

# Guía de comandos para Bacula

Matias D. Banchoff T. - matiasb@cespi.unlp.edu.ar

30 de junio de 2010

## Resumen

*Cheat sheet* con los comandos más usados de la consola de Bacula. La lista completa se encuentra en la página de Bacula[1].



Licencia Atribución-No Comercial-Compartir Obras Derivadas Igual 2.5 Argentina

## 1. Comandos usuales de Bacula

### **autodisplay on/off**

Sirve para configurar que la consola muestre los mensajes automáticamente (*on*) o que nos advierta que hay mensajes encolados para leer (*off*). En el segundo caso, deberemos usar el comando *messages* para ver los mensajes encolados.

### **automount on/off**

Configura el automontaje de un volumen luego de ser etiquetado con el comando *label*. Cuando está desactivado, se debe montar el volumen explícitamente.

### **cancel jobid=number job=job-name ujobid=unique-jobid**

Usado para cancelar jobs. Ejecutado sin argumentos, la consola muestra un listado de jobs activos (ejecutando y preparados para correr) de donde el usuario selecciona cuál cancelar<sup>1</sup>.

### **delete volume=vol-name pool=pool-name job jobid=id**

Borra del catálogo un volumen, un pool o un job, así como los datos asociados creados durante la ejecución de los jobs. Opera sobre el catálogo solamente, por lo que no tiene efecto sobre los datos escritos en el disco rígido (o la cinta, según el caso).

```
delete pool=<pool-name>
```

Borra un pool del catálogo.

```
delete volume=<volume-name> pool=<pool-name>
```

Borra del catálogo un volumen del pool especificado.

```
delete JobId=<job-id> JobId=<job-id2> ...
```

Borra del catálogo los jobs especificados. Y, por último, un ejemplo que muestra que el comando acepta listas y rangos de JobIDs:

```
delete Job JobId=n,m,o-r,t ...
```

### **disable job=job-name**

Sirve para deshabilitar un job al vuelo, sin necesidad de reiniciar **bacula-dir**. Cuando el daemon de Bacula reinicie, el job tomará nuevamente el valor definido en el archivo de configuración.

### **enable job=job-name**

El caso opuesto al de arriba. El funcionamiento es el mismo.

---

<sup>1</sup>Entre la ejecución del comando y la terminación efectiva del job pueden transcurrir un tiempo (desde un minuto hasta dos horas). Además, si al reejecutar el comando se obtiene un mensaje de "Job not found", se debe a que alguno de los daemons de Bacula ya canceló. Estos errores vienen etiquetados según el origen: 1000 para el DIR, 2000 para el FD y 3000 para el SD

## **estimate job=job-name**

La forma completa del comando es:

```
estimate job=job-name listing client=client-name accurate=yes/no
        fileset=fileset-name level=level-name
```

Sirve para estimar la cantidad de archivos y bytes que se resguardarán. Como se muestra, acepta algunos argumentos opcionales.

El cálculo lo realiza el FD en el cliente, usando la cantidad de bloques en el archivo y no la cantidad de bytes, por lo que el tamaño final del job será un poco más chico. Se puede utilizar el argumento "accurate=yes" para que el cálculo esté mejor. La opción de *accurate* está disponible a partir de la versión 3.0.2.

Por último, se puede utilizar el flag *listing* que muestra el listado de todos los archivos involucrados en el job. Por ejemplo:

```
@output /tmp/listado
estimate job=unJob listing level=Incremental
@output
```

## **exit**

Cierra la consola

## **help**

Listado de comandos disponibles

## **label**

Se utiliza para para etiquetar volúmenes, ya sea de disco o cinta. La versión completa del comando es:

```
label storage=<storage-name> volume=<volume-name> slot=<slot>
```

El tipo de medio se obtiene automáticamente de la definición del *storage* elegido. El funcionamiento es sencillo: entrado el comando, la consola contacta al *storage daemon* y le pide que etiquete el volumen. Si el comando tiene éxito, la consola creará un volumen en el pool apropiado.

El nombre del volumen puede tener letras, números, "\_", ":", y ".". Ningún otro caracter es válido para el nombre de volumen.

Tener en cuenta que al etiquetar una cinta en blanco Bacula va a obtener un error de I/O -Cuando verifique que la cinta no tiene ya una etiqueta-. Para evitar este error, escribir un EOF en la cinta antes de ejecutar este comando:

```
mt rewind
mt weof
```

El comando *label* puede fallar por varias razones:

1. Ya existe un volumen con ese nombre

2. Hay montado un volumen en el dispositivo. En este caso, hay que desmontarlo, insertar la cinta en blanco y reejecutar el comando.
3. No hay un volumen en el dispositivo

Hay dos formas de reetiquetar un volumen que ya tiene una etiqueta de Bacula:

1. Para un volumen en disco, hay que borrar el archivo. Luego hay que ejecutar el comando *label*. Para una cinta, hay que escribir el EOF:

```
mt -f /dev/nst0 rewind
mt -f /dev/nst0 weof
```

Sin embargo, esta técnica podría dejar el catálogo inconsistente.

