ChatGPT

Jetson Linux Version Management Script

For anyone working with native Jetson hardware (i.e., system-on-module development kits and SoC modules) and planning to leverage NVIDIA Jetson software (e.g., the TAO Toolkit, DeepStream SDK, TensorRT, Jetson SDK) – either on Jetson devices or with an x86_64 host – this Bash script helps manage Jetson Linux kernel versioning. The goal is to streamline the process of updating a Jetson's kernel or entire Linux distribution, providing clarity for newcomers and flexibility for experienced users.

I made this Bash script tool for managing Jetson Linux kernel updates, and I hope it can shorten the time it takes you to get to the development stage of your project by automating and simplifying the upgrade/downgrade process.

> **Reminder:** The *jetson_kernel_manager.sh* script is divided into sections for clarity. You can run it on your Jetson device or Linux host, but you **must remember to make it executable (**`chmod +x jetson_kernel_manager.sh`**)**. The script should be run with administrative privileges (either as root or with `sudo` when it performs installation steps).

Overview

This comprehensive Bash script is designed to help users upgrade, downgrade, or rebuild the Linux kernel on NVIDIA Jetson devices in an interactive and user-friendly way. It guides the user through selecting a target Jetson Linux (L4T) version (JetPack version) and automates downloading required files, extracting sources, compiling the kernel, installing it, and backing up the current setup for easy reversion if needed. The script provides detailed explanations at each step by default (understandable even for a novice), with options to run silently or non-interactively as desired.

Key features of the script include:

- **Interactive Version Selection:** The user can input a Jetson Linux version, JetPack version, Ubuntu release, or Linux kernel version, and the script will intelligently interpret it (with robust handling of typos and various formats). For example, entering "JetPack 5.1.1" will automatically target Jetson Linux 35.3.1, and entering "Linux 5.10" will list all Jetson Linux releases using the 5.10 kernel (Jetson 35.x series) for the user to choose from.
- **Current System Detection:** The script checks the current Jetson device model and Jetson Linux version installed (e.g., by reading `/etc/nv_tegra_release`) and informs the user. It supports Jetson Linux releases 35.3.1 and newer (JetPack 5.1.1+), which correspond to Ubuntu 20.04 or 22.04 based systems. *(Older releases are not supported by this script.)*
- **Supported Devices:** All Jetson modules from the AGX Xavier/NX family up through the Orin family are supported. This includes Jetson AGX Xavier (and Xavier Industrial), Jetson Xavier NX, and the Jetson Orin series (AGX Orin, Orin NX, Orin Nano). *(Jetson AGX Orin Industrial is supported only on Jetson Linux 35.4.1 and later – the script will prevent selecting Jetson Linux 35.3.1 for that module, because support for the Industrial module was added in 35.4.1.)*

