Capítulo 3: Administración de paquetes

En Linux a menudo se utilizan las palabras *programa* y *paquete* indistintamente, aunque estrictamente hablando no sean iguales. Sin embargo, para nuestros propósitos actuales podemos considerarlos equivalentes. De esa manera, cuando hagamos referencia a la *administración de paquetes* estaremos hablando de *programas*. Con eso en mente, entremos en el apasionante mundo del manejo de software en Linux.

Conceptos esenciales

Comencemos definiendo a la *administración de paquetes* como el conjunto de tareas de *instalación*, *actualización*, y *desinstalación* de recursos de software en nuestro sistema. Hoy en día, la distribución de software en Linux se realiza de tres maneras principales:

- **Repositorios**: son servidores centrales, mantenidos generalmente por una distribución en particular, por la comunidad, o por otros terceros, en los que se dispone de una colección de software para la instalación a través de Internet o de una red local. En términos simples, un repositorio en Linux (también llamado *origen de software*) es una colección de programas compatible con una distribución dada. Un repositorio puede residir en un servidor remoto (que es lo más común) o en un (o varios) dispositivos de almacenamiento locales.

- **Archivos precompilados**: son archivos ejecutables compatibles para una distribución en particular y que pueden instalarse tanto a través de la red como de manera local con una herramienta de la línea de comandos (en unos minutos veremos esto en más detalle).

- **Código fuente**: es la forma inicial mediante la cual se distribuía software en Linux en sus comienzos. Aunque ha sido superada por las dos anteriores, todavía continúa usándose para ofrecer a los usuarios finales la posibilidad de compilar un programa dado con opciones particulares. También se utiliza este método de distribución de software para poner al alcance del público una versión más actualizada que tenga mayores prestaciones que las presentes en los repositorios.

Cada distribución tiene distintas utilidades para la administración de paquetes. Algunos de ellos (como `yum` o `apt-get`, para CentOS y Debian, respectivamente, junto a los miembros de cada familia) tienen la habilidad de instalar junto con un programa dado sus dependencias, es decir, aquellos paquetes adicionales que son necesarios para el funcionamiento del primero. Por esa razón se los conoce más comúnmente con el nombre de *gestores de paquetes*. Otros, como `dpkg` (Debian y derivados) y `rpm` (CentOS y familiares), solamente instalan el programa presente en el archivo **.deb** o **.rpm** pero sin instalar dependencias.

Más sobre repositorios

Por razones prácticas, no todos los repositorios disponibles son incluidos en la instalación inicial de una distribución, pero pueden ser agregados luego cuando se los necesite.

Todas las distribuciones mantienen sus propios repositorios oficiales, y además dispone de otros que son mantenidos por la comunidad o usuarios particulares (en este último caso generalmente se trata de repositorios que contienen algunos pocos paquetes). Además, a menudo los repositorios se clasifican de acuerdo con el tipo de software que contienen.

Aunque cada distribución difiere un poco de las otras en la forma en que maneja sus repositorios, el principio de funcionamiento es el mismo:

1. Se configuran los repositorios en los que desea buscar software para nuestro sistema.

2. Antes de utilizar los nuevos repositorios configurados se requiere actualizar el índice de paquetes local de tal manera que refleje los contenidos de los repositorios.

3. Se utiliza el gestor de paquetes de cada distribución para buscar información sobre un paquete dado o directamente para instalarlo.

---

*Para evitar congestionar un único repositorio, alrededor del mundo existen varias réplicas idénticas o **mirrors** (espejos) de los repositorios centrales de cada distribución.*

En Debian, la lista de repositorios se mantiene en el archivo **/etc/apt/sources.list** y en archivos ubicados dentro de **/etc/apt/sources.list.d**. En uno u otro caso, el formato del archivo es el mismo según mostramos a continuación, tomando como ejemplo el mirror de Debian provisto por la UBA (Universidad de Buenos Aires, Argentina):

```
deb http://ftp.ccc.uba.ar/pub/linux/debian/debian/ stable main contrib no
n-free
deb-src http://ftp.ccc.uba.ar/pub/linux/debian/debian/ stable main contri
b non-free
```

Veamos ahora en detalle cómo están compuestas las líneas de arriba:

- **deb** y **deb-src** indican si el repositorio contiene paquetes binarios compilados previamente o los paquetes fuente (que contienen todo lo necesario para que uno compile un programa en particular), respectivamente.

