Amazon RDS Plugin

Bacula Systems Documentation

Contents

1 Scope 3

2 Features 3

3 Architecture 5

4 Installation 5

5 Configuration 10

6 Best Practices 24

7 Operations 26

8 Limitations 49

9 Troubleshooting 50

Contents

Note

You can download this article as a PDF

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The following article aims at presenting the reader with information about the **Bacula Enterprise Amazon RDS Plugin**.

Amazon Relational Database Service (Amazon RDS) is a managed database service provided by Amazon Web Services (AWS) that simplifies the setup, operation, and scaling of relational databases in the cloud. RDS supports several popular database engines, including PostgreSQL, MySQL, MariaDB, Oracle, and Microsoft SQL Server. The service can be deployed via managed Amazon EC2 instances, or in fully managed format with Amazon Aurora. It includes limited automated backups in the AWS Cloud, software patching, and database monitoring, as well as automatic failover and replication to enhance availability and reliability. Additionally, RDS provides scalable storage and compute resources, enabling users to adjust capacity as needed without experiencing downtime.

The Bacula Enterprise Amazon RDS Plugin provides backup and restore of database instances (whether standalone or clustered) running in Amazon RDS, managing the whole snapshot livecycle without lim-

itations. It also offers the optional feature of automatically exporting data to an S3 bucket in the AWS cloud and to extract it to send it to any other storage system managed by Bacula Enterprise, whether in the cloud or on-premise (including any type of disk, tape or alternative cloud). The entire process operates in an agent-less manner, providing a high degree of flexibility for executing various operations and delivering superior performance when the underlying network is suitable for the backup target. Once the data from a database has been protected by Bacula Enterprise, it can be restored to a new Cloud RDS instance or to a on-premise database engine, allowing the user to use this plugin as an export tool if there is a need to migrate data to a different engine.

Through subchapters, more in-depth information can be found about the topics below.

1 Scope

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

Bacula Enterprise Amazon RDS Plugin currently supports backup and restore database instances or clusters deployed via the Amazon RDS service, provided there is a connection to the appropriate Amazon RDS API.

When export and download service is used, access to the Amazon S3 service and API is also required.

This plugin has been available since **Bacula Enterprise 18.2**, and needs to be deployed in a Linux host.

2 Features

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The main feature of the **Bacula Enterprise Amazon RDS Plugin** is its capability to provide backup and restore for Amazon RDS instances or clusters.

The primary backup layer manages the snapshots of the database instances or clusters, executing them from Bacula Enterprise according to a user-defined schedule and maintaining the desired number of snapshots. On top of this, the plugin can export data to an S3 bucket and to backup the contents of the export operation, resulting in backup jobs that include one Apache Parquet archive for each table, along with several metadata files pertaining to the database and the snapshot. Finally, the plugin is able to manage the complete livecycle of the snapshots created and the exported data. All elements can be retained or cleaned up in a flexible, governed by the configuration settings of Bacula Enterprise.

Apache Parquet is an open source, column-oriented data file format designed for efficient data storage and retrieval. It provides high performance compression and encoding schemes to handle complex data

in bulk. For further details, refer to: - https://parquet.apache.org/

Apache Parquet is an independent database format, and the Bacula Enterprise Amazon RDS Plugin offers the ability to restore protected files to a PostgreSQL or MySQL database running on the same or a completely different underlying architecture. This includes both cloud and on-premise solutions, and it is not required that the source database is the same engine as the destination database, which means this plugin can be used as a migration tool for the protected data.

2.1 General Features

Below, there is a list of general features this plugin offers:

- Backup and restore of Amazon RDS instances
- Backup and restore of Amazon RDS clusters
- Amazon AWS API-based backups
- Automatic multi-threaded download or upload operations
- Network resiliency mechanisms
- Instances and volume discovery capabilities
- List instances and snapshots through query or list commands
- Auto-generation capabilities when combined with the Scan Plugin
- Restore data to Amazon RDS from the previously generated snapshots in the cloud
  - To the original location (region, availability zone, etc.)
  - To a different location (region, availability zone, etc.)
  - With the same original instance or cluster configuration
  - With a different instance or cluster configuration
  - Point-in-Time recovery
- Restore data from snapshots done during the backup or from any existing identifier
- Restore data to a running database engine from the data stored in Apache Parquet format
- Full & Incremental backup levels
  - Snapshots are Incremental backups by nature in Amazon RDS, as they are based in a non-visible EBS layer
  - Exports to S3 include always all data from the selected tables
- Advanced instance selection for backup, filtering by any parameter
- Ability to select whether to export data to S3
- Ability to select whether to download data from S3
- Ability to select if all tables or just part of them should be exported to S3 using Apache Parquet format
- Automatic snapshot cleanup before and after backups based on the specified retention
- Configurable automatic S3 bucket cleanup after backups

3 Architecture

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

Bacula Enterprise Amazon RDS Plugin is a Bacula File Daemon plugin that utilizes the Amazon Web Services APIs to interact with the Amazon Cloud (launching operations in the Amazon Cloud, retrieving data from it and feeding it at restore time). The plugin runs a Java Daemon which employs the official Amazon AWS SDK version 2.x built by Amazon.

All the information is obtained through secure and encrypted HTTPS queries to AWS from the File Daemon (and via the aforementioned Java Daemon), where the plugin is installed. The specific URLs will depend on the region and the operation being performed.

To get more information about AWS implied APIs, visit:

- https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/ProgrammingGuide.html
- https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html

The metadata for each backed up element is stored in Bacula in JSON format, which encompasses database instances, cluster instances and snapshots of both elements. The files related to the database exports are stored in S3 and are temporarily downloaded to the host where the plugin operates, prior to being sent to the Storage Daemon.

The backup and restore processes use different parallelization techniques in order to maximize performance and overcome latency times when communicating with AWS Parallelization. Additionally, the plugin also supports the parallelization of multiple backup jobs.

Below, there is a simplified vision of the architecture of this plugin within a generic **Bacula Enterprise** deployment:

4 Installation

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

This article describes how to install Bacula Enterprise Amazon RDS Plugin.

