NGINX for Windows - Documentation

Not all webservers perform as equal

This document describes all functions which are unique or different from the Linux version, general tips and examples.

Revision: 7 April 2023 (1.9)

Homepage: http://nginx-win.ecsds.eu/

Many more examples can be found in the prove*.zip and ngxLuaDB*.zip archives on site, see the tconf/ or ngxLuDB/conf folder.

Content

1. Preface: [nginx for Windows]

April 7, 2023, It's been awhile since we've been here (this document), there hasn't been enough time to keep up with the rest next to coding and porting challenges, we managed to stay on top developing, hardening and merging code because we consider runtime quality and security our top priority, the rest can wait.

So whats up since 2015 doc?
Humans don't live forever… we've lost family, programmers, developers, testers and Corona didn't help much either… But hey! Those who are alive are still here including new members, a solid customer base and many new use cases.
New code, faster code.
An ever changing Cloud environment where we managed to scale along with (including our own services).

Step by step we're catching up over 8 years on everything except the product that never suffered from whatever got thrown at us.

Let's iterate this again, as it seems it is assumed we register everywhere to be informed, we do get informed by relevant parties but we don't register anywhere <period>:
CVE: any security issues such as vulnerabilities should be reported by email support@ecsystems.nl (start the subject line with "CVE:"), a security engineer ticket will be created and the issue dealt with a.s.a.p.

March 19, 2015, we have made the decision to split away from the original nginx code base.

This has been coming for a while, original nginx code which is absolutely **not** compatible (or better described, not suitable for) with Windows, we've been re-engineering a number of changesets to deal with this and maintain the original code, today (19-3) we have decided to stop re-engineering and have laid out our own roadmap. What exactly this is going to mean for functions and features between nginx Linux and our nginx for Windows is jet unknown.

Of course we will do our best to incorporate new features but we are also aware that this might not always be possible. This also affects new features in the add-ons we use.

The roadmap we have setup will focus on;

- Adding more security measures against any possible way of attack
- Our to-do list
- Getting every possible problem documented and solved (we're doing this already)
- More non-blocking out of the box interfaces, for example Java and BI with NetWeaver for SAP and Oracle, NetAPP, TSM

See also: http://nginx-win.ecsds.eu/anythingispossible.html

nginx 1.9.x

During re-factoring nginx for Windows we've switched code base which makes it easier for us to import original nginx code without Windows issues by using a new native linux <> windows low level API which natively deals with spinlock, mutex locking, Windows event driven technology and full thread separation.

nginx 1.9 is the first such release, for the time being the current 1.7 release will be kept up to date with critical patches and fixes only, no new functions will be added or imported. LTS versions are not affected.

Currently (June 2016) we use 99% of the original codebase/features, amended/adjusted and corrected to suite our requirements.

2. Additional custom 503 error handler via 513

Issue: a "return 503" can only be used once in a location block, when a custom 503 is used for example with limit_req_zone you can't have a second custom 503 for a maintenance page.

Added in 9-3-2014 nginx 1.5.12.1 Cheshire

Example:

```
server {
    listen  80;
    server_name  www.any.nl;
    root    '/webroot/www.any.nl';
    error_page 503 @floodnotice;
    error_page 513 @maintenance;
    location / {
        if (-f $document_root/maintenance_mode.html) { return 513; }
        # Or with pure Lua, no IF issues
        ## rewrite_by_lua '
        ##   local s = 0; local v = 0;
        ##   local source_fname = ngx.var.document_root .. "/maintenance_mode.html";
        ##   local file = io.open(source_fname);
        ##   if file then v=1; file:close(); end;
        ##   if string.find(ngx.var.remote_addr, "^10.10.30.") then v=0; end;
        ##   if v>0 then return ngx.exit(513); end;
        ## ';
        try_files $uri $uri/ =404;
        index  index.html index.htm;
        limit_req zone=floodh burst=32 nodelay;
        # generates a 503 when triggered
        # see limit_req_zone directive how limit_req works
    }
    location @floodnotice {
        root html
        rewrite ^ /floodnotice.html break;
    }
    location @maintenance {
        rewrite ^ /maintenance_mode.html break;
        # process a 513 but return a 503 to client !
    }
}
```

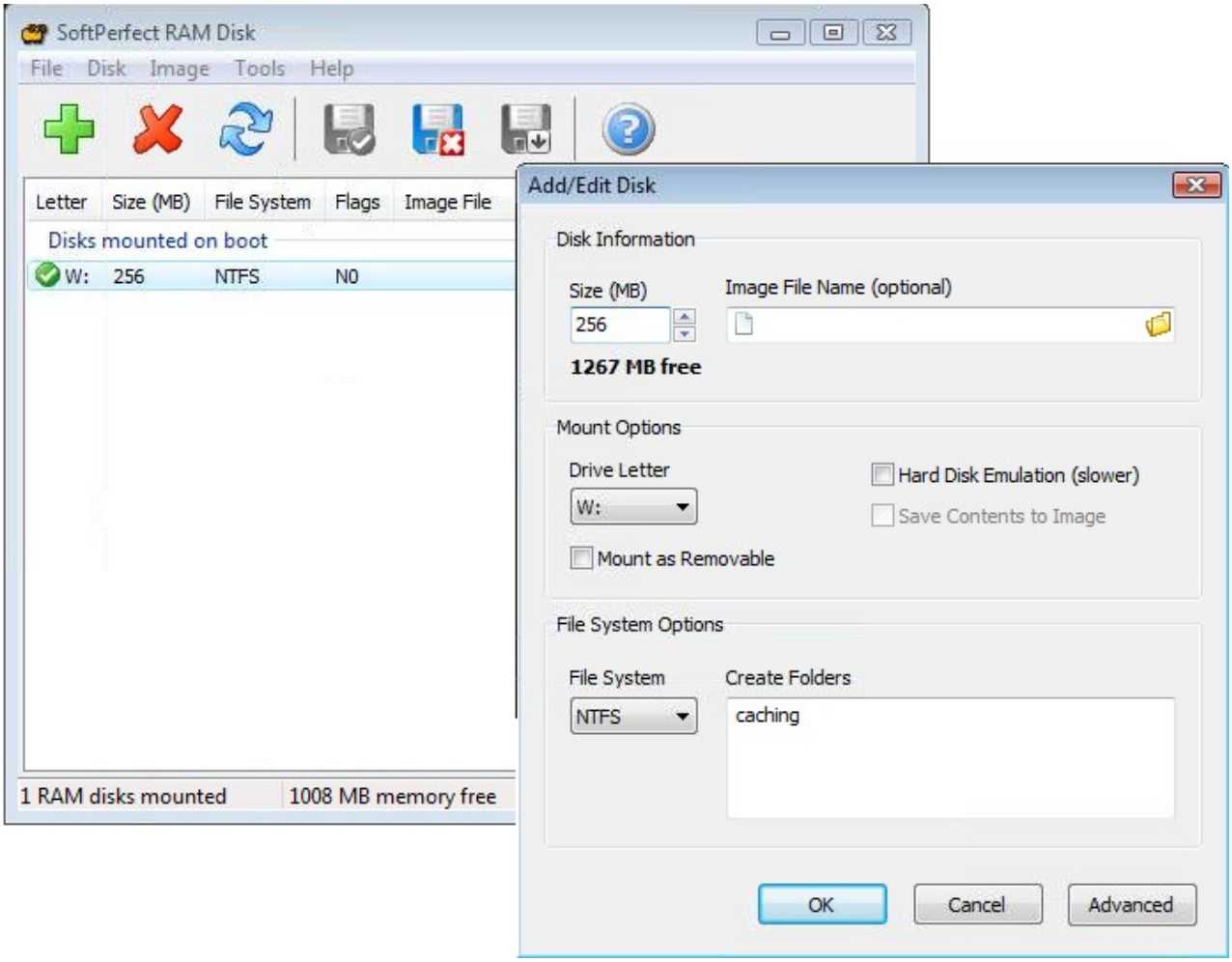
The normal behavior would be (if the file exists) to return the contents of "/maintenance_mode.html" with a "HTTP/1.1 200 OK", or when the 503 error_page is used a 503, however a 503 is often used for other things.

