NGINX

Credit: http://nginx.org

Access services with Nginx reverse proxy

Nick Peers discovers how to open your network services to the internet with this user-friendly implementation of Nginx.

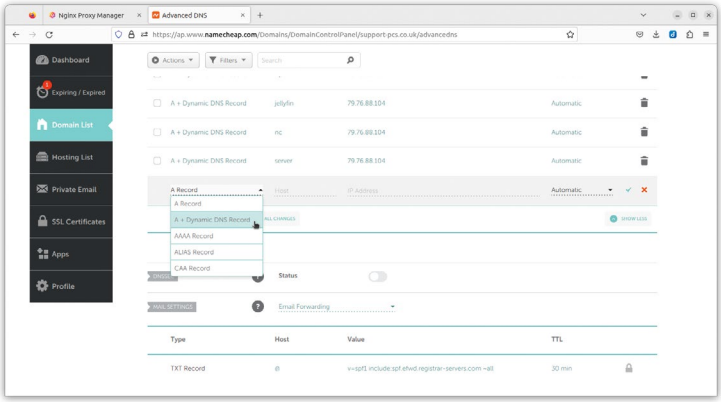
OUR EXPERT

Nick Peers
Is obsessed with self-hosting. From Bitwarden and Jellyfin to Nextcloud and MotionEye, all his services are so much easier to configure for remote access with *Nginx Proxy Manager*. He is the life of the party, too!

If you're looking for a way to access self-hosted services from outside your home, you have two basic choices. For maximum security, you'd set up a VPN tunnel that you'd need to dial into every time you wanted access, but if you'd like a simpler option that's still secure, and gives others access, you want a reverse proxy.

One of the best-known tools for the job is *Nginx* (**https://nginx.org**), but it's not the most user-friendly, even when wrapped up in deployment-friendly containers like Linuxserver's *Swag*. If the idea of messing around with separate config files for each service you want to open up sounds too much like hard work, you'll love *Nginx Proxy Manager*. It enables you to set up, view and administer all your connections via a pleasingly easy-to-use web front-end.

Like *Swag*, *Nginix Proxy Manager* is distributed as a *Docker* container. We're assuming you have *Docker* set up and configured, and have some knowledge of how it works. *Nginx Proxy Manager* can be installed on any machine on your network, but we're assuming you'll want it running 24-7, so you'll want it on your dedicated server where it's likely most other shared services are also running, many of which may be running in *Docker*.



Creating subdomains for each of your services makes them easier to configure for remote access using Nginx Proxy Manager.

Nginx Proxy Manager can redirect traffic anywhere on your network, but if you want to direct traffic to other *Docker* containers, they must all be on the same (custom) bridge network as *Nginx Proxy Manager*. If you've not yet set up this shared bridge, it can be done with a single *Docker* command, substituting `shared-net` with your choice of shared network name:

```
$ docker network create shared-net
```

You need to then connect your existing *Docker* containers to this network – if you're administering them through the *Portainer* web interface, this is simple enough. Navigate to the container's details page, scroll down to Connected Networks, where you can select your new network and click Join Network.

If you're managing *Docker* from the command line, you need to stop and remove each container in turn, then insert the following line into the setup script you use to recreate it:

```
--net=bridge-for-all-seasons \
```

Get your domains in order

You've configured *Docker*, but before you install *Nginx Proxy Manager*, there's a couple more prerequisites to

» INSTALL NGINIX PROXY MANAGER

The quickest and easiest way to deploy *Docker* containers is by pasting the setup script into a text editor, then saving the file in plain text so you can easily copy and paste it into a terminal window, plus make edits whenever required.

The script here, of course, needs adapting to your personal setup, namely the name of your shared network under `--net` and the location of your config and LetsEncrypt certificate folders under `-v`. You may also need to adapt the `-p 81:81 \` line if port 81 is already in use (for example, change it to `-p 82:81` to access *Nginx Proxy Manager* through **http://192.168.x.y:82**).

```
docker run -d \
  --name=nginx-proxy-manager \
  --net=shared-net \
  -p 443:443 \
  -p 80:80 \
  -p 81:81 \
  -v /home/dockeruser/
containers/nginx/data:/data \
  -v /home/dockeruser/
containers/nginx/letsencrypt:/
letsencrypt \
  --restart unless-stopped \
  jc21/nginx-proxy-
manager:latest
```

satisfy. The first is that you need access to a suitable domain to provide access to your servers from the outside world – this can be your own domain, from which you can optionally create separate subdomains for each service you have, such as **jellyfin.domain.com**, or you could make use of a dynamic domain like that offered by NoIP (**www.noip.com**).

