



LIBERTONIA

"Las noticias que le interesan. Si no le interesan, tenemos otras"

Por [amphora](#)

departamento ¿Eso que tienes ahí es un chupetón? , Sección [Software Libre](#)
Puesto a las Thu May 6th, 2004 at 03:24:41 PM CET

Bacula, el vampiro

Trás un tiempo probando amanda, como sistema de backups en cinta y a raíz de una entrada en el [diario](#) de Iñaki Arenaza decidí darle un tiento a bacula.

Si alguno ha probado amanda es posible que se haya sentido bastante perdido en su configuración, la documentación es muy escasa (o al menos a mi me lo parece) y no dá la sensación de que se esté desarrollando activamente. Nunca lo he llegado a poner en producción, tan solo he estado haciendo pruebas y aunque funciona perfectamente, me quedaba ese regustín a que no era todo lo bueno que debiera ser. Como no conocía otras alternativas libres, terminé su configuración y pruebas y no volví a tocar el tema, hasta el momento en que tuviera que ponerse en marcha.

Como ya he dicho, gracias a la entrada de Iñaki en su diario, puse Galeón apuntando a su url, y me he encontrado con un sistema de backup que colma todas mis necesidades (salvo diferentes cuestiones que ya trataré). La documentación es extensa y muy bien estructurada, de hecho se escribe antes que el código y es posible encontrar características que todavía no se han implementado. Todas las tareas que realiza bacula se han modularizado y repartido las tareas entre varios demonios y servicios y además para los que lo necesiten hay un cliente para win32.

Comprendiendo a bacula

Cuesta cinco minutos comprender el funcionamiento del sistema, gracias a la documentación y a la separación lógica de las tareas a realizar. La configuración es algo más compleja, pero en realidad no mucho más, y para eso ya estoy yo escribiendo este artículo. De nada.

Quiero hacer una aclaración, aunque hablo de un sistema de backup en cinta, no quiere decir que unicamente se pueda volcar en ese tipo de dispositivos, se puede utilizar perfectamente en una red doméstica, o en una sola máquina y grabar los respaldos en un CD, pero obviamente bacula está orientado a una red más grande y con uno o varios dispositivos de cinta.

Bacula guarda todas sus operaciones, trabajos, listas de volúmenes, etc en una base de datos estilo SQL. Toda la instalación la he realizado en sistemas debian, salvo algunos clientes, que utilizan otra serie de distribuciones (Red Hat y Mandrake), así que la instalación de bacula se redujo a un apt-get y el solito me creo las bases de datos necesarias. ¿Como se hace en otras distribuciones? Ni idea, en las otras me he limitado a configurar los clientes

Bueno, que me enrolló mucho. El sistema se divide en:

- Bacula director El demonio encargado de gestionar todas las operaciones de backup. El director sabe los trabajos que se van a realizar, cuando , donde y como. Y además se encarga de restaurar los ficheros que le pidamos y su verificación (una especie de suma de comprobación de integridad). Se puede instalar en cualquier máquina de la red.
- Bacula File El cliente. Es necesario instalarlo en todas las máquinas de las que queramos hacer respaldo. Su función es leer y transmitir los ficheros que el director le pida, o restaurarlos.
- Bacula Storage Este demonio se encarga de la lectura/escritura física en los volúmenes que estén definidos (cintas, ficheros)

Además tenemos la consola, con la cual nos conectaremos al director, y desde donde podremos dar ordenes, hacer consultas, etc. Y el catálogo, donde el director guarda y registra todas sus

operaciones.

Aclarado esto, conviene también explicar que son los volúmenes y los "pools":

Bacula se refiere a los volúmenes como los dispositivos físicos donde se guardan los volcados (ficheros o cintas) y a los "pools" como el conjunto de uno o varios volúmenes. Para que quede más claro, nosotros definimos por ejemplo un "pool" que va a servir para hacer copias diarias de diferentes máquinas, y en ese "pool" añadimos diferentes volúmenes y bacula sabe cuales volúmenes pertenecen a cada "pool". De esta manera, bacula conoce donde está cada cosa, y de donde sacarla, o bien donde será necesario escribir cuando el volumen actual esté lleno.

