<https://gist.github.com/denji/8359866>

<https://github.com/denji/nginx-tuning>

log\_format  main '$remote\_addr - $remote\_user [$time\_local] "$request" '

                  '$status $body\_bytes\_sent "$http\_referer" '

                  '"$http\_user\_agent" "$http\_x\_forwarded\_for"'

**NGINX Tuning For Best Performance**

For this configuration you can use web server you like, i decided, because i work mostly with it to use nginx.

Generally, properly configured nginx can handle up to 400K to 500K requests per second (clustered), most what i saw is 50K to 80K (non-clustered) requests per second and 30% CPU load, course, this was 2 x Intel Xeon with HyperThreading enabled, but it can work without problem on slower machines.

**You must understand that this config is used in testing environment and not in production so you will need to find a way to implement most of those features best possible for your servers.**

* [Stable version NGINX (deb/rpm)](https://nginx.org/en/linux_packages.html#stable)
* [Mainline version NGINX (deb/rpm)](https://nginx.org/en/linux_packages.html#mainline)

First, you will need to install nginx

yum install nginx

apt install nginx

Backup your original configs and you can start reconfigure your configs. You will need to open your nginx.conf at /etc/nginx/nginx.conf with your favorite editor.

# you must set worker processes based on your CPU cores, nginx does not benefit from setting more than that

worker\_processes auto; #some last versions calculate it automatically

# number of file descriptors used for nginx

# the limit for the maximum FDs on the server is usually set by the OS.

# if you don't set FD's then OS settings will be used which is by default 2000

worker\_rlimit\_nofile 100000;

# only log critical errors

error\_log /var/log/nginx/error.log crit;

# provides the configuration file context in which the directives that affect connection processing are specified.

events {

# determines how much clients will be served per worker

# max clients = worker\_connections \* worker\_processes

# max clients is also limited by the number of socket connections available on the system (~64k)

worker\_connections 4000;

# optimized to serve many clients with each thread, essential for linux -- for testing environment

use epoll;

# accept as many connections as possible, may flood worker connections if set too low -- for testing environment

multi\_accept on;

}

http {

# cache informations about FDs, frequently accessed files

# can boost performance, but you need to test those values

open\_file\_cache max=200000 inactive=20s;

open\_file\_cache\_valid 30s;

open\_file\_cache\_min\_uses 2;

open\_file\_cache\_errors on;

# to boost I/O on HDD we can disable access logs

access\_log off;

# copies data between one FD and other from within the kernel

# faster than read() + write()

sendfile on;

# send headers in one piece, it is better than sending them one by one

tcp\_nopush on;

# don't buffer data sent, good for small data bursts in real time

tcp\_nodelay on;

# reduce the data that needs to be sent over network -- for testing environment

gzip on;

# gzip\_static on;

gzip\_min\_length 10240;

gzip\_comp\_level 1;

gzip\_vary on;

gzip\_disable msie6;

gzip\_proxied expired no-cache no-store private auth;

gzip\_types

# text/html is always compressed by HttpGzipModule

text/css

text/javascript

text/xml

text/plain

text/x-component

application/javascript

application/x-javascript

application/json

application/xml

application/rss+xml

application/atom+xml

font/truetype

font/opentype

application/vnd.ms-fontobject

image/svg+xml;

# allow the server to close connection on non responding client, this will free up memory

reset\_timedout\_connection on;

# request timed out -- default 60

client\_body\_timeout 10;

# if client stop responding, free up memory -- default 60

send\_timeout 2;

# server will close connection after this time -- default 75

keepalive\_timeout 30;

# number of requests client can make over keep-alive -- for testing environment

keepalive\_requests 100000;

}

Now you can save config and run bottom [command](https://www.nginx.com/resources/wiki/start/topics/tutorials/commandline/#stopping-or-restarting-nginx)

nginx -s reload

/etc/init.d/nginx start|restart

If you wish to test config first you can run

nginx -t

/etc/init.d/nginx configtest

**Just For Security Reason**

server\_tokens off;

**NGINX Simple DDoS Defense**

This is far away from secure DDoS defense but can slow down some small DDoS. Those configs are also in test environment and you should do your values.

