How To Configure Nginx as a Web Server and Reverse Proxy for Apache on One Ubuntu 18.04 Server

Posted July 10, 2018 44.4k NGINX APACHE PHP LET'S ENCRYPT UBUNTU 18.04

By: Jesin A

The author selected the Electronic Frontier Foundation to receive a donation as part of the Write for DOnations program.

Introduction

Apache and Nginx are two popular open-source web servers often used with PHP. It can be useful to run both of them on the same virtual machine when hosting multiple websites which have varied requirements. The general solution for running two web servers on a single system is to either use multiple IP addresses or different port numbers.

Servers which have both IPv4 and IPv6 addresses can be configured to serve Apache sites on one protocol and Nginx sites on the other, but this isn't currently practical, as IPv6 adoption by ISPs is still not widespread. Having a different port number like `81` or `8080` for the second web server is another solution, but sharing URLs with port numbers (such as `http://example.com:81`) isn't always reasonable or ideal.

In this tutorial you'll configure Nginx as both a web server and as a reverse proxy for Apache – all on a single server.

Depending on the web application, code changes might be required to keep Apache reverse-proxy-aware, especially when SSL sites are configured. To avoid this, you will install an Apache module called `mod_rpaf` which rewrites certain environment variables so it appears that Apache is directly handling requests from web clients.

We will host four domain names on one server. Two will be served by Nginx: `example.com` (the default virtual host) and `sample.org`. The remaining two, `foobar.net` and `test.io`, will be served by Apache. We'll also configure Apache to serve PHP applications using PHP-FPM, which offers better performance over `mod_php`.

Prerequisites

To complete this tutorial, you'll need the following:

SCROLL TO TOP

- A new Ubuntu 18.04 server configured by following the Initial Server Setup with Ubuntu 18.04, with a sudo non-root user and a firewall.
- Four fully-qualified domain names configured to point to your server's IP address. See Step 3 of How To Set Up a Host Name with DigitalOcean for an example of how to do this. If you host your domains' DNS elsewhere, you should create appropriate A records there instead.

Step 1 — Installing Apache and PHP-FPM

Let's start by installing Apache and PHP-FPM.

In addition to Apache and PHP-FPM, we will also install the PHP FastCGI Apache module, `libapache2-mod-fastcgi` , to support FastCGI web applications.

First, update your package list to ensure you have the latest packages.

```
$ sudo apt update
```

Next, install the Apache and PHP-FPM packages:

```
$ sudo apt install apache2 php-fpm
```

The FastCGI Apache module isn't available in Ubuntu's repository so download it from kernel.org and install it using the `dpkg` command.

```
$ wget https://mirrors.edge.kernel.org/ubuntu/pool/multiverse/liba/libapache-mod-fastcgi/libapache2-m
$ sudo dpkg -i libapache2-mod-fastcgi_2.4.7~0910052141-1.2_amd64.deb
```

Next, let's change Apache's default configuration to use PHP-FPM.

Step 2 — Configuring Apache and PHP-FPM

In this step we will change Apache's port number to `8080` and configure it to work with PHP-FPM using the `mod_fastcgi` module. Rename Apache's `ports.conf` configuration file:

```
$ sudo mv /etc/apache2/ports.conf /etc/apache2/ports.conf.default
```

Create a new `ports.conf` file with the port set to `8080` :

```
$ echo "Listen 8080" | sudo tee /etc/apache2/ports.conf
```

> **Note:** Web servers are generally set to listen on `127.0.0.1:8080` when configuring a reverse proxy but doing so would set the value of PHP's environment variable **SERVER_ADDR** to the loopback IP address instead of the server's public IP. Our aim is to set up Apache in such a way that its websites do not see a reverse proxy in front of it. So, we will configure it to listen on `8080` on all IP addresses.

Next we'll create a virtual host file for Apache. The `<VirtualHost>` directive in this file will be set to serve sites only on port `8080`.

Disable the default virtual host:

```
$ sudo a2dissite 000-default
```

Then create a new virtual host file, using the existing default site:

```
$ sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/001-default.con
```

Now open the new configuration file:

```
$ sudo nano /etc/apache2/sites-available/001-default.conf
```

Change the listening port to `8080`:

/etc/apache2/sites-available/000-default.conf

```
<VirtualHost *:8080>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Save the file and activate the new configuration file:

```
$ sudo a2ensite 001-default
```

Then reload Apache:

```
$ sudo systemctl reload apache2
```

Verify that Apache is now listening on `8080`:

SCROLL TO TOP

```
$ sudo netstat -tlpn
```

The output should look like the following example, with `apache2` listening on `8080`:

```
Output
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address     Foreign Address      State    PID/Program name
tcp        0      0 0.0.0.0:22        0.0.0.0:*            LISTEN   1086/sshd
tcp6       0      0 :::8080           :::*                 LISTEN   4678/apache2
tcp6       0      0 :::22             :::*                 LISTEN   1086/sshd
```

Once you verify that Apache is listening on the correct port, you can configure support for PHP and FastCGI.

Step 3 — Configuring Apache to Use mod_fastcgi

Apache serves PHP pages using `mod_php` by default, but it requires additional configuration to work with PHP-FPM.

> **Note:** If you are trying this tutorial on an existing installation of LAMP with mod_php, disable it first with `sudo a2dismod php7.2`.

We will be adding a configuration block for `mod_fastcgi` which depends on `mod_action`. `mod_action` is disabled by default, so we first need to enable it:

