Self Hosted Tunnels

- Pangolin
  - Install and Configure Pangolin

Pangolin

Install and Configure Pangolin

https://www.youtube.com/embed/zEBaXAu8zE0

Setup a Server

For this project, you'll need / want a server outside of your network that has a fixed public IPv4 address. For this, you can use services like Digital Ocean, RackNerd, Linode, Vultr, and so many more. I have affiliate links for Digital Ocean and Racknerd, so if you use my links, you help support this content, but feel free to use any service you prefer.

Install updates to our Server

Ubuntu / Debian

```
sudo apt update && sudo apt upgrade -y
```

RedHat / CentOS / Fedora / Alma / Rocky

```
sudo dnf update -y
```

Add a non-root / sudo user on the server

Generally, when you setup a new server, the VPS (Virtual Private Server) service sets up a default "root" user for you. It's considered unsafe to do everything as "root", so let's setup a non-root user who has super user (sudo) privileges.

```
adduser <username>
```

You'll be prompted to enter and confirm a password for this user. You'll also be asked for some user information like Name, etc, but this is not required information. At the end, confirm the entries, and you'll have your new user.

Next, we need to add the user to the super user group.

Ubuntu / Debian

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

RedHat / CentOS / Fedora / Alma / Rocky

```
usermod -aG wheel <username>
```

Now, you can log out of the system, and log back in as your new non-root super user.

Setup DNS Records

Pangoling relies on two DNS records in order to make it easy to route service traffic properly. The first is a record to access the Pangolin Admin Dashboard. We will create this as a specific subdomain of our main (base) domain.

In my case, I own the domain `sysmainit.com` , and through the domain registrar's dashboard I can setup new DNS records for this domain. You should learn how to do this for the domain you own.

I will add a subdomain specifically for accessing the Pangolin admin console (e.g. pang.yourgreatdomain.com), and wildcard subdomain, so all other subdomain requests are handled by the reverse proxy (e.g. *.yourgreatdomian.com).

DNS Record Type	Sub-domain	IP address
A	pang	142.142.254.12
A	*	142.142.254.12

With the above information we tell our domain, that the URL "https://pang.yourgreatdomain.com should point to the public IPv4 address 142.142.254.12, and that any other subdomain (such as https://jellyfin.yourgreatdomain.com) should also point to 142.142.254.12. Although these both point to the same IP, the Pangolin setup will make sure they do very different things.

Install the Pangolin Server Software

1. Use the Pangolin quick Installer to install and run Docker, Docker compose, and Pnagolin all at once with a single script.

You'll need the following information when you run the installer

1. Base Domain (e.g. yourgreatdomain.com)
2. subdomain for accessing the Pnagolin admin console (e.g. pang, or pangolin, or pang-admin, etc)
3. A valid email address for the LetsEncrypt setup.
4. An email to use as the admin user (can be the same as above).
5. A Password for the admin user.
6. A determination on whether you want to install CrowdSec for added security.
7. (Optional) SMTP (Email) server and credential information for Email setup.

Once complete, you'll be able to access the Pangolin admin console at the FQDN you made for it (e.g. https://pang.yourgreatdomain.com). Login using the admin username (email) and password you created while running the installer.

Before you setup your first site (Pangolin node), you'll need to have a server (VM, Incus, LXD, LXC container, or Docker Container) ready to go inside your local network. You can use a system you already have services running on, or a system where you already have Docker and Docker Compose running, or setup a small Incus Virtual Machine (VM) running Ubuntu 24.04 server.

If you're setting up a new server / VM / Container, you should run through the same non-root user with sudo privilege setup as we did for the public facing server. Then install the Newt client software.

Ingress Point Setup into your Network

You can setup a machine where you're already running services as your ingress point, or you can setup a server, VM, or Container (incus, LXC, LXD, or Docker) to act as the ingress server.

I setup an Incus Container running a very tiny Ubuntu 24.04 server install, and it works perfectly.

For this portion of the setup, we are going to setup a special Wireguard VPN client called **Newt**. This client is specifically created to act as the ingress point of a secure tunnel running over Wireguard.

I used the amd64 binary on my ingress server, but feel free to use the one that fits for your architecture. Additionally, you can opt to use a Docker container, but you'll need to check the Newt documentation for the specifics on how to set that up.

Now that we know where we want to run Newt, we can start creating a site in our Pangolin Admin server dashboard.

1. On the sites page, click 'Add Site +'.
2. Give your site (node) a recognizable name.
3. Select the 'Newt Tunnel' option (it should be selected by default).
4. Copy the credentials that are auto-generated for 'Newt Endpoint', 'Newt ID', and 'Newt Secret Key'.
5. Check the box to confirm you have securely saved the site credentials.
6. Select the Operating System / Kernel running on your chosen ingress node. In my case I selected 'Linux'.
7. Select the architecture type for the chosen ingress node. In my case it is 'amd64'.
8. Copy the commands provided, and get ready to paste them into your ingress node's terminal.
9. Click 'Create Site'.
10. Paste the copied commands into the terminal for your ingress node, and press Enter.

