FREE eBook

LEARNING nginx

Free unaffiliated eBook created from
Stack Overflow contributors.

#nginx

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: nginx

It is an unofficial and free nginx ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official nginx.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with nginx

Remarks

NGINX is pronounced like "engine x" and is commonly used as a high performance server for protocols HTTP, HTTPS, SMTP, POP3 and IMAP. It can be used as a reverse proxy server, HTTP cache or load balancing.

It is an open source project with source available here.

Versions

Version	Original release date	Latest version	Status	Release Date
0.5	2006-12-04	0.5.38	Legacy	2009-09-14
0.6	2007-06-14	0.6.39	Legacy	2009-09-14
0.7	2008-05-19	0.7.69	Legacy	2011-07-19
0.8	2009-06-02	0.8.55	Legacy	2011-07-19
1.0	2011-04-12	1.0.15	Legacy	2012-04-12
1.2	2012-04-23	1.2.9	Legacy	2013-05-13
1.4	2013-04-24	1.4.7	Legacy	2014-03-18
1.6	2014-04-24	1.6.3	Legacy	2015-04-07
1.8	2015-04-21	1.8.1	Legacy	2016-01-26
1.9	2015-04-28	1.9.15	Legacy	2016-04-19
1.10	2016-04-26	1.10.3	Stable	2016-05-31
1.11	2016-05-24	1.11.9	Mainline	2016-07-26

Examples

Installation and setup

Nginx is a Web server used to serve HTTP requests over the Internet.

Nginx is available on Linux, Windows and other OSes as direct download, and can also be built from source. For detailed instructions see Nginx official reference.

ubuntu/debian

nginx stable version is available in official repo, it can be installed using

```
sudo apt-get install nginx
```

It will install and configure system startup files, but if you need latest version, you may need to add official ppa.

```
sudo add-apt-repository ppa:nginx/stable
sudo apt-get update
sudo apt-get install nginx
```

above instructions will install latest stable edition.

Nginx installation on Debian and Debian-based distros like Ubuntu

Run command below to install nginx.

```
sudo apt-get install nginx
```

By default, Nginx automatically starts when it is installed. You can access the default Nginx landing page to confirm that the software is running properly by visiting your server's domain name or public IP address in your web browser.

but if you need latest version, you may need to add official ppa.

```
sudo add-apt-repository ppa:nginx/stable
sudo apt-get update
sudo apt-get install nginx
```

Restart NGINX

As a root user:

```
nginx -s restart
```

Ubuntu example

```
sudo service nginx restart
```

Reload NGINX configuration file

As a root user:

```
sudo nginx -s reload
```

Ubuntu 14.04 example

```
sudo service nginx reload
```

Ubuntu 16.04 example

```
sudo systemctl reload nginx
```

Before reloading, it is a good idea to check config for syntax errors:

```
sudo nginx –t
```

Or

```
sudo service nginx configtest
```

Shutdown NGINX

Run as a root user.

Fast shutdown:

```
nginx –s stop
```

Graceful shutdown:

```
nginx –s quit
```

Nginx inside

One of the biggest appeals of Nginx is the difference in how it works internally as compared to the other popular servers, specially Apache.

Servers are busy programs as they have to serve requests from multiple clients. The more requests a server can successfully serve per second, the better.

Nginx works on a concurrency paradigm known as Asynchronous IO.

In a conventional server, one thread is dedicated to one request. This means, once a thread takes up a request, it is effectively unavailable for other requests. But in reality, a thread could do a lot better by accepting a bunch of requests and serving them simultaneously. Asynchronous IO is what enables this.

Nginx, therefore with its Asynchronous IO architecture, can serve many requests within one thread.

Another good thing about Nginx is its relatively leaner resource footprint. Compared to Apache, Nginx is less resource heavy, and this makes it suitable to cloud servers what tend not to be very powerful.

There are certainly other Async IO server out there, but Nginx is the most well supported among all in terms of pluginx (aka Nginx Modules).