- **Automated Download Prompts:** If the required NVIDIA Jetson Linux Driver Package (BSP) and Kernel Sources tarball files for the target version are not found in `~/Downloads`, the script will pause and instruct the user exactly where to download them. It constructs a direct link to NVIDIA's Jetson Linux archive page for the chosen version, so the user can easily find the files. For example, if targeting Jetson Linux 36.4.3 (JetPack 6.2), the script will direct the user to the Jetson Linux R36.4.3 download page to obtain `Jetson_Linux_R36.4.3_aarch64.tbz2` (the BSP) and `public_sources.tbz2` (the source package).
- **Kernel Build Automation:** The script unpacks the downloaded tarballs, prepares the kernel source, and then compiles the kernel, device tree blobs (DTBs), and modules with the appropriate architecture and flags. It uses NVIDIA's default kernel configuration (`tegra_defconfig`) to ensure compatibility. By default, the script sets the kernel local version suffix to "-tegra" (the same suffix NVIDIA uses) for consistency. This way, the new kernel and modules align with NVIDIA's naming (for example, a Jetson Linux 36.4.4 kernel identifies as version 5.15.xxx-tegra on Ubuntu 22.04).
- **Installation and Reboot Guidance:** After a successful build, the script installs the new kernel (`Image`), updates device tree files, and installs kernel modules to `/lib/modules/<version>`. It also updates the initramfs (initial RAM disk) so that the system can boot with the new kernel's modules. The user is then prompted to reboot the device to load the new kernel. *(The actual reboot is left to the user's confirmation.)*
- **Backup and Revert Capability:** Before making any changes, the script creates a backup of the current kernel, modules, and device tree files. These are stored under `/usr/local/src/L4T` in a folder named after the current version (for example, `35.3.1_backup`). If something goes wrong or the user wants to revert to the previous Jetson Linux version, the script provides a `--revert` option that will restore all the backed-up files (kernel, DTBs, modules) to their original locations, effectively rolling back the system to the previous state. This makes experimenting with different Jetson Linux versions much safer.
- **Verbose and Quiet Modes:** By default the script is verbose, printing clear, beginner-friendly explanations before each major step or decision. For advanced users, a quiet mode (`-q` flag) is available to suppress these informative messages and only show essential outputs. All interactive prompts (e.g., "Continue? (Y/n)") can be bypassed with a "yes to all" flag (`-y`), enabling fully non-interactive operation once the target version is specified.
- **Help and Usage Info:** The script includes a `--help` flag that outputs usage instructions and explains all available options. At the start, it also reminds the user that they can run the script with `-h` to see the help message, ensuring users know how to get guidance.
- **Cross-Platform Considerations:** While the script is primarily intended to run directly on a Jetson device, it can also be used on an x86_64 host for cross-compiling the Jetson kernel (with appropriate toolchains installed). If run on a non-Jetson host, the script will warn the user and require that a cross-compilation toolchain (like the Linaro aarch64 GCC) is set up. In cross-build mode, the script will compile the kernel but will not attempt to install it on the host. Instead, it will instruct the user on how to deploy the new kernel and modules to the Jetson device (for example, by copying the output files or using NVIDIA's flash tools).

Overall, this script aims to streamline the process of updating a Jetson's kernel or entire Linux version, providing both clarity for newcomers and flexibility for power users.

(In the following sections, we explain the script's implementation details and usage examples.)

Supported Jetson Linux Versions and Mapping

It's important to understand the relationship between Jetson Linux (L4T) release versions, JetPack SDK versions, and the underlying Ubuntu OS and Linux kernel. The script uses this information to interpret user input and confirm the target environment. Here are the supported releases and their key characteristics (all JetPack 5.1.1+ corresponding to Jetson Linux 35.3.1 and later):

- **Jetson Linux 35.3.1** – corresponds to JetPack 5.1.1. Uses Linux kernel 5.10 on Ubuntu 20.04. This was a production release adding Orin Nano support.
- **Jetson Linux 35.4.1** – corresponds to JetPack 5.1.2. Still uses kernel 5.10 (LTS) on Ubuntu 20.04. This release added support for the Jetson AGX Orin Industrial module.
- **Jetson Linux 36.2.0** – corresponds to JetPack 6.0 (Developer Preview). Introduced Linux kernel 5.15 (a new LTS kernel) and upgraded to an Ubuntu 22.04 base. *(JetPack 6.0 DP was not a production release and lacked some security/OTA features.)*
- **Jetson Linux 36.4.0** – corresponds to JetPack 6.1 (first production release of JetPack 6). Uses Linux kernel 5.15 on Ubuntu 22.04.
- **Jetson Linux 36.4.3** – corresponds to JetPack 6.2. Still kernel 5.15 on Ubuntu 22.04, with additional improvements and bug fixes.
- **Jetson Linux 36.4.4** – corresponds to JetPack 6.2.1. A minor update on kernel 5.15/Ubuntu 22.04, focused on bug fixes and security features.

