rpm -i /appserver/pkgs/pcre-7.8-6.el6.x86\_64.rpm

rpm -i /appserver/pkgs/pcre-devel-7.8-6.el6.x86\_64.rpm

رفتن به فولدر unzip شده nginx و اجرای دستور زیر:

# ./configure --prefix=/etc/nginx --sbin-path=/usr/sbin/nginx --conf-path=/etc/nginx/nginx.conf --error-log-path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --pid-path=/var/run/nginx.pid --lock-path=/var/run/nginx.lock --with-http\_ssl\_module --with-pcre

در صورت مواجه شدن با خطای pcre باید پکیج مربوط به ان کپی شود و ادرس ان بصورت زیر داده شود.

./configure --prefix=/etc/nginx --sbin-path=/usr/sbin/nginx --conf-path=/etc/nginx/nginx.conf --error-log-path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --pid-path=/var/run/nginx.pid --lock-path=/var/run/nginx.lock --with-http\_ssl\_module --with-pcre=./pcre-8.42

./configure

make

sudo make install

nginx -V

در فایل /etc/init.d/nginx دستورات زیر را وارد کنید.

#!/bin/sh

#

# nginx - this script starts and stops the nginx daemon

#

# chkconfig: - 85 15

# description: Nginx is an HTTP(S) server, HTTP(S) reverse \

# proxy and IMAP/POP3 proxy server

# processname: nginx

# config: /etc/nginx/nginx.conf

# pidfile: /var/run/nginx.pid

# user: nginx

# Source function library.

. /etc/rc.d/init.d/functions

# Source networking configuration.

. /etc/sysconfig/network

# Check that networking is up.

[ "$NETWORKING" = "no" ] && exit 0

nginx="/usr/sbin/nginx"

prog=$(basename $nginx)

NGINX\_CONF\_FILE="/etc/nginx/nginx.conf"

lockfile=/var/run/nginx.lock

start() {

[ -x $nginx ] || exit 5

[ -f $NGINX\_CONF\_FILE ] || exit 6

echo -n $"Starting $prog: "

daemon $nginx -c $NGINX\_CONF\_FILE

retval=$?

echo

[ $retval -eq 0 ] && touch $lockfile

return $retval

}

stop() {

echo -n $"Stopping $prog: "

killproc $prog -QUIT

retval=$?

echo

[ $retval -eq 0 ] && rm -f $lockfile

return $retval

}

restart() {

configtest || return $?

stop

start

}

reload() {

configtest || return $?

echo -n $"Reloading $prog: "

killproc $nginx -HUP

RETVAL=$?

echo

}

force\_reload() {

restart

}

configtest() {

$nginx -t -c $NGINX\_CONF\_FILE

}

rh\_status() {

status $prog

}

rh\_status\_q() {

rh\_status >/dev/null 2>&1

}

case "$1" in

start)

rh\_status\_q && exit 0

$1

;;

stop)

rh\_status\_q || exit 0

$1

;;

restart|configtest)

$1

;;

reload)

rh\_status\_q || exit 7

$1

;;

force-reload)

force\_reload

;;

status)

rh\_status

;;

condrestart|try-restart)

rh\_status\_q || exit 0

;;

\*)

echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload|configtest}"

exit 2

esac

Nginx دارای یک و یا بیش از یک پروسس می باشد. پروسس اصلی که پروسس والد می باشد فایلهای کانفیگ را خوانه و ارزیابی می کند و مسئول نگهداری پروسس های کارگر می باشد. پروسس های کارگز مسئول پردازش درخواستها هستند. Nginx وابسته به مکانیزم سیستم عامل درخواستها را بین پروسسهای کارگز توزیع می کند. تعداد پروسس های کارگز با [worker\_processes](http://nginx.org/en/docs/ngx_core_module.html?&_ga=2.104123648.1820226442.1530357386-884681744.1529214691#worker_processes) د ر فایل nginx.conf مشخص می شود که می تواند به صورت اتوماتیک به تعداد هسته های پردازشگر باشد.

بعنوان یک دستورالعمل سطح بالا Contexts گروهی از دستورها هستند که به انواع مختلفی از ترافیک اعمال می شوند.

* [events](http://nginx.org/en/docs/ngx_core_module.html#events) – General connection processing
* [http](http://nginx.org/en/docs/http/ngx_http_core_module.html#http) – HTTP traffic
* [mail](http://nginx.org/en/docs/mail/ngx_mail_core_module.html#mail) – Mail traffic
* [stream](http://nginx.org/en/docs/stream/ngx_stream_core_module.html#stream) – TCP and UDP traffic

دستوراتي که بيرون از اين context ها قرار مي گيرند بعنوان دستورات اصلي ناميده مي شوند.

