Progressive Virtual Fulls™

Virtual Full Backups

How to get the most out of Progressive Virtual Full Backups.

Bacula Systems White Paper

Version 1.2, December 15, 2018

Conceptual Overview We assume the concepts of the *Incremental Forever* approach to be known in general. For the purposes of this introduction we assume that it is planned to do incremental-level backups extensively, and we also assume that Virtual Full backups will be done often.

The underlying idea here is to ensure that, at all times, a sequence of backups exists which consists of one full-level backup and a number of incremental ones. Virtual full backups are done comparatively often, so that in an example case there would always exist a backup history of 30 days of data, and no older backups would have to be kept.

The backup data would then consist of one full backup plus 30 incremental ones. Each day after an incremental backup has been done, a new virtual full backup would be created and the previously-existing one could be immediately removed (or, probably more valuable, would be moved to cheaper storage occasionally). Thus, the history of backed-up data would essentially allow reasonably fast restores to each of the previous 30 days, and not take up any more disk space than required. Essentially, the required backups would slide forward one day, once per day, as shown in figure 1.

Basic Implementation in Bacula Obviously, the key element of the outlined backup scheme is to create a new Virtual Full backup each day, aggregating the previous (virtual) full backup and the oldest incremental backup into a new virtual full one. This is easily done with Bacula, though it requires a bit of scripting to do.

This is due to the fact that, by default, Bacula would aggregate all individual backups of a sequence, whereas here we want to limit the resulting Virtual Full to a much shorter, apparently "older", sequence. From the point of view of a Bacula user, this is done by explicitly providing the Job ids of the backups to consolidate, and these need to be, for this purpose, determined by querying the Catalog database. Once the Job ids are known, however, the actual consolidation can be started using a very straightforward `bconsole` command.

The actual consolidation is done by reading all relevant blocks of data related to the jobs being consolidated – just like it would be at a restore to a point in time – and writing them out as a new backup instead of passing them to a File Daemon as would be done during an actual restoration.

This consolidation or virtual backup job would usually be done outside of the backup

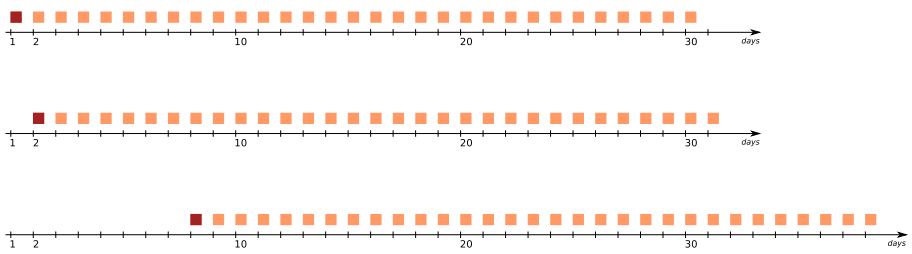


Figure 1: Backup Sequence Slides Forward One Day, Each Day

window, and it would not involve the actual client at all, so no negative impact on network throughput or client machine load would be caused. However, as all the relevant data has to be read and written, the amount of data that is moved can be quite considerable, and an accordingly long time would be required to finish the consolidation.

0.1 Automatic Consolidation

With Bacula version 9.0.0, the consolidation is done automatically when the Virtual Full jobs is run by adding a new directive to the Job resource. This new directive is:

```
Backups To Keep = <number>
```

The number specified is used to indicate how many backups you do not want merged. In the above example this would be done by setting **BackupsToKeep = 30**. The default is 0, which means that the Virtual Full job will run as it normally does.

If you only want to keep a week's worth of
result would be that the last Full and all backups other than the last 7 would be consolidated into a new Full backup.

0.2 Automatic Deletion of Consolidated Jobs

Also in Bacula version 9.0.0, we have added another new directive to the Job resource called:

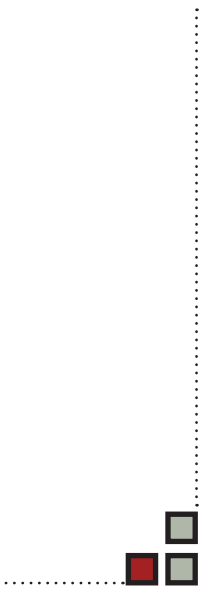
```
Delete Consolidated Jobs = yes/no
```

The default is **no**, but if this directive is set to **yes** then the old backups that were consolidated into a new Full backup will be deleted if the Virtual Full (consolidation) succeeds.
Some limitations and considerations need to be kept in mind:

- Virtual Backups are not compatible with all plugin-generated backup data.
- For long runs of Incrementals, it is advisable to use `Accurate = Yes`, which will increase the load on the client during backups.

For More Information

For more information on Bacula Enterprise Edition, or any part of the broad Bacula Systems services portfolio, visit www.baculasystems.com.



Rev : 2220 V. 1.2
Author(s): ARL,PCH