With this new 513 error_page the same thing can be done but the 513 is replaced with a 503 when the headers are compiled which allows you to use the real 503 for other things.

3. Micro caching

Speed up dynamic access for many concurrent users with a micro cache.

Get '**ramdisk_setup v3.4.6.exe**' on site, install a small **ramdrive** (128-256mb) no-compression / NTFS, assign a drive letter like **W:** to it, add (in advanced) **caching** as path to create at boot.



Example:

```
http {
……..

fastcgi_cache_path w:/caching/fastcgi_cache levels=1:2 keys_zone=microcache:10m max_size=1024m inactive=4h;
fastcgi_temp_path w:/caching/fastcgi_temp;

   server {
……

      location ~ \.php$ {
         try_files $uri $uri/ =404;
……
         fastcgi_ignore_client_abort on;
         fastcgi_pass   myLoadBalancer;
         fastcgi_index  index.php;
         fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
         include        fastcgi_params;
         #Caching microcache parameters
           fastcgi_cache microcache;
           fastcgi_cache_key $scheme$host$request_uri$request_method;
           fastcgi_cache_valid 200 301 302 304 5s;
           fastcgi_cache_use_stale updating error timeout invalid_header http_500;
           fastcgi_pass_header Set-Cookie;
           fastcgi_pass_header Cookie;
           fastcgi_ignore_headers Cache-Control Expires Set-Cookie;
         #Auto Purge the cache
           fastcgi_cache_purge PURGE from 127.0.0.1;
         #Caching microcache parameters
      }
   }
}
```

How does it work? Quite simple, the reply of each request to a backend is stored as a file on disk (*), if such an identical request is received within the cache-valid period (5 seconds) the cache file is returned instead of asking the backend to basically return the same answer again.

This also works for proxy_pass, just change fastcgi* to proxy_pass*

(*) Nb. make sure your anti-virus software is **excluded** from scanning your disk cache location(s).

Nb2. **Inactive** should always be larger than any **valid** value and at least 4 hours.

Nb3. Always make sure **….cache_path** and **….temp_path** are two **different** paths on the **same** drive !

Nb4. *“ignore long locked inactive cache entry”:* Make sure talking to a backend always takes **less** time than **expected** cache entry lifetime.

4. Apache migration tips

Note: a redirect is for the client, it is a common misconception a redirect is a server item.

Pitfall-1: redirect loops, always double check where you are redirecting to is **not** where you are coming from.

Pitfall-2: always use a 302 unless you are absolutely 1000% sure the redirect won't change, a 301 (permanent), once a 301 is used it is near enough impossible to change it (again).

Apache:

```
*VirtualHost 127.0.0.100:812*
ServerName www.mydomain.eu
Redirect / http://www.mynewdomain.eu/new-path/new-destination
*/VirtualHost*
```

nginx:

```
server {
    listen 812;
    server_name www.mydomain.eu;
    return 302 http://www.mynewdomain.eu/new-path/new-destination;
}
```

Apache:

```
*VirtualHost 127.0.0.100:966*
ServerName myotherdomain.eu
ProxyRequests On
ProxyPass / http://192.168.1.33:80/path/
ProxyPassReverse / http://192.168.1.33:80/path/
*/VirtualHost*
```

nginx;

```
server {
    listen 996;
    server_name myotherdomain.eu;
    location / {
    [1] proxy_pass http://192.168.1.33:80/path/;
    [3] include c:/nginx/conf/proxy.conf;
    [3] keepalive_requests 500;
    [3] proxy_http_version 1.1;
    [3] proxy_ignore_client_abort on;
    [2] rewrite /path/([^/]+) /$1 break;
    }
}
```

[1] It might be that url http://myotherdomain.eu:996/bla becomes: http://myotherdomain.eu:996/path/bla
experiment with the ending / (remove or add) in proxy_pass [1] to see how the url is passed on,
use the [2] rewrite line to strip portions of the passed url if needed.
[3] keepalive/httpversion/ignoreclient: are values a backend might need or not.

Apache:

VirtualHost 127.0.0.1:123
ServerName www.mydomain.eu
ProxyRequests On
ProxyPass / http://192.168.1.2:830/
ProxyPassReverse / http://192.168.1.2:830/
ProxyPass /path http://192.168.1.222:80/
ProxyPassReverse /path http://192.168.1.222:80/
/VirtualHost

nginx:

```
server {
   listen 123;
   server_name www.mydomain.eu;
   location / {
      proxy_pass http://192.168.1.2:830;
      include c:/nginx/conf/proxy.conf;
      keepalive_requests 500;
      proxy_http_version 1.1;
      proxy_ignore_client_abort on;
   }
   location /path {
      proxy_pass http://192.168.1.222:80/path;
      include c:/nginx/conf/proxy.conf;
      keepalive_requests 500;
      proxy_http_version 1.1;
      proxy_ignore_client_abort on;
   }
}
```

keepalive/httpversion/ignoreclient: are values a backend might need or not.

Nb. proxy_pass is not working, parts of the website is not showing and I see 404, 500 error entries in the log ?

Change:
```
      proxy_pass http://192.168.1.222:80/path/;
```
In to:
```
      proxy_pass http://192.168.1.222:80/path;
```
(note the trailing **slash** has been **removed**)

Other resources on migrating from Apache to NGINX:
https://www.digitalocean.com/community/tutorials/how-to-migrate-from-an-apache-web-server-to-nginx-on-an-ubuntu-vps
http://blog.donnywals.com/how-i-migrated-from-apache-to-nginx/

5. Speed up image access with a Vary header

```
server {
    listen       80;
    server_name  www.mydomain.eu;
    root   '/webroot/www.mydomain.eu';
    # Caching Static Files
    location ~* \.(jpg|jpeg|png|gif|ico|css|js)$ {
      expires 14d;
      add_header Vary Accept-Encoding;
    }
    location / {
        try_files $uri $uri/ =404;
        index  index.html index.htm;
    }
}
```

Very simple but very effective.

When images are missing make sure '**root'** is set to where your images are, pay attention especially with fast_cgi and proxy_pass **root** locations.

