NGINX

NGINX is open source software for web serving, reverse proxying, caching, load balancing, media streaming, and more. a web server designed for maximum performance and stability. In addition to its HTTP server capabilities, NGINX can also function as a proxy server for email (IMAP, POP3, and SMTP) and a reverse proxy and load balancer for HTTP, TCP, and UDP servers

NGINX created in 2004 and handles 10000 of concurrent connections

Apache	Nginx
• Prefork mode, set of processes each process serve a single request at a time. • Quite slow • Accept the request up to predefined number and reject the rest • <Directory "/var/web/dir1"> Options +Indexes </Directory> • .htaccess enables you to configure website permissions without alter server configuration	• Single nginx can serve multiple requests concurrently based on the resources available for nginx • Unlike apache there is no programming language empeded with it • All requests for dynamic content dealed with separate process like FPM and reverse proxy back to the client via nginx • Resources are separated between nginx and programming language service but almost time it will be a static content with no need for service programming • Faster • location /site { . . . }

Install nginx using package manager

1- # yum install epel-release -y
2- # yum install nginx -y
3- # systemctl start nginx
4- # systemctl enable nginx
5- # ps aux | grep nginx
6- # firewall-cmd --zone=public --permanent --add-service=http
7- # firewall-cmd --zone=public --permanent --add-service=https
8- # firewall-cmd --reload

Install nginx from source

1- Download Binary
   # wget http://nginx.org/download/nginx-1.20.2.tar.gz
2- # tar xvfz nginx-1.20.2.tar.gz
3- # yum groupinstall "Development Tools" -y
4- # yum install pcre* zlib* openssl* -y
   Note: pcre is library for rewrite module allow you to add regular expression (~,=,$,…)
5- # yum install gd gd-devel php-gd
   Note: HTTP image filter module requires the GD library
6- # ./configure --sbin-path=/usr/sbin/nginx --conf-path=/etc/nginx/nginx.conf --error-log-path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --with-pcre --pid-path=/var/run/nginx.pid --with-http_ssl_module --with-http_image_filter_module=dynamic --modules-path=/etc/nginx/modules/ --with-http_v2_module
7- # make
8- # make install
9- # nginx -v

To manage nginx with systemd

1- # vim /lib/systemd/system/nginx.service
2- Past the following:

```
[Unit]
Description=The NGINX HTTP and reverse proxy server
After=syslog.target network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t
ExecStart=/usr/sbin/nginx
ExecReload=/usr/sbin/nginx -s reload
ExecStop=/bin/kill -s QUIT $MAINPID
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

Note: Nginx in windows has:

- poor performance
- Single worker process
- Unsupported modules

Nginx.conf

Context	Directive
Section or scope Examples: http { .... } Can be nested http { .... # server Context server { .... } } Each context inherits from its parent	Examples: # server directive listen: 80; server_name: example.com, www.example.com

Contexts:

1- **Http context**: for anything related to http
2- **Server context**: where we define virtual host
3- **Location context**: for matching Uri locations for incoming request

How to list file context

semanage fcontext -l <file-name>
ls -Z <dir-name>

To change file context

semanage fcontext -at <context> “/sites(/.*)?”
restorecon -Rv <dir-name>

Creating a Virtual Host

vim /etc/nginx/nginx.conf

```
events {}

http {

  include mime.types;

  server {

    listen 80;
    server_name 10.250.10.245;

    root /sites/demo;
  }
}
```

Location blocks

```
events {}

http {

  include mime.types;

  server {

    listen 80;
    server_name 10.250.10.245;

    root /sites/demo;

    # Prefix match
    location /greet {  #any thing starts with /greet
      return 200 'Hello from NGINX "/greet" location.';
    }
    # Preferential Prefix match
    location ^~ /Greet2 { #more priority
      return 200 'Hello from NGINX "/greet" location.';
    }

    # # Exact match
    # location = /greet {
    #   return 200 'Hello from NGINX "/greet" location - EXACT MATCH.';
```

```
  # }

  # # REGEX match - case sensitive
  # location ~ /greet[0-9] { #any thing starts with /greet then number from 0 to 9
  #   return 200 'Hello from NGINX "/greet" location - REGEX MATCH.';
  # }

