## Module ngx\_http\_core\_module

#### Directives

#### 

| Syntax: | **absolute\_redirect** on | off; |
| --- | --- |
| Default: | absolute\_redirect on; |
| Context: | http, server, location |

This directive appeared in version 1.11.8.

If disabled, redirects issued by nginx will be relative.

See also [server\_name\_in\_redirect](https://nginx.org/en/docs/http/ngx_http_core_module.html#server_name_in_redirect) and [port\_in\_redirect](https://nginx.org/en/docs/http/ngx_http_core_module.html#port_in_redirect) directives.

| Syntax: | **aio** on | off | threads[=*pool*]; |
| --- | --- |
| Default: | aio off; |
| Context: | http, server, location |

This directive appeared in version 0.8.11.

Enables or disables the use of asynchronous file I/O (AIO) on FreeBSD and Linux:

location /video/ {

aio on;

output\_buffers 1 64k;

}

On FreeBSD, AIO can be used starting from FreeBSD 4.3. Prior to FreeBSD 11.0, AIO can either be linked statically into a kernel:

options VFS\_AIO

or loaded dynamically as a kernel loadable module:

kldload aio

On Linux, AIO can be used starting from kernel version 2.6.22. Also, it is necessary to enable [directio](https://nginx.org/en/docs/http/ngx_http_core_module.html#directio), or otherwise reading will be blocking:

location /video/ {

aio on;

directio 512;

output\_buffers 1 128k;

}

On Linux, [directio](https://nginx.org/en/docs/http/ngx_http_core_module.html#directio) can only be used for reading blocks that are aligned on 512-byte boundaries (or 4K for XFS). File’s unaligned end is read in blocking mode. The same holds true for byte range requests and for FLV requests not from the beginning of a file: reading of unaligned data at the beginning and end of a file will be blocking.

When both AIO and [sendfile](https://nginx.org/en/docs/http/ngx_http_core_module.html#sendfile) are enabled on Linux, AIO is used for files that are larger than or equal to the size specified in the [directio](https://nginx.org/en/docs/http/ngx_http_core_module.html#directio) directive, while [sendfile](https://nginx.org/en/docs/http/ngx_http_core_module.html#sendfile) is used for files of smaller sizes or when [directio](https://nginx.org/en/docs/http/ngx_http_core_module.html#directio) is disabled.

location /video/ {

sendfile on;

aio on;

directio 8m;

}

Finally, files can be read and [sent](https://nginx.org/en/docs/http/ngx_http_core_module.html#sendfile) using multi-threading (1.7.11), without blocking a worker process:

location /video/ {

sendfile on;

aio threads;

}

Read and send file operations are offloaded to threads of the specified [pool](https://nginx.org/en/docs/ngx_core_module.html#thread_pool). If the pool name is omitted, the pool with the name “default” is used. The pool name can also be set with variables:

aio threads=pool$disk;

By default, multi-threading is disabled, it should be enabled with the --with-threads configuration parameter. Currently, multi-threading is compatible only with the [epoll](https://nginx.org/en/docs/events.html#epoll), [kqueue](https://nginx.org/en/docs/events.html#kqueue), and [eventport](https://nginx.org/en/docs/events.html#eventport) methods. Multi-threaded sending of files is only supported on Linux.

See also the [sendfile](https://nginx.org/en/docs/http/ngx_http_core_module.html#sendfile) directive.

| Syntax: | **aio\_write** on | off; |
| --- | --- |
| Default: | aio\_write off; |
| Context: | http, server, location |

This directive appeared in version 1.9.13.

If [aio](https://nginx.org/en/docs/http/ngx_http_core_module.html#aio) is enabled, specifies whether it is used for writing files. Currently, this only works when using aio threads and is limited to writing temporary files with data received from proxied servers.

| Syntax: | **alias** *path*; |
| --- | --- |
| Default: | — |
| Context: | location |

Defines a replacement for the specified location. For example, with the following configuration

location /i/ {

alias /data/w3/images/;

}

on request of “/i/top.gif”, the file /data/w3/images/top.gif will be sent.

The *path* value can contain variables, except $document\_root and $realpath\_root.

If alias is used inside a location defined with a regular expression then such regular expression should contain captures and alias should refer to these captures (0.7.40), for example:

location ~ ^/users/(.+\.(?:gif|jpe?g|png))$ {

alias /data/w3/images/$1;

}

When location matches the last part of the directive’s value:

location /images/ {

alias /data/w3/images/;

}

it is better to use the [root](https://nginx.org/en/docs/http/ngx_http_core_module.html#root) directive instead:

location /images/ {

root /data/w3;

}

| Syntax: | **auth\_delay** *time*; |
| --- | --- |
| Default: | auth\_delay 0s; |
| Context: | http, server, location |

This directive appeared in version 1.17.10.

Delays processing of unauthorized requests with 401 response code to prevent timing attacks when access is limited by [password](https://nginx.org/en/docs/http/ngx_http_auth_basic_module.html), by the [result of subrequest](https://nginx.org/en/docs/http/ngx_http_auth_request_module.html), or by [JWT](https://nginx.org/en/docs/http/ngx_http_auth_jwt_module.html).

| Syntax: | **chunked\_transfer\_encoding** on | off; |
| --- | --- |
| Default: | chunked\_transfer\_encoding on; |
| Context: | http, server, location |

Allows disabling chunked transfer encoding in HTTP/1.1. It may come in handy when using a software failing to support chunked encoding despite the standard’s requirement.

| Syntax: | **client\_body\_buffer\_size** *size*; |
| --- | --- |
| Default: | client\_body\_buffer\_size 8k|16k; |
| Context: | http, server, location |

Sets buffer size for reading client request body. In case the request body is larger than the buffer, the whole body or only its part is written to a [temporary file](https://nginx.org/en/docs/http/ngx_http_core_module.html#client_body_temp_path). By default, buffer size is equal to two memory pages. This is 8K on x86, other 32-bit platforms, and x86-64. It is usually 16K on other 64-bit platforms.

| Syntax: | **client\_body\_in\_file\_only** on | clean | off; |
| --- | --- |
| Default: | client\_body\_in\_file\_only off; |
| Context: | http, server, location |

Determines whether nginx should save the entire client request body into a file. This directive can be used during debugging, or when using the $request\_body\_file variable, or the [$r->request\_body\_file](https://nginx.org/en/docs/http/ngx_http_perl_module.html#methods) method of the module [ngx\_http\_perl\_module](https://nginx.org/en/docs/http/ngx_http_perl_module.html).

When set to the value on, temporary files are not removed after request processing.

The value clean will cause the temporary files left after request processing to be removed.

| Syntax: | **client\_body\_in\_single\_buffer** on | off; |
| --- | --- |
| Default: | client\_body\_in\_single\_buffer off; |
| Context: | http, server, location |

Determines whether nginx should save the entire client request body in a single buffer. The directive is recommended when using the $request\_body variable, to save the number of copy operations involved.

| Syntax: | **client\_body\_temp\_path** *path* [*level1* [*level2* [*level3*]]]; |
| --- | --- |
| Default: | client\_body\_temp\_path client\_body\_temp; |
| Context: | http, server, location |

Defines a directory for storing temporary files holding client request bodies. Up to three-level subdirectory hierarchy can be used under the specified directory. For example, in the following configuration

client\_body\_temp\_path /spool/nginx/client\_temp 1 2;

a path to a temporary file might look like this:

/spool/nginx/client\_temp/7/45/00000123457

| Syntax: | **client\_body\_timeout** *time*; |
| --- | --- |
| Default: | client\_body\_timeout 60s; |
| Context: | http, server, location |

Defines a timeout for reading client request body. The timeout is set only for a period between two successive read operations, not for the transmission of the whole request body. If a client does not transmit anything within this time, the request is terminated with the 408 (Request Time-out) error.

| Syntax: | **client\_header\_buffer\_size** *size*; |
| --- | --- |
| Default: | client\_header\_buffer\_size 1k; |
| Context: | http, server |

Sets buffer size for reading client request header. For most requests, a buffer of 1K bytes is enough. However, if a request includes long cookies, or comes from a WAP client, it may not fit into 1K. If a request line or a request header field does not fit into this buffer then larger buffers, configured by the [large\_client\_header\_buffers](https://nginx.org/en/docs/http/ngx_http_core_module.html#large_client_header_buffers) directive, are allocated.

| Syntax: | **client\_header\_timeout** *time*; |
| --- | --- |
| Default: | client\_header\_timeout 60s; |
| Context: | http, server |

Defines a timeout for reading client request header. If a client does not transmit the entire header within this time, the request is terminated with the 408 (Request Time-out) error.

| Syntax: | **client\_max\_body\_size** *size*; |
| --- | --- |
| Default: | client\_max\_body\_size 1m; |
| Context: | http, server, location |

Sets the maximum allowed size of the client request body. If the size in a request exceeds the configured value, the 413 (Request Entity Too Large) error is returned to the client. Please be aware that browsers cannot correctly display this error. Setting *size* to 0 disables checking of client request body size.

| Syntax: | **connection\_pool\_size** *size*; |
| --- | --- |
| Default: | connection\_pool\_size 256|512; |
| Context: | http, server |

Allows accurate tuning of per-connection memory allocations. This directive has minimal impact on performance and should not generally be used. By default, the size is equal to 256 bytes on 32-bit platforms and 512 bytes on 64-bit platforms.