ساختار فايل nginx.conf به اين صورت مي باشد که دستورات که درون هيچ بلاکي نيستند دستورات اصلي هستند.

دستوراتي که درون يکي از context هاي بالا قرار ميگيرد، ترافيک مربوط به همان context را کنترل مي کند. و درون هر context مي توانيد virtual server هاي متعددي تعريف کنيد با تگ server.

user nobody; # a directive in the 'main' context

events {

# configuration of connection processing

}

http {

# Configuration specific to HTTP and affecting all virtual servers

server {

# configuration of HTTP virtual server 1

location /one {

# configuration for processing URIs starting with '/one'

}

location /two {

# configuration for processing URIs starting with '/two'

}

}

server {

# configuration of HTTP virtual server 2

}

}

stream {

# Configuration specific to TCP/UDP and affecting all virtual servers

server {

# configuration of TCP virtual server 1

}

}

مثال

#user nobody;

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

upstream backend\_servers {

# least\_conn;

server 10.100.0.213;

server 10.100.0.214;

}

server {

# listen 8080;

location / {

proxy\_pass http://backend\_servers;

}

}

}

بصورت کلي در يک context فرزند که درون يک context پدر مي باشد تنظيمات و دستورات از contxt پدر به ارث برده مي شود. مثلا چندين تنظيمات ممکن است که در چندين context وجود داشته باشد به اين ترتيب مي توانيد اين موارد را با استفاده وراثت مديريت کنيد.

دقت شود که دستوراتي که در context هاي فرزندان وجود دارند بر روي تنظيمات context پدر override مي شوند.

برای فعال کردن https در nginx باید یک فایل cert , key داشته باشید . اگر این فایلها را ندارید باید از طریف اسکریپت generateSSLforNginx این فایلها را بسازید.

و بعد تنظیمات nginx.conf را بصورت زیر انجام دهید.

server {

listen 443 ssl;

ssl on;

ssl\_certificate /etc/ssl/certs/myssl.crt;

ssl\_certificate\_key /etc/ssl/private/myssl.key;

server\_name SERVER\_NAME.com;

location / {

}

}

Reload تنظيمات

براي اينکه فايل تغييرات اعمال شود پروسس nginx بايد reloada شود اين امر به دوصورت امکان پذير مي باشد:

مي توانيد پروسس nginx را reload کنيد و يا اينکه سيگنال reload را به ان ارسال کنيد که در اين صورت بدون وقفه تنظيمات اعمال خواهد شد. بديهي است که درروش اول وقفه صورت خواهد پذيرفت

نحوه ارسال سيگنال به nginx :

nginx -s <SIGNAL>

سيگنال ها مي توانند:

* quit – Shut down gracefully
* reload – Reload the configuration file
* reopen – Reopen log files
* stop – Shut down immediately (fast shutdown)

براي load balance کردن بار http گروهي از سروها ابتدا نياز است که يک گروه با دستور upstream تعريف کنيم که درون context مربوط به http باشد.

سرورهاي موجود در گروه با دستور server بايد پيکر بندي شوند. (با بلاک server اشتباه نشود)

http {

upstream backend {

server backend1.example.com weight=5;

server backend2.example.com;

server 192.0.0.1 backup;

}

}

براي ارسال درخواستها به گروه سرورها نام گروه بايد در دستور [proxy\_pass](https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_pass) مشخص شود.

server {

location / {

proxy\_pass http://backend;

}

}

در اين دستور يک virtual server روي nginx در حال اجرا مي باشد که همه درخواستها را به گروه بالاي دستي به نام **backend ارسال ميکند.**

[max\_fails](https://nginx.org/en/docs/http/ngx_http_upstream_module.html#max_fails) = تعداد تلاش از دست رفته برای ارتباط با سرور را مشخص می کند. که اگر این تعداد تلاش ناموفق اتفاق بیفتد سرور دیگر بار را روی سرور نمیفرستد.

[fail\_timeout](https://nginx.org/en/docs/http/ngx_http_upstream_module.html#fail_timeout)= زمان لازم برای اینکه تعداد تلاشهای پارامتر قبلی را انجام دهد. یعنی در طول این زمان [max\_fails](https://nginx.org/en/docs/http/ngx_http_upstream_module.html#max_fails) تعداد درخواست اگر fail شود سرور از خارج از دسترس در نظر گرفته می شود.

در تنظیمات keepalived برای اینکه بتوانیم پروسس nginx را بررسی کنیم و بفهمیم که اگر پایین بود عملیات سوئیچ رول انجام شود باید دو بلاک زیر را اضافه کنیم.

