# Varnish, nginx and Apache...

erstellt von Tobias Frei — zuletzt verändert: 02.06.2013 11:04

**...without using nginx or Apache as a reverse proxy!**

Varnish is a nice accelerator; why should I tell nginx to act as a reverse proxy when I only need some VCLmagic to deliver the requests to the right backends? ;-)

## The basic idea

```
<ToBeFree> I have a funny idea right now
<ToBeFree> installing nginx, giving it my Apache website root as website root
<ToBeFree> and telling Varnish to redirect *.png and stuff like that to it
<ToBeFree> if this works, it's *the* solution, isn't it xD
<ToBeFree> no additional website space, no restructuring needed,
<ToBeFree> absolutely safe because nginx would be configured to be unable to serve dynamic content, leaving it unvulnerable to whatever attacks one might thi
<TuxBot> <Stary2001@EsperNet> :o
<TuxBot> <Stary2001@EsperNet> that is genius
<ToBeFree> \o/
<ToBeFree> I'm doing something like that for Plone already
<ToBeFree>     if (req.url ~ "^/plone/" || req.url ~ "^/plone$") {
<ToBeFree>         set req.backend = plone;
<ToBeFree> that for *.png... It won't even break mod_pagespeed, I think
<ToBeFree> sadly I need to sleep now, but I'll try this tomorrow xD
```

## Limitations

I didn't expect it, but it actually broke mod_pagespeed because it modifies the URLs in <img>-tags to "optimize" them (expiration etc.). That's not really a problem for me, however, and I can finally get rid of that third-party repository in my sources.list; nginx is even faster than Apache+mod_pagespeed at serving images. Also, simply implementing my old idea 1:1 is a security risk because it would deliver .php files as plain text if no precaution is taken - relying on the VCL to prevent this isn't something you should do. Nginx needs to block these requests itself.

## Configuring nginx

Here's my nginx.conf:

```
user www-data;
worker_processes 4;
pid /run/nginx.pid;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 0; # important! Keepalive kills the whole setup because the browser will also try to get dynamic content using that connection.
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        ##
        # Logging Settings
        ##

        access_log /tmp/nginx-access.log; # placing this in /tmp further improves the performance.
        error_log /var/log/nginx/error.log; # shouldn't be too much input here, anyway.

        ##
        # Gzip Settings
        ##

        gzip on; # actually, we don't really want to serve content which can be gzipped efficiently, but if you think it improves the performance, ...

        # gzip_vary on;
        # gzip_proxied any;
        gzip_comp_level 9;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types *; # ..., you have to enable this, too. Warning: I didn't test if this "*" works and matches all types, or if it does nothing at all.

        ##
        # nginx-naxsi config
        ##
        # Uncomment it if you installed nginx-naxsi
        ##

        #include /etc/nginx/naxsi_core.rules;

        ##
        # nginx-passenger config
        ##
        # Uncomment it if you installed nginx-passenger
        ##

        #passenger_root /usr;
        #passenger_ruby /usr/bin/ruby;

        ##
        # Virtual Host Configs
        ##

        #include /etc/nginx/conf.d/*.conf;
        #include /etc/nginx/sites-enabled/*;    # no, that's silly in my opinion. It might be nice for Apache, but here, it is simply confusing.

        error_page 404 /404.png; # we only serve images and stuff like that - it's always good to have a 404 image file!
        expires 1M; # we only serve static content - no problem to set this to a month.

        server {
                listen          8428;
                server_name     freiwuppertal.de;
                index           http://freiwuppertal.de/404.png; # if we get a request for a directory, something is going wrong or someone found a way to ev
                root            /var/www/webseiten;
                include         /etc/nginx/locationblocker.include; # special thanks to jaybe from #nginx on Freenode who gave me that hint :-)
        }
        server {
                listen          8428;
                server_name     musik.freiwuppertal.de;
                index           http://freiwuppertal.de/404.png;
                root            /var/www/musik;
                include         /etc/nginx/locationblocker.include;
        }
}
```

Now the /etc/nginx/locationblocker.include file:

```
location ~ \.(webp|png|jpg|gif|mp3|ogg|wav|flac|zip|gz|tgz|xz|bz2|exe|js|txt|pdf|css)$ {
  allow all;
}

location / {
  deny all;
}
```

Isn't that a nice way to add this configuration block to every server{}?^^

We need to tell Apache to disable keepalive, too; a simple "KeepAlive Off" in the main apache2.conf works.

Finally, the VCL magic:

```
backend default {
    .host = "127.0.0.1";
    .port = "8008";
    .connect_timeout = 60s;
    .first_byte_timeout = 60s;
    .between_bytes_timeout = 60s;
}
backend nginx {
    .host = "127.0.0.1";
    .port = "8428";
    .connect_timeout = 60s;
    .first_byte_timeout = 60s;
    .between_bytes_timeout = 60s;
}
backend plone {
    .host = "127.0.0.1";
    .port = "8080";
    .connect_timeout = 60s;
    .first_byte_timeout = 60s;
    .between_bytes_timeout = 60s;
}
```

Plone is a good example for a backend which should be excluded from that nginx stuff. This is quite simple:

```
# (this needs to be in sub vcl_recv{})

    if (req.url ~ "^/plone/" || req.url ~ "^/plone$") {
        set req.backend = plone;
    } elsif (req.url ~ "\.(webp|png|jpg|gif|mp3|ogg|wav|flac|zip|gz|tgz|xz|bz2|exe|js|txt|pdf|css)$") {
        set req.backend = nginx;
    } else {
        set req.backend = default;
    }
```

Restart Varnish, Apache and nginx - and everything should work as intended! :D

**[Update]**

Added .js, .txt, .pdf and .css because those should be safe to send via nginx.