  # REGEX match - case insensitive
  location ~* /greet[0-9] { #any thing starts with /greet then number from 0 to 9
    return 200 'Hello from NGINX "/greet" location - REGEX MATCH INSENSITIVE.';
  }
 }
}
```

The order

1- Exact match
2- Preferential prefix
3- Regular expression
4- Prefix match

Variables

A- We set our selves

Ex. Set $var '.....';

B- Nginx module variables

Ex. $http, $uri, $args

```
http {

 include mime.types;

 server {

  listen 80;
  server_name 10.250.10.245;

  root /sites/demo;

  set $mon 'No';

  # Check if weekend
  if ( $date_local ~ 'Monday' ) {
    set $mon 'Yes';
  }
```

```
  location /is_monday {

    return 200 $mon;
  }
 }
}
```

Rewrites & Redirects

```
events {}

http {

 include mime.types;

 server {

  listen 80;
  server_name 10.250.10.245;

  root /sites/demo;

  rewrite ^/user/(\w+) /greet/$1 last; # ^ means anything starts with, \w+ means word char or more
  # /greet/$1 means /greet/john, last means this is the last rewrite
  rewrite ^/greet/john /thumb.png;

  location /greet {

    return 200 "Hello User";
  }

  location = /greet/john {
    return 200 "Hello John";
```

```
  }
 }
}
```

server_name 10.250.10.245

Try Files & Named Locations

```
    events {}

http {

 include mime.types;

 server {

  listen 80;
  server_name 10.250.10.245;

  root /sites/demo;


  try_files $uri /cat.png /greet @friendly_404; # serve the first one if not exist serve the next

  # try_files /cat.png   will serve this and don't care with the uri

  location @friendly_404 { # this called named location
   return 404
```

Logging

Error log: - all logs for error

Access log: - logs for all requests to the server

Hints:

1- Logging is enabled by default
2- The location /var/log/nginx
3- 404 exists in access log
4- You can create multiple access log in the same location and results stored at all of them

Examples

```
  events {}

http {

 include mime.types;

 server {

  listen 80;
  server_name 10.250.10.245;
```

```
  root /sites/demo;

  location /secure {

    # Add context specific log
    access_log /var/log/nginx/secure.access.log;

    # Disable logs for context
    #access_log off;

    return 200 "Welcome to secure area.";
  }

 }
}
```

Inheritance & Directive types

Directive types:

1- **Array directive**: you can declare many times without overwrite
   Example:

```
access_log /var/log/nginx/secure.access.log;
access_log /var/log/nginx/secure.access-2.log;
```

   Note: any context under array directive inherits from it else if you configured access_log in it this will overwrite

2- **Standard directive**: declared once in a given context, and inheritance the same as array directive child can overwrite it with its own declaration

   Example:

```
root /sites/demo;
```

3- **Action directive**: invoke actions in the configuration, inheritance don’
   Example:

```
return 200 "Welcome to secure area.";
```

```
try_files /cat.png
```

```
rewrite ^/greet/john /thumb.png;
```

summary example :

```
events {}

######################
# (1) Array Directive
######################
# Can be specified multiple times without overriding a previous setting
# Gets inherited by all child contexts
# Child context can override inheritance by re-declaring directive
access_log /var/log/nginx/access.log;
access_log /var/log/nginx/custom.log.gz custom_format;

http {

 # Include statement - non directive
 include mime.types;

 server {
  listen 80;
  server_name site1.com;

  # Inherits access_log from parent context (1)
 }

 server {
  listen 80;
  server_name site2.com;

  ########################
  # (2) Standard Directive
  ########################
```

```
  # Can only be declared once. A second declaration overrides the first
  # Gets inherited by all child contexts
  # Child context can override inheritance by re-declaring directive
  root /sites/site2;

  # Completely overrides inheritance from (1)
  access_log off;

  location /images {

   # Uses root directive inherited from (2)
   try_files $uri /stock.png;
  }

  location /secret {
   ########################
   # (3) Action Directive
   ########################
   # Invokes an action such as a rewrite or redirect
   # Inheritance does not apply as the request is either stopped (redirect/response) or re-evaluated (rewrite)
   return 403 "You do not have permission to view this.";
  }
 }
}
```

PHP Processing

- Nginx has no server-side language processing, we will configure standalone service (php-fpm) Nginx will pass the request for processing then receive the response and forward it to the client and nginx here acts as **reverse proxy**

installation:

1- # yum install php-fpm -y
2- # systemctl start php-fpm.service
3- # systemctl enable php-fpm.service
4- # systemctl list-units | grep php

Fastcgi pass request to php-fpm through Unix socket