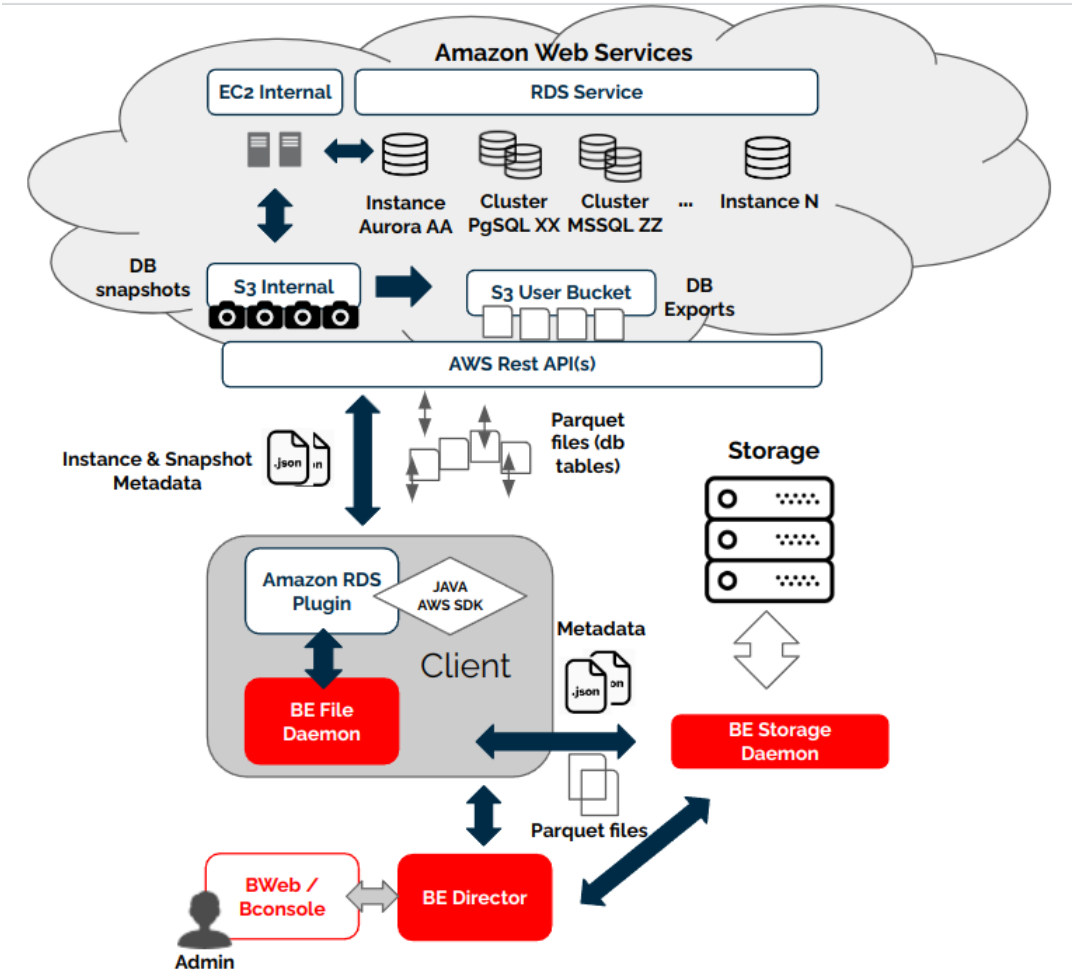

Fig. 1: Amazon RDS Plugin Architecture

4.1 Prerequisites

- The Bacula File Daemon and the Amazon RDS Plugin need to be installed on the host that will connect to the AWS Cloud.
- The plugin must be deployed in a host running Linux. It is possible to use any of the supported **Linux** distributions of Bacula Enterprise, including RHEL, Debian, Ubuntu or Suse Linux Enterprise Server as some examples.
- The plugin works via a Java daemon, therefore Java needs to be installed into the host using a JRE or JDK package (openjdk-11-jre for example). The Java environment should be version 11 or higher, and the Java binary must be accessible in the system PATH.
- The memory and computational requirements can vary significantly based on the configuration and usage of the plugin, such as parallelization and data size for backup etc. It is recommended to have a minimum of **4GB RAM** on the server where the File Daemon runs. By default, each job may use up to 512Mb of RAM in demanding scenarios, although it is typically less. In certain cases, this could be higher. Memory limits can be adjusted (see: Out of memory).
- Amazon Web Services REST APIs are used to perform all plugin operations. Therefore, they must be accessible through HTTPS from the host where Bacula FD and the plugin will be deployed.
- In order to fetch data, an access key comprising a key and secret is used to connect to AWS. Proper RDS and S3 permissions need to be associated to that access key. Details about how to configure such access key are given in the next sections.

4.2 Installation Methods

- *Amazon RDS Plugin Installation with BIM* (recommended)
- *Amazon RDS Plugin Installation with Package Managers*
- *Amazon RDS Plugin Manual Installation*

Amazon RDS Plugin Installation with BIM

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The recommended way to install any Bacula component, including any daemon and any plugin, is using the Bacula Installation Manager (BIM).

Steps

1. Install File Daemon.
2. Select the amazon-rds key in the plugin selection step.

Result

Amazon RDS Plugin is installed. Click here for more details.

For more information, visit Linux: Bacula Enterprise Installation with BIM.

Amazon RDS Plugin Installation with Package Managers

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

Another way to install this plugin involves using the package manager of the used distribution.

In this instance, Debian Bullseye serves as an example of a base operative system. The process will be very similar in any version of Debian or any other Debian-based distribution (e.g., Ubuntu). In the case of using RPM-based distributions, like Red Hat Linux, Oracle Linux or Suse Linux, the primary distinction lies in switching to the appropriate package manager for that distribution, typically *yum*.

Steps

1. Add the repository file suitable for the existing customer subscription and the Debian version utilized. An example would be `/etc/apt/sources.list.d/bacula.list` with the following content:

Listing 1: **APT repositories**

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

2. Run the apt update:

Listing 2: **APT update**

```
apt update
```

3. Install the plugin using:

Listing 3: **APT install**

```
apt install bacula-enterprise-amazon-rds-plugin
```

The plugin has two distinct packages that should be installed automatically with the command shown:

- `bacula-enterprise-amazon-rds-plugin`
- `bacula-enterprise-amazon-rds-plugin-libs`

Result

Amazon RDS Plugin is installed. Click here for more details.

Amazon RDS Plugin Manual Installation

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

Manual installation of the packages may be done after downloading the packages, and then using the package manager to install them.

