

NGINX

Overview

Avinash Thakur | Nginx Overview | 5th Sep 2017

Table of Contents

[NGINX 2](#_Toc508274737)

[HOW NGINX WORKS 3](#_Toc508274738)

[FILE STRUCTURE CONFIGURATION 3](#_Toc508274739)

[PROXY SERVER 3](#_Toc508274740)

[REVERSE PROXY 3](#_Toc508274741)

[Passing a Request to a Proxied Server 4](#_Toc508274742)

[PASSING Request Headers 4](#_Toc508274743)

[OPTIMIZING NGINX FOR HIGH TRAFFIC LOADS 5](#_Toc508274744)

[Worker Processes 5](#_Toc508274745)

[Worker Connections 5](#_Toc508274746)

[Keep alive 5](#_Toc508274747)

[File Descriptors 6](#_Toc508274748)

[Nginx Commands: 6](#_Toc508274749)

[Helpful URLS: 6](#_Toc508274750)

# NGINX

NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server.

Unlike traditional servers, NGINX doesn’t rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but more importantly, predictable amounts of memory under load. Even if you don’t expect to handle thousands of simultaneous requests, you can still benefit from NGINX’s high-performance and small memory footprint.

Nginx has one master process and several worker processes. The main purpose of the master process is to read and evaluate configuration, and maintain worker processes. Worker processes do actual processing of requests. Nginx employs event-based model and OS-dependent mechanisms to efficiently distribute requests among worker processes. The number of worker processes is defined in the configuration file and may be fixed for a given configuration or automatically adjusted to the number of available CPU cores.

# HOW NGINX WORKS

Nginx has one master process and several worker processes. The main purpose of the master process is to read and evaluate configuration, and maintain worker processes. Worker processes do actual processing of requests. Nginx employs event-based model and OS-dependent mechanisms to efficiently distribute requests among worker processes. The number of worker processes is defined in the configuration file and may be fixed for a given configuration or automatically adjusted to the number of available CPU cores.

The way nginx and its modules work is determined in the configuration file. By default, the configuration file is named *nginx.conf.*

# FILE STRUCTURE CONFIGURATION

Nginx consists of modules which are controlled by directives specified in the configuration file. Directives are divided into *simple directives* and *block directives*.

A *simple directive* consists of the name and parameters separated by spaces and ends with a semicolon (;).

A *block directive* has the same structure as a simple directive, but instead of the semicolon it ends with a set of additional instructions surrounded by braces ({ and }).

If a *block directive* can have other directives inside braces, it is called a *context* (examples: [events](https://nginx.org/en/docs/ngx_core_module.html#events), [http](https://nginx.org/en/docs/http/ngx_http_core_module.html#http), [server](https://nginx.org/en/docs/http/ngx_http_core_module.html#server), and [location](https://nginx.org/en/docs/http/ngx_http_core_module.html#location)).

Directives placed in the configuration file outside of any contexts are considered to be in the [main](https://nginx.org/en/docs/ngx_core_module.html) context. The events and http directives reside in the main context, server in http, and location in server. The rest of a line after the # sign is considered a comment.

# PROXY SERVER

One of the frequent uses of nginx is setting it up as a proxy server, which means a server that receives requests, passes them to the proxied servers, retrieves responses from them, and sends them to the clients.

# REVERSE PROXY

Proxying is typically used to distribute the load among several servers, seamlessly show content from different websites, or pass requests for processing to application servers over protocols other than HTTP.

## Passing a Request to a Proxied Server

When NGINX proxies a request, it sends the request to a specified proxied server, fetches the response, and sends it back to the client. It is possible to proxy requests to an HTTP server (another NGINX server or any other server) or a non-HTTP server (which can run an application developed with a specific framework, such as PHP or Python) using a specified protocol. Supported protocols include **FastCGI**, **uwsgi**, [**SCGI**](http://nginx.org/en/docs/http/ngx_http_scgi_module.html), and **memcached**.

