Introduction

Path Options

- `configure` command: specify directory or file paths for a variety of elements

```
./configure --help

# --SWITCH=DEFAULT # Usage
--prefix=/usr/local/nginx       # Base folder Nginx installed
--sbin-path=/sbin/nginx                    # Nginx Binary file
--conf-path=/conf/nginx.conf    # Main Config File
--error-log-path=logs/error.log               # Error Log
--pid-path=/logs/nginx.pid                     # pid file
--lock-path=/logs/nginx.lock                    # Lock File
--with-perl=             # Perl Binary File, used to run Perl scripts
--http-log-path
--http-proxu-temp-path
--http-fastcgi-temp-path
--builddir
```

Miscellaneous Options

Options available in the configuration script

```
--with-mail # enables mail sv proxy module. Sups POP3, IMAP4, SMTP.
Disabled by default
--with-mail_ssl_module # Enables SSL support for the mail server proxy
--without-mail_pop3_module # Disables POP3 module for the mail server
proxy. Enabled by default when mail server proxy module is enabled
--without-mail_imap_module
--without-mail_smtp_module
--with-rtsig_module # enables rtsig
--with-select_module # enables select module. Enabled by default
--without-select_module
--with-poll_module # poll event notification mecanism
```

```
--user=... # Default user account to start nginx worker processes. Used
only if you do not specify the group directive in the configuration file
--group=... # DEfault user group to start Nginx worker processes. Used only
if you do not specify the group directive in the configuration file
```

```
--with-ipv6 # Enables IPv6
--without-http
--without-http-cache
--add-moudle=PATH # Adds a third-party module to the compile process.
--with-debug # Enables aditinal debugging information to be logged
```

Config Examples

Regular HTTP and HTTPS Servers

- HTTP and HTTPS content enabled
- Mail-related options disabled

```
./configre --user=www-data --group=www-data --with-http_ssl_module --with-
http_realip_module
```

Mail Server Proxy

```
./configure --user=www-data --group=www-data --with-mail --with-
mail_ssl_module
```

Compiling and Installing the Program

Once the configure script is succesfully executed you can proceed with compiling Nginx by the make command in the project source directory

```
make
```

A succesful build shoudl result in the appearance of a final message ````make: leaving directory```

The next step is installing the application

```
make install
```

It performs a fre simple operations copying binaries and config files to the install folder. Also creates directories to store log and HTML files.

Controlling Nginx Service

The default location for the output files is /usr/local/nginx.

User And Group

A very common source of trouble wieh setting up Nginx is invalid file access per,issions. You often end up getting 504 Firbidden HTTP errors.

- Nginx master process: This should be started as root, to open TCP sockets on any ports. IF you do not start as root, std ports such 80 or 554 will not be accessible.
- Nginx Worker processes: Automatically spawned by the master process under the account you specified in the configuration file with the user directive.

Starting and Stopping the Daemon

You can start nginx by running Nginx binary without any switches. You may control the daemon by stopping it, restarting it, or simply reloading its configuration. Controlling is done by sending signals to the process using the nginx -s command

- nginx -s stop
- nginx -s quit
- nginx -s reopen
- nginx -s reload

An alternative way to terminate the process in desperate cases only is to use the `kill` or ````killall``` commands with root privileges

```
killall nginx
```

Testing Config

Testing the validity of your config will become crucial if you constantly tweak yoyr server setup. The following command will be useful to allows you to check the syntax, validity and integrity of your configuration

```
/usr/local/nginx/sbin/nginx -t
```

The -t swtich stands for test configuration. Nginx will parse the configuration anew and let you know whether is valid or not. A Valid coinfiguration file does not necssarily eman Nginx will start, thouhgh as there might be additional probles such as socket issues, invalid paths, or incorrect access permissions.

- Manipulating your configuration files when server is in produciton is a dangerous thing to do and should be avoided when possible.
- The best practice in this case is to pace your new configration into a separate teporary file and run the test on that file.
- Nginx makes it possible by offering the -c switch

```
./nginx -t -c /home/username/test.conf
```

This command will [arse /home/alex/test.conf and make sure it is a valid Nginx configuration file. When its done, after making sure that new file is valid, proceed to replacing your current confguration file and reload the server configuration.

```
cp -i /home/alex/test.conf /usr/local/nginx/conf/nginx.conf
./nginx -s reload
```

Adding Nginx as a System Service

In this section we will create a script that will tansform the Nginx daemon into an actual system service. The daemon will be controllable using standard commands and it will be launched automatically con system startup and stopped on system shutdown

The Linux Based system startup process is managed by a daemon called `init` which functions in a way that is inherited from the old SystemV.

This daemon functions on the principle of runlevels, which represent the state of the computer.

0. System is halted
1. single user mode
2. multiuser mode
3. full multiuser mode
4. not used
5. grahic interface mode
6. system reboot

You can manually initiate a runlevel transition using telinit command. 0 to shutdown, 6 to reboot.

For each runlevel transition, a set of services are executed:

- When computer is stopped, runlevel is 0
- Turn it on: 0 to default startup runlevel, defined by your own system configuration in `/etc/inittab`. Debian and Ubntu use runlevel 2.

For each runlevel, there is a directory containing scripts to be executed (rcX.d) Service startup scripts will indeed be placed in init.x and links will be created by tools placing them in the proper directories.

An init script, also kown as service startup script or sysv script, is a shell script respecting a certain standard. The script controls a daemon app byh responding to commands such as start, stop and others, which are triggered at two levels

1. When computer starts, if the service is scheduled to be started for the system runlevel, the init daemon will run the script with the statr argument.

```
# OS with service command
service httpd start
```

```
# OS without service command
/etc/init.d/httpd start
```

The script must accept at least the start, stop, restart, force-reload and status comands, as the will be used by the system respectivamente. It is often interesting to provide further options, such as a reload argument to reload the service configuration. Or a try-restart argument to stop and start the service again

NGINX Basics

Ubuntu Server 22.04 Installation

```
apt-get update
apt install -y curl gnupg2 ca-certificates lsb-release \
    debian-archive-keyring
```

```
curl https://nginx.org/keys/nginx_signing.key | gpg --dearmor \
    | tee /usr/share/keyrings/nginx-archive-keyring.gpg >/dev/null
```

```
apt-get update
apt-get install -y nginx
nginx
```

Directories

/etc/nginx

Default configuration root for the NGINX server. You will find configurration files that instruct NGINX on how to behave.

- /etc/nginx/nginx.conf: This file is the default config entry point used by the NGINX service. Sets up global settings for things like worker processes, tuning, logging, loading, etc.
- /etc/nginx/conf.d/: This directory contains default HTTP server configuration file.
- /var/log/nginx/ directory is the default log location for NGINX. access.log and error.log files.

Commands

```
nginx -h
nginx -v
nginx -V
nginx -t #tests configration
nginx -s signal # Sends a signal to the NGINX process such stop, quit,
reload and reopen
```

- You can alter the default config files and test your changes with the nginx -t command.
- If your test is successful, reload the configuration using `nginx -s reload` command

Serving Static Content

Overwrite the default HTTP server configuration locted in /etc/nginx/conf.d/default.conf wit hthe following nginx configuration example:

```
server {
    listen 80 default_server;
    server_name www.example.com;

    location / {
        root /usr/share/nginx/html;
        # alias /usr/share/nginx/hatml;
        index index.html index.htm;
    }
}
```

- This config serves static files over HTTP on port 80 from the directory /usr/share/nginx/html.
- The first line in this config defines a NEW SERVER BLOCK. This define a new context for NGINX to listen for.
    - line two instructs NGINX to listen port 80
    - serveR_name directive defines the hostname
    - The location block defines a configuration based on the path in the URL.

Basic Nginx Config

Nginx config file can be described as a lsit of directives organized in a logcal structure. The entire bheavior of the application is defined vy de values that you give to those directives.

Nginx makes use of one main config file /etc/nginx/nginx.conf

```
$ nginx -v
nginx version: nginx/1.18.0 (Ubuntu)
$ nginx -h
...
-c filename : set configruation file (default: /etc/nginx/nginx.conf)
$ cat /etc/nginx/nginx.conf
```

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {
    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    types_hash_max_size 2048;
    # server_tokens off;

    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;

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

    ##
    # SSL Settings
    ##

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; #Dropping SSLv3, ref:
POODLE
    ssl_prefer_server_ciphers on;
```

```
    ##
    # Logging Settings
    ##

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    ##
    # Gzip Settings
    ##

    gzip on;

    # gzip_vary on;
    # gzip_proxied any;
    # gzip_comp_level 6;
    # gzip_buffers 16 8;
    # gzip_http_version 1.1;
    # gzip_types text/plain text/css /application/json
application/javascript text/xml application/xml application/xml+rss
text/javascript;

    ##
    # Virtual Host configs
    ##

    include /etc/nginx/conf.d/*conf;
    include /etc/nginx/sites-enabled/*;
}

; mail {
;     # See sample Authentication script at
http://wiki.nginx.org/ImapAutenticateWithApachePhpScript
;     # auth_http localhost/auth.php;
;     # pop3_capabilities "TOP" "USER";
;     # imap_capabilities "IMAP4rev1" "UIDPLUS";

;     server {
;         listen      localhost:110;
;         protocol    pop3;
;         proxy       on;
;     }

;     server {
;         listen      localhost:143;
;         protocol    imap;
;         proxy       on;
;     }
}
```

Directive Blocks

Directives are brought in by modules. Modules may also enable directive blocks.

Events

The events block in the default config is brought in by the Events module. The directives that the module enables can only be used within that block.

