6. GESTIÓN DE PAQUETES DE SOFTWARE

6.1.- Introducción

En 1995 Red Hat desarrolló RPM (**R**ed Hat **P**ackage **M**anager), una forma estándar de gestionar y empaquetar software en LSB para su distribución.

Actualmente muchas distribuciones Linux utilizan el formato RPM: RHEL, Fedora, CentOS, Mandriva, Mageia, Suse Linux, Open Suse, ...

RPM permite a los administradores rastrear qué archivos se instalan/modifican/eliminan con un paquete cuando el paquete se instala/modifica/elimina y chequea que los paquetes necesarios, llamados dependencias, estén instalados previamente en el sistema.

Cuando hablamos de RPM, nos referimos a tres componentes en conjunto: la base de datos RPM, los archivos RPM y el comando *rpm*.

6.1.1.- Base de datos RPM

Es donde se almacena la información de los paquetes instalados en el sistema. Es una base de datos interna local situada en */var/lib/rpm* de archivos hash de base de datos binarios.

Se interacciona con ella con el comando *yum* o *rpm* pero no se administra directamente.

6.1.2.- Archivos RPM

Similar a los archivos tar, un archivo RPM es un empaquetado que contiene múltiples archivos para ser instalados en el sistema con un propósito.

Los archivos RPM tienen un formato estándar en su nombre de archivo:

<nombre_paquete>-<version>-<release>.<arquitectura>

donde:

<*nombre_paquete*>: nombre del empaquetado de software. Será la entrada de base de datos que identifica el paquete.

<*version*>: número de versión del software original.

<*release*>: número de revisión de la versión del paquete.

<*arquitectura*>: tipo del procesador para el cual el archivo fue compilado, p.e. i386, x86_64. Si aparece *noarch* significa que es independiente del tipo de procesador.

Cuando se instalan paquetes desde repositorios, sólo es necesario dar el nombre del paquete y se instalará la última versión disponible en los repositorios activos configurados.

Un archivo RPM tiene tres partes:

- Archivos que se instalan con el paquete.

- Metadatos del paquete, como son su nombre, versión, release, arquitectura, descripción, resumen (pequeña descripción), qué paquetes es necesario que estén previamente instalados en el sistema, licencia, …

  Los paquetes RPM suelen estar firmados digitalmente por la empresa que los proporciona, con una clave privada GPG. Si el paquete se altera de alguna forma o se corrompe, la firma no será válida. Esto permite al sistema verificar la integridad de un paquete antes de instalarlo.
- Scripts que deben ejecutarse al instalar/actualizar/eliminar el propio paquete o que deben ejecutarse al instalar/actualizar/eliminar otros paquetes relacionados con el paquete del archivo RPM. Estos scripts se ejecutan como usuario *root*, por lo que hay que tener cuidado de que los paquetes que se instalen en el sistema sean de fiar.

Tras la actualización de un paquete, no suele ser necesario que la versión anterior esté disponible, luego en el proceso de instalación de una versión superior, puede que se elimine el paquete antiguo.

En el caso de una actualización/instalación del paquete del kernel de Linux, el kernel anterior sí permanece en el equipo permitiendo que, si el nuevo kernel no funciona bien, podamos arrancar en un kernel anterior y eliminar el nuevo kernel problemático sin perder funcionalidad en el sistema.

Los archivos que contiene un archivo RPM se pueden extraer sin necesidad de instalarlos en el sistema usando una herramienta parecida a *tar* o *zip* llamada *cpio*. Se le pasa la salida del comando *rpm2cpio <nombre_paquete>.rpm* por un pipe al comando *cpio -id* y se obtienen los archivos que contiene el paquete. Ejemplo:

```
rpm2cpio ntfs-3g-2015.3.14-2.el7.x86_64.rpm | cpio -id
tree ./
.
└── usr
    ├── bin
    │   ├── lowntfs-3g
    │   ├── ntfs-3g
    │   ├── ntfs-3g.probe
    │   ├── ntfs-3g.secaudit
    │   ├── ntfs-3g.usermap
    │   └── ntfsmount -> ntfs-3g
    ├── lib64
    │   ├── libntfs-3g.so.86 -> libntfs-3g.so.86.0.0
    │   └── libntfs-3g.so.86.0.0
    ├── sbin
    │   ├── mount.lowntfs-3g -> /usr/bin/lowntfs-3g
    │   ├── mount.ntfs -> mount.ntfs-3g
……..
```

6.1.3.- Comando rpm

El comando *rpm* es una utilidad de bajo nivel que se usa principalmente para obtener información de un paquete obteniendo dicha información de la base de datos interna de RPM.

