Guía práctica de Bacula

Matias Banchoff
matiasb@cespi.unlp.edu.ar

19 de mayo de 2009

Resumen

Este documento es una guía práctica sobre Bacula. Muestra la arquitectura de Bacula, cómo agregar un host para que se backupee, como correr un backup desde la consola y como hacer restauraciones

Parte I

Introducción

NOTA: Documento modificado. BorrÃ© algunas cosas del "output" que produce bacula (ips, passwords, etc.).

1. Bacula

Bacula es una solución distribuída de backups. Esto significa que Bacula está compuesto por varios elementos, que pueden o no residir en el mismo host. Por ejemplo, se puede tener un host con el catálogo y en otro el storage[1]

2. Arquitectura de Bacula

Bacula se compone de:

Director Es el que controla el flujo de datos, dice cuando se debe hacer un backup, cuando se debe hacer un restore, le dice a los clientes que comiencen a empaquetar los archivos para su envío a los storages, etc. El drector arranca los backups siguiendo las pautas dadas en los *schedules*, pero también se pueden correr backups y otras operaciones *a mano*, mediante la bconsole[2] Una cosa a tener en cuenta es que los datos no pasan por el director, sino que transitan directamente entre FDs y storage.

Storage El storage se encarga de manejar los dispositivos físicos donde se guardarán los datos efectivamente. Un storage puede administrar varios dispositivos. Por ejemplo, cuando los backups se hacen a disco y a cinta, el storage es el que se encarga de recibir los datos desde los clientes y enviarlos al disco y a la cinta.

Catalog Es una base de datos donde se guarda información sobre los jobs y sobre los datos backupeados. El catalogo permite dos cosas:

- Por un lado, como guarda información de los jobs, pools y volúmenes, Bacula lo usa para saber si hay un backup full para un job, y si no lo hay, *eleva* el backup a full. También lo usa para saber si qué tiene que *prunear*.
- Por otro lado, el catálogo tiene todos los nombres de archivo (y sus atributos, como fecha de última modificación, etc.) que se backupearon, y eso es lo que permite hacer un restore selectivo, es decir, seleccionar (*marcar*, en la jerga de Bacula) individualmente qué archivos y/o directorios restaurar.

FileDaemon/Client Se puede ver al FileDaemon como un agente que corre del lado del cliente, es decir, en la máquina cuyos datos se van a backupear, y que tiene como objetivo *empaquetar* los datos y enviarlos al Storage, donde serán almacenados.

Bconsole La bconsole es un programa, que puede o no correr en el mismo host que el director, y que tiene como propósito interactuar con él. La interacción se hace por línea de comandos, aunque hay webapps y GUIs en desarrollo, como brestore o bat.

[1] El catálogo y el storage son componentes de Bacula que se explicarán más adelante

[2] Hay varias herramientas en desarrollo que servirían como ui, la más fácil de instalar es la bconsole

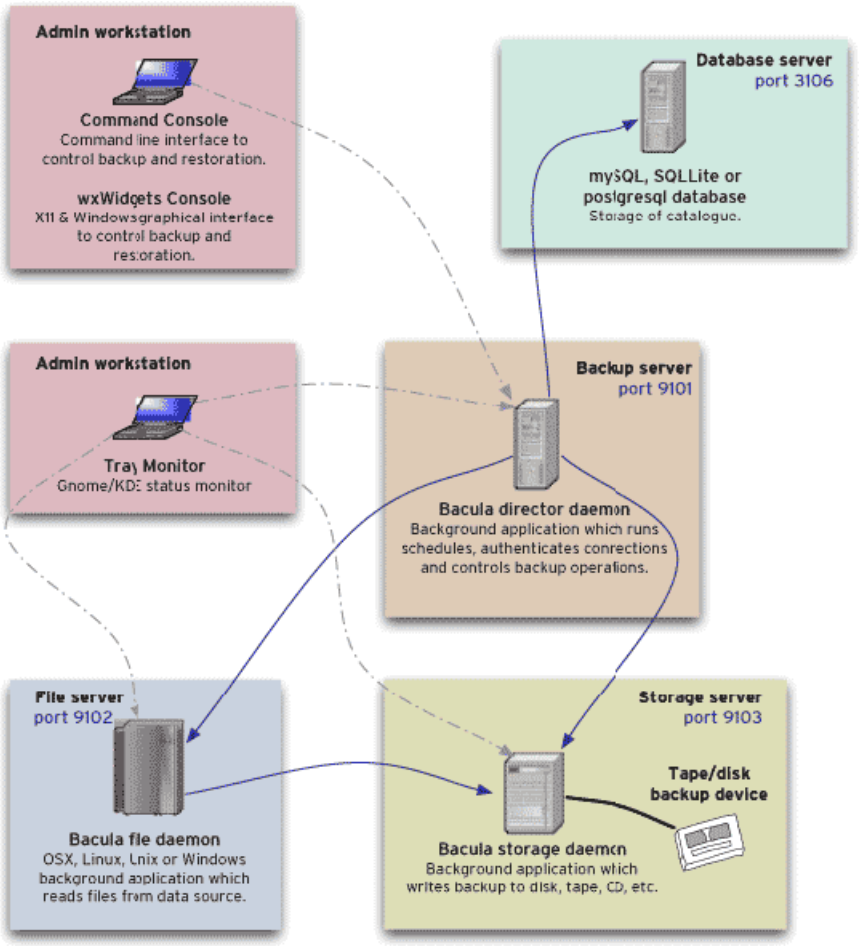

Figura 1: Tomado de la página de Bacula, esta es la arquitectura de Bacula

3. Elementos de configuración

La configuración de Bacula es sencilla e intuitiva. Hay que tener siempre presente que es el director el que se autentica contra los componentes, y no al revés.

