Custom Webserver/NodeJS sysctl.conf file.

Have updated and explanation in the conf file below.

```
# Kernel sysctl configuration file for Red Hat Linux / Centos
#
# For binary values, 0 is disabled, 1 is enabled.  See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Do not accept source routing
net.ipv4.conf.default.accept_source_route = 0

# Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename.
# Useful for debugging multi-threaded applications.
kernel.core_uses_pid = 1

# Controls the use of TCP syncookies
net.ipv4.tcp_syncookies = 0

# Disable netfilter on bridges.
#net.bridge.bridge-nf-call-ip6tables = 0
#net.bridge.bridge-nf-call-iptables = 0
#net.bridge.bridge-nf-call-arptables = 0

# Controls the maximum shared segment size, in bytes
kernel.shmmax = 68719476736

# Controls the maximum number of shared memory segments, in pages
kernel.shmall = 4294967296


#net.ipv4.netfilter.ip_conntrack_max = 256000
#net.ipv4.netfilter.ip_conntrack_tcp_timeout_established = 54000


################################################################################################
#                  _                                          _   _                      __   #
#                 | |                                        | | | |                    / _|  #
```

```
#   ___ _   _ ___| |_ ___  _ __ ___    ___ _   _ ___  ___| |_| |  ___ ___  _ __ | |_   #
#  / __| | | / __| __/ _ \| '_ ` _ \  / __| | | / __|/ __| __| | / __/ _ \| '_ \|  _|  #
# | (__| |_| \__ \ || (_) | | | | | | \__ \ |_| \__ \ (__| |_| || (_| (_) | | | | |    #
#  \___|\__,_|___/\__\___/|_| |_| |_| |___/\__, |___/\___|\__|_(_)___\___/|_| |_|_|    #
#                                           __/ |                                      #
#                                          |___/                                       #
#  Added by : zahmed
##############################################################################################

# Controls the default maxmimum size of a mesage queue
kernel.msgmnb = 100000

# Controls the maximum size of a message, in bytes
kernel.msgmax = 100000

#net.netfilter.nf_conntrack_max = 1024000
#net.netfilter.nf_conntrack_generic_timeout = 120

# "Performance Scalability of a Multi-Core Web Server", Nov 2007
# Bryan Veal and Annie Foong, Intel Corporation, Page 4/10

# fs.file-max : http://www.cyberciti.biz/files/linux-kernel/Documentation/sysctl/fs.txt
# The value in file-max denotes the maximum number of file-
# handles that the Linux kernel will allocate. When you get lots
# of error messages about running out of file handles, you might
# want to increase this limit.
fs.file-max = 2097152


# pid_max: http://www.cyberciti.biz/files/linux-kernel/Documentation/sysctl/kernel.txt
# PID allocation wrap value.  When the kernel's next PID value
# reaches this value, it wraps back to a minimum PID value.
# PIDs of value pid_max or larger are not allocated.
# kernel.pid_max = 2097152

# http://www.cyberciti.biz/files/linux-kernel/Documentation/sysctl/net.txt
# rmem_default
# The default setting of the socket receive buffer in bytes.
#
net.core.rmem_default = 1048576

# rmem_max
# The maximum receive socket buffer size in bytes.
#
net.core.rmem_max = 16777216

# wmem_default
# The default setting (in bytes) of the socket send buffer.
#
net.core.wmem_default = 1048576


# wmem_max
# The maximum send socket buffer size in bytes.
net.core.wmem_max = 16777216
```

```
# optmem_max
#
# Maximum ancillary buffer size allowed per socket. Ancillary data is a sequence
# of struct cmsghdr structures with appended data.
net.core.optmem_max = 25165824

