Cloud

Bacula Systems Documentation

Contents

1 About Cloud Backup 2

2 Cloud Installation 19

3 Cloud Management 25

Contents

---

1 About Cloud Backup

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.

- *Cloud Volume Architecture*
- *Cloud Plugin Installation*
- *Cloud Plugin Functionality*
- *Commands, Resources, and Directives for Cloud Plugin*
- *Creating and Verifying your Cloud Account*
- *Limitations*
- *Best Practices*

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.

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.

1.1 Cloud Volume Architecture

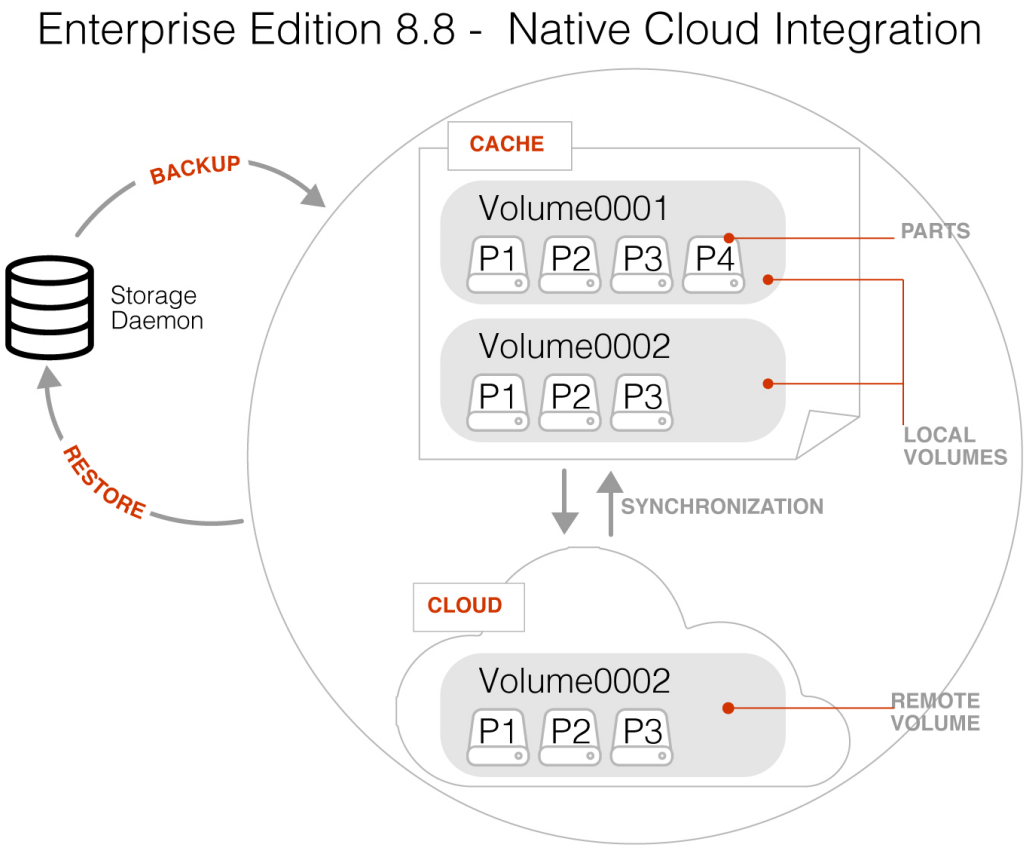

Fig. 1: Bacula Cloud Architecture

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, ...).

1.2 Cloud Plugin Installation

Cloud Installation

1.3 Cloud Plugin Functionality

The Cloud Plugin write the backup data in a local cache before the data is sent to the Cloud.

It is possible to keep the data in the local cache for some time, or to delete it as soon as the data is stored in the remote cloud.

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 the local cache is configured to be pruned/truncated automatically, the Archive Device file system 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.

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.

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`.

Compatibility with Other Bacula Functionality

A Cloud Volume stores the same data as any other Bacula Volume type (disk, tape, etc.). Thus, all the existing Bacula funcionality, with the exception of deduplication, is available with the Bacula Cloud Plugin drivers.

Client encrypted data, volumeencryption, compressed data, other plugins data, etc., can be stored in Cloud Volumes.