```
events {
    worker_connections 1024;
}
```

Some directives must be part at the root of the config file, because they have a global effect on the server. Blocks can be nested into each other.

```
http {
    server {
        listen 80;
        server_name example.com;
        access_log /var/log/nginx/example.com/log;
        location ^~ /admin/ {
            index index.php;
        }
    }
}
```

- http block: Variety of config directives as well as one or more **server** blocks.
- server block: allows to configure a virtual **host**. The server block in this example contains some config that applis to all HTTP requets with a OST header exactly matching example.com
- location blocks: You may insert one or more location blocks to request URI of a specified path.

The configuration is **inherited** within children blocks. The access_log directive specifies that all HTTP requests for this server should be logged into a text file. This is still true within the location child block, although you have the option of disabling by reusing access_log off; directive

Advanced Language Rules

Directives Accept Specific Syntaxes

- Syntaxes are directive-specific. Location and rewrite directive support complex expressions in order to match particular patterns. Rewrite module allows advanced logical structure, and config files will begin to look like programing scripts.

Diminutives

- k or K: Kilobytes. client_max_body_size 2G;
- m or M: Megabytes. 2048M
- g or G: Gigabytes, 2097152k;

- Times values: ms, s, m, h, d, w, M, y.
    - client_body_timeout 4g;
    - client_Body_timeout '3m 15s'; Its possible combine values with diffrerent units using enclosing in quotes.

Variables

Modules provide variables that can be used in the definition of directive values. For example $nginx_version variable.

```
location ^~ /admin/ {
    access_log logs/main.log;
    log_format main '$pid - $nginx_version - $remote_addr';
}
```

Note that some directives do not allow you to use variables, as error_log.

Strings

- Without Quotes: root /home/example.com/www;
- With Quotes: (For special charts as (space), (""), ({}), (😉). root '/home/example.com/my web pages'; "root /home/example.com/my web pages".

Base Module Directives

What Are Base Modules

Base modules offer directives that allow you to define the parameters of the basic functionality of Nginx.

- They cannot be disabled at compile time -> Directives and blocks that they offer are always available.
- Three base modules have been distinguished
    - Core Module: essential features and directives (process managmnt and security)
    - Events module: Lets configure inner mechanism of the networking
    - Config Module: Enables inclusion mechanism

Process Architecture: How Nginx Daemon Works

- The **Master Process** exists in memory since Nginx starts. It is launched with current user and group permissions.
    - Not process any client request
    - Spawns processes that do it
- The **Worker Processes** are spawned by the Master Process
    - Customizables by user and group

From config file you can define the number of worker processes, maximun connections per worker process,user and group, etc.

Core Module Directives

List of directives available by core module/ Must be places at the root of the config file, and can only be used once.

- directive (context) [**default_accepted_value** , accepted_value2]
- daemon [**on** / off]: Ena/Disa daemon mode. If disabled, program will not be started in background, it will stay in foreground when launched from shell. This may come in handy for debugging, when you need to know what causes Nginx to crash when
- debug_points [stop / abort]: Activates debug points. Use stop to interrupt the app when a debug point comes about in order to attach a debugger. USe abort to abort and create core dump file.
- env [MY_VARIABLE=value;]: define or redefine env variables
- error_log (main, http, server, location) [/file/path level;, **logs/error.log error**] Where level is debug, info, notice, warn, error, crit, alert, emerg. Enables error logging at different levels. You can disable error logging.
- lock_file [path]: lock file for mutual exclusion. Disabled by default. Locks are implemented using atomic operations
- log_not_found [**on** / off]: Enables or disables logging of 404 not found HTTP errors.
- master_process [**on** / off]: if ENA, nginx starts multiple processes: a main and workers. If DISA nginx woks with an unique process. Should be used for testing only.
- pcre_jit [**on** / off]: ENA or DISA Just-In-Time comp for regular expressions.
- pid logs/nginx.pid: Path of the pid file for nginx daemon. Default can be config at compile time. Make sure to enable this directive since the pid file may be used by the nginx init script
- ssl_engine enginename: default none. enginename is the name of an available ssl acelerator on your sys.
- thread_pool name threads=number [max_queue=number]: default: 'thread_pool default threads=32 max_queue=65536' Defines a thread pool reference that can be used with the aio directive for serving larger files asynchronously. For load balancing optimization
- timer_resolution 100ms: Controls interval between system calls to gettimeofday(). If is not specified, clock is refreshed after each kernel event notification
- user username groupname; user username; Default defined at compile time. Allows to define user account and user group optionally for starting nginx worker processes. For security **you should make sure to specifiy a user and group with limited privileges.**
- worker_cpu_Affinity: Lets you affect the worker processes to CPU cores.
- worker_priority [**0**]; From -20 (highest) to 19 (lowest), default 0. Kernel run at -5, it is not recommended set the priority 05 or less
- worker_processes: Number of worker processes. Default is 1. It is recommended increase this value if your cpu has more than one core. Alternatively you may use auto value (by default is the amount of CPU cores detected on the system.)
- worker_rlimit_core: Size of core files per worker process
- worker_rlimit_nofile number; default none; Numbe of files that a worker process may use simultaneously
- working_directory: working_directory /usr/local/nginx/: Working directory used for worker processes. The user must have write permissions on this folder to be able to write core files
- worker_aio_requests 10000; Max number of outstanding async IO ops for a single worker process

Events Module

Allows to configure network mecanisms. Some have important impact on apps performance.

- All directives must be placed in the **events** block, at the root of config file

```
user nginx nginx;
master_process on;
worker_processes 4;
events {
    worker_connections 1025;
    use epoll;
}
```

- **accept_mutex** [**on** / off]: Ena or Disa use of an accept mutex to open the listening sockets
- **accept_mutex_delay** [**500ms**]: Defines time that a worker process should wait before trying to acquire the resource again.
- **debug_connection** 172.63.155.31: Writes detailed logs for clients matching this IP address or address block. Specified with error_log directive. Nginx must be compiled with the --debug switch
- **multi_accept** [on / **off**]: If nginx should accept all the incoming connections at once from the listening queue
- **use** [/dev/poll , epoll , eventport , kqueue , rtsig , select] Selects the event model among the available ones. Nginx selects the most appropriate one. You should not have to modify this value
- **worker_connections** 1024: default none, defines number of connections that a worker process may treat simultaneously

Configuration Module

Enable file inclusions with **include** directive. Can be inserted anywhere in the config file, and accepts a single parameter: a file path.

```
include /file/path.conf;
include sites/*.conf;
```

The file path is relative to the config directory.

Necessary Adjustments

- user root root;
  - Dangerous from the security point of view -> CREATE a new user account on your system and make use of it there.
  - Recommended value: user www-data www-data
- worker_processes 1;
  - You should have at least one process per CPU core
  - Recommended value: worker_processes auto;
- worker_priority 0;

    - If your system performs other taks simultaneously, you might want to grant a higher priority to Nginx processes.
    - Recommended value: depends. You should not set it under -5.
- log_not_found on;
    - Specifies whether Nginx should log 404 errors or not. Set this to off if you want t oensure that your log files don;t get cluttered by Error 404 entries.
- worker_connections 1024;
    - Define total number of connections accepted by the server simultaneously. If your server is a huge monster meant to host high traffic sites, you will want to increase this value.

Testing

Create Test Server

A test page comes with default package in the html folder (/usr/local/nginx/html/index.html) and the original nginx.conf is configured to serve this page.