Prior to version 1.9.8, the default value was 256 on all platforms.

| Syntax: | **default\_type** *mime-type*; |
| --- | --- |
| Default: | default\_type text/plain; |
| Context: | http, server, location |

Defines the default MIME type of a response. Mapping of file name extensions to MIME types can be set with the [types](https://nginx.org/en/docs/http/ngx_http_core_module.html#types) directive.

| Syntax: | **directio** *size* | off; |
| --- | --- |
| Default: | directio off; |
| Context: | http, server, location |

This directive appeared in version 0.7.7.

Enables the use of the O\_DIRECT flag (FreeBSD, Linux), the F\_NOCACHE flag (macOS), or the directio() function (Solaris), when reading files that are larger than or equal to the specified *size*. The directive automatically disables (0.7.15) the use of [sendfile](https://nginx.org/en/docs/http/ngx_http_core_module.html#sendfile) for a given request. It can be useful for serving large files:

directio 4m;

or when using [aio](https://nginx.org/en/docs/http/ngx_http_core_module.html#aio) on Linux.

| Syntax: | **directio\_alignment** *size*; |
| --- | --- |
| Default: | directio\_alignment 512; |
| Context: | http, server, location |

This directive appeared in version 0.8.11.

Sets the alignment for [directio](https://nginx.org/en/docs/http/ngx_http_core_module.html#directio). In most cases, a 512-byte alignment is enough. However, when using XFS under Linux, it needs to be increased to 4K.

| Syntax: | **disable\_symlinks** off; **disable\_symlinks** on | if\_not\_owner [from=*part*]; |
| --- | --- |
| Default: | disable\_symlinks off; |
| Context: | http, server, location |

This directive appeared in version 1.1.15.

Determines how symbolic links should be treated when opening files:

off

Symbolic links in the pathname are allowed and not checked. This is the default behavior.

on

If any component of the pathname is a symbolic link, access to a file is denied.

if\_not\_owner

Access to a file is denied if any component of the pathname is a symbolic link, and the link and object that the link points to have different owners.

from=*part*

When checking symbolic links (parameters on and if\_not\_owner), all components of the pathname are normally checked. Checking of symbolic links in the initial part of the pathname may be avoided by specifying additionally the from=*part* parameter. In this case, symbolic links are checked only from the pathname component that follows the specified initial part. If the value is not an initial part of the pathname checked, the whole pathname is checked as if this parameter was not specified at all. If the value matches the whole file name, symbolic links are not checked. The parameter value can contain variables.

Example:

disable\_symlinks on from=$document\_root;

This directive is only available on systems that have the openat() and fstatat() interfaces. Such systems include modern versions of FreeBSD, Linux, and Solaris.

Parameters on and if\_not\_owner add a processing overhead.

On systems that do not support opening of directories only for search, to use these parameters it is required that worker processes have read permissions for all directories being checked.

The [ngx\_http\_autoindex\_module](https://nginx.org/en/docs/http/ngx_http_autoindex_module.html), [ngx\_http\_random\_index\_module](https://nginx.org/en/docs/http/ngx_http_random_index_module.html), and [ngx\_http\_dav\_module](https://nginx.org/en/docs/http/ngx_http_dav_module.html) modules currently ignore this directive.

| Syntax: | **error\_page** *code* ... [=[*response*]] *uri*; |
| --- | --- |
| Default: | — |
| Context: | http, server, location, if in location |

Defines the URI that will be shown for the specified errors. A *uri* value can contain variables.

Example:

error\_page 404 /404.html;

error\_page 500 502 503 504 /50x.html;

This causes an internal redirect to the specified *uri* with the client request method changed to “GET” (for all methods other than “GET” and “HEAD”).

Furthermore, it is possible to change the response code to another using the “=*response*” syntax, for example:

error\_page 404 =200 /empty.gif;

If an error response is processed by a proxied server or a FastCGI/uwsgi/SCGI/gRPC server, and the server may return different response codes (e.g., 200, 302, 401 or 404), it is possible to respond with the code it returns:

error\_page 404 = /404.php;

If there is no need to change URI and method during internal redirection it is possible to pass error processing into a named location:

location / {

error\_page 404 = @fallback;

}

location @fallback {

proxy\_pass http://backend;

}

If *uri* processing leads to an error, the status code of the last occurred error is returned to the client.

It is also possible to use URL redirects for error processing:

error\_page 403 http://example.com/forbidden.html;

error\_page 404 =301 http://example.com/notfound.html;

In this case, by default, the response code 302 is returned to the client. It can only be changed to one of the redirect status codes (301, 302, 303, 307, and 308).

The code 307 was not treated as a redirect until versions 1.1.16 and 1.0.13.

The code 308 was not treated as a redirect until version 1.13.0.

These directives are inherited from the previous configuration level if and only if there are no error\_page directives defined on the current level.

| Syntax: | **etag** on | off; |
| --- | --- |
| Default: | etag on; |
| Context: | http, server, location |

This directive appeared in version 1.3.3.

Enables or disables automatic generation of the “ETag” response header field for static resources.

| Syntax: | **http** { ... } |
| --- | --- |
| Default: | — |
| Context: | main |

Provides the configuration file context in which the HTTP server directives are specified.

| Syntax: | **if\_modified\_since** off | exact | before; |
| --- | --- |
| Default: | if\_modified\_since exact; |
| Context: | http, server, location |

This directive appeared in version 0.7.24.

Specifies how to compare modification time of a response with the time in the “If-Modified-Since” request header field:

off

the “If-Modified-Since” request header field is ignored (0.7.34);

exact

exact match;

before

modification time of a response is less than or equal to the time in the “If-Modified-Since” request header field.

| Syntax: | **ignore\_invalid\_headers** on | off; |
| --- | --- |
| Default: | ignore\_invalid\_headers on; |
| Context: | http, server |

Controls whether header fields with invalid names should be ignored. Valid names are composed of English letters, digits, hyphens, and possibly underscores (as controlled by the [underscores\_in\_headers](https://nginx.org/en/docs/http/ngx_http_core_module.html#underscores_in_headers) directive).

If the directive is specified on the [server](https://nginx.org/en/docs/http/ngx_http_core_module.html#server) level, its value is only used if a server is a default one. The value specified also applies to all virtual servers listening on the same address and port.

| Syntax: | **internal**; |
| --- | --- |
| Default: | — |
| Context: | location |

Specifies that a given location can only be used for internal requests. For external requests, the client error 404 (Not Found) is returned. Internal requests are the following:

* requests redirected by the [error\_page](https://nginx.org/en/docs/http/ngx_http_core_module.html#error_page), [index](https://nginx.org/en/docs/http/ngx_http_index_module.html#index), [random\_index](https://nginx.org/en/docs/http/ngx_http_random_index_module.html#random_index), and [try\_files](https://nginx.org/en/docs/http/ngx_http_core_module.html#try_files) directives;
* requests redirected by the “X-Accel-Redirect” response header field from an upstream server;
* subrequests formed by the “include virtual” command of the [ngx\_http\_ssi\_module](https://nginx.org/en/docs/http/ngx_http_ssi_module.html) module, by the [ngx\_http\_addition\_module](https://nginx.org/en/docs/http/ngx_http_addition_module.html) module directives, and by [auth\_request](https://nginx.org/en/docs/http/ngx_http_auth_request_module.html#auth_request) and [mirror](https://nginx.org/en/docs/http/ngx_http_mirror_module.html#mirror) directives;
* requests changed by the [rewrite](https://nginx.org/en/docs/http/ngx_http_rewrite_module.html#rewrite) directive.

