Bacula Cloud Backup

Bacula Community Version

This document is intended to provide insight into the considerations and processes required to implement Native Cloud Backup i.e. writing directly from your Bacula Storage Daemon to the Cloud.

Bacula Systems White Paper

Version 1.0, December 15, 2018

Contents

1 Native Cloud Backup **2**
- 1.1 Performance considerations . . . 2
  - 1.1.1 Cloud Account . . . 2
  - 1.1.2 Cloud Backup . . . 3
  - 1.1.3 S3 and S3-compatible Cloud Storages . . . 3
- 1.2 Cloud Costs . . . 3
  - 1.2.1 Cloud Volume Architecture . . . 4
  - 1.2.2 Cloud Restore . . . 4
  - 1.2.3 Compatibility . . . 5
  - 1.2.4 Virtual Autochangers and Disk Autochangers . . . 5
  - 1.2.5 Security . . . 5
  - 1.2.6 Cache and Pruning . . . 5
- 1.3 New Commands, Resource, and Directives for Cloud . . . 6
  - 1.3.1 New Bconsole Cloud Commands . . . 6
  - 1.3.2 Cloud Additions to the Director Pool Resource . . . 7
  - 1.3.3 Cloud Additions to the Storage Daemon Device Resource . . . 7
  - 1.3.4 New SD Cloud Device Directives . . . 7
  - 1.3.5 Example Cloud Device Resource . . . 8
  - 1.3.6 The Cloud Resource . . . 9

2 Creating and Verifying your Cloud Account **11**
- 2.1 Amazon S3 . . . 11
  - 2.1.1 Create an Account . . . 11
  - 2.1.2 Install the AWS Command Line Interface . . . 12
  - 2.1.3 Define user and export credentials . . . 12
  - 2.1.4 Create a Bucket in Amazon S3 . . . 12
  - 2.1.5 Copy and sync files . . . 13
  - 2.1.6 Time coordination . . . 14

3 File Driver for the Cloud **14**

4 Emulations **14**
- 4.1 Using FakeS3 Instead of Amazon . . . 14

5 Price and Billing **15**

6 Limitations **15**

Executive summary

IT organizations are constantly being challenged to deliver high quality solutions with reduced total cost of ownership. One of those challenges is the growing amount of data to be backed up, together with limited space to keep backups. Another major challenge is to provide adequate off-site backup. Bacula offers several ways to tackle these challenges, one of them being *Bacula Cloud Backup*, which writes Bacula Volumes to different types of cloud services.

This document is intended to provide insight into the considerations and processes required to successfully implement this backup technique.

1 Native Cloud Backup

1.1 Performance considerations

Cloud storage has been around for a number of years, but until recently, it was not very interesting for backup because of the lack of bandwidth for uploading and downloading the large volumes of data that today's backups use. With the advent of fibre optics and high-speed copper Internet connections, the Cloud has become a viable and cost effective solution for either primary or off-site data backup.

Starting with Bacula Community Version 9.4.0, the Bacula project has implemented drivers for backup and restore to and from several cloud services, whether public or private. The architecture of the Bacula Enterprise cloud backup is new, and it will provide the user with an array of features to keep the cloud costs to a minimum and the performance to a maximum.

To provide this new Cloud functionality we have made significant changes to the Storage daemon backend device architecture. The new Bacula version is more modular and permits the devices to select from multiple drivers to handle different Cloud technologies. For the first release we included an S3 driver that is compatible with a large number of S3 cloud services. However, Google Cloud and Oracle Cloud are not compatible with S3, so are not supported by this version. A Cloud File driver which is useful for testing the Cloud architecture without requiring a Cloud account is also included, and could possibly be useful for a disk media device that is very slow.

1.1.1 Cloud Account

To configure a Bacula Storage daemon to use cloud storage, you must either have your own cloud or an account with an S3 or a compatible Cloud service provider. We have tested our implementation using an **AWS** account with **Amazon**

- `https://aws.amazon.com/`

and with **Wasabi**

- `https://wasabi.com/`

We have also tested our S3 implementation with "FakeS3", a lightweight server clone of **Amazon S3** that simulates most of the commands supported by S3 with minimal dependencies.

