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.