El funcionamiento viene a ser algo así:

El director, que tiene definidos una serie de trabajos y a una determinada fecha y hora se pone en marcha, contacta con el primer cliente que tiene definido, el cliente de esa máquina comprueba los ficheros o directorios que le pide el director y le devuelve los que hayan cambiado (si es una copia diferencial o incremental) al director, este se pone en contacto con el demonio "storage" que almacena en el volumen los ficheros que le mandan, y vuelta a empezar por cada uno de los clientes. Para un trabajo de verificación viene a ser lo mismo, solo que no se guardan los ficheros, sino solo las sumas de comprobación en la base de datos SQL que usemos (a día de hoy SQLite, MySQL o PostgreSQL)

Instalación

Aunque ya he comentado, que la instalación a partir de binarios es sumamente fácil, hay que comentar algunas cosas:

- El kernel Si no viene configurado por defecto (como mi caso) es necesario recompilar el kernel de la máquina que tenga conectado el dispositivo de cinta con soporte para scsi y cintas:

```
<*> SCSi support
- - - SCSi support type (disk, tape, CD-ROM)
<*> SCSi disk support
(40) Maximum number of SCSi disks that can be loaded as modules
<M> SCSi tape support
```
- Los demonios: Al trabajar en red, la base datos puede estar en cualquier parte, el director también, los clientes cada uno en su máquina y el demonio que trata con la cintas en la máquina que tiene las cintas. Mi opinión es no complicarse la vida e instalar el director, la base de datos y el demonio encargado de grabar datos en la máquina con el scsi de cintas. Y la consola en el ordenador normal de trabajo (aunque al instalar el cliente en debian, se te instala también la consola)

Entonces tenemos que:

```
maquina@directora:~#apt-get install bacula-director-mysql bacula-sd
```

```
maquina@cliente:~#apt-get install bacula-fd
```

Una vez que está instalado y configurado, nos ponemos a retocar los ficheros de configuración. Todos los ficheros de configuración son bastante simples y apenas requieren modificación alguna, salvo el del demonio director que es bastante extenso y complicadillo. En realidad, en las últimas versiones de bacula se ha simplificado bastante este fichero ya que se ha implementado un recurso que permite tener definidos una serie de trabajos modelo, que despues se pueden modificar puntualmente para cada cliente/trabajo. Viene a ser algo parecido a las clases de la programación orientada a objetos. La pena es que en debian unstable van por la versión 1.32 y la versión upstream es la 1.34, no pudiendose hacer uso de esa característica. A mi me queda un fichero tal que así:

```
maquina@directora:~# wc -l /etc/bacula/bacula-dir.conf
1099 /etc/bacula/bacula-dir.conf
```

Dejamos para la segunda parte la configuración de los ficheros

Continuamos con la serie. Esta vez hablamos como prometí de los diferentes ficheros de configuración de este estupendo vampiro

Los ficheros de configuración de bacula

Continuamos con la serie. Esta vez hablamos como prometí de los diferentes ficheros de configuración de este estupendo vampiro.

Vamos a ir de los más fáciles a los más difíciles:

- **/etc/console.conf**

La consola de bacula es el medio por el cual nos vamos a comunicar con el sistema director. Su fichero de configuración es este:

```
gandalf:/etc/bacula# cat console.conf
#
# Bacula User Agent (or Console) Configuration File
#

Director {
    Name = maquinadirectora- dir
    DIRport = 9101
    address = maquinadirectora.x.com
    Password = "clave"
}
```

El nombre ha de ser el mismo que más tarde definiremos en el fichero de configuración del director. Lo siguiente es el puerto donde escucha el director. El FQDN o bien su dirección IP. Y finalmente la password que se necesita para conectar al director.