11. Allow the install to work, and you should see Newt start up and start pinging.

> It's important to click the 'Create Site' button before starting the Newt application, or it will give an error that it can't find the endpoint or connect. If you see that error, simply return to your Pangolin dashboard, and finish creating your site by clicking the button. Newt will try again in 10 seconds, and should connect.

Now, we need to create a system service so Newt will run automatically, in the background, and even when we reboot the node.

In the node's terminal, we'll create a file called 'newt.service'.

`nano newt.service`

Now, copy the code block below, and paste it into the file you just created.

```
[Unit]
Description=Newt VPN Client
After=network.target

[Service]
ExecStart=
Restart=always
User=root

[Install]
WantedBy=multi-user.target
```

Now, we need to add the startup command for newt with your node's id, secret key, and endpoint. This was the second part of the command we copied from your Pangolin dashboard.

Save the 'newt.service' file with CTRL + O, press Enter to confirm, and exit the editor with CTRL + X.

Now use the up arrow key on your keyboard to view the previously entered commands.

When you see the long command you pasted earlier, copy the last half of the command starting with `./newt` .

It should look something like:

`/newt --id s0m3l0n6rAnD0MS7r1n6 --secret aN3venL0n6erRAnD0mStr1n6ofchARAC73r5 --endpoint https://proxy.yourgreatdomain.com`

Re-open the 'newt.service' file with

```
nano newt.service
```

and paste this line in after the equal '=' sign on the line with "ExecStart=". Modify the beginning of the command by removing the `./` and adding `/usr/local/bin/` to it. So the whole line should look like:

```
ExecStart=/usr/local/bin/newt --id s0m3l0n6rAnD0MS7r1n6 --secret aN3venL0n6erRAnD0mStr1n6ofchARAC73r5 --endpoint https://proxy.yourgreatdomain.com
```

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

Now we need to move our "newt" application to the `/usr/local/bin` location with:

```
sudo mv newt /usr/local/bin/
```

Next, we need to move the service file we just created to `/etc/systemd/system` with

```
sudo mv newt.service /etc/systemd/system/
```

Finally, we'll start and enable our newt service with the commands:

```
sudo systemctl start newt.service
```

and

```
sudo systemctl enable newt.service
```

We can check the newt.service status to ensure it's active and running with:

```
sudo systemctl status newt.service
```

You should see that it's **active** and that it is running ping checks in the logging section. This indicates that the newt service is running, and that you now have a tunnel from your ingress node out to your Pangolin server.

You are not restricted to a single site (node) with Pangolin. you can have multiple nodes, and can have multiple organizations to help you create tunnels for all your services regardless of where they are hosted.

Service (Resource) Setup on your LAN

Now we want to create access to our services (called Resources in Pangolin).

1. Click on 'Resources' in the left navigation.

2. Click the 'Add Resource +' button near the upper right of the view.
3. Give your Resource (site) a recognizable name. For instance if you are setting access to Jellyfin, then name it Jellyfin.
4. Next, you can choose from any nodes you've setup for the organization in the 'Site' drop down. You need to choose a node which will have local (LAN) access to the service, or localhost access if both the Newt node and service are running on the same machine.
5. For resource type, you can choose 'HTTP Resource'.
6. In the HTTP Settings section, give your resource a sub-domain. For instance, with Jellyfin, you may just want to use 'jellyfin', or maybe just 'jf'. It's up to you.
7. Make sure the proper base domain (e.g. yourgreatdomain.com) is selected in the drop down.
8. Click 'Create Resource'.
9. On the next screen, choose the 'Method' (http or https usually).
10. Then enter the local IPv4 address of the resource (e.g. 192.168.1.24)
11. Finally, enter the port the service runs on (e.g. 8096 for Jellyfin, or 8123 for Home Assistant, etc).
12. Click the 'Add Target' button.
13. Then click the 'Save Target' button.
14. You should now be able to access your application (resource) over the internet by using the FQDN (fully qualified domain name) you've setup for it.

Multiple Domains in Pangolin

If you're like me, then you may own a few domains, and want to use Pangolin to act as a tunnel for all of them. This is 100% possible with Pangolin, but we need to make a quick edit to our Pangolin server.

Let's say you want to use Pangolin to proxy traffic for your domains yourgreatdomain.com, domainsrock.com and opensourcerocks.org. You need to do a few things to make this successful.

Create DNS A Wildcard (*) Records

You need to create a wildcard (*) A-Record for each domain, and make sure it points to your Pangolin Server's public IPv4 address. If your server's public address is 212.145.66.21 then you'd want three A records.

```
Type      Subdomain   Base Domain            IPv4
A             *       .yourgreatdomain.com   212.145.66.21
A             *       .domainsrock.com       212.145.66.21
A             *       .opensourcerocks.org   212.145.66.21
```

You still need the single DNS A Record which points your subdomain to the dashboard access for Pangolin:

`proxy.yourgreatdomain.com  -> 212.145.66.21`

This is still the FQDN you use to access the Pangolin admin dashboard.

