Nginx notes

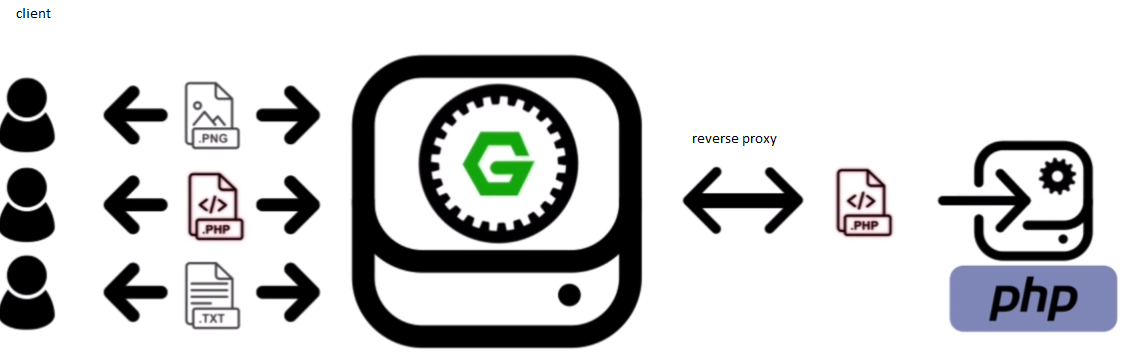
# Configuration

About Nginx(Igor Sysov 10k concurrent conections, high performance, high concurrency, low resource usage). It is web server but in its core it is a reverse proxy server.

Nginx vs apache:

Basic architecture of apache is prefork mode (had spawned a set number of processors, each of which can serve a single request at a time regardless wheathe that request is for php script of image)

Nginx is dealing with requests asynchronously (single process can serve multiple requests concurrently that number depends on system resource available to nginx process) –sidenote unlike apache can’t embed server side programming languages into its own processes meaning that all requests for Dynamic content has to be dealt with completely separate process like php fpm and then reverse proxy to the client via enginx (low resource usage because not hav9ing to deal directly with embedded programming languages). But unlike apache, server side language modules don’t need to be run for every single request that server receives, instead nginx will handle serving static resources without php ever knowing about it(apache will handle every request with that costly overhead)-that is saving on system resources



Nginx is faster than apache (internet collection will allow speed of getting data to the client, but can serve static resources much faster (no need for any server side languages), and handle a much larger number of concurrent requests (can receive 1k of requests on single processing thread. Apache will accept request up to preconfigured number and reject rest) ).

Nginx configuration (requests are interpreted as uri location \*location /images {}\* , this allows nginx to work not only as web server, but also as load balancer to mail server, whereas apache default to filesystem locations \*<Directory “/www/site/images”>\*). .htaccess file can override configuration, there is no such in nginx.

# Install Nginx

* Using package manager

Ssh [anel@163.172.171.154](mailto:anel@163.172.171.154) (to remove ip/remove it from .*ssh/known*\_hosts: *ssh-keygen –R 163.172.171.154* )

With package manager: quick and easy, limited install options, no support for additional modules. Suitable for most basic web servers or testing and development. Almost never best solution

**sudo apt-get install nginx**

With this particular package nginx is also running ( search for process command for all users listing detailed information and include boot processes*=> ps aux | grep nginx* => should have master and worker processes). Ifconfig = > ip in the browser to see does nginx works and listens oh http port 80.

Check for nginx configuration files : **ls –l /etc/nginx**

* Building from the source

<http://nginx.org/>

<https://www.nginx.com/> [nginx+ paid]

<https://www.nginx.com/resources/wiki/>

Main benefit why to build from the source is to have ability to add custom modules or to extend standard nginx functionality (which you cannot do with package manager). Nginx modules exists in 2 form: bundled modules(gzip,spdy,ssl,geoip – come with the source), third party modules.

Navigate to download link: <http://nginx.org/en/download.html>

We will use mainline version.

Copy link: => <http://nginx.org/download/nginx-1.15.5.tar.gz> and use wget. Navigate to workspace folder to get tarball format:

**$wget** [**http://nginx.org/download/nginx-1.15.5.tar.gz**](http://nginx.org/download/nginx-1.15.5.tar.gz)

**$ tar –xzvf nginx-1.15.5.tar.gz**

**$cd nginx-1.15.5**

Configure source code for the build.