3.1. Elementos del bacula-dir.conf

Este es el archivo de configuración del Bacula Director. En este archivo se define el director, los schedules[3], los pools, los jobs.

Los elementos que se deben definir en este archivo son:

Director Se especifica la *configuración* del director[4]. Entre las opciones de configuración, se especifica el par ip:port del director, el directorio donde se guarda el PID, la cantidad de jobs concurrentes.

Storage Acá se le dice al Director cuáles son los storage (ip:port), la password que esperan y el tipo de device

[3]Los schedules junto con los pools implementan la política de backup. Los otros elementos de la configuración sirven para poner en marcha Bacula

[4]No confundir con los elementos *director* en los archivos de configuración de los FDs y bconsole. En estos archivos este elemento indica qué directores se pueden conectar, pero no es la configuración del director

Catalog Tiene la información para conectarse a la base de datos del catalogo (username, password, tablename)

Messages Acá decimos cómo y cuándo se va a comunicar Bacula con el administrador de backups. Por ejemplo, para que envíe emails con el estado de los backups, que loguee a un archivo todos los jobs que salieron mal, que ejecute un script después de cada job, etc.

JobDefs Este elemento es un template para los jobs. Aca se especifican parámetros por defecto para jobs. Si un job hace referencia a un JobDefs y no especifica un parámetro, toma el que está definido en el jobDefs (en caso de que ahí esté definido)

Schedule Sirve para implementar la política de backups, junto con las opciones de recycle, max volume jobs, etc de los pools. Básicamente, decimos cuándo corremos un backup, dónde lo guardamos y qué archivos backupeamos

Pool Es una entidad lógica que sirve para agrupar volúmenes. Todos los volúmenes de un pool comparten las características de ese pool, como max volume jobs, autoprune, etc. Los pools complementan a los schedules, y entre los dos implementan la política de backup.

Job Acá se definen los jobs que se van a correr. Bacula corre los jobs automáticamente sólo si tiene un schedule asociado. Si no lo tiene, el job se puede correr manualmente. Hay tres clases de jobs: Backup, Restore y Verify (verifica que los atributos de los archivos en el filesystem sean iguales a los atributos guardados en el catalogo para esos archivos)

Client Acá le decimos al director cuáles son los hosts que se van a backupear. Se especifica su ip:port y la password con la que el director se tiene que autenticar, entre otras cosas.

FileSet Acá decimos los directorios y archivos que se van a backupear. También se pueden indicar algunas opciones, como que se encripten los datos con md5 o sha1, o que se compriman.

3.2. Elementos del bacula-sd.conf

En este archivo está la configuración del Bacula Storage. Aca se configura el storage y los devices, que son los medios donde se van a almacenar físicamente los datos backupeados (cintas y/o discos rígidos).

Storage Donde se define el storage daemon

Director Donde se especifica los directores que pueden contactar este storage

Device Se especifica el dispositivo donde ser guardarán los datos (un archivo en /dev o un directorio en el filesystem)

Messages

3.3. Elementos del bacula-fd.conf

En este archivo se configura el cliente. Hay un archivo por cada cliente. Este archivo reside en el cliente, no en el director.

3.4. Elementos del bconsole.conf

Este archivo configura la bconsole. Pueden existir varias bconsoles en distintos hosts, cada una con su archivo de configuración.

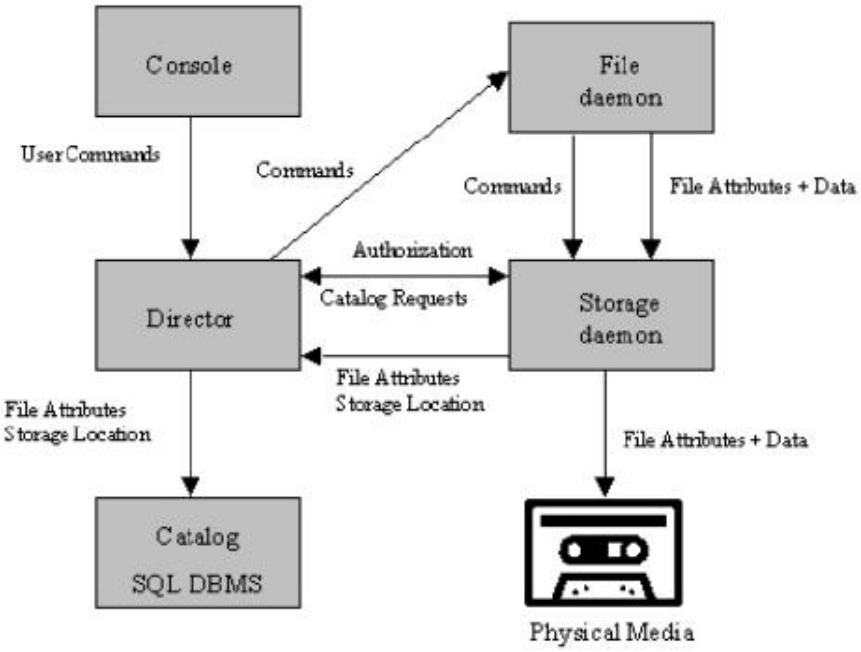

Figura 2: Tomado de la página de Bacula, esta imagen muestra el flujo de datos entre los componentes que forman Bacula

