Security Plugin

Bacula Systems Documentation

Contents

Contents

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.

Bacula Systems 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 will provide the user with an array of features to keep the cloud costs to a minimum and the performance to a maximum.

In a continuous effort to increase end user choices, Bacula Systems has broadened its offer of cloud plugins over the last releases. You can now choose from the following plugins:

- S3/Amazon Cloud Plugin
- Azure Cloud Plugin
- Google Cloud Plugin
- Oracle Cloud Plugin
- Swift Object Storage Plugin

Each plugin can be purchased separately. 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 Cloud accounts

You need to create an account with the different cloud providers (or use an existing one) before you can start to use one of the cloud drivers for the Bacula Storage Daemon:

1.1 S3/Amazon

To configure a Bacula Storage daemon to use S3 cloud storage, you must have an account with an S3 compatible Cloud service provider.

The S3 Cloud Plugin has been tested with:

- AWS: https://aws.amazon.com/
- Wasabi: https://wasabi.com/
- Backblaze: https://www.backblaze.com/
- Huawei Cloud Storage

1.2 Azure

We have tested our **Azure** implementation with **Microsoft Azure** account.

- https://azure.microsoft.com

1.3 Google

- https://cloud.google.com/

Access to the Google Cloud Console is necessary:

- https://console.cloud.google.com/

1.4 Oracle

- https://cloud.oracle.com/

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.

3 Cloud Compatibility Considerations

3.1 S3/Amazon

The S3/Amazon 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

3.2 Azure

Only Microsoft Azure Storage has been validated.

4 Cloud Costs

Note: General cost considerations

As you will already know, storing data in the cloud will create additional costs. Please see below information for the different cloud providers.

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.

4.1 S3/Amazon

S3/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/

4.2 Azure

With Azure redundancy might influence the price:

- https://azure.microsoft.com/en-us/pricing/details/storage/blobs/

4.3 Google

You can refer to the Google Cloud price calculator to estimate your storage costs.

- https://cloud.google.com/products/calculator/

4.4 Oracle

You can refer to the Oracle Cloud Cost Estimator to estimate your storage costs.

- https://cloud.oracle.com/en_US/cost-estimator

Remember that Bacula stores its parts to the Object Storage Oracle format. The bucket storage tier influences the price.

4.5 Swift

Use the Swift Storage price calculator to estimate your storage costs:

- https://www.swiftstack.com/pricing

5 Cloud Volume Architecture

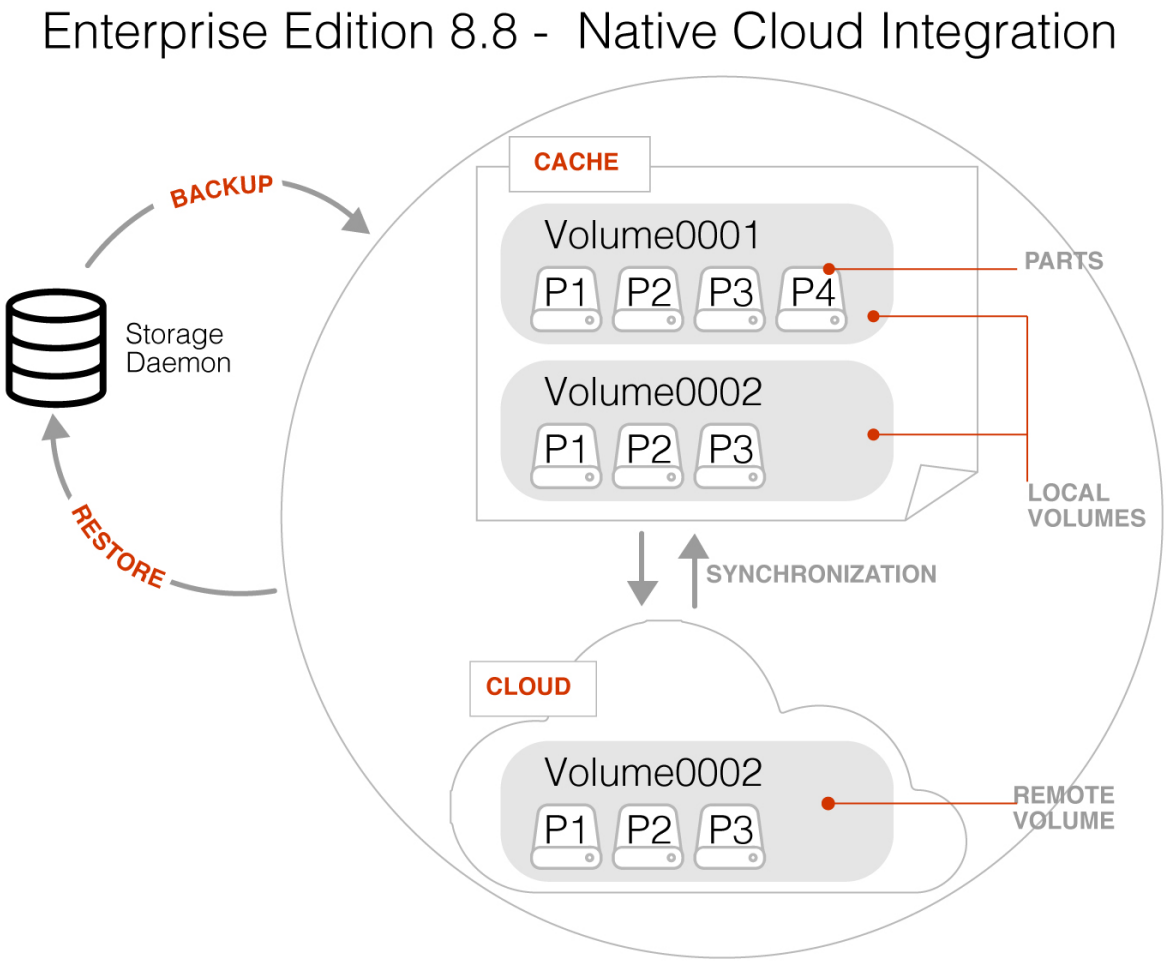

Fig. 1: Bacula Cloud Architecture

In the picture above you see 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, ...).

---

6 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 downloading only 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`.

6.1 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, volumeencryption, compressed data, plugin data, etc. All existing Bacula functionality, with the exception of deduplication, is available with the Bacula Cloud drivers.

6.2 Deduplication and the Cloud

At the current time, Bacula Global Endpoint Deduplication does not support writing to the cloud because the cloud would be too slow to support large hashed and indexed containers of deduplicated data.

6.3 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.

6.4 Security and Data Immutability

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 or volumeencryption.

For additional protection against backup data loss, or for regulatory compliance reasons, cloud stored parts can be set to be immutable, which means they can be downloaded from the cloud many times but uploaded to the cloud only once (Write Once Read Many: WORM).

Bacula Cloud Plugin supports the immutability features available from different cloud providers. Immutability needs to be configured externally in the destination storage entity (S3 Bucket, Azure Blob, Google Storage Bucket…) using the available native tools from each provider. Further information about these features:

- S3 Object Lock: https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html
- Azure Blob Immutable Storage: https://learn.microsoft.com/en-us/azure/storage/blobs/immutable-policy-configure-container-scope?tabs=azure-portal

- Google cloud Bucket Lock: https://cloud.google.com/storage/docs/bucket-lock
- Oracle cloud Retention Rules: https://docs.oracle.com/en-us/iaas/Content/Object/Tasks/usingretentionrules.htm

Once the destination storage has immutability capabilities enabled, Bacula will work transparently with it. The only requirement is to have greater Bacula retention for the implied volumes than the retention configured in the cloud.

6.5 Bucket Versioning

In the case you have bucket versioning enabled in the bucket used to store the Bacula Cloud volumes, you should setup a proper procedure to delete the versioned part files to avoid unnecessary costs.

Versioned part files are created e.g. when Bacula reuses a Cloud volume.

Cloud providers usually propose the setup of Lifecycle policies to delete periodically versioned objects from the bucket.

7 Plugin installation

7.1 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.

7.2 Truncate Cloud Volumes

When Cloud Volumes are truncated by using the `Truncate` bconsole command, all the part files of the volumes are deleted from the Cloud, except the `part.1` file that contains the Bacula Volume label. In the local cache, the `part.2` file is created, with zero bytes, and it is not uploaded to the Cloud. Next time the volume is used by a Bacula Job (backup, copy, etc.), the job will start writing to the Cloud volume `part.2` file.

8 Commands, Resource, and Directives for Cloud

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

8.1 Cloud Additions to the Director Pool Resource

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

For details, click here.