```
http {
    include         mime.types;
    default_type    application/octet-stream;
    sendfile        on;
    keepalive_timeout   65;
    server {
        listen          80;
        server_name     localhost;
        location / {
            root        html;
            index       index.html index.htm;
        }
        error_page      500 502 503 504 /50x.html
        location = /50x.html {
            root        html;
        }
    }
}
```

- Opening a listening socket on port 80
- Accesible at http://localhost/
- Index page index.html

Performance Tests

Methodoloy

1. Run the tests
2. Edit configuration
3. Reload server

4. Run tests again...

Note: You should avoid running testing tool on the same computer that is used to run Nginx.

- httperf: open source itility developed by HP for Linux
- Autobench: wrapper for httperf, improving testing mechanisms
- OpenWebLoad: Smaller scale open source load testing app

HTTPerf

Simple command-line tool that can be downloaded from hpl.hp.com/research/linux/httperf/.

Autobench

Perl script makes useof httperf more efficiently. Runs continuous tests and increases request rates until your server gets saturated. Generate graphs.

Open Web Load

Free open source app

HTTP: Hypertext Transfer Protocol

- Bsae de cualquier intercambio de datos en la Web
- Protocolo estructura cliente-servidor
- Por ejemplo una pagina web completa resulta la union de diferentes documentos recibidos, por ejemplo un css, un js, imagenes y el html
- Funciona en capa de aplicacion, se transmite sobre protocolo TCP o TLS (encriptado) de capa 4.

Se utiliza tanto para las webs como para imagenes, videos, datos de formularios, etc.

Arquitectura

Por lo general, un navegador web (agente) realiza una peticion a un servidor web. Este ultimo gestiona y responde.

1. El navegador solicita un documento HTML al servidor
2. Procesa el documento recibido y envia mas peticiones para scripts, estilos, vidoes, imagenes, etc.
3. Une todos los datos y compone la pagina final.

En cuanto al servidor, pueden ser multiples servidores en un unico computador o un servidor puede estar repartido en multiples ordenadores. (estandar http/1.1)

Proxies

En la estructura de la web, existen dispositivos entre el cliente y el servidor que gestionan los mensajes.

- La mayoria de estos dispositivos gestionan niveles de protocolo infeiores tales como transporte, red o fisica.
- Los dispositivos que operan en capa de aplicacion (mayor procesamiento) por ende gestionan HTTP, son conocidos como proxies. Entre sus funciones destacamos
    - caching (publica o privada)
    - filtrado (anti-virus, control parental)
    - balanceo de carga
    - autentificacion
    - registro de eventos

Caracteristicas de HTTP

- Pensado para ser leido e interpretable por personas
- Extensible
- Utiliza sesiones, pero no estados.
    - No guarda ningun dato entre dos peticiones en la misma sesion.
    - Se permite guardar datos con respecto a la sesion de comunicacion, mediante cookies
    - Requiere trabajar sobre un protocolo fiable de comunicacion, por eso utiliza TCP en lugar de UDP.

Flujo HTTP

1. Cliente abre conexion TCP mediante una o varias peticiones al servidor

2. Se realiza una peticion HTTP
3. Se lee la respuesta del servidor, la cual consiste en informacion necesaria para la conexion seguida del contenido, por ejemplo el archivo html.

Mensajes HTTP

En HTTP/1.1 los mensajes era en formato texto y legibles por las personas. Sin embargo, en HTTP/2 los mensajes son en formato binario. Sin embargo la semantica se mantiene y se puede interpretar los mensajes de HTTP/2 en formato HTTP/1.1

Peticion HTTP

- Metodo: Define la operacion que el cliente quiere realizar
- Direccion del recurso
- Version del protocolo HTTP
- Cabeceras HTTP opcionales
- Cuerpo de mensaje (de ser necesario en el metodo pedido)

Respuestas

- Version del protocolo HTTP
- Codigo de estado
- Mensaje de estado
- Cabeceras HTTP
- Opcionalmente el recurso pedido

Recursos Web

El objetivo de una peticion HTTP es llamada recurso. Puede ser una foto, un documento, etc. Cada recurso se identifica mediante un URI (Uniform Resource Identifier)

URLs

- Forma mas comun de URI es la Uniform Resource Locator (URL) tambien conocida como direccion web.
- gianfrancolasala.com/blog

URNs

Uniform Resource Name (URN) identifica un recurso por un nombre en particular

Sintaxis

- http://www.gianfrancolasala.com:80/path/to/some/file.html?key1=value1&key2=value2#somehwere
- http:// -> Protocolo que el buscador debe utilizar
    - data, file, ftp, http/https, javascript, mailto, ssh, tel, urn, etc.
- www.example.com: Nombre de dominio o autoridad a la cual pertenece el nombre. Es posible acceder directamente la direccion IP.
- :80: puerto. Si el servidor web utiliza el puerto estandara (80 para HTTP y 443 para HTTPS) no es requerido.

- /path/to/some/image.html -> path/direccion del recurso dentro del servidor web.
- Query -> Parametros entregados al servidor de pares llave/valor.
- #Somewhere -> Fragmento. Representa una marca dentro del recurso En un documento es posible que el navegador scrollee hasta la parte mencionada
-

Tipos MIME

- Media Type (Multipurpose Internet Mail Extensions or MIME Type)
- Forma de indicar el format ode un documento, archivo o conjunto de datos [RFC6838]
- Los navegadores utilizan el MIME type para determinar como se procesa un documento
- NO utilizan la extension del archivo

Sintaxis

```
tipo/subitpo

text/plain
text/html
image/jpeg
image/gif
video/mp4
audio/*
```

Tipos Discretos

- Indican categoria del documento
- test, image, audio, application (datos binarios), video

Tipos multiparte

indican documentos que se encuentran en partes, posiblemente con diferentes tipos de MIME. De esta forma se representan documentos compuestos.

multipart/form-data

El tipo ```multipart/form-data```` se utiliza para enviar contenido de un formulario HTML completo desde el browser sl sv.

MIME Sniffing

La ausencia de tipo MIME hace que algunos navegadores adivinen el tipo correcto observando el recurso. Esta practica afecta la seguridad ya que algunos tipos MIME representan contenieo ejecutable

3 HTTP Configuration

- Websites served are also referred as virtual hosts
- http/server/location
- HTTP directives and variables

HTTP Core Module

- Directives
- Components
- Variables

Structure Blocks

HTTP Introduces three logical blocks

- http:
  - At the root of config file
  - Define the directives and blocks for all the modules related to HTTP
  - If defined multiple times (no real purpose) the last block will override the previous ones.
- server:
  - Declare a **website** identified by one or more hostnames
  - Can only be used within http block
- location
  - Define settings to be applied to a particular location on a website
  - Can be used within a server block or another location block

HTTP section encompasses entire web config.

- Contain one or more server blocks
- Defining domains and sub-domains
- define location blocks for each websites to define additional settings to a particular URI.
- A setting at http block level preserves its value in the potentially incorporated server and location blocks

```
http {
    gzip on;
    server:{
        server_name localhost;
        listen 80;
    }
    location /downloads/ {
        gzip off;
    }
}
```

Socket and Host Config

- Directives to configure virtual hosts
- Materializes by `server` blocks
    - hostname or IP address
    - port combination
- TCP socket options

listen

- Context: server
- IP Address
- Port to be used by listening socket
    - 80 HTTP, 443 HTTPS
- `listen [address][:port] [options];`
- Options
    - default_server: default website for any request received at the IP and port
    - ssl: Served over SSL
    - spdy: Support for SPDY protocol if SPDY module is present
    - proxy_protocol: Enables PROXY protocol
    - backlog, rcvbuf, sndbuf, accept_filter, deferred, setfib, bind.

```
listen 192.168.1.1:80;
listen 127.0.0.1;
listen 80 default;
listen [::a8c9:1234]:80;
listen 554 ssl;
```

server_name

- Context: server
- Assigns one or more hostnames to the server block.
- if no server block matches the desired hosts, nginx selects te first server block that matches the parameters on the listen directive.
- Accepts wildcard and regular expressions
-

```
server_name www.website.com;
server_name www.website.com website.com;
server_name *.website.com;
server_name .website.com; # website.com + *.website.com
server_name *.website.*;
server_name ~^(www)\.example\.com$; # $1 = www
```

```
server_name website.com ""; # Catch all the requests
server_name _ ""; # Catch all the request: _ dummy hostname
```

server_name_in_redirect

- Context: http, server, location
- `on or off` default off
- Internal redirects.

server_names_hash_max_size

- Context: http
- default 512
- Hash tables for data collections to spped up processing requests. Max size of the server names hash tables.

server_name_hash_bucket_size

- context: http
- bucket size for server names hash tables. Change this value onl if nginx tells you to do

port_in_redirect

- Defines whether or not nginx should append to port number to the redirection.

tcp_nodelay

- context: http, server, location
- Enables or disables TCP_NODELAY socket option for keep-alive connections only.

tcp_nopush

- http, server, location
- Enables or disables TCP_CORK socket option.

sendfile

- if Ena, nginx uses sendfile kernel call to handle file transmission. If disabled, handles file transfer by itself. Default off.

sendfile_max_chunk

- http, server.
- defines max size of data to be used for each call to sendfile
- default 0.

reset_timedout_connection

- http, server, location
- When connection times out, its associated information may remain in memory depending on its state. Enabling this directive will erase all memory associated with the connection after it times out.
- default off

Paths and Documents

Directives that configure the documents that should be serverd for eachwebsite, as error pages, index page, etc.

root

- context: http, server, location, if. Variables accepted
- Defines document root **containing the files that you wish to serve to your visitors**
- sintax: directory path
- default: html
-

alias

- context: location. Variables+
- assigns different path for nginx to retrieve documents for a specific request.