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.

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 data encryption features AFUDataEncryption.

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.

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.

Cloud Backup 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.

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/

Azure

With Azure redundancy might influence the price:

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

Google

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

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

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.

Swift

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

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

Cloud Accounts

Cloud Management

Cloud Compatibility Considerations

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

Azure

Only Microsoft Azure Storage has been validated.

1.4 Commands, Resources, and Directives for Cloud Plugin

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.

Cloud Resource

The **Cloud** resource has a number of directives that may be specified as shown in some examples later in this document.

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

Directives Used by Cloud Resource

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"`

Amazon Driver 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 storage class for all parts transferred to the cloud, independently of the destination bucket class. Values can be **S3Standard**, **S3StandardIA**, **S3IntelligentTiering**, **S3OneZoneIA**, **S3GlacierInstantRetrieval**, **S3GlacierFlexibleRetrieval**, **S3GlacierDeepArchive**, **S3Rrs**. Please, refer to https://aws.amazon.com/s3/storage-classes/ for more details.
- **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 **Low**. Those values match respectively **Expedited**, **Standard** and **Bulk** transfers tiers within S3.

Note: Please note that according to this https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects-retrieval-options.html, Deep Archive does not support **Expedited** retrieval, hence setting the **TransferPriority** to **High** will not work with retrievals from it.

- **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.

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"`
- **StorageClass** (available with Bacula Enterprise 18.0.4 and later) this directive can be used to specify the storage class (also known as storage tier) for all parts transferred to the cloud, independently of the destination bucket class. Values can be **WASHot**, **WASCool**, **WASCold**, **WASArchive**. Please, refer to https://learn.microsoft.com/en-us/azure/storage/blobs/access-tiers-overview for more details.

Google Directives

- **StorageClass** (available with Bacula Enterprise 18.0.4 and later) this directive can be used to specify the storage class for all parts transferred to the cloud, independently of the destination bucket class. Values can be **GoogleStandard**, **GoogleNearline**, **GoogleColdline**, **GoogleArchive**. Please, refer to https://cloud.google.com/storage/docs/storage-classes for more details.

Oracle Directives

- **StorageClass** (available with Bacula Enterprise 18.0.4 and later) this directive can be used to specify the storage class for all parts transferred to the cloud, independently of the destination bucket class. Values can be **OracleStandard**, **OracleInfrequentAccess**, **OracleArchive**. Please, refer to https://docs.oracle.com/en-us/iaas/Content/Object/Concepts/understandingstoragetiers.htm for more details.

Cloud Resource Examples

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.

Amazon Cloud Storage Plugin Authentication Additions

Use of Amazon `/opt/bacula/working/.aws/credentials` to specify the SecretKey and the AccessKey:

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

Note that the credentials file needs to be placed within the home directory of the user under which the Bacula Storage Daemon operates. By default, the user is **bacula** and the home directory is `/opt/bacula/working`. If that is not the case, the credentials file needs to be placed accordingly.

This configuration method is applicable when there is a need to specify the AWS access key ID and secret key outside the Bacula configuration (e.g., the values need to be periodically modified without the restart of the Storage Daemon) or AWS session tokens (temporary security credentials) need to be utilized. In such instances, the actual values for the AWS access key ID, secret key, and the current AWS session token can be set (and periodically reset, using an external program) in the configuration file `.aws/credentials`, and formatted as shown below.

```
[root@bacula-102 .aws]# cat /opt/bacula/working/.aws/credentials
[default]
aws_access_key_id = <snip>
aws_secret_access_key = <snip>
aws_session_token = <snip>
```

Additionally, the same configuration file can be utilized to specify other options related to the AWS CLI, if required.