- **/etc/bacula- fd.conf**

El fichero de configuración de los clientes

```
gandalf:/etc/bacula# cat bacula- fd.conf
#
# List Directors who are permitted to contact this File daemon
#
Director {
    Name = maquinadirectora- dir
    Password = "clave"
}

#
# "Global" File daemon configuration specifications
#
FileDaemon {
    Name = gandalf- fd
    FDport = 9102
    WorkingDirectory = /var/lib/bacula
    Pid Directory = /var/run/bacula
}

# Send all messages except skipped files back to Director
Messages {
    Name = Standard
    director = maquinadirectora- dir = all, !skipped
}
```

Los comentarios del fichero aunque estén en ingles, son bastante aclaratorios, así que no voy a entrar en detalles. Basicamente, se autoriza al director a conectar con nosotros por el

puerto 9102. Definimos el nombre de nuestro cliente, y se definen el tipo de mensajes que queremos hacer llegar al director.

- **/etc/bacula- sd.conf**

El demonio encargado de escribir en los dispositivos de cinta tiene su propio fichero:

```
maquinadirectora:/etc/bacula# cat bacula- sd.conf
# You may need to change the name of your tape drive
# on the "Archive Device" directive in the Device
# resource. If you change the Name and/or the
# "Media Type" in the Device resource, please ensure
# that dird.conf has corresponding changes.
#

Storage {
    # definition of myself
    Name = maquinadirectora- sd
    SDPort = 9103          # Director's port
    WorkingDirectory = "/var/lib/bacula"
    Pid Directory = "/var/run/bacula"
}

#
# List Directors who are permitted to contact Storage daemon
#
Director {
    Name = maquinadirectora- dir
    Password = "clave"
}

#
# Devices supported by this Storage daemon
# To connect, the Director's bacula- dir.conf must have the
# same Name and MediaType.
#

Device {
    Name = DLT4- 0
    Media Type = DLT4
    Archive Device = /dev/nst0
    LabelMedia = Yes;          # lets Bacula label unlabeled media
    RandomAccess = yes;
    AutomaticMount = yes;      # when device opened, read it
    RemovableMedia = yes;
    AlwaysOpen = no;
}

Device {
    Name = DLT4- 1          #
    Media Type = DLT4
    Archive Device = /dev/nst1
    AutomaticMount = yes;    # when device opened, read it
    LabelMedia = Yes;
    AlwaysOpen = no;
    RemovableMedia = yes;
    RandomAccess = yes;
}

#
# Send all messages to the Director,
# mount messages also are sent to the email address
```

```
#
Messages {
    Name = Standard

    director = maquinadirectora- dir = all
}

```

Esto se va complicando un poquillo ;-P. En este fichero se definen varios recursos:

1. Storage

Definimos el nombre del demonio y el puerto de escucha

3. Director

Al igual que en los otros, el nombre del director que se nos va a conectar, con su password.

5. Devices

Definimos los dispositivos físicos de escritura. En este caso los de cinta. Sé que se pueden definir otro tipo de dispositivos, pero como es algo que no uso, no me he molestado en averiguar. Ya sabeis, RTFM. En este recurso vemos que le damos un nombre al dispositivo, indicamos el tipo de medio, y el nombre físico del cacharro tal y como se ve desde el sistema de ficheros. Lo demás son diferentes opciones, como que se monten automáticamente las cintas o que se permitan etiquetar. Hay un montón de ellas que conviene revisar en la documentación para escoger las que más nos convengan. En este ejemplo, yo defino 2 dispositivos, correspondientes a las dos bocas de mi unidad de cintas.

7. Messages

El tipo de mensajes que queramos recibir.

• /etc/bacula- dir

Ya comenté que este fichero es excesivamente largo, por lo que no voy a poner el mio entero, iré comentando sus secciones por orden y le vais añadiendo lo que necesiteis.

#Definición del director

```
Director {
    # define myself
    Name = maquinadirectora- dir
    DIRport = 9101 # where we listen for UA connections
    QueryFile = "/etc/bacula/scripts/query.sql"
    WorkingDirectory = "/var/lib/bacula"
    PidDirectory = "/var/run/bacula"
    Maximum Concurrent Jobs = 1
    Password = "clave" # Console p$ Messages = Standard
}

```

Definimos el nombre del director, su puerto de escucha y la clave. Además vemos una opción que permite fijar cuantos trabajos se realizarán simultáneamente. En la documentación se recomienda poner a 1. Vosotros vereis.

```
Job {
    Name = cliente1- diaria
    Client = cliente1- fd
    Type = backup
    Level = Incremental
    FileSet = Diaria
    Schedule = Diaria
    Messages = Standard
    Pool = Diaria

    Storage = tape1
}

