Precompiled Kernel Modules: Packaging & Deployment

Kevin Mittman, GTC Fall 2020

AGENDA

Precompiled kernel modules

NVIDIA driver compiled against specific kernel version string

DNF modules for NVIDIA driver

Choose a path depending on your use case

Open source package templates

Available on GitHub

note: This is a follow-up to a presentation at Red Hat Summit 2020

Precompiled kmod Packages

Introduction

Terminology as used in this presentation

- package: DEB, RPM, etc. file archive with pre/post install scriptlets
- package manager: apt/dnf/yum/zypper utility to install packages
- transitive closure: install or remove all packages in stream as one unit
- branch: driver builds from the same major version (ex: 418 or 440)

Precompiled kmod Packages

What are they?

Terminology as used in this presentation

- kmod: Linux kernel module, a set of loadable drivers
- DKMS: mechanism to re-compile out-of-tree modules on kernel update
- precompiled: pre-built NVIDIA drivers for a kernel (without linking)

Precompiled kmod Packages

Why would I want them over DKMS?

Benefits

- Removes *gcc* dependency ⇒ no compiler installation required
- Removes *dkms* dependency ⇒ EPEL repository not required
- Removes *kernel-*{*devel*,*headers*} deps ⇒ no black screen if missing[1]
- Pre-compiled ⇒ Faster boot up after driver and/or kernel updates
- Pre-tested ⇒ Kernel and driver combination has been validated

[1] Mismatched or forgetting to *yum/dnf install kernel-devel-$(uname -r) kernel-headers-$(uname -r)* is the most common NVIDIA driver installation issue. WIth the nouveau driver blacklisted, this can lead to Xorg display server unable to load.

Precompiled kmod Packages

Why would I NOT want them over DKMS?

Limitations

- Only official RHEL kernels supported by NVIDIA (no custom kernels[1])
- Driver version and kernel version string must match exactly
- Reliant on kmod package availability for each kernel update[2]

[1] Instructions for building precompiled packages for custom kernels using the .spec files on GitHub is discussed later in this presentation
[2] To avoid system breakage, a plugin for package manager will prevent install of kernel updates until compatible kmod package available

Precompiled kmod Packages

How does it work?

Building a kmod package

1. Compile .o files for NVIDIA kernel modules targeting a specific kernel.
2. Link the .o files against the kernel version string to build the .ko files
3. Sign .ko with X.509 certificate, detach the signature & delete the .ko[1]
4. Ship the .o files and detached signatures in the resulting RPM package

[1] Distributing proprietary binaries pre-linked against the Linux kernel would be a GPL violation

Precompiled kmod Packages

How does it work?

Installing a kmod package

1. Post-install script links the packaged .o files to reproduce the .ko files
2. Re-attach signature to sign[1] .ko files; verifies they match

[1] If certificate trusted, this would allow for UEFI Secure Boot support; currently not trusted.

Implementation on RHEL7

3 flavors of packages, specially crafted name scheme

kmod-latest-dkms	kmod-nvidia-latest-%{kernel}.r450.xx	kmod-nvidia-branch-418-%{kernel}.r418.xx
nvidia-driver-latest-dkms	nvidia-driver-latest	nvidia-driver-branch-418
nvidia-driver-latest-dkms-NVML	nvidia-driver-latest-NVML	nvidia-driver-branch-418-NVML
nvidia-driver-latest-dkms-NvFBCOpenGL	nvidia-driver-latest-NvFBCOpenGL	nvidia-driver-branch-418-NvFBCOpenGL
nvidia-driver-latest-dkms-cuda	nvidia-driver-latest-cuda	nvidia-driver-branch-418-cuda
nvidia-driver-latest-dkms-cuda-libs	nvidia-driver-latest-cuda-libs	nvidia-driver-branch-418-cuda-libs
nvidia-driver-latest-dkms-devel	nvidia-driver-latest-devel	nvidia-driver-branch-418-devel
nvidia-driver-latest-dkms-libs	nvidia-driver-latest-libs	nvidia-driver-branch-418-libs
nvidia-libXNVCtrl-latest-dkms	nvidia-libXNVCtrl-latest	nvidia-libXNVCtrl-branch-418
nvidia-libXNVCtrl-latest-dkms-devel	nvidia-libXNVCtrl-latest-devel	nvidia-libXNVCtrl-branch-418-devel
nvidia-modprobe-latest-dkms	nvidia-modprobe-latest	nvidia-modprobe-branch-418
nvidia-persistenced-latest-dkms	nvidia-persistenced-latest	nvidia-persistenced-branch-418
nvidia-settings-latest-dkms	nvidia-settings-latest	nvidia-settings-branch-418
nvidia-xconfig-latest-dkms	nvidia-xconfig-latest	nvidia-xconfig-branch-418
DKMS [1] Highest version	Precompiled [2] Highest version	Precompiled [2] Locked @ 418.x