```
$ sudo a2enmod actions
```

Rename the existing FastCGI configuration file:

```
$ sudo mv /etc/apache2/mods-enabled/fastcgi.conf /etc/apache2/mods-enabled/fastcgi.conf.default
```

Create a new configuration file:

```
$ sudo nano /etc/apache2/mods-enabled/fastcgi.conf
```

Add the following directives to the file to pass requests for `.php` files to the PHP-FPM UNIX socket:

/etc/apache2/mods-enabled/fastcgi.conf

```
<IfModule mod_fastcgi.c>
  AddHandler fastcgi-script .fcgi
  FastCgiIpcDir /var/lib/apache2/fastcgi
  AddType application/x-httpd-fastphp .php
```

SCROLL TO TOP

```
  Action application/x-httpd-fastphp /php-fcgi
  Alias /php-fcgi /usr/lib/cgi-bin/php-fcgi
  FastCgiExternalServer /usr/lib/cgi-bin/php-fcgi -socket /run/php/php7.2-fpm.sock -pass-header Autho
  <Directory /usr/lib/cgi-bin>
    Require all granted
  </Directory>
</IfModule>
```

Save the changes and do a configuration test:

```
$ sudo apachectl -t
```

Reload Apache if **Syntax OK** is displayed:

```
$ sudo systemctl reload apache2
```

If you see the warning `Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message.` , you can safely ignore it for now. We'll configure server names later.

Now let's make sure we can serve PHP from Apache.

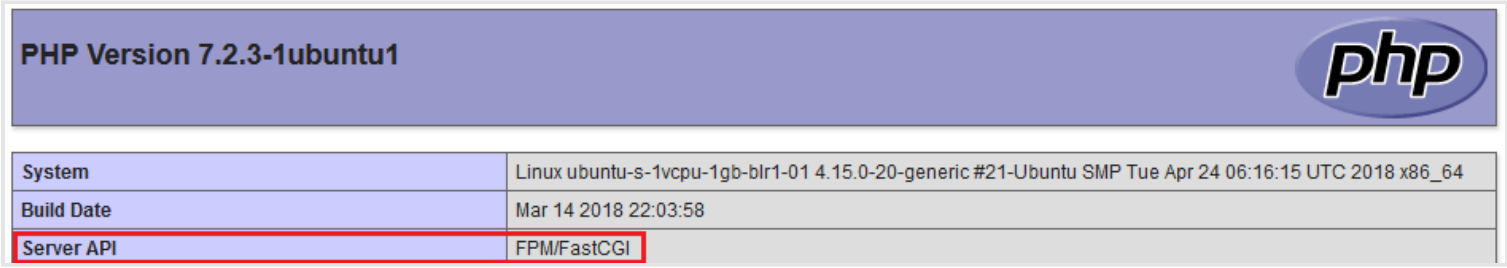
Step 4 — Verifying PHP Functionality

Let's make sure that PHP works by creating a `phpinfo()` file and accessing it from a web browser.

Create the file `/var/www/html/info.php` which contains a call to the `phpinfo` function:

```
$ echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php
```

To see the file in a browser, go to `http://your_server_ip:8080/info.php` . This will give you a list of the configuration settings PHP is using. You'll see output similar to this:



PHP Variables

Variable	Value
$_SERVER['USER']	www-data
$_SERVER['HOME']	/var/www
$_SERVER['ORIG_SCRIPT_NAME']	/php-fcgi
$_SERVER['ORIG_PATH_TRANSLATED']	/var/www/html/info.php
$_SERVER['ORIG_PATH_INFO']	/info.php
$_SERVER['ORIG_SCRIPT_FILENAME']	/usr/lib/cgi-bin/php-fcgi
$_SERVER['SCRIPT_NAME']	/info.php
$_SERVER['REQUEST_URI']	/info.php
$_SERVER['QUERY_STRING']	*no value*
$_SERVER['REQUEST_METHOD']	GET
$_SERVER['SERVER_PROTOCOL']	HTTP/1.0
$_SERVER['GATEWAY_INTERFACE']	CGI/1.1
$_SERVER['REDIRECT_URL']	/info.php
$_SERVER['REMOTE_PORT']	54614
$_SERVER['SCRIPT_FILENAME']	/var/www/html/info.php
$_SERVER['SERVER_ADMIN']	[no address given]
$_SERVER['CONTEXT_DOCUMENT_ROOT']	/usr/lib/cgi-bin/php-fcgi
$_SERVER['CONTEXT_PREFIX']	/php-fcgi
$_SERVER['REQUEST_SCHEME']	http
$_SERVER['DOCUMENT_ROOT']	/var/www/foobar.net
$_SERVER['REMOTE_ADDR']	[redacted]
$_SERVER['SERVER_PORT']	8080
$_SERVER['SERVER_ADDR']	159.89.164.40
$_SERVER['SERVER_NAME']	159.89.164.40
$_SERVER['SERVER_SOFTWARE']	Apache/2.4.29 (Ubuntu)

At the top of the page, check that **Server API** says **FPM/FastCGI**. About two-thirds of the way down the page, the **PHP Variables** section will tell you the **SERVER_SOFTWARE** is Apache on Ubuntu. These confirm that `mod_fastcgi` is active and Apache is using PHP-FPM to process PHP files.

Step 5 — Creating Virtual Hosts for Apache

Let's create Apache virtual host files for the domains `foobar.net` and `test.io`. To do that, we'll first create document root directories for both sites and place some default files in those directories so we can easily test our configuration.

First, create the document root directories:

```
$ sudo mkdir -v /var/www/foobar.net /var/www/test.io
```

Then create an `index` file for each site:

```
$ echo "<h1 style='color: green;'>Foo Bar</h1>" | sudo tee /var/www/foobar.net/index.html
$ echo "<h1 style='color: red;'>Test IO</h1>" | sudo tee /var/www/test.io/index.html
```

Then create a `phpinfo()` file for each site so we can test that PHP is configured properly

SCROLL TO TOP

```
$ echo "<?php phpinfo(); ?>" | sudo tee /var/www/foobar.net/info.php
$ echo "<?php phpinfo(); ?>" | sudo tee /var/www/test.io/info.php
```

Now create the virtual host file for the `foobar.net` domain:

```
$ sudo nano /etc/apache2/sites-available/foobar.net.conf
```

Add the following code to the file to define the host:

/etc/apache2/sites-available/foobar.net.conf

```
<VirtualHost *:8080>
    ServerName foobar.net
    ServerAlias www.foobar.net
    DocumentRoot /var/www/foobar.net
    <Directory /var/www/foobar.net>
        AllowOverride All
    </Directory>
</VirtualHost>
```

The line `AllowOverride All` enables `.htaccess` support.

These are only the most basic directives. For a complete guide on setting up virtual hosts in Apache, see How To Set Up Apache Virtual Hosts on Ubuntu 16.04.

Save and close the file. Then create a similar configuration for `test.io`. First create the file:

```
$ sudo nano /etc/apache2/sites-available/test.io.conf
```

Then add the configuration to the file:

/etc/apache2/sites-available/test.io.conf

```
<VirtualHost *:8080>
    ServerName test.io
    ServerAlias www.test.io
    DocumentRoot /var/www/test.io
    <Directory /var/www/test.io>
        AllowOverride All
    </Directory>
</VirtualHost>
```

Save the file and exit the editor.

Now that both Apache virtual hosts are set up, enable the sites using the `a2ensite` command. This creates a symbolic link to the virtual host file in the `sites-enabled` directory:

SCROLL TO TOP

```
$ sudo a2ensite foobar.net
$ sudo a2ensite test.io
```

Check Apache for configuration errors again:

```
$ sudo apachectl -t
```

You'll see **Syntax OK** displayed if there are no errors. If you see anything else, review the configuration and try again.

Reload Apache to apply the changes once your configuration is error-free:

```
$ sudo systemctl reload apache2
```

To confirm the sites are working, open `http://foobar.net:8080` and `http://test.io:8080` in your browser and verify that each site displays its **index.html** file.

You'll see the following results:

SCROLL TO TOP

Also, ensure that PHP is working by accessing the **info.php** files for each site. Visit `http://foobar.net:8080/info.php` and `http://test.io:8080/info.php` in your browser.

You'll see the same PHP configuration spec list on each site as you saw in Step 4.

We now have two websites hosted on Apache at port `8080`. Let's configure Nginx next.

Step 6 — Installing and Configuring Nginx

In this step we'll install Nginx and configure the domains `example.com` and `sample.org` as Nginx's virtual hosts. For a complete guide on setting up virtual hosts in Nginx, see How To Set Up Nginx Server Blocks (Virtual Hosts) on Ubuntu 18.04.

Install Nginx using the package manager:

```
$ sudo apt install nginx
```

Then remove the default virtual host's symlink since we won't be using it any more:

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

We'll create our own default site later (`example.com`).

Now we'll create virtual hosts for Nginx using the same procedure we used for Apache. First create document root directories for both the websites:

```
$ sudo mkdir -v /usr/share/nginx/example.com /usr/share/nginx/sample.org
```

We'll keep the Nginx web sites in `/usr/share/nginx`, which is where Nginx wants them by default. You could put them under `/var/www/html` with the Apache sites, but this separation may help you associate sites with Nginx.

As you did with Apache's virtual hosts, create `index` and `phpinfo()` files for testing after setup is complete:

```
$ echo "<h1 style='color: green;'>Example.com</h1>" | sudo tee /usr/share/nginx/example.com/index.htm
$ echo "<h1 style='color: red;'>Sample.org</h1>" | sudo tee /usr/share/nginx/sample.org/index.html
$ echo "<?php phpinfo(); ?>" | sudo tee /usr/share/nginx/example.com/info.php
$ echo "<?php phpinfo(); ?>" | sudo tee /usr/share/nginx/sample.org/info.php
```

Now create a virtual host file for the domain `example.com`:

SCROLL TO TOP

```
$ sudo nano /etc/nginx/sites-available/example.com
```

Nginx calls `server {. . .}` areas of a configuration file **server blocks**. Create a server block for the primary virtual host, `example.com`. The `default_server` configuration directive makes this the default virtual host which processes HTTP requests which do not match any other virtual host.

/etc/nginx/sites-available/example.com