```

```
    # JobDefs = "Diaria"
}
```

Comenzamos con lo realmente interesante. Esto de arriba es una definición completa de un trabajo. En donde definimos su nombre, el nombre del cliente, el tipo de trabajo, su nivel, los ficheros a los que se accede, el calendario, los mensajes que manda, el pool a utilizar y el dispositivo de escritura. La mayoría de estos campos los definiremos en este mismo fichero más adelante, no os preocupéis. Pero lo realmente lastimoso es que hay que escribir esta definición por cada tipo de trabajo y cliente. Os comenté que en versiones recientes se había simplificado mucho este problema al permitir definir un modelo de trabajo que luego se utilizaba en todas las definiciones. Si os daís cuenta, tengo ese campo comentado, porque tras escribir el fichero de configuración de esa manera, pude comprobar como la versión instalada en el servidor no permitía esas definiciones. Para los que dispongais de una versión más reciente con que hagais esto os valdría:

```
#Backup diario
JobDefs {
    Name = Diaria
    Type = Backup
    Level = Incremental
    FileSet = Diaria
    Schedule = Diaria
    Messages = Standard
    Pool = Diaria
    Storage = tape0
}
```

Haceis una sola definición por cada tipo de trabajo y despues:

```
Job {
    Client = dell1 - fd
    JobDefs = "Diaria"
}
```

Se simplifica ¿verdad? Cuando solo tienes un par de máquinas, no hay problema, pero cuando llegas a la treintena te puedes acordar de la madre del programador que no pensó en ello antes. Pasamos al siguiente recurso

```
FileSet {
    Name = "Diaria"
    Include = signature=MD5 {
        /var
        /etc
        /root
        /home
    }

    Exclude = { /proc /tmp /.journal /.fsck /var/run }
}
```

Fácil ¿no?. Le decimos que directorios o ficheros queremos que nos respalde, incluyendo una suma md5 y cuales queremos excluir. Y por supuesto le damos un nombre que lo identifique en los trabajos

```
Schedule {
    Name = "Diaria"
    Run = Incremental mon- sat at 20:00
}
```

Los calendarios de los trabajos y su tipo, aunque no es necesario indicarlo porque ya lo tengo definido en los trabajos. Se pueden hacer un montón de diabluras con las fechas y

las horas, que ejecute un trabajo el tercer jueves de cada mes, el último domingo de Mayo, etc. En el ejemplo realiza un trabajo de lunes a sábado a las 20:00 horas

```
Client {  
  
    name = cliente1-fd  
    Address = cliente1.x.com  
    FDPort = 9102  
    Catalog = MyCatalog  
    Password = "clave"  
    File Retention = 30 days      # 30 days  
    Job Retention = 6 months     # six months  
    AutoPrune = yes             # Prune expired Jobs/Files  
}
```

Pasamos a la definición de los clientes, se necesita una de estas por cada uno de ellos. Le ponemos un nombre, que será obligatoriamente el que ya pusimos en el demonio que se ejecuta en la máquina a respaldar, su dirección en la red, el puerto donde escucha, el catálogo (la base de datos), su clave, el tiempo que retendrá los ficheros salvados, el tiempo que quedará registrado el trabajo en la base de datos, y si queremos que se borren automáticamente los registros de la base de datos pasado el tiempo definido.

```
Storage {  
    Name = tape0  
    Address = maquinadirectora    # N.B. Use a fully qualified name here  
    SDPort = 9103  
    Password = "clave"  
    Device = DLT4-0  
    Media Type = DLT4  
}
```

En esta parte nombramos los recursos para conectar con el demonio de almacenamiento. Le damos un nombre, su dirección en la red, el puerto de escucha y su clave de acceso. Además le indicamos el dispositivo y tipo de medio.

```
Catalog {  
    Name = MyCatalog  
    User = bacula  
    dbname = bacula  
    password = "clave"  
}
```