```
http {
    server {
        server_name localhost;
        root/var/www/website.com/html;
        location /admin/ {
            alias /var/www/locked/;
        }
    }
}
```

1. When a request for http://localhost/ is received, files are served from the /var/www/website.com/html folder.
2. IF receives a http://localhost/admin request, tha path used to retrieve the files is var/www/locked
3. Do not forget trailing /

error_page

- context: http, server, location, if. Variables+
-

- Allows to affect URIs to the HTTP response code

```
error_page 404 /not_foud.html;
error_page 500 501 502 503 504 /server_error.html;
error_page 403 http://website.com/;
error_page 404 @notfound; #jump to a named location block
error_page 404 =200 /index.html; #404 error, redirecto to index with 200 ok
responde code
```

if_modified_since

- http, server, location
- HTTP Modified-Since header used by seach engine spiders, such as Google web crawling bots. The robot indicates date and time of the last pass. If the server has not been modified since that time, server simply returns a 304 nod modified response code wit no body.
- accepts values off, exact, before.
- default: exact

index

- Context: http, server, location. Variables+
- defines default page that nginx will serve if no filename is specified
  - autoindex directive to generate auto index of files.
  - Otherwise, 403 forbidden error
- `index file1 [file2...] [absolude_file]`
- default: index.html
- `index index.php index.html intex.htm`

Client Request

The way that Nginx handles client requests. All in `http, server and location` contexts, otherwise will be specified.

- `keepalive_request` max number of requests over a single keep-alive connection -> default 100
- `keepalive_timeout` num of secs the sv will wait before closing a keep-alive connection. Default 75. Second parameter (opt) is transmitted as the value of the timeout http response header.
- `keepalive_disable` disable keepalive func for browser families.
- `send_timeout` amout of time after Nginx closes an inactive conection. Default 60
- `client_body_in_file_only`
  - off (default): does not store request body in a file
  - clean: removes the file after request is processed
  - on
- `client_body_*` timeout, temp_paht, buffer_size
- `client_header_*` buffer_size, timeout,
- `client_max_body_size` default 1m
- lingering_time: amout of time nginx should wait for after sending 413 error, default 30

- lingering_timeout
- lingering_close
- ignore_invalid_headers
- chunked_transfer_enconding
- max_ranges

MIME Types

types

- http, server, location
- establish correlations between MIME types and file extensions

```
types {
  mimetype1 extension1;
  mimrtype2 extension2 [extension3...];
}
```

Nginx includes a basic set of MIME types as a standalone file to be included wit hthe directive

```
include mime.types;
```

If the extension of the served file is not found within the listed types, the fault type is used, as defined by the default_type directive.

force all the files in a folder to be downloaded insthead of being displayed:

```
http {
  include mime.types;
  [...]
  location /downloads/ {
    # removes all MIME types
    types { }
    default_type application/octet-stream;
  }
}
```

default_type

- http, server, location
- defines default MIME type

Limits and Restrictions

This set allow to add restrictions when a client attempts to access a particular locaitn or document.

limit_except

- location
- Prevent the use of all HTTP methods, except the specifiies.

```
location /admin/ {
  limit_except GET {
    allow 192.126.1.0/24;
    deny all;
  }
}

This example applies a restriction to the /admin/ location. Visitors that
have a local IP address are not affected by the restriction.

```yaml
limit_except METHOD1 [METHOD2...] {
  allow | deny | auth_basic | auth_basic_user_file | proxy_oass | perl;
}
```

limit_rate

- http, server, location, if
- Limit transfer rate of individual client connection expressed in B/s
- default: no limit

limit_rate_after

- Define amount of data transferred before limit_rate directive takes effect.
- Default none

satisfy

- location
- defines whether clients require all access conditions to be valid (satisfy all) or al least one (satisfy any)
- default: all

```
location  /admin/ {
  allow 192.168.1.0/24;
  deny all;
  auth_basic "Authentication Required";
  auth_basic_user_file conf/htpasswd;
}
```

There are two conditions in the preceding example for clients to be able to access the resource
```

1. Through the allow and deny directives we only allow clients that have a local IP address; All other are denied.
2. Through the auth_basic and auth_basic_user_file only allow clients that provide a valid username and password

Internal

- location
- specified that location block is internal, it cannot be accessed by external requests.

```
server {
  ...
  server_name .website.com;
  location /admin/ {
    internal;
  }
}
```

File Processing and Caching

- disable_symlinks: Off by default.
- directio: if ena, files with size greater than the specified value will be read with the Direct IO mechanism.
- directio_alignment
- open_file_cache: allows to enable cache. Not store file contents, only descriptors, existence, errors, etc.
- open_file_cache_errors
- open_file_cache_min_uses
- open_file_cache_valid
- read_ahead: pre read from the files. default 0 (enabled)

Other Directives

- log_not_found: disables the logging 404 not found http errors. Default on.
- log_subrequest
- merge_slashes: default off
- msie_padding: Works with Google Chrome browser families.
- msie)refresh
- resolver: Specifies the name servers that should be employed by Nginx to resolver hostnames to IP addresses and vice versa.
    - [IPv4 or IPv6 addresses] [valid=Time] value, ipv6=on|off
    - default: none
    - resolver 127.0.0.1; #local DNS
    - resolver 8.8.8.8.8.8.4.4 valid=1h; # Google DNS

server_tokens

- http, server, location
- Allow to define whether or not Nginx should inform clients of the running version number.

Module Variables

Only a set of directives accept variables in the definition. If uses variables when directive does not accept them, no errors is reported.

Request Headers

Nginx leets access to client request headers under the form of variables

- $http_host: Host HTTP
- $http_user_agent: Indicating the web browser of the client
- $http_referer: Indicating the URL of the previous page from which the client comes
- $http_via: informs possible proxies used by the client
- $http_x_forwarded_for: shows actual IP address of the cilent if the client is behing a proxy
- $http_cookie: cookie data sent by the client

Response Headers

- $sent_http_content_type indicating MIME type of the resource being transmitted
- $sent_httpcontent_lenghth
- $sent_http_location indicates the location of the desired resource is different from the one specified in the original requests
- $sent_http_last_modified: mod date of the requested resource
- $sent_http_connection: definin connection will be kept alive or closed
- $sent_http_transfer_Enconding
- $sent_http_cache_control

Nginx Generated

- $arg_XXX allows to access the query string (GET parameters), where XXX is the name of the parameter
- $args all the arguments combined together
- $binaru_remote_addr IP address of the client as binary data
- $body_bytes_sent
- bytes_Sent
- connection
- connection_Requests
- content_length
- content_type
- cookie_XXX
- document_root
- document_uri
- host

- $hostname system hostname of the server computer
- $https set on for https connections
- $is_args to construct a URI as 'index.phpis_argsargs. If there are a any query string argument in the request, is_args is set to ?, making a valid URI
- $limit_Rate
- $msec current time in seconds + miliseconds
- $nginx_version
- $pid
- $pipe
- $proxy_protocol_addre
- $remote_Addr return the IP address of the client
- $proxy_protocol_addr
- $remote_port port of the client socket
- $remote_user client username if they use authentication
- $realpath_root
- request_body
- request_body_file
- request_completion
- request_filename
- request_length
- request_method
- request_time
- request_uri
- scheme: returns http or https
- server_addre IP address of the server.
- server_name
- server_port
- server_protocol
- status
- time_local
- uri

Location Block

Levels of configuration

1. Protocol level (http)
2. Server level (server)
3. Requested URI level (location)

Location modifier

Nginx allows to define location blocks specifying a pettern that will be matched against requested URI

```
server {
  server_name website.com;
  location /admin/ {
```

```
      #this config applis to http://website.com/admin/
    }
  }
```

- = -> Must match specified pattern exactly. Cannot use regular expression.
- No modifier -> Must begin with the specified pattern. Not recommended regular expressions
- ~ -> case-sensitive match to the specified regular expression
- ~* -> Must be a case-insensitive
- ^~ modifier -> similar to no-symbol. Location URI must begin with the specified pattern. Nginx stops searching fo the other patterns
- @ -> defines a named location block. Canot be accessed by the client but only by internal requests generated by other directives such as try_files or error_page

Nginx searches for the location block that best matches the requested URL.

Module Configuration

Rewrite Module

Perform URL rewriting. Allows to get rid URLs containing multiple para meters and transform to useful informatives new ones.

The mechanism consist of rewriting the URI of the client request after it is received and before serving the file. The rewritted URI is matched against the location blocks in order to find the configuration that should be applied to the request.

Rewriting is performed by `rewrite` directive using regular epressions.

Metacharacters

- ^ Beginning: Entity after must be found at the beginning
  - ^h matches hello, hh, h (anything beginning with h)
- $ End: e$
  - matching sample, e, file (anything ending with e)
- . (dot) Matches any character:
  - hell. matches hello, hellx, hell!. Not matches hell
- [ ] Set: Matches any character withing the specified set
  - [a - z] range
  - [abcd] set
  - [a-z0-9]
  - example hell[a-y123-]
- ^ Negate Set: Matches any character that is not withing the specified set
- | Alternation: Matches the entity placed either before or after |
  - hello|welcome matches hello, welcome, helloes, awelcome
- ( ) Grouping
- \ Escape: allows to scape special characters. Hello. matches Hello., Hello. How are You?, Hi! Hello...
  - not matching Hello,Hello!
-
  - 0 or More Times: The entity preceding * must be found 0 or more times
  - he*llo matches hllo,hello,heeeeeeeello
-
  - 1 or more times
- ? 0 or 1
- {x} x times
- {x,y} x to y times
- {x,} at least x times

Internal Requests

- External requests directly originate from the client. The URI is the nmatched agains possible location blocks
- Inetrnal requests are triggered by Nginx via specific directives.
  - error_page, index, rewrite, try_files, add_before_body, add_after_body directives creates internal requests

Internal Requests Types

- Internal redirects: Nginx redirects the client requests internally, the URI is changed.
- Sub-requests: Additional requests to generat content complementary to the main request.

error_page

error_page directive allow to define sv behavior when a specific error code occurs.