To do this run the configure script in the source code directory. Before enable tool to compile nginx (gcc), and some dependencies : libpcre3, libpcre3-dev, zlib1g zlib1g-dev (for g zipping), libssl-dev (for ssl/https support)

**$ apt-get install libpcre3 libpcre3-dev zlib1g zlib1g-dev libssl-dev**

**$ apt-get install build-essential**

**$./configure**

We still haven’t added any custom configuration flags. To do so run **./configure –help** to see all of them. See documentation <http://nginx.org/en/docs/configure.html>

|  |  |
| --- | --- |
| **Flag** | **Description** |
| **--sbin-path** | = /usr/bin/nginx  location of nginx executable which will use to start and stop the nginx service (usr/bin common location for Ubuntu executables) |
| **--conf-path** | = /etc/nginx/nginx.conf  Path of nginx configuration files. |
| **--error-log-path** | = /var/log/nginx/error.log |
| **--http-log-path** | = /var/log/nginx/access.log |
| **--pid-path** | =/var/run/nginx.pid  We will need to know when configuring nginx |
| **--with-pcre** | Tell the nginx to use system specif library for regular expressions |
| **--with-http\_ssl\_module** | Bundled module |

**$ ./configure --sbin-path=/usr/bin/nginx --conf-path=/etc/nginx/nginx.conf --error-log-path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --pid-path=/var/run/nginx.pid --with-pcre --with-http\_ssl\_module**

We can now go ahead and compile the configuration source by running make:

**$ make**

After this is done install the compiled source with make install:

**$ sudo make install**

See the configuration files: $ ls –l /etc/nginx ; See the binary : **$ls –l /usr/bin |grep nginx**.

Test the version: **$ nginx –V** (see the arguments which should be same as configured). Test **$ps aux|grep nginx** and run from source folder nginx: **$ sudo nginx**

Navigate to the browser to check again.

|  |
| --- |
| *Configuration summary*  *+ using system PCRE library*  *+ OpenSSL library is not used*  *+ using system zlib library*  *nginx path prefix: "/usr/local/nginx"*  *nginx binary file: "/usr/local/nginx/sbin/nginx"*  *nginx modules path: "/usr/local/nginx/modules"*  *nginx configuration prefix: "/usr/local/nginx/conf"*  *nginx configuration file: "/usr/local/nginx/conf/nginx.conf"*  *nginx pid file: "/usr/local/nginx/logs/nginx.pid"*  *nginx error log file: "/usr/local/nginx/logs/error.log"*  *nginx http access log file: "/usr/local/nginx/logs/access.log"*  *nginx http client request body temporary files: "client\_body\_temp"*  *nginx http proxy temporary files: "proxy\_temp"*  *nginx http fastcgi temporary files: "fastcgi\_temp"*  *nginx http uwsgi temporary files: "uwsgi\_temp"*  *nginx http scgi temporary files: "scgi\_temp"* |

## Configuring the system service for nginx (as systemd):

Adding nginx as systemd (newer standard for services, >Ubuntu 15.0.4) service (will enable us to start/stop/restart, reload(configuration) and start on boot.

To see options for nginx navigate to source folder and run : **$nginx –h**

To stop the service use signal flag (-s) and stop (terminated) : **$nginx –s stop**

* **Let’s add systemd service:**

To enable the service we are going to have to add a small script which being the same across all operating systems (init.d are old)

Nginx initscripts : <https://www.nginx.com/resources/wiki/start/topics/examples/initscripts/>

$ touch /lib/systemd/system/nginx.service

This file already exist in /lib/system/system folder but it is different (from Feb 11, 2017)

|  |
| --- |
| **[Unit]**  Description=The NGINX HTTP and reverse proxy server  After=syslog.target network.target remote-fs.target nss-lookup.target  **[Service]**  Type=forking  PIDFile=/var/run/nginx.pid  ExecStartPre=/usr/bin/nginx -t  ExecStart=/usr/bin/nginx  ExecReload=/usr/bin/nginx -s reload  ExecStop=/bin/kill -s QUIT $MAINPID  PrivateTmp=true  **[Install]**  WantedBy=multi-user.target |

Start with ngingx (needed to do daemon reload) :

*Warning: nginx.service changed on disk. Run 'systemctl daemon-reload' to reload units.*

**$ sudo systemctl daemon-reload**

**$sudo systemctl start nginx**

With systemd service enabled, we can also check nginx status using systemd instead of process command. **$sudo systemctl status nginx**