- A continuación, encontramos la ruta al repositorio.
- Luego aparece la palabra **stable** (o el *codename* de la versión estable, como por ejemplo **stretch** a la fecha de hoy).
- Finalmente, tenemos la lista de componentes del repositorio (no hacen falta que figuren los tres necesariamente):
  - **main**: contiene software compatible con las Políticas Debian de Software Libre.
  - **contrib**: contiene software compatible con las Políticas Debian de Software Libre, aunque sus dependencias generalmente están incluídas en *non-free* (ver a continuación).
  - **non-free**: software no compatible con las Políticas Debian de Software Libre.

En el sitio oficial del proyecto Debian se puede consultar la lista de *mirrors* por países. Podemos elegir el país en el que vivamos o utilizar la red de distribución de contenidos (CDN) principal.

En CentOS o similares, los repositorios se configuran mediante archivos individuales dentro de **/etc/yum.repos.d**. Tomemos como ejemplo a **CentOS-Base.repo**, cuyo contenido podemos apreciar a continuación (no se muestra el archivo completo por cuestiones de espacio):

```
[gacanepa@server yum.repos.d]$ cat CentOS-Base.repo
# CentOS-Base.repo
#
# The mirror system uses the connecting IP address of the client and the
# update status of each mirror to pick mirrors that are updated to and
# geographically close to the client.  You should use this for CentOS updates
# unless you are manually picking other mirrors.
#
# If the mirrorlist= does not work for you, as a fall back you can try the
# remarked out baseurl= line instead.
#
#

[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

En la imagen anterior podemos distinguir que la configuración de un repositorio consta de:

- El nombre (**base**).

- Una lista de mirrors (**mirrorlist**) desde la cual se seleccionará el más próximo geográficamente, o un repositorio fijo (**baseurl**).
- La indicación si se necesita utiliza la clave pública del repositorio.
- La ruta al archivo de la clave pública.

Cuando agreguemos un nuevo repositorio a CentOS se nos pedirá que confirmemos que queremos instalar en nuestro equipo la clave GPG que dicho repositorio provee por cuestiones de seguridad.

En algunas ocasiones será necesario agregar manualmente un repositorio de terceros. Generalmente tendremos que recurrir a esta opción cuando deseemos instalar una versión más reciente de un programa que la que se encuentre disponible en los repositorios oficiales de la distribución.

Para ilustrar agregaremos el siguiente repositorio en **/etc/yum.repos.d/mariadb.repo** para instalar la última versión estable de MariaDB. En primer lugar, agreguemos las siguientes líneas dentro del archivo mencionado:

```
# MariaDB 10.3 CentOS repository list - created 2018-06-08 00:28 UTC
# http://downloads.mariadb.org/mariadb/repositories/
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/10.3/centos7-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

---

*El nombre de los archivos de configuración de los repositorios que añadamos manualmente deben finalizar en **.repo**. Lo mismo aplica tanto para CentOS y similares (en **/etc/yum.repos.d**) como para las distribuciones de la familia SUSE dentro de **/etc/zypp/repos.d**.*

Para asegurarnos de que nuestro sistema puede *ver* la lista completa y actualizada de paquetes disponibles en los repositorios debemos sincronizar el índice local con los mismos. En Debian y derivados lo hacemos mediante `apt-get update`, mientras que en CentOS o similares utilizamos `yum makecache fast`. En openSUSE el comando equivalente es `zypper refresh`.

En el próximo apartado explicaremos el próximo paso necesario a seguir antes de efectuar la instalación propiamente dicha de un paquete proveniente de un repositorio de terceros.

Administración de paquetes en Linux

Debian y derivados

Para instalar archivos **.deb** , utilizaremos el comando `dpkg` seguido de la opción `-i` o `--install` y de la ruta al archivo a instalar. Por ejemplo, para instalar **mipaquete.deb** (presente en nuestro directorio actual de trabajo) utilizaremos el siguiente comando:

```
dpkg -i mipaquete.deb
```

Si posteriormente quisiéramos instalar una nueva versión del mismo programa mediante un archivo **.deb** más reciente, podemos utilizar el mismo comando anterior. En cambio, si necesitamos desinstalarlo emplearemos la opción `-r` (también puede ser `--remove`) o `-P` (de `--purge`) simplemente incluyendo el nombre del paquete:

```
dpkg -r mipaquete
```

La diferencia entre `--remove` y `--purge` reside en que esta última remueve el paquete junto con cualquier archivo de configuración que se pueda haber creado como parte de la instalación, mientras que la primera no.