```
server {
    server_name website.com;
    error_page 403 /errors/forbidden.html;
    error_page 404 /errors/not_found.html;
}
```

Conditional Structure

```
server {
    if ($requests_method = POST) {
        ...
    }
}
```

Operators

- None: if ($string): is true if specified variable or data is not equial to an empty string or a string starting with 0 character.
- =, !=
- ~ case sensitive
- ~* case insensitive
- !~, !* negate matching
- -f !-f
    - if (-f $request_filename) tests the existence of a specified file
- -d, !-d similar -f but directory
- -e !-e, similar -f but file, directory or symbolic link
- -x !-x executable

Directives

rewrite

- server, location, if
- allow to rewrite the URI of the curent request
- `rewrite regexp replacement [flag]`
- regexp: the regular expression that the URI should match in order for the replacement to apply
- Flag:

- last
- break
- redirect

break

prevent rewrite directives

return

interrupts the processing of the request and returns the specified htt[status code or specified test

```
return code | text
```

set

initializes or redefines a variable. Note that some variables are reasd only.

```
set variable value
```

when a variable that has not yet been initalized nginx will issue log messages if declared `unitialized_variable_warn on;` before.

rewrite_log

if set to on, nginx will issue log messags for every operation performed by the rewrite engine at the notice error level.

<<<<<<< HEAD

Common rewrite rules

Basic rewrite rules to stisfy basics of dynamic websites that wish to beautify page links.
http://website.com/ was omitted at the beggining of URIs.

Search

- input: /search/some-search-keywords
- rewritten: /search.php?q=some-search-keywords
- `rewrite ^/search/(.*)$ /search.php?q=$1?;`

Multiple Parameters

- Input: /index.php/param1/param2/param3
- Rewritten: /index.php?p1=param1&p2=param2&p3=param3
-

Wiki-like

- /wiki/some_keyword
- /wiki/index.php?title=some_keyword
- rewrite ^/wiki/(*.)$ /wiki/index.php?title=$1?;

News Website Article

- /33526/us-economy
- /article.php?id=33526
- rewrite ^/([0-9]+)/.*$ /article.php?id=$1;

SSI Module

SSI (Server Side Includes) is a sort of server-side programming language interpreted by Nginx. Created in order to render web pages dynamically, from static .html files with client-side scripts to complex pages with server-processes indstrucitons.

Module Directives

- ssi: Enables parsing files for SSI commands. Nginx only parses the files corresonding to the MIME types selected with the ssi_types directive. Syntax on/off, default off; `ssi on;`
- ssi_types: Define MIME types that should be eligible for SSI parsing. `ssi_types type1 [type2] [type3]` default text/html;
- ssi_client_errors: Enabling this option to silence Nginx error ssi messages.
- ssi_value_length: Maximun length accepted by Nginx in characters
- ssi_ignore_recycled_buffers
- ssi_min_file_chunk
- ssi_last_modified

=======

Common rewrite rules

SSI commands

The principle is simple. Disgn regular HTML pages and insert inside SSI commands that looks like regular HTML comments

```
<!--# command param1="value" param2="value2 -->
<!--# include file="header.html" -->
```

This command generates an HTTP sub-request to be processed by Nginx. The body of the response is inserted instead the command itself.

```
<!--# include virtual="/sources/header.php?id=123" -->
```

This sends a sub-request to the server. The directive wait="yes" is using to specify that Nginx should wait for the completion of the request before moving on to other includes.

If the result of the include is empty Nginx inserts 404 or 500 error page.

Variables

Nginx SSI module offers the option of working with variables. Displaying a variable can be done with echo command

```
<!-- var="variable_name" -->
```

- var: Name of the variable to display.
- default: A string to be displayed in case the var is empty
- encoding: encoding method for the string

the command set, insthead of echo, write the variable value

```
<!--# echo var="MYVAR"  --> (none)
<!--# set var=MYNAME" value="Gian" -->
<!--# set var="MYVAR" value="hello $MYNAME" -->
<!--# echo var="MYVAR"  --> (hello Gian)
```

```
<!-- if expr="expression1" -->
...
<!-- elif expr="expression2" -->
...
<!-- else -->
...
<!-- endif -->
```

```
<!--# if expr="variable = some_value" -->
<!--# if expr="variable = /pattern/" -->
```

Access and Logging Module

Allows to config the way visitors access your website and the way your server logs requests.

index

- http, server, location

```
index index.php index.html index.htm
```

Log

Controls the behavior of Nginx regarding the access logs. It is a key module for system administrators, allows to analyzing the runtime behavior of web apps.

access_log

- http, server, location, if, limit_except
- defines
  - acess log file path
  - format of entries in the access log
- `acess_log path [format [buffer=size] | off`
- off to disable access loging at the current level
- format argument corresponds to a tepmlate declared with log_format directive

log_format

- http, server, location
- template to be utilized by access_log directive
- `log_format template_name format_string;`
- default template is called **combined**
- default: `log_format combined '$remote_addr - $remote_user [$time_local] '"$request: $status $body_bytes_sent '"$http_referer" "$http_user_agent"';`
- `log_format simple '$remote_addr $request`

  open_log_file_cache

  - http, server, location
  - configure cache for log file descriptors

  Log Module Variables

  - connection
  - pipe
  - time_local
  - msec
  - request_time
  - status
  - bytes_sent

- body_bytes_sent
- apache_bytes_sent
- request_length

Limits and Restrictions

Regulates access to the coduments of your websites. Restricts the access and require users to:

- authenticate
- match a set of rules

Auth_basic module

enables basic authentication funcitonality. Username and password

```
location /admin/ {
  auth_basic "Admin control panel"; #Variables supported
  auth_basic_user_file access/password_file
}

auth_basic can be set to either off or a text message, referred to as
authentication challenge or realm. Is displayed by browsers in a
username/password box when client attemps to access.

auth_basic_user_file defines the path of the password file relative to the
directory of the configuration file.

```txt
password file syntax"
username:[{SCHEME}]password[:comment].

username: plain text user name
SCHEME: optionally. PAssword hashing method
  - PLAIN
  - SHA (SHA-1 Hashing)
  - SSHA SaltedSHA-1 hashing
password: password
comment: plaint text comment for own use
```

Access Module

- allow and deny directives
- let allow or deny access to a resource for a specific IP address or IP address range.
- `allow/deny IP | CIDR | unix: | all`
- unix represent domain sockets

```
location {
  allow 127.0.0.1;
```
```

```
  allow unix;
  deny all; #deny all other ip addresses
}
```

Limit Connections

Allows to define maximun number of simultaneous connections to the server for a specific zone.

- limit_conn_zone $variable zone=name:size;
- $variable is the mechanism to differentiate one client from another. Typically $binary_remote_Addr
- name is an aribtrary name given to the zone
- size: max size to the table storing session states

```
# Examples
limit_conn_zone $binary_remote_addr zone=myzone:10m;
limit_conn zone_name connection_limit;
```

```
location /downloads/ {
  limit_conn myzone 1;
}
```

Limit request

allows to limit the number of requests for a defined zone.

```
limit_req_zone $variable zone=name:max_memory_size rate=rate;
```

the parameters are identical to limit connection except for the traillling rate, expressd in requests per second or per minute r/s, r/60s.

```
limit_req zone=name burst=burst [nodelay];
```

burst parameters defines maximun possible brsts of requests.

Auth Request

auth_requests allows to deny acccess to a resource based on the result of a sub-request. Nginx calls the URI to specify via auth_request directive. If sub request returns a 2xx response code access is allowed.

```
location /downloads/ {
  # if the script below returns a 200 status code
  # the download is authorized
```

```
    auth_request /autorization.php
  }
```

the auth_request_set allows to set a variable after the sub-request is executed.

Content and encoding

Provides functionalities having an effect of the contents served to the client.

Empty GIF

Serves a 1x1 transparent GIF image from the memory.

```
location = /empty.gif {
  empty_gif;
}
```

MP4

Enable useful funtionality when serving a MP4 file. It parses a special argument `start` which indicates the offset of the section that the client wishes to download or pseudo-stream. The uri to access the file is video.mp4?start=XXX.

To utilize this feautre insert the `mp4 directive` in the location of your choice.

```
location ~* /.mp4 {
  mp4;
}
```

Addition

Allows to add content before or after the body of the HTTP response.

- add_before_body file_uri;
- add_adter_body file_uri;

Substitution

Allows to search and replace text directly from the response body. `sub_filter searched_Text replacement_text;`.

Gzip filter

Allows to compress the response body with gzip before sending to the client.