1. Download the following packages from your Bacula Systems download area.
   - bacula-enterprise-amazon-rds-plugin
   - bacula-enterprise-amazon-rds-plugin-libs
2. After downloading the packages, install them with the command:

Listing 4: **Manual dpkg install**

```
dpkg -i bacula-enterprise-*
```

Result

Amazon RDS Plugin is installed. Click here for more details.

4.3 Result

The installation package includes the following components:

- Jar libraries in `/opt/bacula/lib` (such as `bacula-amazon-rds-plugin-x.x.x.jar` and `bacula-amazon-rds-plugin-libs-x.x.x.jar`). Note that the version of those jar archives is not aligned with the version of the package. However, that version will be shown in the joblog in a following message: "Jar version:X.X.X".
- A plugin connection file (`amazon-rds-fd.so`) in the plugins directory (usually `/opt/bacula/plugins`). Note that amazon-rds acronym refers to Amazon Elastic Compute Cloud.
- A backend file (`amazon-rds-backend`) that invokes the jar files in `/opt/bacula/bin`. This backend file searches for the most recent `bacula-amazon-rds-plugin-x.x.x.jar` file in order to launch it, even though there should generally be only one file present.

Once the plugin is installed, it should be possible to verify it loaded through a status client command in bconsole (`Plugin:` line must contain `amazon-rds`):

Listing 5: **Status client**

```
*st client
Automatically selected Client: 127.0.0.1-fd
```

(continues on next page)

(continued from previous page)

```
Connecting to Client 127.0.0.1-fd at 127.0.0.1:8102

127.0.0.1-fd Version: 18.0.0 (20 Nov 2023)  x86_64-pc-linux-gnu ubuntu 22.04
Daemon started 14-abr-23 10:14. Jobs: run=2 running=0 max=100.
Ulimits: nofile=1024 memlock=2026356736 status=ok
Heap: heap=827,392 smbytes=436,939 max_bytes=5,100,087 bufs=153 max_bufs=248
Sizes: boffset_t=8 size_t=8 debug=600 trace=1 mode=1,2010 bwlimit=0kB/s
Crypto: fips=no crypto=OpenSSL 3.0.2 15 Mar 2022
APIs: !GPFS
Plugin: bpipe(2) amazon-rds(1.0.0)
```

5 Configuration

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The following chapter presents the information on how to configure the plugin. Backup jobs with Bacula Enterprise Amazon RDS Plugin function similarly to other Bacula Enterprise backup jobs after the appropriate Fileset has been established and the access key with necessary permissions is provided.

In the following sections we present how to configure an AWS access key with the associated permissions to use this plugin, and then the parameters to set up a Fileset to invoke Amazon RDS Plugin through a job.

Restore parameters are discussed in the Restore section of *Operations chapter*.

5.1 Access Key Configuration

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The utilization of Bacula Enterprise Amazon RDS Plugin requires the involvement of an AWS IAM service user who possesses a specific set of permissions outlined below.

Subsequently, the user must establish an access key, which should be configured as Fileset parameters. This configuration enables the plugin to effectively retrieve or write data during backup or restore operations.

The plugin needs the following set of permissions to work appropriately:

- Full RDS service permissions for the target instances or clusters

- Full S3 service permissions or permissions to fully manage the destination bucket to be used for the export operation, enabling snapshots to be exported to it and allowing data to be retrieved and sent to the SD
- The user needs to be associated with an export role that needs to be created beforehand (refer to the details below)
- The user needs to have decrypt permissions using a KMS Service Key that needs to be created in advance (refer to the details below)

As an example, the final set of permissions of the user should resemble the image provided below:

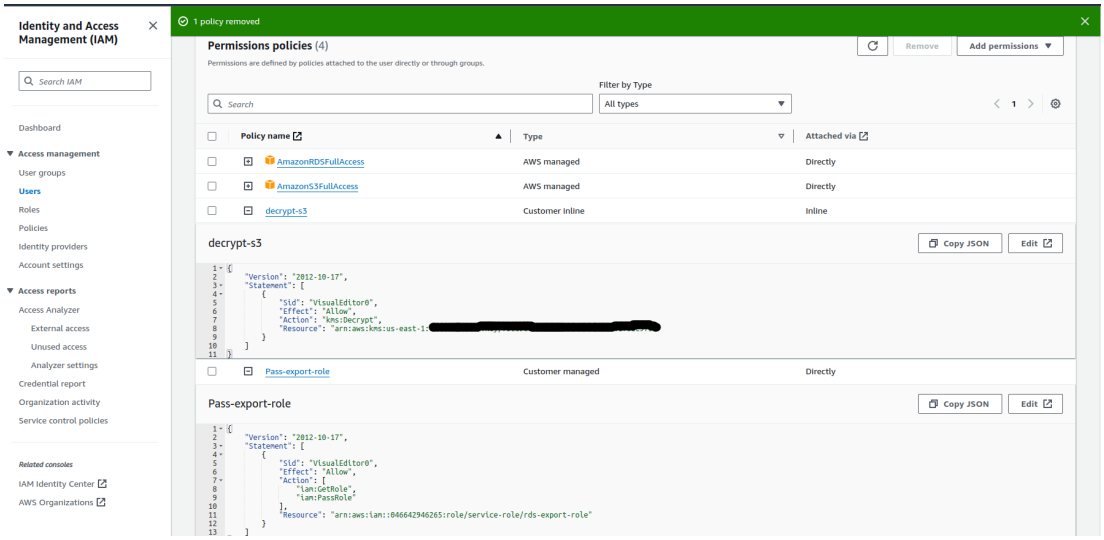

Fig. 2: RDS Permissions Summary for the user

Note that for export operations to succeed, all of the following parameters must be defined: `export_bucket`, `export_role`, `export_key`, `access_key`, `secret_key`.