Example:

error\_page 404 /404.html;

location = /404.html {

internal;

}

There is a limit of 10 internal redirects per request to prevent request processing cycles that can occur in incorrect configurations. If this limit is reached, the error 500 (Internal Server Error) is returned. In such cases, the “rewrite or internal redirection cycle” message can be seen in the error log.

| Syntax: | **keepalive\_disable** none | *browser* ...; |
| --- | --- |
| Default: | keepalive\_disable msie6; |
| Context: | http, server, location |

Disables keep-alive connections with misbehaving browsers. The *browser* parameters specify which browsers will be affected. The value msie6 disables keep-alive connections with old versions of MSIE, once a POST request is received. The value safari disables keep-alive connections with Safari and Safari-like browsers on macOS and macOS-like operating systems. The value none enables keep-alive connections with all browsers.

Prior to version 1.1.18, the value safari matched all Safari and Safari-like browsers on all operating systems, and keep-alive connections with them were disabled by default.

| Syntax: | **keepalive\_requests** *number*; |
| --- | --- |
| Default: | keepalive\_requests 1000; |
| Context: | http, server, location |

This directive appeared in version 0.8.0.

Sets the maximum number of requests that can be served through one keep-alive connection. After the maximum number of requests are made, the connection is closed.

Closing connections periodically is necessary to free per-connection memory allocations. Therefore, using too high maximum number of requests could result in excessive memory usage and not recommended.

Prior to version 1.19.10, the default value was 100.

| Syntax: | **keepalive\_time** *time*; |
| --- | --- |
| Default: | keepalive\_time 1h; |
| Context: | http, server, location |

This directive appeared in version 1.19.10.

Limits the maximum time during which requests can be processed through one keep-alive connection. After this time is reached, the connection is closed following the subsequent request processing.

| Syntax: | **keepalive\_timeout** *timeout* [*header\_timeout*]; |
| --- | --- |
| Default: | keepalive\_timeout 75s; |
| Context: | http, server, location |

The first parameter sets a timeout during which a keep-alive client connection will stay open on the server side. The zero value disables keep-alive client connections. The optional second parameter sets a value in the “Keep-Alive: timeout=*time*” response header field. Two parameters may differ.

The “Keep-Alive: timeout=*time*” header field is recognized by Mozilla and Konqueror. MSIE closes keep-alive connections by itself in about 60 seconds.

| Syntax: | **large\_client\_header\_buffers** *number* *size*; |
| --- | --- |
| Default: | large\_client\_header\_buffers 4 8k; |
| Context: | http, server |

Sets the maximum *number* and *size* of buffers used for reading large client request header. A request line cannot exceed the size of one buffer, or the 414 (Request-URI Too Large) error is returned to the client. A request header field cannot exceed the size of one buffer as well, or the 400 (Bad Request) error is returned to the client. Buffers are allocated only on demand. By default, the buffer size is equal to 8K bytes. If after the end of request processing a connection is transitioned into the keep-alive state, these buffers are released.

| Syntax: | **limit\_except** *method* ... { ... } |
| --- | --- |
| Default: | — |
| Context: | location |

Limits allowed HTTP methods inside a location. The *method* parameter can be one of the following: GET, HEAD, POST, PUT, DELETE, MKCOL, COPY, MOVE, OPTIONS, PROPFIND, PROPPATCH, LOCK, UNLOCK, or PATCH. Allowing the GET method makes the HEAD method also allowed. Access to other methods can be limited using the [ngx\_http\_access\_module](https://nginx.org/en/docs/http/ngx_http_access_module.html), [ngx\_http\_auth\_basic\_module](https://nginx.org/en/docs/http/ngx_http_auth_basic_module.html), and [ngx\_http\_auth\_jwt\_module](https://nginx.org/en/docs/http/ngx_http_auth_jwt_module.html) (1.13.10) modules directives:

limit\_except GET {

allow 192.168.1.0/32;

deny all;

}

Please note that this will limit access to all methods **except** GET and HEAD.

| Syntax: | **limit\_rate** *rate*; |
| --- | --- |
| Default: | limit\_rate 0; |
| Context: | http, server, location, if in location |

Limits the rate of response transmission to a client. The *rate* is specified in bytes per second. The zero value disables rate limiting. The limit is set per a request, and so if a client simultaneously opens two connections, the overall rate will be twice as much as the specified limit.

Parameter value can contain variables (1.17.0). It may be useful in cases where rate should be limited depending on a certain condition:

map $slow $rate {

1 4k;

2 8k;

}

limit\_rate $rate;

Rate limit can also be set in the [$limit\_rate](https://nginx.org/en/docs/http/ngx_http_core_module.html#var_limit_rate) variable, however, since version 1.17.0, this method is not recommended:

server {

if ($slow) {

set $limit\_rate 4k;

}

...

}

Rate limit can also be set in the “X-Accel-Limit-Rate” header field of a proxied server response. This capability can be disabled using the [proxy\_ignore\_headers](https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_ignore_headers), [fastcgi\_ignore\_headers](https://nginx.org/en/docs/http/ngx_http_fastcgi_module.html#fastcgi_ignore_headers), [uwsgi\_ignore\_headers](https://nginx.org/en/docs/http/ngx_http_uwsgi_module.html#uwsgi_ignore_headers), and [scgi\_ignore\_headers](https://nginx.org/en/docs/http/ngx_http_scgi_module.html#scgi_ignore_headers) directives.

| Syntax: | **limit\_rate\_after** *size*; |
| --- | --- |
| Default: | limit\_rate\_after 0; |
| Context: | http, server, location, if in location |

This directive appeared in version 0.8.0.

Sets the initial amount after which the further transmission of a response to a client will be rate limited. Parameter value can contain variables (1.17.0).

Example:

location /flv/ {

flv;

limit\_rate\_after 500k;

limit\_rate 50k;

}

| Syntax: | **lingering\_close** off | on | always; |
| --- | --- |
| Default: | lingering\_close on; |
| Context: | http, server, location |

This directive appeared in versions 1.1.0 and 1.0.6.

Controls how nginx closes client connections.

The default value “on” instructs nginx to [wait for](https://nginx.org/en/docs/http/ngx_http_core_module.html#lingering_timeout) and [process](https://nginx.org/en/docs/http/ngx_http_core_module.html#lingering_time) additional data from a client before fully closing a connection, but only if heuristics suggests that a client may be sending more data.

The value “always” will cause nginx to unconditionally wait for and process additional client data.

The value “off” tells nginx to never wait for more data and close the connection immediately. This behavior breaks the protocol and should not be used under normal circumstances.

To control closing [HTTP/2](https://nginx.org/en/docs/http/ngx_http_v2_module.html) connections, the directive must be specified on the [server](https://nginx.org/en/docs/http/ngx_http_core_module.html#server) level (1.19.1).

| Syntax: | **lingering\_time** *time*; |
| --- | --- |
| Default: | lingering\_time 30s; |
| Context: | http, server, location |

When [lingering\_close](https://nginx.org/en/docs/http/ngx_http_core_module.html#lingering_close) is in effect, this directive specifies the maximum time during which nginx will process (read and ignore) additional data coming from a client. After that, the connection will be closed, even if there will be more data.

| Syntax: | **lingering\_timeout** *time*; |
| --- | --- |
| Default: | lingering\_timeout 5s; |
| Context: | http, server, location |

When [lingering\_close](https://nginx.org/en/docs/http/ngx_http_core_module.html#lingering_close) is in effect, this directive specifies the maximum waiting time for more client data to arrive. If data are not received during this time, the connection is closed. Otherwise, the data are read and ignored, and nginx starts waiting for more data again. The “wait-read-ignore” cycle is repeated, but no longer than specified by the [lingering\_time](https://nginx.org/en/docs/http/ngx_http_core_module.html#lingering_time) directive.

