RHEL8 PACKAGING GUIDE

This guide covers building packages of the NVIDIA driver for Red Hat Enterprise Linux (RHEL) 8 and related derivatives.

1. Multiple driver branches are installable from a single package repository using "modularity streams". The user can choose a specific driver branch or a virtual branch. Only updates on the selected branch will be considered, where the `latest` and `latest-dkms` streams always update to the highest versioned driver release. While the `XXX` and `XXX-dkms` streams lock driver updates to the specified driver branch.
2. When using precompiled drivers, a *dnf plugin* is enabled that prevents upgrading to a kernel for which no precompiled driver exists yet (a warning will be displayed by `dnf` during such an upgrade situation). Stale .ko files are also cleaned up via this `dnf` plugin.
3. Special kernel module packages can be optionally built that implement an alternative to DKMS. The new approach does not require `gcc` to be installed anymore, nor does the EPEL repository need to be enabled. The source files for the driver kmod packages are compiled in advance and then linked at installation time, hence these are called "precompiled drivers".
4. For additional use cases, modularity profiles `default`, `ks`, `fm`, `src` (where applicable) can be combined.

Table of Contents

Prerequisites

Download inputs

1. NVIDIA driver runfile
2. NVIDIA modprobe tarball
3. NVIDIA persistenced tarball
4. NVIDIA settings tarball
5. NVIDIA xconfig tarball

NVIDIA driver runfile

- **Datacenter** location: http://us.download.nvidia.com/tesla/ (not browsable)

ex: http://us.download.nvidia.com/tesla/440.33.01/NVIDIA-Linux-x86_64-440.33.01.run

- **UDA** location: http://download.nvidia.com/XFree86/Linux-x86_64/

  ex: http://download.nvidia.com/XFree86/Linux-x86_64/440.64/NVIDIA-Linux-x86_64-440.64.run

- **GRID** runfiles: `NVIDIA-Linux-${arch}-${driver}-grid.run` are compatible.

  ex: NVIDIA-Linux-aarch64-455.04.18-grid.run

- **CUDA** runfiles: `cuda_${toolkit}_${driver}_linux.run` are not compatible.

  However a NVIDIA driver runfile can be extracted intact from a CUDA runfile:

  ```
  sh cuda_${toolkit}_${driver}_linux.run --tar mxvf
  > ex: sh cuda_11.2.2_460.32.03_linux.run --tar mxvf
  ```

  ```
  ls builds/NVIDIA-Linux-${arch}-${driver}.run
  > ex: ls builds/NVIDIA-Linux-x86_64-460.32.03.run
  ```

NVIDIA modprobe tarball

- **GitHub** location: https://github.com/NVIDIA/nvidia-modprobe/releases

  ex: https://github.com/NVIDIA/nvidia-modprobe/archive/460.32.03.tar.gz

- **UDA** location: https://download.nvidia.com/XFree86/nvidia-modprobe

  ex: https://download.nvidia.com/XFree86/nvidia-modprobe/nvidia-modprobe-460.56.tar.bz2

NVIDIA persistenced tarball

- **GitHub** location: https://github.com/NVIDIA/nvidia-persistenced/releases

  ex: https://github.com/NVIDIA/nvidia-persistenced/archive/460.32.03.tar.gz

- **UDA** location: https://download.nvidia.com/XFree86/nvidia-persistenced

  ex: https://download.nvidia.com/XFree86/nvidia-persistenced/nvidia-persistenced-460.56.tar.bz2

NVIDIA setttings tarball

- **GitHub** location: https://github.com/NVIDIA/nvidia-settings/releases

  ex: https://github.com/NVIDIA/nvidia-settings/archive/460.32.03.tar.gz

- **UDA** location: https://download.nvidia.com/XFree86/nvidia-settings

  ex: https://download.nvidia.com/XFree86/nvidia-settings/nvidia-settings-460.56.tar.bz2

NVIDIA xconfig tarball

- **GitHub** location: https://github.com/NVIDIA/nvidia-xconfig/releases

  ex: https://github.com/NVIDIA/nvidia-xconfig/archive/460.32.03.tar.gz

- **UDA** location: https://download.nvidia.com/XFree86/nvidia-xconfig

  ex: https://download.nvidia.com/XFree86/nvidia-xconfig/nvidia-xconfig-460.56.tar.bz2

