Repositorios

A diferencia de Windows, en los sistemas Linux y Unix no es común contar con programas que incluyan un instalador interactivo (como el típico *install.exe*). En ocasiones, algunos desarrolladores ofrecen scripts de instalación que, en la mayoría de los casos, simplemente descomprimen los archivos y los colocan en los directorios correspondientes.

En Linux es mucho más habitual distribuir aplicaciones, herramientas y actualizaciones en forma de **paquetes** (*packages*). Un paquete es un archivo, a veces de gran tamaño, que contiene el software a instalar junto con una serie de instrucciones o reglas que definen su gestión. Estas reglas pueden incluir:

- **Gestión de dependencias:** el software solo podrá instalarse si los programas que necesita ya están presentes en el sistema.
- **Acciones de preinstalación:** tareas previas necesarias antes de la instalación, como modificar permisos o crear directorios.
- **Acciones de postinstalación:** configuraciones que se ejecutan después de la instalación, como ajustar parámetros de archivos o realizar compilaciones adicionales.

A la hora de instalar software en un sistema linux, tenemos dos maneras de hacerlo: compilando el código fuente o instalando un paquete precompilado. Cada distribución de linux utiliza diferentes formatos de paquetes, siendo dos los más extendidos.

- **.deb**: familia Debian (Ubuntu, Kali, Linux Mint...). Para ello usaremos la herramienta *dpkg*.
- **.rpm**: familia Red Hat (CentOS, Fedora, SUSE...). Tenemos un conjunto de comandos principales para la gestión de paquetes en linux. Para ello usaremos la herramienta *yum*.

.deb

dpkg

Gestiona paquetes **.deb** sin gestión de dependencias. La base de datos de *dpkg* donde figura todo lo instalado está en `/var/lib/dpkg`.

- `dpkg -i {package.deb}` : usado para instalar el paquete.
- `dpkg -R {directory}` : este comando instala todos los paquetes de un estructura de directorio.
- `dpkg -l` : muestra la lista de paquetes instalados.
- `dpkg -c {package.deb}` : lista los archivos en un paquete.
- `dpkg -L {package}` : muestra los archivos proporcionados por un paquete instalado.
- `dpkg -S {pattg/to/file}` : determina los paquetes que pertenecen a un archivo.
- `dpkg -r {package}` : elimina un paquete instalado, pero deja sus archivos de configuración.
- `dpkg -P {package}` :elimina un paquete instalado, incluyendo sus archivos de configuración.
- `dpkg --get-selections` : es utilizado para buscar paquetes instalados, desinstalados y purgados en el sistema operativo.
- `dpkg --configure --pending` : función para reconfigurar cualquier paquete que no haya pasado por un proceso de configuración.
- `dpkg --info pp.deb` : se encarga de mostrar información sobre las dependencias y sus extensiones.
- `dpkg --version`: para consultar la licencia del paquete dpkg.

- `dpkg --unpack nombre_paquete.deb`: usado para descomprimir un archivo deb.
- `dpkg -R –install /deb-files-location/`: instala varios archivos deb de forma simultánea, siempre y cuando se encuentren en la misma carpeta.
- `sudo dpkg --help`: para obtener ayuda sobre cómo usar la herramienta en cuestión.
- `dpkg --force-all --purge nombre_del_paquete`
- `dpkg-reconfigure`: reconfigura paquetes después de su instalación. Introduzca los nombres del paquete o paquetes a reconfigurar. Formulará preguntas de configuración de forma similar a cuando el paquete se instaló en primer lugar.