8.2 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 three new directives: **MaximumPartSize**, **MaximumVolumeParts** and **Cloud**.

For details, click here.

8.3 SD Cloud Device Directives

- Device Type
- Cloud
- Maximum Part Size
- Maximum Volume Parts

8.4 Example Cloud Device Resource

The following is an example of a Bacula Cloud Device resource (more examples below):

```
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 *Bacula Cloud Architecture* 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
```

9 The Cloud Resource

The **Cloud** resource has a number of directives that may be specified as shown in the examples below.

Not all of the directives are mandatory for each plugin (see remarks in individual subsections).

9.1 S3/Amazon

Amazon Driver

Important: The S3 Driver is now deprecated. If you are still using it, move from the S3 driver to the Amazon driver with the same directives and parameters as they are most of them compatible. Both drivers come with the plugin Cloud S3 or the package **bacula-enterprise-cloud-storage-s3**.

Note: Starting with Bacula Enterprise 16.0.6, the **BlobEndpoint** directive needs to be set when Amazon driver is utilized with a non AWS cloud storage.

Default East USA Amazon Region (us-east-1):

```
Cloud {
  Name = MyCloud
  Driver = "Amazon"
  HostName = "s3.amazonaws.com"
  BucketName = "BaculaVolumes"
  AccessKey = "BZIXAIS39DP9YNER5DFZ"
  SecretKey = "beesheeg7iTe0Gaexee7aedie4aWohfuewohGaa0"
  Protocol = HTTPS
  UriStyle = VirtualHost
  Truncate Cache = AfterUpload
  Upload = EachPart
  Region = "us-east-1"
  MaximumUploadBandwidth = 5MB/s
}
```

Amazon Central Europe Region (eu-central-1):


```
Cloud {
  Name = MyCloud
  Driver = "Amazon"
  HostName = "s3-eu-central-1.amazonaws.com"
  BucketName = "BaculaVolumes"
  AccessKey = "BZIXAIS39DP9YNER5DFZ"
  SecretKey = "beesheeg7iTe0Gaexee7aedie4aWohfuewohGaa0"
  Protocol = HTTPS
  UriStyle = VirtualHost
  Truncate Cache = AfterUpload
  Upload = EachPart
  Region = "eu-central-1"
  MaximumUploadBandwidth = 4MB/s
}
```

For Amazon Simple Storage Service (Amazon S3) Regions, 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 interface, use *UriStyle = Path* and set the *BlobEndpoint*:

```
Cloud {
  Name = CEPH_S3
  Driver = "Amazon"
  HostName = ceph.mydomain.lan
  BucketName = "CEPHBucket"
  AccessKey = "xxxXXXxxxx"
  SecretKey = "xxheeg7iTe0Gaexee7aedie4aWohfuewohxx0"
  Protocol = HTTPS
  Upload = EachPart
  UriStyle = Path            # Must be set for CEPH
  BlobEndpoint = "https://ceph.mydomain.lan"
}
```

9.2 Azure

```
Cloud {
  Name = MyCloud
  Driver = "Azure"
  HostName = "MyCloud"  #not used but needs to be specified
  BucketName = "baculaAzureContainerName"
  AccessKey = "baculaaccess"
  SecretKey = "/Csw1SECRETUmZkfQ=="
  Protocol = HTTPS
  UriStyle = Path
}
```


9.3 Google

Since the Google CLI installation already requests credentials, and buckets created with **gsutil** already specify the default localization, Bacula will detect them automatically.

Note: The only mandatory parameters are **Name**, **Driver** and the **BucketName**.

If you have configured **gcloud init** with the `HOME=/opt/bacula/etc/google` parameter, Bacula will search for Google credentials automatically in `/opt/bacula/etc/google`.

```
Cloud {
  Name = MyCloud
  Driver = "Google"
  BucketName = "MyBucket"
}
```

If your Google credentials are stored elsewhere, you can specify a custom **gcloud** configuration **path** through the **HostName** attribute:

```
Cloud {
  Name = MyCloud
  Driver = "Google"
  BucketName = "MyBucket"
  HostName = "/path/to/google-config/folder/"
}
```

9.4 Oracle

Since the OCI CLI installation requests credentials, and buckets created with **oci** already specify the default localization, Bacula will detect these automatically.