Set global variables

notes:

- `$arch` is `x86_64`, `ppc64le`, or `aarch64` (sbsa)
- `$major` is the first `.` delimited field in the driver version,
  ex: `460` in `460.32.03`
- `$extension` is `bz2` OR `gz` depending on the tarballs downloaded
- `$KERNEL` is string including distro tag and architecture,
  ex: `4.18.0-193.28.1.el7.aarch64`
- Supports: `NVIDIA-Linux-${arch}-${version}-grid.run`

```
export version="460.32.03"
export VERSION="$version"
export major="460"
export arch="x86_64"
export extension="gz"
export KERNEL=$(uname -r)
export IGNORE_CC_MISMATCH=1
export RUN_FILE="/path/to/NVIDIA-Linux-*.run"
export OUTPUT="$HOME/rpm-nvidia"
mkdir -p "$OUTPUT"
```

Install build dependencies

note: Enable EPEL to install DKMS

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

note: store the package list in an array (easy `copy` & `paste`)

```
# Packaging
list=("rpm-build")

# Kernel modules (dkms-nvidia, precompiled-kmod)
list+=("dkms")
# Kernel headers and source code (precompiled-kmod)
list+=("kernel-headers-$KERNEL" "kernel-devel-$KERNEL")

# Compilation
list+=("m4" "gcc")
# Misc (nvidia-driver & nvidia-persistenced)
list+=("libappstream-glib" "libtirpc-devel")

# Python (nvidia-plugin)
list+=("python36")
# Repository metadata
list+=("createrepo" "openssl")

# Desktop integration (nvidia-settings)
list+=("gtk2-devel" "gtk3-devel" "jansson-devel" "dbus-devel" "desktop-file-utils")
# X.org utilties (nvidia-settings)
list+=("libXext-devel" "libXrandr-devel")
# GLVND (nvidia-settings)
list+=("mesa-libGL-devel" "mesa-libEGL-devel")
# Video extensions (nvidia-settings)
list+=("libXxf86vm-devel" "libXv-devel" "libvdpau-devel")

# Install all the build dependencies
sudo dnf install ${list[@]}
```

```
> ex: sudo dnf install -y rpm-build dkms m4 gcc \
kernel-headers-$KERNEL kernel-devel-$KERNEL \
libappstream-glib libtirpc-devel python36 createrepo openssl \
gtk2-devel gtk3-devel jansson-devel dbus-devel desktop-file-utils \
libXext-devel libXrandr-devel mesa-libGL-devel mesa-libEGL-devel \
libXxf86vm-devel libXv-devel libvdpau-devel
```

Clone git repositories

1. NVIDIA driver
2. DKMS nvidia
3. NVIDIA kmod common
4. NVIDIA modprobe
5. NVIDIA persistenced
6. NVIDIA settings
7. NVIDIA xconfig
8. NVIDIA plugin
9. NVIDIA precompiled kmod (optional)

note: for RHEL8-derivatives, checkout `rhel8` branch

```
git clone -b rhel8 https://github.com/NVIDIA/yum-packaging-nvidia-driver
git clone -b rhel8 https://github.com/NVIDIA/yum-packaging-dkms-nvidia
git clone -b rhel8 https://github.com/NVIDIA/yum-packaging-nvidia-kmod-common
git clone -b rhel8 https://github.com/NVIDIA/yum-packaging-nvidia-modprobe
git clone -b rhel8 https://github.com/NVIDIA/yum-packaging-nvidia-persistenced
git clone -b rhel8 https://github.com/NVIDIA/yum-packaging-nvidia-settings
git clone -b rhel8 https://github.com/NVIDIA/yum-packaging-nvidia-xconfig
git clone -b rhel8 https://github.com/NVIDIA/yum-packaging-nvidia-plugin
git clone -b rhel8 https://github.com/NVIDIA/yum-packaging-precompiled-kmod
```