Parte II

Configuración

Ahora vamos a ver una configuración en particular de Bacula. En este ejemplo el Director, Storage y Catalog corren en la misma máquina, mientras que la consola corre en una máquina distinta, y en una tercera máquina corre el FD.

4. Director

El director se configura en el archivo *bacula-dir.conf*. En este archivo se definen los siguientes recursos:

Director Acá se define el director (ip, puerto, directorios que usa, etc.)

Job Aca se definen los jobs que se pueden correr (Backup, Restore, Verify)

JobDefs Estos son plantillas para jobs, no se ejecutan, sino que cuando a un job le falta un parámetro, lo toma del jobdef declarado en ese job

Schedule Este recurso permite especificar cuando correr un job y sobre qué pool hacer el backup

FileSet Acá se especifica qué archivos y directorios se van a guardar

Client Acá se definen todos los clientes que se van a backupear (Toda la info necesarias para contactarlos)

Storage En este resource se ingresan los datos de los storages (La info necesaria para contactarlos)

Pool Los pools son entidades abstractas que agrupan volúmenes. Los volúmenes heredan los parámetros del pool en que se encuentra.

Catalog Define el catálogo que se va a usar

Messages Definimos los recursos *mensaje*, que es donde se define la forma en que Bacula se va a comunicar con el administrador de backups. Si va a enviar emails, si va a loguear a archivos, etc.; todo eso se define acá.

A continuación, las opciones más importantes cuando se configura el director. Para más información, ver la documentación de Bacula.

```
Director {
  Name = bacula-dir
  Description = "Bacula director"
  DIRport = 9101
  DIRAddress = mibacula.director.net
  QueryFile = "/etc/bacula/scripts/query.sql"
  WorkingDirectory = "/var/lib/bacula/"
  PidDirectory = "/var/run/bacula/"
  Maximum Concurrent Jobs = 1
  Password = "la password"
    # Matchea con la password del recurso director en la bconsole
  Messages = Standard
}
Storage {
  Name = bacula-disco
  Address = 192.168.2.2             # No usar "localhost"
  SDPort = 9103
  Password = "la password"
    # Matchea con la password del recurso storage en bacula-sd.conf
  Device =  FileStorage
  Media Type = File
}
Catalog {
  Name = catalogo
  dbname = bacula
  DB Address = localhost
  user = bacula
  password = ""
}
Messages {
  Name = Standard
  mailcommand = "/usr/lib/bacula/bsmtp
        -h localhost
        -s \"Bacula: %t %e %n %i %c %l\"
        operador@[192.168.5.5]"
  operatorcommand = "/usr/lib/bacula/bsmtp
        -h localhost
        -s \"ADVERTENCIA OPERADOR: %t %e %i %c %l %j %r job: %n\"
        operador@[192.168.5.5]"
  Mail = operador@[192.168.5.5] = all, !skipped, !terminate
```

```
  append = "/var/lib/bacula/info.log" = info, terminate
  append = "/var/lib/bacula/error_warning.log" = error, warning, !terminate, notsaved
  append = "/var/lib/bacula/all.log" = all
  operator = operador@[192.168.5.5] = mount
  console = error, warning, !terminate
}
JobDefs{
  Name = "default_backup_job"
  Type = Backup
  Storage =  bacula-disco
  Messages = Standard
  Priority = 10
        # Es la opcion por defecto, a mayor numero, menor la prioridada
  Reschedule on error = yes
  Reschedule interval = 1 hour
  Reschedule times = 1
  Max Start Delay = 8 hours
}
Client {
  Name = uncliente-fd
  Address = micliente.dominio.net
  FDPort = 9102
  Catalog = catalogo
  Password = "la password"
  File Retention = 30 days
  Job Retention = 2 months
}
FileSet {
  Name = "uncliente-fs"
  Include {
    Options {
      signature = MD5
      compression=GZIP9
    }
   File = /home/
   File = /var/www
   File = /var/named/
   File = /etc
  }
}
Schedule {
  Name = "uncliente_schedule"
  Run =
        Level = Full
        Pool = uncliente-full
        storage = bacula-disco
        1st sun at 0:30
  Run =
        Level = Differential
        Pool = uncliente-diff
        storage = bacula-disco
```

```
        2nd-5th sun at 0:30
  Run =
        Level = Incremental
        Pool = uncliente-inc
        storage = bacula-disco
        mon-sat at 0:30
}
Pool {
  Name = uncliente-full
  Pool Type = Backup
  Recycle = yes
  AutoPrune = yes
  Volume Retention = 7 weeks
  Accept Any Volume = yes
  Recycle Oldest Volume = yes
  Maximum Volume Jobs = 1
  Label Format = Uncliente-Full-
  Maximum Volumes = 2
}
Pool {
  Name = uncliente-inc
  Pool Type = Backup
  Recycle = yes
  AutoPrune = yes
  Volume Retention = 6 days
  Accept Any Volume = yes
  Recycle Oldest Volume = yes
  Maximum Volume Jobs = 1
  Label Format = Uncliente-Inc-
  Maximum Volumes = 6
}
Pool {
  Name = uncliente-diff
  Pool Type = Backup
  Recycle = yes
  AutoPrune = yes
  Volume Retention = 4 weeks
  Accept Any Volume = yes
  Recycle Oldest Volume = yes
  Maximum Volume Jobs = 1
  Label Format = Uncliente-Diff-
  Maximum Volumes = 4
}
Job {
  Name = "backup_uncliente"
  JobDefs = "default_backup_job"
  Client = uncliente-fd
  Schedule = "uncliente_schedule"
  FileSet = uncliente-fs
  Write Bootstrap = "/var/lib/bacula/uncliente.bsr"
  Pool = uncliente-inc
```

```
        # Dumy pool, porque se sobreescribe en el schedule
}
```