```

```

```
gzip_buffers amount size;
gzip_comp_level 1;
gzip_disable;
gzip_http_version 1.1;
gzip_min_length 0;
gzip_proxied off;
gzip_types mime_type1 [mime_type2..];
gzip_vary off;
gzip_window MAX_WBITS;
gzip_hash MAX_MEM_LEVEL;
postpone_gzipping 0; # minimum data threshold to be reached before comp
gzip_no_buffer off;
```

Image filter

provides image processing functionalities through the GD Graphics Library (gdlib). Works on location block that filter image files only, such as

```
location ~* \.(png|jpg|gif)$ {...}.
```

- image_filter
  - test, size, resize width height, crop width height, rotate.
  - image_filter resize 200 100;
- image_filter_buffed
- image_filter_jpeg_quality
- image_filter_transaprency
- image_filter_sharpen
- image_filter_interlace

About visitors

Provides extra functionality that heps you find oyt more information about the visitors by parsing client requets headers for browser name and version.

Browser

The browser module parses user-agent HTTP header of the client request in order to establish values for the variables that can be emplyed later in the config

- $modern_browser: if client browser is identified as being a modern web browser/
- $ancient_browser
- $msie: is set to 1 if client is using Internet Explorer

```
modern_browser opera 10.0;
```

Map

Map allows to create values depending on a variable

```
map $uri $variable {
  /page.html 0;
  /contact.html 1;
  /index.html 2;
  default 0;
}
rewrite ^ /index.php?page=$variable
```

can only be inserted within the `http` block. The last instruction rewrites the URL accordnly.

Geo

Provide functionality that affects a variable based on the client data. The syntax is slightly different in that you are allowed to specify IPv4 and IPv6 address ranges.

```
geo $variable {
  default unknown;
  127.0.0.1 local;
  123.12.3.0/24 uk;
  92.43.0.0/16 fr;
}
```

directives

- delete
- default
- include
- proxy: defines a subnet of trusted addresses
- proxy_recursive
- ranges `127.0.0.1-127.0.0.255 LOCAL`

GeoIP

Similar to Geo. Provides accurate graphic similarities with the previous one. uses MaxMind GeoIP binary databases.DOwnload the databases files from the MaxMind website and place them in Nginx directory.

```
geoip_country country.dat; # country information db
geoip_city city.dat; # city informatino db
geoip_org geoiporg.dat; # ISP/organization db
```

- $geoip_country_code (two letters c code)
- $geoip_country_name

- $geoip_region, $geoip_city, $geoip_postal_code, $geoip_city_continet_code, $geoip_latitude, $geoip_region_name, $geoip_org.

USerID filter

Assigns an identifier to the clients by issuing cookies. The identifier cab ne accessed from thevariables $uid_got and $uid_set

- userid: enables or disables issuing and logging of cookies.
  - default off.
  - on, v1, log, off
- userid_service: definfes the IP address of the server issuing the cookie
  - userid_service ip;
  - default: IP of the server
- userid_name: name assigned to the cookie
  - userid_name name;
  - default: user identifier
- userid_domain: domain assigned to the cookie
  - default: none
  - `userid_domain domain;`
- userid_path
  - default /
- userid_expires
  - default: no expiration date
  - `userid_expires date | max;`
- userid_p3p

Split Clients

resource-efficient way to split the visitor base into subgroups based on the percentages that specified. Nginx hashes a value privded (visitor IP, address, cookie data, query arguments, etc) and decides a group.

```
split_clients "$remote_addr" $variable {
  50% "group1";
  50% "group2";
}
location ~ /.php$ {
  set $args "${query_string}&group=${variable}";
}
```

SSL and Security

Secure HTTP functionalities through the SSL module, but also offers an extra module called Secure Link to protect the website and visitors.

SSL

Works on http and server contexts.

ssl

- Ena HTTPS for the specified server
- Equivalent to listen 443 or listen port ssl
- default `ssl off`

other directives

- `ssl_certificate file_path #PEM certificate`
- `ssl_certificate_key file_path`
- `ssl_client_certificate file_path`
- `ssl_crl` orders nginx to load Certificate Revocation List file
- `ssl_dhparam file_path`
- `ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2];`
- `ssl_ciphers` default ALL
- ssl_prefer_Server_ciphers
- ssl_verify_client
- ssl_verify_depth
- ssl_session_cache
- ssl_session_timeout
- ssl_password_phrase
- ssl_buffer_size
- ssl_session_tickets
- ssl_session_ticket_key
- ssl_trusted_certificate

Variables

- $ssl_cipher
- $ssl_client_serial
- $ssl_client_s_dn
- $ssl_protocol
- $ssl_client_cert
- $ssl_client_verify: set to success if client cert was successfully verified
- $ssl_session_id

Setting up SSL certificate

Ensure to already have the following elements

- A .key file generated with the following command: `openssl genrsa -out secure.website.comkey 1024`
- a .csr file generated with the following commmand ```openssl req -new -key secure.website.com.key -out secure.website.com.csr
- Website certificate file as issued by Certificate Authority as secure.website.com.crt.
- CA certificate (for example gd_bundle.crt purchased from goDaddy)

1. Merge website certificate and CA certificate

```
cat seucre.website.com.crt gd_bundle.crt > combined.crt
```

```
server {
  listen 443;
  server_name secure.website.com;
  ssl_certificate /path/to/combined.crt;
  ssl_certificate_key /path/to/secure.website.com.key;
}
```

SSL Stapling

SSL Stapling, also called OCSP (Online certificate Status Protocol Stapling), is a technique that allows clients to easily connect and resume sessions to an SSL/TLS server without having to contact the Certificate Authority, thus reducing the SSL negotiation time.

Enabling SSL stapling should thus speed up the communication between server and visitors.

```
ssl_stapling on;
ssl_stapling_verify on;
ssl_trusted_certificate filename; (extension sohuld be .pem)
```

```
#optional directives
ssl_stapling_file
ssl_stapling_responder
```

PHP and Python with Nginx

FastCGI

CGI Mechanism

The original purpose of a web server was merely to resond to requets from clients by serving te files located on a storage device.

1. Client sends request: `GET /index.html HTTP/1.1` to the web server
2. Web Server process request and sends resopnse `HTTP/1.0 OK or 404 file not found`

Static websites are being progressively abandoned atthe exprense of dynamic ones that contains scripts which are processed by applications such as PHP and Python among others.

1. Client entity (typically web browser) sends request: `GET /index.php HTTP/1.1`
2. Web server pre-processes the reequest (URL rewriting, internal redirects)
3. Web server (Nginx, Apache) forwards request using CGI
4. Backend application (Application, Python, NodeJS) processes request
5. Backed application returns response using CGI
6. Web server post-processes response (gzip compression, character encoding, etc)
7. Web server returns response `HTTP/1.0 200 OK`

When client attmps to visit a dynamic page, web server forwards to the application, processes the script and returns the response to the web server.

To communicate with that ap, CGI protocol was invented in the early 1990s.

Common Gateway Interface

- CGI Protocol v1.1 [RFC3875]
- Allows an HTTP server and a CGI script to share responsibility for resonding a cilent requests
- Server: responsible for managing conection, data transfer, transport, network issues
- CGI script: handles the app issues, data access and document processing
- CGI protocol describes the data exchange between web server and gateway application.

Fast Common Gateway Interface

CGI Protocol is inneficient for servers that are subjects to heavy loads.

- PErsistent processes that handle multiple requests
- server and gateway coomunicates using sockets as TCP or POSIX IPC -> Web Server and application can located in different computers
- It can be implemented on any platform with any programming language.
- Not complex to implement.

Once you have the web server and the backend app running, the only difficulty was remains is to establish the conection between.

1. configure nginx perspective to communicate with the FastCGI application via FastCGI module.

Nginx CGI-derived Module Implementations

- uWSGI module allows Nginx to communicate with applciations through the uwsgi protocol.
- SCGI, variant of CGI like FastCGI. SCGI interfaces and modules can be found in a variety of software projects such as Apache, IISm Java, Cherokee, etc.

FastCGI module	uWSGFI equivalent	SCGI equivalent
`fastcgi_pass`	`uwsgi_pass`	`scgi_pass`
`fastcgi_cache`	uswsgi_cache	scgi_cachje
fastcgi_temp_path	uwsgi_temp_path	scgi_temp_path

FastCGI Main Directives

fastcgi_pass

- location, if
- for TCP: `fastcgi_pass hostname:port`
- for Unix: `fastcgi_pass unix:/path/to/fastcgi.socket;`
- to refer upstream blocks: `fast_pass myblock`

```
# examples
fastcgi_pass localhost:9000;
fastcgi_pass 127.0.0.1:9000;
fastcgi_pass unix:/tmp/fastcgi.socket;
# Using an upstream block
upstream fastcgi {
    server 127.0.0.1:9000;
    server 127.0.0.1:9001;
}
location ~* \.php$ {
    fastcgi_pass fastcgi;
}
```

fastcgi_param

Allow to config the request passed to FastCGI

- http, server, location
- SCRIPT_FILENAME and QUERY_STRING are required
- for POST requests REQUEST_METHOD, CONTENT_TYPE, CONTENT_LENGTH are required.

