KBeezie There's no place like ::1

Apache and Nginx Together

I'm not going to get into a lot of details about how to install and configure either http server from scratch. This article is primarily going to be food for thought for those who may want or need to configure nginx along side an existing apache (httpd) configuration. If you would rather ditch Apache all together, consider reading Martin Fjordvald's Nginx Primer 2: From Apache *to* Nginx [http://blog.martinfjordvald.com/2011/02/nginx-primer-2-from-apache-to-nginx/] .

Side by Side

In some cases you may need to run both Apache (httpd) and Nginx on port 80. Such a situation can be a server running Cpanel/Whm and as such doesn't support nginx, so you wouldn't want to mess with the apache configuration at all.

To do this you have to make sure Apache and Nginx are bound to their own IP adddress, In the event of WHM/Cpanel based webserver, you can Release an IP to be used for Nginx in WHM [http://kbeezie.com/view/releasing-ip-addresses-in-whmcpanel/] . At this time I am not aware of a method of reserving an IP, and automatically forcing Apache to listen on a specific set of IPs in a control panel such as DirectAdmin or Plesk. But the link above will show you how with WHM/Cpanel.

Normally Apache will be listening on all interfaces, and such you may see a line like this in your httpd.conf file:

```
Port 80
#or
Listen 80
```

Instead you'll need to tell apache to listen on a specific IP (you can have multiple Listen lines for multiple IPs)

```
Listen 192.170.2.1:80
```

When you change Apache to listen on specific interfaces some VirtualHost may fail to load if they are listening on specific IPs themselves. The address option in <VirtualHost addr[:port] [addr[:port]] …> only applies when the main server is listening on all interfaces (or an alias of such). If you do not have an IP Listed in your <VirtualHost …> tag, then you do not need to worry bout this. Otherwise make sure to remove the addr:port from the VirtualHost tag.

Make sure to update the NameVirtualHost directive [http://httpd.apache.org/docs/2.0/mod/core.html#namevirtualhost] in Apache for name-based virtual hosts.

Once you have apache configured to listen on a specific set of IPs you can do the same with nginx.

```
server {
    listen 192.170.2.2:80;
    ...
}
```

Now that both servers are bound to specific IPs, both can then be started up on port 80. From there you would simply point the IP of the domain to the server you wish to use. In the case of WHM/Cpanel you can either manually configure the DNS entry for the domain going to nginx in WHM, or you can use your own DNS such as with your registrar to point the domain to the specific IP. Using the latter may break your mail/ftp etc configurations if the DNS entries are not duplicated correctly.

Though normally if you are setting up Nginx side-by-side with apache, you'll have your domain configurations separate from the rest of the system. The above paragraph only applies if the domain you are using has already been added by cpanel/whm and you wish to have it served by nginx exclusively. [Scroll down for PHP considerations]

Apache behind Nginx

If you are using a control panel based hosting such as cpanel/whm, this method is not advised. Most of the servers configuration is handled automatically in those cases, and making manual changes will likely lead to problems (plus you won't get support from the control panel makers or your hosting provider in most cases).

Using Nginx as the primary frontend webserver can increase performance regardless if you choose to keep Apache running on the system. One of Nginx's greatest advantage is how well it serves static content. It does so much more efficiently than Apache, and with very little cost to memory or processing. So placing Nginx in front will remove that burdern off Apache, leaving it to concentrate on dynamic request or special scenarios.

This method is also popular for those who don't want to use PHP via fastcgi, or install a separate php-fpm process.

First thing that needs to be done is to change the interface apache listens on:

```
Listen 127.0.0.1:8080
```

Above we bind Apache to the localhost on an alternate port, since only Nginx on the same machine will be communicating with Apache.

Note: Since Nginx needs to access the same files that Apache serves, you need to make sure that Nginx is setup to run as the same user as apache, or to make sure that the Nginx's selected user:group has permission to read the web documents. Often times the webroot is owned by nobody or www-data and Nginx likewise can be setup to run as those users

Then in Nginx listening on port 80 (either on all interfaces such as 0.0.0.0 or to specific IPs, your choice) we would need to configure Nginx to serve the same content. Take for example this virtual host in an apache configuration:

```
<VirtualHost>
      DocumentRoot "/usr/local/www/mydomain.com"
      ServerName mydomain.com
      ServerAlias www.mydomain.com
      CustomLog /var/log/httpd/mydomain_access.log common
      ErrorLog /var/log/httpd/mydomain_error.log
      ...
</VirtualHost>
```

In Nginx the equivalent server configuration would be:

```
server {
      root /usr/local/www/mydomain.com;
      server_name mydomain.com www.mydomain.com;

      # by default logs are stored in nginx's log folder
      # it can be changed to a full path such as /var/log/...
      access_log logs/mydomain_access.log;
      error_log logs/mydomain_error.log;
      ...
}
```