| Syntax: | **listen** *address*[:*port*] [default\_server] [ssl] [http2 | spdy] [proxy\_protocol] [setfib=*number*] [fastopen=*number*] [backlog=*number*] [rcvbuf=*size*] [sndbuf=*size*] [accept\_filter=*filter*] [deferred] [bind] [ipv6only=on|off] [reuseport] [so\_keepalive=on|off|[*keepidle*]:[*keepintvl*]:[*keepcnt*]]; **listen** *port* [default\_server] [ssl] [http2 | spdy] [proxy\_protocol] [setfib=*number*] [fastopen=*number*] [backlog=*number*] [rcvbuf=*size*] [sndbuf=*size*] [accept\_filter=*filter*] [deferred] [bind] [ipv6only=on|off] [reuseport] [so\_keepalive=on|off|[*keepidle*]:[*keepintvl*]:[*keepcnt*]]; **listen** unix:*path* [default\_server] [ssl] [http2 | spdy] [proxy\_protocol] [backlog=*number*] [rcvbuf=*size*] [sndbuf=*size*] [accept\_filter=*filter*] [deferred] [bind] [so\_keepalive=on|off|[*keepidle*]:[*keepintvl*]:[*keepcnt*]]; |
| --- | --- |
| Default: | listen \*:80 | \*:8000; |
| Context: | server |

Sets the *address* and *port* for IP, or the *path* for a UNIX-domain socket on which the server will accept requests. Both *address* and *port*, or only *address* or only *port* can be specified. An *address* may also be a hostname, for example:

listen 127.0.0.1:8000;

listen 127.0.0.1;

listen 8000;

listen \*:8000;

listen localhost:8000;

IPv6 addresses (0.7.36) are specified in square brackets:

listen [::]:8000;

listen [::1];

UNIX-domain sockets (0.8.21) are specified with the “unix:” prefix:

listen unix:/var/run/nginx.sock;

If only *address* is given, the port 80 is used.

If the directive is not present then either \*:80 is used if nginx runs with the superuser privileges, or \*:8000 otherwise.

The default\_server parameter, if present, will cause the server to become the default server for the specified *address*:*port* pair. If none of the directives have the default\_server parameter then the first server with the *address*:*port* pair will be the default server for this pair.

In versions prior to 0.8.21 this parameter is named simply default.

The ssl parameter (0.7.14) allows specifying that all connections accepted on this port should work in SSL mode. This allows for a more compact [configuration](https://nginx.org/en/docs/http/configuring_https_servers.html#single_http_https_server) for the server that handles both HTTP and HTTPS requests.

The http2 parameter (1.9.5) configures the port to accept [HTTP/2](https://nginx.org/en/docs/http/ngx_http_v2_module.html) connections. Normally, for this to work the ssl parameter should be specified as well, but nginx can also be configured to accept HTTP/2 connections without SSL.

The spdy parameter (1.3.15-1.9.4) allows accepting [SPDY](https://nginx.org/en/docs/http/ngx_http_spdy_module.html) connections on this port. Normally, for this to work the ssl parameter should be specified as well, but nginx can also be configured to accept SPDY connections without SSL.

The proxy\_protocol parameter (1.5.12) allows specifying that all connections accepted on this port should use the [PROXY protocol](http://www.haproxy.org/download/1.5/doc/proxy-protocol.txt).

The PROXY protocol version 2 is supported since version 1.13.11.

The listen directive can have several additional parameters specific to socket-related system calls. These parameters can be specified in any listen directive, but only once for a given *address*:*port* pair.

In versions prior to 0.8.21, they could only be specified in the listen directive together with the default parameter.

setfib=*number*

this parameter (0.8.44) sets the associated routing table, FIB (the SO\_SETFIB option) for the listening socket. This currently works only on FreeBSD.

fastopen=*number*

enables “[TCP Fast Open](http://en.wikipedia.org/wiki/TCP_Fast_Open)” for the listening socket (1.5.8) and [limits](https://tools.ietf.org/html/rfc7413#section-5.1) the maximum length for the queue of connections that have not yet completed the three-way handshake.Do not enable this feature unless the server can handle receiving the [same SYN packet with data](https://tools.ietf.org/html/rfc7413#section-6.1) more than once.

backlog=*number*

sets the backlog parameter in the listen() call that limits the maximum length for the queue of pending connections. By default, backlog is set to -1 on FreeBSD, DragonFly BSD, and macOS, and to 511 on other platforms.

rcvbuf=*size*

sets the receive buffer size (the SO\_RCVBUF option) for the listening socket.

sndbuf=*size*

sets the send buffer size (the SO\_SNDBUF option) for the listening socket.

accept\_filter=*filter*

sets the name of accept filter (the SO\_ACCEPTFILTER option) for the listening socket that filters incoming connections before passing them to accept(). This works only on FreeBSD and NetBSD 5.0+. Possible values are [dataready](http://man.freebsd.org/accf_data) and [httpready](http://man.freebsd.org/accf_http).

deferred

instructs to use a deferred accept() (the TCP\_DEFER\_ACCEPT socket option) on Linux.

bind

instructs to make a separate bind() call for a given *address*:*port* pair. This is useful because if there are several listen directives with the same port but different addresses, and one of the listen directives listens on all addresses for the given port (\*:*port*), nginx will bind() only to \*:*port*. It should be noted that the getsockname() system call will be made in this case to determine the address that accepted the connection. If the setfib, fastopen, backlog, rcvbuf, sndbuf, accept\_filter, deferred, ipv6only, reuseport, or so\_keepalive parameters are used then for a given *address*:*port* pair a separate bind() call will always be made.

ipv6only=on|off

this parameter (0.7.42) determines (via the IPV6\_V6ONLY socket option) whether an IPv6 socket listening on a wildcard address [::] will accept only IPv6 connections or both IPv6 and IPv4 connections. This parameter is turned on by default. It can only be set once on start.Prior to version 1.3.4, if this parameter was omitted then the operating system’s settings were in effect for the socket.

reuseport

this parameter (1.9.1) instructs to create an individual listening socket for each worker process (using the SO\_REUSEPORT socket option on Linux 3.9+ and DragonFly BSD, or SO\_REUSEPORT\_LB on FreeBSD 12+), allowing a kernel to distribute incoming connections between worker processes. This currently works only on Linux 3.9+, DragonFly BSD, and FreeBSD 12+ (1.15.1).Inappropriate use of this option may have its security [implications](http://man7.org/linux/man-pages/man7/socket.7.html).

so\_keepalive=on|off|[*keepidle*]:[*keepintvl*]:[*keepcnt*]

this parameter (1.1.11) configures the “TCP keepalive” behavior for the listening socket. If this parameter is omitted then the operating system’s settings will be in effect for the socket. If it is set to the value “on”, the SO\_KEEPALIVE option is turned on for the socket. If it is set to the value “off”, the SO\_KEEPALIVE option is turned off for the socket. Some operating systems support setting of TCP keepalive parameters on a per-socket basis using the TCP\_KEEPIDLE, TCP\_KEEPINTVL, and TCP\_KEEPCNT socket options. On such systems (currently, Linux 2.4+, NetBSD 5+, and FreeBSD 9.0-STABLE), they can be configured using the *keepidle*, *keepintvl*, and *keepcnt* parameters. One or two parameters may be omitted, in which case the system default setting for the corresponding socket option will be in effect. For example,

so\_keepalive=30m::10

will set the idle timeout (TCP\_KEEPIDLE) to 30 minutes, leave the probe interval (TCP\_KEEPINTVL) at its system default, and set the probes count (TCP\_KEEPCNT) to 10 probes.

Example:

listen 127.0.0.1 default\_server accept\_filter=dataready backlog=1024;

| Syntax: | **location** [ = | ~ | ~\* | ^~ ] *uri* { ... } **location** @*name* { ... } |
| --- | --- |
| Default: | — |
| Context: | server, location |

Sets configuration depending on a request URI.

The matching is performed against a normalized URI, after decoding the text encoded in the “%XX” form, resolving references to relative path components “.” and “..”, and possible [compression](https://nginx.org/en/docs/http/ngx_http_core_module.html#merge_slashes) of two or more adjacent slashes into a single slash.

A location can either be defined by a prefix string, or by a regular expression. Regular expressions are specified with the preceding “~\*” modifier (for case-insensitive matching), or the “~” modifier (for case-sensitive matching). To find location matching a given request, nginx first checks locations defined using the prefix strings (prefix locations). Among them, the location with the longest matching prefix is selected and remembered. Then regular expressions are checked, in the order of their appearance in the configuration file. The search of regular expressions terminates on the first match, and the corresponding configuration is used. If no match with a regular expression is found then the configuration of the prefix location remembered earlier is used.

location blocks can be nested, with some exceptions mentioned below.

For case-insensitive operating systems such as macOS and Cygwin, matching with prefix strings ignores a case (0.7.7). However, comparison is limited to one-byte locales.

