# Disk Migration

## Using Bacula

This document is intended to provide insight into the considerations and processes required to migrate your disk volumes from a standard disk based backup to a disk backup using the Virtual Autochanger strategy outlined in the Disk Backup Strategy white paper.

Bacula Systems White Paper

Version 1.4, September 27, 2014

www.baculasystems.com

# Contents

# General

This White Paper will outline the best practices for migrating from a traditional disk backup solution to one using the Virtual Autochanger. We assume that you are running on Bacula version 7.0 or later, and that you already have a standard disk backup system running that may have many volumes already written, and that you would like to "import" those existing volumes into a new Virtual Autochanger setup.

This document has been adapted from the Bacula Enterprise document and is made available for your personal use in the hopes that it will allow community users to make better use of Bacula.

## 1.1 Assumptions

The following are the assumptions that we have made:

- You have a basic understanding of Bacula, Volumes, recycling, and backup strategies.
- You are using Bacula 7.0 or later.
- You are doing only disk based backup.
- You already have disks volumes created by an older Bacula configuration that you want to put into the new Virtual Autochanger.
- You have read and understood the Bacula white paper entitled "Disk Backup Design", which provides the details and strategies for creating a Virtual Autochanger disk backup system.
- You are using PostgreSQL as your Bacula catalog database. This is highly recommended by Bacula Systems.

## 1.2 Existing Configuration

We assume that you have some existing configuration in your **bacula-dir.conf** file that may look something like the following:

```
Storage {
  Name = File
  Address = xxxxx
  SDPort = 9103
  Password = "13TiHes7Y"
  Device = File
  Media Type = File
  Maximum Concurrent Jobs = 10        # run up to 10 jobs a the same time
}
```

You may have multiple different Storage devices to handle your needs. We also assume that your new configuration looks something like the following as is recommended in the Disk Backup Design white paper:

```
# Definition of file Virtual Autochanger device
Storage {
  Name = File1
  Address = xxxxx
  SDPort = 9103
  Password = "13TiHes7Y"
  Device = FileChgr1
  Media Type = File1
  Maximum Concurrent Jobs = 10        # run up to 10 jobs a the same time
  Autochanger = yes
}
```

In the **bacula-sd.conf** file, you had something like:

```
Device {
  Name = File
  Media Type = File
  Archive Device = /tmp
  LabelMedia = yes;
  Random Access = Yes;
  AutomaticMount = yes;
  RemovableMedia = no;
  AlwaysOpen = no;
  Maximum Concurrent Jobs = 5
}
```

And the new configuration looks somewhat like:

```
#
# Define a Virtual autochanger
#
Autochanger {
  Name = FileChgr1
  Device = FileChgr1-Dev1, FileChgr1-Dev2
  Changer Device = /dev/null

# For Bacula 5.2.x and earlier
# Changer Command = ""

  Changer Command = /dev/null
}

Device {
  Name = FileChgr1-Dev1
  Media Type = File1
  Archive Device = /home/bacula/vols
  LabelMedia = yes;
  Random Access = Yes;
  AutomaticMount = yes;
  RemovableMedia = no;
  AlwaysOpen = no;
  Maximum Concurrent Jobs = 5
  Autochanger = yes
}

...
```

Actually, you may have both the old and new configrations in your current Bacula configuration files to ease the migration.

Now, if you have previously written a Volume of name "Vol-001" on the device File, there will be several problems to fix before that Volume can be used in the new Autochanger resource.

- The old Volumes are stored in **/tmp** yet under the new scheme they are expected to be in **/home/bacula/vols**
- The old Volumes have MediaType "File" and the new Volumes must have MediaType "File1".
- The old volumes will be pointing to the File storage device and under the new scheme they should point to the FileChgr1 device. This may not be so obvious since it is something internal to the Bacula catalog. In involved the Storage catalog table and the StorageId fields.

  The StorageId field is part of the Media record (where Volume information is kept in the catalog). This StorageId field is used by Bacula during restores to select what device to use for the restore. Having the StorageId pointing to the wrong Storage resource will likely cause Bacula to choose the wrong Storage during a restore. In the worst case, you can manually set it during a restore, but it is preferable to correct it before hand.

## 1.3 Correcting the Volume Parameters

First do a "list volumes" command to see what we have, in particular we are looking for the MediaId and the MediaType of the Volume to be migrated.

```
./bconsole
...
*list volumes
Pool: Default
No results to list.
Pool: File
+---------+------------+-------------+------+-----------+
| MediaId | VolumeName | VolBytes    | Slot | MediaType |
+---------+------------+-------------+------+-----------+
|       1 | Vol-0001   | 134,399,629 |    0 | File      |
+---------+------------+-------------+------+-----------+
```

Where we have truncated certain columns for presentation purposes. As we mentioned, the MediaType is not what we want.
Now, let's look at the StorageId status, by doing the SQL command "select * from Storage;".

```
*sql
Entering SQL query mode.
Terminate each query with a semicolon.
Terminate query mode with a blank line.
Enter SQL query: select * from Storage;
+-----------+-----------+-------------+
| StorageId | Name      | AutoChanger |
+-----------+-----------+-------------+
|         1 | File      |           0 |
|         2 | FileChgr1 |           1 |
+-----------+-----------+-------------+
```

And as you can see, the StorageId of our File device is 1 while the FileChgr device has StorageId 2.
Let's now correct some of the items. First let's select the items we may want to change for our Volume, which we know from above has MediaId=1

```
./bconsole
*sql
Enter SQL query: select MediaId, StorageId, MediaType from Media where Volumename='Vol-0001';
+---------+-----------+-----------+
| MediaId | StorageId | MediaType |
+---------+-----------+-----------+
|       1 |         1 | File      |
+---------+-----------+-----------+
Enter SQL query: update Media SET StorageId=2 where MediaId=1;
Enter SQL query: select MediaId, StorageId, MediaType from Media where VolumeId=1;
+---------+-----------+-----------+
| MediaId | StorageId | MediaType |
+---------+-----------+-----------+
|       1 |         2 | File      |
+---------+-----------+-----------+
Enter SQL query:
```

First we checked what the current state was, then we issued the "update Media" command to modify the StorageId, and finally, we checked the results.
Now, we do the same with the MediaType with the following command:

```
update Media set MediaType='File1' where MediaId=1;
select MediaId, StorageId, MediaType from Media where VolumeId=1;
+---------+-----------+-----------+
| MediaId | StorageId | MediaType |
+---------+-----------+-----------+
|       1 |         2 | File1     |
+---------+-----------+-----------+
Enter SQL query:
```

At this point, the only problem that remains is that the Volume is in the wrong directory.
There are two options:

- Move the Volume from the old location (/tmp) to the new one (/home/bacula/vols).
- Soft link the Volume into the new location.

The first option can be completed with:

```
mv /tmp/Vol-0001 /home/bacula/vols
```

The second way is to leave it where it was, but supply a soft link so that Bacula can find it.

```
ln -s /tmp/Vol-0001 /home/bacula/vols/Vol-0001
```

Now you should be able to restore using the new Virtual Autochanger configuration.

# Revision History

| Version | Date | Owner | Changes |
|---|---|---|---|
| 1.3 | 18 Sep 2014 | Kern | Adapt to community version |
| 1.4 | 27 Sep 2014 | Kern | Tweaks |