Check you logfiles for 404 entries to see where images supposed to be.

Remember root=local file access, it is not uncommon to use;
root //192.168.3.4/path/to/files
What is in root+location should be the UNC link to a resource.

A direct location for this is also possible like:

```
    location /applicationname/\.(jpg|jpeg|png|gif|ico|css|js)$ {
        proxy_pass http://192.168.140.30:8080;
        expires 14d;
        add_header Vary Accept-Encoding;
        ……..
```

The resource (your images) should then live in http://192.168.140.30:8080/applicationname

See also item 24 about the side effects of using Vary and caching.

6. Deny access to folder(s) (.htaccess conversion)

```
server {
    listen       80;
    server_name  www.mydomain.eu;
    root   '/webroot/www.mydomain.eu';
    ……
    location /cache/ { deny all; }
    location /files/ { deny all; }
    location /store/ { deny all; }
    location /uploads/ { deny all; }
    location /sessions/ { deny all; }
    ……
```

nginx does not support the .htaccess file method, to deny access you have to use above examples.

7. SSL best practices (dd. 7-3-2015 / 2023)

2015:
ssl_prefer_server_ciphers On;
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers
ECDH+AESGCM:ECDH+AES256:ECDH+AES128:ECDH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!eNULL:!MD5:!DSS:!EXP:!ADH:!LOW:!MEDIUM;

2023:
ssl_prefer_server_ciphers on;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers EECDH+AESGCM:EDH+AESGCM:!aNULL:!eNULL:!MD5:!DSS:!EXP:!ADH:!LOW:!MEDIUM;
ssl_ecdh_curve secp384r1;

8. Apache style logging

```
log_format  main  '[$time_local] $remote_addr $remote_port - $remote_user $scheme "$request" '
                  '$status $body_bytes_sent "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for" $upstream_cache_status';
```

9. Redirect all except root (for http to https redirection)

```
http {
   map $request_uri $requri {
      default         1;
      /               0;
   }
……….
   server {
      listen       80;
      server_name  www.mydomain.eu;
      root   '/webroot/www.mydomain.eu';
      if ($requri) { return 301 https://www.mydomain.eu$request_uri; }
      location / {
         try_files $uri $uri/ =404;
         index  index.html index.htm;
      }
   }
```

This will allow you to keep root access via plain HTTP but redirect everything else to HTTPS.

10. Authentication via other methods

```
location /authentication {
   auth_request /check/auth.php;
   proxy_pass http://127.0.0.1:8080;
}
```

Your **auth.php** could then check **Active Directory,** or whatever system, to check if the user is allowed anything.

See also http://nginx.org/en/docs/http/ngx_http_auth_request_module.html

"The ngx_http_auth_request_module module (1.5.4+) implements client authorization based on the result of a subrequest. If the subrequest returns a 2xx response code, the access is allowed. If it returns 401 or 403, the access is denied with the corresponding error code. Any other response code returned by the subrequest is considered an error.

For the 401 error, the client also receives the "WWW-Authenticate" header from the subrequest response."

NB.:
Everything seems to work fine for GET requests but for POST requests I get 499 and 500 errors

proxy_set_header Content-Length "";

It should be possible to do this in Lua and co-sockets which allows you to authenticate thousands of users simultaneously and cache their sessions and rights.

11. Elastic Backend Load Balancer & IWCP

>>> **E**lastic **B**ackend **L**oad **B**alancer **&** **I**nter **W**orker **C**ommunication **P**rotocol <<<

A new feature enabling you to manage your upstreams, using **IWCP** to propagate upstream changes to **all** workers.

Added in 17-1-2015 nginx 1.7.10.1 Gryphon

A simple configuration like:

```
upstream myLoadBalancer {
    server 192.168.169.22:80 weight=1 fail_timeout=5;
    server 192.168.169.17:80 weight=1 fail_timeout=5;
    server 192.168.169.26:80 weight=1 fail_timeout=5;
    server 192.168.169.23:80 weight=1 fail_timeout=5;
    server 192.168.169.27:80 weight=1 fail_timeout=5 down;
    server 192.168.169.28:80 weight=1 fail_timeout=5 down;
    least_conn;
}
upstream myLoadBalancerDDOS {
    server 127.0.0.1:8081 weight=1 fail_timeout=5;
    server 127.0.0.1:8082 weight=1 fail_timeout=5;
    server 127.0.0.1:8083 weight=1 fail_timeout=5 down;
    server 127.0.0.1:8084 weight=1 fail_timeout=5 down;
    server 127.0.0.1:8085 weight=1 fail_timeout=5 down;
    server 192.168.169.254:80 weight=1 fail_timeout=5 down;
    least_conn;
}
```

In **myLoadBalancer** you have set 2 extra backends ready for expanding capacity.

In **myLoadBalancerDDOS** you have set 2 backends (or internal redirects to 503 location blocks) to deal with attacks, a backend (…169.254:80) to serve as a blackhole and 3 more for expanding capacity.

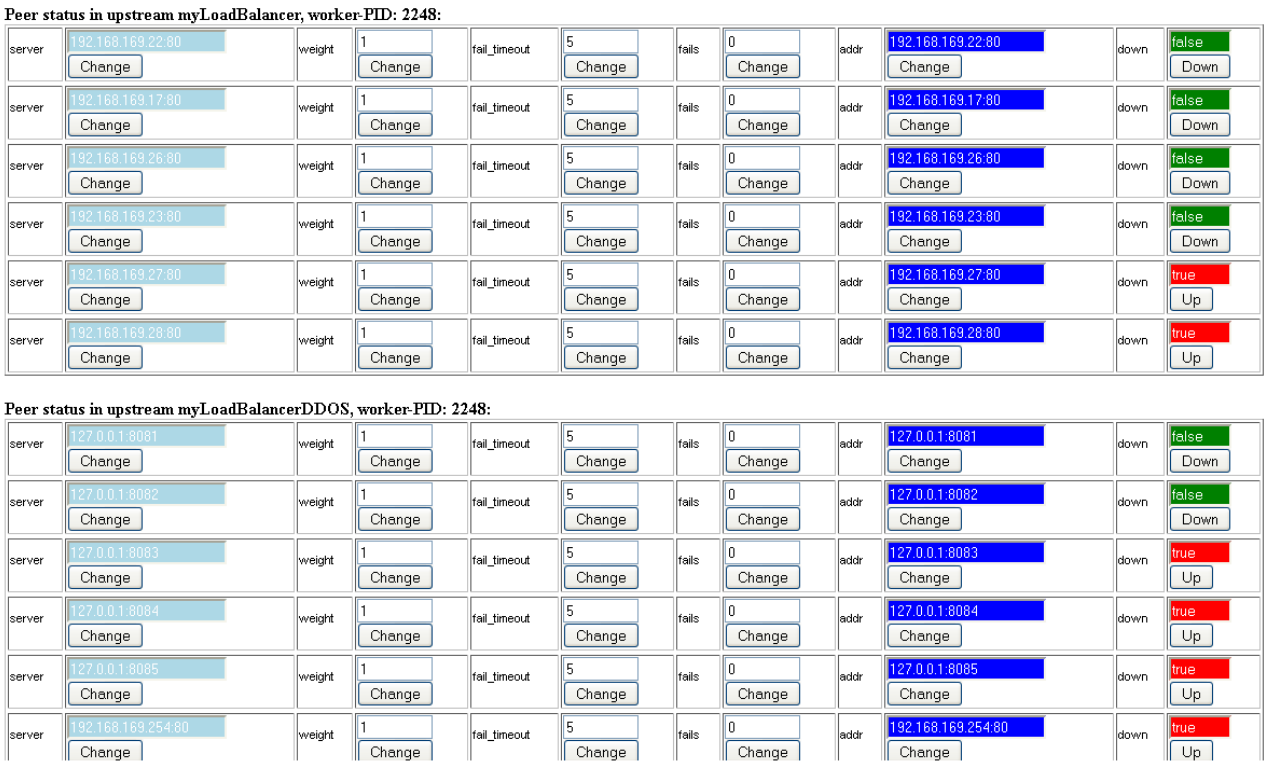
Of course you can set as many backends and their destinations as you like, other webservers, faster blackholes, offloading addresses, swap between cloud providers, etc…