Finalmente, con `-l` podremos ver una lista de todos los paquetes instalados en el sistema. Como la salida será extensa en este caso se recomienda que la visualicemos con less o la filtremos con alguna herramienta como `tail`, `head`, o `grep`.

```
dpkg -l | head
```

Otras dos opciones muy útiles son `--info` y `--contents`. La primera nos permitirá ver la información sobre el paquete en cuestión, mientras que la segunda nos mostrará el contenido de este (en ambos casos debemos especificar el nombre completo del archivo **.deb** a examinar):

```
dpkg --info mipaquete.deb
dpkg --contents mipaquete.deb
```

De forma similar a `--contents`, podemos emplear `--listfiles` (en este caso utilizamos solamente el nombre del paquete, sin la extensión **.deb**) para ver los archivos que se agregaron al sistema al instalar un paquete (sea que lo hayamos hecho mediante `dpkg` o `apt-get`, como veremos a continuación).

Para asegurarnos de que un programa se instale con todas las dependencias necesarias podemos utilizar el gestor de paquetes `apt-get` en vez de `dpkg`. La herramienta asociada `apt-cache` también nos resultará útil para buscar programas en los repositorios y para ver información detallada sobre alguno que haya captado nuestra atención. Ambas utilidades (junto con otras) forman parte de **APT** (**A**dvanced **P**ackage **T**ool) que es el sistema de gestión de paquetes principal de Debian.

Luego de haber agregado repositorios en **/etc/apt/sources.list** o en algún archivo dentro de **/etc/apt/sources.list.d**, sincronicemos la lista de paquetes disponibles con los repositorios de Debian:

```
apt-get update
```

Luego de eso, podremos emplear

```
apt-cache search mipaquete
```

para utilizar a **mipaquete** como patrón de búsqueda, o si ya conocemos el nombre exacto podemos emplear

```
apt-cache show mipaquete
apt-cache showpkg mipaquete
```

para ver información detallada sobre el mismo, o

```
apt-get install mipaquete
```

para instalarlo. Si agregamos la opción `-f` al comando anterior también podremos completar cualquier instalación que hayamos llevado a cabo mediante `dpkg` y que no se haya completado exitosamente por dependencias faltantes.

Para aceptar automáticamente los pedidos de confirmación podemos agregar la opción -y al comando anterior. Esto puede resultar particularmente útil en el caso de que vayamos a automatizar una instalación (o actualización) de alguna manera. Lo mismo aplica al uso de `yum install` en CentOS y similares.

Por otro lado, si **mipaquete** ya estuviera instalado y quisiéramos removerlo, cualquiera de los comandos siguientes nos permitirá hacerlo:

```
apt-get remove mipaquete
apt-get purge mipaquete
```

El último de ellos no solamente eliminará **mipaquete** del sistema sino que también hará lo mismo con cualquier archivo de configuración relacionado con el programa en cuestión.

Para actualizar un paquete a la última versión disponible usaremos

```
apt-get upgrade mipaquete
```

Si en el comando anterior omitiéramos el nombre del paquete, el resultado sería que se actualizarían todos ellos a la última versión disponible en los repositorios configurados del sistema.

Reconfigurar paquetes

Es importante notar que durante el curso de una instalación por lo general se nos ofrece la posibilidad de configurar el programa en cuestión. Si no lo hacemos en ese momento, también podemos hacerlo posteriormente utilizando la herramienta `dpkg-reconfigure`.

En primer lugar, podemos revisar la configuración actual con el comando `debconf` seguido del nombre de un paquete dado. Usemos como ejemplo el paquete **keyboard-configuration**, según se aprecia en la siguiente imagen:

```
debconf-show keyboard-configuration
```

```
root@debianhome:~# debconf-show keyboard-configuration
* keyboard-configuration/xkb-keymap: latam
  keyboard-configuration/ctrl_alt_bksp: false
* keyboard-configuration/store_defaults_in_debconf_db: true
  keyboard-configuration/unsupported_config_options: true
* keyboard-configuration/switch: No temporary switch
  keyboard-configuration/unsupported_config_layout: true
* keyboard-configuration/toggle: No toggling
* keyboard-configuration/altgr: The default for the keyboar
* keyboard-configuration/compose: No compose key
  keyboard-configuration/unsupported_layout: true
* keyboard-configuration/variant: Spanish (Latin American)
* keyboard-configuration/other:
```

Si vemos algo que queremos o vemos la necesidad de cambiar, ahora es el momento de recurrir a `dpkg-reconfigure`. Siguiendo con el mismo ejemplo de arriba, ahora se deberá utilizar el comando

```
dpkg-reconfigure keyboard-configuration
```

para hacer las modificaciones correspondientes utilizando solamente el teclado. La ventaja de utilizar este método para configurar software ya instalado es que nos evitamos tener que editar archivos de texto a mano. Incluso en el caso de que no todas las opciones estén contempladas, nos ahorra trabajo.