# somaxconn - INTEGER
#   Limit of socket listen() backlog, known in userspace as SOMAXCONN.
#   Defaults to 128.  See also tcp_max_syn_backlog for additional tuning
#   for TCP sockets.
net.core.somaxconn = 65536

# netdev_max_backlog
# ------------------
# Maximum number  of  packets,  queued  on  the  INPUT  side, when the interface
# receives packets faster than kernel can process them.
net.core.netdev_max_backlog = 65536

# tcp_moderate_rcvbuf - BOOLEAN
#   If set, TCP performs receive buffer auto-tuning, attempting to
#   automatically size the buffer (no greater than tcp_rmem[2]) to
#   match the size required by the path for full throughput.  Enabled by
#   default.
net.ipv4.tcp_moderate_rcvbuf = 0

# rp_filter - INTEGER
#   0 - No source validation.
#   1 - Strict mode as defined in RFC3704 Strict Reverse Path
#       Each incoming packet is tested against the FIB and if the interface
#       is not the best reverse path the packet check will fail.
#       By default failed packets are discarded.
#   2 - Loose mode as defined in RFC3704 Loose Reverse Path
#       Each incoming packet's source address is also tested against the FIB
#       and if the source address is not reachable via any interface
#       the packet check will fail.
#
#   Current recommended practice in RFC3704 is to enable strict mode
#   to prevent IP spoofing from DDos attacks. If using asymmetric routing
#   or other complicated routing, then loose mode is recommended.
#
#   The max value from conf/{all,interface}/rp_filter is used
#   when doing source validation on the {interface}.
#
#   Default value is 0. Note that some distributions enable it
#   in startup scripts.
net.ipv4.conf.all.rp_filter = 1


# ip_local_port_range - 2 INTEGERS
#   Defines the local port range that is used by TCP and UDP to
#   choose the local port. The first number is the first, the
#   second the last local port number. Default value depends on
#   amount of memory available on the system:
#   > 128Mb 32768-61000
```

```
#   < 128Mb 1024-4999 or even less.
#   This number defines number of active connections, which this
#   system can issue simultaneously to systems not supporting
#   TCP extensions (timestamps). With tcp_tw_recycle enabled
#   (i.e. by default) range 1024-4999 is enough to issue up to
#   2000 connections per second to systems supporting timestamps.
net.ipv4.ip_local_port_range = 1024 65535

# tcp_congestion_control - STRING
#   Set the congestion control algorithm to be used for new
#   connections. The algorithm "reno" is always available, but
#   additional choices may be available based on kernel configuration.
#   Default is set as part of kernel configuration.
net.ipv4.tcp_congestion_control = bic

# tcp_ecn - BOOLEAN
#   Enable Explicit Congestion Notification (ECN) in TCP. ECN is only
#   used when both ends of the TCP flow support it. It is useful to
#   avoid losses due to congestion (when the bottleneck router supports
#   ECN).
#   Possible values are:
#       0 disable ECN
#       1 ECN enabled
#       2 Only server-side ECN enabled. If the other end does
#         not support ECN, behavior is like with ECN disabled.
#   Default: 2
net.ipv4.tcp_ecn = 0

# tcp_max_syn_backlog - INTEGER
#   Maximal number of remembered connection requests, which are
#   still did not receive an acknowledgment from connecting client.
#   Default value is 1024 for systems with more than 128Mb of memory,
#   and 128 for low memory machines. If server suffers of overload,
#   try to increase this number.
net.ipv4.tcp_max_syn_backlog = 100000