```
user www-data;

events {}

http {

 include mime.types;

 server {

  listen 80;
  server_name 10.250.10.245;

  root /sites/demo;
```

```
  index index.php index.html;
  #serve index.php first if exist, if not then serve index.html

  location / {
    try_files $uri $uri/ =404;
    # only for static content
  }

  location ~\.php$ {
    # Pass php requests to the php-fpm service (fastcgi)
    include fastcgi.conf;
    fastcgi_pass unix:/run/php/php7.1-fpm.sock;
    # unix socket exists in /etc/php-fpm.d/www.conf  beside listen directive
  }

 }
}
```

5- # echo "<?php phpinfo();?>" > /sites/demo/info.php
6- # systemctl reload nginx
7- # ps aux | grep nginx
8- # ps aux | grep php
   to see the user who run the service, so at the top of configuration write
   user www-data;
9- # echo '<h1> date: <?php echo date ("l jS F");?></h1>' /sites/demo/index.php

Worker processes

systemctl status nginx

- **process master**: actual nginx service
- **process worker**: responses for clients requests

  Default number for nginx worker is one but to modify write in nginx.conf
  worker_processes 3; > in the main context
  then reload configuration

- increase nginx worker doesn't necessary increase performance
- worker processes request handling is asynchronous
- single nginx worker process run on a single cpu core
- to know the number of cores > # nproc # lscpu
- worker_processes auto; > make worker process per core
- # ulimit -n > max files can be opened at the same time

Note: we can reconfigure **pid** at nginx.conf
pid /var/run/new_nginx.pid;

Example:

```
user www-data;

worker_processes auto;

events {
  worker_connections 1024; # number of connections each worker process can accept
}

http {

  include mime.types;

  server {

    listen 80;
    server_name 167.99.93.26;

    root /sites/demo;

    index index.php index.html;

    location / {
      try_files $uri $uri/ =404;
    }

    location ~\.php$ {
      # Pass php requests to the php-fpm service (fastcgi)
      include fastcgi.conf;
      fastcgi_pass unix:/run/php/php7.1-fpm.sock;
```

```
  }

 }
}
```

Buffers & Timeouts

Buffering: read data from memory or ram before writing it to its own destination, the request is written in memory, if we have small buffer then we will write some of it to disk

Time out: cut of time for a given event, prevent client to have endless stream of data

Example:

```
user www-data;

worker_processes auto;

events {
 worker_connections 1024;
}

http {

 include mime.types;

 # Buffer size for POST submissions
 client_body_buffer_size 10K; # if increased it will waste memory  and if it small piece of it will be written to disk
 client_max_body_size 8m; # don't accept any post request more than 8m else you will have 413 error(too large entity)

 # Buffer size for Headers
 client_header_buffer_size 1k; # amount of memory to read request headers

 # Max time to receive client headers/body
 client_body_timeout 12; # 12 is in millisecond and this time is between read operations from buffer
 client_header_timeout 12;

 # Max time to keep a connection open for
 keepalive_timeout 15;

 # Max time for the client accept/receive a response
 send_timeout 10;

 # Skip buffering for static files and read data from disk
 sendfile on;
```

```
 # Optimise sendfile packets
 tcp_nopush on;

 server {

  listen 80;
  server_name 167.99.93.26;

  root /sites/demo;

  index index.php index.html;

  location / {
   try_files $uri $uri/ =404;
  }

  location ~\.php$ {
   # Pass php requests to the php-fpm service (fastcgi)
   include fastcgi.conf;
   fastcgi_pass unix:/run/php/php7.1-fpm.sock;
  }

 }
}
```

Adding Dynamic Modules

1- You need to install nginx from source
2- # ./configure -- help | grep dynamic
3- # ./configure …….. --xxxxxx=dynamix -- module-path=/etc/nginx/modules
4- # make
5- # make install
6- # nginx -v
7- In nginx.conf
   Load_module modules/xxxxxx.so > xxxxx is module name

Example:

```
user www-data;

worker_processes auto;

load_module modules/ngx_http_image_filter_module.so;

events {
 worker_connections 1024;
}

http {
```

```
  include mime.types;