Regular expressions can contain captures (0.7.40) that can later be used in other directives.

If the longest matching prefix location has the “^~” modifier then regular expressions are not checked.

Also, using the “=” modifier it is possible to define an exact match of URI and location. If an exact match is found, the search terminates. For example, if a “/” request happens frequently, defining “location = /” will speed up the processing of these requests, as search terminates right after the first comparison. Such a location cannot obviously contain nested locations.

In versions from 0.7.1 to 0.8.41, if a request matched the prefix location without the “=” and “^~” modifiers, the search also terminated and regular expressions were not checked.

Let’s illustrate the above by an example:

location = / {

[ configuration A ]

}

location / {

[ configuration B ]

}

location /documents/ {

[ configuration C ]

}

location ^~ /images/ {

[ configuration D ]

}

location ~\* \.(gif|jpg|jpeg)$ {

[ configuration E ]

}

The “/” request will match configuration A, the “/index.html” request will match configuration B, the “/documents/document.html” request will match configuration C, the “/images/1.gif” request will match configuration D, and the “/documents/1.jpg” request will match configuration E.

The “@” prefix defines a named location. Such a location is not used for a regular request processing, but instead used for request redirection. They cannot be nested, and cannot contain nested locations.

If a location is defined by a prefix string that ends with the slash character, and requests are processed by one of [proxy\_pass](https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_pass), [fastcgi\_pass](https://nginx.org/en/docs/http/ngx_http_fastcgi_module.html#fastcgi_pass), [uwsgi\_pass](https://nginx.org/en/docs/http/ngx_http_uwsgi_module.html#uwsgi_pass), [scgi\_pass](https://nginx.org/en/docs/http/ngx_http_scgi_module.html#scgi_pass), [memcached\_pass](https://nginx.org/en/docs/http/ngx_http_memcached_module.html#memcached_pass), or [grpc\_pass](https://nginx.org/en/docs/http/ngx_http_grpc_module.html#grpc_pass), then the special processing is performed. In response to a request with URI equal to this string, but without the trailing slash, a permanent redirect with the code 301 will be returned to the requested URI with the slash appended. If this is not desired, an exact match of the URI and location could be defined like this:

location /user/ {

proxy\_pass http://user.example.com;

}

location = /user {

proxy\_pass http://login.example.com;

}

| Syntax: | **log\_not\_found** on | off; |
| --- | --- |
| Default: | log\_not\_found on; |
| Context: | http, server, location |

Enables or disables logging of errors about not found files into [error\_log](https://nginx.org/en/docs/ngx_core_module.html#error_log).

| Syntax: | **log\_subrequest** on | off; |
| --- | --- |
| Default: | log\_subrequest off; |
| Context: | http, server, location |

Enables or disables logging of subrequests into [access\_log](https://nginx.org/en/docs/http/ngx_http_log_module.html#access_log).

| Syntax: | **max\_ranges** *number*; |
| --- | --- |
| Default: | — |
| Context: | http, server, location |

This directive appeared in version 1.1.2.

Limits the maximum allowed number of ranges in byte-range requests. Requests that exceed the limit are processed as if there were no byte ranges specified. By default, the number of ranges is not limited. The zero value disables the byte-range support completely.

| Syntax: | **merge\_slashes** on | off; |
| --- | --- |
| Default: | merge\_slashes on; |
| Context: | http, server |

Enables or disables compression of two or more adjacent slashes in a URI into a single slash.

Note that compression is essential for the correct matching of prefix string and regular expression locations. Without it, the “//scripts/one.php” request would not match

location /scripts/ {

...

}

and might be processed as a static file. So it gets converted to “/scripts/one.php”.

Turning the compression off can become necessary if a URI contains base64-encoded names, since base64 uses the “/” character internally. However, for security considerations, it is better to avoid turning the compression off.

If the directive is specified on the [server](https://nginx.org/en/docs/http/ngx_http_core_module.html#server) level, its value is only used if a server is a default one. The value specified also applies to all virtual servers listening on the same address and port.

| Syntax: | **msie\_padding** on | off; |
| --- | --- |
| Default: | msie\_padding on; |
| Context: | http, server, location |

Enables or disables adding comments to responses for MSIE clients with status greater than 400 to increase the response size to 512 bytes.

| Syntax: | **msie\_refresh** on | off; |
| --- | --- |
| Default: | msie\_refresh off; |
| Context: | http, server, location |

Enables or disables issuing refreshes instead of redirects for MSIE clients.

| Syntax: | **open\_file\_cache** off; **open\_file\_cache** max=*N* [inactive=*time*]; |
| --- | --- |
| Default: | open\_file\_cache off; |
| Context: | http, server, location |

Configures a cache that can store:

* open file descriptors, their sizes and modification times;
* information on existence of directories;
* file lookup errors, such as “file not found”, “no read permission”, and so on.Caching of errors should be enabled separately by the [open\_file\_cache\_errors](https://nginx.org/en/docs/http/ngx_http_core_module.html#open_file_cache_errors) directive.

The directive has the following parameters:

max

sets the maximum number of elements in the cache; on cache overflow the least recently used (LRU) elements are removed;

inactive

defines a time after which an element is removed from the cache if it has not been accessed during this time; by default, it is 60 seconds;

off

disables the cache.

Example:

open\_file\_cache max=1000 inactive=20s;

open\_file\_cache\_valid 30s;

open\_file\_cache\_min\_uses 2;

open\_file\_cache\_errors on;

| Syntax: | **open\_file\_cache\_errors** on | off; |
| --- | --- |
| Default: | open\_file\_cache\_errors off; |
| Context: | http, server, location |

Enables or disables caching of file lookup errors by [open\_file\_cache](https://nginx.org/en/docs/http/ngx_http_core_module.html#open_file_cache).

| Syntax: | **open\_file\_cache\_min\_uses** *number*; |
| --- | --- |
| Default: | open\_file\_cache\_min\_uses 1; |
| Context: | http, server, location |

Sets the minimum *number* of file accesses during the period configured by the inactive parameter of the [open\_file\_cache](https://nginx.org/en/docs/http/ngx_http_core_module.html#open_file_cache) directive, required for a file descriptor to remain open in the cache.

| Syntax: | **open\_file\_cache\_valid** *time*; |
| --- | --- |
| Default: | open\_file\_cache\_valid 60s; |
| Context: | http, server, location |

Sets a time after which [open\_file\_cache](https://nginx.org/en/docs/http/ngx_http_core_module.html#open_file_cache) elements should be validated.

| Syntax: | **output\_buffers** *number* *size*; |
| --- | --- |
| Default: | output\_buffers 2 32k; |
| Context: | http, server, location |

Sets the *number* and *size* of the buffers used for reading a response from a disk.

Prior to version 1.9.5, the default value was 1 32k.

| Syntax: | **port\_in\_redirect** on | off; |
| --- | --- |
| Default: | port\_in\_redirect on; |
| Context: | http, server, location |

Enables or disables specifying the port in [absolute](https://nginx.org/en/docs/http/ngx_http_core_module.html#absolute_redirect) redirects issued by nginx.

The use of the primary server name in redirects is controlled by the [server\_name\_in\_redirect](https://nginx.org/en/docs/http/ngx_http_core_module.html#server_name_in_redirect) directive.

| Syntax: | **postpone\_output** *size*; |
| --- | --- |
| Default: | postpone\_output 1460; |
| Context: | http, server, location |

If possible, the transmission of client data will be postponed until nginx has at least *size* bytes of data to send. The zero value disables postponing data transmission.

| Syntax: | **read\_ahead** *size*; |
| --- | --- |
| Default: | read\_ahead 0; |
| Context: | http, server, location |

Sets the amount of pre-reading for the kernel when working with file.

On Linux, the posix\_fadvise(0, 0, 0, POSIX\_FADV\_SEQUENTIAL) system call is used, and so the *size* parameter is ignored.