5. Storage

Los recursos que se configuran en un storage están en el archivo bacula-sd.conf, en la máquina donde corre el storage, y son:

Storage Define las propiedades del demonio storage (ip, puerto, pid, etc.)

Director Define los directores que pueden contactar este storage

Device Define los dispositivos que maneja

Messages Define cual de los mensajes definidos en bacula-dir.conf va a usar

Configuración típica de un storage con un director y dos dispositivos:

```
Storage {
  Name = mistorage-sd
  SDPort = 9103
  SDAddress = 192.168.0.2
  WorkingDirectory = "/var/lib/bacula"
  Pid Directory = "/var/run/bacula"
  Maximum Concurrent Jobs = 20
}
Director {
  name = bacula-dir
  password = "la password"
    # matchea con la password en el recurso storage de bacula-dir.conf
}
#Dispositivo archivo
Device {
  Name = FileStorage
  Media Type = File
  Archive Device = /var/backups/bacula
  LabelMedia = yes;
  Random Access = Yes;
  AutomaticMount = yes;
  RemovableMedia = no;
  AlwaysOpen = no;
}
#Dispositivo cinta
Device {
  Name = TapeStorage
  Media Type = "Super DLT I"
  Archive Device = /dev/nst0
  AutomaticMount = yes;
  AlwaysOpen = yes;
  RemovableMedia = yes;
  RandomAccess = no;
```

```
  LabelMedia = yes;
}
Messages {
  Name = Standard
  director = bacula-dir = all
}
```

6. Catalog

El catálogo no tiene un archivo de configuración propio, se configura en el recurso Catalog del bacula-dir.conf

7. Cliente (FD)

El archivo bacula-fd.conf está en cada cliente Bacula. Los recursos que se tienen que configurar son:

FileDaemon Define el cliente actual

Director Dice qué directores se van a conectar a este cliente

Messages Define cual de los mensajes definidos en el director se va a usar

```
Director {
  Name = bacula-dir
  Password = "la password"
}
FileDaemon {
  Name = uncliente-fd
  FDport = 9102
  WorkingDirectory = /var/lib/bacula
  Pid Directory = /var/run/bacula
  Maximum Concurrent Jobs = 20
}
Messages {
  Name = Standard
  director = uncliente-dir = all, !skipped
}
```

8. bConsole

El archivo de configuración de la bConsole es bconsole.conf, con las siguientes opciones de configuración para el director:

Name El nombre del director

DirPort Puerto en el que escucha el Director al que no conectaremos

Address Dirección del director

Password Password usada, debe matcher con el atributo password del recurso director en el bacula-dir.conf

```
Director {
  Name = bacula-dir
  DIRport = 9101
  address = midirector.undominio.net
  Password = "lapassword!!"
}
```

Parte III

Backupear

Hay dos formas de correr un backup:

1. Automáticamente. Para que Bacula corra jobs automáticamente, ese job debe tener asociado un schedule (y el job tiene que estar habilitado, que lo está por defecto).
2. A mano. Es posible armar un job a mano y correrlo, así como también es posible correr un job tal cual está, sin modificar ninguno de sus parámetros.

Generalmente, los backups se corren automáticamente, mientras que los restores se corren "bajo demanda". A continuación vamos a ver cómo armar un job de backup y correrlo, todo desde la bConsole.

Correr un backup job a mano implica lo siguiente:

1. Entrar a la bConsole y elegir un backup job
2. Configurar ese job de acuerdo a nuestras necesidades
3. Ejecutar el job.

Antes de comenzar, algunos tips sobre la bConsole:

- En cualquier momento se puede cancelar lo que se está haciendo ingresando un punto ("."). Por ejemplo, si se está configruando un job para correr y queremos cancelar la configuración, ingresamos un ".".
- La consola acepta comandos incompletos. Por ejemplo, en vez de "messages", "mess"; en vez de "status storage", "stat sto".
- Se pueden escribir scripts que usen la bConsole. Personalmente no lo probé.
- No confiarse. Es muy fácil borrar algo del catálogo si no se tiene cuidado, haciendo un prune.
- El comando "help" lista los comandos disponibles en la bConsola, junto con una descripción

Ahora sí, el ejemplo.