Test if your changes in nginx.config are valid

Ubuntu 14.04 example

```
sudo nginx -t
```

Read Getting started with nginx online: https://riptutorial.com/nginx/topic/1121/getting-started-with-nginx

Chapter 2: Logging

Examples

Basic example

Syntax

```
Syntax:  log_format name string ...;
Syntax:  access_log path [format [buffer=size] [gzip[=level]] [flush=time] [if=condition]];
access_log off;
```

Using them together

```
log_format compression '$remote_addr - $remote_user [$time_local] '
                       '"$request" $status $bytes_sent '
                       '"$http_referer" "$http_user_agent" "$gzip_ratio"';

access_log /spool/logs/nginx-access.log compression buffer=32k;
error_log /spool/logs/nginx-error.log;
```

Reopen log files

As a root user run:

```
nginx -s reopen
```

Avoid logging for favicon.ico and robots.txt

```
location = /favicon.ico {
    log_not_found off;
    access_log off;
}

location = /robots.txt {
    allow all;
    log_not_found off;
    access_log off;
}
```

Read Logging online: https://riptutorial.com/nginx/topic/2551/logging

Chapter 3: Nginx Configurations

Examples

Configuration File's Structure

nginx consists of modules which are controlled by directives specified in the configuration file.

Simple Directives

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

Block Directive

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 }).

Context

If a block directive can have other directives inside braces, it is called a context (examples: events, http, server, and location).

Directives placed in the configuration file outside of any contexts are considered to be in the main context. The events and http directives reside in the main context, server in http, and location in server.

Comment

The rest of a line after the # sign is considered a comment.

Test NGINX configuration file

You can check for syntax errors and referenced files in an NGINX configuration file before running it with:

```
nginx -t
```

Alternatively you can run service script

```
service nginx configtest
```

While both of these commands will tell you if your new nginx configuration is ok [without killing your current instance]. Configtest uses the running service and tells you if it passes or fails the check whereas nginx -t will not only check the config but print any info, warning as well as error messages.

Source: http://devget.net/nginxapache/nginx-configtest-vs-nginx-t/

Specify configuration file to load

```
nginx -c <file name>
```

Start NGINX with an explicit configuration file.

config changes without restart

```
nginx -s reload
```

Logging to Syslog, incl. mapping of HTTP-Codes to Syslog severities.

Paste this snippet somewhere in the `http {}` Block; or place it in it's own file in the `/etc/nginx/conf.d/` folder. Also see the official docs for logging to syslog.

```
#
# Access Log
#
log_format fmt_syslog '[$time_local] $status $remote_addr $http_host "$request"
$body_bytes_sent $request_time "$http_user_agent" $remote_user';
map $status $log_is_error { "~^5\d\d"     1; default 0; }
map $status $log_is_warn  { "~^4[0-8]{2}" 1; default 0; }
map $status $log_is_info  { "~^[1-3]\d\d" 1; default 0; }
access_log
syslog:server=unix:/run/systemd/journal/syslog,nohostname,facility=local2,severity=error
fmt_syslog if=$log_is_error;
access_log
syslog:server=unix:/run/systemd/journal/syslog,nohostname,facility=local2,severity=warn
fmt_syslog if=$log_is_warn;
access_log
syslog:server=unix:/run/systemd/journal/syslog,nohostname,facility=local2,severity=info
fmt_syslog if=$log_is_info;
#
# Error Log
#
error_log syslog:server=unix:/run/systemd/journal/syslog,nohostname,facility=local2 error;
```

This example assumes rsyslog (or similar) is listening on Socket `/run/systemd/journal/syslog` - as it's default on Debian 8 when journald has activated ForwardToSyslog. Using this socket, you bypass journald. If that socket is not available try `/dev/log` instead.

Feel free to use another facility instead of local2. You may also change the log_format to suit your needs.