El recurso relacionado con la base de datos que creamos previamente a la instalación de bacula. El nombre, el usuario permitido en la base, la base misma y el password.

```
Messages {  
    Name = Standard  
    mailcommand = "/usr/lib/bacula/smtp -h localhost -f \"\$(Bacula) %r\" -s  
    \"Ba$ operatorcommand = "/usr/lib/bacula/smtp -h localhost -f \"\$(Bacula)  
    %r\" -s $ mail = root@localhost = all, !skipped  
    operator = root@localhost = mount  
    console = all, !skipped, !saved  
    append = "/var/lib/bacula/log" = all, !skipped  
}
```

Todos los mensajes que genera el director y a donde mandarlos, se configuran aquí. Tal y como está puesto, genera bastante ruido (un mensaje por trabajo), pero a mi particularmente me gusta enterarme de todo.

```
Pool {  
    Name = Diaria  
    Pool Type = Backup
```

```

Recycle = yes           # Bacula can automatically recycle Volumes
AutoPrune = yes        # Prune expired volumes
Volume Retention = 100 days
Accept Any Volume = yes      # write on any volume in the pool

}

```

Definiciones de los pools. En mi caso tengo, un pool para las copias diarias, otro para las semanales y otro para las mensuales, cada uno con sus cintas diferentes.

Y esto es todo de momento, seguiremos proximately con el manejo de bacula desde la consola.

Bacula en ejecución

Entramos de lleno en el funcionamiento de Bacula, y los errores más frecuentes que podemos encontrar

Una vez que hemos configurado bacula, podemos comprobar que los ficheros estén correctamente con la opción -t de los tres demonios:

```
maquinadirectora:~#bacula-dir -t /etc/bacula-dir.conf
```

Si tenemos un error en la configuración nos lo dirá. Eso sí, solo nos dirá si el error es de sintaxis. Problemas de conexión a causa de la red, o errores en los passwords/nombres tendremos que averiguar nosotros la causa.

Arrancalo Carlos, por Dios

Nos vamos a la maquinadirectora y ejecutamos si no estaban ya corriendo:

```

maquinadirectora:~# /etc/init.d/mysql start
maquinadirectora:~# /etc/init.d/bacula-sd start
maquinadirectora:~# /etc/init.d/bacula-fd start
maquinadirectora:~# /etc/init.d/bacula-dir start

```

Arrancamos también cada uno de los clientes que vayamos a utilizar, y si todo ha ido bien, en ninguna de las máquinas habremos obtenido un error. En este momento lo único que puede haber sucedido es que tengamos algún error sintáctico en alguno de los ficheros, y el demonio se niegue a arrancar. Lo solucionamos y pasamos al siguiente paso, que es etiquetar las cintas que vayamos a usar y añadirlas a los pools que tengamos definidos. Aquí ya entraría en juego la estrategia de cada administrador a la hora de establecer una política de backups. Hay que tener en cuenta la importancia de los datos, la cantidad de cintas que tenemos etc, para organizarlo todo. En la documentación de Bacula hay una sección referente a ello (que siendo sinceros ni me he mirado) que se supone ayuda a establecer esa política. En mi caso he definido tres pools: Diaria con copias incrementales, Semanal con copias diferenciales, Mensual con copias completas, y un trabajo diario de verificación de sumas de los ficheros. Una vez tengamos esto claro y distribuidas las cintas por los diferentes pools hay que etiquetarlas físicamente (el cartoncito que llevan pegado a la cinta para distinguirlas nosotros) y por software para que Bacula las reconozca. Ejecutamos la consola de bacula:

```
gandalf:~# bconsole
```

Y obtenemos el prompt del director:

```

Connecting to Director maquinadirectora.x.com:9101
1000 OK: directora-dir Version: 1.32f- 5 (09 Mar 2004)
Enter a period to cancel a command.
*

```

Desde aquí tenemos pleno control del director y ahora es cuando las podemos etiquetar ejecutando el comando **label** Que nos preguntará por los diferentes dispositivos que tengamos configurados. En este caso, las dos bocas de la unidad de cintas

The defined Storage resources are:

1: Tape0

2: Tape1

Select Storage resource (1-2):

Enter new Volume name:

Le ponemos un nombre y si no ha sido ya definido previamente, nos preguntará por el nombre del pool al que queremos añadir la cinta. Una vez se lo facilitemos, bacula lo tendrá marcado para solicitarnoslo cada vez que lo necesite. Repetimos la operación por cada uno de los volúmenes a utilizar. Una vez terminado ejecutamos **list media** y nos deberían salir todos los pools definidos con sus respectivos volúmenes, bytes escritos, última escritura y un sinfín de datos más.

