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