Limit request methods

A usual Website just needs 3 HTTP Methods: `GET`, `HEAD` and `POST`. Block all other Methods by using limit_except:

```
location / {
    [...]
    # Note: GET includes HEAD
    limit_except GET POST {
        deny  all;
    }
    [...]
}
```

Read Nginx Configurations online: https://riptutorial.com/nginx/topic/2550/nginx-configurations

Chapter 4: nginx reverse proxy

Examples

simple reverse proxy

```
# Define which servers to include in the load balancing scheme.
# It's best to use the servers' private IPs for better performance and security.

upstream backend {

    ip_hash;
    server 10.10.10.10 slow_start=30s max_fails=3 fail_timeout=15s;
    server 10.10.10.12 slow_start=30s max_fails=3 fail_timeout=15s;

    # Activates the cache for connections to upstream servers.
    keepalive 20;
}

# This server accepts all traffic to port 80 and passes it to the upstream.
# Notice that the upstream name and the proxy_pass need to match.

server {
    listen 80;
    server_name example.com;

    location / {
        proxy_pass http://backend/;
    }
}
```

Read nginx reverse proxy online: https://riptutorial.com/nginx/topic/7431/nginx-reverse-proxy

Chapter 5: Useful Redirects

Examples

HTTPS Redirect

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name example.com www.example.com;
    return 307 https://$host$request_uri;
}
```

A 301 redirect is also an alternative but if a POST is made to a 301, then many clients will resubmit the request as a GET - which almost always will fail serverside. A 307 mandates the same request type.

Redirect requests to another server

```
server {
   server_name example.com;
   return 301 $scheme://example.net$request_uri;
}
```

Mobile Site Redirection

Nginx configuration to detect request from mobile user-agent and redirect them to mobile site.

```
location / {

    #mobile site handling as per user agent
    set $mobile_rewrite do_not_perform;     // variable to store action. default set to not
perform redirection to mobile site.

    if ($http_user_agent ~*
"(android|bb\d+|meego).+mobile|avantgo|bada\/|blackberry|blazer|compal|elaine|fennec|hiptop|iemobile|ip
|maemo|midp|mmp|mobile.+firefox|netfront|opera m(ob|in)i|palm(
os)?|phone|p(ixi|re)\/|plucker|pocket|psp|series(4|6)0|symbian|treo|up\.(browser|link)|vodafone|wap|win
ce|xda|xiino") {
        set $mobile_rewrite perform;
    }

    if ($http_user_agent ~* "^(1207|6310|6590|3gso|4thp|50[1-6]i|770s|802s|a
wa|abac|ac(er|oo|s\-
)|ai(ko|rn)|al(av|ca|co)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|attw|au(di|\-m|r |s
)|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-(n|u)|c55\/|capi|ccwa|cdm\-
|cell|chtm|cldc|cmd\-|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-s|devi|dica|dmob|do(c|p)o|ds(12|\-
d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-7]0|os|wa|ze)|fetc|fly(\-|_)|g1 u|g560|gene|gf\-
5|g\-mo|go(\.w|od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-|hi(pt|ta)|hp( i|ip)|hs\-c|ht(c(\-|
|_|a|g|p|s|t)|tp)|hu(aw|tc)|i\-(20|go|ma)|i230|iac( |\-
|\/)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kgt(
```