The whole point of **EBLB** is to allow you to **add/remove** and **change resources** with the click of a mouse or fully automated with a tool like curl or even wget. You can't expand an existing pool yet so configure as much as you can and use the GUI / curl to manage them.

Use curl from monitoring devices or other trigger generating tools.
See conf/EBLB/upstream_EBLB_with_IWCP.txt for curl scripting examples.

nginx has a few internal monitoring values you could use in order to trigger a change in resources.

Through the **GUI** of **EBLB** (http://127.0.0.1/upstreamstatus) this example will look like:



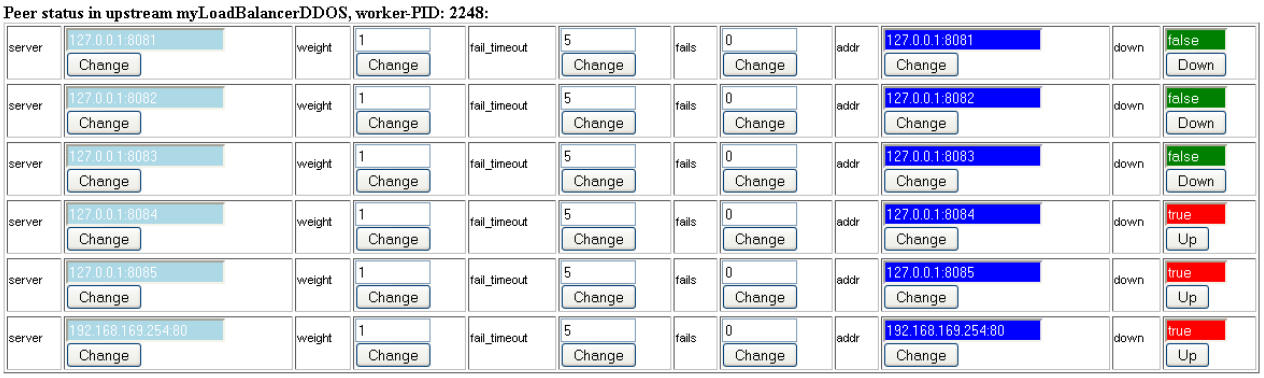
When you change a backend with the mouse or script via curl you're going to see **IWCP in action** in your logfiles:

2015/03/06 16:55:50 [error] 3268#4084: [lua] iworkcomproto.lua:48: message for worker: 3268,
key: IWCP_MSG_PU_14256573465869, msg: #2248#!myLoadBalancerDDOS,2,1,
context: ngx.timer
2015/03/06 16:55:50 [error] 3268#4084: [lua] iworkcomproto.lua:66: **result: true**, for worker: 3268,
err: nil, key: IWCP_MSG_PU_14256573465869, context: ngx.timer
2015/03/06 16:55:51 [error] 2940#2116: [lua] iworkcomproto.lua:48: message for worker: 2940,
key: IWCP_MSG_PU_14256573465869, msg: #3268#2248#!myLoadBalancerDDOS,2,1,
context: ngx.timer
2015/03/06 16:55:51 [error] 2940#2116: [lua] iworkcomproto.lua:66: **result: true**, for worker: 2940,
err: nil, key: IWCP_MSG_PU_14256573465869, context: ngx.timer
2015/03/06 16:55:51 [error] 2632#2552: [lua] iworkcomproto.lua:48: message for worker: 2632,
key: IWCP_MSG_PU_14256573465869, msg: #2940#3268#2248#!myLoadBalancerDDOS,2,1,
context: ngx.timer
2015/03/06 16:55:51 [error] 2632#2552: [lua] iworkcomproto.lua:66: **result: true**, for worker: 2632,
err: nil, key: IWCP_MSG_PU_14256573465869, context: ngx.timer

We have 4 workers so you see 3 messages initiated by one worker, all processed in less than 1 second.

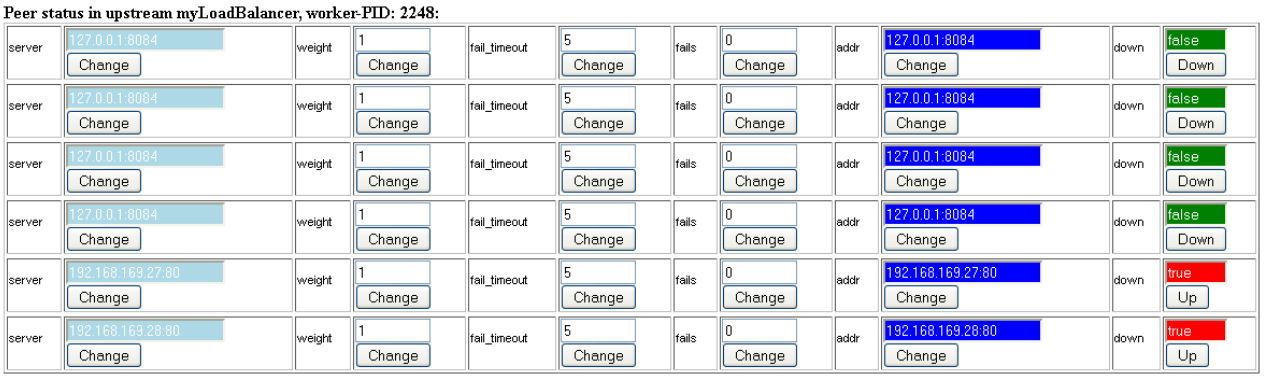
Nb. messages are written to the error log because there is no 'normal' way to write to the access log.

The **GUI** returns with:



And we have added a backend to the **myLoadBalancerDDOS** stream in **real-time** immediately ready for use for **all workers** via **IWCP**.

Let's do something dramatic with **myLoadBalancer** …



Within seconds we have redirected the main stream to 127.0.0.1:8084 which is a local location block **returning a 503**.

```
server {
    listen       8084;
    server_name  localhost;
    location / {  return 503;  }
}
```

No configuration reload, no downtime or the possibility of a mistake in the configuration file when editing under stress, just change and click or fire a curl script !