S3 Object Storages using CEPH Interface

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
```

(continues on next page)

(continued from previous page)

```
  UriStyle = Path            # Must be set for CEPH
  BlobEndpoint = "https://ceph.mydomain.lan"
}
```

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
}
```

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/"
}
```


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"
}
```

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
}
```


File Driver

As already mentioned, it is possible to use 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 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.

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).

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.

Storage Daemon Cloud 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.

Storage Daemon Cloud Device Directives

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

Cloud Device Example

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

```
Device {
  Name = CloudAmazon-dev0
  Device Type = Cloud
  Cloud = MyCloud
  Archive Device = /opt/bacula/backups
```

(continues on next page)

(continued from previous page)

```
  Maximum Part Size = 10 MB
  Maximum File Size = 10 MB
  Media Type = CloudType
  LabelMedia = yes
  Random Access = Yes;
  AutomaticMount = yes
  RemovableMedia = no
  AlwaysOpen = no
}
```

Note: This is an example of a single device. Usually, you should use an Autochanger having many devices for an optimal performance with concurrent jobs.

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
```

Commands Used with Cloud Plugin

Using bconsole, the *cloud* command can be used to manage the cloud volumes and local cache.

See Bconsole Cloud Command.

Status Storage and 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  ␣
```

(continues on next page)

(continued from previous page)

```
↪  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.

1.5 Creating and Verifying your Cloud Account

Cloud Management

1.6 Limitations

- The support of RHEL 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.

1.7 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
  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**.

2 Cloud Installation

2.1 Installation of the Cloud S3/Amazon Plugin

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. The S3 and the Amazon driver are part of the same package named *bacula-enterprise-cloud-storage-s3*.

Note: The difference between S3 and Amazon Drivers is that S3 uses libs3, while Amazon uses AWS CLI API. The newest solution is the Amazon driver based on Amazon native library, while the S3 driver is based on unofficial S3 libraries. Amazon Driver is recommended as it is continuously supported and maintained, unlike S3 Driver.

AWS CLI is a requirement for the Cloud S3/Amazon Plugin. Refer to the AWS Documentation: https://aws.amazon.com/

Both drivers S3 and Amazon drivers backup data to the cloud and are installed on the Storage Daemon. The Cloud S3 plugin backing up data from S3 cloud objects to the Storage Daemon can be found here.

On the Bacula Storage Daemon that you want to connect to your cloud storage, extend the repository file for your package manager to contain a section for the Cloud plugin. For example in RHEL, `/etc/yum.repos.d/bacula.repo`:

```
[Bacula]
name=Bacula Enterprise
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/bin/@version@/
↪rhel9-64/
enabled=1
protect=0
gpgcheck=0

[Bacula EnterpriseCloudPlugin]
name=Bacula Enterprise Cloud Plugin
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/cloud-s3/
↪@version@/rhel9-64/
enabled=1
protect=0
gpgcheck=0
```

or in Debian Bookworm, `/etc/apt/sources.list.d/bacula.list`:

```
#Bacula Enterprise deb
https://baculasystems.com/dl/@customer-id@/debs/bin/@version@/bookworm-64/
bookworm main deb
https://baculasystems.com/dl/@customer-id@/debs/cloud-s3/@version@/bookworm-
↪64/
bookworm cloud-s3
```

Then perform a `yum update` or `apt-get update`, and after that install the package *bacula-enterprise-cloud-storage-s3*.

Installation of the S3/Amazon Glacier plugin (Deprecated)

Important: This section is deprecated. It is recommended to use the "Amazon Driver" for the Glacier feature which does not need any additional package or plugin.

Note: Prerequisite: *bacula-enterprise-cloud-storage-glacier* requires *bacula-enterprise-cloud-storage-s3* to be installed first.

Additionally, if you also purchased the Glacier plugin (available with Bacula Enterprise 12.2.0 and later), extend the repository file for your package manager to contain a section for the Glacier plugin. For example in RHEL, `/etc/yum.repos.d/bacula.repo`:

```
name=Bacula Enterprise
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/bin/@version@/
↪rhel9-64/
enabled=1
protect=0
gpgcheck=0

[Bacula EnterpriseCloudPlugin] name=Bacula Enterprise Cloud Plugin
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/cloud-s3/
↪@version@/rhel9-64/
enabled=1
protect=0
gpgcheck=0

[Bacula EnterpriseGlacierPlugin] name=Bacula Enterprise Glacier Plugin
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/cloud-glacier/
↪@version@/rhel9-64/
enabled=1
protect=0
gpgcheck=0
```

or in Debian Bookworm, `/etc/apt/sources.list.d/bacula.list`:

```
#Bacula Enterprise deb
https://baculasystems.com/dl/@customer-id@/debs/bin/@version@/bookworm-64/␣
↪bookworm main deb
https://baculasystems.com/dl/@customer-id@/debs/cloud-s3/@version@/bookworm-
↪64/ bookworm cloud-s3 deb
https://baculasystems.com/dl/@customer-id@/debs/cloud-glacier/@version@/
↪bookworm-64/ bookworm cloud-glacier
```