When the user inputs a JetPack version, the script translates it to the corresponding Jetson Linux release. For example, JetPack 6.2.1 → Jetson Linux 36.4.4, JetPack 6.2 → Jetson Linux 36.4.3, JetPack 6.1 → Jetson Linux 36.4.0, JetPack 5.1.2 → Jetson Linux 35.4.1, and JetPack 5.1.1 → Jetson Linux 35.3.1. Likewise, if the user provides an Ubuntu version (20.04 or 22.04) or a kernel version (5.10 or 5.15), the script knows which Jetson Linux releases apply. *(For example, kernel 5.10 implies Jetson Linux 35.3.1 or 35.4.1, whereas kernel 5.15 implies one of the 36.x releases.)* This mapping data is built into the script for accuracy.

Script Structure and Flow

For clarity, the script is organized into logical sections and functions. It performs pre-run checks, parses arguments, interacts with the user for target version selection, then proceeds to download verification, extraction, the build process, and finally the installation and backup/revert steps. Each section prints informative messages (if not in quiet mode) to explain what is happening. Here's a high-level outline of what the script does, in order:

1. **Initialization & Help:** Defines usage information and parses command-line options (`-h/--help`, `-q/--quiet`, `-y/--yes`, `--revert`, `-t/--target`, `--dry-run`). If `--help` is used, it prints usage instructions and exits. If `--revert` is used, it triggers the revert routine (restoring a previous backup) and then exits.
2. **Pre-checks:** Determines the current system's Jetson model and Jetson Linux version. It warns if run on a non-Jetson (e.g., running on an x86_64 host) and prepares for cross-compiling in that case (requires the user to have an ARM64 cross-toolchain and to set the `CROSS_COMPILE` environment variable). It also ensures required tools like `tar`, `make`, etc., are available, and that the user has root privileges or sudo access for later steps.

3. **Version Selection:** If the target Jetson Linux version was not already provided via `-t`, the script interacts with the user to determine the target. It prompts the user to input a desired version or JetPack. The input is then normalized and validated:
4. The script accepts inputs like “36.4.4”, “JetPack 6.2.1”, “L4T 35.4.1”, “Ubuntu 22.04”, “kernel 5.10”, etc., and will smartly interpret these.
5. If the input is incomplete or matches multiple possibilities, the script will list the matching versions. For example, entering “5.10” (to target kernel 5.10) would result in two options: Jetson Linux 35.3.1 (JetPack 5.1.1) and Jetson Linux 35.4.1 (JetPack 5.1.2), since both use a 5.10 kernel. The user can then choose one of the options.
6. After resolving the input to a specific Jetson Linux release (e.g., 36.4.4), the script displays a summary of that version including its JetPack number, kernel version, and Ubuntu OS base. For example: “Jetson Linux 36.4.4 (JetPack 6.2.1) uses Linux kernel 5.15 and Ubuntu 22.04.” The user is asked to confirm if this is correct before proceeding.
7. **Tarball Download Check:** NVIDIA distributes Jetson Linux in two main tarball files per release: the BSP (Board Support Package) and the Kernel Sources. The script constructs the expected filenames and checks if they are present in the `~/Downloads` directory:
8. **BSP tarball** – `Jetson_Linux_R<version>_aarch64.tbz2` (for example, `Jetson_Linux_R36.4.4_aarch64.tbz2` for Jetson Linux 36.4.4).
9. **Kernel source tarball** – `public_sources.tbz2` (this name is the same for all versions).

If either file is missing, the script will explain which files need to be downloaded and from where. It provides the URL to the specific Jetson Linux release page on NVIDIA’s developer site where the user can find the downloads. The user is prompted to download the files, then press **Y** to continue once done. The script will pause until the user confirms and will re-check the `~/Downloads` folder. *(If the files still aren’t found, it will either repeat the prompt or abort, depending on user input.)*

