

# Rocky Linux 8

**ROCKY LINUX:** <https://rockylinux.org>

*Oh you know... just a little of this and a little of that...*

- NGINX / PHP / MARIADB INSTALL
- LIBHEIF INSTALL
- IMAGICK WITH HEIC (nginx/php)
- EXIFTOOL

# NGINX / PHP / MARIADB

## INSTALL

This will just be a quick write up of the steps that can be used to setup a Nginx web server using the nginx repo, along with PHP from the Remi repo, and mariadb from the MariaDB repo. By using the Nginx and MariaDB repositories, I can have a bit of comfort knowing that the applications are clean, maintained, and patched. There are other methods out there to get PHP loaded in, but I find that the Remi repo does a pretty good job at maintaining the packages, and they also have a great collection of already built modules that just slip right in and work. So let me get started.

**ROCKY LINUX:** <https://rockylinux.org>

**NGINX:** <https://nginx.org>

**REMI PHP:** <https://rpms.remirepo.net>

**MARIADB:** <https://mariadb.com>

**EPEL:** <https://docs.stg.fedoraproject.org/en-US/epel/>

```
# ---- FIND A NICE PLACE TO WORK ----
```

```
cd /opt
```

```
# ---- SWAP CENTOS LOGOS, ADD EPEL AND PHP REPOS ----
```

```
dnf -y swap centos-logos-httpd rocky-logos-httpd
```

```
dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

```
dnf -y config-manager --set-enabled powertools
```

```
dnf -y install https://rpms.remirepo.net/enterprise/remi-release-8.rpm
```

```
dnf config-manager --enable remi
```

```
# ---- RESET DNF MODULES AND ENABLE REMI PHP-----
```

```
dnf -y module reset php
```

```
dnf -y module reset nginx
```

```
dnf -y module reset httpd
```

```
dnf -y module disable php*
```

```
dnf -y module disable composer*
```

```
dnf -y module enable php:remi-8.2
```

```
dnf -y module enable composer
```

```
# ----- ADD MARIADB REPO -----
```

```
wget https://downloads.mariadb.com/MariaDB/mariadb_repo_setup
```

```
chmod +x mariadb_repo_setup
```

```
./mariadb_repo_setup --mariadb-server-version="mariadb-11.4"
```

```
rm -f mariadb_repo_setup
```

```
# ----- ADD NGINX REPO FILE -----
```

```
cat > /etc/yum.repos.d/nginx.repo << "EOF"
```

```
[nginx-stable]
```

```
name=nginx stable repo
```

```
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
```

```
gpgcheck=1
```

```
enabled=0
```

```
gpgkey=https://nginx.org/keys/nginx_signing.key
```

```
module_hotfixes=true
```

```
[nginx-mainline]
```

```
name=nginx mainline repo
```

```
baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/
```

```
gpgcheck=1
```

```
enabled=1
```

```
gpgkey=https://nginx.org/keys/nginx_signing.key
```

```
module_hotfixes=true
```

```
EOF
```

```
# ----- ENABLE MAINLINE REPO -----
```

```
dnf config-manager --enable nginx-mainline
```

```
# ----- UPDATE THE SYSTEM AND INSTALL DEVELOPMENT TOOLS -----
```

```
dnf -y update --refresh
```

```
dnf -y group install "Development Tools"
```

```
# ----- INSTALL NGINX, MARIADB, AND A COUPLE PHP MODULES -----
```

```
dnf -y install \  
nginx \  
php-mysqlnd php-pecl-mysql \  
MariaDB-server MariaDB-client mariadb-tools
```