Then perform a `yum update` or `apt-get update`, and after that the package *bacula-enterprise-cloud-storage-s3* and, optionally, *bacula-enterprise-cloud-storage-glacier* can be installed with `yum install` or `apt-get install`.

If you prefer to manually install the packages, you can also download them from your download area, and use one of the low level package manager tools (`rpm` or `dpkg`) to do the plugin installation.

2.2 Installation of the Cloud Azure Plugin

Installation of the Azure CLI

Previous to the plugin installation, the corresponding distribution package of the CLI must be installed.

For RHEL, refer to:

- https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-yum?view=azure-cli-latest

For Debian or Ubuntu, refer to:

- https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-apt?view=azure-cli-latest

For other platforms, refer to:

- https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest

Installation of the plugin

On the Bacula Storage Daemon that you want to connect to your cloud storage, extend the repository file for your package manager to contain a section for the Cloud plugin.

For example in RHEL, `/etc/yum.repos.d/bacula.repo`:

```
[Bacula]
name=Bacula Enterprise
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/bin/@version@/
↪rhel9-64/
enabled=1
protect=0
gpgcheck=0

[Bacula EnterpriseCloudPlugin]
name=Bacula Enterprise Cloud Plugin
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/cloud-azure/
↪@version@/rhel9-64/
enabled=1
protect=0
gpgcheck=0
```

or in Debian Bookworm, `/etc/apt/sources.list.d/bacula.list`:

```
#Bacula Enterprise
deb https://baculasystems.com/dl/@customer-id@/debs/bin/@version@/bookworm-64/
↪ bookworm main
deb https://baculasystems.com/dl/@customer-id@/debs/cloud-azure/@version@/
↪bookworm-64/ bookworm cloud-azure
```

Then perform a `yum update` or `apt-get update`, and after that the package *bacula-enterprise-cloud-storage-azure* can be installed with `yum install` or `apt-get install`.

If you prefer to manually install the packages, you can also download them from your download area, and use one of the low level package manager tools (`rpm` or `dpkg`) to do the plugin installation.

2.3 Installation of the Cloud Google Plugin

On the Bacula Storage Daemon that you want to connect to your cloud storage, extend the repository file for your package manager to contain a section for the Cloud plugin. For example in RHEL, `/etc/yum.repos.d/bacula.repo`:

```
[Bacula]
name=Bacula Enterprise
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/bin/@version@/
↪rhel9-64/
enabled=1
protect=0
gpgcheck=0

[Bacula EnterpriseCloudPlugin]
name=Bacula Enterprise Cloud Plugin
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/cloud-google/
↪@version@/rhel9-64/
enabled=1
protect=0
gpgcheck=0
```

or in Debian Bookworm, `/etc/apt/sources.list.d/bacula.list`

```
#Bacula Enterprise
deb https://baculasystems.com/dl/@customer-id@/debs/bin/@version@/bookworm-64/
↪ bookworm main
deb https://baculasystems.com/dl/@customer-id@/debs/cloud-google/@version@/
↪bookworm-64/ bookworm cloud-google
```

Then perform a `yum update` or `apt-get update`, and after that the package *bacula-enterprise-cloud-storage-google* can be installed with `yum install` or `apt-get install`.

If you prefer to manually install the packages, you can also download them from your download area, and use one of the low level package manager tools (`rpm` or `dpkg`) to do the plugin installation.