On FreeBSD, the fcntl(O\_READAHEAD, *size*) system call, supported since FreeBSD 9.0-CURRENT, is used. FreeBSD 7 has to be [patched](http://sysoev.ru/freebsd/patch.readahead.txt).

| Syntax: | **recursive\_error\_pages** on | off; |
| --- | --- |
| Default: | recursive\_error\_pages off; |
| Context: | http, server, location |

Enables or disables doing several redirects using the [error\_page](https://nginx.org/en/docs/http/ngx_http_core_module.html#error_page) directive. The number of such redirects is [limited](https://nginx.org/en/docs/http/ngx_http_core_module.html#internal).

| Syntax: | **request\_pool\_size** *size*; |
| --- | --- |
| Default: | request\_pool\_size 4k; |
| Context: | http, server |

Allows accurate tuning of per-request memory allocations. This directive has minimal impact on performance and should not generally be used.

| Syntax: | **reset\_timedout\_connection** on | off; |
| --- | --- |
| Default: | reset\_timedout\_connection off; |
| Context: | http, server, location |

Enables or disables resetting timed out connections and connections [closed](https://nginx.org/en/docs/http/ngx_http_rewrite_module.html#return) with the non-standard code 444 (1.15.2). The reset is performed as follows. Before closing a socket, the SO\_LINGER option is set on it with a timeout value of 0. When the socket is closed, TCP RST is sent to the client, and all memory occupied by this socket is released. This helps avoid keeping an already closed socket with filled buffers in a FIN\_WAIT1 state for a long time.

It should be noted that timed out keep-alive connections are closed normally.

| Syntax: | **resolver** *address* ... [valid=*time*] [ipv6=on|off] [status\_zone=*zone*]; |
| --- | --- |
| Default: | — |
| Context: | http, server, location |

Configures name servers used to resolve names of upstream servers into addresses, for example:

resolver 127.0.0.1 [::1]:5353;

The address can be specified as a domain name or IP address, with an optional port (1.3.1, 1.2.2). If port is not specified, the port 53 is used. Name servers are queried in a round-robin fashion.

Before version 1.1.7, only a single name server could be configured. Specifying name servers using IPv6 addresses is supported starting from versions 1.3.1 and 1.2.2.

By default, nginx will look up both IPv4 and IPv6 addresses while resolving. If looking up of IPv6 addresses is not desired, the ipv6=off parameter can be specified.

Resolving of names into IPv6 addresses is supported starting from version 1.5.8.

By default, nginx caches answers using the TTL value of a response. An optional valid parameter allows overriding it:

resolver 127.0.0.1 [::1]:5353 valid=30s;

Before version 1.1.9, tuning of caching time was not possible, and nginx always cached answers for the duration of 5 minutes.

To prevent DNS spoofing, it is recommended configuring DNS servers in a properly secured trusted local network.

The optional status\_zone parameter (1.17.1) enables [collection](https://nginx.org/en/docs/http/ngx_http_api_module.html#resolvers_) of DNS server statistics of requests and responses in the specified *zone*. The parameter is available as part of our [commercial subscription](http://nginx.com/products/).

| Syntax: | **resolver\_timeout** *time*; |
| --- | --- |
| Default: | resolver\_timeout 30s; |
| Context: | http, server, location |

Sets a timeout for name resolution, for example:

resolver\_timeout 5s;

| Syntax: | **root** *path*; |
| --- | --- |
| Default: | root html; |
| Context: | http, server, location, if in location |

Sets the root directory for requests. For example, with the following configuration

location /i/ {

root /data/w3;

}

The /data/w3/i/top.gif file will be sent in response to the “/i/top.gif” request.

The *path* value can contain variables, except $document\_root and $realpath\_root.

A path to the file is constructed by merely adding a URI to the value of the root directive. If a URI has to be modified, the [alias](https://nginx.org/en/docs/http/ngx_http_core_module.html#alias) directive should be used.

| Syntax: | **satisfy** all | any; |
| --- | --- |
| Default: | satisfy all; |
| Context: | http, server, location |

Allows access if all (all) or at least one (any) of the [ngx\_http\_access\_module](https://nginx.org/en/docs/http/ngx_http_access_module.html), [ngx\_http\_auth\_basic\_module](https://nginx.org/en/docs/http/ngx_http_auth_basic_module.html), [ngx\_http\_auth\_request\_module](https://nginx.org/en/docs/http/ngx_http_auth_request_module.html), or [ngx\_http\_auth\_jwt\_module](https://nginx.org/en/docs/http/ngx_http_auth_jwt_module.html) modules allow access.

Example:

location / {

satisfy any;

allow 192.168.1.0/32;

deny all;

auth\_basic "closed site";

auth\_basic\_user\_file conf/htpasswd;

}

| Syntax: | **send\_lowat** *size*; |
| --- | --- |
| Default: | send\_lowat 0; |
| Context: | http, server, location |

If the directive is set to a non-zero value, nginx will try to minimize the number of send operations on client sockets by using either NOTE\_LOWAT flag of the [kqueue](https://nginx.org/en/docs/events.html#kqueue) method or the SO\_SNDLOWAT socket option. In both cases the specified *size* is used.

This directive is ignored on Linux, Solaris, and Windows.

| Syntax: | **send\_timeout** *time*; |
| --- | --- |
| Default: | send\_timeout 60s; |
| Context: | http, server, location |

Sets a timeout for transmitting a response to the client. The timeout is set only between two successive write operations, not for the transmission of the whole response. If the client does not receive anything within this time, the connection is closed.

| Syntax: | **sendfile** on | off; |
| --- | --- |
| Default: | sendfile off; |
| Context: | http, server, location, if in location |

Enables or disables the use of sendfile().

Starting from nginx 0.8.12 and FreeBSD 5.2.1, [aio](https://nginx.org/en/docs/http/ngx_http_core_module.html#aio) can be used to pre-load data for sendfile():

location /video/ {

sendfile on;

tcp\_nopush on;

aio on;

}

In this configuration, sendfile() is called with the SF\_NODISKIO flag which causes it not to block on disk I/O, but, instead, report back that the data are not in memory. nginx then initiates an asynchronous data load by reading one byte. On the first read, the FreeBSD kernel loads the first 128K bytes of a file into memory, although next reads will only load data in 16K chunks. This can be changed using the [read\_ahead](https://nginx.org/en/docs/http/ngx_http_core_module.html#read_ahead) directive.

Before version 1.7.11, pre-loading could be enabled with aio sendfile;.

| Syntax: | **sendfile\_max\_chunk** *size*; |
| --- | --- |
| Default: | sendfile\_max\_chunk 0; |
| Context: | http, server, location |

When set to a non-zero value, limits the amount of data that can be transferred in a single sendfile() call. Without the limit, one fast connection may seize the worker process entirely.

| Syntax: | **server** { ... } |
| --- | --- |
| Default: | — |
| Context: | http |

Sets configuration for a virtual server. There is no clear separation between IP-based (based on the IP address) and name-based (based on the “Host” request header field) virtual servers. Instead, the [listen](https://nginx.org/en/docs/http/ngx_http_core_module.html#listen) directives describe all addresses and ports that should accept connections for the server, and the [server\_name](https://nginx.org/en/docs/http/ngx_http_core_module.html#server_name) directive lists all server names. Example configurations are provided in the “[How nginx processes a request](https://nginx.org/en/docs/http/request_processing.html)” document.

| Syntax: | **server\_name** *name* ...; |
| --- | --- |
| Default: | server\_name ""; |
| Context: | server |

Sets names of a virtual server, for example:

server {

server\_name example.com www.example.com;

}

The first name becomes the primary server name.

Server names can include an asterisk (“\*”) replacing the first or last part of a name:

server {

server\_name example.com \*.example.com www.example.\*;

}

Such names are called wildcard names.

The first two of the names mentioned above can be combined in one:

server {

server\_name .example.com;

}

It is also possible to use regular expressions in server names, preceding the name with a tilde (“~”):

server {

server\_name www.example.com ~^www\d+\.example\.com$;

}

Regular expressions can contain captures (0.7.40) that can later be used in other directives:

server {

server\_name ~^(www\.)?(.+)$;

location / {

root /sites/$2;

}

}

server {

server\_name \_;

location / {

root /sites/default;

}

}

Named captures in regular expressions create variables (0.8.25) that can later be used in other directives:

server {

server\_name ~^(www\.)?(?<domain>.+)$;

location / {

root /sites/$domain;

}

}

server {

server\_name \_;

location / {

root /sites/default;

}

}

If the directive’s parameter is set to “$hostname” (0.9.4), the machine’s hostname is inserted.

It is also possible to specify an empty server name (0.7.11):

server {

server\_name www.example.com "";

}

It allows this server to process requests without the “Host” header field — instead of the default server — for the given address:port pair. This is the default setting.

Before 0.8.48, the machine’s hostname was used by default.