  # Buffer size for POST submissions
  client_body_buffer_size 10K;
  client_max_body_size 8m;

  # Buffer size for Headers
  client_header_buffer_size 1k;

  # Max time to receive client headers/body
  client_body_timeout 12;
  client_header_timeout 12;

  # Max time to keep a connection open for
  keepalive_timeout 15;

  # Max time for the client accept/receive a response
  send_timeout 10;

  # Skip buffering for static files
  sendfile on;

  # Optimise sendfile packets
  tcp_nopush on;

  server {

    listen 80;
    server_name 167.99.93.26;

    root /sites/demo;

    index index.php index.html;

    location / {
      try_files $uri $uri/ =404;
    }

    location ~\.php$ {
      # Pass php requests to the php-fpm service (fastcgi)
      include fastcgi.conf;
      fastcgi_pass unix:/run/php/php7.1-fpm.sock;
    }

    location = /thumb.png {
      image_filter rotate 180;
    }

  }
}
```

Headers & Expires

- We need to store resource at client side and not to order it from server each time

  Example:

```
user www-data;

worker_processes auto;

events {
  worker_connections 1024;
}

http {

  include mime.types;

  server {

    listen 80;
    server_name 167.99.93.26;

    root /sites/demo;

    index index.php index.html;

    location / {
```

```
    try_files $uri $uri/ =404;
  }

  location ~\.php$ {
   # Pass php requests to the php-fpm service (fastcgi)
   include fastcgi.conf;
   fastcgi_pass unix:/run/php/php7.1-fpm.sock;
  }

  location ~* \.(css|js|jpg|png)$ {      # anything ends with these
   access_log off;
   add_header my_header “helloworld”;      # generic header
   # note my_header is header name and hello world is header value
   add_header Cache-Control public; # tell the client that this resource can be cashed in any way
   add_header Pragma public;
   add_header Vary Accept-Encoding; # the response can vary based on the request header
   expires 1M; # for the response
  }

 }
}
```

```
# systemctl reload nginx
# curl -I http://URL/thumb.png
# curl -I http://URL/style.css
```

Compressed Responses with gzip

- We need to compress resource file at server side to minimize the time consumed to transfer the resource from server to client but we need to decompress it again at client side

  Example:

```
    user www-data;

worker_processes auto;

events {
 worker_connections 1024;
}

http {
```

```
 include mime.types;

 gzip on;                    # global but any child can override
 gzip_comp_level 3;    # with increase its value consume more resources

 gzip_types text/css;
 gzip_types text/javascript;

 server {

  listen 80;
  server_name 167.99.93.26;

  root /sites/demo;

  index index.php index.html;

  location / {
   try_files $uri $uri/ =404;
  }

  location ~\.php$ {
   # Pass php requests to the php-fpm service (fastcgi)
   include fastcgi.conf;
   fastcgi_pass unix:/run/php/php7.1-fpm.sock;
  }

  location ~* \.(css|js|jpg|png)$ {
   access_log off;
   add_header Cache-Control public;
   add_header Pragma public;
   add_header Vary Accept-Encoding;
   expires 1M;
  }

 }
}
```

```
# systemctl reload nginx
# curl -I -H “Accept-Encoding : gzip ,deflate” http://URL
```

FastCGI Cache

Micro cache: Server-side cache that allows us to store dynamic language responses
So, it minimizes server-side language processing

Micro cache: Store FPM response at nginx server side to server it directly from server in the next requested time without referring to FPM

```
user www-data;

worker_processes auto;

events {
  worker_connections 1024;
}

http {

  include mime.types;

  # Configure microcache (fastcgi)
  fastcgi_cache_path /tmp/nginx_cache levels=1:2 keys_zone=ZONE_1:100m inactive=60m;
  #  levels=1:2 , 1 the last number , 2 the two numbers before the last one
  fastcgi_cache_key "$scheme$request_method$host$request_uri";
  # scheme http&https  , request as GET , host as domain.com
```

```
# if you remove scheme then you will need an entry for http and another for https
add_header X-Cache $upstream_cache_status;   # to know the response come from cache or not

server {

 listen 80;
 server_name 167.99.93.26;

 root /sites/demo;

 index index.php index.html;

 # Cache by default
 set $no_cache 0;  # skip cache

 # Check for cache bypass
 if ($arg_skipcache = 1) {
  set $no_cache 1;
 }

 location / {
  try_files $uri $uri/ =404;
 }

 location ~\.php$ {
  # Pass php requests to the php-fpm service (fastcgi)
  include fastcgi.conf;
  fastcgi_pass unix:/run/php/php7.1-fpm.sock;