2.4 Installation of the Cloud Oracle Plugin

On the Bacula Storage Daemon that you want to connect to your cloud storage, extend the repository file for your package manager to contain a section for the Cloud plugin. For example in RHEL, `/etc/yum.repos.d/bacula.repo`:

```
[Bacula]
name=Bacula Enterprise
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/bin/@version@/
↪rhel9-64/
enabled=1
protect=0
gpgcheck=0

[Bacula EnterpriseCloudPlugin]
name=Bacula Enterprise Cloud Plugin
baseurl=https://www.baculasystems.com/dl/@customer-id@/rpms/cloud-oracle/
```

(continues on next page)

(continued from previous page)

```
↪@version@/rhel9-64/
enabled=1
protect=0
gpgcheck=0
```

or in Debian Bookworm, `/etc/apt/sources.list.d/bacula.list`:

```
#Bacula Enterprise
deb https://baculasystems.com/dl/@customer-id@/debs/bin/@version@/bookworm-64/
↪ bookworm main
deb https://baculasystems.com/dl/@customer-id@/debs/cloud-oracle/@version@/
↪bookworm-64/ bookworm cloud-oracle
```

Then perform a `yum update` or `apt-get update`, and after that the package *bacula-enterprise-cloud-storage-oracle* can be installed with `yum install` or `apt-get install`.

If you prefer to manually install the packages, you can also download them from your download area, and use one of the low level package manager tools (`rpm` or `dpkg`) to do the plugin installation.

2.5 Installation of the Cloud Swift Plugin

On the Bacula Storage Daemon that you want to connect to your cloud storage, extend the repository file for your package manager to contain a section for the Cloud plugin. For example in RHEL, `/etc/yum.repos.d/bacula.repo`:

```
[Bacula]
name=Bacula Enterprise
baseurl=https://www.baculasystems.com/dl/@customer-string@/rpms/bin/@version@/
↪rhel9-64/
enabled=1
protect=0
gpgcheck=0

[Bacula EnterpriseCloudPlugin]
name=Bacula Enterprise Cloud Plugin
baseurl=https://www.baculasystems.com/dl/@customer-string@/rpms/cloud-swift/
↪@version@/rhel9-64/
enabled=1
protect=0
gpgcheck=0
```

or in Debian Jessie, `/etc/apt/sources.list.d/bacula.list`:

```
#Bacula Enterprise
deb https://www.baculasystems.com/dl/@customer-string@/debs/bin/@version@/
↪jessie-64/ jessie main
deb https://www.baculasystems.com/dl/@customer-string@/debs/cloud/@version@/
↪jessie-64/ jessie cloud
```

Then perform a `yum update` or `apt-get update`, and after that the package *bacula-enterprise-cloud-storage* can be installed with `yum install` or `apt-get install`. On Oracle Linux/AlmaLinux/Rocky Linux, the EPEL package needs to be installed to avoid a dependency problem with *libxml++*.

If you prefer to manually install the packages, you can also download them from your download area, and use one of the low level package manager tools (`rpm` or `dpkg`) to do the plugin installation.

3 Cloud Management

3.1 S3/Amazon Account Management

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.

Create an Account

Amazon offers high-level S3 commands with the AWS Command Line Interface. The tool (aws) is mandatory for the Bacula Amazon driver to work. It can also be used to verify manually 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.

Go to the following link and create an S3/Amazon account.

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

Then you might want to use the S3/Amazon tutorial. Note that some of the information in the tutorial is repeated below for your convenience.

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

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. Amazon recommends to have at least Python version 2.7.

Make sure that Python is installed:

```
[root@localsd1 ~]# python --version
Python 3.10.6
[root@localsd1 ~]#
```

The pip tool was not 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
```

Define user and export credentials

This step is not mandatory if you plan to use aws only thru the Bacula Amazon cloud driver, since Bacula will automatically use the Bacula Cloud resource credentials, but it is recommended for using aws as a standalone command line interface.

To be able to access your S3/Amazon 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)

Create a Bucket in S3/Amazon

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
```

Copy and sync files

You can list your S3/Amazon 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.

S3/Amazon Glacier Instant/Flexible Retrieval (Deprecated)

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. The S3 and the Amazon driver are part of the same package named *bacula-enterprise-cloud-storage-s3*.