También es capaz de instalar/actualizar/eliminar paquetes del sistema pero no se recomienda, es mejor usar el comando *yum* que almacena un histórico de transacciones de lo que se ha instalado en el sistema y resuelve dependencias.

El comando *rpm* no resuelve dependencias, los paquetes prerequisitos para instalar un paquete, deben estar instalados en el sistema previamente.

Cuando usamos el comando *rpm*, el nombre del paquete se puede especificar además de dando la ruta al archivo RPM o el nombre del paquete, usando una URL de HTTP o FTP; el comando descargará el archivo de dicha URL a local en */var/tmp* antes de ejecutar la acción pedida.

Usos del comando *rpm*:

→ *Consulta de los paquetes instalados*: usando la opción *-q* o *--query* combinada con :

- *-a*: todos los paquetes instalados en el sistema. Añadiendo además *--last* ordena la salida por fecha de instalación apareciendo primero los más recientes.
- *-f <archivo>*, *--whatprovides <archivo>*: qué paquete proporciona ese archivo.
- *--whatrequires <archivo>*: qué paquetes necesitan el archivo dado.
- Sin opciones pasando el nombre de un paquete, me devuelve el nombre del archivo RPM correspondiente a ese paquete.

→ *Obtener información de un archivo RPM*: usando la opción *-q* o *--query* combinada con:

- *-i <nombre_paquete>*: da información del paquete.
- *-l <nombre_paquete>*, *--provides <nombre_paquete>*: lista los archivos que contiene el paquete.
- *-R*, *--requires <nombre_paquete>*: dependencias del paquete.
- *-c <nombre_paquete>*: archivos de configuración que contiene el paquete.
- *-d <nombre_paquete>*: archivos de documentación que contiene el paquete.
- *--scripts <nombre_paquete>*: scripts que contiene el paquete.

→ *Instalar/actualizar/eliminar un paquete:*

- *-i <archivo_paquete>.rpm*: se instala el paquete dado comprobando previamente si existen las dependencias necesarias, que no se sobrescriba ningún archivo y si hay suficiente espacio en disco. Si no se cumple alguna de las condiciones, no se instala. Añadiendo las opciones *-p* y *-h* muestra el porcentaje de la instalación usando el carácter #.

- *-U <archivo_paquete>.rpm*: actualiza el paquete dado. La opción *--oldpackage* con una versión más antigua del paquete a la que haya instalada en el sistema, hará un downgrade del paquete en lugar de un upgrade.
- *-e <nombre_paquete>*: elimina del sistema el paquete con el nombre dado si el paquete no es una dependencia de otro paquete del sistema. Se puede forzar la eliminación en este caso si añadimos *--nodeps* pero es peligroso hacerlo así ya que los paquetes dependientes del que estamos eliminando no funcionarán bien.

A estas opciones se les puede añadir *--test* que hace la comprobación de lo que ocurriría al instalar/actualizar/eliminar el paquete pero no lo realiza. Con *--nocripts* no ejecutaría los scripts que contiene el paquete.

→ *Otros usos de rpm*:

- *rpm --import <archivo_clave_publica>*: instala la clave pública dada en */var/lib/rpm*. Se pueden ver todas las claves públicas instaladas con *rpm -aq gpg-key**.
- *rpm -Va*: verifica la integridad de todos los paquetes instalados listando los paquetes modificados tras su instalación.
- *rpm --checksig <nombre_paquete>, rpm -V <nombre_paquete>*: verifica la integridad de un paquete dado.
- *rpm --setperms <nombre_paquete>*: restablece los permisos de los archivos del paquete a como estaban cuando se instaló el paquete.
- *rpm --rebuilddb*: se reconstruye la base de datos de rpm si exiten problemas de corrupción.