Otra situación en la que `dpkg-reconfigure` puede resultar útil es para restaurar la configuración de un paquete a su estado original. Si cometimos algún error y no hicimos copias de respaldo antes de comenzar, esta herramienta puede ayudarnos a salir del aprieto. Simplemente aceptando las opciones que se nos presenten por defecto deberíamos resolver el inconveniente.

CentOS y similares

RPM (**R**ed Hat **P**ackage **M**anager) es el gestor de paquetes utilizado por defecto en Red Hat Enterprise Linux como en SuSE (y distribuciones derivadas como CentOS, Fedora, u OpenSUSE). Fue desarrollado por Red Hat y luego adoptado por otras distribuciones. Podemos decir que consta de 2 componentes fundamentales: 1) el comando `rpm`, y 2) una base de datos local. Esta última contiene la lista de paquetes instalados e información sobre estos y se encuentra dentro del directorio **/var/lib/rpm**.

Para empezar, tomemos como ejemplo a **bc-1.06.95-13.el7.x86_64.rpm**. Podemos descargar el mismo desde el listado de paquetes disponibles para CentOS 7. En el nombre del archivo podemos encontrar a primera vista los siguientes datos:

- El *nombre* del paquete: **bc**
- La *versión*: **1.06-95-13.el7**
- La *arquitectura*: **x86_64**

Por lo general, la instalación se realiza con las opciones combinadas `-Uvh` o `-Fvh` en vez del simple `-i` (o su equivalente `--install`, también válidos). De las dos alternativas anteriores, la primera instala o actualiza (según sea el caso), mientras que la segunda solamente instalará el paquete y no realizará ninguna acción si ya existe una versión previa del mismo. Para nuestra conveniencia, `-v` muestra el nombre del paquete durante la instalación y `-h` imprime una serie de #'s para indicar el progreso de la instalación. Ambas son útiles en particular si estamos instalando varios paquetes a la vez.

Para desinstalar un paquete que hayamos previamente instalado con `rpm`, utilizaremos la misma herramienta, pero esta vez con la opción `-e` seguida del nombre del paquete. No es necesario incluir la versión de este ni la arquitectura.

Otras opciones útiles del comando `rpm` que no pueden faltar en nuestro arsenal son (siempre seguidas de `-q`, de *query*):

- `-a`: muestra la lista de todos los paquetes instalados.
- `-l`: lista de los archivos instalados por un paquete dado.
- `-ip`: ver información detallada sobre un archivo **.rpm** que no ha sido instalado.
- `-f`: devuelve el nombre del paquete que contiene el archivo que se especifica a continuación.

- `-Rp`: muestra la lista de dependencias del paquete.

En la imagen siguiente vemos los siguientes ejemplos ilustrados:

- Verificamos si el paquete `htop` está instalado usando `rpm -qa | grep htop`
- Mostramos la lista de archivos agregados al sistema como resultado de haber instalado `htop` con `rpm -ql htop`
- Vemos las dependencias del paquete `tuxpaint` mediante `rpm -qRp tuxpaint-0.9.22-1.x86_64.rpm`

```
[gacanepa@server ~]$ rpm -qa | grep htop
htop-2.1.0-1.el7.x86_64
[gacanepa@server ~]$ rpm -ql htop
/usr/bin/htop
/usr/share/applications/htop.desktop
/usr/share/doc/htop-2.1.0
/usr/share/doc/htop-2.1.0/AUTHORS
/usr/share/doc/htop-2.1.0/ChangeLog
/usr/share/doc/htop-2.1.0/README
/usr/share/licenses/htop-2.1.0
/usr/share/licenses/htop-2.1.0/COPYING
/usr/share/man/man1/htop.1.gz
/usr/share/pixmaps/htop.png
[gacanepa@server ~]$ rpm -qRp tuxpaint-0.9.22-1.x86_64.rpm
/bin/bash
/bin/sh
SDL >= 1.2.4
SDL_Pango
SDL_image
SDL_mixer
SDL_ttf
```

Para ver el contenido completo de un archivo **.rpm** utilizar los comandos `rpm2cpio` y `cpio` de la siguiente manera:

```
rpm2cpio tuxpaint-0.9.22-1.x86_64.rpm | cpio -idv
```

A pesar de lo útil que es `rpm`, quizás la manera más práctica de instalar un programa determinado (si es que se encuentra en los repositorios de nuestra distribución, o si la versión disponible cumple con nuestras necesidades), es utilizar el gestor de paquetes `yum`. El mismo contiene todas las utilidades necesarias para satisfacer las

dependencias requeridas y configurar el programa ya instalado (en síntesis, ¡nos ahorra mucho trabajo!). Utilizando esta herramienta podemos:

- Buscar un paquete por palabra clave (**mipaquete**, por ejemplo): `yum search mipaquete`
- Ver información sobre **mipaquete**: `yum info mipaquete`
- Instalarlo: `yum install mipaquete`
- Actualizar a la última versión disponible: `yum update mipaquete`

Tal como sucede en el caso de Debian con `apt-get upgrade`, si en este caso omitimos el nombre del paquete se actualizarán todos los presentes en el sistema.