Iniciamos la consola (Dependiendo de la configuración del host, quizás necesite privilegios de root)

```
$ bconsole
```

Una vez dentro de la bConsola podemos hacer un "status dir" para ver el estado del Director

```
*stat dir
silicio-dir Version: 1.36.3 (22 April 2005) i486-pc-linux-gnu debian testing/unstable
Daemon started 06-Nov-07 13:21, 36 Jobs run since started.

Scheduled Jobs:
Level          Type     Pri  Scheduled          Name               Volume
```

```
================================================================================
Incremental    Backup    10  09-Nov-07 00:30    backup_unlp        Unlp-Inc-0003
Incremental    Backup    20  09-Nov-07 01:05    backup_testing_tape *unknown*
====

Running Jobs:
No Jobs running.
====

Terminated Jobs:
 JobId  Level     Files      Bytes     Status   Finished        Name
====================================================================
  5116  Incr         69      7,936,572 OK       08-Nov-07 01:07 backup_unlp_tape
  5121  Incr        103     15,875,686 OK       08-Nov-07 01:32 backup_testing_tape
====
*
```

Arriba se puede apreciar, entre otras cosas, los jobs que se corrieron, el estado en que salieron y los próximos jobs a ejecutarse.

Podemos hacer un "stat sto" para verificar el estado del Storage:

```
*stat sto
The defined Storage resources are:
     1: silicio-sd-disco
     2: silicio-sd-cinta
Select Storage resource (1-2):
```

El sistema tiene dos Storages, uno para hacer backups en cinta y otro, en disco. Ingresando 1, se tiene:

```
Select Storage resource (1-2): 1
Connecting to Storage daemon silicio-sd-disco at una-ip:9103

silicio-sd Version: 1.36.3 (22 April 2005) i486-pc-linux-gnu debian testing/unstable
Daemon started 06-Nov-07 13:21, 33 Jobs run since started.

Running Jobs:
No Jobs running.
====

Terminated Jobs:
 JobId  Level   Files          Bytes Status   Finished        Name
====================================================================
  5116  Incr         69      7,944,619 OK       08-Nov-07 01:07 backup_unlp_tape
  5121  Incr        103     15,891,720 OK       08-Nov-07 01:32 backup_testing_tape
====

Device status:
Archive "FileStorage" is not open or does not exist.
Device "/dev/nst0" is mounted with Volume "Full-Backups-Tape-0008"
    Total Bytes=8,326,839,143 Blocks=129,093 Bytes/block=64,502
    Positioned at File=37 Block=0
*
```

Esto muestra el estado del servidor Storage, y el estado para cada device que administra. La parte más útil de este comando es donde muestra la información sobre la cinta. Ahí se puede ver la etiqueta con la que se montó la cinta, la cantidad de bytes escritos y si el dispositivo está o no bloqueado (BLOCKED).

El dispositivo de cinta se puede bloquear por varias razones:

1. La cinta no está en el dispositivo
2. Cada tanto se debe limpiar el dispositivo de cintas. Para esto se usa una cinta limpiadora, esecpecial para ese modelo de dispositivo. Si el dispositivo necesita una limpieza no va a aceptar cintas hasta que se lo limpie. Es un caso muy raro, pero ocurre...
3. La cinta no está montada. En este caso la cinta está en el dispositivo, pero no está montada en Bacula[5]
4. La cinta no está etiquetada[6]

Figura 3: Dispositivo de cinta

Ahora sí, para correr un backup job, ejecutar "run". Este comando nos lista los jobs que podemos correr:

```
*run
A job name must be specified.
The defined Job resources are:
     1: default_restore_job
     2: backup_unlp
Select Job resource (1-5):
```

Figura 4: Listado de jobs configurados en Bacula. Notar el primero, que es un restore

Notar el restore job listado en la primera posición. Esto es porque el comando "run" también se usa para ejecutar restore jobs. Pero en este caso vamos a elegir un backup job.

Vamos a hacer un full backup de unlp a disco. Comenzamos eligiendo respondiendo con 2.

```
Select Job resource (1-18): 2
Run Backup job
JobName:  backup_unlp
FileSet:  unlp-fs
Level:    Incremental
```

[5]En Linux los dispositivos de cinta (/dev/nst0, por ejemplo) no se montan. Sin embargo, en Bacula sí existe el concepto de montar la cinta, y para que Bacula pueda usar una cinta primero hay que montarla. Para montar/desmontar un dispositivo en Bacula, se usa el comando mount/umount de la bConsole

[6]Para etiquetar una cinta se usa el comando *label*. Las cintas se pueden etiquetar bajo demanda (a medida que las pide Bacula) o se pueden etiquetar todas y de una sola vez. La ventaja de lo segundo es que luego Bacula sabe qué cinta pedir

```
Client:   unlp-fd
Storage:  silicio-sd-disco
Pool:     unlp-inc
When:     2007-11-08 17:38:24
Priority: 10
OK to run? (yes/mod/no):
```

Al elegir un job, el 2 en este caso (que corresponde al job backup_unlp), Bacula nos da la configruación que tiene para ese job[7] y nos pregunta si el job está o no listo para ejecutarse. En el caso de que lo esté, ingresando "yes" comienza la ejecución del job, o lo pone en la cola del scheduler si es que no ha llegado la hora de ejecutarse todavia (campo "when").