[1] The yum-plugin-nvidia filters each flavor, which have slightly different package *Provides*, *Conflicts*, *Requires* dependencies
[2] Not officially supported on RHEL7

Implementation on RHEL8

List of packages

kmod-nvidia-${driver}-%{kernel}-${driver}	nvidia-kmod-common
kmod-nvidia-latest-dkms	nvidia-libXNVCtrl
nvidia-driver	nvidia-libXNVCtrl-devel
nvidia-driver-cuda	nvidia-modprobe
nvidia-driver-cuda-libs	nvidia-persistenced
nvidia-driver-devel	nvidia-settings
nvidia-driver-libs	nvidia-xconfig
nvidia-driver-NvFBCOpenGL	dnf-plugin-nvidia
nvidia-driver-NVML	

One set of packages [1]

Modularity is used to select streams

[1] This is the same mechanism for Fedora

Implementation on RHEL8

DNF plugin

Blocks kernel updates

- If opted into a precompiled stream, it hides kernel packages until a compatible kmod package is available in the repository
- Python script that hooks into DNF transactions
- Prevents system from getting to a state where NVIDIA driver is unable to load

DNF Modules

Modularity Streams

NVIDIA driver	Precompiled stream	Legacy DKMS stream
Highest version	latest	latest-dkms
Locked @ 450.x	450	450-dkms
Locked @ 440.x	440	440-dkms
Locked @ 418.x	418	418-dkms

$ sudo dnf module install nvidia-driver: <stream>

DNF Modules

Modularity Profiles

	Profile	Use case
Default	/default	Installs all the driver packages in a stream
Kickstart[1]	/ks	Unattended Linux OS installation via config file
NvSwitch	/fm	Installs all plus Fabric Manager and NSCQ

[1] %packages
@^Minimal Install
@nvidia-driver:latest-dkms/ks
%end

$ sudo dnf module install nvidia-driver: <stream> <profile>

DNF Modules

modules.yaml

modules.yaml

```yaml
document: modulemd
version: 2
data:
    name: nvidia-driver
    stream: latest
    version: 20200903080136
    arch: x86_64
    summary: Nvidia driver for latest branch
    description: >-
        This package provides the most recent NV
        hardware accelerated rendering with rece

        For the full product support list, pleas
        driver version 450.51.06.
```

```yaml
    artifacts:
        rpms:
            - nvidia-driver-3:450.51.06-1.el8.x86_64
            - nvidia-driver-libs-3:450.51.06-1.el8.x86_64
            - nvidia-driver-devel-3:450.51.06-1.el8.x86_64
            - nvidia-driver-NVML-3:450.51.06-1.el8.x86_64
            - nvidia-driver-NvFBCOpenGL-3:450.51.06-1.el8.x86_64
            - nvidia-driver-cuda-3:450.51.06-1.el8.x86_64
            - nvidia-driver-cuda-libs-3:450.51.06-1.el8.x86_64
            - nvidia-persistenced-3:450.51.06-1.el8.x86_64
            - nvidia-modprobe-3:450.51.06-1.el8.x86_64
            - nvidia-settings-3:450.51.06-1.el8.x86_64
            - nvidia-libXNVCtrl-3:450.51.06-1.el8.x86_64
            - nvidia-xconfig-3:450.51.06-1.el8.x86_64
            - nvidia-kmod-common-3:450.51.06-1.el8.noarch
            - cuda-drivers-0:450.51.06-1.x86_64
            - dnf-plugin-nvidia-0:1.6-1.el8.noarch
            - kmod-nvidia-450.51.06-4.18.0-193.14.3-3:450.51.06-2.el8_2.x86_64
    profiles:
```