# limit the number of connections per single IP

limit\_conn\_zone $binary\_remote\_addr zone=conn\_limit\_per\_ip:10m;

# limit the number of requests for a given session

limit\_req\_zone $binary\_remote\_addr zone=req\_limit\_per\_ip:10m rate=5r/s;

# zone which we want to limit by upper values, we want limit whole server

server {

limit\_conn conn\_limit\_per\_ip 10;

limit\_req zone=req\_limit\_per\_ip burst=10 nodelay;

}

# if the request body size is more than the buffer size, then the entire (or partial)

# request body is written into a temporary file

client\_body\_buffer\_size 128k;

# buffer size for reading client request header -- for testing environment

client\_header\_buffer\_size 3m;

# maximum number and size of buffers for large headers to read from client request

large\_client\_header\_buffers 4 256k;

# read timeout for the request body from client -- for testing environment

client\_body\_timeout 3m;

# how long to wait for the client to send a request header -- for testing environment

client\_header\_timeout 3m;

Now you can do again test config

nginx -t # /etc/init.d/nginx configtest

And then [reload or restart your nginx](https://www.nginx.com/resources/wiki/start/topics/tutorials/commandline/#stopping-or-restarting-nginx)

nginx -s reload

/etc/init.d/nginx reload|restart

You can test this configuration with tsung and when you are satisfied with result you can hit Ctrl+C because it can run for hours.

**Increase The Maximum Number Of Open Files (nofile limit) – Linux**

Two ways to raise the nofile/max open files/file descriptors/file handles limit for NGINX in RHEL/CentOS 7+. With NGINX running, checking current limit on master process

$ cat /proc/$(cat /var/run/nginx.pid)/limits | grep open.files

Max open files 1024 4096 files

**And worker processes**

ps --ppid $(cat /var/run/nginx.pid) -o %p|sed '1d'|xargs -I{} cat /proc/{}/limits|grep open.files

Max open files 1024 4096 files

Max open files 1024 4096 files

Trying with the worker\_rlimit\_nofile directive in {,/usr/local}/etc/nginx/nginx.conf fails as SELinux policy doesn't allow setrlimit. This is shown in /var/log/nginx/error.log

015/07/24 12:46:40 [alert] 12066#0: setrlimit(RLIMIT\_NOFILE, 2342) failed (13: Permission denied)

**And in /var/log/audit/audit.log**

type=AVC msg=audit(1437731200.211:366): avc: denied { setrlimit } for pid=12066 comm="nginx" scontext=system\_u:system\_r:httpd\_t:s0 tcontext=system\_u:system\_r:httpd\_t:s0 tclass=process

**nolimit without Systemd**

# /etc/security/limits.conf

# /etc/default/nginx (ULIMIT)

$ nano /etc/security/limits.d/nginx.conf

nginx soft nofile 65536

nginx hard nofile 65536

$ sysctl -p

**nolimit with Systemd**

$ mkdir -p /etc/systemd/system/nginx.service.d

$ nano /etc/systemd/system/nginx.service.d/nginx.conf

[Service]

LimitNOFILE=30000

$ systemctl daemon-reload

$ systemctl restart nginx.service

**SELinux boolean httpd\_setrlimit to true(1)**

This will set fd limits for the worker processes. Leave the worker\_rlimit\_nofile directive in {,/usr/local}/etc/nginx/nginx.conf and run the following as root

setsebool -P httpd\_setrlimit 1

**DoS**[**HTTP/1.1 and above: Range Requests**](https://tools.ietf.org/html/rfc7233#section-6.1)

By default [max\_ranges](https://nginx.org/r/max_ranges) is not limited. DoS attacks can many Range-Requests (Impact on stability I/O).

**Socket Sharding in NGINX 1.9.1+ (DragonFly BSD and Linux 3.9+)**

| **Socket type** | **Latency (ms)** | **Latency stdev (ms)** | **CPU Load** |
| --- | --- | --- | --- |
| Default | 15.65 | 26.59 | 0.3 |
| accept\_mutex off | 15.59 | 26.48 | 10 |
| reuseport | 12.35 | 3.15 | 0.3 |

[**Thread Pools**](https://nginx.org/r/thread_pool)**in NGINX Boost Performance 9x! (Linux)**

[Multi-threaded](https://nginx.org/r/aio) sending of files is currently supported only Linux. Without [sendfile\_max\_chunk](https://nginx.org/r/sendfile_max_chunk) limit, one fast connection may seize the worker process entirely.