If using your own domain, start by making a note of your current public IP address (use **www.whatsmyip.com**), then log into your domain provider and make sure the domain (plus any relevant subdomains if applicable) are pointing to the same IP address.

Unless this is a fixed IP address, you also need to install a DNS updater, so when your ISP changes your IP address, your domains are automatically redirected. Check your domain provider – many support *DDClient* (**https://ddclient.net/protocols.html**), either as a standalone tool or via the Linuxserver *Docker* instance (**https://docs.linuxserver.io/images/docker-ddclient**).

With your domains set up and pointing towards your home, it's time to configure your router to redirect all web requests to the server you'll install *Nginx Proxy Manager on.* Consult your router's documentation for full details, but look in the port forwarding section and direct traffic on both port 80 (http) and 443 (https) to your server's private IP address (**192.168.x.y**). This reveals another benefit of a proxy – you don't need to configure lots of ports in your router; simply funnel everything through standard web ports to your server, then let *Nginx Proxy Manager* distribute them.

Nginx Proxy Manager

It's time to add *Nginx Proxy Manager* to your server. The install box *(opposite page)* reveals the script you need to adapt to your own setup, but first you need to create two folders, inside which you'll store your *Nginx* configuration data and SSL certificates – in the example script, we've set up an **nginx** parent folder, inside which are two subfolders: **data** (for **config**) and **letsencrypt** (for the SSL certificates).

A quick summary of what the script does: it pulls and installs the latest version of *Nginx Proxy Manager,* joins it to the shared network you set up earlier and redirects three outside ports into Nginx: 80 and 443, as we've already explained, while port 81 is used to connect to its web-based interface.

If you've set things up correctly, the container starts without a hitch and should be available within seconds.

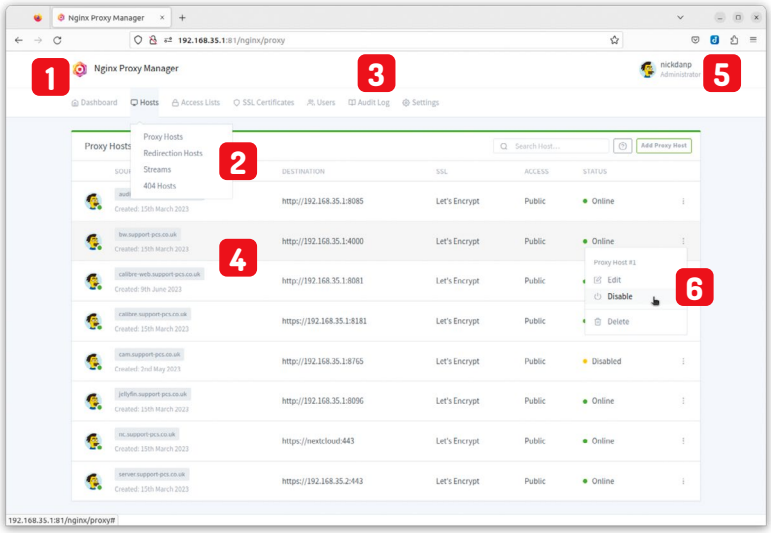
Browser-based access

Once installed, *Nginx Proxy Manager* can be accessed through any device on your network via any web browser. Simply navigate to **http://192.168.x.y:81**, substituting **192.168.x.y** with your server's IP address, which takes you to a login screen.

Log in using **admin@example.com** as your username, and **changeme** as your password. You'll then be prompted to change the email address (as well as provide a name and nickname). Click Save to change the password – make it a secure password generated by (and stored in) your password manager, even if you only intend accessing *Nginx Proxy Manager* locally.

Once done, you'll find yourself at the Users screen. Above this you'll see there are seven distinct sections to navigate – start by clicking Dashboard for an overview. Here you'll see *Nginx Proxy Manager* supports four types of proxy: Proxy Hosts, Redirection Hosts, Streams and 404 Hosts.

NGINX PROXY MANAGER OVERVIEW



1 Overview
Click on the Dashboard tab to get a quick overview of how many proxies you have set up.

2 Hosts
Nginx Proxy Manager enables you to set up one of four different types of proxy.

3 Audit log
Click here to get a complete – and detailed – list of all the operations and changes you've performed.

4 View summary
Get an overview of each proxy host with this handy and easy-to-read list.

5 User access
Click Users to give others access – this can be read-only or administrator, depending on your level of trust.