Multiple YAML documents

Metadata for each stream embedded

List of each driver RPM

Cumulative over time, new kmod package for each driver + kernel combination

DNF Modules

modules.yaml

```
profiles:
    default:
        description: Default installation
        rpms:
            - cuda-drivers
            - nvidia-driver
            - nvidia-driver-NVML
            - nvidia-driver-NvFBCOpenGL
            - nvidia-driver-cuda
            - nvidia-driver-cuda-libs
            - nvidia-driver-devel
            - nvidia-driver-libs
            - nvidia-kmod-common
            - nvidia-libXNVCtrl
            - nvidia-modprobe
            - nvidia-persistenced
            - nvidia-settings
            - nvidia-xconfig
```

Default profile

Full stack of NVIDIA driver packages

```
        ks:
            description: Installation via kickstart
            rpms:
                - nvidia-driver
                - nvidia-driver-NVML
                - nvidia-driver-NvFBCOpenGL
                - nvidia-driver-cuda
                - nvidia-driver-cuda-libs
                - nvidia-driver-devel
                - nvidia-driver-libs
                - nvidia-kmod-common
                - nvidia-libXNVCtrl
                - nvidia-modprobe
                - nvidia-persistenced
                - nvidia-settings
                - nvidia-xconfig
...
---
document: modulemd
```

Kickstart profile (ks)

Does not install cuda-drivers[1] metapackage

```
        fm:
            description: FabricManager installation
            rpms:
                - cuda-drivers
                - nvidia-driver
                - nvidia-driver-NVML
                - nvidia-driver-NvFBCOpenGL
                - nvidia-driver-cuda
                - nvidia-driver-cuda-libs
                - nvidia-driver-devel
                - nvidia-driver-libs
                - nvidia-kmod-common
                - nvidia-libXNVCtrl
                - nvidia-modprobe
                - nvidia-persistenced
                - nvidia-settings
                - nvidia-xconfig
                - nvidia-fabricmanager-450
```

NvSwitch profile (fm)

Additional packages

[1] It attempts to cleanup existing driver RUNFILE installations with *nvidia-uninstall* via a `%pretrans` hook

Open Source Templates

RPM .spec files on GitHub

kmod-nvidia packages

- https://github.com/NVIDIA/yum-packaging-precompiled-kmod

dnf-plugin-nvidia & yum-plugin-nvidia packages

- https://github.com/NVIDIA/yum-packaging-nvidia-plugin

Coming soon

More driver packaging git repos with RPM .spec templates

Contributions welcome

Fork, commit, pull request

Open Source Templates

Build kmod-nvidia package

1. git clone -b $distro *https://github.com/NVIDIA/yum-packaging-precompiled-kmod*
2. Generate a X.509 certificate and copy into the repo
3. Build .spec with the appropriate parameters [1]

```
$ rpmbuild --define "%_topdir $(pwd)" --define "debug_package %{nil}" \
  --define "kernel $kernel" --define "kernel_release $release" \
  --define "kernel_dist $distro" --define "driver $version" --define "epoch 3" \
  --define "driver_branch $stream" -v -bb SPECS/kmod-nvidia.spec
```

4. Sign the RPM package with your GPG key

[1] For precompiled stream should be ‘latest’ or to lock to a branch ‘XXX’ (RHEL8) / ‘branch-XXX’ (RHEL7)

Open Source Templates

RPM repository

5. Copy {yum,dnf}-plugin-nvidia from the CUDA repository to RPMS/<arch>
6. Copy the rest of the driver packages (of same version & flavor) to RPMS/<arch>
7. Generate the repodata

```
$ createrepo_c -v --database .
```