NOTA: No se debe actualizar/instalar una nueva versión del kernel del sistema usando *rpm*, la recomendación es usar el comando *yum*.

6.2.- Getionar paquetes con yum

Yum (**Y**ellowdog **U**pdater **M**odified) es una herramienta libre escrita en Python de gestión de paquetes basados en RPM diseñada para resolver dependencias. Se usa tanto para consultar información de paquetes instalados en el sistema como paquetes existentes en los repositorios, instalar paquetes individuales o grupos de paquetes y mantiene un histórico de transacciones.

Automáticamente determina las dependencias necesarias y las instala buscándolas en los repositorios activos que haya configurados en el sistema.

Yum almacena en un histórico las transacciones que se van realizando al ejecutar el comando *yum*.

Una transacción es todas las operaciones que se realizaron en una ejecución del comando *yum* para instalar/actualizar/eliminar un/os paquete/s.

Yum maneja grupos de paquetes, conjunto de paquetes que se instalan juntos con un propósito como por ejemplo, el grupo de paquetes llamado “Office Suite and Productivity” que incluye los paquetes de LibreOffice de *writer* (procesador de textos), *calc* (hoja de cálculo), *impress* (presentaciones),…

Dentro de un grupo de paquetes, hay paquetes predeterminados (los necesarios para el resto de paquetes del grupo), obligatorios (paquetes importantes del grupo) y optativos (paquetes extra).

Cuando se instala un grupo de paquetes, se instalarán los paquetes por defecto y obligatorios, este comportamiento se puede modificar para que los instale todos en el archivo de configuración de Yum */etc/yum.conf* cambiando el parámetro *group_package_types* de *“default, mandatory”* a *“default, mandatory, optional”*. Sin embargo, cuando se elimina un grupo de paquetes, no se tiene en cuenta de qué tipo son los paquetes del grupo, se eliminan todos los paquetes pertenecientes al grupo.

Un usuario normal, puede hacer consultas usando el comando *yum* pero habría que darle permisos para permitirle hacer modificaciones en sistema como instalar, actualizar, eliminar, ...

El archivo de configuración principal de Yum está en */etc/yum.conf* y los archivos de configuración de los repositorios en */etc/yum.repos.d*, con extensión *.repo*. Además tenemos un archivo de log llamado *yum.log* dentro de */var/log* que registra todos los sucesos de Yum.

6.2.1.- Subcomandos de yum

Es muy habitual cuando se administra un sistema, instalar paquetes y es importante conocer a fondo todas las posibilidades que ofrece el comando *yum* a través de sus subcomandos. El formato del comando *yum* es:

```
yum <subcomando> <opciones> [argumentos]
```

Tenemos subcomandos de *yum* de:

→ *Instalación*:

- *install <nombre-paquete>:* Instala la última versión existente en los repositorios activos el paquete dado. Instala los paquetes de dependencias que se necesiten. Se pueden poner varios paquetes separados por espacios.
- *install <archivo_paquete>.rpm* o *localinstall <archivo_paquete>.rpm*: Instala el paquete desde el archivo rpm en la versión que exista en dicho archivo.
- *group install <nombre_grupo_paquete>* o *groupinstall <nombre_grupo_paquete>*: instala los paquetes por defecto y obligatorios del grupo de paquetes.

→ *Actualización*:

- *update*: actualiza a la última versión todos los paquetes instalados en el sistema de los cuales existan actualizaciones en los repositorios activos. Si se añade --*exclude*=<*nombre_paquete*>, actualiza todos los paquetes menos los listados en la opción --*exclude*.
- *update* <*nombre_paquete*>: actualiza a la última versión disponible en los repositorios activos el paquete dado. Se pueden poner varios paquetes separados por espacios
- *localupdate* <*archivo_paquete*>*.rpm*: actualiza el paquete que corresponda al archivo de paquete dado a la versión existente en el archivo RPM.
- *group update* <*nombre_grupo_paquetes*>: actualiza a la última versión todos los paquetes instalados con el subcomando *group install* o *groupinstall* del grupo de paquetes dado.