Update the Pangolin Server Config

SSH or access your Pangolin server, and find the files that the Pangolin install script created. You should see something like this:

```
$~    ls
config  docker-compose.yml  installer
$~
```

We need to go into the "config" folder,

`cd config`

Now we need to open the "config.yml" file and make some additions.

`sudo nano config.yml`

You'll see something that looks like the below:

```
# To see all available options, please visit the docs:
# https://docs.fossorial.io/Pangolin/Configuration/config

app:
    dashboard_url: "https://proxy.yourgreatdomain.com"
    log_level: "info"
    save_logs: false

domains:
    domain1:
        base_domain: "yourgreatdomain.com"
        cert_resolver: "letsencrypt"

server:
    external_port: 3000
    internal_port: 3001
    next_port: 3002
    internal_hostname: "proxy"
    session_cookie_name: "p_session_token"
    resource_access_token_param: "p_token"
```

```
    resource_access_token_headers:
        id: "P-Access-Token-Id"
        token: "P-Access-Token"
    resource_session_request_param: "p_session_request"
    secret: somesuperlongcharacterstringoflength
    cors:
        origins: ["https://proxy.yourgreatdomain.com"]
        methods: ["GET", "POST", "PUT", "DELETE", "PATCH"]
        allowed_headers: ["X-CSRF-Token", "Content-Type"]
        credentials: false

traefik:
    cert_resolver: "letsencrypt"
    http_entrypoint: "web"
    https_entrypoint: "websecure"

gerbil:
    start_port: 51820
    base_endpoint: "proxy.yourgreatdomain.com"
    use_subdomain: false
    block_size: 24
    site_block_size: 30
    subnet_group: 100.89.240.0/20

rate_limits:
    global:
        window_minutes: 1
        max_requests: 500

users:
    server_admin:
        email: "admin@yourgreatdomain.com"
        password: "a-server-admin-password"

flags:
    require_email_verification: false
    disable_signup_without_invite: true
    disable_user_create_org: false
    allow_raw_resources: true
    allow_base_domain_resources: true
```

In this file, we want to add more domains subsections. Under the main 'domains' section, we'll create a new line, then add the appropriate spaces.

> Remember, YAML is space dependent. We want to use the proper number of spaces so each subsection lines up with the same indentation level as the subsection above it.

Looking at the modified 'config.yml' file below, you'll notice I've added two more domain subsections to the yaml file. They look like this within the 'domains:' section.

```
    domain2:
        base_domain: "domainsrock.com"
        cert_resolver: "letsencrypt"
    domain3:
        base_domain: "opensourcerocks.org"
        cert_resolver: "letsencrypt"
```

First, you add a new domain count subsection (e.g. 'domain2', 'domain3', and so on). Next, under each subsection we add the item for the 'base_domain', and the associated 'cert_resolver'. In most cases, you'll want the 'cert_resolver' to be 'letsencrypt'.

Add the appropriate subsections for your additional domains into the 'config.yml' file.

```
# To see all available options, please visit the docs:
# https://docs.fossorial.io/Pangolin/Configuration/config

app:
    dashboard_url: "https://proxy.yourgreatdomain.com"
    log_level: "info"
    save_logs: false

domains:
    domain1:
        base_domain: "yourgreatdomain.com"
        cert_resolver: "letsencrypt"
    domain2:
        base_domain: "domainsrock.com"
        cert_resolver: "letsencrypt"
    domain3:
        base_domain: "opensourcerocks.org"
        cert_resolver: "letsencrypt"
```

```
server:
    external_port: 3000
    internal_port: 3001
    next_port: 3002
    internal_hostname: "proxy"
    session_cookie_name: "p_session_token"
    resource_access_token_param: "p_token"
    resource_access_token_headers:
        id: "P-Access-Token-Id"
        token: "P-Access-Token"
    resource_session_request_param: "p_session_request"
    secret: somesuperlongcharacterstringoflength
    cors:
        origins: ["https://proxy.yourgreatdomain.com"]
        methods: ["GET", "POST", "PUT", "DELETE", "PATCH"]
        allowed_headers: ["X-CSRF-Token", "Content-Type"]
        credentials: false

traefik:
    cert_resolver: "letsencrypt"
    http_entrypoint: "web"
    https_entrypoint: "websecure"

gerbil:
    start_port: 51820
    base_endpoint: "proxy.yourgreatdomain.com"
    use_subdomain: false
    block_size: 24
    site_block_size: 30
    subnet_group: 100.89.240.0/20

rate_limits:
    global:
        window_minutes: 1
        max_requests: 500

users:
    server_admin:
        email: "admin@yourgreatdomain.com"
        password: "a-server-admin-password"
```

```
flags:
    require_email_verification: false
    disable_signup_without_invite: true
    disable_user_create_org: false
    allow_raw_resources: true
    allow_base_domain_resources: true
```

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

Now we need to restart the Pangolin server with the commands:

`docker compose down`

Once that completes, enter

`docker compose up -d`

Wait for the start up to complete, return to your Pangolin dashboard in the browser, and refresh it. Now when you go to add a resource, the base_domain drop-down field should list all of your added domains as options.