Another effective strategy is to use Bacula direct restoration from S3/Amazon Glacier Instant/Flexible Retrieval (available with Bacula Enterprise 12.2.0 and later) or S3/Amazon Glacier Deep Archive (available with Bacula Enterprise 14.0.0 and later). Configure the bucket lifecycle to automatically handle transition to S3 Glacier Instant/Flexible Retrieval or S3/Amazon Glacier Deep Archive after backup:

- https://docs.aws.amazon.com/en_pv/AmazonS3/latest/user-guide/create-lifecycle.html

and use the **TransferPriority** and **TransferRetention** Cloud directives to configure rehydration before restoration. See *Amazon Driver Directives* for details. Bacula will monitor the rehydration process until its completion and proceed normally with parts download and restoration afterwards.

S3/Amazon Glacier Deep Archive (Deprecated)

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. The S3 and the Amazon driver are part of the same package named *bacula-enterprise-cloud-storage-s3*.

Bacula direct restoration can also be used to restore from S3/Amazon Glacier Deep Archive (available with Bacula Enterprise 12.2.0 and later. S3/Amazon Glacier Instant Retrieval and S3/Amazon Glacier Deep Archive support is provided as a separate option). Configure the bucket lifecycle to automatically handle transition to S3/Amazon Glacier Deep Archive after backup:

- https://docs.aws.amazon.com/en_pv/AmazonS3/latest/user-guide/create-lifecycle.html

and use the same **TransferPriority** and **TransferRetention** Cloud directives as for Glacier to configure rehydration before restoration. see *Amazon Driver Directives* for details. Bacula will monitor the rehydration process until its completion and proceed normally with parts download and restoration afterwards.

Time coordination

It seems that S3/Amazon 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

S3 Object Lock

If **Object Lock** is configured on your target bucket, do not use the **S3 driver** to backup to it, but rather the more recent **Amazon driver**, by changing the Driver keyword of the Cloud resource in your SD configuration from "S3" to "Amazon".

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.

Note: To read more about security and data immutability, click *here*.

3.2 Azure Account Management

Create a Microsoft Azure Account

Please go to the following link and create an Microsoft Azure account. Trials are available.

- https://azure.microsoft.com

Then follow the following link to login.

- https://portal.azure.com

Configure your Azure Blob Storage

You have to do a few configuration steps to prepare your Azure portal for backup with Bacula Enterprise. The Bacula Systems Support Team can provide screenshots for the below steps upon request.

1. Login to your Azure portal and create a new *Storage account* that you either assign to an existing resource group (or you can also create a new one on the fly).
2. Give it a name; Must be lowercase letters and numbers only, and must be unique in all of Azure In the "Account kind" field select "Blob storage" For "Resource Group", click "Use existing", then choose the Resource Group from the drop-down Click on "Create" to save. This may take a minute to deploy.
3. Select the new storage account in your *Storage accounts overview* page and navigate to *Settings* and then *Access keys* to retrieve the **Secret Key** that you will need to specify in your Bacula Storage Daemon configuration. The *Storage account Name* will be the **Access Key** in your Cloud resource in `bacula-sd.conf`.
4. Now click on *Blob service* → *Containers* to create a new container with public access level *Blob (anonymous read access for blobs only)*. This will be the **BucketName** that we use in the Bacula Cloud resource.
5. Confirm that the Settings in the Azure portal which were just created/edited match the Bacula Cloud resource config file.
6. Stop the bacula-sd daemon: `systemctl stop bacula-sd`
7. Test the SD config syntax: `/opt/bacula/bin/bacula-sd -u bacula -t`
8. Start the bacula-sd daemon: `systemctl start bacula-sd`
9. Run a backup job to the Azure storage.

3.3 GCP Account Management

The Python application `gsutil` lets you access Cloud Storage from the command line. It is part of the Google Cloud SDK which installation is platform dependent.

Please, browse to this location:

- https://cloud.google.com/sdk/docs/

Choose your platform and follow the google-cloud-sdk installation steps:

Debian/Ubuntu. Follow the installation steps up to `apt-get install google-cloud-sdk`. Optional steps are not required.

RedHat. Follow the installation steps up to `yum install google-cloud-sdk`. Optional steps are not required.

Initialize Cloud Platform access

In order to authorize the bacula user to access Google Cloud Platform through gsutil, run the `setup_google_cloud_cli` script located in `/opt/bacula/scripts` as root to guide you through this steps.

You will be asked for your account information and to select or create a *project*. These project and credentials will later be used by the Plugin.