NOTA: En el caso de actualización del kernel, realmente el nuevo kernel se instala sin eliminar el anterior pese a que se ejecute un *yum upgrade* en lugar de un *yum install*. En los archivos de configuración de Yum, existe una lista de paquetes que, pese a que se indique una actualización, realmente se hace una instalación y el kernel está entre ellos.

→*Eliminación*:

- *erase* <*nombre_paquete*> o *remove* <*nombre_paquete*>: elimina del sistema el paquete dado.
- *group remove* <*nombre_grupo_paquetes*> o *groupremove* <*nombre_grupo_paquetes*>: Elimina los paquetes instalados con el subcomando *group install* o *groupinstall* del grupo de paquetes dado.

→ *Información*:

- *check-update*: muestra las actualizaciones disponibles de los paquetes instalados en los repositorios activos. No actualiza nada, sólo muestra información.
- *check-update* <*nombre_paquete*>: muestra la actualización disponible del paquete dado.
- *deplist* <*nombre_paquete*>: lista las dependencias del paquete dado.
- *list*: muestra los paquetes instalados en el sistema y disponibles en los repositorios activos.
- *list installed*: lista todos los paquetes instalados en el sistema
- *list updates*: lista todos los paquetes actualizados en el sistema.
- *list kernel*: lista los kernel instalados en el sistema y disponibles en los repositorios activos.
- *provides* <*archivo*>, *provides* <*ruta*>: muestra el paquete (instalado o disponible en los

repositorios) que proporciona el archivo o la ruta dada.

- *search <cadena>*: busca la cadena dada en el nombre o descripción de los paquetes instalados y disponibles.
- *info <nombre_paquete>*: da información del paquete dado si está instalado. Si no lo está lo busca en los repositorios activos.
- *group list* o *grouplist*: lista los grupos de paquetes instalados y disponibles.
- *group info <nombre_grupo_paquetes>* o *groupinfo <nombre_grupo_paquetes>*: Muestra la información de los paquetes incluidos en el grupo de paquetes dado. Delante del nombre del paquete aparece + (el paquete se instalará al instalar/actualizar el grupo), - (el paquete no se instalará al instalar/actualizar el grupo), = (el paquete está instalado y se instaló al instalar/actualizar el grupo) o nada (si el paquete fue instalado pero de forma independiente, no con la instalación del grupo).
- *group summary*: muestra los grupos instalados y disponibles con una breve descripción.
- *repolist*: lista los repositorios configurados que estén activos y el número de paquetes que contienen. Añadiendo *enabled*, *disabled* o *all* se muestran los repositorios activos, desactivados o todos respectivamente. Delante del nombre del repositorio puede aparecer el carácter **!** si los metadados del repositorio han expirado.
- *repoinfo*, *repoinfo <repositorio>*: muestra la información de todos los repositorios o la información del repositorio dado.

→ *Otros*:

- *yum clean*: limpia las caches de yum.
- *yum clean packages*: elimina las versiones anteriores de los paquetes actualizados.
- *yum makecache*: obtiene la metainformación de los repositorio.
- *yum history*: muestra las transacciones. Cada transacción tiene un id asociado que usaremos para ver la transacción en detalle con *yum history info <id>*. Para deshacer todo lo que se hizo en una transacción se ejecuta *yum history undo <id>*, para volver a realizar lo que se hizo en una transacción *yum history redo <id>*. También existe la posibilidad de deshacer todas las transacciones hechas desde una dada incluida esa con *yum history roll-back <id>*.
- *yum ckeck:* chequea la base de datos de RPM en busca de problemas.
- *yumdb search from_repo <nombre_repo>*: lista los paquetes instalados desde el repositorio dado.