```
fastcgi_param SCRIPT_FILENAME /home/website.com/www$fastcgi_script_name;
fastcgi_param QUERY_STRING $query_string;
```

```
fastcgi_param REQUEST_METHOD $request_method;
fastcgi_param CONTENT_TYPE $content_type;
fastcgi_param CONTENT_LENGTH $content_length;
```

the fastcgi_params file in Nginx config folder already includes all necessarsy parameter definitios you need.

If the parameter begins with HTTP_ it will override the potentially existing HTTP headers of the client request

fastcgi_bind

- http, server, location
- blinkds the socket to a local ip address, allowing to specify the network interface to use for FastCGI communications
-

fastcgi_pass_header

- specify additional headers that should be passed to the FastCGI server

fastcgi_hide_header

headers that should be hidden from the FastCGI server)Nginx does not forward)

fastcgi_index

FastCGI server does not support automatic directory indexes. IF the requestd URI ends with a /, nginx appends the value fastcgi_index

- fastcgi_index filename;
- fastcgi_index index.php;

Others fastcgi_* directives

- ignore_client_abort
- intercepts_errors
- read_timeout
- connect_timeout
- send_timeout
- split_path_info
- store
- store_access
- temp_path
- max_temp_file_size

- temp_file_write_size
- send_lowat
- pass_request_body
- ignore_headers
- next_upstream
- next_upstream_timeout
- next_upstream_tries
- catch_stderr
- keep_conn
- force_ranges
- limit_rate

Others caching and buffering

- cache
- cache_key
- cache_methos
- cache_min_uses
- cache_path
- cache_use_stale
- cache_valid
- no_cache
- cache_bypass
- cache_revalidate
- bvuffering
- buffers

Config example

```
fast_cacye phpcache;
fastcgi_cache_key "$scheme$host$request_uri"; # request_uri includes the
request arguments such as /page.php?arg=value
fastcgi_cache_min_uses 2; # after 2 hits, a request receives a cached
response
fastcgi_cache_path /tmp/cache levels=1:2 keys_zone=phpcache:10m
inactive=30m max_size=500M
fastcgi_cache_use_stale updating timeout;
fastcgi_cache_valid 404 1m;
fascgi_cache_valid 500 502 504 5m;
```

These directives are valid for pretty much any virtual host configuration. May to save thewse in a separate file `fastcgi_cache` that you can include at the appropiate place

```
server {
    server_name website.com;
    location ~* /.php$ {
        fastcgi_pass 127.0.0.1:9000;
```

```
        fastcgi_param SCRIPT_FILENAME
/home/website.com/www$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_script_name;
        include fastcgi_params;
        include fastcgi_cache;
    }
}
```

PHP With Nginx

PHP 5+ came with FastCGI process manager. PHP should be confiured with --enable-fpm argument.

Architecture

PHP supports the FastCGI protocol. The PHP binary processes scvripts and is stable to interact with nginx via sockets. The FastCGI Process Manager is kown as PHP-FPM

PHP-FPM

- Script tha tmanages the PHP processes.
- It awaits and receives instructions from Nginx
- Automatically daemonizing PHP
- Execute scripts in chrooted environment
- Improved logging
- IP address restrictions

Nginx Configuration

```
server {
    server_name .website.com;
    listen 80;
    root /home/website/www;
    index index.php;
    location ~* /.php$ { # for requests ending with .php
        # Specify the listening address and port configured in php
        fastcgi_pass 127.0.0.1:9000;
        # the document path to be passesd to PHP-FPM
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        # the script filename to be passed to PHP-FPM
        fastcgi_param PATH_INFO $fastcgi_script_name;
        # include other FastCGI realted configuration settings
        include fastcgi_params;
    }
}
```

```
/usr/local/nginx/sbin/nginx -s reload
or
service nginx reload
```

```
echo "<?php phpinfo(); ?>" > /home/website/www/index.php
```

load http://localhost/ or website URL.

Chapter 6: Apache and Nginx Together

Nginx as Reverse Proxy

Cases to use Nginx as reverse proxy

- Already installed with complex configuration files that can hardly be ported to Nginx
- When sistem opertas a frontend system management such as cPanel.
- When functionality that the project requires is available with apache but not with nginx

In any other case a complete switch to Nginx would be a better choice.

Reverse Proxy Mechanism

- Nginx as a frontend sever, direct communication with outside world
- Apache running as a backend server

Nginx

- Positioned as a frontend web server (reverse proxy)
- Receives all the requests comming from the outside net

Apache

- Runs as backend server and
- The listening port must be edited to leave port 80 available to Nginx.
- Alternatively, can employ multiple backend servers on different machines and share the load

To communicate and interact each other -> FastCGI. Nginx acts as a proxy server. Receives HTTP requests from the client and forwards them to the backend server. The mechanism is handled by the proxy module of Nginx.

Load Balancing and Optimization

The concept of load balancing has the potential to solve problems pertaining to scalability, availability and performance.

Concept of Load Balancing

The load balancing concept consists of distributing the worload (CPU, HD, etc) across several servers, completely transparent to the visitors.

There are several techniques available for load balancing. The simplet is ?DNS load balancing. To achieve DNS load balancing simply associate multiple IP addresses with a domain. The OS of the visitors will select one of the IP addresses by a round-robin algorithm. This solution cannot always be applied to high traffic websites for many reasons.

Session Affinity

Session affinity is an expression that designates the persistent assgnment of a client to a particular server in a load-balanced infraestrctura. The word session describe a set of requests performed by a client on a server. When a visitor browses a website, the often visit more than one page. So the server conserves the data related to the operations performed during the visit, as session, shopping cart, login credentials, etc.

DNS load balancing does not ensure session affinity. Nginx helps to achieve it.

Upstream Module

The implementation of load balancing in Nginx is particularly clever as it allows you to distributo the load at several levels of the infrastructure. It isn't limited to proxying HTTP requests acrsoss backend servers: also offers to distribute requests across FastCGI backends, or queries to memcached servers.

The first step is to declare this group of servers with the help of the upstream block, which must be placed within http blockl. Within the upstream block, declare one or more servers with the server directive:

```
hettp {
    upstream MyUpstream {
        server 10.0.0.201;
        server 10.0.0.202;
        server 10.0.0.203;
    }
}
```

Now the server group is declared, you can reference it in your virtual host configuration. For example, distribute the incoming HTTP requests across the server group by simply proying them

```
server {
    server_name example.com;
```

```
    listen 80;
    root /home/example.com/www;
    # proxy all requests to the MyUpstream serve rgroup
    proxy_pass http://MyUpstream;
    ...
}
```

Request Distribution Mechanism

Weight Flag Problem

- Weight flag: Servidor de arquitectura heterogenea.Balancear la carga entre los servidores

```
upstream MyUpstream {
    server 10.0.0.201 weight=3;
    server 10.0.0.202 weight=2;
    server 10.0.0.203;
}
```

by default server have a weight 1, unless otherwise. So every 6 http requests received, Nginx will systematically distribute 3 requests to the .201, 2 to 202 and 1 to 203.

State of Servers

- fail_timeout=N; # N is the number of seconds before a requests id considered to have failed
- max_fails=N; # once by default
- backup mark: Only use it if another server fails it
- dow nmarks: as permanently unavailable.

Achieve Session Afinity

- directives in the upstream block
- up_hash: instructs Nginx to calculate a hash from the first three bytes of the client's IP address. Keep the client assigned to a particular server based on the hash. As long as the client's ipa ddress remains the asme, Nginx will always forward requets to the same server.

```
upstream {
    server .201;
    server .202;
    ip_hash;
}
```

To deal with dynamic IP addresses, instead of client's IP address, separate the requests based on the criteria of your choice.

- hash $cookie_username; # for example

Nginx as TCP Load Balancer

- Distribute load across any form networked servers (database servers, e-mail servers, web server, etc)

Stream Module

The TCP load balancing works similar to HTTP load balancing. The module is not include in default build.

OFfers a new block called stream which must be placed the the root of the configuration file. (outside of the http block).

```
--with-stream
```

directives

- server: declares a TCP server listening on a particular port and optionally a network interface, with or without SSL.
- upstream: defines a server group in a similar manner, as seen previously.

Example of MySQL Load Balancing

Nginx configured to receive MySQL connections and balance them across two backend servers, as follows:

```
stream {
    upstream MyGroup {
        # IP Address Distribution
        hash $remote_addr;
        server 10.0.0.201 weight=2;
        server 10.0.0.202;
        server 10.0.0.203 backup; # back up only
    }
    server {
        # listen on the default MySQL port
        listen 3306;
        proxy_pass MyGroup;
    }
}
```

Additional documentation nginx.org

Chapter 9: Case Studies

Deploying a WordPress site

System requirements

- PHP
- MySQL

PHP Configuration

- set cgi.fix_pathinfo to 0.
- post_max_size: increase if necessary
- upload_max_filesize: upload if necessary
- date.timezone: find the proper value on php.net/manual/en/timezones.php

PHP-FPM side

- php-fpm.conf does not require immediate changes
    - create a pool declaring `[wordpress]`

```
; specify user account and group for the pool
; assume created wordpress user and group
user=wordpress
group=wordpress
; Network interface and listening port
; user 127.0.0.1 if nginx runs on the same machine
listen=127.0.0.1:9000
; only allow connections from local computer
; Change this value if Nginx runs on a different machine
allowed_clients=127.0.0.1
```

Optionally enable chrooting: specify a root directroy for the PHP processes of this pool. For example if you set the chroot to /home/wordpress/www PHP scripts will only be able to read the files and directories within the specified path. Anyt attemp to read or write a file or directroy outside /home/wordpress/www will fail. Highly recommended for security. Attackers must only be able to expoloit files within the reach of PHP process, the rest of the server wohuld not to be compromised.