In order to use the **EBLB**-GUI and **IWCP** you need to copy conf/EBLB/*.lua to your /conf folder and 4 items in your nginx.conf file.

```
http {
   .....
[1]   lua_shared_dict iworkcomproto 1m;
   .....
[2]   init_worker_by_lua_file conf/iworkcomproto.lua;
   .....
   server {
      .....
      location = /upstreamcontrol {
        default_type text/html;
[3]           content_by_lua_file conf/upstreamcontrol.lua;
      }
      location = /upstreamstatus {
         default_type text/html;
[4]           content_by_lua_file conf/upstreamstatus.lua;
      }
      .....
```

1) A shared memory area where **IWCP** messages are stored
2) This loads **IWCP** inside every worker
3) This location block pre-processes **IWCP** commands and **EBLB**
4) This location block is the GUI of **EBLB**

See also: conf/EBLB/upstream_candc.conf
Items 1 and 2 are not required when using only 1 worker.

NB. either use allow/deny for management IP addresses or auth_basic in the location blocks to prevent someone else from changing your upstreams.

Ea.:

```
http {
   .....
   map $remote_addr $lcladdr {
      default                1;
      ~^(10.10.10.*)$        0;
      ~^(192.168.*.*)$       0;
   }
   .....
   server {
      .....
      location / {
         if ($lcladdr) { return 403; }
         .....
```

Or:

```
      location / {
         auth_basic           "Upstreams";
         auth_basic_user_file  /nginx/access.txt;   # see htpasswd.exe in the download section
         .....
```

12. PHP Caching

There are 2 main caching systems we like and are going to mention here, **Xcache** and **Opcache**.

Xcache: see conf/php-xcache-example.ini how to download, add and configure this in your PHP.INI file.

Xcache is very fast, a fraction faster than Opcache and deals better with frequent changing files, but it lacks the ability to use shared memory between instances, each PHP-CGI process will have its own cache which will take a while to be fully filled for optimum performance.

Xcache is supported for PHP 5.1 and higher.

Opcache: see conf/php-opcache-example.ini how to add and configure this in your PHP.INI file.

Opcache is part of PHP as of 5.5 and has support for shared memory which greatly improves all PHP-CGI instances by all using the same cache.

(nb. if xcache would use the same shared memory model we'd be back to xcache in a flash)

13. Error logging

Digitalocean has a nice article how error logging works, which we have taken over here.

Error_log Syntax

The "error_log" directive is used to handle logging general error messages. If you are coming from Apache, this is very similar to Apache's "ErrorLog" directive.

The error_log directive takes the following syntax:

error_log log_file [log_level]

The "log_file" in the example specifies the file where the logs will be written. The "log_level" specifies the lowest level of logging that you would like to record.

Logging Levels.

The error_log directive can be configured to log more or less information as required. The level of logging can be any one of the following:

- **emerg**: Emergency situations where the system is in an unusable state.
- **alert**: Severe situation where action is needed promptly.
- **crit**: Important problems that need to be addressed.
- **error**: An Error has occurred. Something was unsuccessful.
- **warn**: Something out of the ordinary happened, but not a cause for concern.
- **notice**: Something normal, but worth noting has happened.
- **info**: An informational message that might be nice to know.
- **debug**: Debugging information that can be useful to pinpoint where a problem is occurring.

The levels higher on the list are considered a higher priority. If you specify a level, the log will capture that level, **and any level higher** than the specified level.

For example, if you specify "error", the log will capture messages labeled "error", "crit", "alert", and "emerg".

*** This is a corrected summery from dialogs of the internet such as blogs/forums/wiki ***

14. Virtual Host Traffic Status, Monitoring for your NOC

A new feature adding monitoring for your NOC (Network Operations Center) enabling you to monitor your server(s), virtual hosts and upstreams.

Added in 14-3-2015 nginx 1.7.11.2 Gryphon

See **/conf/vhts** (/VHTS.txt) for instructions.
See also the *.js files for customizing the language, layout and color triggers.

Server main

Version	Uptime	Connections				Requests			
		active	reading	writing	waiting	accepted	handled	Total	Req/s
1.7.10	4m 9s	1	0	1	0	45920	45920	46169	0

Server zones

Zone	Requests		Responses						Traffic			
	Total	Req/s	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s
server2	8940	0	0	8940	0	0	0	8940	2.8 MiB	742.1 KiB	0 B	0 B
server3	7983	0	0	7983	0	0	0	7983	2.5 MiB	662.7 KiB	0 B	0 B
server0	19626	0	0	19626	0	0	0	19626	979.6 MiB	1.6 MiB	835 B	333 B
server1	9619	0	0	9619	0	0	0	9619	3.0 MiB	798.5 KiB	0 B	0 B
*	46168	0	0	46168	0	0	0	46168	987.9 MiB	3.7 MiB	835 B	333 B

Upstreams

group0

Server	State	Response Time	Weight	MaxFails	FailTimeout	Requests		Responses						Traffic			
						Total	Req/s	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s
10.10.10.21:80	up	2ms	1	1	10	9619	0	0	9619	0	0	0	9619	3.0 MiB	798.5 KiB	0 B	0 B
10.10.10.21:80	up	2ms	1	1	10	9619	0	0	9619	0	0	0	9619	3.0 MiB	798.5 KiB	0 B	0 B

group1

Server	State	Response Time	Weight	MaxFails	FailTimeout	Requests		Responses						Traffic			
						Total	Req/s	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s
10.10.10.31:80	up	2ms	10	1	10	8462	0	0	8462	0	0	0	8462	2.6 MiB	702.4 KiB	0 B	0 B
10.10.10.32:80	up	2ms	10	1	10	8461	0	0	8461	0	0	0	8461	2.6 MiB	702.3 KiB	0 B	0 B
10.10.10.33:80	down	0ms	10	1	10	0	0	0	0	0	0	0	0	0 B	0 B	0 B	0 B
10.10.10.34:80	backup	0ms	10	1	10	0	0	0	0	0	0	0	0	0 B	0 B	0 B	0 B

::nogroups

Server	State	Response Time	Weight	MaxFails	FailTimeout	Requests		Responses						Traffic			
						Total	Req/s	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s
127.0.0.1:9000	up	4ms	0	0	0	12711	0	0	12711	0	0	0	12711	963.7 MiB	1.0 MiB	0 B	0 B
127.0.0.1:8080	up	1ms	0	0	0	6666	0	0	6666	0	0	0	6666	15.7 MiB	488.2 KiB	0 B	0 B

Server zones: this area will fill up when each zone (server) is used (requested). When nginx has just started this list might be close to empty.