```
server {
    listen 80 default_server;

    root /usr/share/nginx/example.com;
    index index.php index.html index.htm;

    server_name example.com www.example.com;
    location / {
        try_files $uri $uri/ /index.php;
    }

    location ~ \.php$ {
        fastcgi_pass unix:/run/php/php7.2-fpm.sock;
        include snippets/fastcgi-php.conf;
    }
}
```

Save and close the file. Now create a virtual host file for Nginx's second domain, `sample.org`:

```
$ sudo nano etc/nginx/sites-available/sample.org
```

Add the following to the file:

/etc/nginx/sites-available/sample.org

```
server {
    root /usr/share/nginx/sample.org;
    index index.php index.html index.htm;

    server_name sample.org www.sample.org;
    location / {
        try_files $uri $uri/ /index.php;
    }

    location ~ \.php$ {
        fastcgi_pass unix:/run/php/php7.2-fpm.sock;
        include snippets/fastcgi-php.conf;
    }
}
```

SCROLL TO TOP

Save and close the file.

Then enable both sites by creating symbolic links to the `sites-enabled` directory:

```
$ sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/example.com
$ sudo ln -s /etc/nginx/sites-available/sample.org /etc/nginx/sites-enabled/sample.org
```

Then test the Nginx configuration to ensure there are no configuration issues:

```
$ sudo nginx -t
```

Then reload Nginx if there are no errors:

```
$ sudo systemctl reload nginx
```

Now access the `phpinfo()` file of your Nginx virtual hosts in a web browser by visiting http://example.com/info.php and http://sample.org/info.php. Look under the PHP Variables sections again.

PHP Variables

Variable	Value
$_SERVER['USER']	www-data
$_SERVER['HOME']	/var/www
$_SERVER['HTTP_ACCEPT']	*/*
$_SERVER['HTTP_USER_AGENT']	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0
$_SERVER['HTTP_HOST']	example.com
$_SERVER['REDIRECT_STATUS']	200
$_SERVER['SERVER_NAME']	example.com
$_SERVER['SERVER_PORT']	80
$_SERVER['SERVER_ADDR']	159.89.164.40
$_SERVER['REMOTE_PORT']	4775
$_SERVER['REMOTE_ADDR']	[redacted]
$_SERVER['SERVER_SOFTWARE']	nginx/1.14.0
$_SERVER['GATEWAY_INTERFACE']	CGI/1.1
$_SERVER['REQUEST_SCHEME']	http
$_SERVER['SERVER_PROTOCOL']	HTTP/1.1
$_SERVER['DOCUMENT_ROOT']	/usr/share/nginx/example.com

["**SERVER_SOFTWARE**"] should say `nginx`, indicating that the files were directly served by Nginx. ["**DOCUMENT_ROOT**"] should point to the directory you created earlier in this step for each Nginx site.

At this point, we have installed Nginx and created two virtual hosts. Next we will configure Nginx to proxy requests meant for domains hosted on Apache.

Step 7 — Configuring Nginx for Apache's Virtual Hosts

Let's create an additional Nginx virtual host with multiple domain names in the `server_name` directives. Requests for these domain names will be proxied to Apache.

Create a new Nginx virtual host file to forward requests to Apache:

SCROLL TO TOP

```
$ sudo nano /etc/nginx/sites-available/apache
```

Add the following code block which specifies the names of both Apache virtual host domains and proxies their requests to Apache. Remember to use the public IP address in `proxy_pass`:

/etc/nginx/sites-available/apache

```
server {
    listen 80;
    server_name foobar.net www.foobar.net test.io www.test.io;

    location / {
        proxy_pass http://your_server_ip:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Save the file and enable this new virtual host by creating a symbolic link:

```
$ sudo ln -s /etc/nginx/sites-available/apache /etc/nginx/sites-enabled/apache
```

Test the configuration to ensure there are no errors:

```
$ sudo nginx -t
```

If there are no errors, reload Nginx:

```
$ sudo systemctl reload nginx
```

Open the browser and access the URL `http://foobar.net/info.php` in your browser. Scroll down to the **PHP Variables** section and check the values displayed.

PHP Variables

Variable	Value
$_SERVER['USER']	www-data
$_SERVER['HOME']	/var/www
$_SERVER['ORIG_SCRIPT_NAME']	/php-fcgi
$_SERVER['ORIG_PATH_TRANSLATED']	/var/www/foobar.net/info.php
$_SERVER['ORIG_PATH_INFO']	/info.php
$_SERVER['ORIG_SCRIPT_FILENAME']	/usr/lib/cgi-bin/php-fcgi
$_SERVER['SCRIPT_NAME']	/info.php
$_SERVER['REQUEST_URI']	/info.php
$_SERVER['QUERY_STRING']	*no value*
$_SERVER['REQUEST_METHOD']	GET
$_SERVER['SERVER_PROTOCOL']	HTTP/1.0
$_SERVER['GATEWAY_INTERFACE']	CGI/1.1
$_SERVER['REDIRECT_URL']	/info.php
$_SERVER['REMOTE_PORT']	54614
$_SERVER['SCRIPT_FILENAME']	/var/www/foobar.net/info.php
$_SERVER['SERVER_ADMIN']	[no address given]
$_SERVER['CONTEXT_DOCUMENT_ROOT']	/usr/lib/cgi-bin/php-fcgi
$_SERVER['CONTEXT_PREFIX']	/php-fcgi
$_SERVER['REQUEST_SCHEME']	http
$_SERVER['DOCUMENT_ROOT']	/var/www/foobar.net
$_SERVER['REMOTE_ADDR']	159.89.164.40
$_SERVER['SERVER_PORT']	80
$_SERVER['SERVER_ADDR']	159.89.164.40
$_SERVER['SERVER_NAME']	foobar.net
$_SERVER['SERVER_SOFTWARE']	Apache/2.4.29 (Ubuntu)
$_SERVER['SERVER_SIGNATURE']	<address>Apache/2.4.29 (Ubuntu) Server at foobar.net Port 80</address>
$_SERVER['PATH']	/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$_SERVER['HTTP_ACCEPT']	*/*
$_SERVER['HTTP_USER_AGENT']	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0
$_SERVER['HTTP_CONNECTION']	close
$_SERVER['HTTP_X_FORWARDED_PROTO']	http
$_SERVER['HTTP_X_FORWARDED_FOR']	[redacted]
$_SERVER['HTTP_X_REAL_IP']	[redacted]