- https://github.com/eproxus/fakes3-docker

1.1.2 Cloud Backup

A major problem of Cloud backup is that data transmission to and from the Cloud is very slow compared to traditional backup to disk or tape. The Bacula Cloud drivers provide a means to quickly finish the backups and then transfer the data from the local cache to the Cloud in the background. This is done by first splitting the data Volumes into small parts that are cached locally and then uploading those parts to the Cloud storage service in the background, either while the job continues to run or after the backup Job has terminated. Once the parts are written to the Cloud, they may either be left in the local cache for quick restores or they may be removed (truncate cache). Truncating cache volumes may also be configured to occur automatically in the background during a job, or after the job has completed. Truncation may also be disabled, or configured to be run manually.

1.1.3 S3 and S3-compatible Cloud Storages

The S3 driver is also compatible with any of the following cloud storage technologies:

- Ceph Object Storage, using S3 Gateway
- Swift3 Middleware for OpenStack Swift, using AWS Signature Version 4

1.2 Cloud Costs

A note about costs

As you will already know, storing data in the cloud will create additional costs. Amazon for instance has a pricing model for each of its storage tiers, and in addition the costs will vary with the region you use:

https://aws.amazon.com/s3/pricing/

Data transfer needs to be considered as well. While upload of data is typically free or very low cost, the download is typically not free, and you will be charged which means that restores from the cloud can become expensive. Also note, that when you write data to a volume with Bacula, some data will go the opposite direction for data verification and other important tasks.

You might be able to reduce your storage costs by enabling one of Bacula's available data compression techniques.

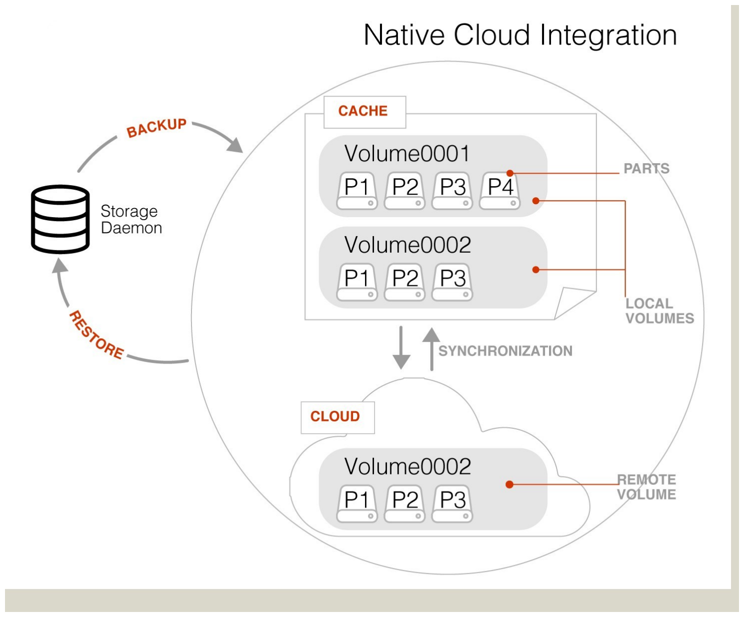

Figure 1: Bacula Cloud Architecture

1.2.1 Cloud Volume Architecture

Figure 1 shows two Bacula Cloud Volumes (Volume0001 and Volume0002) with their parts in the local cache. Below the cache, one can see that Volume0002 has been uploaded, or "synchronized" with the Cloud.

Note: Regular Bacula Disk Volumes are implemented as standard files that reside in the user defined **Archive Device** directory. Each regular Bacula Disk Volume is just one file. On the other hand, Bacula Cloud Volumes are directories that reside in the user defined **Archive Device** directory. Each Volume directory contains the Cloud Volume parts which are implemented as numbered files (part.1, part.2, ...).

1.2.2 Cloud Restore

During a restore, if the needed Cloud Volume parts are in the local cache, they will be immediately used, otherwise, they will be downloaded from the cloud as necessary. In such a case, Bacula is efficient and attempts to be as cost effective as possible by only dowloading the actual parts of the Cloud Volumes required to perform the restore. The restore starts with Cloud Volume parts already in the local cache but will wait in turn for any part that needs to be downloaded. The downloads proceed in the background while the restore is running.