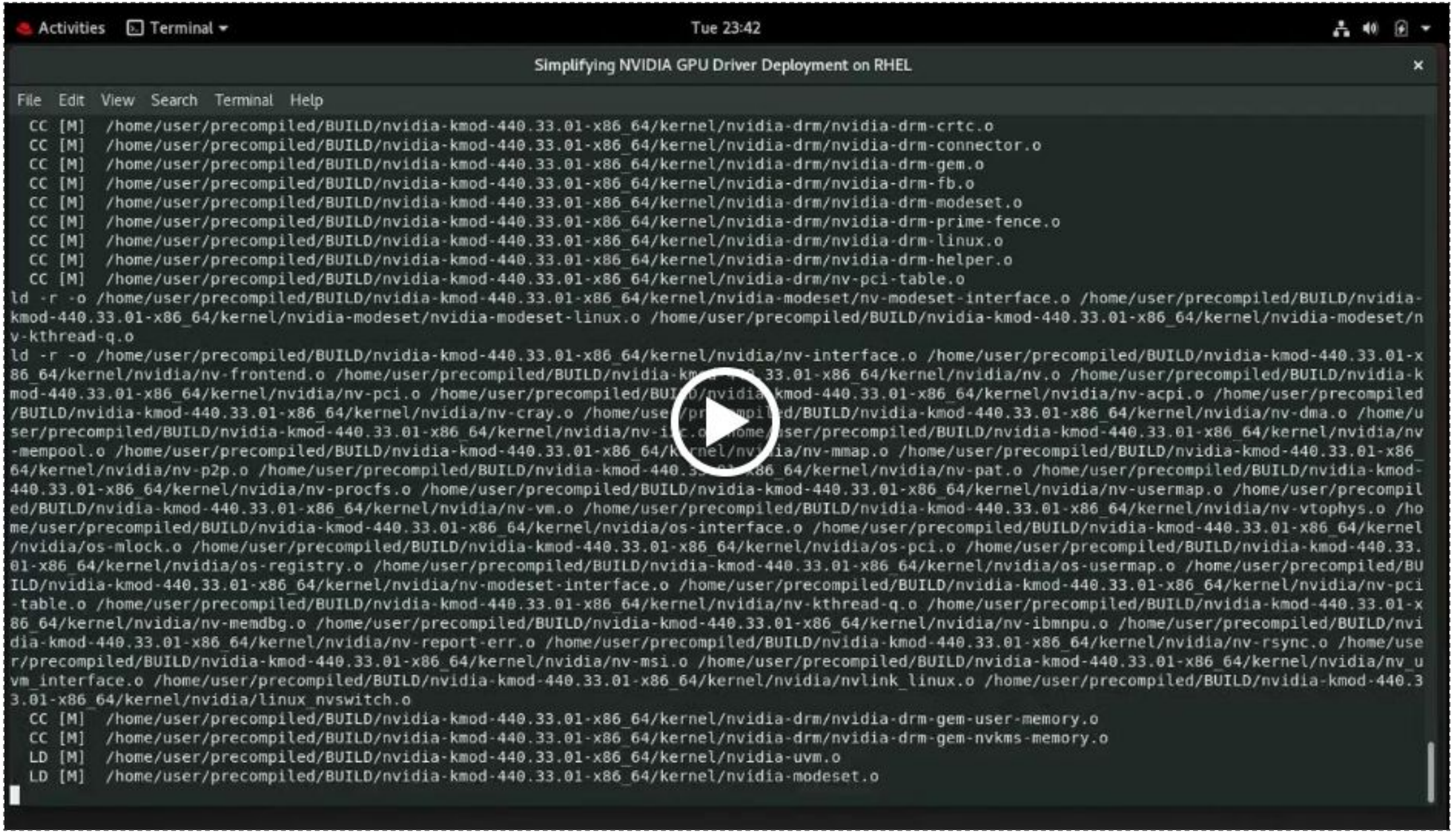
RHEL8 & Fedora

8. `$ python3 ./genmodules.py . modules.yaml`
9. `$ modifyrepo_c modules.yaml ./repodata`