The variables **SERVER_SOFTWARE** and **DOCUMENT_ROOT** confirm that this request was handled by Apache. The variables **HTTP_X_REAL_IP** and **HTTP_X_FORWARDED_FOR** were added by Nginx and should show the public IP address of the computer you're using to access the URL.

We have successfully set up Nginx to proxy requests for specific domains to Apache. Next, let's configure Apache to set the `REMOTE_ADDR` variable as if it were handling these requests directly.

Step 8 — Installing and Configuring mod_rpaf

In this step you'll install an Apache module called `mod\_rpaf` which rewrites the values of **REMOTE_ADDR**, **HTTPS** and **HTTP_PORT** based on the values provided by a reverse proxy. Without this module, some PHP applications would require code changes to work seamlessly from behind a proxy. This module is present in Ubuntu's repository as `libapache2-mod-rpaf` but is outdated and doesn't support certain configuration directives. Instead, we will install it from source.

Install the packages needed to build the module:

SCROLL TO TOP

```
$ sudo apt install unzip build-essential apache2-dev
```

Download the latest stable release from GitHub:

```
$ wget https://github.com/gnif/mod_rpaf/archive/stable.zip
```

Extract the downloaded file:

```
$ unzip stable.zip
```

Change into the new directory containing the files:

```
$ cd mod_rpaf-stable
```

Compile and install the module:

```
$ make
$ sudo make install
```

Next, create a file in the `mods-available` directory which will load the `rpaf` module:

```
$ sudo nano /etc/apache2/mods-available/rpaf.load
```

Add the following code to the file to load the module:

/etc/apache2/mods-available/rpaf.load

```
LoadModule rpaf_module /usr/lib/apache2/modules/mod_rpaf.so
```

Save the file and exit the editor.

Create another file in this directory called `rpaf.conf` which will contain the configuration directives for `mod_rpaf`:

```
$ sudo nano /etc/apache2/mods-available/rpaf.conf
```

Add the following code block to configure `mod_rpaf`, making sure to specify the IP address of your server:

/etc/apache2/mods-available/rpaf.conf

```
<IfModule mod_rpaf.c>
    RPAF_Enable             On
```

SCROLL TO TOP

```
    RPAF_Header             X-Real-Ip
    RPAF_ProxyIPs           your_server_ip
    RPAF_SetHostName        On
    RPAF_SetHTTPS           On
    RPAF_SetPort            On
</IfModule>
```

Here's a brief description of each directive. See the `mod_rpaf` README file for more information.

- **RPAF_Header** - The header to use for the client's real IP address.
- **RPAF_ProxyIPs** - The proxy IP to adjust HTTP requests for.
- **RPAF_SetHostName** - Updates the vhost name so `ServerName` and `ServerAlias` work.
- **RPAF_SetHTTPS** - Sets the `HTTPS` environment variable based on the value contained in `X-Forwarded-Proto`.
- **RPAF_SetPort** - Sets the `SERVER_PORT` environment variable. Useful for when Apache is behind a SSL proxy.

Save `rpaf.conf` and enable the module:

```
$ sudo a2enmod rpaf
```

This creates symbolic links of the files `rpaf.load` and `rpaf.conf` in the `mods-enabled` directory. Now do a configuration test:

```
$ sudo apachectl -t
```

Reload Apache if there are no errors:

```
$ sudo systemctl reload apache2
```

Access the `phpinfo()` pages `http://foobar.net/info.php` and `http://test.io/info.php` in your browser and check the **PHP Variables** section. The **REMOTE_ADDR** variable will now also be that of your local computer's public IP address.

Now let's set up TLS/SSL encryption for each site.

Step 9 — Setting Up HTTPS Websites with Let's Encrypt (Optional)

In this step we will configure TLS/SSL certificates for both the domains hosted on Apache certificates through [Let's Encrypt](https://letsencrypt.org]. Nginx supports SSL terminatio

SSL without modifying Apache's configuration files. The `mod_rpaf` module ensures the required environment variables are set on Apache to make applications work seamlessly behind a SSL reverse proxy.