With most cloud providers uploads are free of charge. On the other hand, downloads of data from the cloud are usually billed. By using local cache and multiple small parts, you can configure Bacula to substantially reduce your Cloud download costs during restores.

The `MaximumFileSize` Device directive is still valid within the Storage Daemon and defines the granularity of a restore chunk. In order to limit volume parts to download during a restore (especially when restoring single files), it might be useful to set the `MaximumFileSize` to a value smaller than or equal to the `MaximumPartSize`.

1.2.3 Compatibility

Since a Cloud Volume contains the same data as an ordinary Bacula Volume, all existing types of Bacula data may be stored in the cloud – that is client encrypted data, compressed data, plugin data, etc. All existing Bacula functionality, with the exception of deduplication, is available with the Bacula Cloud drivers.

1.2.4 Virtual Autochangers and Disk Autochangers

If you use a Bacula Virtual Autochanger you will find it compatible with the new Bacula Cloud drivers. However, if you use a third party disk autochanger script such as **vchanger**, unless or until it is modified to handle Volume directories, it may not be compatible with Bacula Cloud drivers.

1.2.5 Security

All data that is sent to and received from the cloud by default uses the HTTPS protocol, so your data is encrypted while being transmitted and received. However, data that resides in the Cloud is not encrypted by default. If you wish extra security of your data while it resides in the cloud, you should consider using Bacula's PKI data encryption feature during the backup.

1.2.6 Cache and Pruning

The Cloud Cache is treated much like a normal Disk based backup, so that when configuring Cloud backups, the administrator should take care to set "Archive Device" in the Device resource to a directory where he/she would normally store data backed up to disk. Obviously, unless he/she uses the truncate/prune cache commands, the Archive Device will continue to fill.

The retention of Cloud volumes in the local Cache is controlled per Volume with the new “CacheRetention” Volume attribute. The default value is 0, meaning that the pruning of Cloud Cache Volumes is disabled. The new “CacheRetention” attribute for Cloud Volumes is configured as a Directive in a Pool and is inherited by Cloud Volumes created in this Pool just as with other inherited attributes for regular disk based Pools and Volumes.

The “CacheRetention” value for a volume may be modified with the bconsole “update” command.

1.3 New Commands, Resource, and Directives for Cloud

To support Cloud backups in Bacula Enterprise 8.8 there are new **bconsole** commands, new Storage Daemon directives, the new Pool directive mentioned above, and a new Cloud resource that is specified in the Storage Daemon's configuration.

1.3.1 New Bconsole Cloud Commands

- **Cloud** The new **bconsole** "cloud" command allows you to do a number of things with cloud volumes. The options are the following:
  - **None** If you specify no arguments to the command, bconsole will prompt with:

    ```
    Cloud choice:
      1: List Cloud Volumes in the Cloud
      2: Upload a Volume to the Cloud
      3: Prune the Cloud Cache
      4: Truncate a Volume Cache
      5: Done
    Select action to perform on Cloud (1-5):
    ```

    The different choices should be rather obvious.
  - **Truncate** This command will attempt to truncate the local cache for the specified Cloud Volume. Bacula will prompt you for the information needed to determine the Volume name or names. To avoid the prompts, the following additional command line options may be specified:
    - **Storage=xxx**
    - **Volume=xxx**
    - **AllPools**
    - **AllFromPool**
    - **Pool=xxx**
    - **Storage=xxx**
    - **MediaType=xxx**
    - **Drive=xxx**
    - **Slots=nnn**
  - **Prune** This command will attempt to prune the local cache for the specified Volume. Bacula will respect the CacheRetention volume attribute to determine if the cache can be truncated or not. Only parts that have been successfully uploaded to the cloud will be deleted from the cache. Bacula will prompt you for the information needed to determine the Volume name or names. To avoid the prompts, the following additional command line options may be specified:
    - **Storage=xxx**
    - **Volume=xxx**
    - **AllPools**
    - **AllFromPool**