During searching for a virtual server by name, if the name matches more than one of the specified variants, (e.g. both a wildcard name and regular expression match), the first matching variant will be chosen, in the following order of priority:

1. the exact name
2. the longest wildcard name starting with an asterisk, e.g. “\*.example.com”
3. the longest wildcard name ending with an asterisk, e.g. “mail.\*”
4. the first matching regular expression (in order of appearance in the configuration file)

Detailed description of server names is provided in a separate [Server names](https://nginx.org/en/docs/http/server_names.html) document.

| Syntax: | **server\_name\_in\_redirect** on | off; |
| --- | --- |
| Default: | server\_name\_in\_redirect off; |
| Context: | http, server, location |

Enables or disables the use of the primary server name, specified by the [server\_name](https://nginx.org/en/docs/http/ngx_http_core_module.html#server_name) directive, in [absolute](https://nginx.org/en/docs/http/ngx_http_core_module.html#absolute_redirect) redirects issued by nginx. When the use of the primary server name is disabled, the name from the “Host” request header field is used. If this field is not present, the IP address of the server is used.

The use of a port in redirects is controlled by the [port\_in\_redirect](https://nginx.org/en/docs/http/ngx_http_core_module.html#port_in_redirect) directive.

| Syntax: | **server\_names\_hash\_bucket\_size** *size*; |
| --- | --- |
| Default: | server\_names\_hash\_bucket\_size 32|64|128; |
| Context: | http |

Sets the bucket size for the server names hash tables. The default value depends on the size of the processor’s cache line. The details of setting up hash tables are provided in a separate [document](https://nginx.org/en/docs/hash.html).

| Syntax: | **server\_names\_hash\_max\_size** *size*; |
| --- | --- |
| Default: | server\_names\_hash\_max\_size 512; |
| Context: | http |

Sets the maximum *size* of the server names hash tables. The details of setting up hash tables are provided in a separate [document](https://nginx.org/en/docs/hash.html).

| Syntax: | **server\_tokens** on | off | build | *string*; |
| --- | --- |
| Default: | server\_tokens on; |
| Context: | http, server, location |

Enables or disables emitting nginx version on error pages and in the “Server” response header field.

The build parameter (1.11.10) enables emitting a [build name](https://nginx.org/en/docs/configure.html#build) along with nginx version.

Additionally, as part of our [commercial subscription](http://nginx.com/products/), starting from version 1.9.13 the signature on error pages and the “Server” response header field value can be set explicitly using the *string* with variables. An empty string disables the emission of the “Server” field.

| Syntax: | **subrequest\_output\_buffer\_size** *size*; |
| --- | --- |
| Default: | subrequest\_output\_buffer\_size 4k|8k; |
| Context: | http, server, location |

This directive appeared in version 1.13.10.

Sets the *size* of the buffer used for storing the response body of a subrequest. By default, the buffer size is equal to one memory page. This is either 4K or 8K, depending on a platform. It can be made smaller, however.

The directive is applicable only for subrequests with response bodies saved into memory. For example, such subrequests are created by [SSI](https://nginx.org/en/docs/http/ngx_http_ssi_module.html#ssi_include_set).

| Syntax: | **tcp\_nodelay** on | off; |
| --- | --- |
| Default: | tcp\_nodelay on; |
| Context: | http, server, location |

Enables or disables the use of the TCP\_NODELAY option. The option is enabled when a connection is transitioned into the keep-alive state. Additionally, it is enabled on SSL connections, for unbuffered proxying, and for [WebSocket](https://nginx.org/en/docs/http/websocket.html) proxying.

| Syntax: | **tcp\_nopush** on | off; |
| --- | --- |
| Default: | tcp\_nopush off; |
| Context: | http, server, location |

Enables or disables the use of the TCP\_NOPUSH socket option on FreeBSD or the TCP\_CORK socket option on Linux. The options are enabled only when [sendfile](https://nginx.org/en/docs/http/ngx_http_core_module.html#sendfile) is used. Enabling the option allows

* sending the response header and the beginning of a file in one packet, on Linux and FreeBSD 4.\*;
* sending a file in full packets.

| Syntax: | **try\_files** *file* ... *uri*; **try\_files** *file* ... =*code*; |
| --- | --- |
| Default: | — |
| Context: | server, location |

Checks the existence of files in the specified order and uses the first found file for request processing; the processing is performed in the current context. The path to a file is constructed from the *file* parameter according to the [root](https://nginx.org/en/docs/http/ngx_http_core_module.html#root) and [alias](https://nginx.org/en/docs/http/ngx_http_core_module.html#alias) directives. It is possible to check directory’s existence by specifying a slash at the end of a name, e.g. “$uri/”. If none of the files were found, an internal redirect to the *uri* specified in the last parameter is made. For example:

location /images/ {

try\_files $uri /images/default.gif;

}

location = /images/default.gif {

expires 30s;

}

The last parameter can also point to a named location, as shown in examples below. Starting from version 0.7.51, the last parameter can also be a *code*:

location / {

try\_files $uri $uri/index.html $uri.html =404;

}

Example in proxying Mongrel:

location / {

try\_files /system/maintenance.html

$uri $uri/index.html $uri.html

@mongrel;

}

location @mongrel {

proxy\_pass http://mongrel;

}

Example for Drupal/FastCGI:

location / {

try\_files $uri $uri/ @drupal;

}

location ~ \.php$ {

try\_files $uri @drupal;

fastcgi\_pass ...;

fastcgi\_param SCRIPT\_FILENAME /path/to$fastcgi\_script\_name;

fastcgi\_param SCRIPT\_NAME $fastcgi\_script\_name;

fastcgi\_param QUERY\_STRING $args;

... other fastcgi\_param's

}

location @drupal {

fastcgi\_pass ...;

fastcgi\_param SCRIPT\_FILENAME /path/to/index.php;

fastcgi\_param SCRIPT\_NAME /index.php;

fastcgi\_param QUERY\_STRING q=$uri&$args;

... other fastcgi\_param's

}

In the following example,

location / {

try\_files $uri $uri/ @drupal;

}

the try\_files directive is equivalent to

location / {

error\_page 404 = @drupal;

log\_not\_found off;

}

And here,

location ~ \.php$ {

try\_files $uri @drupal;

fastcgi\_pass ...;

fastcgi\_param SCRIPT\_FILENAME /path/to$fastcgi\_script\_name;

...

}

try\_files checks the existence of the PHP file before passing the request to the FastCGI server.

Example for Wordpress and Joomla:

location / {

try\_files $uri $uri/ @wordpress;

}

location ~ \.php$ {

try\_files $uri @wordpress;

fastcgi\_pass ...;

fastcgi\_param SCRIPT\_FILENAME /path/to$fastcgi\_script\_name;

... other fastcgi\_param's

}

location @wordpress {

fastcgi\_pass ...;

fastcgi\_param SCRIPT\_FILENAME /path/to/index.php;

... other fastcgi\_param's

}

| Syntax: | **types** { ... } |
| --- | --- |
| Default: | types {  text/html html;  image/gif gif;  image/jpeg jpg;  } |
| Context: | http, server, location |

Maps file name extensions to MIME types of responses. Extensions are case-insensitive. Several extensions can be mapped to one type, for example:

types {

application/octet-stream bin exe dll;

application/octet-stream deb;

application/octet-stream dmg;

}

A sufficiently full mapping table is distributed with nginx in the conf/mime.types file.

To make a particular location emit the “application/octet-stream” MIME type for all requests, the following configuration can be used:

location /download/ {

types { }

default\_type application/octet-stream;

}

| Syntax: | **types\_hash\_bucket\_size** *size*; |
| --- | --- |
| Default: | types\_hash\_bucket\_size 64; |
| Context: | http, server, location |

Sets the bucket size for the types hash tables. The details of setting up hash tables are provided in a separate [document](https://nginx.org/en/docs/hash.html).

Prior to version 1.5.13, the default value depended on the size of the processor’s cache line.

| Syntax: | **types\_hash\_max\_size** *size*; |
| --- | --- |
| Default: | types\_hash\_max\_size 1024; |
| Context: | http, server, location |

Sets the maximum *size* of the types hash tables. The details of setting up hash tables are provided in a separate [document](https://nginx.org/en/docs/hash.html).

| Syntax: | **underscores\_in\_headers** on | off; |
| --- | --- |
| Default: | underscores\_in\_headers off; |
| Context: | http, server |

Enables or disables the use of underscores in client request header fields. When the use of underscores is disabled, request header fields whose names contain underscores are marked as invalid and become subject to the [ignore\_invalid\_headers](https://nginx.org/en/docs/http/ngx_http_core_module.html#ignore_invalid_headers) directive.