```
chroot=/home/wordpress/www;
```

My SQL Config

```
mysql -u root -p
mysql> CREATE DATABASE wordpress;
```

```
mysql> GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'localhost'
IDENTIFIED BY 'password';
mysql> exit
# mysql -u wordpress -p
mysql> SHOW DATABASES;
```

Downloading and Extracting WordPress

- wordpress.org/

Ngix Config

HTTP Blocks

Open configuration file nginx.conf and insert update on the following directives

```
# Sets the user and group under which the worker processes will run
# The following values are valid assuming the server will only be hosting
one website
user wordpress wordpress;
pid /var/run/nginx.pid;

events {
    # Edit this value depending on the server hardware
    worker_connections 768;
}

http {
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;

    # Default Nginx values
    keepalive_timeout 65;
    types_hash_max_size 2048;
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    # Set access and error log paths
    access_log /var/log/nginx/acess.log;
    error_log /var/log/nginx/error.log;

    # Enable gzipping of files matching the given mime types
    gzip on;
    gzip_disable "msie6";
    gzip_types text/plain text/css application/json application/x-
javascript text/xml application/xml application/xml+rss text/javascript;

    # Inclunde virtual host configuration files;
    # Edit path accordingly
```

```
    include /etc/nginx/sites-enabled/*;
}
```

Server Block

Create a new file in the directory specified prevously. For example, create a file called wordpress.conf in the /etc/nginx/sites-enabled/ folder. Define virtual host configuration by inserting or updating the following directives

```
server {
    # Listen all network interfaces on port 80
    listen 80;

    # Specify the host name(s) that will match the site
    # The following value allows both www. and no subdomain
    server_name .example.com;

    # Set the path of the WordPress files
    root /home/wordpress/www;

    # Load index.php
    index index.php

    client_body_in_file_only clean;
    client_body_buffer_size 32K;

    client_max_body_size 300M;

    send_timeout 10s;

    # applies to static files:
    location ~* ^.+/.(jpg|jpeg|png|gif|ico|css|js)$ {
        access_log off; # disable logging
        # allow client browsers to cache files for long period
        expires 180d;
    }

    # Applies to every request
    location / {
        # Try servingt the requestd URI:
        # If the file does not exist, append /
        # If directory does not exist, redirect
            # to /index.php forwarding the request URI
        try_files $uri $uri/ /index.php?q=$uri&$args;
    }

    # Applies to every PHP file
    location ~ /.php$ {
        # Ensure file really exists
        if (!-e $request_filename) {
            return 404;
```

```
        }
        # Pass the request to your PHP-FPM backend
        fastcgi_pass 127.0.0.1:9000
        fastcgi_index index.php;
        fastcgi_param PATH_INFO $fastcgi_script_name;
        include fastcgi_params;
    }
}
```

Self Signed Certificates

- Self-signed certificates can be generated by yourselff on your own server
- Certificated signed by a trusted certificate authority offer an additional level of security: a tihird party ascertains the authenticity of the server to the visitors

For testing env or websites for a restrictued amount of visitors, self signed certificates can be an option

```
apt-get install openssl
openssl genrsa --out example.com.key 2048
openssl req -new -key example.com.key example.com.csr
```

Enabling HTTPS in NGINX Configurtion

```
cat your_site_certificate.crt certificate_authority.crt > example.com.crt
```

1. Site certificate
2. CA certificate

```
listen 443 default_server ssl;

# Generated certificated file
ssl_certificate /etc/ssl/private/example.com.crt
# Private key file generated
ssl_certificate_key /etc/ssl/private/example.com.key;

ssl_session_cache shared:SSL:20m;
ssl_session_timeout 60m;

# Disable SSL in favor of TLS
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
```

Own Cloud Drive

SImilar Dropbox services that allow to store files onlijne and retrieve them easily from all sorts of devices, including mobile phones and tablets

1. Install PHP
2. Install MySQL
3. Configure PHP taking care of directives regarding maximum file upload size.
4. Create PHP-FPM pool dedicated to ownCloud.
5. Set up a SQL database and user.

- www.owncloud.org

Troubleshooting

Tips

Checking Access Permissions

- Loag of errors are caused by invalid access permissions. You are offered to specify a user and a group for the Nginx worker processes to run:

1. When configuring the build with the configure command
2. In the configuration file, user directive allows to specify the user and group

Testing Configuration

A common mistake: after having modified the config file (often without a backup) they reload nginx to apply the new configuration

If the configuration file contains syntax or semantic errors, application wil refuse the reload or Nginx is stopped.

Recommendations:

- keep a backup of working configuration files in case something goes wrong
- Before reloading, test ```nginx -t -c /path/to/config/file.conf
- reload server instead of restarting. `service nginx reload`. It will keep existing connections alive.

Reload Server

```
service nginx reload
/etc/init.d/nginx reload
/usr/local/nginx/sbin/nginx -s reload
```

Checking Logs

The error should be located in the /logs/ directroy of Nginx setup. Default is /usr/localnginx/ogs or /var/log/nginx.

Forbidden Custom Error Page

deny or allow directives: clients being denied will fall back on 403 forbidden error page.

```
server {
    allow 192.168.0.0/16;
    deny all;
    error_page 403 /error403.html
}
```

The problem is simply. Nginx also denies access to custom 403 error page. You need to override the access rules in a location block specifically matching the page.

```
server {
    location / {
        error_page 403 /error403.html;
        allow 192.168.0.0/16;
        deny all;
    }
    location = /error403.html {
        allow all;
    }
}
```

```
# Form more than just one error page
server {
    location / {
        error_page 504 /error403.html;
        error_page 404 /error404.html;
        allow 192.168.0.0/16;
        deny all;
    }
    location ~ "^/error[0-9]{3}\.html$" {
        allow all;
    }
}
```

400 Bad Request

Only stops happening when visitors clear their cache and cookies. Is caused by an overly large header field sent by the client. This ocurrs when cookie data exceeds a certain size.

```
# Increase the  buffer to allow larger cookie data size
large_client_header_buffers
large_client_header_buffers 4 16k
```

Truncated or invalid FastCGI responses

- Setup a writable directory for the temporary FastCGI files. `fastcgi_temp_path`

Location Block Prioritties

When using multiple location blocks in the same server block in the same server block is that the config does not apply as you thought it would.

Location blocks are processed in a specific order by priorities.

If Block Issues

Avoid using if blocks

Nginx Docker Hub

About

- Nginx is an open source reverse proxy server for
  - HTTP
  - HTTPS
  - STMP
  - POP3
  - IMAP
- Load Balancer
- HTTP Cache
- Web Server

Adventajes

- High Performance
- High Concurrency
- Low Memory Usage

Image Usage

Hosting Static Content

```
docker run --name my-nginx-container -v
/host/path/nginx.conf:/etc/nginx/nginx/.conf:ro -d nginx
```

Dockerfile can be used to generate a new image that includes the necessary content

```
FROM nginx
COPY static-html-directory /usr/share/nginx/html
```

Place this file in the same directory as your directory of content, run `docker build -t some-content-nginx .` then start the container

```
docker run --name some-nginx -d -p 8080:80 some-content-nginx
```

then you can visit `http://localhost:8080/` in the browser.

Complex Config

```
docker run --name my-custom-nginx-container -v
/host/path/nginx.conf:/etc/nginx/nginx.conf:ro -d nginx
```

to addapt default configuration file

```
docker run --name tmp-nginx-container -d nginx
docker cp tmp-nginx-container:/etc/nginx/nginx.conf /host/path/nginx.conf
docker rm -f tmp-nginx-container
```

```
FROM nginx
COPY nginx.conf /etc/nginx/nginx.conf
```

To add custom `CMD` in Dockerfile be sure to include `-g daemon off;` in the CMD order for nginx to stay in the foreground, so that docker can track the preocess properly.

Then build the image with `docker build -t custom-nginx .` an run it as

```
docker run --name my-custom-nginx-container -d custom-nginx
```

Environment Variables

- Nginx does not support environment variables inside most configuration blocks
- This image extract environment variables before nginx starts.

Using docker-compose:

```
web:
  image: nginx
  volumes:
    - ./templates:/etc/nginx/templates
  ports:
    - "8080:80"
  environment:
    - NGINX_HOST=foobar.com
    - NGINX_PORT=80
```

This function reads template files in /etc/nginx/templates/*.template and outputs the result of executing `envsubst` to `/etc/nginx/conf.d`.

Nginx Read Only Mode

1. Mount a docker volume to every location where nginx writes information.

2. Default Nginx config requires write-access to /var/cache and /var/run
3. To advanced configuration that requires nginx to write other location, add more volume mounts to those locations.

```
docker run -d -p 80:80 --read-only -v $(pwd)/nginx-cache:/var/cache/nginx -
v $(pwd)/nginx-pid:/var/run nginx
```

Debug Mode

```
docker run --name my-nginx -v
/host/path/nginx.conf:/etc/nginx/nginx.conf:ro -d nginx nginx-debug -g
'daemon off;'
```

```
web:
  image: nginx
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf:ro
  command: [nginx=debug, '-g', 'daemon-off;']
```