    * **Pool=xxx**
    * **Storage=xxx**
    * **MediaType=xxx**
    * **Drive=xxx**
    * **Slots=nnn**
  - **Upload** This command will attempt to upload the specified Volumes. It will prompt you for the information needed to determine the Volume name or names. To avoid the prompts, you may specify any of the following additional command line options:
    * **Storage=xxx**
    * **Volume=xxx**
    * **AllPools**
    * **AllFromPool**
    * **Storage=xxx**
    * **Pool=xxx**
    * **MediaType=xxx**
    * **Drive=xxx**
    * **Slots=nnn**
  - **List** This command will list volumes stored in the Cloud. If a volume name is specified, the command will list all parts for the given volume. To avoid the prompts, you may specify any of the following additional command line options:
    * **Storage=xxx**
    * **Volume=xxx**

1.3.2 Cloud Additions to the Director Pool Resource

Within the **bacula-dir.conf** file, for each **Pool** resource there is an additional directive **CacheRetention** that may be specified.

1.3.3 Cloud Additions to the Storage Daemon Device Resource

Within the **bacula-sd.conf** file, for each **Device** resource there is an additional keyword **Cloud** that must be specified as the **Device Type** directive, and two new directives **MaximumPartSize** and **Cloud**.

1.3.4 New SD Cloud Device Directives

- **DeviceType** The Device Type has been extended to include the new keyword **Cloud** to specify that the device supports cloud Volumes. Example:

```
Device Type = Cloud
```

- **Cloud** A Device's new Cloud directive specifies a Cloud resource. As with other Bacula resource specifications, this directive specifies the name of the Cloud resource that this device will use. Example:

```
Cloud = MyCloud
```

- **MaximumPartSize** This directive specifies the maximum size for each Cloud Volume part. Smaller part sizes will reduce restore costs, but may require a small additional overhead to handle multiple parts. The maximum number of parts permitted in a Cloud Volume is 524,288. The maximum size of any given part is approximately 17.5TB.

1.3.5 Example Cloud Device Resource

The following is an example of a Bacula Cloud Device resource:

```
Device {
  Name = CloudStorage
  Device Type = Cloud
  Cloud = MyCloud
  Archive Device = /opt/bacula/backups
  Maximum Part Size = 10 MB
  Media Type = CloudType
  LabelMedia = yes
  Random Access = Yes;
  AutomaticMount = yes
  RemovableMedia = no
  AlwaysOpen = no
}
```

As you can see from the above example, the **Cloud** directive in the **Device** resource contains the name (**MyCloud**) of the **Cloud** resource that is shown below. Note also the **Archive Device** is specified in the same manner as one would use for a File device, that is, it simply points to a directory.
However, since this is a Cloud Device, instead of the Storage Daemon writing one file per Bacula Volume in this directory, the Storage daemon will create one directory per Cloud Volume here, and in each of these Cloud Volume directories, the Volume parts will be written.
With the above Device resource example, the two cache Volumes shown in figure 1 on page 4 above would have the following layout on disk:

```
  /opt/bacula/backups
     /opt/bacula/backups/Volume0001
        /opt/bacula/backups/Volume0001/part.1
        /opt/bacula/backups/Volume0001/part.2
        /opt/bacula/backups/Volume0001/part.3
        /opt/bacula/backups/Volume0001/part.4
     /opt/bacula/backups/Volume0002
        /opt/bacula/backups/Volume0002/part.1
        /opt/bacula/backups/Volume0002/part.2
        /opt/bacula/backups/Volume0002/part.3
```

1.3.6 The Cloud Resource

The **Cloud** resource has a number of directives that may be specified as shown in the following examples:
default S3 east USA location:

```
Cloud {
  Name = MyCloud
  Driver = "S3"
  HostName = "s3.amazonaws.com"
  BucketName = "BaculaVolumes"
  AccessKey = "BZIXAIS39DP9YNER5DFZ"
  SecretKey = "beesheeg7iTe0Gaexee7aedie4aWohfuewohGaa0"
  Protocol = HTTPS
  UriStyle = VirtualHost
  Truncate Cache = No
  Upload = EachPart
  Region = "us-east-1"
  MaximumUploadBandwidth = 5MB/s
}
```