  # Enable cache
  fastcgi_cache ZONE_1;
  fastcgi_cache_valid 200 60m;
  fastcgi_cache_bypass $no_cache;
  fastcgi_no_cache $no_cache;
 }

 }
}
```

- We will use apache bench to test performance
- # yum install httpd-tools
- # ab
- # ab - n 100 - c 10 http://URL/ n for requests ,, c for concurrency connection
- In the Output you will see request per second & time per request
- To increase process time add sleep function to the php file
  <?php sleep(1); ?>
  <h1> date: <?php echo date (“l jS F”); ?> </h1>

- # curl -I http://URL/

x-cache: HIT >> this is from cache
x-cache: miss >> not from cache

- # curl -I http://URL/?skipcache=1

  x-cache: BYPASS

HTTP2

- From version 1.9.5 nginx include HTTP2
- HTTP2 is a binary protocol where http is textual protocol
- Binary reduce the chance of errors during data transfer
- HTTP2 compress its response header so it reduces transfer time
- HTTP2 uses persistent connections
- HTTP2 uses multiplex streaming, for example .css & .js & .html can be combined in a single stream of binary data and transmitted over a single connection, while http requires a dedicated connection for each resource
- HTTP2 support server push, so you can request index.html and the server replies with .html and .css and .js
- In HTTP one connection handle one request, here we will stick with connection limit and at some cases will need to wait till some connections ended
- HTTP2 require SSL or HTTPS after adding module
- To add HTTP2 # ./configure -- help | grep http_v2
- # ./configure …….. -- with-http_ssl_module – with-http_v2_module
- # make
- # make install
- # systemctl restart nginx
- Then open Firefox and navigate to development mode, if you look at network tap then reload you will find 3 connection that have http
- To make SSL certificate
- # mkdir /etc/nginx/ssl
- # openssl req -x 509 -days 10 -nodes -newkey rsa:2048 -keyout /etc/nginx/ssl/self.key -out /etc/nginx/ssl/self.crt

- # systemctl restart nginx
- Open Firefox you will find https and ssl working so to enable http2
  Server{
  Listen 443 ssl http2;
  }
- # systemctl reload nginx
- If you refresh Firefox you will find http2 working now

Examples:

```
    user www-data;

worker_processes auto;

events {
 worker_connections 1024;
}

http {

 include mime.types;

 server {

  listen 443 ssl http2;
  server_name 167.99.93.26;

  root /sites/demo;

  index index.php index.html;

  ssl_certificate /etc/nginx/ssl/self.crt;
  ssl_certificate_key /etc/nginx/ssl/self.key;

  location / {
   try_files $uri $uri/ =404;
  }

  location ~\.php$ {
   # Pass php requests to the php-fpm service (fastcgi)
   include fastcgi.conf;
   fastcgi_pass unix:/run/php/php7.1-fpm.sock;
  }

 }
}
```

http2 server push

- We will use a tool called nghttp
- to install it # apt-get install nghttp2-client
- nghttp -nys https://URL/index.html n param to discard responses ys to print response statistics and ignore certificate
- nghttp -nysa https://URL/index.html
  a is assets for the request

Example:

```
    user www-data;

worker_processes auto;

events {
 worker_connections 1024;
}

http {

 include mime.types;

 server {

  listen 443 ssl http2;
  server_name 167.99.93.26;

  root /sites/demo;

  index index.php index.html;

  ssl_certificate /etc/nginx/ssl/self.crt;
  ssl_certificate_key /etc/nginx/ssl/self.key;

  location = /index.html {
   http2_push /style.css;
   http2_push /thumb.png;
  }

  location / {
   try_files $uri $uri/ =404;
  }

  location ~\.php$ {
   # Pass php requests to the php-fpm service (fastcgi)
   include fastcgi.conf;
   fastcgi_pass unix:/run/php/php7.1-fpm.sock;
  }

 }
}
```

- # systemctl reload nginx
- # nghttp -nys https://URL/index.html

  Note: * indicates server push responses

Security

HTTPS (SSL)

- HTTPS: to prevent http we will redirect to https

```
http {

include mime.types;

 # Redirect all traffic to HTTPS
 server {
  listen 80;
  server_name <server-ip>;
  return 301 https://$host$request_uri;  #301 will redirect the request
 }
```

- The above example if you go to the ip it will redirect you to https directly by default
- # systemctl reload nginx
- # curl -IK http://URL/

  SSL: secure socket layer **replaced** with **TLS**: transport layer security

- # openssl dhparam 2048 -out /etc/nginx/ssl/dhparam.pem

    2048 same as rsa in key

  Example:

```
 user www-data;

worker_processes auto;

events {
 worker_connections 1024;
}

http {

 include mime.types;

 # Redirect all traffic to HTTPS
 server {
  listen 80;
  server_name 167.99.93.26;
  return 301 https://$host$request_uri;
 }
```

```
 server {

  listen 443 ssl http2;
  server_name 167.99.93.26;

  root /sites/demo;

  index index.html;

  ssl_certificate /etc/nginx/ssl/self.crt;
  ssl_certificate_key /etc/nginx/ssl/self.key;

  # Disable SSL as now it is encrypted using TLS only not SSL
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

  # Optimise cipher suits
  ssl_prefer_server_ciphers on;
  # search for up to date combination
  ssl_ciphers ECDH+AESGCM:ECDH+AES256:ECDH+AES128:DH+3DES:!ADH:!AECDH:!MD5;

  # Enable Diffie hellman Params for secure key exchange
  ssl_dhparam /etc/nginx/ssl/dhparam.pem;

  # Enable HSTS to tell the browser not to load any http
  add_header Strict-Transport-Security "max-age=31536000" always;

  # SSL sessions
  # cache handshake
  ssl_session_cache shared:SSL:40m;
  # shared for all worker , SSL is the name , 40m is the size
  ssl_session_timeout 4h;
  # provide the browser with a ticket that validate ssl session, and allow to bypass reading from cache of the
session, optimize server resources and make the browser to talk to the cache directly
  ssl_session_tickets on;

  location / {
    try_files $uri $uri/ =404;
  }

  location ~\.php$ {
    # Pass php requests to the php-fpm service (fastcgi)
    include fastcgi.conf;
    fastcgi_pass unix:/run/php/php7.1-fpm.sock;
  }

 }
}
```

server name

- # systemctl reload nginx

Rate Limiting

- Module in nginx

  Security: brute force protection
  Reliability: prevent traffic spikes
  Shaping: service priority like download service

- We will use a tool called **siege**

  # apt-get install siege
  # siege -v -r 2 -c 5 https://URL/thumb.png
  v: verbose r: number of tests c: concurrent connection

  Example:

```
user www-data;

worker_processes auto;

events {
 worker_connections 1024;
}

http {

 include mime.types;

 # Define limit zone
 limit_req_zone $request_uri zone=MYZONE:10m rate=1r/s;

 # Redirect all traffic to HTTPS
 server {
  listen 80;
  server_name 167.99.93.26;
  return 301 https://$host$request_uri;
 }

 server {

  listen 443 ssl http2;
  server_name 167.99.93.26;

  root /sites/demo;

  index index.html;

  ssl_certificate /etc/nginx/ssl/self.crt;
  ssl_certificate_key /etc/nginx/ssl/self.key;
```

```
  # Disable SSL
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

  # Optimise cipher suits
  ssl_prefer_server_ciphers on;
  ssl_ciphers ECDH+AESGCM:ECDH+AES256:ECDH+AES128:DH+3DES:!ADH:!AECDH:!MD5;

  # Enable DH Params
  ssl_dhparam /etc/nginx/ssl/dhparam.pem;

  # Enable HSTS
  add_header Strict-Transport-Security "max-age=31536000" always;

  # SSL sessions
  ssl_session_cache shared:SSL:40m;
  ssl_session_timeout 4h;
  ssl_session_tickets on;

  location / {
    # do all the 6 as fast as you can
    limit_req zone=MYZONE burst=5 nodelay;
    try_files $uri $uri/ =404;
  }

  location ~\.php$ {
    # Pass php requests to the php-fpm service (fastcgi)
    include fastcgi.conf;
    fastcgi_pass unix:/run/php/php7.1-fpm.sock;
  }

 }
}
```

systemctl reload nginx

siege -v -r 2 -c 5 https://URL/thumb.png

Basic auth

yum install httpd-tools

htpasswd - c /etc/nginx/.htpasswd user1

Password:
Password:

C create it in a file

cat /etc/nginx/.htpasswd

```
user www-data;

worker_processes auto;

events {
 worker_connections 1024;
}

http {

 include mime.types;