If the directive is specified on the [server](https://nginx.org/en/docs/http/ngx_http_core_module.html#server) level, its value is only used if a server is a default one. The value specified also applies to all virtual servers listening on the same address and port.

| Syntax: | **variables\_hash\_bucket\_size** *size*; |
| --- | --- |
| Default: | variables\_hash\_bucket\_size 64; |
| Context: | http |

Sets the bucket size for the variables hash table. The details of setting up hash tables are provided in a separate [document](https://nginx.org/en/docs/hash.html).

| Syntax: | **variables\_hash\_max\_size** *size*; |
| --- | --- |
| Default: | variables\_hash\_max\_size 1024; |
| Context: | http |

Sets the maximum *size* of the variables hash table. The details of setting up hash tables are provided in a separate [document](https://nginx.org/en/docs/hash.html).

Prior to version 1.5.13, the default value was 512.

#### Embedded Variables

The ngx\_http\_core\_module module supports embedded variables with names matching the Apache Server variables. First of all, these are variables representing client request header fields, such as $http\_user\_agent, $http\_cookie, and so on. Also there are other variables:

$arg\_*name*

argument *name* in the request line

$args

arguments in the request line

$binary\_remote\_addr

client address in a binary form, value’s length is always 4 bytes for IPv4 addresses or 16 bytes for IPv6 addresses

$body\_bytes\_sent

number of bytes sent to a client, not counting the response header; this variable is compatible with the “%B” parameter of the mod\_log\_config Apache module

$bytes\_sent

number of bytes sent to a client (1.3.8, 1.2.5)

$connection

connection serial number (1.3.8, 1.2.5)

$connection\_requests

current number of requests made through a connection (1.3.8, 1.2.5)

$connection\_time

connection time in seconds with a milliseconds resolution (1.19.10)

$content\_length

“Content-Length” request header field

$content\_type

“Content-Type” request header field

$cookie\_*name*

the *name* cookie

$document\_root

[root](https://nginx.org/en/docs/http/ngx_http_core_module.html#root) or [alias](https://nginx.org/en/docs/http/ngx_http_core_module.html#alias) directive’s value for the current request

$document\_uri

same as $uri

$host

in this order of precedence: host name from the request line, or host name from the “Host” request header field, or the server name matching a request

$hostname

host name

$http\_*name*

arbitrary request header field; the last part of a variable name is the field name converted to lower case with dashes replaced by underscores

$https

“on” if connection operates in SSL mode, or an empty string otherwise

$is\_args

“?” if a request line has arguments, or an empty string otherwise

$limit\_rate

setting this variable enables response rate limiting; see [limit\_rate](https://nginx.org/en/docs/http/ngx_http_core_module.html#limit_rate)

$msec

current time in seconds with the milliseconds resolution (1.3.9, 1.2.6)

$nginx\_version

nginx version

$pid

PID of the worker process

$pipe

“p” if request was pipelined, “.” otherwise (1.3.12, 1.2.7)

$proxy\_protocol\_addr

client address from the PROXY protocol header (1.5.12)

The PROXY protocol must be previously enabled by setting the proxy\_protocol parameter in the [listen](https://nginx.org/en/docs/http/ngx_http_core_module.html#listen) directive.

$proxy\_protocol\_port

client port from the PROXY protocol header (1.11.0)

The PROXY protocol must be previously enabled by setting the proxy\_protocol parameter in the [listen](https://nginx.org/en/docs/http/ngx_http_core_module.html#listen) directive.

$proxy\_protocol\_server\_addr

server address from the PROXY protocol header (1.17.6)

The PROXY protocol must be previously enabled by setting the proxy\_protocol parameter in the [listen](https://nginx.org/en/docs/http/ngx_http_core_module.html#listen) directive.

$proxy\_protocol\_server\_port

server port from the PROXY protocol header (1.17.6)

The PROXY protocol must be previously enabled by setting the proxy\_protocol parameter in the [listen](https://nginx.org/en/docs/http/ngx_http_core_module.html#listen) directive.

$query\_string

same as $args

$realpath\_root

an absolute pathname corresponding to the [root](https://nginx.org/en/docs/http/ngx_http_core_module.html#root) or [alias](https://nginx.org/en/docs/http/ngx_http_core_module.html#alias) directive’s value for the current request, with all symbolic links resolved to real paths

$remote\_addr

client address

$remote\_port

client port

$remote\_user

user name supplied with the Basic authentication

$request

full original request line

$request\_body

request body

The variable’s value is made available in locations processed by the [proxy\_pass](https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_pass), [fastcgi\_pass](https://nginx.org/en/docs/http/ngx_http_fastcgi_module.html#fastcgi_pass), [uwsgi\_pass](https://nginx.org/en/docs/http/ngx_http_uwsgi_module.html#uwsgi_pass), and [scgi\_pass](https://nginx.org/en/docs/http/ngx_http_scgi_module.html#scgi_pass) directives when the request body was read to a [memory buffer](https://nginx.org/en/docs/http/ngx_http_core_module.html#client_body_buffer_size).

$request\_body\_file

name of a temporary file with the request body

At the end of processing, the file needs to be removed. To always write the request body to a file, [client\_body\_in\_file\_only](https://nginx.org/en/docs/http/ngx_http_core_module.html#client_body_in_file_only) needs to be enabled. When the name of a temporary file is passed in a proxied request or in a request to a FastCGI/uwsgi/SCGI server, passing the request body should be disabled by the [proxy\_pass\_request\_body off](https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_pass_request_body), [fastcgi\_pass\_request\_body off](https://nginx.org/en/docs/http/ngx_http_fastcgi_module.html#fastcgi_pass_request_body), [uwsgi\_pass\_request\_body off](https://nginx.org/en/docs/http/ngx_http_uwsgi_module.html#uwsgi_pass_request_body), or [scgi\_pass\_request\_body off](https://nginx.org/en/docs/http/ngx_http_scgi_module.html#scgi_pass_request_body) directives, respectively.

$request\_completion

“OK” if a request has completed, or an empty string otherwise

$request\_filename

file path for the current request, based on the [root](https://nginx.org/en/docs/http/ngx_http_core_module.html#root) or [alias](https://nginx.org/en/docs/http/ngx_http_core_module.html#alias) directives, and the request URI

$request\_id

unique request identifier generated from 16 random bytes, in hexadecimal (1.11.0)

$request\_length

request length (including request line, header, and request body) (1.3.12, 1.2.7)

$request\_method

request method, usually “GET” or “POST”

$request\_time

request processing time in seconds with a milliseconds resolution (1.3.9, 1.2.6); time elapsed since the first bytes were read from the client

$request\_uri

full original request URI (with arguments)

$scheme

request scheme, “http” or “https”

$sent\_http\_*name*

arbitrary response header field; the last part of a variable name is the field name converted to lower case with dashes replaced by underscores

$sent\_trailer\_*name*

arbitrary field sent at the end of the response (1.13.2); the last part of a variable name is the field name converted to lower case with dashes replaced by underscores

$server\_addr

an address of the server which accepted a request

Computing a value of this variable usually requires one system call. To avoid a system call, the [listen](https://nginx.org/en/docs/http/ngx_http_core_module.html#listen) directives must specify addresses and use the bind parameter.

$server\_name

name of the server which accepted a request

$server\_port

port of the server which accepted a request

$server\_protocol

request protocol, usually “HTTP/1.0”, “HTTP/1.1”, or “[HTTP/2.0](https://nginx.org/en/docs/http/ngx_http_v2_module.html)”

$status

response status (1.3.2, 1.2.2)

$tcpinfo\_rtt, $tcpinfo\_rttvar, $tcpinfo\_snd\_cwnd, $tcpinfo\_rcv\_space

information about the client TCP connection; available on systems that support the TCP\_INFO socket option

$time\_iso8601

local time in the ISO 8601 standard format (1.3.12, 1.2.7)

$time\_local

local time in the Common Log Format (1.3.12, 1.2.7)

$uri

current URI in request, [normalized](https://nginx.org/en/docs/http/ngx_http_core_module.html#location)

The value of $uri may change during request processing, e.g. when doing internal redirects, or when using index files.