S3 central europe location:

```
Cloud {
  Name = MyCloud
  Driver = "S3"
  HostName = "s3-eu-central-1.amazonaws.com"
  BucketName = "BaculaVolumes"
  AccessKey = "BZIXAIS39DP9YNER5DFZ"
  SecretKey = "beesheeg7iTe0Gaexee7aedie4aWohfuewohGaa0"
  Protocol = HTTPS
  UriStyle = VirtualHost
  Truncate Cache = No
  Upload = EachPart
  Region = "eu-central-1"
  MaximumUploadBandwidth = 4MB/s
```

For Amazon Cloud, refer to http://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region to get a complete list of regions and corresponding endpoints and use them respectively as Region and HostName directives.
For CEPH S3 interface:

```
Cloud {
  Name = CEPH_S3
  Driver = "S3"
  HostName = ceph.mydomain.lan
  BucketName = "CEPHBucket"
  AccessKey = "xxxXXXxxxx"
  SecretKey = "xxheeg7iTe0Gaexee7aedie4aWohfuewohxx0"
  Protocol = HTTPS
  Upload = EachPart

  UriStyle = Path            # Must be set for CEPH
}
```

The directives of the Cloud resource examples above are defined as follows:

- **Name** The name of the Cloud resource. Shown as **MyCloud** above.
- **Description** The description is used for display purposes in Bweb as it is the case with all resources. Not shown in examples above.
- **Driver** This defines which driver to use. Valid options are **S3**. There is also a **File** driver, which is used mostly for testing. The specific differences for the Cloud directives that are different in the File driver are discussed in the next section.
- **Host Name** This directive specifies the hostname to be used in the URL. Each Cloud service provider has a different and unique hostname. The maximum size is 255 characters.
- **Bucket Name** This directive specifies the bucket name that you wish to use on the Cloud service. This name is normally a unique name that identifies where you want to place your Cloud Volume parts. With Amazon S3, the bucket must be created previously on the Cloud service. The maximum bucket name size is 255 characters.
- **Access Key** The access key is your unique user identifier given to you by your cloud service provider.
- **Secret Key** The secret key is the security key that was given to you by your cloud service provider. It is equivalent to a password.
- **Protocol** The protocol defines the communications protocol to use with the cloud service provider. The two protocols currently supported are: **HTTPS** and **HTTP**. The default is **HTTPS**.
- **Uri Style** This directive specifies the URI style to use to communicate with the cloud service provider. The two Uri Styles currently supported are: **VirtualHost** and **Path**. The default is **VirtualHost**.
- **Truncate Cache** This directive specifies if and when Bacula should automatically remove (truncate) the local cache Volume parts. Local cache Volume parts can only be removed if they have been successfully ploaded to the cloud. The currently implemented values are:
  - **No** Do not remove cache Volume parts. With this setting, you must manually delete the cache parts with a **bconsole Truncate Cache** command, or do so with an **Admin** Job that runs a **Truncate Cache** command. If not specified, this is the default.
  - **AfterUpload** Each cache Volume part will be removed just after it is uploaded. Note, if this option is specified, all restores will require a download from the Cloud.
  - **AtEndOfJob** At the end of the Job, every part that has been successfully uploaded to the Cloud will be removed (truncated). Note, if this option is specified, all restores will require a download from the Cloud. **N**ote: Not yet implemented.

- **Upload** This directive specifies when local cache Volume parts will be uploaded to the Cloud. The options are:
  - **Manual** Do not upload Volume cache parts automatically. With this option you must manually upload the Volume cache parts with a **bconsole Upload** command, or do so with an **Admin** Job that runs an **Upload** command. If not specified, this is the default.
  - **EachPart** With this option, each cache Volume part will be uploaded when it is complete i.e. when the next Volume part is created or at the end of the Job.
  - **AtEndOfJob** With this option all cache Volume parts that have not been previously uploaded will be uploaded at the end of the Job. **N**ote: Not yet implemented.
- **Maximum Upload Bandwidth** The default is unlimited, but by using this directive, you may limit the upload bandwidth used globally by all devices referencing this Cloud resource.
- **Maximum Download Bandwidth** The default is unlimited, but by using this directive, you may limit the download bandwidth used globally by all devices referencing this Cloud resource.
- **Region** the Cloud resource may be configured to use a specific endpoint within a region. This directive is required for AWS-V4 regions. ex: `Region="eu-central-1"`