```
# ----- ENABLE AND START MARIADB AND RUN THE SECURE SCRIPT -----
```

```
systemctl enable mariadb  
systemctl start mariadb
```

```
source mariadb-secure-installation
```

```
### I ANSWER THE QUESTIONS INITIALLY LIKE SO...
```

```
Enter current password for root (enter for none): 'enter'
```

```
Switch to unix_socket authentication [Y/n] 'n'
```

```
Change the root password? [Y/n] 'n'
```

```
Remove anonymous users? [Y/n] 'y'
```

```
Disallow root login remotely? [Y/n] 'n'
```

```
Remove test database and access to it? [Y/n] 'y'
```

```
Reload privilege tables now? [Y/n] 'y'
```

```
### RUN THE FOLLOWING TO ENABLE REMOTE ROOT ACCESS
```

```
### WARNING: CHANGE THE PASSOWRD FROM 'PASSWORD'
```

```
#
```

```
# mariadb
```

```
# CREATE USER 'root'@'%' IDENTIFIED BY 'PASSWORD';
```

```
# GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;
```

```
# FLUSH PRIVILEGES;
```

```
# exit
```

```
#
```

```
# ----- FIX UP SOME PERMISSIONS AND ENABLE SERVICES-----
```

```
chown -R nginx:nginx /usr/share/nginx/html  
chown -R nginx:nginx /var/lib/php/session/  
chown -R root:nginx /var/lib/php/{opcache,wsdlcache}  
chown -R nginx:nginx /var/log/{nginx,php-fpm}
```

```
systemctl enable nginx
systemctl enable php-fpm
```

```
# ----- QUICK CONFIG FOR PHP-FPM -----
```

```
mv /etc/php-fpm.d/www.conf /etc/php-fpm.d/www.bak
```

```
cat > /etc/php-fpm.d/www.conf << "EOF"
```

```
[www]
```

```
user = nginx
```

```
group = nginx
```

```
listen.owner = nginx
```

```
listen.group = nginx
```

```
listen.mode = 0660
```

```
listen = /var/run/php-fpm/www.sock
```

```
listen.allowed_clients = 127.0.0.1
```

```
pm = dynamic
```

```
pm.max_children = 25
```

```
pm.start_servers = 6
```

```
pm.min_spare_servers = 6
```

```
pm.max_spare_servers = 18
```

```
pm.process_idle_timeout = 10s
```

```
pm.max_requests = 250
```

```
pm.status_path = /status
```

```
slowlog = /var/log/php-fpm/www-slow.log
```

```
php_admin_value[error_log] = /var/log/php-fpm/www-error.log
```

```
php_admin_flag[log_errors] = on
```

```
php_value[session.save_handler] = files
```

```
php_value[session.save_path] = /var/lib/php/session
```

```
php_value[soap.wsdl_cache_dir] = /var/lib/php/wsdlcache
```

```
; Default Value: clean env
```

```
clear_env = no
```

```
;env[HOSTNAME] = $HOSTNAME
```

```
;env[PATH] = /usr/local/bin:/usr/bin:/bin
```

```
;env[TMP] = /tmp
;env[TMPDIR] = /tmp
;env[TEMP] = /tmp
```

EOF

```
# ----- QUICK DEFAULT SITE CONFIG WITH PHP ENABLED -----
mv /etc/nginx/conf.d/default.conf /etc/nginx/conf.d/default.bak
cp /etc/nginx/nginx.conf /etc/nginx/nginx.bak
cat > /etc/nginx/nginx.conf << "EOF"
# CUSTOM CONF
```

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;
```

```
events {
    worker_connections 1024;
}
```

```
http {
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';
```

```
    access_log    /var/log/nginx/access.log;
```

```
    sendfile      on;
    tcp_nopush     on;
    tcp_nodelay    on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
```

```
    include       /etc/nginx/mime.types;
    default_type   application/octet-stream;
```

```
    include       /etc/nginx/conf.d/*.conf;
```

```
    server {
```

```

listen    80 default_server;
server_name _;

root      /usr/share/nginx/html;

include   /etc/nginx/default.d/*.conf;

location  / {

}

error_page 404 /404.html;
    location = /40x.html {
}

error_page 500 502 503 504 /50x.html;
    location = /50x.html {
}
}
EOF

# ----- DROP IN A PHPINFO FILE -----
cat > /usr/share/nginx/html/phpinfo.php << "EOF"
<?php
    phpinfo();
?>
EOF

# ----- START NGINX AND PHP-FPM -----

systemctl start nginx
systemctl start php-fpm

```

You should be able to hit the default nginx welcome page and phpinfo.php to validate php is working.

http://<ip / hostname>/  
http://<ip / hostname>/phpinfo.php

Hope this was helpful! :)

# LIBHEIF INSTALL

I wanted this tool for HEIC image conversions on my own system. Once I got started trying to figure it all out, I found a whole world out there doing cool stuff with it. I also found that my Rocky 8 load was gunna give me some challenges. Here is a install process that I hope overcomes those challenges, for you.