VHTS can also display different languages for an outsourced NOC.
see /conf/vhts/vtsvalues-**xy**.js (default is English)

This feature will fill up your access.log file quickly, you can filter them with the following (tested) example;

```
map $request_uri $loggablevhts {
    default                     1;
    /ngxvtstatus                0;  # zero=do not log
    /vtsvalues.js               0;  # zero=do not log
    /vtsvalues-eop.js           0;  # zero=do not log
    /vtsvalues-xy.js            0;  # zero=do not log (see language js files)
    /ngxvtstatus/format/json    0;  # zero=do not log
}
map $remote_addr $lcladdrvhts {
    default                     1;
    ~^(127.0.0.*)$              0;  # zero=do not log
}
# don't log vhts entries when request is local or from management interface
map $loggablevhts$lcladdrvhts $loggable {
    default                     0;
    ~1                          1;
}

access_log   /path/to/access.log  combined  if=$loggable;
```

“A request will not be logged if the (IF) condition evaluates to "0" or an empty string”

Two simple ‘maps’ which are then combined tested in the third ‘map’ which is used in the IF evaluation of the log directive.

Nb. *“[emerg]: could not build the map_hash, you should increase map_hash_bucket_size: 32”*
You might need to increase the value of map_hash_bucket_size to;
map_hash_bucket_size 64;

15. Upstream timed out (10060.....)

The configuration is like this:

```
<snip>
location /api {

    proxy_pass http://localhost:3000;
    proxy_set_header X-Real-IP $remote_addr;

}
</snip>
```

First request goes fine, second request is being loaded after 60 seconds. An error appears on the error log of upstream error:

*2014/01/30 10:32:32 [error] 6760#3604: *1 upstream timed out (10060: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond) while connecting to upstream, client: 127.0.0.1, server: localhost, request: "GET /api/xxxxx HTTP/1.1", upstream: "http://[::1]:3000/api/xxxxx", host: "localhost"*

Solution is to change proxy_pass to;

proxy_pass http://**127.0.0.1**:3000;

There might also be an issue when using domain names **which are external** while you are internal to nginx, ea. All traffic going **out through** a router and **back in again**, use a local DNS (or your 'host' file to keep such traffic local)

Also make sure your timeout values are within range,
Ea. client_body_timeout, client_header_timeout, keepalive_timeout, send_timeout, keepalive_requests

And the values in conf/proxy.conf

Nb. In some cases we have seen that the local dns service on a workstation is not updating properly, if you see such things happening then disable the DNS Client service.

"Every other 60 second timeouts"
Have a good look at the settings of the network cards, energy saving mode, tcp offloading, QOS, top/dis-responders, dns settings (like auto register this machine at....), unused netbios, calls to active directory which may interrupt, bugs with jumboframes, MTU, etc. Multiple LAN cards and using more than one default gateway (ARP/MAC 'blocking').

The only way to really dig into this specific issue is wireshark/tcpdump on both ends to determine if calls arrive, what happens with them and what happens when they return to nginx (port closures, handshake error, timeout, protocol errors, etc.)

16. Accessing remote resources

Consider the following situation,

Windows: Net use p: //192.168.123.55/data password /user:username

nginx conf: root p:/files

Error Log: *10 CreateFile() "p:/files/index.html" failed (5: Access is denied)

Can be solved with the following scenario;

1. create an user on your remote resource with an username/password that **is the same** as nginx is using.
2. then use the resource directly like: **root //192.168.123.55/data**
3. Done !

A network drive can and does work with nginx but when using a service to run nginx this network drive (mapped drive) is not visible or accessible inside the service profile.

There are solutions mentioned when you search for them but none really keep on working or don't work at all.

Matching username and password between the profile used for nginx and your remote resource always works and will keep on working.

- *What about a shared folder in VirtualBox?*
Don't use this feature for nginx inside a VM, create a real shared resource on the host instead.

17. PCRE is now JIT enabled

PCRE: Perl Compatible Regular Expressions (As a build in component with nginx).

Added in 3-12-2015 nginx 1.9.8.2 Kitty

Enables the use of "just-in-time compilation" (PCRE JIT) for the regular expressions known by the time of configuration parsing.

PCRE JIT can speed up processing of regular expressions significantly.

Large performance benefits are possible when (for example) the calling program utilizes the feature with compatible patterns that are executed repeatedly.

*** This is a corrected summery from dialogs of the internet such as blogs/forums/wiki ***

18. Sticky: Enables session affinity (module)

Enables session affinity, which causes requests from the same client to be passed to the same server in a group of servers. See also **item 25** (restful API).

Added in 22-12-2015 nginx 1.9.10.1 Kitty

For examples see **nginx-sticky-module-x.y.pdf** in our download area.

Session example: **Get a cookie**

C:\nginx>curl -i http://127.0.0.1/backend
HTTP/1.1 200 OK
Server: nginx/1.9.8.3 Kitty
Date: Wed, 09 Dec 2015 14:00:41 GMT
Content-Type: text/html
Content-Length: 1347 *(content from backend 1, cookie set to hash from this backend)*
Connection: keep-alive
Set-Cookie: route=6fd05ef29ac471a01914964d79ae23fa55980dc4; Expires=Wed, 09-Dec-2015 15:00:41 GMT; Path=/
Vary: Accept-Encoding
Last-Modified: Tue, 05 Jun 2012 20:36:30 GMT
ETag: "0-543-4fce6dce"
Accept-Ranges: bytes

Session example: **Use cookie**

C:\nginx>curl -v --cookie "route=6fd05ef29ac471a01914964d79ae23fa55980dc4" http://127.0.0.1/backend
* About to connect() to 127.0.0.1 port 80 (#0)
* Trying 127.0.0.1...
* Connected to 127.0.0.1 (127.0.0.1) port 80 (#0)
> GET /backend HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 127.0.0.1
> Accept: */*
> Cookie: route=6fd05ef29ac471a01914964d79ae23fa55980dc4
>
< HTTP/1.1 200 OK
< Server: nginx/1.9.8.3 Kitty
< Date: Wed, 09 Dec 2015 14:01:00 GMT
< Content-Type: text/html
< Content-Length: 1347 *(content from backend 1, cookie set from hash to this backend)*
< Connection: keep-alive
< Vary: Accept-Encoding
< Last-Modified: Tue, 05 Jun 2012 20:36:30 GMT
< ETag: "0-543-4fce6dce"
< Accept-Ranges: bytes

Example nginx configuration:

```
upstream backendus {
   server 192.168.2.2:80 weight=1 fail_timeout=5;
   server 192.168.2.6:80 weight=1 fail_timeout=5;
   sticky name=route hash=sha1 expires=1h;
}

server {
   listen 80;
   server_name localhost;

   location /backend {
      proxy_ignore_client_abort on;
      proxy_set_header Host $host;
      proxy_pass http://backendus;
   }
}
```

Example method of combining Sticky and Least_conn loadbalancing:

```
   # Our sticky pool, once stuck to one member, clients will stay stuck
   upstream backendus {
      server 127.0.0.1:81 weight=1 fail_timeout=5;
      server 127.0.0.1:82 weight=1 fail_timeout=5;
      sticky name=route hash=sha1 expires=1h;
      # pass a stuck session to an internal pool who can deal with sticky sessions
   }
   upstream stickybackendsPA {
      server 192.168.2.10:80 weight=1 fail_timeout=5;
      server 192.168.2.11:80 weight=1 fail_timeout=5;
      least_conn;
      # pool A loadbalanced sticky servers (who can deal with sticky session data)
   }
   upstream stickybackendsPB {
      server 192.168.2.20:80 weight=1 fail_timeout=5;
      server 192.168.2.21:80 weight=1 fail_timeout=5;
      least_conn;
      # pool B loadbalanced sticky servers (who can deal with sticky session data)
   }

   server {
      listen       80;
      server_name  localhost;
      location /backend {
         keepalive_requests 500;
         proxy_http_version 1.1;
         proxy_ignore_client_abort on;
         proxy_set_header Host $host;
         proxy_pass http://backendus/; # go to our sticky pool
      }
   }
```

```
server {
    listen       81;
    location / {
        proxy_pass http://stickybackendsPA;
        # from our sticky pool to loadbalanced sticky servers
    }
}
server {
    listen       82;
    location / {
        proxy_pass http://stickybackendsPB;
        # from our sticky pool to loadbalanced sticky servers
    }
}
```

Which is basically a loop on the nginx machine and yes it doubles the connections needed, but for nginx the load of 100k users is just as easy as 200k.