Export Role

It is necessary to grant tasks write-access permission for the snapshot export tasks to the Amazon S3 bucket designated for exporting the backups. This can be accomplished in 3 steps with a user with permissions to manage policies and roles within the IAM service:

Listing 6: **Export Role Procedure**

```
# 1. Create a policy using the proper bucket name instead of EXAMPLE-BUCKET
$ aws iam create-policy  --policy-name ExportPolicy --policy-document '{
      "Version": "2012-10-17",
      "Statement": [
         {
               "Sid": "ExportPolicy",
               "Effect": "Allow",
               "Action": [
                  "s3:PutObject*",
                  "s3:ListBucket",
                  "s3:GetObject*",
                  "s3:DeleteObject*",
```

(continues on next page)

(continued from previous page)

```
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::EXAMPLE-BUCKET",
                "arn:aws:s3:::EXAMPLE-BUCKET/*"
            ]
        }
    ]
}'

# 2. Create a IAM role, so that Amazon RDS can assume this IAM role on␣
↪your behalf to access your Amazon S3 buckets.
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-
↪document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "export.rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
    }'

# 3. Attach the IAM policy to the just created role
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-
↪s3-export-role

# 4. In the User Management screen, associate the created role to the user␣
↪that will contain the access key for Bacula Enterprise Amazon RDS Plugin
```

For more information, consult: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ExportSnapshot.html#USER_ExportSnapshot.SetupIAMRole

Encryption Key

Create a symmetric encryption AWS KMS key for the server-side encryption. The KMS key is used by the snapshot export task to set up AWS KMS server-side encryption when writing the export data to S3.

The KMS key policy must include both the kms:CreateGrant and kms:DescribeKey permissions. For more information on using KMS keys in Amazon RDS, visit AWS KMS key management official documentation.

Below is an example of a KMS-created key along with its policy.

After creating the key, the IAM user employed for the Amazon RDS Plugin needs to reference the key with a policy similar to what was shown before, using the proper ARN. Below, the policy example from it:

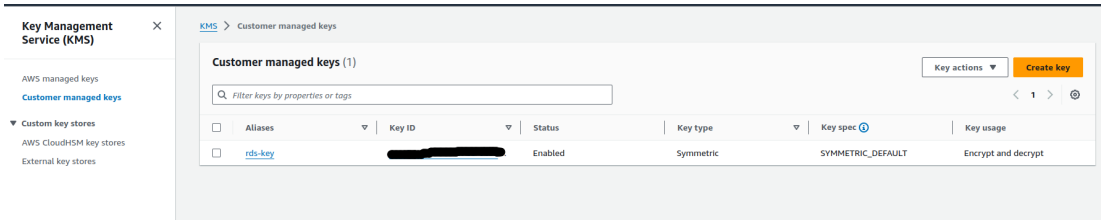

Fig. 3: KMS Key

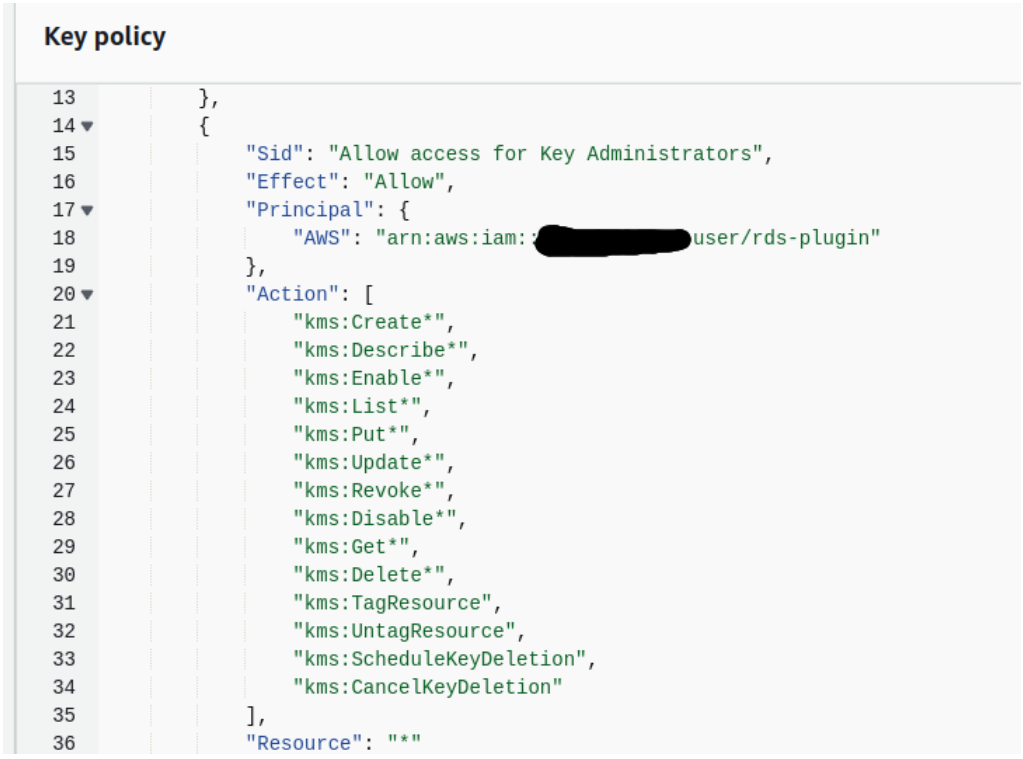

Fig. 4: KMS Key Policy

Listing 7: **Policy for kms key in the user**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-1:046642946265:key/xxxxxxx-
↪yyyy-zzzz-aaaa-bbbbbbbbbbbbbbb"
        }
    ]
}
```

Access Key

After the user has been set up, it is essential to navigate to *access keys* and generate a new one:

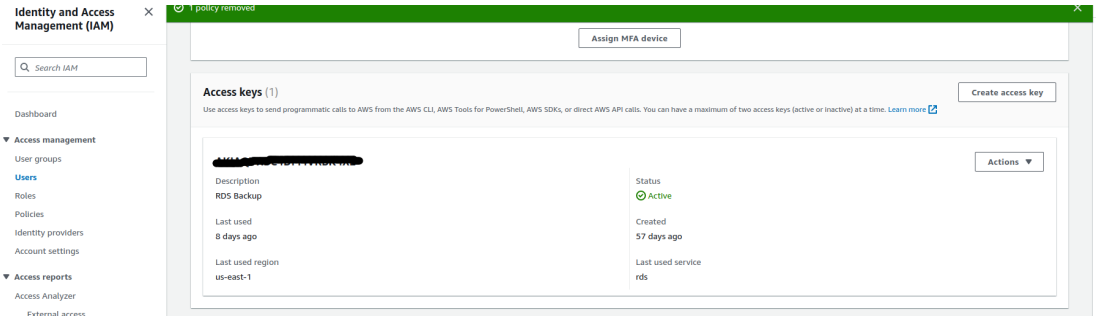

Fig. 5: Amazon RDS Access Key

The Id of the key and the associated secret are the parameters to configure in the plugin Fileset in `access_key` and `access_secret` parameters. Adding also the `region` parameter should be enough to allow the plugin to connect to the target dataset to protect.

See also

- Go to Fileset Configuration

5.2 Fileset Configuration

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

Once the plugin is successfully authorized, it is possible to define regular Filesets for backup jobs in Bacula, where we need to include a line similar to the one below, in order to invoke the Amazon RDS Plugin:

Listing 8: **Fileset RDS**