6 Quick actions
Click the vertical ellipsis to make changes to existing proxies, plus disable them temporarily or delete them.

Your very first proxy

Let's start with Proxy Hosts. The step-by-step guide *(next page)* reveals how to set up a simple proxy host, which enables you to redirect traffic from a specific domain, subdomain or dynamic domain to your choice of IP address and port. When choosing the scheme (http or https), stick with whatever you use to connect locally – for example, the self-hosted Bitwarden *Docker* instance Vaultwarden only uses http. Don't worry, this doesn't mean your connection is insecure – that's configured separately.

You have a choice between hostname and IP address of the machine you're redirecting to – if your service is on a machine with a static IP address, use that; if it's on machine with an IP address allocated to it by your router, try the hostname route. If that doesn't work, tie that machine down to a static IP address.

When bad actors try to attack websites, they often use rather basic techniques, such as attempting to inject malicious text into insecure web forms. When you open your servers up to internet access, you put them at risk from such attacks, which could have wider consequences for your network.

Nginx Proxy Manager offers a Block Common Exploits switch for all proxy and redirection hosts, which offers protection against the most common forms of attack. These are listed in the **block-exploits.conf file**, which can be found by navigating to **https://github.com/NginxProxyManager/nginx-proxy-manager**

QUICK TIP

If you're unable to get Access Lists working the way you want them to, consider examining other possible solutions – for example, your router's firewall might offer a way of blocking unwanted traffic from known bad actors or restricting access to IP addresses from a specific part of the world.

QUICK TIP

Select Users on the *Nginx Proxy Manager* dashboard to give others access to *Nginx*. You can give them view-only access to your proxies or give them full administrative access – ostensibly for their own services, but they also have full access to yours, so be warned.

and then drilling down to **docker/rootds/etc/nginx/conf.d/include**. You'll see it's a simple script that intercepts both SQL and file injections, as well as common exploits, spam and user agents.

If you're using a single domain with subfolders, as outlined in step three of the walkthrough, you can configure all those subfolders in a single proxy host. You still need to fill in redirection details for the parent domain on the Details tab, so consider where you'd like people entering that address to be redirected.

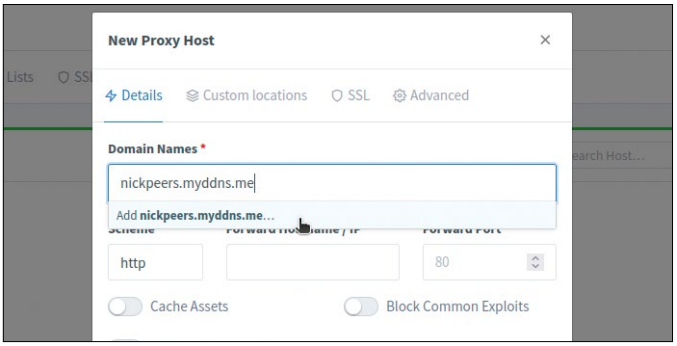
When you come to secure outside web connections to your server, you need an SSL certificate. While you can provide this yourself, *Nginx Proxy Manager* has built-in support for requesting free certificates from LetsEncrypt. Not only does it configure these automatically, but it also handles the otherwise arduous chore of remembering to renew the certificate every three months. Once added, you can view (and manage) your SSL certificates via their own dedicated section – you get a handy summary of each, including the certificate provider, its expiry date and options to renew manually now, download the certificate to your PC, test the server reachability and remove if you have no further use for it.

Adding user credentials

Opening local services to access from outside your network comes with obvious risks, only some of which can be mitigated using the switch to block common exploits. While some services may have built-in authentication (for example, Nextcloud or Jellyfin), others may give anyone with knowledge of your domain name unfettered access. *Nginx Proxy Manager's* Access Lists offer you two ways to mitigate this: simple http authorisation, which forces anyone visiting the site to enter a username and password before they can proceed, and a combination of whitelists and blacklists, enabling you to restrict access to specific clients only.