Nota: Existe una diferencia fundamental entre eliminar y purgar. Cuando instalamos un paquete es normal que existan configuraciones personales que se almacenan en el `/home` de cada usuario, una eliminación del paquete hace que estas configuraciones permanezcan, lo que hace que si posteriormente se instala de nuevo el paquete estas permanecen en el sistema. Por otro lado, purgar el paquete implica eliminar todos esos archivos de creación personal.

```
root@debian:~/Descargas# ls
debian-refcard_12.0_all.deb

root@debian:~/Descargas# dpkg -i debian-refcard_12.0_all.deb
Seleccionando el paquete debian-refcard previamente no seleccionado.
(Leyendo la base de datos ... 260028 ficheros o directorios instalados
actualmente.)
Preparando para desempaquetar debian-refcard_12.0_all.deb ...
Desempaquetando debian-refcard (12.0) ...
Configurando debian-refcard (12.0) ...

root@debian:~/Descargas# dpkg -l debian-refcard
Deseado=desconocido(U)/Instalar/eliminaR/Purgar/retener(H)
| Estado=No/Inst/ficheros-Conf/desempaqUetado/medio-conF/medio-inst(H)/espera-
disparo(W)/pendienTe-disparo
|/ Err?=(ninguno)/requiere-Reinst (Estado,Err: mayúsc.=malo)
||/ Nombre         Versión      Arquitectura Descripción
+++-==============-============-============
=============================================
ii  debian-refcard 12.0         all          printable reference card for the
Debian system

root@debian:~/Descargas# dpkg -L debian-refcard
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/debian-refcard
/usr/share/doc/debian-refcard/changelog.gz
/usr/share/doc/debian-refcard/copyright
/usr/share/doc/debian-refcard/index.html
/usr/share/doc/debian-refcard/openlogo-nd-25.png
/usr/share/doc/debian-refcard/refcard-bg-a4.pdf.gz
/usr/share/doc/debian-refcard/refcard-ca-a4.pdf.gz

root@debian:~/Descargas# dpkg -P debian-refcard
```

```
(Leyendo la base de datos ... 260072 ficheros o directorios instalados
actualmente.)
Desinstalando debian-refcard (12.0) ...

root@debian:~/Descargas# dpkg -l debian-refcard
dpkg-query: no se ha encontrado ningún paquete que corresponda con debian-refcard.
```

Nota: Cuando ejecutamos *dpkg -l* puede darnos que el paquete esté instalado como *ii* o eliminado como *rc*.
Nota: Si queremos volver a realizar la configuración de un paquete cuando se instaló (preguntas que contesta el usuario durante la instalación), tenemos el comando dpkg-reconfigure.

apt

Facilita la instalación de paquetes **.deb**, descargándolos de los repositorios y gestionando las dependencias de estos, usamos la herramienta *Advanced Package Tool: apt-get*. Es importante tener en cuenta que al ser un repositorio online, si cada vez que tuvieramos que realizar una descarga tuvieramos que comprobar en que repositorio se encuentra el paquete el proceso podría demorarse mucho, por este motivo se guarda en el sistema en la ruta `/etc/apt/sources.list` con los respositorios en los cuales se encuentran los paquetes y en una caché local los paquetes que se encuentran en cada repostiorio así como la versión de los mismos. Es buena práctica actualizar la información de esta caché antes de realizar la descarga de un paquete.

- `get`: Obtiene los paquetes de los repositorios pero no los instala. Actualmente ya no es necesario.
- `search`: Accede a la base de datos local de los repositorios para poder buscar el paquete deseado.
- `install`: Instala uno o varios paquetes.
- `remove`: Elimina uno o varios paquetes.
- `autoremove`: Elimina paquetes huérfanos, es decir, no requeridos por ningún otro paquete.
- `purge`: Elimina uno o varios paquetes y también sus archivos de configuración. Alternativamente podemos utilizar `sudo apt remove --purge nombre_del_paquete`.
- `download`: Descarga el paquete .deb sin instalarlo.
- `update`: Actualiza la lista de paquetes disponibles en los repositorios. No instala nada en el sistema.
- `upgrade`: Actualiza todos los paquetes instalados a las versiones más recientes disponibles de los repositorios y si no están actualizados nos dirá que es necesario instalar.
- `dist-upgrade` o `full-upgrade`: En caso de que la distribución cambie, es decir, exista una nueva distribución, incluyendo la resolución de dependencias y la eliminación o instalación de nuevos paquetes si es necesario para completar la actualización de forma coherente de los paquetes ya existentes.
- `list --installed`: En linux, ejecutar `apt list --installed` permite mostrar todos los paquetes deb instalados en el sistema.
- `clean`: Elimina todos los archivos .deb en la caché (`/var/apt/cache/archives`), liberando así espacio y dejando la caché vacía.
- `autoclean`: Elimina solo los archivos .deb almacenados en la caché que ya no pueden ser descargados desde los repositorios, es decir, versiones antiguas u obsoletas que no se usan ni se pueden actualizar. Esto ayuda a mantener la caché limpia sin eliminar archivos que podrían ser útiles para reinstalar paquetes actuales.