Note: The only mandatory parameters are **Name**, **Driver**, **BucketName** and **HostName**.

The HostName holds the oci config file location (see ocicliconfig).

If you configured oci into /opt/bacula/etc/oci/, you don't need to specify HostName:

```
Cloud {
  Name = MyCloud
  Driver = "Oracle" #mandatory
  BucketName = "MyBucket" #mandatory
}
```

You can specify a custom oci config **file** through the HostName attribute:

```
Cloud {
  Name = MyCloud
  Driver = "Oracle" #mandatory
  BucketName = "MyBucket" #mandatory
  HostName = "/path/to/oci/config"
}
```


9.5 Swift

Note: The only mandatory parameters are Name, Driver, BucketName, HostName, Protocol, AccessKey and SecretKey.

```
Cloud {
  Name = MyCloud
  Driver = "Swift"         # mandatory
  HostName = "MySwiftHost" # mandatory
  Protocol = "http"        # mandatory. Values: "http" or "https"
  BucketName = "MySwiftContainer" # mandatory
  AccessKey = "MySwiftUser" # mandatory
  SecretKey = "MySwiftPassword" # mandatory
  # optional directives
  Truncate Cache = AfterUpload
  Upload = EachPart
  MaximumUploadBandwidth = 5MB/s
}
```

9.6 Plugin directives

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

- **Name** The name of the Cloud resource, for instance **MyCloud** in some of the examples 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 (depending on the installed cloud driver): **Amazon**, **S3** (deprecated), **Azure**, **Google**, **Oracle**, **Swift** and **File** (**File** is used mostly for testing). **Amazon** is similar to **S3** but uses bacula-sd-cloud-aws-driver instead of bacula-sd-cloud-s3-driver. 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. This directive is not used with the Azure and Swift cloud drivers, but it needs to be specified with dummy data to satisfy the parser. For the Google driver this directive needs to be specified when gcloud has been installed in a custom location. For the Oracle driver this directive needs to be specified when the oci config file is installed in a custom location.
- **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. With Azure Storage it is generally referred to as Container and it can be created automatically by Bacula when it does not exist. With Google Cloud, the bucket must be created previously on the Cloud service. With Oracle Cloud, the bucket must be created previously in the OCI Console.

  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.

  This directive needs to be specified for Bacula Storage daemon compatibility but is not relevant with the Swift driver.

Note: When dealing with the S3 Plugin (Amazon or S3 drivers), confirm the `AccessKeyId value` provided by your S3 Cloud provider is compatible with the Amazon API as it doesn't allow some special characters.

- **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.

  This directive needs to be specified for Bacula Storage deamon compatibility but is not relevant with the Swift driver.

- **Protocol** The protocol defines the communication 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 uploaded 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.

- **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.

- **Maximum Concurrent Uploads** The default is 3, but by using this directive, you may set it to any value that fits your environment.

- **Maximum Concurrent Downloads** The default is 3, but by using this directive, you may set it to any value that fits your environment.

- **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"`

9.7 S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive directives

- **TransferPriority** (available with Bacula Enterprise 12.2.0 and later. S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive support is provided as a separate option in Amazon Glacier Instant Retrieval and Amazon Glacier Deep Archive). When restoring directly a part from Glacier, this directive indicates the rehydration priority level. Values can be **High**, **Medium** or **Low**. Default is **High**. Those values match respectively **Expedited**, **Standard** and **Bulk** transfers tiers within S3.
- **TransferRetention** (available with Bacula Enterprise 12.2.0 and later. S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive support is provided as a separate option in Amazon Glacier Instant Retrieval and Amazon Glacier Deep Archive). This directive indicates the time S3 should keep the rehydrated part online. The value should be at least 1 day. Default is 5 days.

9.8 Azure directives

- **BlobEndpoint** this directive can be used to specify a custom URL for Azure Cloud blob

  https://docs.microsoft.com/en-us/azure/storage/blobs/storage-custom-domain-name.
- **EndpointSuffix** use this directive to specify a custom URL postfix for Azure. Example: `EnbpointSuffix="core.chinacloudapi.cn"`

9.9 Amazon directives

- **BlobEndpoint** (available with Bacula Enterprise 16.0.6 and later) this directive can be used to specify a custom URL to Amazon S3 Cloud blob. Example: ``BlobEndpoint=”https://my.s3.endpoint”

Note: Starting with Bacula Enterprise 16.0.6, the **BlobEndpoint** directive needs to be set when Amazon driver is utilized with a non AWS cloud storage.

