Crater Invoices

Open Source, Self Hosted Invoicing, Estimating, and Quoting solution

- Crater Invoice

Crater Invoice

https://www.youtube.com/embed/velKYXN3A_w

Crater is a really nice, well polished, invoicing system that is open source, and can be self hosted. I have struggled a bit to get this running, but with a couple of bug reports from the docker install side, I've gotten it to run.

In this tutorial, I'll show you how to get Crater up and running with little to no issues. And show you the couple of small issues I've come across in my efforts to document the process for you.

What You'll Need

- Docker-CE and Docker-Compose installed on a host where you want to run the Crater software
- (optional) NGinX Proxy Manager (or another reverse proxy you are familiar with) to get a Domain / Sub-domain of your choice pointing to your install (best for those wanting to access crater from outside their network, or provide client portals to their invoices).
- About 20 minutes of your time.

Installation

Installing Docker-CE and Docker-Compose

If you already have Docker and Docker-Compose installed, feel free to skip down to the next section.

You may want to install some pre-requisite softwre as well:

Debian / Ubuntu

```
sudo apt install git curl wget
```

Fedora / Redhat

```
dnf install git curl wget
```

Arch

```
sudo pacman -Sy git curl wget
```

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 permissios 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.

For instance, you may want to answer 'y' to NGinX Proxy Manager, and Portainer-CE if you don't already use these in your system.

At some point, you'll 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.

Installing Crater

Having docker-compose setup and ready, we'll create our folder structure first. If you don't already have a "docker" folder setup, I like to create a parent "docker" folder where I keep all of my other application container folders so I only have to zip up and backup the parent "docker" folder to get a solid backup of my working system.

Create a "crater" folder inside the parent level "docker" folder.

```
mkdir -p docker/crater
```

Move into that folder, then clone the "crater" repository.
```
cd docker/crater
```

```
git clone https://github.com/crater-invoice/crater
```

Now move into the newly cloned "crater" folder. Your full directory path starting with the parent docker folder will look like `docker/crater/crater`

`cd crater/`

Next, we need to copy the example .env file, and modify it for our install.

`cp .env.example .env`

The "." before our filename indicates that it's a hidden file. So if you run just the `ls` command, you won't see these files. Instead you should run the `ls -al` command to see any hidden files / folders.

Open the new .env file in our text editor (nano).

`nano .env`

Inside this file, you'll want to edit any values to match your network, system, or requirements. Let's take a look at the file and it's contents.

```
APP_ENV=production
APP_KEY=base64:ThisKeyShouldBeInPlaceAlreadyFromTheClone=
APP_DEBUG=true
APP_LOG_LEVEL=debug
APP_URL=http://<your.domain.com>

DB_CONNECTION=mysql
DB_HOST=db
DB_PORT=3306
DB_DATABASE=crater
DB_USERNAME=< crater >
DB_PASSWORD=< "crater" >

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_DRIVER=sync
SESSION_DRIVER=cookie
SESSION_LIFETIME=1440

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
```

```
MAIL_HOST=< smtp.domain.com >
MAIL_PORT=< 25 | 465 | 587 >
MAIL_USERNAME=< my@domain.com >
MAIL_PASSWORD=< someLongStrongEma1lPa5sW0rd >
MAIL_ENCRYPTION=< tls >

PUSHER_APP_ID=
PUSHER_KEY=
PUSHER_SECRET=

SANCTUM_STATEFUL_DOMAINS=http://< your.domain.com >
SESSION_DOMAIN=http://< your.domain.com >

TRUSTED_PROXIES="*"

CRON_JOB_AUTH_TOKEN=""
```

In the above file, you should change the following values:

- API_URL
- DB_USERNAME
- DB_PASSWORD
- MAIL_HOST
- MAIL_PORT
- MAIL_USERNAME
- MAIL_PASSWORD
- MAIL_ENCRYPTION
- SANCTUM_STATEFUL_DOMAINS
- SESSION_DOMAIN

In my case, I just used the same domain for API_URL, SANCTUM_STATEFUL_DOMAINS, and SESSION_DOMAIN. Most likely this will be the case for a home / small business user as well.

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

Once we've made the necessary modifications to our .env file, we'll check for any needed modifications to the docker-compose.yml file as well.

