Understanding Package Management in Linux

What Are Repositories?

In the context of Linux, repositories are online or local storage locations where software packages and their metadata are stored. These packages can be freely accessed and managed through package managers like `apt` for Debian-based systems (Ubuntu). Repositories ensure users have access to the latest software versions and updates.

Purpose of `/etc/apt/sources.list`

The `/etc/apt/sources.list` file on Debian-based systems lists the "sources" or repositories that the system uses to fetch software packages. This file can contain links to multiple repositories, which might include the main Ubuntu repositories, third-party repositories, or even private repositories for specific software. Entries in this file dictate from where `apt` can install or upgrade packages.

Cheatsheet: Common `apt` Commands and Flags

- `sudo apt update` - Updates the package index files from their sources. It should be run before new installations or upgrades to ensure you're working with the latest package listings.
- `sudo apt install [package-name]` - Installs a new package along with its dependencies.
- `Sudo apt install -y [package-name]` - The `-y` flag automatically answers "yes" to prompts, streamlining automated scripts or batch installs.
- `sudo apt upgrade` - Upgrades all upgradable packages on the system.
- `sudo apt-get remove [package-name]` - Removes a package but leaves configuration files intact.
  When you remove software using `sudo apt-get remove [package-name]`, the package manager deletes the binary files associated with the package but retains configuration files. These configuration files are typically stored in `/etc` or a subdirectory within it, depending on the package.

  Why Keep Configuration Files? This behavior is intended to preserve custom settings. If you reinstall the package later, you won't need to reconfigure it from scratch. This is particularly useful in environments where installations are temporary or experimental but where retaining settings is beneficial.

Example: If you remove the `nginx` web server but plan to reinstall it later:

```
sudo apt-get remove nginx
```

This command deletes the nginx executable but keeps the server configuration files located typically in `/etc/nginx`. If you reinstall nginx, it will use the same settings configured previously.

Adding a Repository Manually and Verifying Installation

Sometimes the software you need is not available in the default repositories provided by your Linux distribution. In such cases, you can add third-party repositories. This three-step instruction outlines how to securely add a custom repository and install software from it.

Repository Keys: Repository keys are part of a security mechanism to verify the integrity and origin of packages. By adding a repository key, you're telling your system to trust packages signed with this key.

Let’s say we want to install package named ‘example-package’, which source is outside default package repositories trusted and used by apt package manager.

To add a repository manually and ensure it's trusted, follow these steps

1. **Add Repository Key**: Download the package via curl. Download Secure APT uses keys to ensure package integrity.
   This reads a GPG key from standard input (provided by a preceding command, typically a `curl` command fetching the key as in the command below) and adds it to the system's list of trusted keys. This ensures packages from the corresponding repository are authenticated and thus safe to install.

```
curl -fsSL https://download.example.com/gpg | sudo apt-key add -
```

2. **Add Repository to `sources.list`:**

```
echo "deb [arch=amd64] https://download.example.com/ubuntu $(lsb_release -cs)
main" | sudo tee /etc/apt/sources.list.d/example.list
```

The `deb` prefix indicates that the line is specifying a repository for Debian package sources. `[arch=amd64]` specifies that the repository is intended for 64-bit systems. You should choose the architecture that matches your system, commonly `amd64` for modern systems.

`lsb_release -cs` prints the codename of the Linux distribution on which the command is run. This helps in specifying the correct version of the repository tailored to your distribution version, ensuring compatibility.

The `tee` command reads from standard input and writes both to standard output and the file specified. Here, it is used to write the repository details to a new file in `/etc/apt/sources.list.d/`, which `apt` checks to know where to find packages. This way, the repository is added without having to manually edit the file, enhancing both safety and convenience.

3. **Update and install package:**

```
sudo apt update
sudo apt install example-package
```

The "example-package" in `sudo apt install example-package` ties back to the previous steps where the new repository was added. `apt` knows where to find this package due to the entry added in `/etc/apt/sources.list.d/`. When you run `sudo apt update`, `apt` updates its package database, fetching package lists from all repositories, including the newly added one. This process allows `apt` to recognize "example-package" and any other packages available through the new repository.