Open Source Templates

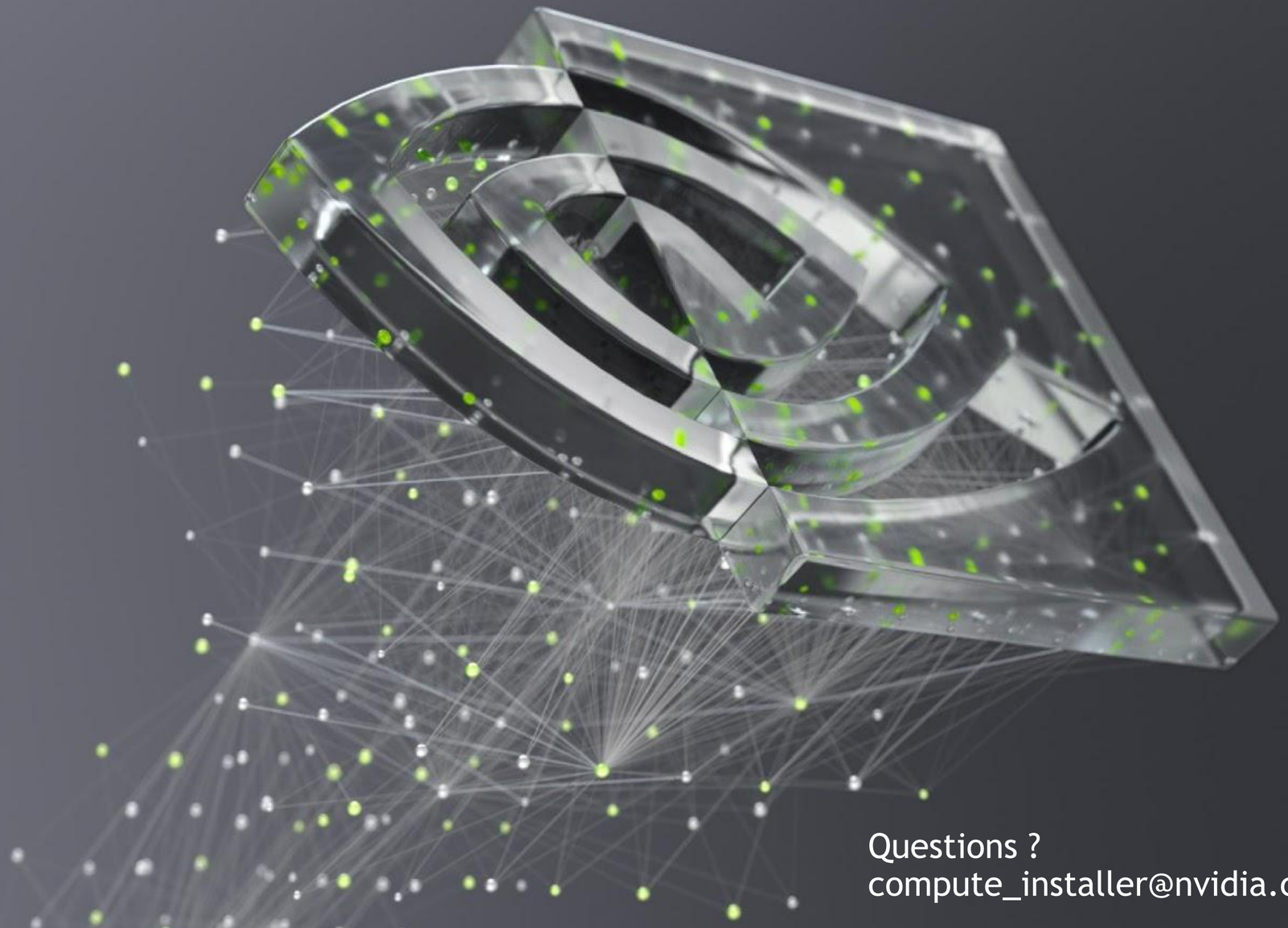
RPM repodata

- The package manager first locates repodata/repomd.xml
- It points to several filename-hashed metadata files, ex:
  - *-primary.xml.gz
  - *-modules.yaml.gz



Terminal recording (asciinema)

Link to these slides



Special Thanks

- Akshay Taneja
- Karthikeyan Somasundaram
- Harmandeep Singh
- Samhita Jayasimha
- Timm Bäder
- Torvald Riegel

Questions ?
compute_installer@nvidia.com

NVIDIA