**NEGATIVO17:** <https://negativo17.org>

**REPO:** <https://negativo17.org/repos>

```
# ----- FIND A NICE PLACE TO WORK -----

cd /opt

# ----- ADD NEGATIVO17 REPO -----

dnf config-manager --add-repo=https://negativo17.org/repos/epel-multimedia.repo
dnf -y update

# ----- INSTALL LIBRARIES AND TOOLS -----

dnf -y install \
binutils-x86_64-linux-gnu brotli brotli-devel cmake cmake-data cmake-doc cmake-filesystem cmake-rpm-macros \
doxygen ffmpeg ffmpeg-devel ffmpeg-libs fontconfig-devel freeglut-devel freetype-devel fribidi-devel \
gcc-c++-x86_64-linux-gnu gcc-gfortran gcc-go gcc-objc gcc-objc++ gcc-x86_64-linux-gnu gdk-pixbuf2-devel \
giflib giflib-devel git git-core git-core-doc glibc-static gmp-c++ graphviz graphviz-devel harfbuzz-devel \
ilmbase-devel iso-codes jbigkit-devel kvazaar kvazaar-devel kvazaar-libs libaom-devel libavcodec-devel \
libavfilter-devel libavformat-devel libavif libavutil-devel libdav1d-devel libgs-devel libjpeg libjpeg-devel \
libjpeg-turbo-utils libjxl-devel libopenjph-devel libpng libpng-devel librsvg2-devel libswscale \
libswscale-devel libtiff libtiff-devel libwebp-devel libwmf libwmf-devel libwmf-lite libXext libXext-devel \
meson nasm ninja-build OpenEXR-developenh264-developenh264-libspango-develperl-develpixmap-devel \
qt5-qtbase-gui qtsinglecoreapplication-qt5 qtsinglecoreapplication-qt5-develSDL2SDL2-develsvt-vp9-devel \
turbojpeg turbojpeg-devel vvdec vvdec-devel vvdec-libs vvenc vvenc-devel vvenc-libs x264-devel x264-libs \
x265-devel x265-libs yasm

# ----- INSTALL CARGO/RUSTUP -----
```



```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs -o rustup.sh
chmod +x rustup.sh
./rustup.sh --no-modify-path -y -q
rm -f rustup.sh

# ----- PRE-CREATE, CREATE, AND EXPORT MISSING PATHS -----

cat > /etc/ld.so.conf.d/local-lib.conf << "EOF"
/usr/local/lib
/usr/local/lib64
EOF

mkdir -p {/usr/local/lib64/pkgconfig,/usr/local/share/pkgconfig}
cat >> /root/.bashrc << "EOF"

PKGPATH=$(pkg-config --variable pc_path pkg-config);
PKG_CONFIG_PATH="$PKGPATH:/usr/local/lib64/pkgconfig:/usr/local/share/pkgconfig"
export PKG_CONFIG_PATH

PREPATH=$(echo $PATH);
CARPATH="$HOME/.cargo/bin"
PATH="$PREPATH:$CARPATH"
export PATH

EOF

# ----- CHECK YOUR BASHRC AND MAKE SURE THE NEW PATHS ARE GTG -----

cat /root/.bashrc

# ----- IF ALL IS GOOD, SOURCE AND LOAD -----

source /root/.bashrc
ldconfig
rustup update
```