First we will separate the `server {...}` blocks of both the domains so that each of them can have their own SSL certificates. Open the file `/etc/nginx/sites-available/apache` in your editor:

```
$ sudo nano /etc/nginx/sites-available/apache
```

Modify the file so that it looks like this, with `foobar.net` and `test.io` in their own `server` blocks:

/etc/nginx/sites-available/apache

```
server {
    listen 80;
    server_name foobar.net www.foobar.net;

    location / {
        proxy_pass http://your_server_ip:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
server {
    listen 80;
    server_name test.io www.test.io;

    location / {
        proxy_pass http://your_server_ip:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

We'll use Certbot to generate our TLS/SSL certificates. Its Nginx plugin will take care of reconfiguring Nginx and reloading the config whenever necessary.

First, add the official Certbot repository:

```
$ sudo add-apt-repository ppa:certbot/certbot
```

Press `ENTER` when prompted to confirm you want to add the new repository. Then updat to pick up the new repository's package information:

SCROLL TO TOP

```
$ sudo apt update
```

Then install Certbot's Nginx package with `apt`:

```
$ sudo apt install python-certbot-nginx
```

Once it's installed, use the `certbot` command to generate the certificates for `foobar.net` and `www.foobar.net`:

```
$ sudo certbot --nginx -d foobar.net -d www.foobar.net
```

This command tells Certbot to use the `nginx` plugin, using `-d` to specify the names we'd like the certificate to be valid for.

If this is your first time running `certbot`, you will be prompted to enter an email address and agree to the terms of service. After doing so, `certbot` will communicate with the Let's Encrypt server, then run a challenge to verify that you control the domain you're requesting a certificate for.

Next, Certbot will ask how you'd like to configure your HTTPS settings:

```
Output
Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
-------------------------------------------------------------------------------
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
-------------------------------------------------------------------------------
Select the appropriate number [1-2] then [enter] (press 'c' to cancel):
```

Select your choice, then press `ENTER`. The configuration will be updated, and Nginx will reload to pick up the new settings.

Now execute the command for the second domain:

```
$ sudo certbot --nginx -d test.io -d www.test.io
```

Access one of Apache's domains in your browser using the `https://` prefix; visit `https://foobar.net/info.php` and you'll see this:

PHP Variables

Variable	Value
$_SERVER['USER']	www-data
$_SERVER['HOME']	/var/www
$_SERVER['ORIG_SCRIPT_NAME']	/php-fcgi
$_SERVER['ORIG_PATH_TRANSLATED']	/var/www/foobar.net/info.php
$_SERVER['ORIG_PATH_INFO']	/info.php
$_SERVER['ORIG_SCRIPT_FILENAME']	/usr/lib/cgi-bin/php-fcgi
$_SERVER['SCRIPT_NAME']	/info.php
$_SERVER['REQUEST_URI']	/info.php
$_SERVER['QUERY_STRING']	*no value*
$_SERVER['REQUEST_METHOD']	GET
$_SERVER['SERVER_PROTOCOL']	HTTP/1.0
$_SERVER['GATEWAY_INTERFACE']	CGI/1.1
$_SERVER['REDIRECT_URL']	/info.php
$_SERVER['REMOTE_PORT']	54626
$_SERVER['SCRIPT_FILENAME']	/var/www/foobar.net/info.php
$_SERVER['SERVER_ADMIN']	[no address given]
$_SERVER['CONTEXT_DOCUMENT_ROOT']	/usr/lib/cgi-bin/php-fcgi
$_SERVER['CONTEXT_PREFIX']	/php-fcgi
$_SERVER['REQUEST_SCHEME']	https
$_SERVER['DOCUMENT_ROOT']	/var/www/foobar.net
$_SERVER['REMOTE_ADDR']	[redacted]
$_SERVER['SERVER_PORT']	443
$_SERVER['SERVER_ADDR']	159.89.164.40
$_SERVER['SERVER_NAME']	foobar.net
$_SERVER['SERVER_SOFTWARE']	Apache/2.4.29 (Ubuntu)
$_SERVER['SERVER_SIGNATURE']	<address>Apache/2.4.29 (Ubuntu) Server at foobar.net Port 443</address>
$_SERVER['PATH']	/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$_SERVER['HTTP_ACCEPT']	*/*
$_SERVER['HTTP_USER_AGENT']	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0
$_SERVER['HTTP_CONNECTION']	close
$_SERVER['HTTP_X_FORWARDED_PROTO']	https
$_SERVER['HTTP_X_FORWARDED_FOR']	[redacted]
$_SERVER['HTTP_X_REAL_IP']	[redacted]
$_SERVER['HTTP_HOST']	foobar.net
$_SERVER['HTTPS']	on

Look in the **PHP Variables** section. The variable **SERVER_PORT** has been set to **443** and **HTTPS** set to **on**, as though Apache was directly accessed over HTTPS. With these variables set, PHP applications do not have to be specially configured to work behind a reverse proxy.

Now let's disable direct access to Apache.

Step 10 — Blocking Direct Access to Apache (Optional)

Since Apache is listening on port `8080` on the public IP address, it is accessible by everyone. It can be blocked by working the following IPtables command into your firewall rule set.

