
# APP_CIPHER=AES-256-CBC
```

In the example above I have added comments (anything after a "#" symbol) so you know what you need to change.

Once you've changed it save with CTRL + O, then press Enter to confirm, and exit the nano editor with CTRL + X.

Now, let's quickly create that APP_KEY and get it put into the 'env' file as well. For this we'll be running a docker command. The command will pull down the Invoice Ninja image we need later, start a container from the image, run a special command inside the temporary container, and spit out an APP_KEY. We'll then copy that key and paste it into our 'env' file.

Run this command to create your key:

```
docker run --rm -it invoiceninja/invoiceninja php artisan key:generate --show
```

You'll get some output on the screen. This is normal, so be patient while everything works. Once it's done you should see a line toward the bottom of the output that starts with "base64:". You need to copy that entire line (including the 'base64:' part, and paste it into the 'env' file for the APP_KEY value. So, copy it, then do

`nano env`

to open the file, and remove the text inside of the angle brackets '<' & '>', as well as the angle brackets themselves. When you're done pasting your line in the 'env file should look something like this.

`APP_KEY=base64:RGK672GY56Df#$SR3998JHKYVBE45=`

Now, save the file with CTRL + O, then press Enter to confirm, and exit the nano editor with CTRL + X.

Next, we need to open our docker-compose.yml file, and make one change. To do this, enter

`nano docker-compose.yml`

Now, find the 'ports' where it read `- "80:80"`.

We want to change the port on the left side of the colon to some other port than 80. This is a very common port for webservers, and shouldn't be used. Let's change it to port 8035. So the mapping should now look like `- "8035:80"`.

This maps port 8035 on the host machine to port 80 inside the docker container, and allows us to access the Invoice Ninja web user interface on port 8035.

Save your file, with CTRL + O, then press Enter to confirm, and exit the nano editor with CTRL + X.

Before we start up our containers, we need to do a couple of more things. The folders where data and configurations will be stored need to have certain ownership (specifically by the www-data user) and permissions. So let's run the following two commands to get that setup.