Podemos ver los repositorios donde `apt` hace sus busquedas a continuación:

```
usuario@debian:~$ cat /etc/apt/sources.list | sed -e '/^#/d' -e '/^$/d'
deb http://es.archive.ubuntu.com/ubuntu/ jammy main restricted
deb http://es.archive.ubuntu.com/ubuntu/ jammy-updates main restricted
deb http://es.archive.ubuntu.com/ubuntu/ jammy universe
deb http://es.archive.ubuntu.com/ubuntu/ jammy-updates universe
deb http://es.archive.ubuntu.com/ubuntu/ jammy multiverse
deb http://es.archive.ubuntu.com/ubuntu/ jammy-updates multiverse
deb http://es.archive.ubuntu.com/ubuntu/ jammy-backports main restricted universe
multiverse
deb http://security.ubuntu.com/ubuntu jammy-security main restricted
deb http://security.ubuntu.com/ubuntu jammy-security universe
deb http://security.ubuntu.com/ubuntu jammy-security multiverse
```

Buscar e instalar un paquete individual

El comando `apt-cache` permite buscar paquetes en la base de datos local de APT, ya sea por su nombre o por su descripción. Por ejemplo, si queremos localizar el paquete **Neovim**, un editor de texto moderno inspirado en Vim, podemos realizar la búsqueda de la siguiente manera:

```
root@debian:~# apt update
...

root@debian:~# apt-cache search neovim
interception-caps2esc - interception plugin for dual function Esc/Ctrl key at
CapsLock
dh-vim-addon - debhelper addon to help package Vim/Neovim addons
lua-nvim - Lua client for Neovim
lua-nvim-dev - Lua client for Neovim (development files)
neovim - heavily refactored vim fork
neovim-runtime - heavily refactored vim fork (runtime files)
neovim-qt - neovim client library and GUI
vim-redact-pass - stop pass(1) passwords ending up in Vim cache files
python3-neovim - transitional dummy package
python3-pynvim - Python3 library for scripting Neovim processes through its
msgpack-rpc API
ruby-neovim - Ruby client for Neovim
vim-ale - Asynchronous Lint Engine for Vim 8 and NeoVim
```

La salida es más completa empleando el comando de forma abreviada, que nos indica además si ese paquete está o no ya instalado en el equipo.

```
root@debian:~# apt search neovim
Ordenando... Hecho
Buscar en todo el texto... Hecho
dh-vim-addon/oldstable 0.4 all
  debhelper addon to help package Vim/Neovim addons

interception-caps2esc/oldstable 0.3.2-1+b1 amd64
```

```
  interception plugin for dual function Esc/Ctrl key at CapsLock

lua-nvim/oldstable 0.2.4-1-1 amd64
  Lua client for Neovim

lua-nvim-dev/oldstable 0.2.4-1-1 amd64
  Lua client for Neovim (development files)

neovim/oldstable 0.7.2-7 amd64
  heavily refactored vim fork

neovim-qt/oldstable 0.2.16-1 amd64
  neovim client library and GUI

neovim-runtime/oldstable 0.7.2-7 all
  heavily refactored vim fork (runtime files)

python3-neovim/oldstable 0.4.2-2 all
  transitional dummy package

python3-pynvim/oldstable 0.4.2-2 all
  Python3 library for scripting Neovim processes through its msgpack-rpc API

ruby-neovim/oldstable 0.8.1-1 all
  Ruby client for Neovim

vim-ale/oldstable 3.3.0-1 all
  Asynchronous Lint Engine for Vim 8 and NeoVim

vim-redact-pass/oldstable 1.7.4-6 all
  stop pass(1) passwords ending up in Vim cache files
```