El significado de cada campo es el siguiente:

JobName Es el nombre del job que se va a ejecutar. Este parámetro no se puede modificar

FileSet Es un fileset definido en la configuración de Bacula. Permite elegir el fileset a usar en este caso. Generalmente se tiene un fileset por cliente, aunque si se tienen muchos clientes con la misma estructura de directorios, se puede reusar ese fileset.

Level Dice el nivel de backup con que se va a ejecutar este job. Los niveles son Incremental, Diferencial y Full

client Especifica el cliente (fd) que se va a backupear. O sea, de este cliente se van a backupear los archivos y directorios especificados por el fileset elegido en la opción fileset.

Storage Es el storage (el dispositivo, no el servidor) donde se van a guardar los datos backupeados

Pool Es el pool de donde se va a usar el volumen donde se guardarán los datos.

When Establece la fecha y hora de inicio del backup. Por defecto, es "ahora".

Priority Establece la prioridad del job. Cuanto más grande sea el número, menor la prioridad

OK tu run? Nos pregunta si los valores para las opciones son correctos. En caso de serlos, escribiendo "yes" ejecuta el job (o lo pone en la cola del schedule, según corresponda). En caso de haber algún error o de querer modificar algo, escribir "mod". Si no se quiere ejecutar el job, escribir "no" (o escribir ".").

Vamos a correr un backup full, por lo que se debe modificar la opción de Level. Esto se hace escribiendo "mod" en el prompt y apretando ENTER.

```
OK to run? (yes/mod/no): mod
Parameters to modify:
     1: Level
     2: Storage
     3: Job
     4: FileSet
     5: Client
     6: When
     7: Priority
     8: Pool
Select parameter to modify (1-8):
```

Cuando escribimos "mod" Bacula nos pregunta qué queremos modificar. Vamos a modificar el nivel, así que escribimos 1.

```
Select parameter to modify (1-8): 1
Levels:
     1: Base
     2: Full
     3: Incremental
```

[7] La configuración inicial, esta que se ve en el ejemplo, la saca del archivo donde aparece definido ese job.

```
     4: Differential
     5: Since
Select level (1-5):
```

Todos estos son los niveles implementados (en realidad sólo están implementados los niveles Full, Incremental y Differential, los otros lo estarán en el futuro). Seleccionamos 2.

```
Select level (1-5): 2
Run Backup job
JobName:  backup_unlp
FileSet:  unlp-fs
Level:    Full
Client:   unlp-fd
Storage:  silicio-sd-disco
Pool:     unlp-inc
When:     2007-11-08 18:08:31
Priority: 10
OK to run? (yes/mod/no):
```

Bacula cambia el nivel del job a Full y nos vuelve a preguntar si el job está listo para su ejecución. Como queremos hacer el backup en la cinta, debemos modificar el Storage, porque está configurado para hacerlo en el disco. Nuevamente escribimos "mod", y en el listado de opciones a configurar, seleccionamos 2.

Ahora Bacula lista todos los dispositivos donde se puede grabar el backup. Notar que no son los servidores de Storage, sino los devices. En este ejemplo se tiene un Storage server y dos dispositivos, es por eso que en el listado aparecen dos entradas, una para cada dispositivo:

```
The defined Storage resources are:
     1: silicio-sd-disco
     2: silicio-sd-cinta
Select Storage resource (1-2):
```

Seleccionamos 2, que corresponde al dispositivo de cinta. Ahora Bacula muestra el cambio hecho al job, y nos vuelve a preguntar si el job ya está configurado para su ejecución:

```
Select Storage resource (1-2): 2
Run Backup job
JobName:  backup_unlp
FileSet:  unlp-fs
Level:    Full
Client:   unlp-fd
Storage:  silicio-sd-cinta
Pool:     unlp-inc
When:     2007-11-08 18:08:31
Priority: 10
OK to run? (yes/mod/no):
```

Queda por hacer un último cambio: En nuestro sistema todos los backups en cinta se hacen en un solo pool, que no es el que está configurado en ese job. Entonces hay que modificar la configuración para cambiar el pool. Escribimos "mod" y en el listado de opciones, elegimos 8, que es la opción que corresponde al atributo Pool:

```
Select parameter to modify (1-8): 8
The defined Pool resources are:
     1: full-backups-tape
     2: unlp-full
     3: unlp-inc
     4: unlp-diff
Select Pool resource (1-7):
```

En el listado aparecen los pools del sistema. Los backups en cinta usan volúmenes que están en el primer pool, así que elegimos 1, quedando:

```
Select Pool resource (1-7): 1
Run Backup job
JobName:  backup_unlp
FileSet:  unlp-fs
Level:    Full
Client:   unlp-fd
Storage:  silicio-sd-cinta
Pool:     full-backups-tape
When:     2007-11-08 18:08:31
Priority: 10
OK to run? (yes/mod/no):
```

Ahora sí está configurado el job para correr. Escribimos "yes". En este momento Bacula ejecuta el job.