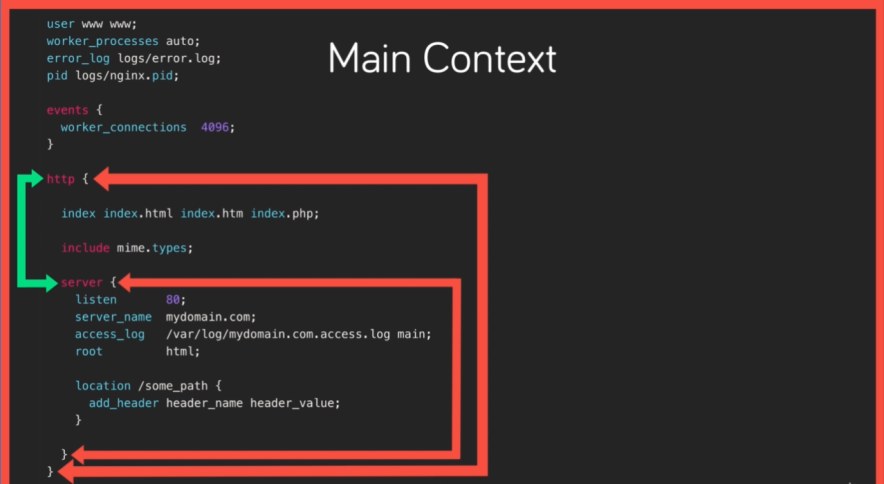
Enable startup on boot : **$sudo systemctl enable nginx**

|  |
| --- |
| *Synchronizing state of nginx.service with SysV init with /lib/systemd/systemd-sysv-install...*  *Executing /lib/systemd/systemd-sysv-install enable nginx* |

## Configuration terms:

*Context –* section within the configuration, where directives can be set for that given context. It is like scope (::) and like scope they can be nested and inherit from parents.Topmost context is configuration file itself - Main context (global directives that apply to master process). Http context , server context(apache V host), location context(matching uri locations on incoming requests to the parent server context)

*Directive* – specific configuration option that get set in configuration file (consists of name and value : *server\_name mydomain.com*)



## Creating a Virtual Host or server context:

To serve static files from directory on our server.

$ *mkdir my\_sites && cd my\_sites && mkdir demo && cd demo*

Server context is responsible for listening on some port (80 http, 443 https) for given ip address or domain.

Start with */etc/nginx/nginx.conf* file : add server context and *directive listen 80; server\_name IPadess; root /home/anel/..* and for variety of types in http context *include mimo.types*;

Using *nginx –t* you can verify does you configuration is successful.

|  |
| --- |
| *nginx: the configuration file /etc/nginx/nginx.conf syntax is ok*  *nginx: configuration file /etc/nginx/nginx.conf test is successful* |

Using *systemctl reload nginx* you can load a new configuration (with *systemctol restart nginx = >* it will stop and if it is not good old configuration will not be reloaded). Mime.types are used for providing the server content type for various of extensions.

To test the MIME type check the stylesheet header using curl (content-type: text/plain is not good) ; **curl –I http://163.172.171.154/style.css**

## Location context:

Added in server context.

According to the priority we have:

* Exact match :*location = /anel {return 200 “Status message!”;}* # when returning return status code and message
* Preferential prefix match : *location ^~ /Anel7 {return 200 “Status message!”;}*
* Regex match (case sensitive, for case insensitive put *~\**) : *location ~ /anel[0-9] {return 200 “Status message!”;}*
* Prefix match (can be treated as /anelh, /anelax/mariadb etc) *location /anel {return 200 “Status message!”;}*

## Variables:

Aside of concept of scope, includes there are variables and conditionals in nginx.

There are 2 types of variables:

* Configuration variables (user defined, we set ourselves used with **set** keyword (set $anel ‘Anel’))
* Nginx builtin global/modules variables (Alpabetical index of variables with modules that makes it available <http://nginx.org/en/docs/varindex.html> ).Modules like ngx\_http\_core\_module, ngx\_http\_log\_module are already part of nginx (not needed to add them manually).

***$host, $uri,***

***$args*** (test it with <http://163.172.171.154/demo?name=anel> ) will return *name=anel.*

To obtain only anel from query try with***$arg\_name****!(based on the query string nginx compiles named variable for each parameter prefixed with arg),*

***$date\_local*** *(*holds ISO date time string for the local time *:* Sunday, 07-Oct-2018 13:52:59 UTC*)*

*Basic conditionals*

Use of nginx conditionals inside nginx location context is highly discouraged (<https://www.nginx.com/resources/wiki/start/topics/depth/ifisevil/> ) (use only **return** and **rewrite** inside location context, everything else could give **SIGSEGV**).