# tcp_max_orphans - INTEGER
#   Maximal number of TCP sockets not attached to any user file handle,
#   held by system. If this number is exceeded orphaned connections are
#   reset immediately and warning is printed. This limit exists
#   only to prevent simple DoS attacks, you _must_ not rely on this
#   or lower the limit artificially, but rather increase it
#   (probably, after increasing installed memory),
#   if network conditions require more than default value,
#   and tune network services to linger and kill such states
#   more aggressively. Let me to remind again: each orphan eats
#   up to ~64K of unswappable memory.

net.ipv4.tcp_max_orphans = 262144

# tcp_max_tw_buckets - INTEGER
#   Maximal number of timewait sockets held by system simultaneously.
#   If this number is exceeded time-wait socket is immediately destroyed
#   and warning is printed. This limit exists only to prevent
#   simple DoS attacks, you _must_ not lower the limit artificially,
```

```
#   but rather increase it (probably, after increasing installed memory),
#   if network conditions require more than default value.
net.ipv4.tcp_max_tw_buckets = 2000000

# tcp_sack - BOOLEAN
#   Enable select acknowledgments (SACKS).
net.ipv4.tcp_sack = 1

# tcp_timestamps - BOOLEAN
#   Enable timestamps as defined in RFC1323
net.ipv4.tcp_timestamps = 1

# tcp_fin_timeout - INTEGER
#   Time to hold socket in state FIN-WAIT-2, if it was closed
#   by our side. Peer can be broken and never close its side,
#   or even died unexpectedly. Default value is 60sec.
#   Usual value used in 2.2 was 180 seconds, you may restore
#   it, but remember that if your machine is even underloaded WEB server,
#   you risk to overflow memory with kilotons of dead sockets,
#   FIN-WAIT-2 sockets are less dangerous than FIN-WAIT-1,
#   because they eat maximum 1.5K of memory, but they tend
#   to live longer. Cf. tcp_max_orphans.
net.ipv4.tcp_fin_timeout = 10

# tcp_slow_start_after_idle - BOOLEAN
#   If set, provide RFC2861 behavior and time out the congestion
#   window after an idle period.  An idle period is defined at
#   the current RTO.  If unset, the congestion window will not
#   be timed out after an idle period.
#   Default: 1
net.ipv4.tcp_slow_start_after_idle = 0

# tcp_mem - vector of 3 INTEGERs: min, pressure, max
#   min: below this number of pages TCP is not bothered about its
#   memory appetite.
#
#   pressure: when amount of memory allocated by TCP exceeds this number
#   of pages, TCP moderates its memory consumption and enters memory
#   pressure mode, which is exited when memory consumption falls
#   under "min".
#
#   max: number of pages allowed for queueing by all TCP sockets.
#
#   Defaults are calculated at boot time from amount of available
#   memory.
#net.ipv4.tcp_mem = 30000000 30000000 30000000
net.ipv4.tcp_mem = 65536 131072 262144

# tcp_rmem - vector of 3 INTEGERs: min, default, max
#   min: Minimal size of receive buffer used by TCP sockets.
#   It is guaranteed to each TCP socket, even under moderate memory
#   pressure.
#   Default: 8K
#
#   default: initial size of receive buffer used by TCP sockets.
```

```
#   This value overrides net.core.rmem_default used by other protocols.
#   Default: 87380 bytes. This value results in window of 65535 with
#   default setting of tcp_adv_win_scale and tcp_app_win:0 and a bit
#   less for default tcp_app_win. See below about these variables.
#
#   max: maximal size of receive buffer allowed for automatically
#   selected receiver buffers for TCP socket. This value does not override
#   net.core.rmem_max.  Calling setsockopt() with SO_RCVBUF disables
#   automatic tuning of that socket's receive buffer size, in which
#   case this value is ignored.
#   Default: between 87380B and 4MB, depending on RAM size.
#net.ipv4.tcp_rmem = 30000000 30000000 30000000
net.ipv4.tcp_rmem = 8192 87380 16777216
net.ipv4.udp_rmem_min = 16384