- **StorageClass** (available with Bacula Enterprise 14.0.5 and later) this directive can be used to specify the storageClass for all parts tranfered to the cloud, independently of the destination bucket class. Values can be **S3Standard**, **S3StandardIA**, **S3IntelligentTiering**, **S3OneZoneIA**, **S3GlacierInstantRetrieval**, **S3GlacierFlexibleRetrieval**, **S3GlacierDeepArchive**, **S3Rrs**.

10 Creating and Verifying your Cloud Account

11 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.

The following Cloud directives are ignored: **Bucket Name**, **Access Key**, **Secret Key**, **Protocol**, **Uri Style**. The directives **Truncate Cache** and **Upload** work on the local cache in the same manner as they would for a Cloud.

Host Name, specifies the local destination directory for the Cloud Volume files, and this **Host Name** must be different from the **Archive Device** name, or there will be a conflict between the local cache (in the **Archive Device** directory) and the destination Cloud Volumes (in the **Host Name** directory).

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.

12 Statistic Explained

We have information about the upload of cloud volume part files in either the job log or in the output of the bconsole "status storage" command when there is a cloud storage configured.

Job log example

```
20-Apr 16:31 bacula-sd JobId 27: Cloud Upload transfers:
20-Apr 16:31 bacula-sd JobId 27: cloudvolume-Vol-0001/part.1     state=done     size=401␣
↪B duration=8s
20-Apr 16:31 bacula-sd JobId 27: cloudvolume-Vol-0001/part.2     state=done     size=9.
↪999 MB duration=17s
20-Apr 16:31 bacula-sd JobId 27: cloudvolume-Vol-0001/part.3     state=done     size=9.
↪999 MB duration=17s
20-Apr 16:31 bacula-sd JobId 27: cloudvolume-Vol-0001/part.4     state=done     size=9.
↪999 MB duration=14s
```

** Example of a bconsole status storage command output**

```
Cloud transfer status:
   Uploads   (1.642 MB/s) (ETA 0 s) Queued=0 0 B, Waiting=0 0 B, Processing=0 0 B,␣
↪Done=53 524.7 MB, Failed=0 0 B
   Downloads (0 B/s) (ETA 0 s) Queued=0 0 B, Waiting=0 0 B, Processing=0 0 B, Done=0 0 B,
↪ Failed=0 0 B
```

We also have information in BWeb.

Duration value needs to be considered as the amount of time the cloud upload operation took. It can be for a single part file or for multiple part files as multiple part files can be uploaded into the cloud. The main goal of the duration value is to have an idea if something takes too long and it is possible that another process is affecting the cloud volumes upload speed.

Timestamp in the job log of each part file upload doesn't match exactly with the value of the part file in the remote cloud as this value is the timestamp of the part file creation in the bucket and they can be different.

Uploads value (XXX KB/s) means that the upload rate is XXX KB/s. It is not a median of all the values uploaded, but a rate of the last uploaded part file or even part files if they have been uploaded concurrently. So, the Uploads value is not an average nor cumulative value.

On the other hand, the **Queued**, **Waiting**, and the **Processing** values are computed for all part files which been uploaded, even from other jobs.

Then, the **Done** and **Failed** values are cumulative values since the Storage Daemon startup. As soon as the Storage Daemon is restarted, these values are list and are reset to zero.

ETA = Data to transfer/global speed, and ETA for each transfer is the same, but applied only to the transfer. Different ETAs are computed at a given time with the given resources allocated to the cloud transfer and the global network capabilities. It's done based on the last part transfer duration vs size. So it's as close to an "instant" transfer rate that it can be. The values can change over the time.