Alternatively, you can launch the following command as root:

```
sudo -u bacula HOME=/opt/bacula/etc/google gcloud init
```

If you plan to use a service account to authenticate, use the following command using the JSON file that contains your service account key:

```
sudo -u bacula HOME=/opt/bacula/etc/google gcloud auth activate-service-
↪account --project=<project_id> --key-file=/path-to-json-file/service-
↪account-xxxxxxxxx.json
```

The Google credentials will be stored inside `/opt/bacula/etc/google` and will belong to the unix user "bacula". To use another location, it will be required to adapt the Cloud resource definition.

Create a Bucket in Google Cloud

There are at least 2 ways you can use to create a Bucket in your selected project. We recommend to use the web console since it's giving you usefull information on fields and pricing.

Use the web console

Log into the web console:

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

From the top right menu, select **Storage**, then click **Create Bucket**. Specify the *name*, *class* and *location* of the bucket with the help of context menus. Click **Create**.

Use the command line mb (make bucket) gsutil command

From a terminal, type:

```
gsutil mb [-c class] [-l location] gs://<some-bucket>
```

- *class*. Optional. The storage **class** you choose for this bucket (see https://cloud.google.com/storage/docs/storage-classes for details).
- *location*. Optional. The storage **region** location you choose for this bucket (see https://cloud.google.com/storage/docs/bucket-locations for details).
- <*some-bucket*> is your bucket **name**.

The storage *class* influences the storage price. Depending on your backup strategy, **nearline** or even **coldline** might be the most appropriated classes.

Configure objects to be retention locked

This configuration is optional

Retention locking of objects can provide additional security against premature loss of backup data.

See https://cloud.google.com/storage/docs/bucket-lock for details.

For an existing bucket, use

```
gsutil retention set <seconds>s gs://<some-bucket>/
```

- <*seconds*> is the number of seconds an object is going to be locked.
- <*some-bucket*> is the name of an existing bucket used to store backups in, probably one just created.

To verify retention lock time, use a command such as

```
gsutil retention get gs://<some-bucket>/
```

3.4 Oracle Cloud Account Management

Cloud Account

To configure your Bacula Storage daemon with Oracle Cloud Infrastructure, you need to create an account through the Oracle Cloud portal.

- https://cloud.oracle.com/

Note that the registration process requires a Default Data Region. Bacula Storage deamon stores to the Oracle Cloud Object Storage. Make sure you that you select a region that supports it. See the following link for details on regions

- https://cloud.oracle.com/data-regions/

Once you've entered the required information, you should be granted access to the Oracle Services Dashboard and receive a confirmation e-mail containing a link to connect the Oracle Services Dashboard.

Obtain Oracle Cloud Infrastructure Console URL

For the next operations, you'll need to access your Oracle Cloud Infrastructure Console. Follow the next steps to retrieve the OCI Console URL. (If you already know your OCI console URL, you can skip this section.)

- Follow the link provided in the registration e-mail to reach the **Services Dashboard**.
- In the top right corner of the Dashboard, select you **account Icon** and choose **My Admin Accounts** from the drop down menu.
- From the **Administrative Accounts** list Copy the URL associated to **Compute (OCI) Users**. That the URL is region-based. It should start like this: `https://console.<region>.oraclecloud.com/...`

Keep track of this URL, it's your main access point to the Oracle Storage.

Create a Bucket in Oracle Cloud

Before starting using Plugin, you need to create a Bucket in your Oracle Cloud Infrastructure Console. Bacula will create all its backup volumes in this Bucket.

1. Browse to your OCI Console
2. From the top left Menu, select **Object Storage**
3. Click **Create Bucket**
4. Fill the different creation fields, specifically:

- **Bucket Name**: Will be used as specified here within the Bacula Cloud Resource
- **Storage Tier**: The bucket Storage Tier influences the storage price and cannot be changed afterward: https://cloud.oracle.com/storage/archive-storage/faq

  Standard will provide instant access to the bucket objects, while with **Archive** you'll have to **manually restore your objects with the OCI client before Bacula can restore them**: https://docs.cloud.oracle.com/iaas/tools/oci-cli/latest/oci_cli_docs/cmdref/os/object/restore.html
- **Tags**: Tags can be attached to your bucket to organize your tenancy. Bacula will not consider them.