Interpretando lo anterior podemos decir que nos muestra todos los paquetes relacionados con *neovim*. Podemos interpretar la salida como:

```
neovim/oldstable 0.7.2-7 amd64
  heavily refactored vim fork
```

- `neovim`: nombre del paquete.
- `oldstable`: versión o rama de Debian a la que pertenece el paquete.
- `0.7.2-7`: versión específica del paquete.
- `amd64`: arquitectura para la que está compilado (64 bits).
- `heavily refactored vim fork`: breve descripción del paquete (Neovim es un fork de Vim con muchas mejoras).

Para realizar la instalación realizamos los siguientes pasos indicando exactamente cual de los paquetes anteriores vamos a instalar. El siguiente comando descarga el paquete y realiza directamente la instalación del mismo, si queremos descargar el paquete pero no proceder a su instalación podemos hacer uso de `apt download`. Instalamos de la siguiente forma:

```
root@debian:~# apt install neovim -y
...

root@debian:~# apt list --installed | grep neovim

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

neovim-runtime/oldstable,now 0.7.2-7 all [instalado, automático]
neovim/oldstable,now 0.7.2-7 amd64 [instalado]

root@debian:~# nvim /etc/network/interfaces
...
```

En linux los archivos *.deb* que apt descarga para instalar paquetes se guardan temporalmente en la carpeta `/var/cache/apt/archives/`. Estos paquetes pueden ser eliminados de forma automática por el sistema o de forma manual haciendo uso del comando `apt clean`.

Si queremos eliminar el paquete procedemos del siguiente modo:

```
root@debian:~# apt purge neovim -y
...
Utilice «apt autoremove» para eliminarlos.
...

root@debian:~# apt list --installed | grep neovim

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

neovim-runtime/oldstable,now 0.7.2-7 all [instalado, autodesinstalable]
```

Nota: Tambien podemos hacer uso del comando *apt remove --purge neovim -y* para eliminar los archivos de configuración.

Nota 2: Una dependencia es otro paquete que un programa necesita para funcionar correctamente. Cuando instalas un paquete, el gestor de paquetes también instala automáticamente todas sus dependencias, que son librerías o programas auxiliares que el paquete principal requiere para ejecutarse sin errores.

En este punto solo queda eliminar las dependencias huerfanas del sistema como se indica anteriormente una vez eliminado el paquete.

```
root@debian:~# apt autoremove
...

root@debian:~# apt list --installed | grep neovim

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

Nota: Una dependencia es un paquete o libreria necesario para el funcionamiento del paquete que queremos instalar. Del mismo modo que `apt remove` no elimina los archivos de configuración del paquete original, `apt autoremove` no elimina si existen los archivos de configuración de las dependencias, para ello existe `apt autoremove --purge`.

Actualizar la distribución de debian 12.10 12.11

```
root@debian:~# cat /etc/debian_version
12.10
root@debian:~# apt update
...
root@debian:~# apt dist-upgrade -y
...
root@debian:~# reboot
...
usuario@debian:~$ cat /etc/debian_version
12.11
```

Actualizar la distribución de Debian 12 (Bookworm) a Debian 13 (Trixie)