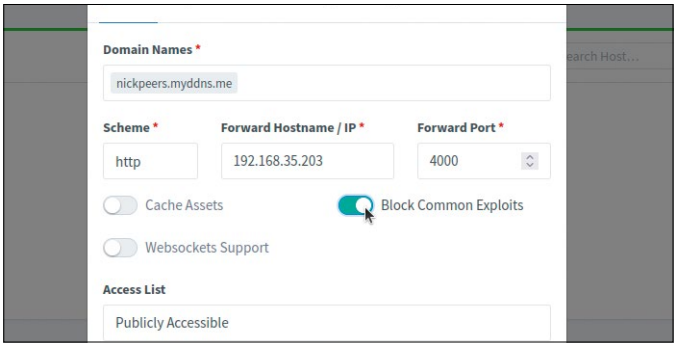
Here's a simple example that uses http authorisation. Select Access Lists in *Nginx Proxy Manager,* then Add Access List to get started. Give

SET UP A PROXY HOST



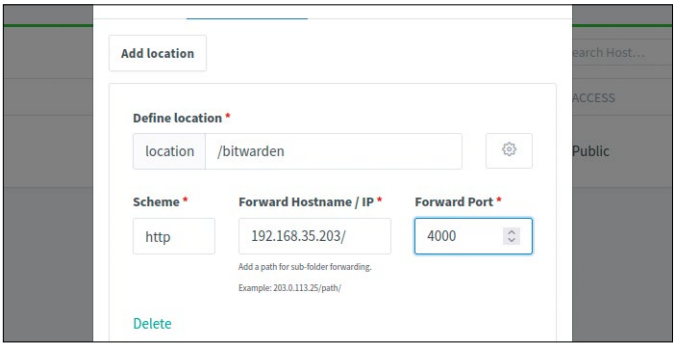
1 Configure basic settings

Click X Proxy Hosts followed by Add Proxy Host. The New Proxy Host window opens to show four tabs. Start by entering the domain, subdomain or dynamic domain you'd like to use for this proxy into the Domain Names field, clicking Add when done. Next, enter the details (scheme, host and port) of where you're redirecting the domain.



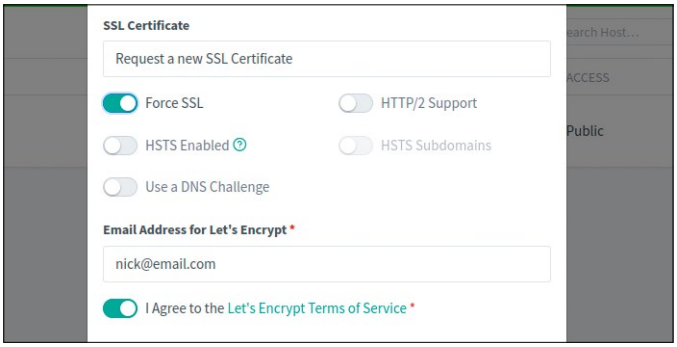
2 Set optional switches

There are three more switches: Cache Assets can speed things up if the service is on a different computer from *Nginx Proxy Manager* and pulls in lots of static content. Block Common Exploits is discussed in the main copy – it's worth switching on for security reasons. Leave Websockets Support disabled until you've tested the connection; try enabling it if it doesn't work.



3 Define custom locations

If you want to connect to a domain using subfolders (such as **https://domain.com/bitwarden**), switch to Custom Locations and click Add Location to fill in the details – it's self-explanatory: choose a path name, then select the scheme, hostname/IP address and forwarding port as in step two. Leave a **/** after the hostname or IP address (such as **192.168.x.y/**) or it won't work.



4 Add SSL certificate

Switch to the SSL tab to add an SSL certificate for the site. Click None under SSL Certificate and select Request A New SSL Certificate (going forward, you can select previously configured certificates from here, too). Provide your email address, agree to the terms and – if required – enable the Force SSL switch to only allow secure (https) connections before clicking Save.

your list a suitably descriptive name and flick the Satisfy Any switch to On. Now switch to the Authorization tab and enter your desired username and password. You can, if you wish, add multiple usernames and passwords, in fact, any of which will allow visitors to access the server.

Click Save, then test your new rule by switching back to the Proxy Hosts tab. Click the vertical ellipsis button next to one of your proxy hosts and choose Edit to make changes. Click under Access List and select your new rule before clicking Save. Now browse to the web address associated with this proxy host and you should see a prompt asking you for a username and password. Enter the details you created, and you should gain access to your server.

Keep it local

Our second example prevents anyone from outside your local network accessing your server. Create a new rule, but this time leave Satisfy Any switched off. This time, switch to the Access tab and type the following into the allow box: **192.168.0.0/24** (substitute **192.168.0** with whatever's used by your network, such as **192.168.1**).