```
Fileset {
   Name = FS_RDS
   Include {
      Options {
        signature = MD5
        ...
      }
      Plugin = "amazon-rds: <amazon-rds-parameter-1>=<amazon-rds-value-1>
↪<amazon-rds-parameter-2>=<amazon-rds-value-2> ..."
   }
}
```

It is **strongly recommended** to use only one `Plugin` line within each Fileset. The plugin offers the needed flexibility to combine various module backups within the same plugin line. In instances where multiple rds servers exist, it is essential to employ distinct Filesets and separate jobs.

In this plugin, any parameter allowing a list of values can be assigned with a list of values separated by ",".

Below, in the subsections, there are lists that present all the parameters you can use to control Amazon RDS Plugin behavior.

Fileset Connection Parameters

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The following parameters control the connection of the Amazon RDS Plugin AWS.

Option	Require	Default	Values	Example	Description
access_	**Yes**		String: access key with proper permissions	AKIAIOS-FODNN7EXAMPLI	Access key defined in the desired AWS account with access to the target instances and having the right permissions
secret_	**Yes**		String: secret key associated to provided access key	wJalrXUtn-FEMI/K7MDENG/b	Secret associated to the access key
region	No	eu-west-1	String: region code	eu-east-1	Existing region in AWS where the instances to backup reside

Fileset Backup Parameters

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The following list of parameters controls what will be incorporated into the corresponding backup:

Option	Required	Default	Values	Example	Description
instanc	No		Valid database instance or cluster identifiers separated by ‘,’	myInstance1, myInstance2	Comma separated list of database instance or cluster identifiers to backup. If no instance is provided the plugin will list all of them and will backup them
instanc	No		Valid database instance or cluster identifiers separated by ‘,’	excludeExample, i-1233sablkbl	Comma separated list of database instance or cluster identifiers to exclude from backup. If this is the only parameter found for selection, all elements will be included and this list will be excluded
instanc	No		Valid regex	.*-includedSuf	Backup matching instances or clusters by its identifier. Please, only provide list parameters (instances + instances_exclude) or regex ones. But do not try to combine them
instanc	No		Valid regex	.*-excludedSuf	Exclude matching instances or clusters by its identifier. Please, only provide list parameters (instances + instances_exclude) or regex ones. But do not try to combine them. If this is the only parameter found for user selection, all instances will be included and matching instances will be excluded
instanc	No		Tag keys or values (format: tagkey1=value	backup=yes	Backup instances or clusters containing the specified tags
instanc	No		Tag keys or values (format: tagkey1=value	backup=no	Exclude instances or clusters containing the specified tags
start_	No		0, no, No, false, FALSE, false, off ; 1, yes, Yes, TRUE, true, on	Yes	Backup only the list of ebs volumes indicated by this parameter from the selected instances. If not specified, all disks from each instance will be backed up
snapshots_	No	60	Integer	100	Retention or number of snapshots to be kept in Amazon RDS Service. Older ones exceeding this number will be removed
export_	No		String representing an existing bucket in the region	mybucket	Name of the bucket to send the exported parquet files if export is enabled
export_	No		String representing a folder name	myfolder	Optional folder inside the bucket to store the exported files
export_	No		String representing a IAM Role name with the proper export permissions	arn:aws:iam role/rds-export-role	Name of the s3 export role needed to perform export operations
ex	No		String rep	75gt0800	ARN of the AWS Encryption key needed to per

Note

When the `export_download = no` parameter is set, the backup is not sent to the Bacula storage, but there is still the metadata that is copied to the Bacula storage. This is necessary to restore the data backed up in the user's S3 bucket later.

Fileset Common Parameters

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

These parameters control the generic characteristics of the behavior of the Amazon RDS Plugin. You may also encounter these parameters in other Bacula Enterprise Plugins that produce similar effects, so you might already be acquainted with them.

Option	Require	Default	Values	Example	Description
abor	No	No	No, Yes	Yes	If set to **Yes**: Abort job as soon as any error is found with any element. If set to **No**: Jobs can continue even if it they found a problem with some elements. They will try to backup or restore the other and only show a warning
config_fi	No		The path pointing to a file containing any combination of plugin parameters	/opt/bacula/ rds.settings	Allows to define a config file where configure any parameter of the plugin. Therefore you don't need to put them directly in the Plugin line of the fileset
log	No	/opt/bacula rds/amazor rds-debug.log	An existing path with enough permissions for File Daemon to create a file with the provided name	/tmp/amazo rds.log	Generates additional log in addition to what is shown in job log. This parameter is included in the backend file, so, in general, by default the log is going to be stored in the working directory.
debug	No	0	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	Debug level. Greater values generate more debug information	Generates the working/amazon-rds/amazon-rds-debug.log* files containing debut information which is more complete with a greater debug number
path	No	/opt/bacula	An existing path with enough permissions for File Daemon to create any internal (and usually temporary) plugin file	/mnt/my-vol/	Uses this path to store metadata and temporary files

This is an example of a `config_file` with the FilesetRDSConnectionParameters options:

```
# cat /opt/bacula/etc/amazon-rds.settings
region=us-east-1
access_key=AKIAQTESTKEY12134g
secret_key=m23480ahpqwre894qwrffsfdSecretExample
```

Fileset Tuning Parameters

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The following parameters can be utilized to adjust the plugin's behavior for increased flexibility in challenging network conditions, high job concurrency scenarios, and similar situations.

It is generally unnecessary to alter them in most situations.

Option	Require	Default	Values	Example	Description
concurrent_threads	No	500	1-1000	100	Number of maximum concurrent backup threads running in parallel in order to download blocks from a given EBS volume
api_timeout	No	9000	Positive integer (seconds)	2000	Total timeout for AWS API calls
api_read_timeo	No	600	Positive integer (seconds)	5000	Timeout for AWS API calls to respond
general_network_r	No	600	Positive integer (number of retries)	10	Number of retries for failed requests to the AWS API
general_network_d	No	50	Positive integer (seconds)	100	Delay between retries to the AWS API
snapshot_operations	No	1800	Positive integer (seconds)	3600	Timeout for snapshot related operations (like make snapshot) before giving up
export_operations	No	1800	Positive integer (seconds)	3600	Timeout for export operations before giving up
instance_operatio	No	1800	Positive integer (seconds)	3600	Timeout for instance related operations (like waiting for them to be in an operational status) before giving up

Fileset Advanced Parameters

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The following parameters are advanced ones, and they should not be modified in the great majority of cases:

Option	Re-quire	Default	Values	Ex-am-ple	Description
stream_sl	No	1	Positive integer (1/10 sec-conds)	5	Time to sleep when reading header packets from FD and not having a full header available
stream_m	No	120	Positive integer (sec-onds)	360	Max wait time for FD to answer packet requests
time_max	No	86400	Positive integer (sec-onds)	4320	Maximum time to wait to overwrite a debug log that was marked as being used by other process
log-ging_max	No	50MB	String size	300M	Maximum size of a single debug log fileGenerates the working/amazon-rds/amazon-rds-debug.log* files containing debut information which is more complete with a greater debug number
log-ging_max	No	25	Positive integer (number of files)	50	Maximum number of log files to keep
log_rollin	No	amazon-rds.log.%d{ MMM}.log.	String file name pattern	...	Log pattern for rotated log files
split_conf	No	=	Charac-ter	:	Character to be used in config_file parameter as separator for keys and values
pub-lisher_qu	No	1200	Positive integer (sec-onds)	3600	Timeout when internal object publisher queue is full

The internal plugin logging framework presents some relevant features that we are going to describe:

- The `.log` files are rotated automatically. Currently, each file can be 50Mb at maximum and the plugin will keep 25 files.
  - This behavior can be changed using the internal advanced parameters: `logging_max_file_size` and `logging_max_backup_index`.
- The `.err` file can show contents even if no actual error has occurred during the jobs. It can also present contents even if debugging is disabled. This file is not rotated, but it is generally anticipated to remain small in size. If you still need to rotate it, you can include it in a general rotating tool such as `logrotate`.
- Backups in parallel and also failed backups will generate several log files. For example: amazon-rds-debug-0.log, amazon-rds-debug-1.log, and so on.

Fileset Examples

Enterprise

> Bacula Enterprise Only
>
> This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

In this section, some Fileset examples are presented:

Listing 9: **Fileset: for a selection of instances by name, just snapshot**

```
Fileset {
   Name = fs-amazon-rds-instances-a-b
   Include {
      Options { signature = MD5 }
      Plugin = "amazon-rds: region=us-east-1 access_key=AKIAQTESTKEY12134g␣
↪secret_key=m23480ahpqwre894qwrffsfdSecretExample instances=myInstanceA,
↪myInstanceB"
   }
}
```

Listing 10: **Fileset: using a config file**

```
Fileset {
   Name = fs-amazon-rds-instance-a
   Include {
      Options { signature = MD5 }
      Plugin = "amazon-rds: config_file=/opt/bacula/etc/amazon-rds.settings␣
↪instances=myInstanceA "
   }
}

Config file contents in stored in the same File Daemon host in /opt/bacula/
↪etc/amazon-rds.settings:
region=us-east-1
access_key=AKIAQTESTKEY12134g
secret_key=m23480ahpqwre894qwrffsfdSecretExample
```

Listing 11: **Fileset: Export and download the database**

```
Fileset {
   Name = fs-amazon-rds-instance-a
   Include {
      Options { signature = MD5 }
      Plugin = "amazon-rds: config_file=/opt/bacula/etc/amazon-rds.settings␣
↪instances=myInstanceA export_bucket=myExportBucket export_
↪role=arn:aws:iam::046642946265:role/service-role/rds-export-role export_
↪key=75gt9800-719e-44b1-63h0-4197d0gt9015"
   }
```

(continues on next page)

(continued from previous page)

```
}
```

Listing 12: **Fileset: Backup instances contaning tag bacula**

```
Fileset {
   Name = fs-amazon-rds-instances
   Include {
      Options { signature = MD5 }
      Plugin = "amazon-rds: config_file=/opt/bacula/etc/amazon-rds.settings␣
↪instances_tags=\"bacula\""
   }
}
```

Listing 13: **Fileset: Backup instances contaning tag backup=yes**

```
Fileset {
   Name = fs-amazon-rds-instances
   Include {
      Options { signature = MD5 }
      Plugin = "amazon-rds: config_file=/opt/bacula/etc/amazon-rds.settings␣
↪instances_tags=\"backup=yes\""
   }
}
```

Listing 14: **Fileset: Tweak snapshot retention and start instances if they are offline**

```
Fileset {
   Name = fs-amazon-rds-instance-a
   Include {
      Options { signature = MD5 }
      Plugin = "amazon-rds: config_file=/opt/bacula/etc/amazon-rds.settings␣
↪instances=myInstanceA start_instances=yes snapshot_retention=100"
   }
}
```


Listing 15: **Fileset: by tag, but exclude A**