```
|\/)|klon|kpt |kwc\-|kyo(c|k)|le(no|xi)|lg( g|\/(k|l|u)|50|54|\-[a-w])|libw|lynx|m1\-
w|m3ga|m50\/|ma(te|ui|xo)|mc(01|21|ca)|m\-
cr|me(rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\-| |o|v)|zz)|mt(50|p1|v )|mwbp|mywa|n10[0-
2]|n20[2-3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\-
|on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(13|\-([1-
8]|c))|phil|pire|pl(ay|uc)|pn\-2|po(ck|rt|se)|prox|psio|pt\-g|qa\-a|qc(07|12|21|32|60|\-[2-
7]|i\-)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\/|sa(ge|ma|mm|ms|ny|va)|sc(01|h\-|oo|p\-
)|sdk\/|se(c(\-|0|1)|47|mc|nd|ri)|sgh\-|shar|sie(\-|m)|sk\-
0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\-|v\-|v
)|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\-|tdg\-|tel(i|m)|tim\-|t\-
mo|to(pl|sh)|ts(70|m\-|m3|m5)|tx\-9|up(\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-
3]|\-v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\-| )|webc|whit|wi(g
|nc|nw)|wmlb|wonu|x700|yas\-|your|zeto|zte\-)") {
        set $mobile_rewrite perform;
    }

    #google bot mobile handling
    if ($http_user_agent ~* "(googlebot-mobile)") {
        set $mobile_rewrite perform;
    }

    if ($mobile_rewrite = perform) {
        proxy_pass http://www.mobile-domain.com:$port;
    }
}
```

HTTP location on HTTPS server

HTTPS server with http location:

```
server {
    listen 443;
    root /var/www/
    location / {
       ...
    }
    location /http {
        rewrite ^ http://$host$request_uri? permanent;
    }
}
```

HTTP server redirects to HTTPS except one location:

```
server {
    root /var/www/
    location / {
        rewrite ^ https://$host$request_uri? permanent;
    }
    location /http {
       ...
    }
}
```

Disallow Requests Based on Host or Country Code

Use Nginx map to parse fields and reject requests.

```
# Allowed hosts
map $http_host $name {
    hostnames;

    default       no;

    example.com   yes;
    *.example.com yes;
    example.org   yes;
    *.example.org yes;
    .example.net  yes;
    wap.*         yes;
}

# Allowed countries
map $geoip_country_code $allowed_country {
    default no;
    country_code_1 yes;
    country_code_2 yes;
}
```

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # Disallow access based on hostname
    if ($name = no) {
        return 444;
    }

    # Disallow access based on GeoIP
    if ($allowed_country = no) {
        return 444;
    }
...
```

Read Useful Redirects online: https://riptutorial.com/nginx/topic/3631/useful-redirects

Chapter 6: Using nginx to provide clean browser URLs

Examples

Redirect vs reverse proxy

Professionally made web applications don't expose the internal details of the server environment to the user. When you place an order at your retailer, you don't see (or have to type) https://mydealer.com:8443/Dealerapp/entryPage.html, but just mydealer.com, although the app server needs all the details given in the longer URL. This can be achieved via:

```
if ($scheme = http) {
    return 301 https://$server_name$request_uri;
}
```

This does a **redirect**. Even if the client didn't request a secure connection, the browser is at once redirected to it. In countries having strict data privacy laws, you might even be obliged to do this for any commercial site. The redirect way is taken because here the browser needs to know about the secure connection, otherwise it wouldn't negotiate with the server to make it secure.

```
location /app/ {
    proxy_pass https://mydealer.com:8443/Dealerapp/entryPage.html;
}
```

This is a **reverse proxy**. It is transparent to the browser. So you can completely hide from the end user whether you have one or more app servers, on which ports they listen, or how their applications are named. Otherwise, if your datacenter needs to move the app to another server that listens on 8543, all bookmarks of your clients would become invalid. This example also directs the user to your entry page. This could be omitted if you name the entry page index.html. But maybe you have several entry pages within an app that the user might want to bookmark, so you are more flexible if you're not bound to index.html.

I postfixed the company URL with /app. This could be omitted if your domain only serves this app. In case that in addition to the app, there also is some static content, like your company description, you may want to have the simple URL for that.