- Desintalarlo: `yum remove mipaquete`
- Averiguar en qué paquete está incluído un comando determinado (por ejemplo, **micomando**): `yum whatprovides "*/micomando"`

La configuración de `yum` se encuentra en **/etc/yum.conf**. Si bien podemos agregar repositorios en este archivo, por prolijidad y orden se prefiere hacerlo mediante archivos individuales (con la extensión **.repo**) dentro del directorio **/etc/yum.repos.d** como explicamos antes.

Cuando agreguemos un nuevo repositorio a CentOS se nos pedirá que confirmemos que queremos instalar en nuestro equipo la clave GPG que dicho repositorio provee. Sin este paso no será posible descargar ningún paquete de ese origen.

Si en algún momento nos interesa repasar la actividad de `yum`, podemos consultar el archivo de registros (o *logs*) en **/var/log/yum.log**.

Ya sea cuando agregamos un repositorio de terceros (como en el caso de MariaDB antes) o incluso en **/etc/yum.repos.d/CentOS-Base.repo**, podemos valernos de la directiva **enabled** para indicar si se debe habilitar inmediatamente o no. Por ejemplo, en la imagen podemos ver la configuración del repositorio **centosplus** y notar **enabled=1**. Esto indica que **centosplus** está habilitado en nuestro sistema:

```
#additional packages that extend functionality of existing packages
[centosplus]
name=CentOS-$releasever - Plus
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$b
#baseurl=http://mirror.centos.org/centos/$releasever/centosplus/$bas
gpgcheck=1
enabled=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

Debido a que **centosplus** contiene paquetes que son mejoras para los presentes en los repositorios **base** y **updates**, es posible que queramos deshabilitarlo en ocasiones para que no interfieran con estos últimos. Podemos hacerlo al editar el valor de la directiva a **enabled=0** antes de hacer `yum update` o bien indicar `yum update --disablerepo=centosplus`. Lo mismo aplica si estuviera deshabilitado y quisiéramos habilitarlo con `yum update --enablerepo=centosplus`.

---

*El habilitar o deshabilitar un repositorio mediante la línea de comandos tiene efecto solamente durante la transacción actual. Para hacerlo de manera permanente debemos cambiar el valor correspondiente en la directiva **enabled** como mostramos antes.*

En Fedora se ha comenzado a utilizar un gestor de paquetes llamado `dnf` desde hace un par de años (fue presentado con la versión 18 y pasó a ser la herramienta por defecto a partir de la 22). Guarda semejanza con las opciones disponibles de `yum` que hemos visto, aunque es más rápido y consume menos memoria que su antecesor. El proyecto Fedora provee documentación detallada sobre `dnf` en su sitio oficial.

Familia SUSE

Estas distribuciones, entre las que se destaca openSUSE, también utilizan la herramienta `rpm` para administrar paquetes individuales. Además, también disponen de un gestor de paquetes muy completo llamado `zypper`. A continuación, veremos las operaciones más comunes que podemos realizar con `zypper`:

- Refrescar (actualizar) repositorios: `zypper refresh`
- Búsqueda por palabra clave (**apache** en este caso): `zypper search apache`
- Información sobre un paquete en particular: `zypper info apache2`
- Instalación de un paquete: `zypper install apache2`
- Actualización: `zypper update apache2`
- Desintalación: `zypper remove apache2`
- Ayuda general: `zypper help`
- Ayuda sobre una operación específica (**install** en este caso): `zypper help install`

Cada una de las operaciones anteriores tienen un equivalente breve: **ref** (**refresh**), **se** (**search**), **if** (**info**), **in** (**install**), **up** (**update**), y **rm** (**remove**).

Debido a la gran cantidad de opciones disponibles para el uso de `zypper` no es posible listarlas a todas aquí. Para más información, se recomienda utilizar el *man page* y examinar cuidadosamente la ayuda para cada operación como mencionamos en los últimos dos puntos de arriba.