13 Limitations

- The support of Redhat 6 has been deprecated with Bacula Enterprise Cloud storage driver version 8.10.
- **MaximumVolumeBytes** and **MaximumPoolBytes** directives is not implemented in cloud volumes like in regular volumes. Other directives, like **MaximumVolumeParts**, **VolumeUseDuration** or **MaximumVolumeJobs** (with concurrent jobs =1) must be used to set a limit on these volumes. The **MaximumVolumeParts** can be used to limit the size of a cloud volume, used along with the **MaximumPartSize** directive. Setting **MaximumPartSize** will also help controlling the part size on the cloud.
- On restore, all cloud volume parts that are needed will be downloaded from the cloud into the local cache storage before being transferred to the FD. This means that if you have a Full backup of 20TB (for example) that needs to be restored, you will need to have at least 20TB of local cache storage available. Consider also, if there are backups running at the time of restore, or if there are local cloud volume parts that have not yet been truncated, there will be cloud storage space already in use so you will need to pay close attention before and during large restores from the cloud to be sure that the local cloud cache storage space is not filled to capacity.
- The `restart` command has limitations with plugins, as it initiates the Job from scratch rather than continuing it. Bacula determines whether a Job is restarted or continued, but using the `restart` command will result in a new Job.

14 Best Practices

- Set the **MaximumFileSize** to a value smaller than or equal to the **MaximumPartSize** as the **MaximumFileSize** defines the granularity of the restore chunk. Having the **MaximumFileSize** directive defined this way will allow Bacula to only download the required part file(s) to restore specific files and/or directories in the case of partial restores, thus reducing download costs.
- Have a reasonable value configured for **MaximumPartSize**. Smaller part sizes will reduce restore costs, but may require a small additional overhead to handle multiple parts. Also, some cloud providers limit the size of a single object to be uploaded. A part file is considered a single object and it will use a single upload operation. Thus you must set a value lower or equal than this limit for the **MaximumPartSize**, otherwise the upload of part files will fail. Please confirm with your cloud provider the maximum object size upload, if any.
- Regularly run `cloud upload` and `cloud truncate` **bconsole** commands. They can be scheduled in an Admin Job. Even when the Cloud resource is configured to automatically upload volume part files and/or truncate them from the local cache, connection issues can prevent some part files from being successfully uploaded and a local cache truncation may not occur for part files which have not yet been uploaded. It is recommended to retry the volume part file upload and/or local cache truncation in the case of failures by manually running the `cloud upload` and `cloud truncate` **bconsole** commands. You can also use a Bacula Admin Job that will retry the part file upload automatically on a regular basis.

  If the restore process takes a long time, it is recommended to temporarily deactivate the `cloud upload` (if `TruncateCache = yes`) and/or the `cloud truncate` commands to prevent essential parts needed for the restore from being truncated from the local cache.

  Please find an example for such an Admin Job that can be used to periodically trigger the cloud upload and the cloud truncate commands:

```
Job {
  Name = CloudUpload-adminjob
  Type = Admin
  Client = bacula-fd  # any client can be used
  Schedule = DailyCloudUpload
```

(continues on next page)

(continued from previous page)

```
  RunScript {
    RunsOnClient = No
    RunsWhen = Always
    Console = "cloud upload storage=<cloud_storage_name> allpools"
    Console = "cloud truncate storage=<cloud_storage_name> allpools"
  }
  Storage = <cloud_storage_name>
  Messages = Default
  Pool = Fake-pool
  FileSet = Fake-fileset
}
```

- After a restore, the downloaded part files are not truncated from the local cache even if **Truncate Cache** is configured in the Cloud resource (to truncate the local cache **AfterUpload** or **AtEndOfJob**). If you have your environment configured to truncate the local cache after the part is successfully uploaded or at the end of a job, we recommend to either manually truncate the local cache after the restore successfully finishes or to have it truncated by an Admin Job that periodically truncates the local cache (the Admin Job as recommended in the previous point).
- It is considered a best practice to set **MaximumConcurentJobs = 1** in all of the Cloud Device resources (**DeviceType = Cloud**) that are defined on the SD. This will guarantee that only one job is writing to the device at one time, so data belonging to different jobs will not be interleaved in the part files. When dealing with cloud backups, this helps to minimize the overall download costs by downloading only the required part files from specific backup jobs during a restore. Of course, you need to consider defining a larger number of Cloud Devices, appropriate to the maximum number of jobs you are expecting to run concurrently.
- Retention locked objects (terminology varies among providers) can be used, but to avoid failures when Volumes are recycled, it is strongly recommended to have Bacula's retention times set in a way that they expire only after the object is no longer in retention. This applies particularly to **VolumeRetention**, but **JobRetention** can indirectly affect Volume recycling too. If Bacula attempts to recycle a Volume that is still in locked state, the corresponding job will fail, and the offending Volume will be marked with status **Error**.