The environment should be pretty well setup now. This next part is the meat and potatoes. I am going to keep these builds around for awhile so I created scripts for each one, and will locate them inside a `codecs` directory. This is a bit out of the norm if you're reading the "libheif" documentation as well. But you're here, so here it goes...

```
# ----- CREATE CODECS DIRECTORY -----
```

```
mkdir -p /opt/codecs && cd /opt/codecs
```

```
# ----- CLEAN INSTALL OF GCC 11 COMPILER, CLANG, AND ALL ITS TOOLS -----
```

```
dnf -y remove clang* gcc-toolset*
```

```
dnf -y install gcc-toolset-11* clang
```

```
# ----- GENERATE THE SCRIPTS -----
```

```
cat > libde265.sh << "EOF"
```

```
#!/usr/bin/env bash
```

```
git clone https://github.com/strukturag/libde265.git
```

```
cd libde265
```

```
./autogen.sh && ./configure --enable-encoder
```

```
make
```

```
make install
```

```
ldconfig
```

```
EOF
```

```
cat > openjpeg2.sh << "EOF"
```

```
#!/usr/bin/env bash
```

```
git clone -b v2.5.3 --depth 1 https://github.com/uclouvain/openjpeg.git
```

```
mkdir -p openjpeg/build
```

```
cd openjpeg/build
```

```
cmake .. -DCMAKE_BUILD_TYPE=Release -DBUILD_THIRDPARTY:BOOL=ON -DBUILD_CODEC:bool=on
```

```
make
```

```
make install
```

```
ldconfig
```

```
EOF
```

```
cat > aom.sh << "EOF"
```

```
#!/usr/bin/env bash
git clone -b v3.12.0 --depth 1 https://aomedia.googlesource.com/aom

cd aom
cmake -S . -B build.libavif -G Ninja \
-DMAKE_INSTALL_PREFIX="$(pwd)/dist" \
-DMAKE_BUILD_TYPE=Release \
-DENABLE_DOCS=0 \
-DENABLE_EXAMPLES=0 \
-DENABLE_TESTDATA=0 \
-DENABLE_TESTS=0 \
-DENABLE_TOOLS=0 \
-DMAKE_POSITION_INDEPENDENT_CODE=ON

ninja -C build.libavif
ninja -C build.libavif install

cd ..

A='#!/usr/bin/env bash'
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/aom/dist/lib64/pkgconfig"
C='PKGCONFPATH'

sed -i "s%$A%$A\n$B%" $C

EOF

cat > dav1d.sh << "EOF"
#!/usr/bin/env bash
git clone -b 1.5.0 --depth 1 https://code.videolan.org/videolan/dav1d.git

cd dav1d
meson build --default-library=static --buildtype release --prefix "$(pwd)/dist" $@
ninja -C build
ninja -C build install

cd ..

A='#!/usr/bin/env bash'
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/dav1d/dist/lib64/pkgconfig"
```

```
C='PKGCONFPATH'
```

```
sed -i "s%$A%$A\n$B%" $C
```

```
EOF
```

```
cat > libwebp.sh << "EOF"
```

```
#!/usr/bin/env bash
```

```
git clone --single-branch https://chromium.googlesource.com/webm/libwebp
```

```
mkdir -p ./libwebp/build
```

```
cd ./libwebp/build
```

```
cmake -G Ninja -DCMAKE_INSTALL_PREFIX="$(pwd)/dist" -DBUILD_SHARED_LIBS=OFF -
```

```
DCMAKE_BUILD_TYPE=Release ..
```

```
ninja sharpvuv
```

```
ninja install
```

```
cd ../..
```

```
A='#!/usr/bin/env bash'
```

```
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/libwebp/build/dist/lib64/pkgconfig"
```

```
C='PKGCONFPATH'
```

```
sed -i "s%$A%$A\n$B%" $C
```

```
EOF
```

```
cat > rav1e.sh << "EOF"
```

```
#!/usr/bin/env bash
```

```
git clone -b v0.7.1 --depth 1 https://github.com/xiph/rav1e.git
```

```
cd rav1e
```

```
cargo update
```

```
cargo install cargo-c
```

```
cargo cinstall --crt-static --release --prefix="$(pwd)/dist" --library-type=staticlib
```

```
cd ..
```

```
A='#!/usr/bin/env bash'
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/rav1e/dist/lib64/pkgconfig"
C='PKGCONFPATH'

sed -i "s%$A%$A\n$B%" $C

EOF

cat > svt.sh << "EOF"
#!/usr/bin/env bash
git clone -b v3.0.0 --depth 1 https://gitlab.com/AOMediaCodec/SVT-AV1.git

cd SVT-AV1/Build/linux

./build.sh release static no-apps disable-lto prefix=$(pwd)/install install

cd ../../..

A='#!/usr/bin/env bash'
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/SVT-AV1/Build/linux/install/lib64/pkgconfig"
C='PKGCONFPATH'

sed -i "s%$A%$A\n$B%" $C

EOF

cat > ultrahdr.sh << "EOF"
#!/usr/bin/env bash
git clone -b v1.4.0 --depth 1 https://github.com/google/libultrahdr.git ultrahdr

mkdir -p ./ultrahdr/build && cd ./ultrahdr/build

cmake -G Ninja -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX="$(pwd)/dist" -
DUHDR_BUILD_EXAMPLES=ON -DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ ..
ninja
ninja install

cd ../../..

A='#!/usr/bin/env bash'
```

```
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/ultrahdr/build/dist/lib64/pkgconfig"
```

```
C='PKGCONFPATH'
```

```
sed -i "s%$A%$A\n$B%" $C
```

```
EOF
```

```
# ----- QUICK SCRIPT TO RUN ALL THE SCRIPTS -----
```

```
cat > allofum.sh << "EOF"
```

```
#!/usr/bin/env bash
```

```
# ----- RUN THESE TWO SCRIPTS FIRST SO THEY ARE INSTALLED TO THE SYSTEM -----
```

```
cd /opt/codecs && source libde265.sh
```

```
cd /opt/codecs && source openjpg2.sh
```

```
# ----- NOW RUN THROUGH THE REST -----
```

```
cd /opt/codecs && source aom.sh
```

```
cd /opt/codecs && source dav1d.sh
```

```
cd /opt/codecs && source libwebp.sh
```

```
cd /opt/codecs && source rav1e.sh
```

```
cd /opt/codecs && source svt.sh
```

```
cd /opt/codecs && source ultrahdr.sh
```

```
exit
```

```
EOF
```

```
chmod +x *.sh
```

```
# ----- PRE-CREATE A "PKGCONFPATH" FILE -----
```

```
cat > PKGCONFPATH << "EOF"
```

```
#!/usr/bin/env bash
```

```
export PKG_CONFIG_PATH
```

```
EOF
```

```
# ----- RUN ALL OF UM -----
```

```
cd /opt/codecs && source allofum.sh
```