```
chmod 755 docker/app/public
```

```
sudo chown -R 1500:1500 docker/app
```

Now, we're ready to start our server running. Let's use the "docker compose" command to bring up our services, and then log out the status as it's starting to the terminal.

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

Now, sit back while the images are pulled down, and the containers are started. You may see a couple of MySQL connection errors, but the system should keep re-trying until it connects. Once the output stops flowing for about a minute, you should see an INFO message with some text about things being successful.

At this point, you should be ready to go. Type CTRL + C to stop the log output in the terminal, and move over to your favorite modern browser. Enter the local IP address of your server host, and the port 8035. This should open up to the Invoice Ninja login page.

I went to http://192.168.10.50:8035

and was greeted by the login page. If you see the login page, and you only intend to run Invoice Ninja on your local network, then you can stop here and start logging in and setting up Invoice Ninja for your use. If, however you're running this for a buinsess, and you want to have a domain attached to this install, then keep moving forward with me.

Setting up our Reverse Proxy

A Proxy is a machine you have to go through to reach the internet from inside of certain networks. A revese proxy is a machine that outside devices go through in order to reach services running inside your private network. There needs to be a port foward happening on your Firewall / Router so that ports 80 adn 443 point to the host where your reverse proxy is running. That's a bit beyond the scope of this tutorial, but there are tons of resources on the internet to help you with that process on your router model iif needed.

Next, we'll login to our reverse proxy, and create a new Proxy Host. In NGinx Proxy Manager (NPM from this point on), we need to enter the domain or sub-domain name we setup with an A record in our DNS. Press 'Tab' to make it a chip, then move to the IP address field, and enter the private IP address of your host machine. Now, move to the port field, and enter the port you mapped in your docker compose file (in this case I used 8035). Enable the options for 'Block Common Explits' and 'Websocket support', then click 'Save'.

Now test the domain in your browser to make sure you are forwarded to your Invoice Ninja login page. If so, AWESome! We are moving along nicely. If not, you'll want to start troubleshooting. Make sure you setup your Domain Name, DNS A Record, Firewall port forwarding all correctly. Then start trying to ping to see where the requests go, and so on. Hopefully everything is up and running for you though.

Now, we need to get an SSL certificate so we can reach our site with full encryption. Go back to NPM, and click the 3-dot icon at the right end of your Invoice Ninja entry. Select 'Edit' from the pop-up menu, and move to the SSL tab in the window that opens. In the 'None' drop-down, select 'Request a New Certificate', then enable 'Force SSL', HTTP/2 Support', and both HSTS options. Make sure you email is filled in, then enable to accept the LetsEncrypt terms of service. Click 'Save'. The spinner will spin while LetsEncrypt challenges your domain to make sure it can reach it on port 80, and if everything is working properly, the pop-up will just close on it's own with no errors. You should see the LetsEncrypt indicator on the Invoice Ninja row now. You can again go to your invoice ninja domain and you should now be accessing it using https and full encryptiong. You can now login and begin the setup of your Invoice Ninja install.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/awesomeopensource

InvoiceShelf

Replace / Rebirth of Crater Invoice

InvoiceShelf

Install and Configure Invoice Shelf

https://www.youtube.com/embed/GtFs0yJeFUU

A few years back I showed you how to setup and install Crater Invoice. It was a clean, simple, and powerful little invoicing software. After a while Crater sadly went into and unsupported state. I have checked back in occasionally hoping to see it revived by someone, and that persistence paid off. I saw a new note stating the project had been forked, and was being updated over at InvoiceShelf. Mark one in the win column for open source!

Now, the is very exciting for me. I run a small business on the side, and have been using InvoiceNinja for about a year now due to the lack of continued dwork on Crater, but I honestly loved Crater. It was exactly what I needed, and Invoice Ninja is a bit overkill in my book.

That said, if you're looking for a clean, simple, but elegant and powerful invoicing option, then InvoiceShelf may just be for you.

What you'll need

- A server, VM, or LXC, LXD, Incus Container to run the software on
- Docker and Docker Compose installed on the machine
- (optional) A domain or sub-domain if you want your clients to also be able to access their invoices and quotes
- (optional) A Reverse Proxy
- About 15 minutes of your time.

Install Docker and Docker compose

First, we need to install Docker and Docker Compose on our server. In most cases for Linux based servers, you can do this with a single command. Open your terminal, and enter the following:

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

When prompted, enter your super user password. This will attempt to install both Docker-CE (Community Edition) and Docker Compose for you.

If you are not using one fo the many distributions supported by this script, you'll need to look up how to install docker and docker compose on your system. In particular Windows supports docker both natively through the Docker Desktop app, as well as via WSL2 (the Windows Subsystem for Linux).

Add your user to the 'docker' group with:

`sudo usermod -aG docker $USER`

then log out and back into the system.

Setup our Domain / Subdomain Name and DNS A Record

In order to have a domain name (like invoice.mygreatdomain.com) take us to our install of InvoiceShelf, we need to

1. Own the domain we want to use (e.g. mygreatdomain.com)
2. Have access to create new DNS records for the domain (you can usually do this through your domain registrar's control panel)
3. Have a public IP address either for the server we are running our software on, that leads to our ISP given internet connection, or to a VPS (small server in the cloud) where we run a VPN (a private network that allows us to connect the VPS to our home / small business network).
4. Access for traffic to be allowed on ports 80 and 443 to that public IP address.

IF you have all of the above, then let's get your A Record setup. There are a few ways to do this. If you are only wanting to run InvoiceShelf and no other services or web applications from your Public IP, then you can setup a single A-Record. In your DNS settings on your registrar, you need to create a new DNS record, and select the Type as 'A'. Then enter the subdomain for your InvoiceShelf to be accessed from.

A sub-domain is the little part before the fully qualified domain name.

```
invoice.                    mygreatdomain.com
  ^                               ^

sub-domain                   fully qualified domain name (FQDN)
```

Next, enter in your public IPv4 address for the server / network where you'll be hosting the server.

Why Do I Need a Reverse Proxy?

A reverse proxy can provide a ton of benefits.

1. It can act as the traffic director from outside your LAN to machines inside your LAN.

2. It can be the only machine on your LAN exposed directly to the internet if needed.
3. You can setup a Reverse Proxy to help direct traffic to multiple services all sharing the same Public IPv4 Address.

If your host server is not directly available via a public IPv4 address to the web, then you'll likely want a reverse proxy.

I use the NGinX Proxy Manager reverse proxy, but feel free to use any you prefer. I have a video on NGinX Proxy Manager here, if you'd like to set this up as well.

In this case, we'll setup our sub-domain using our reverse proxy. We'll enter the subdomain name, give it the local (private) IPv4 address and port number we'll be pointing to for this particular application. In my case I'm going to setup InvoiceShelf on a machine with the private IPv4 address of 192.168.10.144, and it's going to run on port 8215 (a random number I chose above 8000, but not already in use by another service).

We'll tell our reverse proxy to block common exploits, and support websockts. We'll come back in a bit to get an SSL certificate for this entry, but for now, let's setup our InvoiceShelf service.

Install InvoiceShelf

On the host VM / Server you want to instaall InvoiceShelf on, let's create a directory structure to hold this application.

```
mkdir -p docker
```

Now we'll clone the InvoiceShelf repository, and pull down the files we need in order to run InvoiceShelf.

```
git clone https://github.com/InvoiceShelf/docker
```

This will create a new folder for us called 'docker'. Let's rename this to invoiceshelf to avoid confusion later.