2. Por eso se prefiere esta otra técnica, que consiste en *purgar* el volumen utilizando el comando *purge* y luego ejecutando el comando *relabel*.

**Diferencia entre *st* y *nst*:** El driver *st* crea dos tipos de dispositivos: *st* y *nst*. A los primeros, el driver le envía el comando **REWIND** cuando se los cierra. Cuando el mismo dispositivo se abre usando la otra forma, *nst*, el driver no envía este comando. Notar que si se abre el dispositivo usando *st* y se posiciona la cinta en algún lugar en particular usando, por ejemplo, *mt*, el siguiente comando se ejecuta sobre el inicio de la cinta, porque al terminar de ejecutar *mt*, el driver retrocede automáticamente la cinta.

## list

Lista el contenido en el catálogo. Algunos ejemplos de la página de documentación:

```
list jobs

list jobid=<id>

list ujobid=<unique job name>

list job=<job-name>

list jobname=<job-name> <limit=nn>

list jobmedia

list jobmedia jobid=<id>

list jobmedia job=<job-name>

list files jobid=<id>

list files job=<job-name>

list pools
```

```
list clients

list jobtotals

list volumes

list volumes jobid=<id>

list volumes pool=<pool-name>

list volumes job=<job-name>

list volume=<volume-name>

list nextvolume job=<job-name>

list nextvol job=<job-name>

list nextvol job=<job-name> days=nnn
```

## **llist**

Toma los mismos argumentos que el comando anterior, con la diferencia que lista todo el contenido de los registros en el catálogo:

```
PoolId: 1
Name: Default
NumVols: 0
MaxVols: 0
UseOnce: 0
UseCatalog: 1
AcceptAnyVolume: 1
VolRetention: 1,296,000
VolUseDuration: 86,400
MaxVolJobs: 0
MaxVolBytes: 0
AutoPrune: 0
Recycle: 1
PoolType: Backup
LabelFormat: *
```

```
PoolId: 2
Name: Recycle
NumVols: 0
MaxVols: 8
UseOnce: 0
UseCatalog: 1
AcceptAnyVolume: 1
VolRetention: 3,600
VolUseDuration: 3,600
MaxVolJobs: 1
```

```
MaxVolBytes: 0
AutoPrune: 0
Recycle: 1
PoolType: Backup
LabelFormat: File
```

### messages

Muestra todos los mensajes pendientes.

### memory

Imprime el uso actual de memoria.

### mount

Es la forma de decirle a Bacula que hay un volumen y que Bacula puede leerlo. La forma completa del comando es:

```
mount storage=storage-name [ slot=num ] [ drive=num ]
mount [ jobid=id | job=job-name ]
```

### prune

Borra los registros relacionados a Jobs, Volumes y estadísticas en el catálogo. El comando borra los datos respetando los tiempos de retención definidos. Los datos que se borran son los del catálogo (Los datos en los volúmenes se mantienen hasta su reciclado).

Se pueden *prunear* registros Files, Jobs y Volumes, siempre que hayan expirado. Para prunear un volumen, el flag *VolStatus* debe estar en *Full*, *Used* o *Append* (De otra forma, no se lo *prunea*).

La versión completa del comando es:

```
prune files|jobs|volume|stats client=client-name volume=volume-name
```

### purge

Borra los registros relacionados con Jobs y Volumes sin tener en cuenta el período de retención. Como *prune*, borra sólo los datos del catálogos, quedando los volúmenes intactos hasta su reciclado. El peligro de usar este comando es que se pueden borrar registros en el catálogo asociados a backups actuales. En la documentación se recomienda **no** usarlo a menos que se sepa lo que se está haciendo.

Las formas del comando son:

```
purge files jobid=jobid|job=job-name|client=client-name
```

```
purge jobs client=client-name (of all jobs)
```

```
purge volume|volume=vol-name (of all jobs)
```

Nuevamente, para que se pueda operar sobre un Volume, el flag *VolStatus* debe estar en *Full*, *Used*, *Append* y se agrega el estado de *Error*.