```
Fileset {
   Name = fs-amazon-rds-instances
   Include {
      Options { signature = MD5 }
      Plugin = "amazon-rds: config_file=/opt/bacula/etc/amazon-rds.settings␣
↪instances_tags=\"backup=yes\" instances_exclude=myInstanceA"
   }
}
```

6 Best Practices

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The following article presents best practices regarding jobs distribution, concurrency and performance.

6.1 Jobs Distribution

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

It is recommended to split the target backup between different instances or clusters, ideally having one job per instance or cluster.

Following this recommendation, errors in a single job will not compromise the whole backup cycle, allowing for successful backups of certain instances or clusters, even if others encounter issues. This also makes it easier to identify the cause of any error.

6.2 Concurrency

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

When using Amazon RDS APIs, it is possible to find a variety of boundaries that need to be considered. We highlight some of them below:

- Amazon RDS limits: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Limits.html
- Capabilities of the host serving the RDS Service
- Service usage during the backup window

If a boundary is crossed, the corresponding request will usually fail. Bacula Amazon RDS Plugin is prepared to wait for a certain amount of time and then retry it, thus offering a degree of resiliency. However, it is crucial to plan an adequate strategy to backup all the elements without frequently approaching any boundaries. This entails managing the number of concurrent requests made during the backup window.

The recommended strategy to backup a new environment is to plan a step-by-step testing scenario prior to deployment, where the number of instances and the concurrency of the jobs are increased progressively. Another important aspect is the timing schedule, as some boundaries are related to time-frames (i.e., the number of request per time unit). If you detect you are reaching boundaries when running all your backups during a single day of the week, try to increase the time window and distribute the load throughout it in order to enhance performance results.

Note that from an architectural and AWS service point of view, you can also consider to:

- Run your File Daemon directly in the cloud (if your SD is also in the cloud)
- Run your Storage Daemon and File Daemon in the same host, so you skip one network hop in the process (recommended)
- Use a dedicated AWS connection (https://aws.amazon.com/directconnect/)

6.3 Performance

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The performance of the plugin is highly dependent on various external factors:

- Latency and bandwidth to AWS
- Network infrastructure
- FD Host hardware
- FD Load
- Ratio number of elements/size
- And many more.

In short, it is not possible to establish an exact reference for the duration required to complete a backup.

Typically, backups that do not use the export and download functions will proceed as quickly as AWS permits However, when using the export and download functions, the aforementioned factors become increasingly pertinent, as certain files will be created in S3 that must subsequently be downloaded.

As a guideline concerning the number of records per table:

- Numerous small tables to protect: More files (one file per table) per time period, but smaller speed (MB/s).
- Large tables to protect: Fewer files (one file per table) per time period, but higher speed (MB/s).

It is also important to note that the snapshot cleanup process requires some time to finalize, in addition to the export operation itself.

It is recommended to benchmark your own environment based on your specific requirements and needs.

There are various strategies available to use this plugin, so it is recommended to evaluate which option best meets your needs prior to deploying the jobs across your entire environment, ensuring optimal results:

- You can have a job per instance or cluster (recommended)
- You can have multiple instances per job or even all your instances or clusters in the same job (not recommended)
- You can split your workload through a schedule, or try to run all your jobs together
- You can specifically select the instances you want to backup or backup them all
- You can specifically select the clusters you want to backup or backup them all
- You can specifically select the tables you want to export and backup or process them all
- You can run your File Daemon on premise or in the cloud
- You can use default internet connection to AWS or use a dedicated AWS connection (https://aws.amazon.com/directconnect/)
- You can run your Storage Daemon and File Daemon in the same host, so you skip one network hop in the process (recommended).

7 Operations

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The following article describes details regarding backup, restore or list operations with **Bacula Enterprise Amazon RDS Plugin**.

7.1 Backup

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The overall backup operation involves the following steps:

1. Identify all selected target database instances or clusters.
2. Trigger a snapshot of the selected targets. This snapshot is inherently incremental compared to the previous one (if applicable).
3. If export is enabled, trigger an export operation of the selected tables to S3 (to a bucket configured in the plugin).
4. If download is enabled, download the exported files and store them in Bacula (send them to the Storage Daemon).
5. If export delete option is enabled, delete the exported files from the S3 bucket.
6. Remove old snapshots according to the established snapshot retention policy (which also includes deleting associated export files if the `export_delete` is enabled).

Backup File Structure

A single database backup with export and download enabled will produce contents in the backup similar to the following ones:

Listing 16: **Files protected with the backup**

```
$ dir
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/
-rw-r-----   1 nobody   nogroup          645  2024-06-06 14:38:30  /@amazon-
↪rds/jg-pg-aws/export_info_plugintest-20240606-135513-03.json
-rw-r-----   1 nobody   nogroup        72801  2024-06-06 14:38:30  /@amazon-
↪rds/jg-pg-aws/export_tables_info_plugintest-20240606-135513-03_from_1_to_38.
↪json
-rw-r-----   1 nobody   nogroup         3192  2024-03-28 11:27:41  /@amazon-
↪rds/jg-pg-aws/jg-pg-aws.i.json
-rw-r-----   1 nobody   nogroup         1103  1970-01-01 00:59:59  /@amazon-
↪rds/jg-pg-aws/plugintest-20240606-135513-03.isnap.json

$ pwd
cwd is: /@amazon-rds/jg-pg-aws/

$ cd bacula
cwd is: /@amazon-rds/jg-pg-aws/bacula/

$ dir
```

(continues on next page)

(continued from previous page)

```
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.basefiles/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.cdimages/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.client/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.counters/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.device/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.events/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.file/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.fileevents/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.filemedia/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.fileset/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.job/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.jobhisto/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.jobmedia/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.location/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.locationlog/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.log/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.malwaremd5/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.malwaresha256/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.media/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.mediatype/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.metaattachment/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.metaemail/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.object/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.path/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.pathhierarchy/
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.pathvisibility/
```

(continues on next page)

(continued from previous page)

```
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.pool/
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.restoreobject/
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.snapshot/
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.status/
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.storage/
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.tagclient/
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.tagjob/
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.tagmedia/
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.tagobject/
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.unsavedfiles/
 ----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/jg-pg-aws/bacula/public.version/