2 Creating and Verifying your Cloud Account

2.1 Amazon S3

2.1.1 Create an Account

Amazon offers high-level S3 commands with the AWS Command Line Interface. This tool (**aws**) can be used to verify your account and list everything you have stored in you Amazon S3 buckets. The advantage of testing your setup with the **aws** command is that the same Amazon credentials are used in accessing S3 via **aws** as in the Bacula Cloud resource.
Please go to the following link and create an Amazon S3 account. If you just want to test, you can sign up for a free 1 year trial subscription that allows small scale testing at no charge.

- `https://aws.amazon.com/s3`

Then you might want to use the Amazon tutorial. Note some of the same information in the tutorial is repeated below for your convenience.

- `https://aws.amazon.com/s3/getting-started/`

2.1.2 Install the AWS Command Line Interface

A guide which explains how to install the AWS Command Line Interface (CLI) can be found here:

- http://docs.aws.amazon.com/cli/latest/userguide/installing.html

We chose pip to install the AWS CLI. This worked despite the fact that our CentOS 6.7 SD only offered Python 2.6, which produced several warnings. Amazon recommends to have at least Python version 2.7.
Make sure that Python is installed:

```
[root@localsd1 ~]# python --version
Python 2.6.6
[root@localsd1 ~]#
```

The pip tool wasn't installed in our case which can be checked with:

```
[root@localsd1 ~]# pip --help
-bash: pip: command not found
[root@localsd1 ~]#
```

Installation is easy (but produces warnings with Python versions < 2.7):

```
[root@localsd1 ~]# curl -O https://bootstrap.pypa.io/get-pip.py
[root@localsd1 ~]# python get-pip.py
```

Once you have pip, you can install the AWS CLI and check if the installation was successful:

```
[root@localsd1 ~]# pip install awscli
[root@localsd1 ~]# aws help
```

2.1.3 Define user and export credentials

To be able to access your Amazon S3 buckets, you will need to create an authorized user in the web interface: *Services* → *Security & Identity* → IAM. Attach the policy *AmazonS3FullAccess* to this user. Create access keys in the web portal (tab *Security Credentials* in IAM) and export them as a CSV file.
Use the configure command:

```
[root@localsd1 ~]# aws configure
```

to store the credentials locally (see also http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html)

2.1.4 Create a Bucket in Amazon S3

Log into your AWS console (https://aws.amazon.com/console/), select the correct region, and choose *Services* → *Storage & Content Delivery* → S3. Create a bucket and give it a unique name. In our test scenario we created a bucket called *bsyssdsync* with one folder *volumes* inside it.
You can also create the bucket with the AWS Command Line Interface:

```
[root@localsd1 ~]# aws s3api --endpoint-url='https://s3.amazonaws.com' \
  create-bucket --bucket bsyssdsync
```

2.1.5 Copy and sync files

You can list your Amazon S3 buckets with:

```
[root@localsd1 ~]# aws s3 ls
2016-04-11 21:13:02 bsyssdsync
[root@localsd1 ~]#
```

To copy volumes from your local SD to the cloud, use:

```
[root@localsd1 ~]# cd /srv/bacula-storage
[root@localsd1 bacula-storage]# ls
Vol-0002  Vol-0005
[root@localsd1 bacula-storage]#
[root@localsd1 bacula-storage]# aws s3 cp Vol-0002 s3://bsyssdsync/volumes/
upload: ./Vol-0002 to s3://bsyssdsync/volumes/Vol-0002
[root@localsd1 bacula-storage]#
```

This of course only makes sense when you have only one job per volume and your volumes are marked Full by Bacula after the job. You could trigger the upload to the cloud in a RunScript after the job and then delete the local copy. You would have to make sure though that in the restore case all volumes are available again in the local file system.
The **aws s3 cp** works in both directions:

```
[root@localsd1 ~]# aws s3 cp <source> <destination>
```

and behaves like the UNIX cp command (more details: http://aws.amazon.com/cli/). However, when you have more than one job per volume, and volumes with fixed maximum size configured in Bacula, you will want to sync the directory with Bacula volumes to S3. The AWS CLI has a command for that:

```
[root@localsd1 ~]# aws s3 sync /srv/bacula-storage s3://bsyssdsync/volumes
```