Building packages

nvidia-driver

Generate tarballs from runfile

note: make sure `$VERSION` variable is set

```
cd yum-packaging-nvidia-driver
rm -rf temp
./nvidia-generate-tarballs.sh
```

note: please wait, this step will take several minutes to complete

```
ls *.tar.xz
> nvidia-driver-${version}-${arch}.tar.xz  # x86_64 script does not have -${arch} suffix
> nvidia-driver-${version}-i386.tar.xz     # 32-bit libraries for x86_64 only
> nvidia-kmod-${version}-${arch}.tar.xz    # not used here
```

rpmbuild

```
cd yum-packaging-nvidia-driver
mkdir BUILD BUILDROOT RPMS SRPMS SOURCES SPECS
cp *.conf SOURCES/
cp *.xml SOURCES/
cp nvidia-driver-${version}-${arch}.tar.xz SOURCES/
cp nvidia-driver.spec SPECS/

rpmbuild \
    --define "%_topdir $(pwd)" \
    --define "debug_package %{nil}" \
    --define "version $version" \
    --define "epoch 3" \
    --target "${arch}" \
    -v -bb SPECS/nvidia-driver.spec

find -name "*.rpm" -exec cp -v {} $OUTPUT/ \;
```

```
cd -
```

dkms-nvidia

nvidia-kmod tarball

- Copy tarball from `yum-packaging-nvidia-driver`

```
cd yum-packaging-dkms-nvidia
rsync -av ../yum-packaging-nvidia-driver/nvidia-kmod-${version}-${arch}.tar.xz $PWD/
cd -
```

or

- Generate tarball from runfile

```
cd yum-packaging-dkms-nvidia
sh "$RUN_FILE" --extract-only --target extract
mkdir nvidia-kmod-${version}-${arch}
mv extract/kernel nvidia-kmod-${version}-${arch}/
tar -cJf nvidia-kmod-${version}-${arch}.tar.xz nvidia-kmod-${version}-${arch}
cd -
```

rpmbuild

```
cd yum-packaging-dkms-nvidia
mkdir BUILD BUILDROOT RPMS SRPMS SOURCES SPECS
cp dkms-nvidia.conf SOURCES/
cp nvidia-kmod-${version}-${arch}.tar.xz SOURCES/
cp dkms-nvidia.spec SPECS/

rpmbuild \
    --define "%_topdir $(pwd)" \
    --define "debug_package %{nil}" \
    --define "version $version" \
    --define "epoch 3" \
    --target "${arch}" \
    -v -bb SPECS/dkms-nvidia.spec

find -name "*.rpm" -exec cp -v {} $OUTPUT/ \;
cd -
```

nvidia-kmod-common

rpmbuild

```
cd yum-packaging-nvidia-kmod-common
mkdir BUILD BUILDROOT RPMS SRPMS SOURCES SPECS
cp 60-nvidia.rules SOURCES/
cp 99-nvidia.conf SOURCES/
cp nvidia.conf SOURCES/
cp nvidia-kmod-common.spec SPECS/

rpmbuild \
    --define "%_topdir $(pwd)" \
    --define "debug_package %{nil}" \
    --define "version $version" \
    --define "epoch 3" \
    --target "noarch" \
    -v -bb SPECS/nvidia-kmod-common.spec

find -name "*.rpm" -exec cp -v {} $OUTPUT/ \;
cd -
```

nvidia-modprobe

rpmbuild

```
cd yum-packaging-nvidia-modprobe
mkdir BUILD BUILDROOT RPMS SRPMS SOURCES SPECS
cp ../nvidia-modprobe-${version}.tar.* SOURCES/
cp *.patch SOURCES/
cp nvidia-modprobe.spec SPECS/

rpmbuild \
    --define "%_topdir $(pwd)" \
    --define "debug_package %{nil}" \
    --define "version $version" \
    --define "epoch 3" \
    --define "extension $extension" \
    -v -bb SPECS/nvidia-modprobe.spec

find -name "*.rpm" -exec cp -v {} $OUTPUT/ \;
cd -
```

nvidia-persistenced

rpmbuild

```
cd yum-packaging-nvidia-persistenced
mkdir BUILD BUILDROOT RPMS SRPMS SOURCES SPECS
cp ../nvidia-persistenced-${version}.tar.* SOURCES/
cp *init* SOURCES/
cp *.service SOURCES/
cp nvidia-persistenced.spec SPECS/

rpmbuild \
    --define "%_topdir $(pwd)" \
    --define "debug_package %{nil}" \
    --define "version $version" \
    --define "epoch 3" \
    --define "extension $extension" \
    -v -bb SPECS/nvidia-persistenced.spec

find -name "*.rpm" -exec cp -v {} $OUTPUT/ \;
cd -
```