To pass a request to an HTTP proxied server, the proxy\_pass directive is specified inside a location. For example:

|  |
| --- |
| location /some/path/ {  proxy\_pass http://www.example.com/link/;  } |

Note: To pass a request to a non-HTTP proxied server, the appropriate \*\*\_pass directive should be used:

1. [fastcgi\_pass](http://nginx.org/en/docs/http/ngx_http_fastcgi_module.html#fastcgi_pass) -> passes a request to a FastCGI server
2. [uwsgi\_pass](http://nginx.org/en/docs/http/ngx_http_uwsgi_module.html#uwsgi_pass) -> passes a request to a uwsgi server
3. [scgi\_pass](http://nginx.org/en/docs/http/ngx_http_scgi_module.html#scgi_pass) -> passes a request to an SCGI server
4. [memcached\_pass](http://nginx.org/en/docs/http/ngx_http_memcached_module.html#memcached_pass)-> passes a request to a memcached server

## PASSING Request Headers

By default, NGINX redefines two header fields in proxied requests, “Host” and “Connection”, and eliminates the header fields whose values are empty strings. “Host” is set to the **$proxy\_host**variable, and “**Connection**” is set to close.

To change these setting, as well as modify other header fields, use the [**proxy\_set\_header**](http://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_set_header) directive. This directive can be specified in a **location** or higher. It can also be specified in a particular [**server**](http://nginx.org/en/docs/http/ngx_http_core_module.html#server) context or in the [**http**](http://nginx.org/en/docs/http/ngx_http_core_module.html#http) block. For example:

|  |
| --- |
| location /some/path/ {  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_pass http://localhost:8000;  } |

# [OPTIMIZING NGINX FOR HIGH TRAFFIC LOADS](http://blog.martinfjordvald.com/2011/04/optimizing-nginx-for-high-traffic-loads/)

## 

## Worker Processes

The worker process is the backbone of nginx, once the master has bound to the required IP/ports it will spawn workers as the specified user and they’ll then handle all the work. Workers are not multi-threaded so they do not spread the per-connection across CPU cores. Thus it makes sense for us to run multiple workers, usually 1 worker per CPU core. For most workloads anything above 2-4 workers is overkill as nginx will hit other bottlenecks before the CPU becomes an issue and usually you’ll just have idle processes.

Command: grep processor /proc/cpuinfo | wc –l;

No of worker process = output \*1;

## Worker Connections

[Worker connections](https://www.nginx.com/resources/wiki/) effectively limits how many connections each worker can maintain at a time.

Command: ulimit –n;

## [Keep alive](https://www.nginx.com/resources/wiki/)

[Keep alive](https://www.nginx.com/resources/wiki/) is a HTTP feature which allows user agents to keep the connection to your server open for a number of requests or until the specified time out is reached.

## File Descriptors

File descriptors are operating system resources used to handle things such as connections and open files. nginx can use up to two file descriptors per connection. For example, if it is proxying, there is generally one file descriptor for the client connection and another for the connection to the proxied server, though this ratio is much lower if HTTP keep alives are used. For a system serving a large number of connections, these settings may need to be adjusted.

# Nginx Commands:

1. To stop your web server, you can type

sudo service nginx stop

1. To start your web server when it is stopped, you can type

sudo service nginx start

1. To stop and start your web server, you can type

sudo service nginx restart

1. We can make sure that our web server will restart automatically when the server is rebooted, you can type

sudo update-rc.d nginx default

1. To locate the nginx config, you can type

/etc/nginx/nginx.conf

# Nginx Configuration Directory Hierarchy

Nginx stores its configuration files within the "/etc/nginx" directory.

# Nginx Configuration Reverse proxy header read

1. Read header value in nginx.

If ( $http\_Authorization = ‘test’) {

Proxy\_pass http://google.com

}

# Helpful URLS:

<http://blog.martinfjordvald.com/2011/04/optimizing-nginx-for-high-traffic-loads/>

<https://www.digitalocean.com/community/tutorials/how-to-optimize-nginx-configuration>