## query

Ejecuta una consulta SQL en el catálogo. Las consultas SQL predefinidas se encuentran en el archivo definido en la propiedad *QueryFile* en el archivo de configuración del Director. Por ejemplo:

```
*query
```

```
Available queries:
```

- 1: List Job totals:
- 2: List up to 20 places where a File is saved regardless of the directory:
- 3: List where the most recent copies of a file are saved:
- 4: List last 20 Full Backups for a Client:
- 5: List all backups for a Client after a specified time
- 6: List all backups for a Client
- 7: List Volume Attributes for a selected Volume:
- 8: List Volumes used by selected JobId:
- 9: List Volumes to Restore All Files:
- 10: List Pool Attributes for a selected Pool:
- 11: List total files/bytes by Job:
- 12: List total files/bytes by Volume:
- 13: List Files for a selected JobId:
- 14: List Jobs stored in a selected MediaId:
- 15: List Jobs stored for a given Volume name:

```
Choose a query (1-15):
```

## quit

Cierra la consola. Primero la consola envía el comando al Director y espera el ACK. Pero si el Director está ocupado ejecutando un comando introducido anteriormente, puede tardar en responder con el ACK. Para esto, usar el comando *.quit*. quit

## relabel

Etiqueta un volumen físico. Para que el comando se puede ejecutar, el estado del volumen debe ser *Purged* o *Recycle*. Una vez reetiquetado, se borran todos los datos escritos en el volumen. La forma completa del comando es:

```
relabel storage=storage-name oldvolume=old-volume-name volume=newvolume-name
```

## release

Hace que el Storage rebobine *rewind* la cinta y que relea la etiqueta en el próximo uso.

```
release storage=storage-name
```

Notar que luego de ejecutar el comando, Bacula mantiene abierto el archivo, por lo que ningún otro programa puede usar la cinta. Si se quiere utilizar el dispositivo con otro programa (como *mt*) se debe desmontar el dispositivo para que Bacula cierre completamente el dispositivo.

## reload

Hace que el Director relea la configuración y aplique los nuevos valores.

Los cambios toman efecto inmediatamente a partir del siguiente Job. Tener en cuenta que el Scheduler agenda los Jobs con hasta dos horas de anticipación, por lo que los cambios que se espera tengan efecto durante las dos horas siguientes pueden verse demorados.

Los Jobs que se hayan agendado para correr (que hayan pasado su tiempo de inicio) continúan con los valores viejos. Los Jobs nuevos sí utilizan los valores nuevos.

Las configuraciones viejas se quedan en memoria mientras algún Job que usa esa configuración está corriendo (Luego se la quita de memoria). El Director puede mantener hasta diez configuraciones viejas en memoria (Luego rechaza el comando).

En la documentación se recomienda que, luego de un cambio en la configuración y un *reload*, se debe reiniciar el Director ni bien se tenga la oportunidad.

### **restore**

Permite elegir uno o más Jobs para la restauración de los datos que guardaron. Luego de elegir los Jobs, Bacula arma una estructura de directorios virtual para que elijamos los archivos y directorios que vamos a restaurar.

La forma completa del comando es:

```
restore storage=storage-name client=backup-client-name where=path pool=pool-name
  fileset=fileset-name restoreclient=restore-client-name select current all done
```

El flag *current* indica que restaure automáticamente el backup más actual. El flag *all* indica que restaure todos los archivos y directorios. El argumento *client* especifica el cliente de donde se copiaron los archivos y es el cliente por default donde se restaurarán los archivos. Sin embargo, si se especifica un *restoreclient*, la restauración se hará en ese cliente.<sup>2</sup>

### **run**

Por medio de este comando ejecutamos un Job a mano. La forma completa del comando es:

```
run job=job-name client=client-name fileset=FileSet-name level=level-keyword
  storage=storage-name where=directory-prefix when=universal-time-specification
  spooldata=yes|no yes
```

Cualquier dato necesario para ejecutar el Job y que no sea dado como argumento al comando, la consola lo solicitará mostrando un listado de opciones. Además, antes de la ejecución<sup>3</sup>, Bacula muestra tres opciones: Aceptar y ejecutar el Job; Rechazar y cancelar el Job; o Modificar algún parámetro del Job.

### **setdebug**

Permite configurar el nivel de debug en cada daemon:

---

<sup>2</sup>La documentación tiene un capítulo entero dedicado a la restauración de backups

<sup>3</sup>A menos que se haya usado el flag *yes*, en cuyo caso el Job se ejecuta directamente

```
setdebug level=nn [trace=0/1 client=client-name | dir | director |
storage=storage-name | all]
```

El argumento *trace* sirve para que el daemon entre en modo *trace*, escribiendo la traza en el archivo *bacula.trace*, en el directorio actual del daemon. Generalmente esta opción se usa con los clientes Win32, que no tienen un sistema de logging donde enviar sus eventos.

### setip

Configura una IP nueva para un cliente.