Note: The script does not automatically download the files for the user (since NVIDIA’s download links often require login). Instead, it guides the user to the official download page. *(Checksum verification of the downloads is planned for the future – the script has placeholders for it – but it is not implemented in this version.)* 5. **Extraction and Preparation:** Once the tarballs are available, the script creates a working directory under `/usr/local/src/L4T` for the target version (for example, `/usr/local/src/L4T/36.4.4/`). It then: - Extracts the Jetson Linux BSP tarball into that directory, yielding a `Linux_for_Tegra` folder containing binaries, flashing tools, and a sample root filesystem. - Extracts the `public_sources.tbz2` into the same directory, populating the `Linux_for_Tegra/source/public/` subdirectory with source code packages. - Unpacks the kernel source package (found at `Linux_for_Tegra/source/public/kernel_src.tbz2`). After this, the full kernel source tree is available (e.g., under `Linux_for_Tegra/source/public/kernel/kernel-5.15/` for a 5.15 kernel).

If the working directory for that version already exists (from a previous run), the script will reuse it to save time (skipping re-extraction of files that are already there). It will still ensure the kernel build starts from a clean state by using a separate output directory for compilation. 6. **Kernel Compilation:** The script sets up the build environment and compiles the kernel: - It uses an out-of-tree build approach (using the `O=<build_dir>` make parameter) to keep the source directory clean. All compiled output goes into a dedicated build directory (e.g., `/usr/local/src/L4T/36.4.4/kernel_build/`). - The script applies the default Jetson kernel configuration by running `make ARCH=arm64 tegra_defconfig`. This ensures the `.config` matches NVIDIA’s official configuration for the release. *(Alternatively, if you wanted to preserve the current system’s exact config, you could extract it from* `/proc/config.gz`*, but using* `tegra_defconfig` *is*

the standard method to get NVIDIA's supported config.) - It then invokes `make ARCH=arm64 -j<n>` to build the kernel Image, modules, and device tree blobs (DTBs) all at once. The `-j<n>` flag uses multiple cores to speed up compilation (where *n* is the number of CPU cores on the system). This step can take some time, especially on a large kernel like 5.15 with debug features. The script provides feedback and only proceeds when the build is successful. If any compilation error occurs, the script will stop and report the failure.

During cross-compiling on a PC (if taking that route), the script will set the appropriate cross-compilation variables (e.g., `CROSS_COMPILE`) and then follow the same steps. It will not attempt to install files to the host system directories. Instead, it will build the kernel and then advise the user on how to manually deploy the resulting files to the Jetson (for example, by copying the Image and modules, or by creating a package). 7. **Installation of Kernel and Modules:** After a successful build, the script installs the new kernel and related files to the Jetson's filesystem (when running on the Jetson itself): - It backs up the current kernel, DTBs, and modules before replacing them (see the next step for details on backups). The new kernel Image is copied to `/boot/Image`, overwriting the existing one (this is the file the bootloader uses to boot the kernel). All new device-tree `.dtb` files are copied into `/boot/dtb/`, replacing the old hardware configuration files. - The kernel modules are installed by running `make modules_install`. This places the new kernel's modules under `/lib/modules/<kernel-version>/` (for example, `/lib/modules/5.15.xx-tegra/`). The script then runs `depmod` to update the module dependency listings for the new kernel version, ensuring that module loading works properly. - Next, the initramfs is updated. The initramfs (initial RAM filesystem) is a boot-time disk image that may contain drivers needed during early boot. The script generates a new initramfs including the new modules by calling `update-initramfs -c -k <new-version>`. On Ubuntu, this creates a file like `/boot/initrd.img-<new-version>`. The script then updates the default initrd symlink (or config) to point to this new file so that the bootloader will use the updated initramfs. *(The original initrd is saved in the backup in case a revert is needed.)* This step ensures that if any crucial drivers (for example, for storage or filesystems) are modular, they will be present at boot time for the new kernel.