# tcp_wmem - vector of 3 INTEGERs: min, default, max
#   min: Amount of memory reserved for send buffers for TCP sockets.
#   Each TCP socket has rights to use it due to fact of its birth.
#   Default: 4K
#
#   default: initial size of send buffer used by TCP sockets.  This
#   value overrides net.core.wmem_default used by other protocols.
#   It is usually lower than net.core.wmem_default.
#   Default: 16K
#
#   max: Maximal amount of memory allowed for automatically tuned
#   send buffers for TCP sockets. This value does not override
#   net.core.wmem_max.  Calling setsockopt() with SO_SNDBUF disables
#   automatic tuning of that socket's send buffer size, in which case
#   this value is ignored.
#   Default: between 64K and 4MB, depending on RAM size.
#net.ipv4.tcp_wmem = 30000000 30000000 30000000
net.ipv4.tcp_wmem = 8192 65536 16777216

# max_map_count:
#
# This file contains the maximum number of memory map areas a process
# may have. Memory map areas are used as a side-effect of calling
# malloc, directly by mmap and mprotect, and also when loading shared
# libraries.
#
# While most applications need less than a thousand maps, certain
# programs, particularly malloc debuggers, may consume lots of them,
# e.g., up to one or two maps per allocation.
#
# The default value is 65536.
vm.max_map_count = 262144

# optionally, avoid TIME_WAIT states on localhost no-HTTP Keep-Alive tests:
#    "error: connect() failed: Cannot assign requested address (99)"
# On Linux, the 2MSL time is hardcoded to 60 seconds in /include/net/tcp.h:
# #define TCP_TIMEWAIT_LEN (60*HZ)
# The option below is safe to use:

# tcp_tw_reuse - BOOLEAN
```

```
#   Allow to reuse TIME-WAIT sockets for new connections when it is
#   safe from protocol viewpoint. Default value is 0.
#   It should not be changed without advice/request of technical
#   experts.
net.ipv4.tcp_tw_reuse = 1

# tcp_tw_recycle - BOOLEAN
#   Enable fast recycling TIME-WAIT sockets. Default value is 0.
#   It should not be changed without advice/request of technical
#   experts.
net.ipv4.tcp_tw_recycle= 1

## Increase backlog for UNIX sockets.

# max_dgram_qlen - INTEGER
#   The maximum length of dgram socket receive queue
#   Default: 10
net.unix.max_dgram_qlen = 100

# ip_nonlocal_bind - BOOLEAN
#   If set, allows processes to bind() to non-local IP addresses,
#   which can be quite useful - but may break some applications.
#   Default: 0
net.ipv4.ip_nonlocal_bind = 1

# `net.ipv4.tcp_synack_retries`
# Number of times SYNACKs for a passive TCP connection attempt will be retransmitted.
# Should not be higher than 255. Default value is 5, which corresponds to ~180seconds.
# Number of times SYNACKs for passive TCP connection.
net.ipv4.tcp_synack_retries = 2

# tcp_syn_retries - INTEGER
#   Number of times initial SYNs for an active TCP connection attempt
#   will be retransmitted. Should not be higher than 255. Default value
#   is 5, which corresponds to ~180seconds.
net.ipv4.tcp_syn_retries = 2

# min_free_kbytes:
#
# This is used to force the Linux VM to keep a minimum number
# of kilobytes free.  The VM uses this number to compute a
# watermark[WMARK_MIN] value for each lowmem zone in the system.
# Each lowmem zone gets a number of reserved free pages based
# proportionally on its size.
#
# Some minimal amount of memory is needed to satisfy PF_MEMALLOC
# allocations; if you set this to lower than 1024KB, your system will
# become subtly broken, and prone to deadlock under high loads.
#
# Setting this too high will OOM your machine instantly.
vm.min_free_kbytes = 65536
```

More information:

https://engineering.gosquared.com/optimising-nginx-node-js-and-networking-for-heavy-workloads

http://www.cyberciti.biz/files/linux-kernel/Documentation/sysctl/net.txt
http://www.cyberciti.biz/files/linux-kernel/Documentation/sysctl/fs.txt
http://www.cyberciti.biz/files/linux-kernel/Documentation/sysctl/kernel.txt
http://www.cyberciti.biz/faq/linux-kernel-etcsysctl-conf-security-hardening/