```
$ sudo iptables -I INPUT -p tcp --dport 8080 ! -s your_server_ip -j REJECT --reject-with tcp-reset
```

SCROLL TO TOP

Be sure to use your server's IP address in place of the example in red. Once port `8080` is blocked in your firewall, test that Apache is unreachable on it. Open your web browser and try accessing one of Apache's domain names on port `8080`. For example: http://example.com:8080

The browser should display an "Unable to connect" or "Webpage is not available" error message. With the IPtables `tcp-reset` option in place, an outsider would see no difference between port `8080` and a port that doesn't have any service on it.

> **Note:** IPtables rules do not survive a system reboot by default. There are multiple ways to preserve IPtables rules, but the easiest is to use `iptables-persistent` in Ubuntu's repository. Explore this article to learn more about how to configure IPTables.

Now let's configure Nginx to serve static files for the Apache sites.

Step 11 — Serving Static Files Using Nginx (Optional)

When Nginx proxies requests for Apache's domains, it sends every file request for that domain to Apache. Nginx is faster than Apache in serving static files like images, JavaScript and style sheets. So let's configure Nginx's `apache` virtual host file to directly serve static files but send PHP requests on to Apache.

Open the file `/etc/nginx/sites-available/apache` in your editor:

```
$ sudo nano /etc/nginx/sites-available/apache
```

You'll need to add two additional `location` blocks to each server block, as well as modify the existing `location` sections. In addition, you'll need to tell Nginx where to find the static files for each site.

If you've decided not to use SSL and TLS certificates, modify your file so it looks like this:

/etc/nginx/sites-available/apache

```
server {
    listen 80;
    server_name test.io www.test.io;
    root /var/www/test.io;
    index index.php index.htm index.html;

    location / {
        try_files $uri $uri/ /index.php;
    }

    location ~ \.php$ {
        proxy_pass http://your_server_ip:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location ~ /\.ht {
        deny all;
    }
}

server {
    listen 80;
    server_name foobar.net www.foobar.net;
    root /var/www/foobar.net;
    index index.php index.htm index.html;

    location / {
        try_files $uri $uri/ /index.php;
    }

    location ~ \.php$ {
        proxy_pass http://your_ip_address:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

If you also want HTTPS to be available, use the following configuration instead:

/etc/nginx/sites-available/apache

```
server {
    listen 80;
    server_name test.io www.test.io;
    root /var/www/test.io;
    index index.php index.htm index.html;

    location / {
        try_files $uri $uri/ /index.php;
    }

    location ~ \.php$ {
        proxy_pass http://your_server_ip:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location ~ /\.ht {
        deny all;
    }

    listen 443 ssl;
    ssl_certificate /etc/letsencrypt/live/test.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/test.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
    listen 80;
    server_name foobar.net www.foobar.net;
    root /var/www/foobar.net;
    index index.php index.htm index.html;

    location / {
        try_files $uri $uri/ /index.php;
    }

    location ~ \.php$ {
        proxy_pass http://your_ip_address:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location ~ /\.ht {
        deny all;
    }

    listen 443 ssl;
    ssl_certificate /etc/letsencrypt/live/foobar.net/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/foobar.net/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
```

The `try_files` directive makes Nginx look for files in the document root and directly serve them. If the file has a `.php` extension, the request is passed to Apache. Even if the file is not found in the document root, the request is passed on to Apache so that application features like permalinks work without problems.

Warning: The `location ~ /\.ht` directive is very important; this prevents Nginx from ser
of Apache configuration files like `.htaccess` and `.htpasswd` which contain sensitive information.

SCROLL TO TOP

Save the file and perform a configuration test:

```
$ sudo nginx -t
```

Reload Nginx if the test succeeds:

```
$ sudo service nginx reload
```

To verify things are working, you can examine Apache's log files in `/var/log/apache2` and see the `GET` requests for the `info.php` files of `test.io` and `foobar.net`. Use the `tail` command to see the last few lines of the file, and use the `-f` switch to watch the file for changes:

```
$ sudo tail -f /var/log/apache2/other_vhosts_access.log
```

Now visit `http://test.io/info.php` in your browser and then look at the output from the log. You'll see that Apache is indeed replying:

```
Output
    test.io:80 your_server_ip - - [01/Jul/2016:18:18:34 -0400] "GET /info.php HTTP/1.0" 200 20414 "-'
```

Then visit the `index.html` page for each site and you won't see any log entries from Apache. Nginx is serving them.

When you're done observing the log file, press `CTRL+C` to stop tailing it.

With this setup, Apache will not be able to restrict access to static files. Access control for static files would need to be configured in Nginx's `apache` virtual host file, but that's beyond the scope of this tutorial.

Conclusion

You now have one Ubuntu server with Nginx serving `example.com` and `sample.org`, along with Apache serving `foobar.net` and `test.io`. Though Nginx is acting as a reverse-proxy for Apache, Nginx's proxy service is transparent and connections to Apache's domains appear be served directly from Apache itself. You can use this method to serve secure and static sites.

By: Jesin A

Editor:
Brian Hogan

Related Tutorials

How To Create Temporary and Permanent Redirects with Apache and Nginx

How To Sync and Share Your Files with Seafile on Debian 9

How To Deploy a PHP Application with Kubernetes on Ubuntu 16.04

How To Ensure Code Quality with SonarQube on Ubuntu 18.04

How To Set Up a Private Docker Registry on Ubuntu 18.04

11 Comments

Leave a comment...