```
mv docker invoiceshelf
```

Now let's move into the invoiceshelf folder, and configure the settings we need.

```
cd invoiceshelf
```

```
ls
```

First, let's choose which version we want to use. You'll see several files in this directory that start with 'docker-compose' and end with '.yml'. This is just different versions and methods of running the same application. IN my case I chose to run the version with a Postgres Database backend. You can choose any one you like, and the steps for setup should be similar to mine.

Let's rename the 'docker-compose.pgsql.yml' to be just 'compose.yaml'. This allows docker-compose to know which file to run automatically, and shortens the startup command for us later.

```
mv docker-compose.pgsql.yml compose.yaml
```

Now we'll open our 'compose.yaml' file to edit it.

```
nano compose.yaml
```

In this file we need to change a few options very quickly.

1. In the section called 'invoiceshelf_db' we need to change the POSTGRES_PASSWORD value to something long, strong, and un-guessable. Use upper and lower case letters, and numbers only. Don't use symbols or spaces, as Postgres doesn't like that.
2. Next, in the 'volumes' sub-section, add a`./` to the front of 'postgres', sot he line looke like this:

```
  - ./postgres:/var/lib/postgressql/data
```

3. After this, we'll move down to the 'invoiceshelf' section. If you have port 90 in use, on your host server / system, then you can change the port number on the left side of the colon ':', but not the number on the right.

If you change that port, remember what you change it to.

4. Next, under the sub-section for 'environment', set your timezone for the two variables that have TZ in them. I am in America/Chicago, but you'll want to set your proper timezone.

5. For the DB_PASSWORD variable, make sure it matches the password you set in the first section. Copy / Paste that password from the first section to here to ensure they match exactly.

6. If you want to setup email, you need to uncomment the lines starting with a hashtag '#' sign, and then fill in the values for MAIL on each line. You need to know your SMTP values for this.

7. Finally change the port number at the end of the APP_URL and SANCTUM_STATEFUL_DOMAINS if you changed the port number in this section from 90 to a different port number. Make sure to put the port number you entered on these lines.

8. Vinally remove the lines at the bottom including the one that sys 'Volumes'.

Now you can save with CTRL + O, then press Enter to confirm, and press CTRL + X to exit the nano editor.

Pull the InvoiceShelf Images

Now let's pull down our images for Invoice Shelf.

```
docker compose pull
```

You should see the images being pulled down from Docker Hub. If you get an error, make sure you have the spacing correct, and make sure you have all of the values correct. The errors can tell you what's wrong.

Start our Containers

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

You'll see the containers start up, and then you'll see the logs being output to the terminal. You're just watching for anything that looks liek an obvious error as it starts up.

Once it's all started, you can go to your favorite modern browser, and go to the private IPv4 address of your VM or Server, and then if you used a port other than 80, make sure to put a colon ':' then the port number after.

If you left it with Port 90, using my IP as an example we'd put "http://192.168.10.144:90".

If all has gone well, you should see the Setup Wizard for InvoiceShelf.

If you don't intend to have your clients access their invoices through this system, nor to access it from outside your network, you're done other than walking through the wizard to get things setup. You can access and use this system from this IP and port.

Finishing our Reverse Proxy Entry

If, in the start, you weren't sure what to do for the port number, you can go back and set that port in your reverse proxy. YOu can then use the SSL features of you rproxy to request a new certificate, then get it all setup for SSL encrypted access. This is important if you'll be using this outside of your private network (on the internet). You always want encrypted traffic when using anything over the internet, and it's free to sdo these days, so there's no reason not to do it.

Now try and access your site by the subdomain / domain name you wanted (for instance: https://invoice.mygreatdomain.com). You will hopefully again see the setup wizard in your browser, and now it's time to go through the wizard.

Most screens are self explanatory, however, when you get to the screen Site URL and Database, it's important you do this with your domain already setup if you are using a domain.

The Wizard defaults the values for a MySQL / MariaDB setup, so if you used that version, you'll be good other than on echange. If, however, you did Postgres like me, then we need to change a few values here.

1. Database Connection = pgsql
2. Database port = 5432
3. Database Name, Username, and Password you can get from yoru compose.yaml file. Copy / Paste is your friend.

4. Database Host = invoiceshelf_db

That last one is very important. You need to set this no matter which database backend you chose. If you don't, it will likely fail when trying to connect. Even if it makes it to a connection this time, it could fail in the future, so be sure to set it the way I have in number 4 above.

Now you can continue through the Setup Wizard.

Anything you set in the wizard can be changed in the application settings later, except for the currency type. **Make sure you choose your primary currency type properly to start with, as this cannot be changed later.**

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/awesomeopensource

Buy me a Beer / Coffee:

https://paypal.me/BrianMcGonagill