nvidia-settings

rpmbuild

```
cd yum-packaging-nvidia-settings
mkdir BUILD BUILDROOT RPMS SRPMS SOURCES SPECS
cp ../nvidia-settings-${version}.tar.* SOURCES/
cp *.desktop SOURCES/
cp *.patch SOURCES/
cp *.xml SOURCES/
cp nvidia-settings.spec SPECS/

rpmbuild \
    --define "%_topdir $(pwd)" \
    --define "debug_package %{nil}" \
    --define "version $version" \
    --define "epoch 3" \
    --define "extension $extension" \
    -v -bb SPECS/nvidia-settings.spec

find -name "*.rpm" -exec cp -v {} $OUTPUT/ \;
```

```
cd -
```

nvidia-xconfig

rpmbuild

```
cd yum-packaging-nvidia-xconfig
mkdir BUILD BUILDROOT RPMS SRPMS SOURCES SPECS
cp ../nvidia-xconfig-${version}.tar.* SOURCES/
cp *.patch SOURCES/
cp nvidia-xconfig.spec SPECS/

rpmbuild \
    --define "%_topdir $(pwd)" \
    --define "debug_package %{nil}" \
    --define "version $version" \
    --define "epoch 3" \
    --define "extension $extension" \
    -v -bb SPECS/nvidia-xconfig.spec

find -name "*.rpm" -exec cp -v {} $OUTPUT/ \;
cd -
```

nvidia-plugin

rpmbuild (dnf-plugin-nvidia)

```
cd yum-packaging-nvidia-plugin
mkdir BUILD BUILDROOT RPMS SRPMS SOURCES SPECS
cp nvidia-dnf.py SOURCES/
cp dnf-plugin-nvidia.spec SPECS/

rpmbuild \
    --define "%_topdir $(pwd)" \
    --define "debug_package %{nil}" \
    -v -bb SPECS/dnf-plugin-nvidia.spec

find -name "*.rpm" -exec cp -v {} $OUTPUT/ \;
cd -
```

precompiled-kmod

> *note:* this is an optional step

nvidia-kmod tarball

- Copy tarball from `yum-packaging-nvidia-driver`

```
cd yum-packaging-precompiled-kmod
rsync -av ../yum-packaging-nvidia-driver/nvidia-kmod-${version}-${arch}.tar.xz $PWD/
```

or

- Generate tarball from runfile

```
cd yum-packaging-precompiled-kmod
sh "$RUN_FILE" --extract-only --target extract
mkdir nvidia-kmod-${version}-${arch}
mv extract/kernel nvidia-kmod-${version}-${arch}/
tar -cJf nvidia-kmod-${version}-${arch}.tar.xz nvidia-kmod-${version}-${arch}
```

X.509 Certificate

- Generate X.509 `public_key.der` and `private_key.priv` files.

  Example x509-configuration.ini. Replace `$USER` and `$EMAIL` values.

  ```
  cd yum-packaging-precompiled-kmod
  openssl req -x509 -new -nodes -utf8 -sha256 -days 36500 -batch \
    -config x509-configuration.ini \
    -outform DER -out public_key.der \
    -keyout private_key.priv
  ```

Parse kernel string

```
export kernel_main=$(echo "$KERNEL" | awk -F "-" '{print $1}')
export kernel_suffix=$(echo "$KERNEL" | awk -F "-" '{print $2}' | sed "s|\.$arch||")
export kernel_dist=$(echo "$kernel_suffix" | awk -F "." '{print "."$NF}')
export kernel_release=$(echo "$kernel_suffix" | sed "s|\.$kernel_dist||")

> ex:
kernel_main="4.18.0"
kernel_release="240.22.1"
kernel_dist=".el8_3"
```

rpmbuild

note: compilation may take up to 10 minutes (depending on hardware)

```
cd yum-packaging-precompiled-kmod
mkdir BUILD BUILDROOT RPMS SRPMS SOURCES SPECS
cp nvidia-kmod-${version}-${arch}.tar.xz SOURCES/
cp public_key.der SOURCES/
cp private_key.priv SOURCES/
cp kmod-nvidia.spec SPECS/

# latest
rpmbuild \
    --define "%_topdir $(pwd)" \
    --define "debug_package %{nil}" \
    --define "kernel $kernel_main" \
    --define "kernel_release $kernel_release" \
    --define "kernel_dist $kernel_dist" \
    --define "driver $version" \
    --define "epoch 3" \
    --define "driver_branch ${major}"
    --target ${arch}
    -v -bb SPECS/kmod-nvidia.spec

find -name "*.rpm" -exec cp -v {} $OUTPUT/ \;
cd -
```