این قسمت در روت فایل keepalived.conf اضافه شود

vrrp\_script chk\_http\_port {

script "/usr/bin/killall -0 nginx"

interval 2

weight 2

}

قسمت زیر در vrrp\_instance اضافه شود.

chk\_http\_port {

chk\_nginx

}

برای تست کردن دستور زیر را اجرا کنید.

ip add sh eth0

برای چک کردن ترافیک شبکه بین دو نود keepalived از دستور زیر استفاده می شود.

tcpdump -i eth0 | grep "vrid 33"

برای سرویسی کردن keepalived باید سریس زیر را در مسیر /etc/init.d/keepalived ایجاد کنید.

#!/bin/sh

#

# Startup script for the Keepalived daemon

#

# processname: keepalived

# pidfile: /var/run/keepalived.pid

# config: /etc/keepalived/keepalived.conf

# chkconfig: 35 21 79

# description: Start and stop Keepalived

# Global definitions

PID\_FILE="/var/run/keepalived.pid"

# source function library

. /etc/init.d/functions

RETVAL=0

start() {

echo -n "Starting Keepalived for LVS: "

daemon keepalived -D

RETVAL=$?

echo

[ $RETVAL -eq 0 ] && touch /var/lock/subsys/keepalived

return $RETVAL

}

stop() {

echo -n "Shutting down Keepalived for LVS: "

killproc keepalived

RETVAL=0

echo

[ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/keepalived

return $RETVAL

}

reload() {

echo -n "Reloading Keepalived config: "

killproc keepalived -1

RETVAL=$?

echo

return $RETVAL

}

# See how we were called.

case "$1" in

start)

start

;;

stop)

stop

;;

restart)

stop

start

;;

reload)

reload

;;

status)

status keepalived

;;

condrestart)

[ -f /var/lock/subsys/keepalived ] && $0 restart || :

;;

\*)

echo "Usage: $0 {start|stop|restart|reload|condrestart|status}"

exit 1

esac

exit 0

* در حالت عادی سرور nginx بعنوان یک reverse proxy عمل می کند و همه درخواستها به انجایی ما گفته ایم ارسال می کند .

این به این مفهوم می باشد که خود سرور nginx بعنوان یک کلاینت برای سرورهای backend در نظر گرفته می شود و اگر فایل access.log سرورهای backend را ببینیم ادرس ip سرور nginx را بعنوان کلاینت می بینیم.

اگر بخواهیم که سرور nginx ادرس ip کلاینت اصلی را برای سرور ها در هدر request ارسال کند این هدر را در location مد نظر بصورت زیر ست می کنیم.

proxy\_set\_header X-Real-IP $remote\_addr;

* مشکل بعدی که ممکن است در استفاده از nginx بعنوان reverse proxy با ان روبرو شویم این است که سرور nginx بعد از اینکه درخواست را از کلاینت اصلی گرفت هدر Host انرا پاک می کند و بعد درخواست را برای سرورهای backend می فرستد و در جاهایی که در دو سرور متفاوت یک فایل با نام یکسان وجود داشته باشد و همان فایل را کاربر درخواست دهد سرور nginx نمی داند که باید این درخواست را از کدام location بگیرد و ممکن است که جواب اشتباه به کاربرد بدهد.(مانند kp.net , kp.com)

برای رفع این موضوع از هدر زیر در location استفاده می کنیم.

proxy\_set\_header Host $host;

If you want to pass the variable to your proxy backend, you have to set it with the proxy module.

location / {

proxy\_pass http://example.com;

proxy\_set\_header Host example.com;

proxy\_set\_header HTTP\_Country-Code $geoip\_country\_code;

proxy\_pass\_request\_headers on;

}

در حالت عادی هر worker process برای نگهداری وضعیت سرورهای backend از حافظه مختص خودش استفاده می کند. یعنی هر کسی وضعیت سرورها را برای خودش نگه میدارد. در حالیکه اگر چند worker داشته باشیم بهتر است که از یک shared memory استفاده کنند تا همه از یک جا اطلاعات مربوط به سرورها را بدست بیاورند.

upstream {

zone upstream\_dynamic 64k;

}

Location / {

healcheck;

}

Healthcheck هر 5 ثانیه یک درخواست get به سرور میزند و چک می کند که سرور بالاست یانه.

* دستور زیر به این معناست که:

server 192.0.2.1 max\_fails=3 fail\_timeout=50s

اگر سه درخواست به سرور ارسال شود و هر سه درخواست failشود سرور برای لودبالانسر پایین در نظر گرفته می شود و تا 50 ثانیه دیگر درخواستی را به ان نمیفرستد و بعد از 50 ثانیه اگر دوباره سه درخواست دیگر به سرور ارسال شود و جواب ندهد باز هم تا 50 ثانیه سرور پایین در نظر گرفت می شود.

دقت شود که در این حالت پاسخ درست از سرور دیگر به کاربر نشان داده می شود که به این موضوع active-passive healch check گفته می شود.