Retrieve the OCID keys

These keys are required to complete OCI Command Line Interface installation.

Tenancy's OCID

The tenancy OCID is in the Console in the Tenancy Details page: From the OCI Console, open the **User menu** in the top right corner and click The **tenancy OCID** is shown under Tenancy Information.

User's OCID

The user OCID is in the Console in the User Settings page: From the OCI Console, open the **User menu** in the top right corner and click The **user OCID** is shown under User Settings.

Install the OCI Command Line Interface tool

The OCI Command Line Interface (CLI) is a Python application that allows access to Oracle Cloud Storage from the command line. Its installation is platform dependent as described in:

- https://docs.cloud.oracle.com/iaas/Content/API/SDKDocs/cliinstall.htm

Since you should be running on a Unix-like platform, run the following bash command as root:

The installed binaries must be on the path so it's important to modify the default location. We recommand installing in **/usr/local/bin**

- *install location* should be changed to **/usr/local/lib/oracle-cli**
- *OCI executable location* should be changed to **/usr/local/bin**
- *OCI scripts location* should be changed to **/usr/local/bin/**
- later on, you may be asked to modify the path. Answer no.

Eventually, you should get prompted with *Installation successful*.

Setup the OCI CLI config

Once the OCI CLI has been installed, you can run the setup_oracle_cloud_cli script located in */opt/bacula/scripts* as root to guide you through this step.

You'll be asked to provide the following information:

- *config location*. Replace with **/opt/bacula/etc/oci/config**
- *user OCID*.
- *tenancy OCID*.
- *region:*
- *RSA key pair*. The script can generate it for you or you can provide your own: https://docs.cloud.oracle.com/iaas/Content/API/Concepts/apisigningkey.htm.

  Make sure to locate them in **/opt/bacula/etc/oci**.

Alternatively, you can manually run

```
sudo -u bacula /usr/local/bin/oci setup config
```

Upload public pem key to the OCI Console

Next the public key must be uploaded to the OCI Console. The public key is named *oci_api_key_public.pem* and is located in **/opt/bacula/etc/oci**. View the details for the user who will be calling the API with the key pair: Click **Identity**, **Users**, and then select the user from the list. Click **Add Public Key**. Paste the contents of the PEM public key in the dialog box and click Add.

Retrieve the Bucket namespace and the compartment id

From the OCI Console, open the navigation menu in the top left corner. Select Object Storage, then the Bucket used for Bacula backup. In the Bucket Information, note the **namespace** and copy the **compartment id**.

Compartment permissions

It may be necessary to set advanced policies such as below in order to allow access to the bucket:

```
allow group GROUPNAME to manage buckets in compartment COMPARTMENTNAME

allow group GROUPNAME to use buckets in compartment COMPARTMENTNAME

allow group GROUPNAME to manage objects in compartment COMPARTMENTNAME
```


Create the resource control file

Edit the config file (should be */opt/bacula/etc/oci/config*) and add the following section

```
[CLI]
# globally scoped default for all operations with a -compartment-id parameter
compartment-id = <compartment-id>
# globally scoped default for all operations with a -namespace parameter
namespace = <namespace>

[OCI_CLI_SETTINGS]
default_profile=CLI
```

Replace <*compartment-id*> and <*namespace*> with the values you retrieved in *Retrieve the Bucket namespace and the compartment id*. Save the config file.

Test OCI CLI

In a terminal, type on the same line:

```
sudo -u bacula /usr/local/bin/oci os object list -bn MyBucket  --cli-rc-file /
↪opt/bacula/etc/oci/config --config-file /opt/bacula/etc/oci/config
```

Where *MyBucket* is the name of the bucket created in *Create a Bucket in Oracle Cloud*.

You should get no error message and the following reply:

```
"prefixes": []
```

3.5 Swift Account Management

Install the python-swiftclient Command Line Interface tool

Openstack provides a Python application that lets you access their Cloud Storage from the command line. It is part of the Swift Stack which installation is platform dependent as described here: https://pypi.org/project/python-swiftclient/ .

Choose your platform and follow the installation steps accordingly.