This proxy only works for the entry page. I usually need two more URL schemes, one for static content of the web app, like JavaScript, CSS, and images. I put all those in a folder structure below a folder called serverapp, and write the following proxy:

```
location /app/serverapp/ {
    proxy_pass https://mydealer.com:8443/Dealerapp/serverapp/;
}
```

And another path for REST services; here the rest URL path doesn't match a folder, but a path of a jax-rs REST service:

```
location /rest/ {
    proxy_pass https://mydealer.com:8443/Dealerapp/rest/;
}
```

One more step, rarely mentioned anywhere, needs to be taken. The app server runs under the URL path /Dealerapp, so it will issue a session cookie having property path=Dealerapp. The browser doesn't know this path, and due to its Same Origin Policy will ignore the cookie. We could convince it via Cross-Origin Resource Sharing to allow this, but it's probably easier to modify the cookie path by setting the path to /, writing something like

```
<session-config>
    <session-timeout>720</session-timeout>
    <cookie-config>
        <name>SZSESSION</name>
        <path>/</path>
        <http-only>true</http-only>
        <secure>true</secure>
    </cookie-config>
</session-config>
```

to our web.xml.

Read Using nginx to provide clean browser URLs online:
https://riptutorial.com/nginx/topic/6270/using-nginx-to-provide-clean-browser-urls

Chapter 7: Wordpress blog integration with rails application using nginx

Examples

nginx server block

Once you have setup and configured the php5-fpm and wordpress settings you can configure the /etc/nginx/conf/nginx.conf file as below.

You need to define the location blocks inside the server block and rewrite the url there as defined.

```
server {
listen 443 ssl;
        server_name abc.co.uk;
    root /home/ubuntu/www/abc/current/public;
        try_files $uri/index.html $uri @unicorn;
    ssl on;
    ssl_certificate /etc/nginx/ssl/abc.crt;
    ssl_certificate_key /etc/nginx/ssl/abc.key;
    location /blog/wp-admin/ {
        root /var/www/html/;
        index index.php;
        try_files $uri $uri/ /index.php?$args;
        location ~* \.(js|css|xml|txt|jpg)$ {
             expires 14d;
         root /var/www/html/;
             access_log off;
           }

    location ~ \.php$ {
                try_files $uri $uri/ /index.php;
                fastcgi_pass unix:/var/run/php5-fpm.sock;
                fastcgi_param SCRIPT_FILENAME $request_filename;
                fastcgi_index index.php;
                include /etc/nginx/conf/fastcgi_params;
    }

}
location ^~ /blog {
    root /var/www/html/;
    index index.php;
    try_files $uri $uri/ /index.php?$args;
    rewrite ^/blog/(.*)+$ /blog/index.php?$1;
    location ~* \.(js|css|xml|txt|jpg)$ {
         expires 14d;
     root /var/www/html/;
         access_log off;
       }

    location ~ \.php$ {
                try_files $uri $uri/ /index.php;
                fastcgi_pass unix:/var/run/php5-fpm.sock;
                fastcgi_param SCRIPT_FILENAME $request_filename;
```

```
                fastcgi_index index.php;
                include /etc/nginx/conf/fastcgi_params;
    }

  }



}
```

Read Wordpress blog integration with rails application using nginx online: https://riptutorial.com/nginx/topic/3891/wordpress-blog-integration-with-rails-application-using-nginx

Credits

S. No	Chapters	Contributors
1	Getting started with nginx	Bbak, Community, I Am Batman, James, Marek Skiba, Mark Stosberg, Neo, Przemysław Jagielski, rajarshig, RamenChef, RationalDev, theDrifter, treecoder, Xevaquor
2	Logging	Gustav, RationalDev, timbo
3	Nginx Configurations	Bbak, James, Pablo Fernandez, RationalDev
4	nginx reverse proxy	smart-developer
5	Useful Redirects	Aleksey Deryagin, Alexandre Maciel, Alexey Ten, Gaurav Kumar, Joshua DeWald, Justin W., Keelan, Muaaz Rafi, smart-developer, timbo
6	Using nginx to provide clean browser URLs	TAM
7	Wordpress blog integration with rails application using nginx	Abid Iqbal