

# **Linux Administration**

## **Software management**

Xavier Belanger

**This work is licensed under  
a Creative Commons Attribution-ShareAlike 4.0 International License.**

<http://creativecommons.org/licenses/by-sa/4.0/>

**You are free to:**

- **Share** — copy and redistribute the material in any medium or format
- **Adapt** — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

**Under the following terms:**

- **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

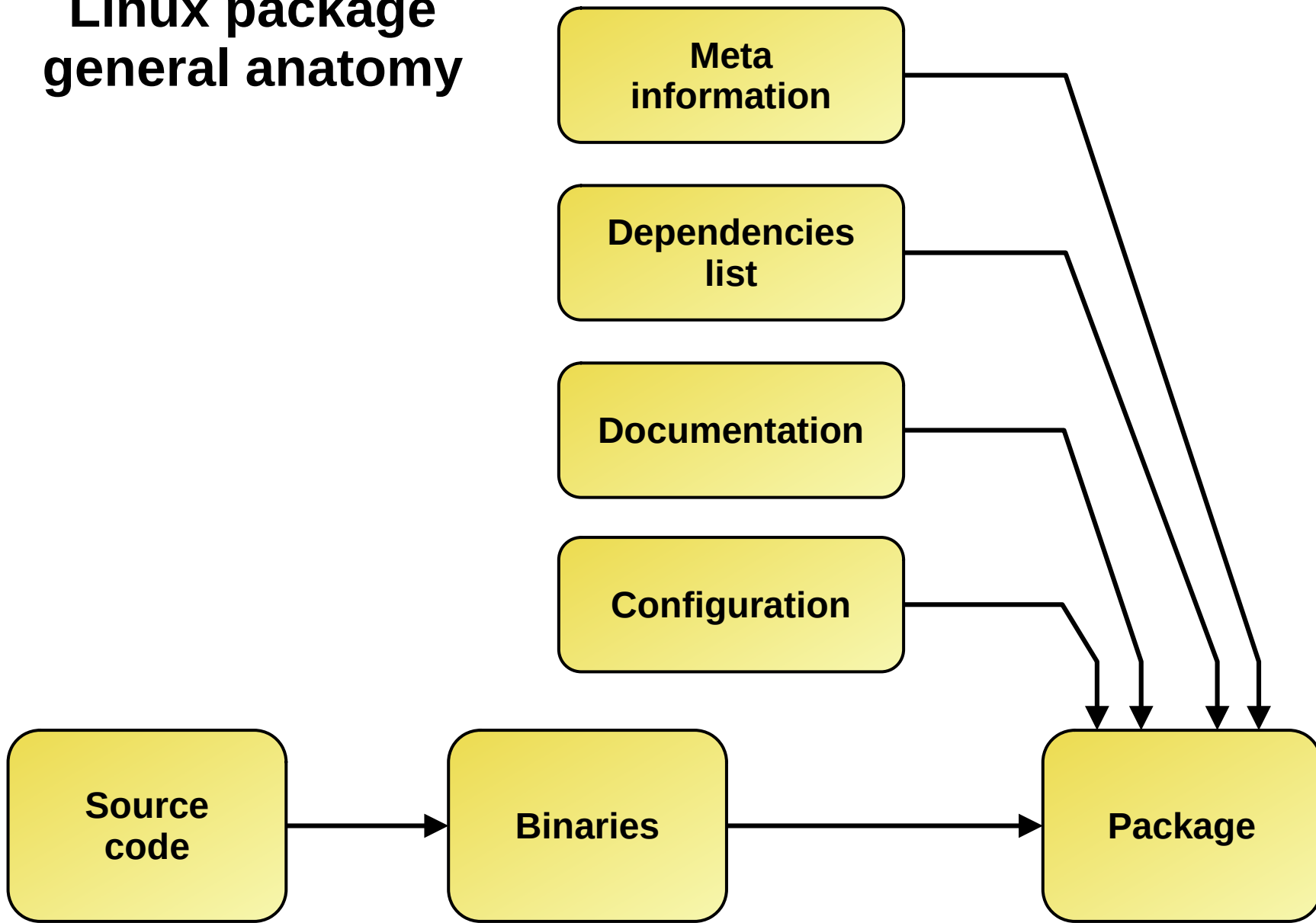
# Installing new applications

- By default, most applications added to a Linux system are already made available by the distribution that you are using. The process boils down to identify and install the right package.
- On occasion, you may need to install an application directly from source and compile it.
- Last, few applications (mostly commercial ones) will come as an independent binary and will have their own installation process.