Possible solution for gradually server switching:

1. user session normally tracked by cookie, so check the cookie to identify old/new session
2. route new session to specific server, route old session to its sticked server

Configuration:

```
upstream app_pool {
    sticky;
    server a;
    server b;
}
upstream upgrade_pool {
    sticky;
    server a;
    server b down;
}

server {
    location xxxx {
        set $poolname "app_pool";
        if ($cookie_XXXSESSIONID = "") { set $poolname "upgrade_pool"; }
        proxy_pass http://$poolname;
    }
}
```

This module is available as part of our custom commercial subscription

19. $realip_remote_addr $realip_remote_port

For example:

```
log_format  main   '[$time_local] $remote_addr $remote_port - $remote_user $scheme "$request" '
                   '$status $body_bytes_sent "$http_referer" '
                   '"$http_user_agent" "$http_x_forwarded_for" $upstream_cache_status '
                   '$realip_remote_addr $realip_remote_port';
```

$realip_remote_addr was recently added to the main core, we've added $realip_remote_**port**

dd. 12-6-2016 this is now part of the nginx core.

20. Rewrite based on user language setting

You may have noticed our website now supports multiple languages, which is also set automatically when a user has set their preferred language. Here's how we do that.

```
http {
……….
# prepare a variable to use in location {}
   map $http_accept_language $sublang {
      default        '';
      ~*nl           '-nl';
      ~*de           '-de';
   }
# do it again but create a logical variable for testing
   map $http_accept_language $sublang1 {
      default        0;
      ~*nl           1;
      ~*de           1;
   }
# is user coming in at root and have they set a preferred language ?
   map $request_uri $sublang2 {
      default        0;
      /              $sublang1;
   }
……….
     location / {
       if ($sublang2) { rewrite ^(.*)$ /index$sublang.html break; }
# finally rewrite to a specific language template html file
```

21. Video streaming with rtmp and vod

Streaming modules **rtmp** and **vod** (MP4 Re-packager) have been ported for non-blocking use in Windows, both modules have a proven track record, documentation can be found in our download area.

- *These modules are available as part of our basic commercial subscription*

22. Closed source versus Open source

From our (old) FAQ:

> Q: Are the sources available?
> A: Short answer: No. Long answer: we have contracted two external auditing companies for validation of processes and coding to ensure quality. Due to the merge/build processes it's no longer possible to use a single repository. It would be impossible to maintain our environment and a public one, neither do we have time for lengthy coding discussions, after more than two years of development we're fairly sure we know what we're doing
>
> A note about subscriptions: There are **NO limits imposed** on any version, subscription rates vary because the support requirements for 10k users per day and 100k are very different, large busy sites may require more than one engineer to address problems or to handle support requests

So there it is, we've said it now which most likely will upset some people but we firmly believe that time, money and quality for such a complex yet extremely powerful system will suffer badly if left as open source.
And to be honest we don't think Nginx Inc. thinks differently with their nginx+ product line.

Of course you can always get the original open source nginx code and **compile your own build**, there is no one stopping you from doing this (but also accept the fact that such a build will have no proper ASLR/DEP support, no multiple worker support, no high performance support through select-boost, etc.), **but if you want more**, hassle free high performance compiled builds, specific (subscription) modules or functionality (ported for Windows) with professional 24/7 support then you need **professional paid support**.

In our download area you will **always** find our latest high performance unlimited completely **free** builds !

A note about the relation between modules and subscription versions: a module is part of the basic, custom or enterprise subscription depends on its complexity and interaction relations (Linux <-> Windows API) with other components.

23. Stream {} server_name ?

TCP has no concept of server names, so this is not possible. It only works in HTTP because the client sends the hostname it is trying to access as part of the request, allowing nginx to match it to a specific server block.

To put in to better wording:

The 'hostname' (server_name) technique has 2 parts,

- **part 1** is a receiver (nginx) receiving a request containing a hostname which it can match (or not) to an item in its configuration, and
- **part 2** the DNS where this name is recorded against its IP address.

With stream {} you can only rely on **part 2**.

*** This is a corrected summery from dialogs of the internet such as blogs/forums/wiki ***

24. Multiple cache files for the same key

Multiple cache files for the same key can be created if a backend response uses the **Vary** mechanism to allow multiple resource variants. It is supported by nginx and taken into account when caching.

If responses are really the same, consider removing Vary from backend responses.
If this is not possible for some reason, you can use proxy_ignore_headers to stop nginx from handling Vary in responses, e.g.:

proxy_ignore_headers Vary;

Some additional details can be found in the original nginx documentation here:
http://nginx.org/r/proxy_ignore_headers

*** This is a corrected summery from dialogs of the internet such as blogs/forums/wiki ***

25. Restful interfaces and Headers

This may be obvious to some but not so obvious to others, in a restful API you need to send back any value the receiver expects or needs to perform functionality based on such values.

For example with cookies: nginx only sends it once, it is the browser (or Curl or restapi) responsibility to always send the cookie back (if it's not expired, if it is expired the receiver will send a new one back). This is what identifies the 'user' being tied (sticky) to a specific backend. See also **item 18**.

26. CIS, DHS, OWASP

In our download area you will find procedures (documents) which describe how to enforce and perform hardening, change and patch management, guidelines to be and stay attack resilient on Windows.

Having a NGINX secure environment on Windows on its own is not going to do much good if the rest does not follow the same security principles and guidelines.

27. Sendfile on or off ?

22 March 2016:

Despite the recent changes dealing with *sendfile* issues we still recommend ***sendfile*** to be **off**.

28. Dynamic TLS (Optimizing TLS over TCP)

Original blog post https://blog.cloudflare.com/optimizing-tls-over-tcp-to-reduce-latency/
10 Jun 2016 by John Graham-Cumming.

Dynamic TLS has been added in nginx 1.11.2.1 WhiteKnight

ssl_dyn_rec_size_lo: the TLS record size to start with.
Defaults to 1369 bytes (designed to fit the entire record in a single TCP segment: 1369 = 1500 - 40 (IPv6) - 20 (TCP) - 10 (Time) - 61 (Max TLS overhead))

ssl_dyn_rec_size_hi: the TLS record size to grow to.
Defaults to 4229 bytes (designed to fit the entire record in 3 TCP segments)

ssl_dyn_rec_threshold: the number of records to send before changing the record size.

