Setting up an NGINX Reverse Proxy with SSL

A reverse proxy is a function of a web server that allows it to forward requests onto other web servers. Typically they are set up in combination with a wildcard SSL certificate, allowing each subdomain to go to a different server. Apache2 and NGINX can both function as reverse proxies, but NGINX is much easier to set up!

Installing NGINX

Install using apt

First, install nginx using apt:

```
sudo apt install nginx
```

Testing with telnet

Telnet is great for testing TCP ports. NGINX listens on TCP port 80 by default, so we can test it with telnet:

```
telnet 127.0.0.1 80
```

If the server is responding correctly, you should see an output similar to below:

```
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

Configuring NGINX

> NGINX configuration files are stored in /etc/nginx/ - the site configuration files are stored in sites-available/, then these are enabled by creating a symbolic link to them in sites-enabled/

Disabling the default site configuration file

Now that we know the server is running, we are going to disable the default NGINX configuration file. This is so that we can create a new one that is more focused around the reverse proxy abilities of NGINX.

To remove the default configuration file, run this command:

```
sudo rm /etc/nginx/sites-enabled/default
```

Creating a new site configuration file

We will now create a new site configuration file, using our domain name as the file name:

```
sudo nano /etc/nginx/sites-available/YOUR.DOMAIN
```

Here is an example template, which retains NGINX's default site configuration for the root domain and does reverse proxy for a subdomain to a Proxmox server:

```
# We are defining our Let's Encrypt certificate here
# If your NGINX server will serve multiple domains, you will instead need to place it within
the 443 definition
ssl_certificate /etc/letsencrypt/live/YOUR.DOMAIN/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/YOUR.DOMAIN/privkey.pem;

# Required when reverse proxying to a Proxmox server
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}

# This will forward all HTTP requests to HTTPS
server {
listen 80;
listen [::]:80;
server_name *.YOUR.DOMAIN;
rewrite ^ https://$host$request_uri? permanent;
}

# This will serve NGINX's default web page, via HTTPS
server {
listen 443 ssl;
listen [::]:443 ssl;
server_name YOUR.DOMAIN;
```

```
        # Define the web root
        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }
}

# This subdomain will reverse proxy requests to a Proxmox web interface
server {
        listen 443 ssl;
        listen [::]:443 ssl;
        server_name proxmox.YOUR.DOMAIN;

                location / {
                        proxy_pass              https://YOUR.PROXMOX.IP.ADDRESS:8006;
                        proxy_set_header        Upgrade $http_upgrade;
                        proxy_set_header        Connection "upgrade";
                }
}
```

Once modified to suit your setup, use Control-S to save and Control-C to quit nano.

Enabling the new configuration file

We now have a new configuration file, and we are going to enable it by creating a symbolic link:

```
sudo ln -s /etc/nginx/sites-available/YOUR.DOMAIN /etc/nginx/sites-enabled/YOUR.DOMAIN
```

Testing the configuration

We can now run nginx -t in order to test the new configuration file:

```
sudo nginx -t
```

Assuming all is well, you should see an output similar to below:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Restarting NGINX

Once we've confirmed that the configuration file tests successfully, we can apply it by restarting the NGINX service:

```
sudo systemctl restart nginx
```

Testing port 443 with telnet

We already know that NGINX was listening on TCP port 80, but since we have just enabled SSL, we want to make sure that it's also listening on port 443:

```
telnet 127.0.0.1 443
```

The output should be similar to before:

```
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

Created 2024-11-22 14:55:45 UTC by Nicolas
Updated 2025-01-31 23:35:26 UTC by Nicolas