# Package management systems

- There is two main packaging systems used across various Linux distributions:
  - apt, originally used with Debian
  - rpm, originally used with Red Hat
- One cannot be used instead of or with the other.
- Few distributions may have a packaging system of their own.

# Linux package general anatomy



# Packages repositories

- Packages managed by a distribution are hosted in repositories (repos, for short).
- Applications are already compiled, for specific platforms and with specific options.
- The source is also available, if you need to repackage an application yourself.
- Various repositories can be used, usually by release and architecture.
- Repositories can be mirrored on the Internet.

# Updates frequency

- Depending on the Linux distribution and the branch that you are using (stable, testing), updates may be available at various rates.
- Subscribe to the announce distribution list (or RSS feed or similar system) to know when new patches are released.

# apt packages

- Stands for “Advanced package tool”.
- *apt* is used by Debian, Ubuntu and other Debian-based distributions.
- *apt* (and other related commands) is a front-end for the *dpkg* tool (Debian package manager).
- File names are using the `.deb` extension.



# apt basic operations

- Searching for a package  
*apt search <pattern>*
- Installing a package  
*apt install <package>*
- Removing a package  
*apt remove <package>*
- Updating the package list  
*apt update*
- Upgrading all installed packages  
*apt upgrade*

# rpm packages

- Originally standing for “Red Hat Package Manager”, now used as “RPM Package Manager”.
- *yum* is the most common front-end to manage rpm packages. *dnf* is a newest tool designed to replace *yum*.
- RPM is mostly used by Red Hat Enterprise Linux, Fedora, CentOS, Rocky Linux, Oracle Linux and SUSE Linux.
- File names are using the .rpm extension.

# yum basic operations

- Searching for a package  
*yum search <pattern>*
- Installing a package  
*yum install <package>*
- Removing a package  
*yum remove <package>*
- Updating the package list  
*yum check-update*
- Upgrading all installed packages  
*yum upgrade*

# Installing software from sources

- Some applications may not be packaged for your distribution (too old or too recent).
- Some applications may require options defined at compilation time; a package may not include the ones that you need.
- You may want to create a patch or join the development team for a project; compiling code will be part of that process.

# Building environment

- Compiling some core, low-level applications (glibc, openssl, ...) should be done with caution as you do not want to replace the ones provided by your distribution.
- Compiling some applications may use some resources (CPU, storage space, ...).
- It is recommended to set a specific, independent system for that purpose.

# Preparatory steps

- Check for documentation provided on the software website, with the source files.
- Check for dependencies, including version numbers.
- Subscribe to the announce distribution list (or RSS feed or similar system) to know when new versions are released.
- If you get stuck, ask on forums, distribution lists or even contact the developers. Always provide details.

# Obtaining the source files

- Download the source files and check their validity (checksum, digital signature).
- You can also clone a git repository, but that may not give you a stable version.
- Decompress the files in an appropriate directory (typically */usr/local/src*).
- Check for documentation files, configuration examples and release notes.

# GNU Autotools

- For many applications developed in C, the most common build system are the *GNU Autotools*; it's a suite of applications to configure, check and compile source code.
- In it's most basic expression, three commands are used:
  - *./configure*
  - *make*
  - *make install*



# ./configure

- This step is the one that define the build options and other options on how to use the software:
  - specific features
  - configuration files, log files locations, ...
  - linked libraries, databases connections, ...
- *./configure* usually provides some internal help

# make

- This is the main operation, it could be really time consuming.
- Just wait.

# make install

- After a successful compilation, the software can be installed into the system (binaries, configuration files, manual pages, ...).
- In some cases, updating the list of shared libraries could be needed before using the new application (*ldconfig*).
- Some applications also provide a *make uninstall* target.

# Maintaining your application

- Repeat the same steps for each new version, with updates for new features as needed.
- The *config.log* file keeps track of the options used with the previous installation.