Each connection starts with records of the size **ssl_dyn_rec_size_lo**. After sending **ssl_dyn_rec_threshold** records the record size is increased to **ssl_dyn_rec_size_hi**. After sending an additional **ssl_dyn_rec_threshold** records with size **ssl_dyn_rec_size_hi** the record size is increased to **ssl_buffer_size**.

ssl_dyn_rec_timeout: if the connection idles for longer than this time (in seconds) that the TLS record size is reduced to **ssl_dyn_rec_size_lo** and the logic above is repeated. If this value is set to 0 then dynamic TLS record sizes are disabled and the fixed **ssl_buffer_size** will be used instead.

New nginx.conf options (the default values are good enough):
ssl_dyn_rec_enable
ssl_dyn_rec_timeout
ssl_dyn_rec_size_lo
ssl_dyn_rec_size_hi
ssl_dyn_rec_threshold

*** This is a corrected summery from dialogs of the internet such as blogs/forums/wiki ***

29. Combining SSL and SSH (SSLH)

SSLH: A SSL/SSH Multiplexer.

sslh accepts connections in HTTPS, SSH, or any other protocol that can be tested using a regular expression, on the same port. This makes it possible to connect to any of these servers on port 443 while still serving HTTPS on that port.

This has been possible for awhile now (since 19-10-2016 nginx 1.11.6.1 Lion), see for example:
https://stackoverflow.com/questions/69245993/nginx-ssl-preload-ssh-http-https-setup
And
https://ostechnix.com/sslh-share-port-https-ssh/

Let's make this work for Windows:

```
stream {

  log_format strmain '[$time_local] $remote_addr '
                '$protocol $ssl_preread_server_name $status $bytes_sent $bytes_received '
                '$session_time "$upstream_addr" '
                '"$upstream_bytes_sent" "$upstream_bytes_received" "$upstream_connect_time"';

  map $ssl_preread_server_name $sslhservice {
      ~*\.              webserverfarm;
      default           sshserverfarm;
  }

 tcp_nodelay     on;

  upstream webserverfarm {
      server 10.10.10.102:443; # Web services
      server 10.10.10.103:443; # Web services
      least_conn;
  }
  upstream sshserverfarm {
      server 10.10.20.102:22; # SSH services
      server 10.10.20.103:22; # SSH services
      least_conn;
  }

  server {
      listen  443;
      proxy_protocol on;
      error_log   logs/stream_sslh.error.log;
      access_log  logs/stream_sslh.access.log strmain;
      proxy_connect_timeout 10s;
      proxy_pass  $sslhservice;
      ssl_preread on;
  }

}
```

- Appendixes -

A. F.A.Q.

1. How do the version number and name come together?
   ⇨ The version number follows nginx release version numbers + a subversion number specific to our Windows releases + a random theme name based on Alice in Wonderland.

2. Why did you not compile in all c++ dependencies? (vcredist)?
   ⇨ These dependencies are not always the same between Windows versions, with the current native build it will run on any *supported* Windows OS.

3. Isn't using select slow?
   ⇨ No, select is not slow, it's only seems cpu hungry but not in way that hinders high performance usage, Linux really suffers with select but not Windows despite the Linux slander about this, our select-boost api has solved this issue.

4. Is this a long term project?
   ⇨ We have 36 months LTS support, how long do you want long term to be ? :)

5. Do you port every change?
   ⇨ No is the short answer, there is a complex evaluation process which determines if a change is valid or not, it's not just nginx code but add-on's code interactions which needs to be evaluated.

6. Why can't I change drives?
   ⇨ You can but you need to write a path like Linux, D:\Path becomes d:/Path

7. Where is the documentation?
   ⇨ See our website FrontPage and the /download/documentation-pdf/ folder where we maintain all documentation files. Take care using any examples which has references to Linux paths.

8. What determines changes or addons to be added?
   ⇨ It must be useful, it must be an add-on unless the change is minimal and low impact on the core, it must compile cleanly under Windows, it can't have external dependencies unless it can be linked to a single dll module, it must not use OS exclusive functionality. Add-ons are not removed once they are in unless they are bugged beyond repair or when their functionality has been merged.

9. nginx stop/reload on Windows fails with a 'Access is denied'
   ⇨ Run the nginx service as a user (and jail that user), then create a simple cmd file;
   - runas /savecred /env /user:nginxuser "nginx -s reload"
   - choice /ty,2 /C:ync (or "sleep 2")
   - runas /savecred /env /user:nginxuser "nginx -s reopen"

   At the first run you need to enter nginxuser's password, after that '/savecred' will take care of this.

10. XP is no longer supported after april 2014, are you going to stop support nginx/xp?
    ⇨ Are you a lemming? Neither are we, we will keep supporting XP until it's technically impossible. NGINX for Windows native build runs on Windows XP SP3 and higher, both 32 and 64 bit.

11. Sometimes we see a delay with LAN traffic, nginx is not doing anything but the request is stalling.
    ⇨ Make sure you have deactivated netbios (smb) from the lan interface(s).

12. Does naxsi work for phpbb?
    ⇨ Yes it does, but you need to add a few white-list lines:
    BasicRule wl:1000 "mz:$URL:/ucp.php|BODY|NAME";
    BasicRule wl:1310 "mz:$URL:/ucp.php|BODY|NAME";
    BasicRule wl:1311 "mz:$URL:/ucp.php|BODY|NAME";

13. What is nginx_basic?
    ⇨ Basically a one on one replacement for the windows nginx version which is made by nginx themselves but with all the benefits and work that has gone into its big brother, which you find here, without add-ons.

14. WSARecv() failed (10054: An existing connection was forcibly closed by the remote host) while reading upstream (backend = tomcat, java or similar applications)
    ⇨ apply these settings to the proxy:
    keepalive_requests 500;
    proxy_http_version 1.1;
    context: http, server, location
    Version 1.1 is recommended for use with keepalive connections

15. Windows Server 2012 message that msvcr100.dll is missing?
    ⇨ In some cases:
    manually remove "C:\Windows\System32\msvcr100.dll" and (Re)install the program vcredist_x64 from here http://nginx-win.ecsds.eu/
    In other cases:
    Install both the 32-bit C++ Runtime and the 64-bit version as well.

16. Is this version production ready? And who is using it in production?
    ⇨ Yes it is! And has been for a while, we are using it in a production environment and we are aware there are quite a few others running our builds.

17. Do I need lua51.dll?
    ⇨ With nginx.exe you do need it, **nginx_basic.exe does not** need this library.

18. Can I use other DLL functions with Lua and import them?
    ⇨ Yes but they need to be compiled against http://luajit.org/download.html (at the moment v2.0.4), use 'findstr "LuaJIT " lua51.dll' to see which version we have shipped, functions can be used for example: 'local functionname = package.loadlib("External.dll", "luaopen_Function");' See also these examples http://www.scilua.org/ljsqlite3.html
    Introducing ngxLuaDB (nginx Lua Database) powered by NGINX for Windows. See the download section.