 # Define limit zone
 limit_req_zone $request_uri zone=MYZONE:10m rate=1r/s;

 # Redirect all traffic to HTTPS
 server {
  listen 80;
  server_name 167.99.93.26;
  return 301 https://$host$request_uri;
 }

 server {

  listen 443 ssl http2;
  server_name 167.99.93.26;

  root /sites/demo;

  index index.html;

  ssl_certificate /etc/nginx/ssl/self.crt;
  ssl_certificate_key /etc/nginx/ssl/self.key;

  # Disable SSL
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

  # Optimise cipher suits
```

```
  ssl_prefer_server_ciphers on;
  ssl_ciphers ECDH+AESGCM:ECDH+AES256:ECDH+AES128:DH+3DES:!ADH:!AECDH:!MD5;

  # Enable DH Params
  ssl_dhparam /etc/nginx/ssl/dhparam.pem;

  # Enable HSTS
  add_header Strict-Transport-Security "max-age=31536000" always;

  # SSL sessions
  ssl_session_cache shared:SSL:40m;
  ssl_session_timeout 4h;
  ssl_session_tickets on;

  location / {
    auth_basic "Secure Area";
    auth_basic_user_file /etc/nginx/.htpasswd;
    try_files $uri $uri/ =404;
  }

  location ~\.php$ {
    # Pass php requests to the php-fpm service (fastcgi)
    include fastcgi.conf;
    fastcgi_pass unix:/run/php/php7.1-fpm.sock;
  }

 }
}
```

```
# systemctl reload nginx
```

Harding nginx

- # apt-update
- # apt-upgrade
- # nginx -v search for its fixes and patches

Example:

```
user www-data;

worker_processes auto;

events {
 worker_connections 1024;
}

http {

 include mime.types;
 # to hide nginx version from the output of curl command
 server_tokens off;

 # Redirect all traffic to HTTPS
 server {
  listen 80;
  server_name 167.99.93.26;
  return 301 https://$host$request_uri;
 }

 server {

  listen 443 ssl http2;
  server_name 167.99.93.26;

  root /sites/demo;

  index index.html;
  # to prevent people from taking your website content and calling it
  add_header X-Frame-Options "SAMEORIGIN";
  # cross site scripting
  add_header X-XSS-Protection "1; mode=block";

  ssl_certificate /etc/nginx/ssl/self.crt;
  ssl_certificate_key /etc/nginx/ssl/self.key;

  # Disable SSL
  ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

  # Optimise cipher suits
  ssl_prefer_server_ciphers on;
  ssl_ciphers ECDH+AESGCM:ECDH+AES256:ECDH+AES128:DH+3DES:!ADH:!AECDH:!MD5;

  # Enable DH Params
  ssl_dhparam /etc/nginx/ssl/dhparam.pem;
```

```
  # Enable HSTS
  add_header Strict-Transport-Security "max-age=31536000" always;

  # SSL sessions
  ssl_session_cache shared:SSL:40m;
  ssl_session_timeout 4h;
  ssl_session_tickets on;

  location / {
    try_files $uri $uri/ =404;
  }

  location ~\.php$ {
    # Pass php requests to the php-fpm service (fastcgi)
    include fastcgi.conf;
    fastcgi_pass unix:/run/php/php7.1-fpm.sock;
  }

 }
}
```

Fastcgi

- Remove unneeded modules
  as http_autoindex_module

  # ./configure …….-- without-http_autoindex_module

Let’s encrypt ssl certificate

- In order to generate certificate and renewal we need a tool called **certbot**
- Install certbot
- # certbot --nginx
  you must put server_name in the config file of nginx
- # ls -l /etc/letsencrypt/live/my-domain/
- Letsencrypt certificate are valid for 90 days only
- # certbot renew
- # certbot renew --dry-run
- to do daily renew make cronjob
  # crontab -e
    @daily certbot renew

    @daily means midnight each day

- # crontab -l

Reverse proxy & LB

- Install nginx on your computer
- Create config file using **touch**
- # nginx -c /users/….../nginx/nginx.conf
  C for config

- # Curl http://localhost:port
- Create php server
  >php -S localhost:9999

- On Firefox
  http://localhost:9999
  will open index.html if it exists at nginx home dir.

Reverse Proxy

- Client nginx **PHP, Ruby, NodeJS**

  >php -S localhost:9999 resp.txt
  # curl http://localhost:9999

- How to access php from nginx in conf of nginx