Ya que tenemos esto definido, podemos pasar ahora a comprobar el estado de los diferentes demonios. Si nos hemos conectado al director es evidente que está en marcha, pero no sabemos mucho de los clientes ni del demonio de almacenamiento. Para esto utilizamos las ordenes **status client** y **status storage**

*status client

The defined Client resources are:

1: cliente1- fd

Connecting to Client cliente1- fd at cliente1.x.com:9102

cliente1- fd Version: 1.32f- 5 (09 Mar 2004) i386- pc- linux- gnu debian testing/unstable
Daemon started 30- Apr- 04 11:51, 7 Jobs run.

Terminated Jobs:

| JobId | Level | Files | Bytes | Status | Finished | Name |
|-------|-------|--------|-------------|--------|-------------------|-------------------|
| 40 | Sinc | 4,463 | 92,982,884 | OK | 30- Apr- 04 12:03 | cliente1- diaria |
| 52 | Sinc | 532 | 3,741,110 | OK | 30- Apr- 04 20:03 | cliente1- diaria |
| 65 | Sinc | 3,778 | 75,600,731 | OK | 03- May- 04 20:04 | cliente1- diaria |
| 78 | Sinc | 2,139 | 59,748,148 | OK | 04- May- 04 20:04 | cliente1- diaria |
| 90 | Full | 18,689 | 175,881,481 | OK | 05- May- 04 13:56 | cliente1- semanal |
| 91 | Sinc | 2,109 | 6,092,577 | OK | 05- May- 04 20:03 | cliente1- diaria |
| 114 | Sinc | 2,347 | 6,960,272 | OK | 06- May- 04 20:03 | cliente1- diaria |

Director connected at: 07- May- 04 11:54

No jobs running.

Vemos como el cliente funciona y además nos informa de los trabajos realizados hasta la fecha, los ficheros copiados el tipo de trabajo realizado y si hay alguno en ejecución. Es el momento de hacer la comprobación para cada uno de los clientes.

El comando **status** admite además los parámetros *dir* y *all*.

Si queremos información de la situación de los pools, volúmenes, trabajos realizados etc, haremos uso del comando **list** por ejemplo:

*list nextvol job=cliente1- diaria

The next Volume to be used by Job "cliente1- diaria" will be Diaria1

Nos informa de cual será el volumen requerido para el proximo trabajo de cliente1- diaria

El comando **messages** nos mostrará los mensajes que tenga pendiente el director por comunicarnos aunque también los recibiremos por correo, si así lo tenemos configurado. Otra orden que no podía faltar es **help** la cual nos mostrara el juego completo de ordenes y su misión.

Ejecución de los trabajos.

Aunque la manera normal de funcionar de Bacula, es programar la ejecución de los trabajos mediante un calendario. Es posible ejecutarlos manualmente, para comprobar su correcto funcionamiento. La forma de hacerlo es con el comando **run**

run

A job name must be specified.
The defined Job resources are:
1: cliente1- diaria

Select Job resource (1): 1
Run Backup job
JobName: cliente1- diaria
FileSet: Diaria
Level: Incremental
Client: cliente1- fd
Storage: tape1
Pool: Diaria
When: 2004- 05- 07 12:33:28
Priority: 10
OK to run? (yes/mod/no):

En este momento se nos cuestiona si queremos pasar a la ejecución del trabajo, modificar los diferentes parámetros o su cancelación. Como ya digo es posible alterar los valores que tuvieramos por defecto para ese trabajo si lo hacemos manualmente. Quizá nos interesara copiar los datos en otra parte, o hacer una copia completa etc.