SCROLL TO TOP

Log In to Comment

dhezineadvertising *July 16, 2018*

0 Hi, great article.

I was successful configuring nginx as a reverse proxy by following this article, but i use 127.0.0.1 as a value of proxy*pass. I don't have a static public IP, so i'm using no-ip. Is it possible to make SERVER*ADDR returns my public ip address instead of loopback? How do i configure it and also configure rpaf mod RPAF_ProxyIPs?

Thank you.

---

a4amk *July 24, 2018*

0 I followed all the steps but I'm not able to access http://my_ip:8080/info.php and I want to host only one Wordpress Website, I do not have 4 domains or I think I'm having problem in understanding the guide.

Leez0r *August 8, 2018*

0 try `sudo ufw disable`

Leez0r *August 8, 2018*

0 This tutorial doesn't work if you allowed ubuntu firewall in the initial setup.

I've tried `sudo ufw enable Apache` but that didn't work...

Only thing that worked for me was disabling the firewall completely (which is ok for me) by `sudo ufw disable`

---

dingleling *August 9, 2018*

0 Ok, so I have been working on this for awhile now. I got everything up and running, its looking good. The last step, getting the SSL up is giving me an error. One that I missed earlier.

sudo nginx -t gives me:
conflicting server name "domain.com" on 0.0.0.0:80, ignored which is not letting me install my SSL. Been at it for awhile, brain hurts. Tried removing extra domains listed in the conf files nothing. It looks like solved most peoples issues, but not sure if they were configuring reverse proxy.

Please help!

ChinmayB *September 3, 2018*

0 If you are following up this instruction, do change

FastCgiExternalServer /usr/lib/cgi-bin/php-fcgi -socket /run/php/php7.2-fpm.sock -pass-header Authorization
to
FastCgiExternalServer /usr/lib/cgi-bin/php-fcgi -socket /run/php/php7.0-fpm.sock -pass-header Authorization

SCROLL TO TOP

foodiezalpha *September 21, 2018*

0 what ubuntu version do you use?

niyazitoros *October 8, 2018*

0 Hi,

I dont use Apache2.Follow previos tutorial and my Ubuntu 18.04 machine I install nginx and configure the block. I also use certbot and get the SSL certificate from Letsencrypt. But this tutoarial is very very advance for me :)

I am using dart.io aqueduct.io server as middleware for json file. I need to connect to nginx on 443 and use reverse proxy to http:127.0.0.1:8888/. How do I do that?

My aqueduct server on http:127.0.0.1:8888/ is working. my domain on nginix also works. how to connect nginx and dart aqueduct server on ubuntu using the nginx reverse proxy?

nginx:443 https://mydomain.com --------> **to aqueductserver http**://127.0.0.1:8888/
aqueductserver http://127.0.0.1:8888/ ----------> **nginx:443 https**://mydomain.com

Help apriciated.

niyazi@niyazi-virtual-machine:~$ sudo netstat -tlpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

***tcp 0 0 0.0.0.0:8888 0.0.0.0:* LISTEN 10931/dart ** *
tcp 0 0 127.0.0.1:7001 0.0.0.0:* LISTEN 3383/nxnode.bin

tcp 0 0 0.0.0.0:443 0.0.0.0:* LISTEN 7896/nginx: master

tcp6 0 0 :::443 :::* LISTEN 7896/nginx: master

tcp6 0 0 :::80 :::* LISTEN 7896/nginx: master

Goodvalley *October 30, 2018*

0 Hi jesin,

I've been following these tutorials on Nginx as a reverse proxy for Apache since 2015, so I've tried for each Ubuntu version, 14.04, 16.04 and now 18.04.

All have been very good, but each of them has the same problem: in the PhP Variables section, `$_SERVER['SERVER_PORT']` is supposed to be `443` and `$_SERVER['HTTPS']` must be set to 'on'. They seem to be wrong. The first one is port `80` and the second is simply not there.

As a matter of fact, all is fine and the websites work flawlessly, so it seems there's no real problem. Looking at PhP Variables, I can see that `$_SERVER['SERVER_SIGNATURE']` is set to `<address>Apache/2.4.37 (Ubuntu) Server at foobar.net Port 80</address>`, `$_SERVER['HTTP_CONNECT` `close` and `$_SERVER['HTTP_X_FORWARDED_PROTO']` is set to `https`.

SCROLL TO TOP

So it 'should' work and it does work, I even got an AA+ SSL qualification for a real domain.

But I just couldn't get it out of my head, so I tried to know what's going on. I've tried everything, even having Apache with port 8080 and Nginx with only port 443, no port 80 anywhere at all.

Do you know what's going on?
Thanks in advance.

---

alexsasharegan *December 22, 2018*

0 Things are mostly working fine for me on this, except I continue to get a download prompt for the mime-type `application/x-httpd-fastphp`. I have the mods-enabled/fastcgi.conf file setup, and the fastcgi module is definitely loaded.

The only curious thing I've found thus far is that the first path reference here doesn't contain `php-fcgi`. I see the socket file at runtime though, so I don't know what this means.

```
FastCgiExternalServer /usr/lib/cgi-bin/php-fcgi -socket /run/php/php7.2-fpm.sock -pass-hea
```

`sudo find / -name 'php-fcgi'` returns nothing.

---

alexsasharegan *December 22, 2018*

0 Follow up: *everything was working*. Just be sure you check all the `.htaccess` files in you site to make sure they don't set some conflicting `AddHandler` directives like I ran into!

SCROLL TO TOP