En los subcomandos de yum de *install, update, remove,info* se puede usar en lugar del nombre del paquete, el nombre del grupo de paquetes precedido del carácter @ y es equivalente a usar los subcomandos de *groupinstall, groupupdate, groupremove, groupinfo*.

6.2.2.- Opciones relevantes del comando yum

Las opciones más habituales del comando *yum* en cuanto a la gestión de paquetes son:

- *-y*: para que los subcomandos de instalación, actualización y desinstalación no pidan confirmación. No es recomendable usar esta opción en desinstalaciones.
- *--nogpgcheck*: no verifica las firmas digitales.
- *--downloadonly:* guarda en local el archivo RPM del paquete sin realizar la acción indicada en el comando. Por defecto en */var/cache/yum/<arq>/<versionSO>/<repo>/packages;* se puede añadir la opción *--downloaddir <directorio>* para que lo guarde en ese directorio en concreto.

6.2.3.- Gestión de repositorios yum

Un repositorio Yum es la localización donde se almacenan los paquetes RPM y se hace disponible dicha localización al sistema local o vía HTTP o FTP a otros sistemas.

Cuando se instala un sistema CentOS, por defecto vienen configurados los repositorios públicos del proyecto CentOS pero en otras distribuciones no es así.

→ Configurar un repositorio

Basta crear su archivo de configuración con extensión *.repo* en */etc/yum.repos.d*. En dicho archivo aparece el id del repositorio entre corchetes (nombre único en el sistema para referirnos al repositorio, sin espacios) y un parámetro por línea con el formato <parámetro>=<valor>. Los parámetros son:

- *name*: nombre del repositorio, descripción de hasta 255 caracteres alfanuméricos.
- *baseurl*: una URL (http://, ftp:// o file://) con la ubicación del repositorio (directorio padre del directorio repodata).
- *enabled*: valores 1 o 0 para indicar si el comando *yum* tiene que usar el repositorio o no. Si no aparece, su valor por defecto es 1.
- *gpgcheck*: valores 1 o 0 para indicar si tiene que verificar la clave GPG definida en el parámetro *gpgkey* (o instalada en el sistema) o no. Si no aparece, por defecto su valor es 1.
- *gpgkey*: URL de la clave GPG si *gpgkey* es 1. Si hemos instalado la clave pública con *rpm --import*, no es necesario que aparezca.

También se puede configurar un repositorio utilizando el comando *yum-config-manager*, del paquete *yum-utils*, con la opción *--add-repo* y la URL al archivo .repo del repositorio como argumento. El comando se encargará de crear en local el archivo .repo con los valores por defecto de los parámetros.

→ *Crear un repositorio*

Otra cosa es que queramos crear un repositorio en un sistema de archivos local. Para esto, utilizaremos el comando *createrepo*, del paquete con el mismo nombre. Para crear un repositorio seguiremos los pasos:

1. Instalar el paquete *createrepo.*

2. Crear el directorio donde va a estar el repositorio.

3. Copiar en el directorio creado los paquetes rpm que queramos que estén en el repositorio.

4. Ejecutar el comando *createrepo* pasándole como argumento el directorio creado anteriormente. Cada vez que se hagan cambios en el directorio, habrá que volverlo a ejecutar.

5. Si queremos usar el repositorio de forma local, crear una configuración local para el repositorio. Si queremos que se pueda usar el repositorio por HTTP o FTP, habrá que configurar estos servicios para que hagan disponible el repositorio a través de ello y en el sistema donde se vayan a usar, crear una configuración de repositorio remoto.

→ *Activar/desactivar configuraciones de repositorios:*

- de forma puntual en la ejecución de un comando *yum*, con las opciones:

  --enablerepo=<id_repo>: además de los repositorios activos, se activa el repositorio dado. Si en lugar de poner el id de un repositorio, se pone '*' (con las comillas simples), activa todos los repositorios configurados en el sistema.

  --disablerepo=<nombre_repo>: se desactiva el repositorio dado. Si en lugar de poner el id de un repositorio, se pone '*' desactiva todos los repositorios.

  Ambas opcione se pueden usar a la vez.