### show

Muestra la configuración de Bacula según *bacula-dir.conf* (o cualquiera que sea el archivo de configuración del Director). Sirve más que nada a los propósitos de debugging, siendo el comando *list* mucho más legible para el operador de backups.

```
show catalogs, clients, counters, devices, directors,
filesets, jobs, messages, pools, schedules, storages, all, help.
```

### sqlquery

Pone la consola en modo SQL, donde entramos consultas de SQL. Es más fácil usar la consola de MySQL o de PSQL.

### status

Muestra el estado de todos los componentes de Bacula (Director, Storage y Client). La forma completa es:

```
status [all | dir=dir-name | director [days=nnn] | client=client-name |
[slots] storage=storage-name]
```

Ejecutando *status dir* la consola muestra los Jobs corriendo, los Jobs que correrán en las próximas veinticuatro horas y un listado de los últimos diez terminados. Sobre los Jobs agendados para correr, hay que tener en cuenta que la información de los volúmenes no es exacta: por un lado, Bacula no tiene en cuenta los tiempos de *pruning* y el reciclaje. Por ejemplo, un Job anterior podría llenar el volumen.

Un ejemplo de la documentación:

```
2507 Catalog MatouVerify.2004-03-13_05.05.02 is waiting execution
5349 Full CatalogBackup.2004-03-13_01.10.00 is waiting for higher
priority jobs to finish
5348 Differe Minou.2004-03-13_01.05.09 is waiting on max Storage jobs
5343 Full Rufus.2004-03-13_01.05.04 is running
```

Para ver los jobs agendados para dentro de tres días, se puede indicar ese tiempo mediante el flag *days*.

Ejemplo de correr *status storage=File*:

Connecting to Storage daemon File at 192.168.68.112:8103

rufus-sd Version: 1.39.6 (24 March 2006) i686-pc-linux-gnu redhat (Stentz)  
Daemon started 26-Mar-06 11:06, 0 Jobs run since started.

Running Jobs:  
No Jobs running.  
=====

Jobs waiting to reserve a drive:  
=====

Terminated Jobs:

JobId	Level	Files	Bytes	Status	Finished	Name
59	Full	234	4,417,599	OK	15-Jan-06 11:54	kernsave

=====

Device status:

Autochanger "DDS-4-changer" with devices:

"DDS-4" (/dev/nst0)

Device "DDS-4" (/dev/nst0) is mounted with Volume="TestVolume002"

Pool="\*unknown\*"

Slot 2 is loaded in drive 0.

Total Bytes Read=0 Blocks Read=0 Bytes/block=0

Positioned at File=0 Block=0

Device "DVD-Writer" (/dev/hdc) is not open.

Device "File" (/tmp) is not open.

=====

In Use Volume status:

=====

#### **time**

Imprime la hora actual

#### **trace**

Configura en on/off la traza a archivos

#### **umount/unmount**

Hace que el Storage daemon desmonte el dispositivo indicado. Bacula no podrá usar el dispositivo hasta que no se monte un volumen ahí. Si Bacula necesita acceder a ese dispositivo y no hay un volumen montado, se bloquea y envía periódicamente mensajes pidiendo que se monte un volumen.

umount storage=<storage-name> [ drive=<num> ]

umount [ jobid=<id> | job=<job-name> ]

#### **update**

Actualiza algún registro en el Catálogo. Se aplica a los Pools y los Volumes.

El problema es mantener la coherencia entre la configuración en memoria y la que está en el archivo de configuración. update La forma completa del comando es:

```
update volume=xxx pool=yyy slots volstatus=xxx VolRetention=ddd VolUse=ddd
      MaxVolJobs=nnn MaxVolBytes=nnn Recycle=yes|no slot=nnn
      enabled=n recyclepool=zzz
```

#### **use**

Permite cambiar el catálogo a utilizar:

```
use database-name
```

#### **version**

Muestra la versión del Director

#### **wait**

Usado generalmente en modo *batch* para esperar a que termine un job determinado antes de continuar:

```
wait [jobid=nn] [jobuid=unique id] [job=job name]
```

## **2. Comandos en Bash**

El comando usado en Bash para manejar la cinta es *mt*. Su uso es bastante sencillo:

```
$ mt -f /dev/nst0 COMANDO
```

Donde COMANDO puede ser *status*, *weof*, *rewind*, entre otros. Para más información, ver *man mt*.

Además, Bacula viene con aplicaciones para ejecutar desde la línea de comandos, como *bls*, *bscan*, *bextract*. Generalmente estas herramientas no se utilizan, y sirven para acceder a los datos sin pasar por Bacula.

## **Referencias**

- [1] Página de Bacula con los comandos de la consola de Bacula  
[http://www.bacula.org/3.0.x-manuals/en/console/console/Bacula\\_Console.html](http://www.bacula.org/3.0.x-manuals/en/console/console/Bacula_Console.html)