At this point all dependencies are in place. Moving on to "libheif"! This will require jumping into a newer GCC toolset as Rocky 8 rolls with GCC 8.5. We will use version 11 which we loaded earlier.

```
# ----- GRAB LIBHEIF -----
```

```
cd /opt
git clone https://github.com/strukturag/libheif.git
cd libheif
```

```
# ----- SWITCH TO THE GCC 11 ENVIRONMENT -----
```

```
scl enable gcc-toolset-11 bash      # HAD CASES WHERE THIS COMMAND WOULD NOT SWITCH OVER, SOOO...
source /opt/rh/gcc-toolset-11/enable # THROWING THIS ONE IN THE MIX AS WELL.
```

```
# ----- MAKE SURE YOUR GCC IS VERSION 11 -----
```

```
gcc --version
```

```
# ----- SOURCE AND LOAD OUR PATHS -----
```

```
source ~/.bashrc
source /opt/codecs/PKGCONFPATH
ldconfig
```

```
# ----- YOU CAN MIX AND MATCH OPTIONS - HERE ARE A COUPLE THAT I BUILT SUCCESSFULLY -----
```

```
## OPTION 1 ## ----- BUILD LIBHEIF RELEASE (SHARED) -----
```

```
mkdir build && cd build
cmake .. --preset=release \
-DWITH_UVG266=OFF \
-DENABLE_MULTITHREADING_SUPPORT=ON \
-DENABLE_PARALLEL_TILE_DECODING=ON
doxygen -u
make -j$(nproc)
```

```
## OPTION 2 ## ----- BUILD LIBHEIF FUZZING (STATIC) -----
```

```
mkdir build && cd build
cmake .. --preset=fuzzing \
-DWITH_UVG266=OFF \
-DENABLE_MULTITHREADING_SUPPORT=ON \
-DENABLE_PARALLEL_TILE_DECODING=ON
doxygen -u
make -j$(nproc)

# ----- I WILL BE USING THE "RELEASE" OPTION IN SOME OTHER WORK, SO I WILL INSTALL THAT ONE -----

make install

# ----- EXIT GCC TOOLKIT -----

exit
```

You should now have a monster libheif built on your system. It's a powerful app. Some of the tools to use it can be found in */usr/local/bin* (such as `heif-enc` and `heif-dec`).