`nano docker-compose.yml`

Again, let's look through the code block for items we need / want to modify.

```
version: '3'

services:
  app:
    build:
      args:
        user: crater-user
        uid: 1000
      context: ./
      dockerfile: Dockerfile
    image: crater-php
    restart: unless-stopped
    working_dir: /var/www/
    volumes:
      - ./:/var/www
      - ./docker-compose/php/uploads.ini:/usr/local/etc/php/conf.d/uploads.ini:rw,delegated
    networks:
      - crater
    depends_on:
      - db

  db:
    image: mariadb
    restart: always
    volumes:
      #- db:/var/lib/mysql
      # If you want to persist data on the host, comment the line above this one...
      # and uncomment the line under this one.
      - ./docker-compose/db/data:/var/lib/mysql:rw,delegated
    environment:
      MYSQL_USER: <crater>
      MYSQL_PASSWORD: <crater>
      MYSQL_DATABASE: crater
      MYSQL_ROOT_PASSWORD: <crater>
    ports:
      - '33006:3306'
    networks:
      - crater

  nginx:
```

```
    image: nginx:1.17-alpine
    restart: unless-stopped
    ports:
      - <80>:80
    volumes:
      - ./:/var/www
      - ./docker-compose/nginx:/etc/nginx/conf.d/
    networks:
      - crater

  cron:
    build:
      context: ./
      dockerfile: ./docker-compose/cron.dockerfile
    volumes:
      - ./:/var/www
    networks:
      - crater

volumes:
  db:

networks:
  crater:
    driver: bridge
```

In the above file, you'll want to change the DB_USERNAME, DB_PASSWORD, and DB_ROOT_PASSWORD to something different and more secure than the defaults before you open the system up to the world. Additionally, you'll likely want to change the left side of the port mapping in the "nginx" block to something other than "80". I used 8019. This just keeps you from trying to use a port that is already in use on your host machine. Leave the right side of the port mapping alone. It needs to be "80".

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

Once we've made all needed modifications to the docker-compose.yml file, we can start our images pulling down / building with the command:

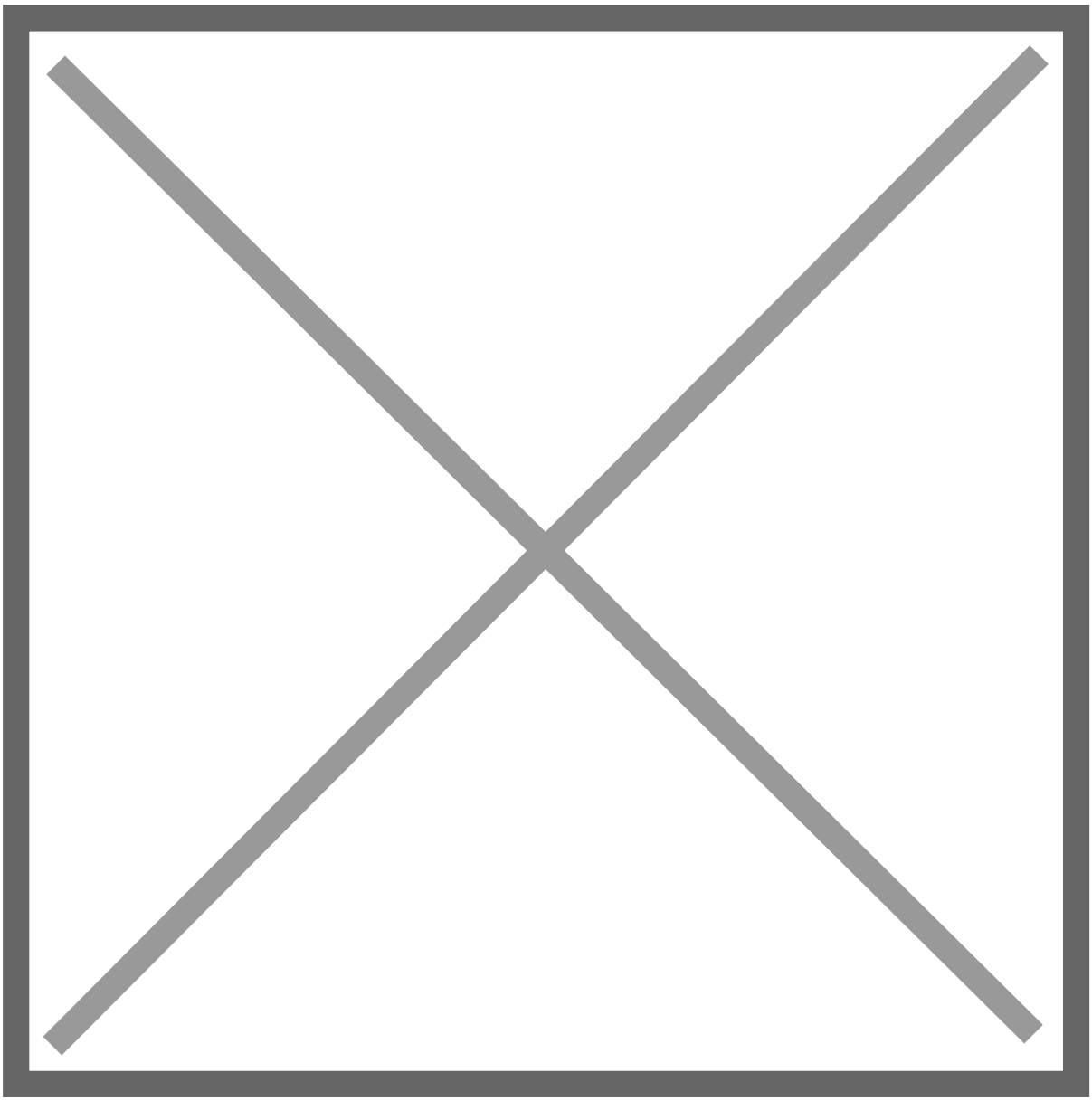
`docker-compose up -d`

This will take a while, as it builds the application and container the first time you run it. You'll want to give it a solid 15 to 20 minutes.

When it's all done running, we'll check that everything is up and running still witht he command:

```
docker-compose ps
```

You should see output similar to this if all is working properly:



Now we need to run a startup / first run script provided in the git repository, under the docker-compose folder.

```
./docker-compose/setup.sh
```

You may see an error about plugins not being allowed. If you do, you'll need to make an addition to the composer.json file found in your crater folder from the git clone.

Open the file with our text editor (nano), and make the addition as stated below.

```
nano composer.json
```

At the end of the composer.json file, change the "config" section to have `"allow-plugins: true"` in it.

```
"config": {
    "optimize-autoloader": true,
    "preferred-install": "dist",
    "sort-packages": true,
    "allow-plugins": true
  },
```

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

Now re-run the "setup.sh" script from the "docker-compose " folder.

`./docker-compose/setup.sh`

Now, you shoudl have a new Crater site up and running.

Reverse Proxy

You may want to setup a reverse proxy to your site as well before you begin through the first run wizard. If you intend to access your invoicing software from outside your network, or via a domain / subdomain, definitely consider the need for a reverse proxy entry.

i use NGinX Proxy Manager for my setup. The steps you need to take are:

1. Get NGinX Proxy Manager installed in docker (my script can do this for you if you used it to install docker and docker-compose earlier.)
2. Port forward 80 and 443 from outside your network (the Internet / WAN) to inside your network to the host ip where you have NGinX Proxy Manager running (or the LAN).
3. Get a domain name from a good provider that gives you access to create A-Records and CNAME records in DNS for your domain.
4. Create a sub-domain of *..<org,com,net,biz,etc>, and point it with an A-record to your pulbic IP address.
5. Now wait the pre-set amount of time for DNS to update (usually about 10 minutes, but sometimes longer), then test to make sure *..<org,com,net,biz,etc> (if your domain is supergreat.org, then you can try test.supergreat.org for example... really any sub-domain should work if you've pointed *.supergreat.org at your public IP) can reach your NGinX Proxy Manager host. You'll get a generic 'Congratulations' page if it's working.
6. Create an NGinX Proxy Manager entry to handle requests for your crater install at the subdomain of your choice. Let's say you want it to be "invoice.supergreat.org". in NPM add a new proxy host, and put that subdomain in the URL field, then press Tab or Enter to save that entry as a chip. In the IP field, you can enter the local docker0 IP of the container for crater NGinX, or you can use localhost. If the crater app is running on a

different machine than your NGinX Proxy Manager, you'll use the private IP of the host where Crater is running. Next enter the port number for your NGinX container. I used 8019 in my example, so if you followed my example, that's the port you want to enter. Now tick the box for Websocket support, and Block common exploits, then Save.

7. Check that your Crater site opens, by clicking the URL in the NPM listing for proxy hosts. If it does, let's edit the entry, and make it secure with SSL. Click the 3-dots at the right side of the entry, and select 'Edit' from the pop-up menu.

   Go to the 'SSL' tab, and select the drop-down that says 'None'. Now select 'Request a New Certificate' from the drop down, and then select the 'Force SSL' tick box, and the 'HTTP/2' tick box. Tick the box to accept the TOS of LetsEncrypt, and make sure your email is in the box.

   Now click Save. If all goes well, your edit window will close without errors, and you can now test again. You should now be accessing your Crater install via SSL.

Now, run through the first run wizard.

A couple of items you may want are as follows:

Database address will be the container name of the Postgres database.

You can get the names by using a tool like Portainer, or by using the `docker-compose ps` command in the terminal.

Mine is called "crater_db_1".

Once through the wizard, explore the settings for Crater, and get the system customized to your needs.

Support my Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/bePatron?u=234177