Create repository

Generate metadata

```
mkdir my-custom-repo
# NVIDIA driver packages
cp -v $OUTPUT/*.rpm my-custom-repo/
# Modularity script
cp -v yum-packaging-precompiled-kmod/genmodules.py $PWD/
```

```
createrepo_c -v --database my-custom-repo
```

```
python3 genmodules.py my-custom-repo modules.yaml
modifyrepo_c modules.yaml my-custom-repo/repodata/
```

Enable local repo

- **Create `custom.repo` file**

```
[custom]
name=custom
baseurl=file:///path/to/my-custom-repo
enabled=1
gpgcheck=0
```

- **Copy to system path for `yum` package manager**

```
sudo cp custom.repo /etc/yum.repos.d/
```

- **Clean `dnf` cache**

```
sudo dnf clean all
```

Pre-install actions

Remove any existing NVIDIA driver installation

- To uninstall a CUDA toolkit runfile installation

```
sudo /usr/local/cuda-X.Y/bin/cuda-uninstall
```

- To uninstall a standalone NVIDIA driver runfile installation:

```
sudo /usr/bin/nvidia-uninstall
```

- To uninstall an RPM installation:

```
sudo dnf module remove nvidia-driver
sudo dnf remove "*nvidia-driver*" "*nvidia-settings*"
sudo dnf module reset nvidia-driver
```

- To disable CUDA repository:

```
sudo dnf config-manager --set-disabled cuda
```

Package manager installation

- **RHEL8** streams: `latest`, `XXX`, `latest-dkms`, `XXX-dkms`

```
sudo dnf module install nvidia-driver:${stream}
> ex: sudo dnf module install nvidia-driver:latest
```

To use a modularity profile (default: `default`): `default`, `ks`, `fm`, `src`

```
sudo dnf module install nvidia-driver:${stream}/${profile}
> ex: sudo dnf module install nvidia-driver:465-dkms/ks
```

note: Multiple profiles can be combined

```
> ex: sudo dnf module install nvidia-driver:460/{fm,src}
```

Select an installation branch

To select an installation branch, choose only one from the four options below:

1. Always update to the highest versioned driver (precompiled).

   ```
   sudo dnf module install nvidia-driver:latest
   ```

2. Lock the driver updates to the specified driver branch (precompiled).

   ```
   sudo dnf module install nvidia-driver:XXX
   ```

   note: `XXX` is the first `.` delimited field in the driver version, ex: `460` in `460.32.03`

3. Always update to the highest versioned driver (*non-precompiled*).

   ```
   sudo dnf module install nvidia-driver:latest-dkms
   ```

   note: DKMS install uses compilation for `kmod-nvidia-latest-dkms` package (make take up to 10 minutes depending on hardware)

4. Lock the driver updates to the specified driver branch (*non-precompiled*).

   ```
   sudo dnf module install nvidia-driver:XXX-dkms
   ```

   note: DKMS install uses compilation for `kmod-nvidia-latest-dkms` package (make take up to 10 minutes depending on hardware)
   note: `XXX` is the first `.` delimited field in the driver version, ex: `460` in `460.32.03`

References

- Blog post: https://developer.nvidia.com/blog/streamlining-nvidia-driver-deployment-on-rhel-8-with-modularity-streams/
- Precompiled status page: https://developer.download.nvidia.com/compute/cuda/repos/rhel8/x86_64/precompiled/
- Presentations: https://github.com/NVIDIA/yum-packaging-precompiled-kmod#Presentations
- Report a bug: https://developer.nvidia.com/nvidia_bug/add

note: If you are not already a member, join the NVIDIA Developer Program: https://developer.nvidia.com/join

---