Note: If the script is run in cross-compile mode on a host PC, it skips the installation steps on the host. Instead, it will advise the user on which files to copy to the Jetson's `/boot` and how to install the modules on the Jetson. Typically this involves copying the new `Image` and DTBs into the Jetson's `/boot` directory and transferring the module folder (or packaging it as NVIDIA does in a `kernel_supplements.tbz2`). 8. **Backup and Revert:** The script maintains backups to enable easy restoration of the previous state: - Before the new kernel is installed, the script creates a backup directory (e.g., `/usr/local/src/L4T/35.3.1_backup/` for a system that was on 35.3.1) and copies into it the existing `/boot/Image`, `/boot/initrd`, the entire `/boot/dtb/` folder, and the `/lib/modules/<old-version>/` directory. This backup captures all components of the running system's kernel. - If the user wants to revert to the previous Jetson Linux version (for example, if the new kernel is not working as expected), they can run the script again with the `--revert` option. The script will detect the available backup(s) in `/usr/local/src/L4T` and either automatically choose the most recent one or present a list if multiple backups are found. It will then restore the backed-up files to their original locations: the old kernel Image, DTBs, initrd, and modules will be copied back. After a reboot, the Jetson will be exactly as it was before the update. The script issues warnings and ensures the user really wants to revert before proceeding, since this will overwrite the currently installed kernel. - The backup mechanism is designed to be robust: even if you upgrade through multiple versions (say from 35.3.1 to 35.4.1, then to 36.4.4), each step's original files are saved in its own backup folder. You can revert step-by-step or all the way back, as needed. *(Note: User-space libraries from different JetPack versions are not automatically handled by this script – it focuses on the Linux kernel and low-level BSP components. Minor JetPack updates should not break functionality if only the kernel is swapped, but major*

changes (like Ubuntu 20.04 vs 22.04) could have user-space incompatibilities. The script assumes kernel customization or minor version transitions as the primary use cases.)

Usage Example

Suppose you have a Jetson AGX Orin running Jetson Linux 35.3.1 (JetPack 5.1.1) and you want to upgrade to Jetson Linux 36.4.4 (JetPack 6.2.1) to take advantage of the new Ubuntu 22.04 base and kernel 5.15. Here's how you could use the script:

1. Download the files `Jetson_Linux_R36.4.4_aarch64.tbz2` (the BSP) and `public_sources.tbz2` (the kernel sources) from NVIDIA's Jetson Linux 36.4.4 download page. (The script will provide you the exact link and filenames needed.)
2. Place those files into your `~/Downloads` directory on the Jetson.
3. Run the script:

   ```
   sudo ./jetson_kernel_manager.sh
   ```

4. When prompted for the target version, enter **JetPack 6.2.1** (or **36.4.4** – the script will recognize either). The script will confirm that this corresponds to Jetson Linux 36.4.4 with kernel 5.15 on Ubuntu 22.04.
5. The script will then detect the tarballs, extract them, compile the new kernel and modules, back up your current kernel and modules (saving them under `/usr/local/src/L4T/35.3.1_backup`), and install the new kernel and modules.
6. Reboot the Jetson. It should now boot into Jetson Linux 36.4.4. You can verify by running `uname -a` (to check the kernel version) and `cat /etc/nv_tegra_release` (to see the L4T release string, which should show "R36, Revision 4.4").
7. If anything goes wrong or you need to revert to the previous 35.3.1 environment, run the script with the revert option:

   ```
   sudo ./jetson_kernel_manager.sh --revert
   ```

   The script will find the backup for 35.3.1 and restore all those files. After a reboot, you'll be back on the old JetPack 5.1.1 setup, as if nothing happened.

Conclusion

This script simplifies managing Jetson Linux versions by automating the tedious parts and wrapping them in a safe, interactive workflow. It ensures that users are clearly informed of what is happening at each step (with the ability to suppress details for advanced users) and that they have an easy escape route (through backups and the revert functionality) if something doesn't work out. By handling version mapping and file management, the script helps avoid common pitfalls (such as using the wrong kernel sources or forgetting to update the initramfs), saving developers time and effort. Users can confidently experiment with new JetPack releases or custom kernel builds, knowing they can easily revert to a known-good state if needed.