I hope this helps! ;)



# IMAGICK WITH HEIC (nginx/php)

This write up will cover building ImageMagick with HEIC support and also building pecl's imagick in order to bring that functionality into PHP for use in web applications. A lot of documentation that has been put out on this topic seems to be exclusive to Ubuntu and/or just totally misses the PHP portion. In addition, the RHEL based world just wants to push everyone to OS version 9. I just can't want to right at this moment. So, this write up will cover that hole.

For the sake of brevity, I have already written a document on installing **nginx** and **php**, in addition a document on installing **libheif**. Please work through those write ups before continuing through on this one. You may already have a loaded web server and if so, just make sure that you cover the **libheif** install. That install gets the components in place that we will need in order to make this next procedure go smoothly (*libheif holds the keys to heic*).

Here are links to the previous two documents:

- [NGINX / PHP / MARIADB INSTALL](#)
- [LIBHEIF INSTALL](#)

If all is well... here we go.

**PECL:** <https://pecl.php.net>

**IMAGEMAGICK:** <https://imagemagick.org> (**GIT:** <https://github.com/ImageMagick/ImageMagick>)

```
#!/usr/bin/env bash

WRKDIR="/opt"
CODECDIR="/opt/codecs"
IMDIR="/opt/imagick"

# ----- JUMP ON OVER TO OUR HAPPY PLACE -----

cd ${WRKDIR}

# ----- STOP WEB SERVICES -----
```

```
systemctl stop nginx php-fpm
```

```
# ----- MAKE A DIRECTORY AND PULL DOWN THE SOURCE FILES -----
```

```
IMK="3.7.0"
```

```
mkdir imagick && cd imagick
```

```
wget https://www.imagemagick.org/download/ImageMagick.tar.gz
```

```
wget https://pecl.php.net/get/imagick-${IMK}.tgz
```

Ok. I need to interrupt myself for a sec.

If ImageMagick and/or the php-pecl-imagick are installed, they need to **NOT** be anymore. Remove them if they are.

```
# ----- MOVE OUT OF IMAGICK DIRECTORY -----
```

```
cd ~
```

```
# ----- DNF REMOVE IMAGEMAGICK AND IMAGICK -----
```

```
dnf -y remove *ImageMagick* *imagick*
```

OK. If verified that the system is free and clear of those packages, then, back to the not so regularly scheduled program.

```
# ----- UNPACK THE PACKAGES -----
```

```
cd ${IMDIR}
```

```
tar xzf imagick-${IMK}.tgz
```

```
tar xzf ImageMagick.tar.gz
```

```
IM7=$(tar -tzf ImageMagick.tar.gz | head -1 | cut -f1 -d"/");
```

```
# ----- SOURCE THE CODECS AND LOAD LIBRARIES -----
```

```
## NOTE: THE PATHS BELOW ARE TAKEN FROM MY PREVIOUS
```

```
## WRITE UP ON INSTALLING LIBHEIF.
```

```
## THIS PART ALONE IS KEY TO THE ENTIRE INSTALL!!
```

```
source /root/.bashrc
```

```
source /opt/codecs/PKGCONFPATH
```

```
ldconfig
```

```
# ----- INSTALL SOME LIBRARY/DEV PACKAGES (should have most of them from performing the libheif install)-----
```

```
dnf -y install \
```

```
fontconfig-devel \
```

```
freetype-devel \
```

```
graphviz-devel \
```

```
djvulibre-devel \
```

```
fftw-devel \
```

```
jbigkit-devel \
```

```
libgs-devel \
```

```
libjxl-devel \
```

```
liblqr-1 \
```

```
libraqm-devel \
```

```
LibRaw-devel \
```

```
librsvg2-devel \
```

```
libwmf-devel \
```

```
libzip-devel \
```

```
OpenEXR-devel \
```

```
OpenEXR-libs \
```

```
libtool-ltdl-devel
```

```
# ----- CONFIGURE IMAGEMAGICK -----
```

```
cd ${IMDIR}/${IM7}
```

```
./configure \
```

```
--with-bzlib \
```

```
--with-djvu \
```

```
--with-fftw \
```

```
--with-fontconfig \  
--with-freetype \  
--with-gslib \  
--with-gvc \  
--with-heic \  
--with-jbig \  
--with-jpeg \  
--with-jxl \  
--with-lqr \  
--with-lzma \  
--with-openexr \  
--with-openjp2 \  
--with-pango \  
--with-perl \  
--with-png \  
--with-raqm \  
--with-rsvg \  
--with-tiff \  
--with-uhdr \  
--with-webp \  
--with-wmf \  
--with-xml \  
--with-zip \  
--with-zlib \  
--with-zstd \  
--with-modules \  
--with-utilities \  
--enable-hdri \  
--enable-64bit-channel-masks \  
--enable-year2038 \  
--enable-legacy-support \  
--enable-hugepages
```

```
# ----- MAKE SURE THAT HEIC WAS FOUND (AND OTHERS YOU WANT) -----
```

```
make -j$(nproc)
```

```
make install
```

