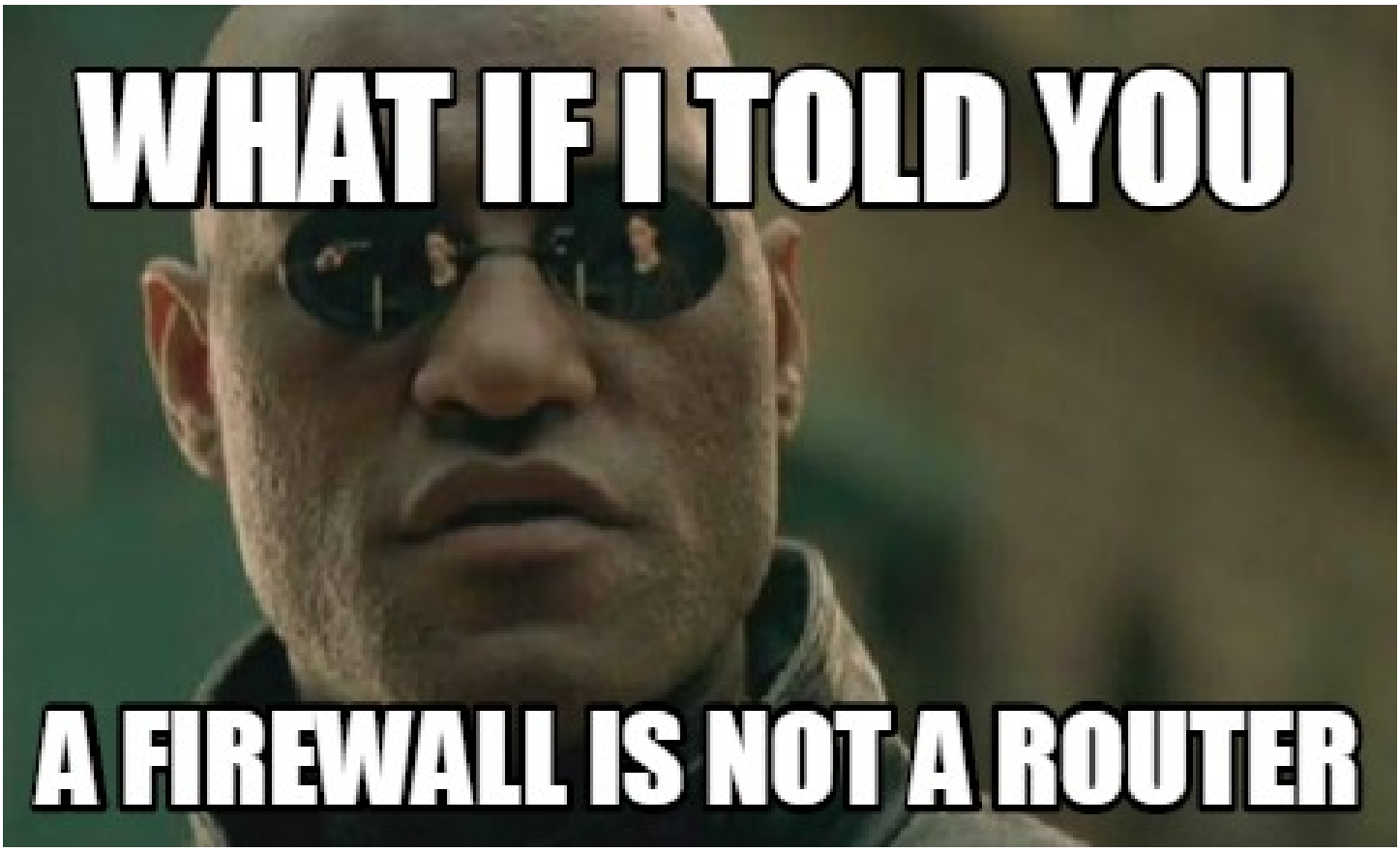
J A M E S
L O P E R

How to run a server behind a firewall without port forwarding



Normally without sufficient access to the router, it's not possible to expose your computer to the internet so it can act as a server. This is often the case, you have a powerful computer behind internet you have no control over. Here I will show the best way to circumvent that via a gateway server.

I initially wanted to create a sort of hollow gateway to the network interface on the target server, however this is actually a real waste of the gateway server. The gateway could serve other apps, or proxy more than one remote computer. A better way would be to use name-based virtual hosts to route incoming requests to the right server. In this guide I use NGINX to handle requests and routing, SSH to handle the forwarding, and systemd to ensure SSH stays connected at all times.

Here is the config file for ssh for convenience

```
sudo nano /etc/ssh/sshd_config
sudo service ssh restart
```

Configure the target computer for basic usage of SSH RemoteForward

We need to make the target computer reach out to the proxy to initiate the forwarding. Edit `~/.ssh/config` and add the following hostname block. This forms the groundwork for all to come.

```
Host proxy
  HostName example.com
  User ubuntu
  RemoteForward 4000 localhost:4000
  ServerAliveInterval 10
```

At this point, if you run `ssh proxy` on the target computer, you'll be able to navigate to `example.com:4000` If that's all you need, great! Next we'll bulk up this thing by daemonizing it and integrating with NGINX.

Configure SSH public key authentication

Since this is going to be a daemon, there will be no prompt for the SSH password, so the remote server is going to have must authenticate with a public key.

```
ssh-keygen -t rsa
ssh-copy-id ubuntu@example.com
```

Test your new configuration with `ssh proxy` and you should be able to login without a password prompt. Make sure you can do this now because it'll be harder to diagnose once you begin daemonizing it.

Use systemctl to daemonize the SSH RemoteForward process

In Ubuntu, `systemctl` is ideally suited for this, since it supports waiting until network access is established after boot, and can restart the process when it exits. Create the following config file in `/etc/systemd/system/proxy.service`

```
[Unit]
Description=Proxy
After=network-online.target

[Service]
Restart=always
RestartSec=20
User=ubuntu
ExecStart=/bin/ssh -NT proxy

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

After making the service file, you must reload `systemd`, then start it for the first time and enable it to launch at boot.

```
sudo systemctl daemon-reload
sudo systemctl start proxy
sudo systemctl enable proxy
```

You can diagnose any problems by checking the log with `journalctl -u proxy.service`

Install the latest NGINX on the proxy server

```
sudo add-apt-repository ppa:nginx/stable
sudo apt-get update
sudo apt install nginx
```

Set up host names to forward traffic to the remote computer

Add a new virtual host, in this guide, the proxy server will have the hostname `proxy.example.com` and forward to port 4000. Edit the NGINX config file at `/etc/nginx/sites-enabled/default`

```
server {
  listen 80;
  server_name proxy.example.com;
  location / {
    proxy_pass http://localhost:4000;
  }
}
```

Restart NGINX with `sudo service nginx reload`. You can now go to proxy.example.com and it should forward to port 4000 on the target computer!

Posted on March 18, 2020 in ubuntu, linux, it, networking, ssh

James Loper
896 billion followers

Instagrams

Related

How to tell if a port is in use in Ubuntu

Install Microsoft fonts in Ubuntu

MongoDB setup guide for Ubuntu 20

Fix MacOS mds_stores taking 100% cpu

NGINX cheatsheet for Ubuntu Server 20