```
Location /php{
        Proxy_pass 'http://localhost:9999/';
        }
```

- # nginx - s reload
- # curl http://localhost:8888/php
  Hello from php server
- # curl http://localhost:8888
  Hello from nginx

- In nginx.conf
  If we need to call something in the internet

  Location /nginxorg {
  Proxy_pass 'https://nginx.org/';
  }

- # nginx -s reload

- # curl http://localhost:8888/nginxorg

>touch show_request.php

<? php
echo 'Path: '.$_SERVER['REQUEST_URI'];

save

php -S localhost:9999 show_request.php
nginx -s reload
curl http://localhost:8888/php

Path:/
/ is the one exists at the end of the URI in the config file

- If you remove / from nginx config uri the output will be
Path: /php

- # curl http://localhost:8888/php/some/URI
Path://some/url

To pass header to the client in config

Location /php {
Add_header proxied nginx;
}

- # nginx -s reload
- # curl -I http://localhost:8888/php

- To send the header to the php server we will edit show_request.php

<?php

var_dump (getallheader());

in nginx.conf

location /php{

proxy_set_header proxied nginx;

}

- # nginx -s reload
- # curl http://localhost:8888/php/some/url
["proxied"]

Load balancer

```
# echo ‘php server 1’ > s1
# echo ‘php server 2’ > s2
# echo ‘php server 3’ > s3

# php -S localhost:10001 s1
# php -S localhost:10002 s2
# php -S localhost:10003 s3

# curl http://localhost:10001
  php server 1

# vim loadbalancer.conf
        Location /{
         proxy_pass ‘http://;ocalhost:10001/’;
}

# nginx -c /user/---/---/loadbalancer.conf
# curl http://localhost:8888
  php server1

# while sleep 0.5; do curl http://localhost:8888;done
  php server 1
  php server 2
```

So, let’s implement LB

In nginx.conf

```
http {
upstream php_servers {
server localhost:10001;
server localhost:10002;
server localhost:10003;
}
server {
listen 8888;
location /{
proxy_pass http://php_server;
}
}
}
```

- # nginx -s reload
- # while sleep 1; do

LB method is round-robin

LB options

- **LB** using IP hash (Sticky Sessions)

  In **config**

  ```
  upstream php_servers{
  ip_hash;
  server ……..;
  server ……..;
  server ……..;
  }

  # nginx -s reload
  # while ……
    server1
    server2
    server3
  ```

- **LB** using load (send to the one that has less load)

  slow.php

  ```
  <?php
    Sleep (20);
    Echo “sleepy server finally done! \n”;

  # php -S localhost:10001 slow.php
  ```

  In **config**

  ```
  upstream php_servers{
          least_conn;
          server ……..;
          server ……..;
          server ……..;

  # nginx -s reload
  # while ……...
  ```

Nginx init script from github

Copy nginx RAW URL

cd /etc/init.d/ >> scripts for different Linux services
wget https://...../nginx
chmod +x nginx
update-rc.d -f nginx defaults
echo “NGINX_CONF_FILE=/etc/nginx/nginx.conf” > /etc/default/nginx
echo “DAEMON=/usr/bin/nginx” >> /etc/default/nginx
service nginx status

To update or downgrade nginx

1- download the needed version
2- . /configure
3- # nginx -v
4- # service nginx reload

GeoIP

- Module …… --with-http_geoip_module
- #. /configure ……. –with-http_geoip_module
- # make
- # make install
- # nginx -V
- # nginx -s reload
- # mkdir /etc/nginx/geoip
- # cd /etc/nginx/geoip

** visit dev.maxmind.com
Click on geolite legacy Downloadable DB

- Download geolite country db
- Download geolite city db
- # wget https://.....
- # wget https://.....

In nginx.conf

```
http {
        geoip_country   /etc/nginx/geoip/geoip.dat;
        geoip_city   /etc/nginx/geoip/geolitecity.dat;
}
```

- # nginx -t
- # service nginx reload

In nginx.conf

```
Location /geo_country{

Return 200 “visiting from: $geoip_country_name”;
}
```

- # systemctl reload nginx
- In Firefox open URL /geo_country
- In Firefox open URL /geo_city

Video streaming

MP4 module

#. /configure …… --with-http_mp4_module

In nginx.conf

```
Location ~ \.mp4${
      root /sites/downloads/;
      mp4;
       mp4_buffer_size 4m;
       mp4_max_buffer_size 10m;
}
```

service nginx reload

Test using firefox

http://……/duck.mp4?start=15

here will start from second 15

http://……/duck.mp4

start from the beginning