The above would serve all static content from the same location, however PHP will simply come back as PHP source. For this we need to send any PHP requests back to Apache.

```
server {
      root /usr/local/www/mydomain.com;
      server_name mydomain.com www.mydomain.com;

      access_log logs/mydomain_access.log;
      error_log logs/mydomain_error.log;

      location / {
            # this is the location block for the document root of this vhost
      }

      location ~ \.php {
            # this block will catch any requests with a .php extension
            # normally in this block data would be passed to a FastCGI process

            # these two lines tell Apache the actual IP of the client being forwarded
            # Apache requires mod_proxy (http://bit.ly/mod_proxy) for these to work
            # Most Apache 2.0+ servers have mod_proxy configured already

            proxy_set_header X-Real-IP  $remote_addr;
            proxy_set_header X-Forwarded-For $remote_addr;

            # this next line adds the Host header so that apache knows which vHost to serve
            # the $host variable is automatically set to the hostname Nginx is responding to

            proxy_set_header Host $host;
```

```
            # And now we pass back to apache
            # if you're using a side-by-side configuration the IP can be changed to
            # apache's bound IP at port 80 such as http://192.170.2.1:80

            proxy_pass http://127.0.0.1:8080;
      }

       # if you don't like seeing all the errors for missing favicon.ico in root
       location = /favicon.ico { access_log off; log_not_found off; }

       # if you don't like seeing errors for a missing robots.txt in root
       location = /robots.txt { access_log off; log_not_found off; }

       # this will prevent files like .htaccess .htpassword .secret etc from being served
       # You can remove the log directives if you wish to
       # log any attempts at a client trying to access a hidden file
       location ~ /\. { deny all; access_log off; log_not_found off; }
}
```

In the case of rewriting (mod_rewrite) with apache behind nginx you can use an incredibly simple directive called try_files. Here's the same code as above, but when changes made so that any files or folder not found by nginx gets passed to apache for processing (useful for situations like WordPress, or .htaccess based rewrites when the file or folder is not caught by Nginx)

```
server {
      root /usr/local/www/mydomain.com;
      server_name mydomain.com www.mydomain.com;

      access_log logs/mydomain_access.log;
      error_log logs/mydomain_error.log;

      location / {
            # try_files attempts to serve a file or folder, until it reaches the fallback at the end
            try_files $uri $uri/ @backend;
      }

      location @backend {
            # essentially the same as passing php requests back to apache

            proxy_set_header X-Real-IP  $remote_addr;
            proxy_set_header X-Forwarded-For $remote_addr;
            proxy_set_header Host $host;
            proxy_pass http://127.0.0.1:8080;
      }
      # ... The rest is the same as the configuration above ...
}
```

With the above configuration, any file or folder that exists will be served by Nginx (the php block will catch files ending in .php), otherwise it will be passed back to apache on the backend.

In some cases you may wish for everything to be passed to Apache unless otherwise specified, in which case you would have a configuration such as this:

```
server {
      root /usr/local/www/mydomain.com;
      server_name mydomain.com www.mydomain.com;

      access_log logs/mydomain_access.log;
      error_log logs/mydomain_error.log;

      location / {
            proxy_set_header X-Real-IP  $remote_addr;
            proxy_set_header X-Forwarded-For $remote_addr;

            proxy_set_header Host $host;
```

```
            proxy_pass http://127.0.0.1:8080;
        }

        # Example 1 - Specify a folder and it's content
        # To serve everything under /css from Nginx-only
        location /css { }

        # Example 2 - Specify a RegEx pattern such as file extensions
        # Serve certain files directly from Nginx
        location ~* ^.+\.(jpg|jpeg|gif|png|css|zip|pdf|txt|js|flv|swf|html|htm)$
        {
            # this will basically match any file of the above extensions
            # course if the file does not exist you'll see an Nginx error page as opposed
            # to apache's own error page.
        }
}
```

Personally I prefer to leave Nginx to handle everything, and specify exactly what needs to be passed back to Apache. But the last configuration above would more accurately allow .htaccess to function during almost-every request since almost everything gets passed back to apache. The css folder or any files found by the other location block would be served directly by nginx regardless if there's an .htaccess file there, since only Apache processes .ht* files.

PHP Considerations

If you are not familiar with Nginx, then it should be noted that Nginx does not have a PHP module like apache's mod_php, instead you either need to build PHP with FPM (ie: php-fpm/fastcgi), or you need to pass the request to something that can handle PHP.

In the case of using Nginx with Apache you essentially have two choices:

1) Compile and Install PHP-FPM, thus running a seperate PHP process to be used by Nginx. This option is usually good if you want to keep the two webserver configurations separated from each other, that way any changes to one won't affect the other. But of course it adds additional memory usage to the system.

2) Utilize Apache's mod_php. This option is ideal when you will be passing data to apache as a backend. Even in a side-by-side scenario, you can utilize mod_php by proxying any php request to Apache's IP. But in the side-by-side scenario you have to make sure that Apache is also configured to serve the same virtualhost that Nginx is requesting.

Speed-wise both methods are about the same when PHP is configured with the same set of options on both. Which option you choose depends solely on your needs. Another article that may be of interest in relations to this one would be Nginx as a Proxy to your Blog [http://kbeezie.com/view/nginx-proxy-change-ip/] , which can be just as easily utilized on a single server (especially if you get multiple IP ranges like I do).

- Previous Entry: Securing a Thumb Drive with TrueCrypt [http://kbeezie.com/secure-thumb-drive/]
- Next Entry: Protecting Folders with Nginx [http://kbeezie.com/protecting-folders-with-nginx/]

Posted in Apache , Nginx , Webservers
Tags: Apache httpd Nginx
You can follow any responses to this entry through the RSS 2.0 Feed . Both comments and pings are currently closed.

Comments are closed.

Back to Top