En el caso de que hayamos modificado el atributo When y hayamos puesto "When: 2007-11-08 19:00:00", Bacula esperaría una hora antes de ejecutar el job.

Si hacemos un "status dir" en este momento veremos lo siguiente:

```
...... Mas informacion arriba

Running Jobs:
 JobId Level   Name                       Status
======================================================================
  5124 Full    backup_unlp.2007-11-08_18.23.04 is running
====

...... Mas informacion debajo
```

Esto indica que el job que acabamos de configurar se está ejecutando.

Si queremos cancelar el backup, podemos usar el comando "cancel". Si hay un solo backup corriendo, nos pregunta si lo queremos cancelar. Si hay varios backups corriendo, nos pregunta cuál de todos cancelar.

Parte IV

Restaurar

Ahora vamos a ver cómo ejecutar un backup job. La restauración se puede hacer ejecutando un Restore job que se haya definido en la configuración de Bacula (recordar el listado de jobs en FIGURA!!!) o ejecutando el comando *restore* en la bConsole. Las dos modalidades presentan diferentes opciones, así que primero se explicará cómo restaurar con un Restore job y después corriendo el comando *restore*.

La restauración de archivos en Bacula tiene tres etapas definidas:

1. Buscar y seleccionar los jobs que backupearon los archivos que se queremos recuperar. Esto se hace ejecutando un *run* o un *restore* en la bconsole.
2. Armar el árbol de directorios de donde seleccionar (marcar) los archivos y directorios a restaurar. Esto lo hace Bacula, en base a los jobs seleccionados en el paso anterior
3. Por último, debemos seleccionar qué restaurar. Esto lo hacemos a mano, utilizando el comando *mark*

En la bConsole escribimos *run*, y seleccionamos el restore job definido en 1.

```
The defined Job resources are:
     1: default_restore_job
     2: backup_unlp
     3: backup_unlp_tape
Select Job resource (1-5): 1
Please enter a JobId for restore:
```

Nos pregunta el Backup job que queremos restaurar

```
Please enter a JobId for restore: 4
Run Restore job
JobName:    default_restore_job
Bootstrap:  *None*
Where:      /var/cache/raid/backups/restores
Replace:    always
Client:     silicio-fd
Storage:    silicio-sd-disco
JobId:      4
When:       2007-11-09 18:38:29
Catalog:    catalogo
Priority:   10
OK to run? (yes/mod/no):
```

Cuando seleccionamos el job, Bacula arma un restore job para restaurar los datos guardados en en backup job seleccionado. Luego muestra configuración del restore job y nos pregunta si los datos son correctos. La forma de trabajar en este paso es la misma en que se trabajaba con los backup jobs: se verifican que los valores sean correctos, si no lo son, se ingresa "mod" y se escriben los valores correctos, así hasta que todos los valores sean los indicados. Cuando se haya terminado de configrurar el job, se lo ejecuta escribiendo "run" en la bConsole.

Bootstrap

Where Es el directorio en el cliente donde se van a guardar los archivos restaurados

Replace Dice si reemplaza los archivos existentes por los restaurados. Posibles valores: always, ifnewer, ifolder, never

Client Es el host que recibirá los archivos restaurados. Este host va a guardar los archivos en el directorio especificado en *Where*

Storage Es el dispositivo del que se van a sacar archivos para restaurarlos en el cliente

JobId Es el job que se va a restaurar

La otra forma de restaurar un backup es hacerlo mediante el comando *restore*. A continuación vamos a hacer un Full Restore de Isis.

Escribimos *restore* en la bConsole, a lo cual Bacula nos responde con el siguiente listado de opciones:

```
*restore

First you select one or more JobIds that contain files
to be restored. You will be presented several methods
of specifying the JobIds. Then you will be allowed to
select which files from those JobIds are to be restored.

To 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 comma separated 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):
```

Las opciones para restaurar archivos específicos requieren el path completo del archivo, tal cual estaba en el cliente. Por ejemplo, para restaurar el archivo */home/matiasb/archivo.txt* hay que buscarlo como */home/matias/archivo.txt*; *archivo.txt* sería un archivo totalmente distinto.

Esta posibilidad de restaurar archivos específicos se puede hacer gracias al catálogo que usa Bacula. Si por algún motivo el catálogo se borra, ya sea a propósito o por un descuido, no se podrán hacer restores selectivos para las entradas en el catálog que fueron borradas. Por ejemplo, si borro del catálogo todos los jobs relacionados con los backups hechos para Isis, entonces no podría hacer un restore de Isis de esta forma. De ocurrir esto, se puede hacer un restore usando las herramientas extra-consola que provee Bacula, como bls, bextract, etc.[8].

Notar que en el catálogo sólo se guarda información sobre los archivos y no los archivos en sí, así que por más que se pierda el catálogo, los datos seguirán almacenados en los volúmenes, ya sea en el disco duro o en la cinta, dependiendo del dispositivo donde se hayan almacenado.