Now that ImageMagick is compiled up real good and installed. We just need to get the pecl imagick built and in its place.

```
# ----- CHANGE TO THE IMAGICK DIRECTORY -----

cd ${IMDIR}/imagick-${IMK}

# ----- WE NEED PHP HEADERS - INSTALL THE DEV PACKAGE -----

dnf -y install php-devel

# ----- PHP'IZE THE CODE -----

phpize

# ----- CONFIGURE, MAKE, AND INSTALL -----

./configure
make -j$(nproc)
make install

# ----- CREATE CONIG TO ENABLE IMAGICK WITHIN php.d AND php-zts.d -----

cat > /etc/php.d/40-imagick.ini << "EOF"
; Enable imagick extension module
extension=imagick.so

; Documentation: http://php.net/imagick

; Don't check builtime and runtime versions of ImageMagick
imagick.skip_version_check=1

; Fixes a drawing bug with locales that use ',' as float separators.
;imagick.locale_fix=0

; Used to enable the image progress monitor.
;imagick.progress_monitor=0
```

```
; multi-thread management
;imagick.set_single_thread => 1 => 1
;imagick.shutdown_sleep_count => 10 => 10

; to allow null images
;imagick.allow_zero_dimension_images => 0 => 0
EOF
```

```
cat > /etc/php-zts.d/40-imagick.ini << "EOF"
```

```
; Enable imagick extension module
extension=imagick.so
```

```
; Documentation: http://php.net/imagick
```

```
; Don't check builtime and runtime versions of ImageMagick
imagick.skip_version_check=1
```

```
; Fixes a drawing bug with locales that use ',' as float separators.
;imagick.locale_fix=0
```

```
; Used to enable the image progress monitor.
;imagick.progress_monitor=0
```

```
; multi-thread management
;imagick.set_single_thread => 1 => 1
;imagick.shutdown_sleep_count => 10 => 10
```

```
; to allow null images
;imagick.allow_zero_dimension_images => 0 => 0
EOF
```

```
# ----- COPY MODULE TO PHP-ZTS (so they match) -----
```

```
cp -a /usr/lib64/php/modules/imagick.so /usr/lib64/php-zts/modules/
```

```
# ----- LOAD LIBS AND RESTART SERVICES -----
```

```
ldconfig
```

```
systemctl restart nginx php-fpm
```



# EXIFTOOL

This will be super quick... like... and... go!

**ExifTool Source:** <https://github.com/exiftool/exiftool>

```
#!/usr/bin/env bash
# INSTALL EXIFTOOL (by Phil Harvey)
#
# rev. 20250324
#

VER="13.25"

cd /opt

dnf -y install perl-Time-Piece perl-IO-Compress perl-Archive-Extract-lzma-unlzma perl-Archive-Zip perl-Time-
Piece perl-Unicode-LineBreak
git clone -b ${VER} --depth 1 https://github.com/exiftool/exiftool.git

cd exiftool

perl Makefile.PL
make
make test && sleep 3
make install

ldconfig

exit
```

Phew! That was fun!