When using “**if ($arg\_name != “anel”)**” make space between if and ( , as well as between arguments and operator hence it will be an error.

## Rewrites and redirects:

**rewrite pattern uri**

**return status\_code message**

In case where status\_code being 307 variant (which is for redirects) return behavior changes it accept URI as a second parameter.

But when using : return 307 /thumb.png url will be redirected, but the url wil be changed from route from location context to redirected link (/thumb.png), what is a main difference between rewrite and redirects ! Redirect tells the client preforming the request where to go instead, while rewrite mutates request internaly.

Rewrite: In server context before matching the request to any location context, we should write : **rewrite ^/user/\w+ /greet** (meaning first argument: regex **^** ==starting with, **\w+** more than 1 word character) (meaning second argument: new url to rewrite). The browser will be pointing to the same location of first argument!

Note: When uri is rewritten it is also re-evaluated by nginx, as a completely new request !!! Meaning that new /greet will start from the top of nginx.conf and will get re-evaluated again ! Reevaluation makes rewrites more powerful but requires more system resources then return.

Feature of rewrite is ability to capture certain parts of the original url using standard regex capture groups (wrap that part in braces, and the capture group can be accessed with $). For capturing user name:

**rewrite ^/user/(\w+) /greet/$1**

So now /user/anel will be rewritten to /greet/anel. After /greet/anel will be reevaluated , skipping past rewrite and exact match will take priority instead (if exist) prefix match.

Passing the **last** flag:

Add new rewrite bellow the previous ; **rewrite ^/greet/anel /thumb.png.** First rewrite will match to the same route, it will be rewritten to /greet/anel and will be re-evaluted and will match the second rewrite and again will be rewritten as /thumb.png and served as such.

With last flag will tell url not to be rewritten any more. So in this case with modification of the first rewrite as: **rewrite ^/user/(\w+) /greet/$1** l**ast**

/user/anel will be rewritten to /greet/anel and after re-evaluation it will skip the second rewrite.

**rewrite ^/user/(\w+) /greet/$1**

Try files & named locations:

## Additional problems that occurred:

In order to start nginx use systemctl : **sudo systemctl start nginx && ps aux|grep nginx**

If there is an error in your nginx.conf file then: **sudo systemctl reload nginx will fail**

|  |
| --- |
| *root 10658 0.0 0.0 45408 2016 ? Ss 15:21 0:00 nginx: master process /usr/sbin/nginx -g daemon on; master\_process on;*  *www-data 10659 0.0 0.0 45776 3004 ? S 15:21 0:00 nginx: worker process*  *www-data 10660 0.0 0.0 45776 3004 ? S 15:21 0:00 nginx: worker process*  *www-data 10661 0.0 0.0 45776 3004 ? S 15:21 0:00 nginx: worker process*  *www-data 10662 0.0 0.0 45776 3004 ? S 15:21 0:00 nginx: worker process*  *www-data 10663 0.0 0.0 45776 3004 ? S 15:21 0:00 nginx: worker process*  *www-data 10664 0.0 0.0 45776 3004 ? S 15:21 0:00 nginx: worker process*  *www-data 10665 0.0 0.0 45776 3004 ? S 15:21 0:00 nginx: worker process*  *www-data 10666 0.0 0.0 45776 3004 ? S 15:21 0:00 nginx: worker process* |

User that is worker is *www-data* , because that is specified in */etc/nginx/nginx.conf* file (user *www-data*)

For temporary enabling/disabling services : **sudo service apache2 stop**

With systemd we can do the following: **systemclt start apache2**

**401 – unauthorized status code**

Still the same version:

sudo nginx -v

nginx version: nginx/1.10.3 (Ubuntu)

Not 1.15.5 as expected !

/usr/bin/nginx -V

nginx version: nginx/1.15.5

$ whereis nginx

/usr/bin/nginx.old

/usr/bin/nginx

/usr/sbin/nginx (this is problematic)

/usr/local/nginx/

/usr/share/nginx/

/etc/nginx/

$ type nginx

nginx is hashed (/usr/sbin/nginx)

$ which nginx

/usr/sbin/nginx

PATH is used to find executables, shell needs to search the path. Shell keep list of programs that has already found – hash

$ hash –l # to list the hashes

$ hash –r # to reset the hashes

$ hash –l # list again

To list the year of files use $ ls --full-time

Echo $PATH

How to add to $PATH = > export $PATH ????