- de forma persistente: se puede modificar a mano el parámetro enabled a 0 o a 1 del archivo de configuración del repositorio o utilizar *yum-config-manager --active <id_repositorio>* o *yum-config-manager --disable <id_repositorio>*. Si en lugar de poner el id de un repositorio utilizamos * lo hace sobre todos los repositorios configurados.

6.3.- Caso práctico

En la instalación de **central** se configuró un repositorio de CentOS 7 de la unidad de CD que tiene configurada con el nombre de *centos7*. Además se va a configurar el repositorio público EPEL (proyecto open que contiene un conjunto de paquetes adicionales para Enterprise Linux) con el id *epel* ubicado en http://mirror.airenetworks.es/epel//7/x86_64/. En **central** se desactivarán los repositorios de CentOS que vienen con la instalación dejando sólo los repositorios de *centos7* y *epel*.

En **central** se creará un repositorio local llamado *centralpackages* que estará en */var/www/html/centralpackages* para que desde **server1** se pueda configurar. Dicho repositorio va a contener el paquete *ntfs-3g* (utilidades para poder acceder a particiones con formateo NTFS) que previamente se obtendrá del repositorio EPEL. El acceso al repositorio *centralpackages* desde **server1** se hará por HTTP.

En **server1** sólo deben estar activos los repositorios de *centralpackages* y *centos7*.

Tras configurar los repositorios, en **server1** se instalarán los paquetes *bash-completion*, *ntfs-3g* y *tree*.

RESOLUCIÓN

- Vemos cómo está configurado el repositorio *centos7* en central:

La configuración de kickstart de **central**, *central.ks*, hace que de forma persistente se monte la unidad de CD, donde está la ISO de CentOS7, en */var/www/html/centos7/repo*, dentro del document root de Apache. Esto es, existe una línea en el */etc/fstab*:

```
UUID="2016-12-05-13-52-39-00"   /var/www/html/centos7/repo      iso9660 _netdev 0 0
```

de forma que, usando la URL http://www.central.miempresa.com/centos7/repo podamos configurar un repositorio remoto en **server1**.

Además, en dicho archivo kickstart, ya se ha configurado un repositorio local de este punto de montaje. Se creó el archivo */etc/yum.repos.d/centos7.repo* con el contenido:

```
[centos7]
name=Repositorio local CentOS 7
baseurl=file:///var/www/html/centos7/repo
enabled=1
gpgcheck=1
gpgkey=file:///var/www/html/centos7/repo/RPM-GPG-KEY-CentOS-7
```

Vamos a dejar activo sólo este repositorio en **central**, primero desactivamos todos los repositorios y comprobamos que no hay ninguno activo:

```
[root@central ~]# yum-config-manager --disable \*
```

```
[root@central ~]# yum repolist
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
repolist: 0
```

Ahora activamos el repositorio con *id centos7* y comprobamos que sólo está este activo:

```
[root@central ~]# yum-config-manager --enable centos7
[root@central ~]# yum repolist
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
repo id                repo name                          status
centos7                Repositorio local CentOS 7         9,363
repolist: 9,363
```

- Configuramos el repositorio con id *centos7* en **server1**, tenemos dos posibilidades: crear a mano el archivo de configuración .repo en */etc/yum.repos.d* o usar el comando *yum-config-manager --add-repo http://central.miempresa.com/centos7/repo*. Como para la segunda opción, necesitamos instalar el paquete *yum-utils*, y puede que no tengamos configurados repositorios, lo hacemos con la primera opcion.