OK to run? (yes/mod/no): yes
Run command submitted.
*

*status client
The defined Client resources are:
1: cliente1- fd

cliente1- fd Version: 1.32f- 5 (09 Mar 2004) i386- pc- linux- gnu debian testing/unstable
Daemon started 30- Apr- 04 11:51, 7 Jobs run.

Director connected at: 07- May- 04 12:37
JobId 126 Job cliente1- diaria.2004- 05- 07_12.36.48 is running.
Backup Job started: 07- May- 04 12:36
Files=10 Bytes=115,580 Bytes/sec=3,611
Files Examined=1,346
Processing file: /var/cache/locate/locatedb
SDReadSeqNo=5 fd=7

Y comprobamos su ejecución mediante **status client**.

Recuperando ficheros

Para recuperar ficheros desde las cintas se usa el comando **restore**, aunque previamente deberíamos crear un trabajo que cumpla esa funcionalidad. Puesto que no podemos contemplar a priori que necesitamos restaurar es útil definir un trabajo por defecto y modificar despues cuando nos lo soliciten los valores necesarios, como el cliente, el fichero o directorio que necesitemos, etc. Despues Bacula nos preguntará a cerca del tipo de restauración que queremos llevar a cabo, como vereis la lista es extensa:

o select the JobIds, you have the following choices:
1: List last 20 Jobs run
2: List Jobs where a given File is saved
3: Enter list of JobIds to select
4: Enter SQL list command
5: Select the most recent backup for a client
6: Select backup for a client before a specified time
7: Enter a list of files to restore
8: Enter a list of files to restore before a specified time
9: Cancel

Select item: (1- 9):

Una vez hecha la selección, Bacula nos preguntará por el cliente y acto seguido por el juego de ficheros (FileSet) definido. En este momento Bacula realizará una consulta a la base de datos para intentar localizar los trabajos que le hemos pedido, y nos presenta el prompt **\$** a la espera de recibir los comandos necesarios para realizar el trabajo de restauración. Si escribimos **help** obtendremos una explicación de cuales son estos comando y su cometido.

Una vez que hemos marcado los ficheros o directorio a restaurar, escribimos **done** y Bacula nos pedirá los volúmenes donde estén situados esos ficheros. Solicitará confirmación para ejecutar el trabajo presentando las opciones del trabajo que va a llevar a cabo y una vez lo obtenga empezará con el mismo.

Y con esto doy por finalizado la serie. Aún quedan muchas cosas por contar, pero donde mejor las vais a comprender es leyendo la documentación que merece la pena. Solo he tratado de dar una visión global del funcionamiento y puesta en marcha de Bacula, además de que he ido aprendiendo según lo iba configurando, y todavía me queda bastante para tener bajo control todos los detalles. Si alguien tiene alguna duda, que no tenga reparo en escribirme y le ayudaré en la medida de mis posibilidades.

Comentarios:

cosas importantes (5.00 / 2) (#1)

por emeteo ([emeteo at escomposlinux.org](mailto:emeteo@escomposlinux.org)) a las Wed May 12th, 2004 at 01:11:33 PM CET
([Información Usuario](#)) <http://cernicalo.escomposlinux.org>

Si queremos que la señora de la limpieza que tiene correo electrónico nos haga el trabajo sucio, o sea, cambiar las cintas cuando sea necesario, hay que configurar específicamente el bacula- sd.conf:

```
AutomaitcMount = yes;
```

Y acordarse de expulsar la cinta cuando termine, esto se usa mediante el RunAfterJob dentro de la definición del último job de cada día, llamando a un script que primero 'rewind' y luego 'eject' la cinta.

Otra cosa que hay que entender son los períodos de retención, el autoprune y el recycle.

Hay dos tipos de períodos de retenciones, las que afectan a los trabajos almacenados en los volúmenes, y las que afectan al registro en la BBDD. Afinando el primero nos permite reciclar eficientemente las cintas.

Así,

si se tiene AutoPrune=yes, una vez transcurrido el período de retención configurado, el volumen será considerado para poder volver a reutilizarlo. El período de retención que afecta a la BBDD son los configurados en la definición del cliente, y sólo afecta a los registros en la BBDD, esto nos permite controlar el tamaño de esta.