```
root@debian:~# cat /etc/debian_version
12.x
root@debian:~# apt update
root@debian:~# apt upgrade -y
root@debian:~# apt dist-upgrade -y
root@debian:~# sed -i 's/bookworm/trixie/g' /etc/apt/sources.list
root@debian:~# sed -i 's/bookworm/trixie/g' /etc/apt/sources.list.d/*.list
root@debian:~# apt update
root@debian:~# apt upgrade -y
root@debian:~# apt dist-upgrade -y
root@debian:~# reboot
...
usuario@debian:~$ cat /etc/debian_version
13.x
```

A considerar sobre la instalación de un paquete .deb

Cuando se usa el comando apt install paquete, no se guarda el archivo .deb descargado permanentemente en un lugar visible para el usuario. El sistema APT descarga temporalmente el archivo .deb en un directorio temporal controlado por el sistema, generalmente dentro de `/var/cache/apt/archives/`. Ahí se almacenan los archivos de paquete descargados para su instalación.

```
root@debian:/var/cache/apt/archives# ls -la | wc
    451    4052   35116
root@debian:/var/cache/apt/archives# apt clean
```

```
root@debian:/var/cache/apt/archives# ls -la | wc
      5      38     195
```

Uno puede tener la duda entre la diferencia de los comandos `clean`, `remove` y `purge` en apt. Podemos resumirlo de la siguiente forma:

- **apt clean**: Este comando elimina todos los archivos `.deb` descargados que se guardan en la caché del sistema, específicamente en `/var/cache/apt/archives/`. Esto libera espacio en disco al eliminar los paquetes de instalación ya descargados, pero no afecta a los paquetes instalados ni a sus configuraciones.
- **apt remove**: Este comando desinstala el paquete especificado, eliminando sus archivos binarios y ejecutables, pero **deja los archivos de configuración** en el sistema. Esto es útil si se planea reinstalar el paquete más adelante y se desea mantener la configuración previa.
- **apt purge**: Es una versión más completa que `remove`. Además de eliminar el paquete, también borra los archivos de configuración asociados. Esto deja el sistema más limpio, sin rastros del paquete eliminado.

Además hay otros comandos relacionados que ayudan a limpiar el sistema:

- `apt autoremove` elimina paquetes que fueron instalados como dependencias pero que ahora ya no se necesitan.
- `apt autoclean` elimina solo los archivos `.deb` de paquetes antiguos y obsoletos que ya no están en los repositorios.

Resumido en forma de tabla:

Comando	Qué elimina	Configuración eliminada	Archivos descargados eliminados
`apt clean`	Archivos `.deb` descargados en caché	No	Sí
`apt remove`	Paquete instalado	No	No
`apt purge`	Paquete instalado	Sí	No
`apt autoremove`	Paquetes dependientes no necesarios	No	No
`apt autoclean`	Archivos `.deb` antiguos y obsoletos	No	Sí (selectivo)

Cada uno tiene su función para mantener el sistema ordenado, limpio y con espacio libre en disco.

.rpm

rpm

Gestiona paquetes .rpm sin resolver automáticamente las dependencias. Todo lo instalado con *rpm* se almacena en la base de datos de `/var/lib/rpm`. Con el comando *rpm --rebuilddb*, regenera los archivos de la base de datos de RPM en `/var/lib/rpm/`. Corrige inconsistencias cuando la base de datos está corrupta. No afecta los paquetes instalados, solo repara la información sobre ellos. Esa información la podemos consultar con:

- `rpm -qa php`: Consulta información de cuales son los paquetes de un servicio instalado.
- `rpm -qi paquete`: Informacion sobre el paquete.
- `rpm -ql php`: Nos dice que ficheros instala el paquete y donde.
- `rpm -qf /etc/passwd`: Le damos un archivo y nos dice que paquete lo instalo.
- `rpm -qp paquete`: Nos indica las dependencias de los paquetes.

Parámetros de `rpm`:

- `-i`: Instala un paquete.
- `-U`: Actualiza un paquete o lo instala si no está presente.
- `-F`: Actualiza solo si el paquete ya está instalado.
- `-e`: Elimina un paquete.
- `-V`: Verifica la integridad de los archivos instalados por un paquete.
- `--rebuilddb`: Reconstruye la base de datos de rpm si hay problemas de corrupción.

Nota: El comando `rpm` no resuelve automáticamente las dependencias. Si intentas instalar un paquete y faltan dependencias, deberás instalarlas manualmente o usar una herramienta como `yum` o `dnf`.