1.- Instalamos en **server1** la clave GPG del repositorio:

```
[root@server1 ~ ]# rpm --import http://central.miempresa.com/centos7/repo/RPM-GPG-KEY-CentOS-7
```

2.- Creamos en **server1** el archivo */etc/yum.repos.d/centos7.repo* con el contenido:

```
[centos7]
name=Repositorio local CentOS 7 en central
baseurl=http://www.central.miempresa.com/centos7/repo
enabled=1
```

3.- Instalamos el paquete *yum-utils* haciendo que use el repositorio *centos7*:

```
[root@server1 ~] yum install --disablerepo='*' --enablerepo=centos7 yum-utils
```

4.- Eliminamos todos los repositorios salvo *centos7*:

```
[root@server1 ~]# yum-config-manager --disable \*
[root@server1 ~]#  yum-config-manager --enable centos7
```

5.- Verificamos con *yum repolist*:

```
[root@server1 ~ ]# yum repolist --disablerepo='*' --enablerepo=centos7
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
```

repo id repo name status

centos7 Repositorio local CentOS 7 en central 9,363

repolist: 9,363

- Configuramos en **central** el repositorio *epel* de http://mirror.airenetworks.es/epel//7/x86_64/:

[root@central ~]# rpm -import http://mirror.airenetworks.es/epel//RPM-GPG-KEY-EPEL-7

[root@central ~]# cat > /etc/yum.repos.d/epel.repo << EOF

[epel]

name=Repositorio EPEL

baseurl= http://mirror.airenetworks.es/epel//7/x86_64/

enabled=1

EOF

[root@central ~]# yum repolist

Loaded plugins: fastestmirror

Loading mirror speeds from cached hostfile

repo id repo name status

centos7 Repositorio local CentOS 7 9,363

epel Repositorio EPEL 11,817

repolist: 21,180

- Creamos el repositorio en **central** *centralpackages:*

[root@central ~]# yum install -y createrepo

[root@central ~]# mkdir /var/www/html/centralpackages

[root@central ~]# yum info ntfs-3g

Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Available Packages
Name : ntfs-3g
Arch : x86_64
Epoch : 2
Version : 2017.3.23
Release : 1.el7
Size : 263 k
Repo : epel
Summary : Linux NTFS userspace driver
URL : http://www.ntfs-3g.org/
License : GPLv2+
Description : NTFS-3G is a stable, open source, GPL licensed, POSIX, read/write NTFS
: driver for Linux and many other operating systems. It provides safe
: handling of the Windows XP, Windows Server 2003, Windows 2000, Windows
: Vista, Windows Server 2008 and Windows 7 NTFS file systems. NTFS-3G can
: create, remove, rename, move files, directories, hard links, and streams;
: it can read and write normal and transparently compressed files, including

```
                : streams and sparse files; it can handle special files like symbolic links,
                : devices, and FIFOs, ACL, extended attributes; moreover it provides full
                : file access right and ownership support.
[root@central ~]# yum install --downloadonly
                                        --downloaddir=/var/www/html/centralpackages ntfs-3g

[root@central ~]# ls /var/www/html/centralpackages/

ntfs-3g-2017.3.23-1.el7.x86_64.rpm

[root@central ~]# createrepo /var/www/html/centralpackages/

[root@central ~]# ls /var/www/html/centralpackages/
ntfs-3g-2017.3.23-1.el7.x86_64.rpm  repodata
```

Ahora tendremos un repositorio en **central** accesible desde la URL:
http://central.miempresa.com/centralpackages

- Configuramos el repositorio *centralpackages* en **server1**, lo haremos con *yum-config-manager*:

```
[root@server1 ~]# yum-config-manager --add-repo
http://central.miempresa.com/centralpackages
Loaded plugins: fastestmirror
adding repo from: http://central.miempresa.com/centralpackages

[central.miempresa.com_centralpackages]
name=added from: http://central.miempresa.com/centralpackages
baseurl=http://central.miempresa.com/centralpackages
enabled=1
[root@server1 ~]# yum repolist
Loaded plugins: fastestmirror
central.miempresa.com_centralpackages                    | 2.9    kB  00:00:00
central.miempresa.com_centralpackages/primary_db         | 2.6 kB  00:00:00
Loading mirror speeds from cached hostfile
repo id          repo name                                status
centos7          Repositorio local CentOS 7 en central    9,363
central.miempresa.com_centralpackages   added from: …     1
repolist: 9,364
```

- Instalamos en **server1** los paquetes *bash-completion*, *ntfs-3g* y *tree*:

```
[root@server1 ~]# yum install -y bash-completion ntfs-3g tree
```