Click Save, add the rule as before to a proxy host, then test: try connecting through your local network using the URL, which should work as normal. Once confirmed, try connecting from outside your home network (by using your mobile's cellular data, say), where you should find access is now denied.

You can combine both examples in a single rule (make sure Satisfy Any is enabled), which would mean anyone on your home network can access the service without limits, while those connecting remotely would be prompted for a username and password before they could gain access.

Thoroughly test that the services continue to work as you'd expect after setting up your rules – some may no longer function correctly, as we discovered while trying to access our Audiobookshelf libraries. The rules may render access from outside your web browser – such as through a mobile app – impossible, too.

Access Lists is a little basic in that it only works with IP addresses and IP address ranges. There's a push among the *Nginx Proxy Manager* community to include GeoIP2, a module that can be used to restrict users by geographical location, enabling you to – for example – block access from outside your home country. Another workaround – if your router supports it (Synology ones do) – can be found in the Quick Tip *(page 57)*.

Beyond proxy hosts

Proxy hosts are specifically designed for web (http/https) traffic that needs directing to a specific device and port. A broader option for non-web traffic can be found using Streams *(outlined in the box below)*.

Nginx Proxy Manager supports two further types of host. Redirection Hosts enable you to redirect traffic from one domain to another (the domain redirected must be configured to point to your home network), while 404 Hosts allow you to redirect domains pointing at your home network to a customised error page.

This error page is defined under Settings > Default Site – by default, you redirect users to a simple (and irrelevant) congratulations page that's a reminder to set up the host for access, but you can replace it with a generic 404 error page, redirect to another site or insert your own message using custom HTML code. LXF

QUICK TIP

Visit https://nginxproxymanager.com for a comprehensive introduction to *Nginx Proxy Manager*. You'll find setup instructions as well as a link to the project's GitHub page, where you'll discover an engaged community and a project frequently updated with new features.

» SET UP STREAMS

If you want to manage non-web traffic, such as a *Syncthing* peer-to-peer connection or a game server, you need to set up Streams. These funnel TCP and/or UDP traffic from a specified port outside your network to a specified device within your LAN.

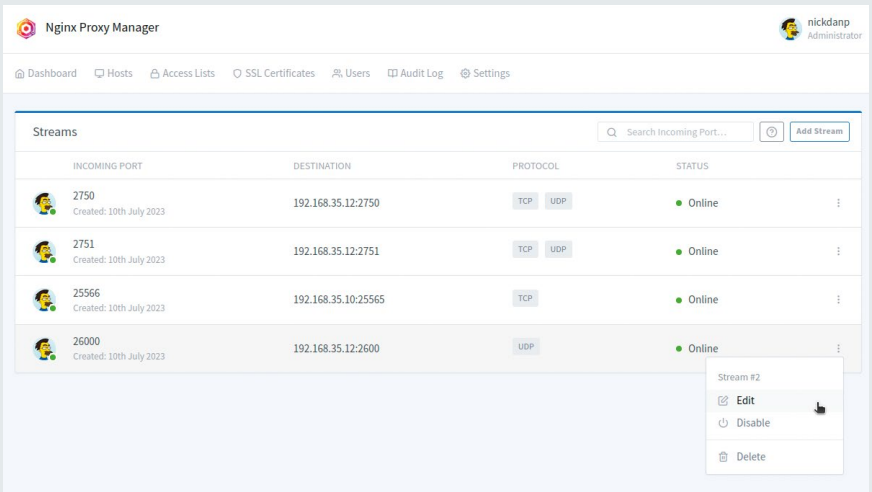
First, you need to configure your router to forward any traffic on those ports to the server running *Nginx Proxy Manager*. Once in place, you need to shut down and destroy the container:

```
$ docker stop nginx-proxy-manager
$ docker remove nginx-proxy-manager
```

Next, edit the *Docker* script *(see first box)* to add the necessary `-p` lines to open the required ports – note the use of `/udp` and `/tcp` to denote the protocols used. This example opens port 1000 for both UDP and TCP traffic:

```
-p 1000:1000/udp \
-p 1000:1000/tcp \
```

Copy and paste the edited script into your terminal window to recreate the container with the port(s) now accessible, then navigate to Hosts > Streams. Click Add Stream to set it up in a similar way to a proxy host, except this time all you need to do is supply the incoming port, forward host and port, plus which protocols to use. Click Save.



You need to configure individual Streams for every single port you want to redirect UDP and TCP traffic on