SelfHosted Gateway

https://www.youtube.com/embed/VCH8-XOikQc

When you self host, one of the primary goals is to have your services available, yet secure. Many of us run into hindrances to having publicly available self hosted services because of the way our ISPs handle oru internet access.

They block ports like 80 and 443, and common email ports as well. They give you ever changing, dynamic DNS addresses so you can't depend on the IP routing you need. They put you behind two routers, often called "double NAT" or "cg-NAT". All things that make their life easier, but make our self hosted aspirations much more difficult to achieve.

Fortunately, there are teams like the FractalNetworksCo who build amazing little open source tools like the SelfHosted-Gateway. This great little piece of software can really improve your ability to self host your applications, and access them through a fully qualified domain name (FQDN) without much hassle at all. So today, we go through the setup of this application.

What You'll Need

- An external server (VPS) to tunnel to

This can be something like a Digital Ocean droplet, a Linode server, or an Oracle Free Tier server.

- An Internal Server where you host your applications (using Docker-Compose will be best in this case).
- Docker-CE and Docker-Compose installed on both servers.
- The SelfHosted Gateway Software cloned to the external and internal server.
- Make - installed on both servers.
- SSH access from the internal server to the external server (preferrably setup with SSH Keys vs. password).
- About 20 minutes of your time.

Installation

Installing Docker and Docker-Compose on your Servers

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 SelfHosted Gateway on the Cloud (external) Server

Create a new server on a VPS, then setup a subdomain with an A record pointing to it.

If this will be your main gateway for all apps in your home network, then make it a domain using a domain name registrar like Hover, GoDaddy, Cloudlfare, NameCheap, etc. You can then point subdomains to the A record of the main domain using a CNAME record, or you can make a wildcard domain using *.yourdomain.com with an A Record pointing to the IP Address (public) of yoru VPS.

1. Update the server to ensure you have the latest updates available.

2. Add non-root (sudo) user

On ubuntu based servers, do

```
adduser <your preferred username>
```

then follow the prompts.

3. Adjust ssh keys for new user

Make sure you setup SSH Keys for your new user to be able to access this server. You can generate ssh keys in linux with the command:

`ssh-keygen -t rsa 2048` and this will create a 2048 bit encryption key set. Take the defaults, and the keys will be generated named `id_rsa` and `id_rsa.pub`. Never share the `id_rsa` key, but put the `id_rsa.pub` key on your VPS. You can transfer a key to a VPS with the command:

`ssh-copy-id <your username>@<your server ip or fqdn>` So, it will look similar to `ssh-copy-id bobby2shoes@vps.bobbysboots.com`.

4. Login as new user

```
ssh <your username>@<your server ip or fqdn>
```

5. Install Docker-CE and Docker-Compose with script using the instructiosn above, if you haven't already done so.

6. clone the self hosted gateway repo

```
git clone https://github.com/fractalnetworksco/selfhosted-gateway.git
```

7. Run the following commands:

`cd selfhosted-gateway`

NOTE: You need make installed for this part.

`make setup`

`make gateway`

Now on the Client server:

1. Again, install Docker-CE and Docker-Compose if needed.

2. Clone the same repo as above.

`git clone https://github.com/fractalnetworksco/selfhosted-gateway.git`

3. cd into the repo folder that is created.

`cd selfhosted-gateway`

4. You need to make sure your users SSH keys from this server are set for use on the VPS, just lilke we did above. Then, run the following to make sure SSH is ready on the system.

`eval `ssh-agent -s``

`ssh-add ~/.ssh/id_rsa`

4a. Run the command

`make docker`

5. Run the following command for each application you want to tunnel to:

```
make
 link GATEWAY=<your ssh user and host e.g. bob@mysuperdomain.com>
FQDN=<your app subdomain e.g. myapp.mysupeerdomain.com>
EXPOSE=<your app container name and port it runs on in docker
container e.g. get_my:3000>
```

For my app, called `get_my` that runs on port 3000, I created a subdomain called `myget.routemehome.org`. I own `routemehome.org`, and added `myget` to it, using an A-record to point the sub-domain to my VPS public IP address. I used the user `brian` to ssh to this server, so my ssh command looks like `ssh brian@myget.routemehome.org`. So the command I run to get my tunnel generated is as follows:

`make link GATEWAY=brian@myget.routemehome.org FQDN=myget.routemehome.org EXPOSE=get_my:3000`

NOTE: while the domain for my ssh is the same as my FQDN, it's not necessary for it to be the same. If I were to create another tunnel for my Jellyfin app, and I gave it an FQDN of jellyfin.routemehome.org, the command to generate the tunnel would be:

```
make link GATEWAY=brian@myget.routemehome.org FQDN=jellyfin.routemehome.org EXPOSE=jellyfin:8096
```

The output from this command should give you yaml code for a docker-compose segment to add to the compose file of the app you are tunneling.

If that command doesn't work, then use the create-link.sh script found in `../gateway/scripts/`

The syntax will be very similar:

```
link:
    image: fractalnetworks/gateway-client:latest
    environment:
      LINK_DOMAIN: myapp.mysuperdomain.com
      EXPOSE: get_my:3000
      GATEWAY_CLIENT_WG_PRIVKEY: OJpZOqmUiYEEEMsYLSmlnOP/HoHM8LOf5hxgtx4mEHA=
      GATEWAY_LINK_WG_PUBKEY: DBdD2HbaapfqQQtPLQi9lj54MR96q5Fs418JHpO3TBg=
      GATEWAY_ENDPOINT: 109.45.23.221:49153
    cap_add:
      - NET_ADMIN
```

Take this yaml code, and add it as a new service section in your docker-compose.yml file, then do

```
docker-compose up -d
```

to bring up the tunnel and app. Now go to your subdomain on https://myapp.mysuperdomain.com and your app should come up... tunneled out of your local server through your vps with encryption via WireGuard.

Repeat this tunnel generation for as many apps as you want, and happy self-hosting!

Support my Channel and Content

Support my Channel and ongoing efforts through Patreon:

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

Revision #2
Created 2022-09-30 15:07:02 UTC by Brian McGonagill
Updated 2023-02-13 23:59:13 UTC by Brian McGonagill