**Selecting an upstream based on SSL protocol version**

map $ssl\_preread\_protocol $upstream {

"" ssh.example.com:22;

"TLSv1.2" new.example.com:443;

default tls.example.com:443;

}

# ssh and https on the same port

server {

listen 192.168.0.1:443;

proxy\_pass $upstream;

ssl\_preread on;

}

**Happy Hacking!**

**Reference links**

* [**https://github.com/trimstray/nginx-quick-reference**](https://github.com/trimstray/nginx-quick-reference)
* [**https://github.com/GrrrDog/weird\_proxies/wiki/nginx**](https://github.com/GrrrDog/weird_proxies/wiki/nginx)
* [**https://github.com/h5bp/server-configs-nginx**](https://github.com/h5bp/server-configs-nginx)
* [**https://github.com/leandromoreira/linux-network-performance-parameters**](https://github.com/leandromoreira/linux-network-performance-parameters)
* <https://github.com/nginx-boilerplate/nginx-boilerplate>
* <https://www.nginx.com/blog/thread-pools-boost-performance-9x/>
* <https://www.nginx.com/blog/socket-sharding-nginx-release-1-9-1/>
* <https://www.nginx.com/blog/nginx-1-13-9-http2-server-push/>
* <https://www.nginx.com/blog/performing-a-b-testing-nginx-plus/>
* <https://www.nginx.com/blog/10-tips-for-10x-application-performance/>
* <https://www.nginx.com/blog/http-keepalives-and-web-performance/>
* <https://www.nginx.com/blog/overcoming-ephemeral-port-exhaustion-nginx-plus/>
* <https://www.nginx.com/blog/tcp-load-balancing-udp-load-balancing-nginx-tips-tricks/>
* <https://www.nginx.com/blog/introducing-cicd-with-nginx-and-nginx-plus/>
* <https://www.nginx.com/blog/testing-the-performance-of-nginx-and-nginx-plus-web-servers/>
* <https://www.nginx.com/blog/smart-efficient-byte-range-caching-nginx/>
* <https://nginx.org/r/pcre_jit>
* <https://nginx.org/r/ssl_engine> (openssl engine -t )
* <https://www.nginx.com/blog/mitigating-ddos-attacks-with-nginx-and-nginx-plus/>
* <https://www.nginx.com/blog/tuning-nginx/>
* <https://github.com/intel/asynch_mode_nginx>
* <https://openresty.org/download/agentzh-nginx-tutorials-en.html>
* <https://www.maxcdn.com/blog/nginx-application-performance-optimization/>
* <https://www.nginx.com/blog/nginx-se-linux-changes-upgrading-rhel-6-6/>
* <https://medium.freecodecamp.org/a8afdbfde64d>
* <https://medium.freecodecamp.org/secure-your-web-application-with-these-http-headers-fd66e0367628>
* <https://gist.github.com/CMCDragonkai/6bfade6431e9ffb7fe88>
* <https://gist.github.com/denji/9130d1c95e350c58bc50e4b3a9e29bf4>
* <https://8gwifi.org/docs/nginx-secure.jsp>
* <http://www.codestance.com/tutorials-archive/nginx-tuning-for-best-performance-255>
* <https://ospi.fi/blog/centos-7-raise-nofile-limit-for-nginx.html>
* <https://www.linode.com/docs/websites/nginx/configure-nginx-for-optimized-performance>
* <https://haydenjames.io/nginx-tuning-tips-tls-ssl-https-ttfb-latency/>

**Static analyzers**

* <https://github.com/yandex/gixy>

**Syntax highlighting**

* <https://github.com/chr4/sslsecure.vim>
* <https://github.com/chr4/nginx.vim>
* <https://github.com/nginx/nginx/tree/master/contrib/vim>

**NGINX config formatter**

* <https://github.com/1connect/nginx-config-formatter>
* <https://github.com/lovette/nginx-tools/tree/master/nginx-minify-conf>

**NGINX configuration tools**

* <https://github.com/nginxinc/crossplane>
* <https://github.com/valentinxxx/nginxconfig.io>

**BBR (Linux 4.9+)**

* <https://blog.cloudflare.com/http-2-prioritization-with-nginx/>
* Linux v4.13+ as no longer required FQ (q\_disc) with BBR.
* <https://github.com/google/bbr/blob/master/Documentation/bbr-quick-start.md>
* <https://git.kernel.org/pub/scm/linux/kernel/git/davem/net-next.git/commit/?id=218af599fa635b107cfe10acf3249c4dfe5e4123>
* <https://github.com/systemd/systemd/issues/9725#issuecomment-413369212>
* If the latest Linux kernel distribution does not have tcp\_bbr enabled by default:

modprobe tcp\_bbr && echo 'tcp\_bbr' >> /etc/modules-load.d/bbr.conf

echo 'net.ipv4.tcp\_congestion\_control=bbr' >> /etc/sysctl.d/99-bbr.conf

# Recommended for production, but with Linux v4.13rc1+ can be used not only in FQ (`q\_disc') in BBR mode.

echo 'net.core.default\_qdisc=fq' >> /etc/sysctl.d/99-bbr.conf

sysctl --system