Ingresando 5, Bacula nos pregunta cuál de todos los clientes restaurar. En nuestro caso, los clientes son:

```
Select item:  (1-9): 5
Defined Clients:

     4: isis-fd
Select the Client (1-4):
```

Elijo Isis. Inmediatamente Bacula lee el catálogo para armar un árbol de directorios virtual de donde seleccionar los archivos y directorios a restaurar:

```
Automatically selected FileSet: isis-fs
+-------+-------+-----------+---------------------+-----------------------+-----------+-------------+----------------+
| JobId | Level | JobFiles  | StartTime           | VolumeName            | StartFile | VolSessionId | VolSessionTime |
+-------+-------+-----------+---------------------+-----------------------+-----------+-------------+----------------+
| 5,038 | F     | 2,766,128 | 2007-11-04 00:41:53 | Isis-Full-0002        |         9 |          291 |  1,192,533,342 |
```

[8] Ver el apéndice para más información

```
...................... Muchas lineas mas.....
+-------+-------+----------+--------------------+-----------------------+----------+-------------+----------------+
You have selected the following JobIds: 5038,5060,5068,5076,5084,5093,5101,5109,5117,5126,5134

Building directory tree for JobId 5038 ...
```

En el listado de arriba se ve como Bacula elige los jobs y arma el árbol. Cuando termina y el árbol está creado, nos da un prompt donde podemos marcar y desmarcar archivos y directorios.

```
$ pwd
cwd is: /etc/bacula/
$ ls
bacula-fd.conf
scripts/
$
```

Con el comando *mark* se selecciona lo que queremos restaurar y con el comando *unmark* desmarcamos algo que por error marcamos. Por ejemplo, marco el archivo de configuración del cliente Bacula y luego lo desmarco:

```
$ mark bacula-fd.conf
1 files marked.
$ unmark bacula-fd.conf
1 files unmarked.
$
```

Supongamos que marcamos para restaurar el archivo */etc/bacula/bacula-fd.conf* y el directorio */etc/bacula/scripts*.

Con el comando *estimate* Bacula nos da una estimación de cuántos bytes se van a restaurar (Notar cómo marcando un directorio, automáticamente se marcan todos los archivos que tenemos dentro de ese directorio):

```
$ mark scripts
5 files marked.
$ estimate
2785207 total files; 8 marked to be restored; 2,576 bytes.
$
```

Cuando hayamos marcado todo lo que debemos restaurar, con el comando *done* le indicamos a Bacula que puede empezar a restaurar los datos. En cualquier momento podemos usar el comando *help*, que imprime un listado de comandos disponibles. Además, si se quiere cancelar el restore, se puede usar el comando ".", que sirve para volver a la bConsole.

Parte V
Herramientas de línea de comandos

9. Comando *mt*

La cinta se maneja con el comando *mt*. Este comando recibe como argumento el archivo de dispositivo asociado a la cinta. Estos archivos son de */dev/st0* y */dev/nst0*. El primero rebobina la cinta al inicio después de cada operación, mientras que el segundo no rebobina la cinta.

10. Obtener más información

Se puede usar el FileSystem */proc* para obtener información sobre los dispositivos SCSI conetcados al equipo.

```
soporte@silicio:~$ cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 06 Lun: 00
  Vendor: QUANTUM  Model: DLT VS160        Rev: 2C00
  Type:   Sequential-Access                ANSI SCSI revision: 02
Host: scsi2 Channel: 00 Id: 00 Lun: 00
  Vendor: ADAPTEC  Model: Device 1         Rev: V1.0
  Type:   Direct-Access                    ANSI SCSI revision: 02
soporte@silicio:~$
```

11. Herramientas de Bacula

Bacula tiene varias herramientas de linea de comandos que sirven para manipular los volúmenes. Por ejemplo, *bls* sirve para listar el contenido de un volumen, como si se hiciera un *ls -l*. Otras herramientas que provee son:

bls Lista el contenido de un volumen, en disco o tape

bscan Se usa para recrear el catálogo a partir de los volúmenes que se tienen en el disco o de la cinta

bextract Extraer archivos directamente desde el volumen en el dispositivo, sin usar la bConsola

bcopy Copiar un volumen a otro

bsmtp Se usa para enviar emails a los operadores, más flexible que las herramientas de email del sistema.

Las primeras cuatro herramientas se usan desde la linea de comandos, mientras que bsmt se usa generalmente desde el resource *Messages* en la configuraciÃ³n*deBacula*.

Parte VI

Apéndices

Comandos de la bConsole Los comandos de la bConsole se dividen en dos: Por un lado están los comandos que se usa en la bConsole para manipular los jobs, los pools y los volúmenes; y por otro, los comandos que se usan cuando se seleccionan archivos para restaurar. Estos últimos comandos están accesibles solamente cuando se restaura y crea el árbol de restauración.

Referencias

Referencias

[1] Bacula Manual: restore command. *http://www.bacula.org/rel-manual/Bacula_Consol_Restor_Comman.html*

[2] Correr restore desde la linea de comandos. Shell scripts *http://www.bacula.org/rel-manual/Bacula_Consol_Restor_Comman.html#SECTION00021400000000000000*