```
[root@centos8 ~]# ls
nano-2.9.8-1.el8.x86_64.rpm

[root@centos8 ~]# rpm -i nano-2.9.8-1.el8.x86_64.rpm
# Instala el paquete nano

[root@centos8 ~]# rpm -q nano
nano-2.9.8-1.el8.x86_64

[root@centos8 ~]# rpm -ql nano
/usr/bin/nano
/usr/share/doc/nano
/usr/share/man/man1/nano.1.gz

[root@centos8 ~]# rpm -e nano
# Elimina el paquete nano

[root@centos8 ~]# rpm -qa | grep nano
# Comprueba si nano sigue instalado
```

Nota: Para ver a qué paquete pertenece un archivo, puedes usar `rpm -qf /ruta/al/archivo`.

yum

Facilita la instalación de paquetes .rpm, resolviendo automáticamente las dependencias y gestionando los repositorios. Es el gestor estándar en distribuciones como CentOS, RHEL y Fedora (en versiones recientes, `yum` es reemplazado por `dnf` el cual mantiene toda la compatibilidad con los comandos de `yum`).

- `install`: Instala uno o varios paquetes.
- `check-update`: Comprueba si hay actualizaciones disponibles para los repositorios. Es el equivalente a *apt update*.
- `update`: Actualiza todos los paquetes instalados o uno específico. Es el equivalente a *apt upgrade*.
- `search`: Busca paquetes por nombre o descripción.
- `list`: Lista paquetes disponibles, instalados o actualizables.
- `info`: Muestra información detallada de un paquete.
- `remove`: Elimina uno o varios paquetes.
- `autoremove`: Elimina paquetes huérfanos (no requeridos por otros).
- `repolist`: Lista los repositorios configurados y su estado.
- `downloadonly`: Descarga únicamente el paquete sin instalarlo.
- `downloaddir`: Indica el lugar donde se descarga.
- `grouplist`: Nos da el conjunto de metapaquetes que contienen otros paquetes y están agrupados en base a una utilidad concreta, como la gestión de redes, servidores web...

Nota: Los repositorios de yum se configuran en archivos con extensión `.repo` dentro de `/etc/yum.repos.d/`.

```
[root@centos8 ~]# yum install nano
# Instala el paquete nano y todas sus dependencias

[root@centos8 ~]# yum list installed nano
# Muestra si nano está instalado

[root@centos8 ~]# yum remove nano
# Elimina el paquete nano

[root@centos8 ~]# yum search editor
# Busca paquetes relacionados con "editor"

[root@centos8 ~]# yum update
# Actualiza todos los paquetes instalados

[root@centos8 ~]# yum clean all
# Limpia la caché de yum
```

Nota: `yum` resuelve automáticamente las dependencias y descarga los paquetes desde los repositorios configurados, facilitando la gestión de software en sistemas basados en RPM. *Nota*: A parte de `yum` existe el comando `dnf`. *YUM (Yellowdog Updater, Modified)* es un gestor de paquetes para sistemas Linux basados en RPM, como Red Hat, CentOS y Fedora. Permite instalar, actualizar y eliminar paquetes y sus dependencias automáticamente desde repositorios de software *DNF (Dandified YUM)* es el sucesor moderno de YUM, diseñado para ser más rápido, eficiente y con mejor resolución de dependencias. Es el gestor de paquetes predeterminado en las versiones recientes de Fedora y otras distribuciones RPM, manteniendo comandos similares a YUM pero con mejoras en rendimiento y manejo de transacciones. A mayores encontramos herramientas como *zypper* desarrolladas por openSUSE.