In this case you would identify Full volumes with a Bacula Catalog query and delete them after all backups have been run and the volumes have been synced. Again, you will need to make sure that they are available when you want to restore data. When it comes to Bacula reusing volumes (after the configured retention times have passed), you would probably use a different configuration approach in a cloud scenario: Configure retention times to be indefinitely long (i.e. years), the volumes will be synced away into the cloud, deleted from disk, and then you will use Amazon mechanisms to tier the volumes to less expensive storage, from

General Purpose (Amazon S3 Standard)
↓
Infrequent Access (Amazon S3 Standard - Infrequent Access)
↓
Archive (Amazon Glacier)

and finally delete them. Learn more about Amazon Storage Classes: https://aws.amazon.com/s3/storage-classes/. Policies can be set for each bucket independently in the AWS web portal.

2.1.6 Time coordination

It seems that Amazon S3 syncing is sensitive to clock differences. If you get a S3-RequestTimeTooSkewed error during **aws s3 sync** you should use Amazon NTP servers:
http://www.emind.co/how-to/how-to-fix-amazon-s3-requesttimetooskewed

3 File Driver for the Cloud

As mentioned above, one may specify the keyword **File** on the **Driver** directive of the Cloud resource. Instead of writing to the Cloud, Bacula will instead create a Cloud Volume but write it to disk. The rest of this section applies to the **Cloud** resource directives when the File driver is specified.

As noted above, the File driver is mostly used for testing purposes. However, if you have a particularly slow backup device you might want to stage your backup data into an SSD or disk using the local cache feature of the Cloud device, and have your Volumes transferred in the background to a slow File device.

4 Emulations

4.1 Using FakeS3 Instead of Amazon

If you wish to test Bacula's S3 cloud driver without having an account at Amazon, it is possible using the **FakeS3** program. **FakeS3** can be installed by running the 'fakes3' program as root.

- **host** is the **HostName** you want to specify in your Bacula **Cloud** resource to access this S3 simulator.
- **root** is the directory on your local machine where FakeS3 will store data that is uploaded to it.
- **port** is a unique port number (such as 8080) that you use in your **Cloud** resource to access the FakeS3 simulator.

Assuming you used a host hame of **localhost** and an output directory **/opt/bacula/S3_Volumes** and a port of **8080**, your **bacula-sd.conf** Cloud resource would look like the following:

```
Cloud {
  Name = CloudSimulator
  Driver = "S3"
  HostName = "localhost:8080"
  BucketName = "BaculaVolumes"
  AccessKey = "1234"
  SecretKey = "1234"
  Protocol = HTTP
  UriStyle = Path
```

```
   Truncate Cache = No
   Upload = EachPart
 }
```

Note, prior to running **FakeS3**, you must create a directory named: **/opt/bacula/volumes/BaculaVolumes**. Note also, that the AccessKey and the SecretKey are not used, but must be specified. For FakeS3 (my version) you must use an **HTTP** protocol and the **Path** UriStyle.
Once that is in place, you start FakeS3 as follows:

```
fakes3 --hostname=localhost --root=/opt/bacula/S3_Volumes --port=8080
```

Then you can start and run Bacula to backup to the FakeS3 cloud.

5 Price and Billing

- The cloud storage costs are depending on the cloud provider and are subject to changes. Thus, Bacula can't guaranty or estimate those costs since we're not the one charging for them.
- Nevertheless, as a guideline at the time this whitepaper is written, the following information can be used to estimate the storage price.
- It's mostly important to evaluate the total size of the uploaded parts, since the storage size used is generally the biggest part of the final bill.
- Some providers charge for the upload, the download or both. It might be important to get an idea of the backup and restore frequency to get a better price evaluation.
- In addition, a fixed amount per basic operation is generally charged, although it should represents a small amount of the final cost.

6 Limitations

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 : 2 V. 1.0
Author(s): KES