 cd public.status/1
 cwd is: /@amazon-rds/jg-pg-aws/bacula/public.status/1/

 $ dir
 -rw-r-----   1 nobody   nogroup         1421  2024-06-06 14:36:49  /@amazon-
↪rds/jg-pg-aws/bacula/public.status/1/part-00000-e8ed76c2-8a29-49c7-9e0f-
↪771a666ebe74-c000.gz.parquet
```

The JSON files represent information about the database instance, the snapshot and the export operation. Additionally, there exists a directory for each table of the database, wherein one can find an Apache Parquet file containing the data of that specific table.

Snapshot and Exports Livecycle

The Bacula Enterprise Amazon RDS Plugin controls the whole livecycle of the snapshots created within the Amazon Web Services Cloud, as well as the livecycle of the data exported to S3.

Amazon RDS Snapshots are actually Amazon EBS Snapshots containing also certain transaction logs. The EBS layer is transparent to the user and cannot be directly controlled.

The database snapshots are sent to an S3 bucket, which is also not directly controllable. When an RDS instance or cluster snapshot is created, Amazon automatically manages these details in the background, creating the appropriate EBS device snapshot and storing the data in S3. Similarly, when a snapshot is removed from RDS, the corresponding data in S3 will be automatically removed.

Bacula Enterprise Amazon RDS Plugin will remove exported files from S3 after each backup if the `export_delete` flag is enabled.

Snapshots that exceed the `snapshots_retention` value will be deleted. Their associated exports, if they still exist, will also be deleted if `export_delete` flag is enabled.

7.2 Restore

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

Restore Procedure

Bacula Enterprise Amazon RDS Plugin offers four distinct restore modes:

1. Generate a new instance from a given snapshot (from the backup or from an arbitrary snapshot identifier).
2. Generate a new instance with a specific state using Point-In-Time Recovery.
3. Restore metadata or Apache Parquet exported data locally to the filesystem for subsequent processing.
4. Restore Apache Parquet information into a running MySQL or PostgreSQL database.

Depending on the desired result, the restore operation should be parametrized differently.

Restore Parameters

Amazon RDS Plugin is able to perform a raw restore to any local filesystem mounted over the host where the File Daemon operates, or directly to the Amazon RDS environment. The restore method is selected determined by the value of the `where` parameter at restore time:

- Empty or '/' (example: `where=/`) -> Amazon RDS restore will be triggered
- Any other path for where (example: `where=/tmp`) -> Raw restore to the local file system will be triggered.

This principle generally applies to other Bacula plugins as well. Nevertheless, in this specific plugin, it is advised against using the raw restore method by the File Daemon. This recommendation stems from the fact that the raw restore method retains the headers generated during the backup process in the volume disk files. If the intention is to restore the filesystem, one can trigger an Amazon RDS restore using the `where=/` parameter. It is important to also use the restore variable `to_local_path` and specify the desired destination. By doing so, the mentioned headers will be automatically removed, resulting in a disk image that is appropriate for any future purposes.

When using the Amazon RDS restore method, the following parameters are available to control the restore behavior within the "Plugin Options" menu during a BConsole restore session:

Option	Require	Default	Values	Example	Description
instanc	No	Generated from Restore job name	String for the new instance or cluster that will be generated with the restore. Only a-zA-Z or '-' characters are allowed	MyRestoredInstance	Set the identifier for a new restored database instance or cluster. If none is provided and the restore operation needs to create a new instance or cluster, the name will be generated from the restore job
from_	No		Existing snapshot identifier in RDS service	my-snap-2024-05-16-12-30	Set a snapshot identifier, existing in AWS to restore an instance or cluster directly from it. If not specified, the identifier will be get from the backup information
instanc	No	Original backup value	Accepted instance type in Amazon RDS. Defined in: https://aws.amazon.com/rds/instance-types/	c7g.mediun	Set database instance class
availability_zo	No	Original backup value	String: availability zone existing in: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZo html	us-east-1a	Set the destination availability zone for the instance(s) and its/their volume(s)
parameter_gr	No	Original backup value	String: Name of the parameter group	mygroup	Set the name(s) of the parameter groups that will be applied to the restored database instance (separated by ',')
vpc_s	No	Original backup value	String: Name of the security group id	sg-03237oe44	Set the id(s) of the VPC Security Groups that will be applied to the restored database instance (separated by ',')
tags	No		List of key value strings: key1=value1, key2=value2		Set tags to the restored instance(s) (tag1key:tag1value,tag2key:tag2value...)
multi_	No	Original backup value	0, no, No, false, FALSE, false, off ; 1, yes, Yes, TRUE, true, on	yes	Configure the restored database instance as Multi AZ
db_pc	No	Original backup value	Integer	2140	Set the database port
storage_ty	No	Original backup value	Accepted volume type in Amazon EBS. Defined in: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html		Set the instance storage type
storage_th	No	Original backup value	Positive integer		Set the instance storage throughput
iops	No	Original backup value	Positive integer		Iops value for instance storage

Restore Use Cases

The following restore scenarios are supported, and steps to execute them are described:

a) Generate a new instance or cluster from a the snapshot that was taken during the selected backup

   1. Run a restore session selecting appropriate backup jobs
   2. Select all the contents of the backup
   3. Use `Where=/`

b) Generate a new instance or cluster from an arbitrary snapshot that is existing in Amazon RDS, and configure a specific new instance id:

   - Follow previously described 'a' scenario.
   - Set `instance_identifier` with the value of the desired new name
   - Set `from_snapshot_id` with the value of the desired previous snapshots

c) Restore an instance in Point-In-Time Recovery mode:

- Follow previously described 'a' scenario
- Before confirming the restore operation, set `pitr_from_instance_id` with the desired instance or cluster to use, or let the restore used the instance from the backup job selected
- You may want to run a `.query` command (parameter=instance) to check the last available restore time for your instance
- Set `pitr_date` with the desired value, or set `pitr_latest` to yes.

d) Restore an instance to Amazon RDS, but adjust any configuration parameter of it (advanced):

- Follow previously described 'a', 'b' or 'c' scenarios.
- Adjust any desired parameter from the following list: `instance_class`, `availability_zone`, `parameter_groups`, `vpc_security_groups_ids`, `tags`, `multi_az`, `db_port`, `storage_type`, `storage_throughput`, `iops`
- Note that the configuration you set here will be applied as entered. Therefore, the consistency of that configuration will depend on all elements being correct (existing and consistent) at Amazon RDS side for your particular infrastructure

e) Restore an instance to Amazon RDS, but to a different location:

- Follow previously described 'a', 'b', 'c' or 'd' scenarios.
- Adjust `region`, `access_key`, `secret_key` with the destination values

f) Restore instance Apache Parquet files to a running database in the File Daemon host:

- Follow previously described 'a' scenario.
- Adjust database connection with 'parquet_xxx' parameters
- By default, if no host is established, a socket connection will be attempted. If a host is specified, the connection will use TCP/IP
- Pay attention to the `parquet_save_mode`, which will allow you to control how the restore should behave in terms of finding previous values in your database

g) Restore instance files or volume files to a local directory:

   1. Run a restore session selecting appropriate backup jobs
   2. Select the desired files (or all of them) inside a given `i-xxxxxxx/` folder

3. Use `Where=/`
4. Adjust `to_local_path` to the desired path

Restore Example Session

Note

It is also possible to run backup or restore operations from any of the Bacula Graphical User Interfaces.

Listing 17: **Restore bconsole session**

```
restore jobid=1 Client=127.0.0.1-fd where="/"
Using Catalog "MyCatalog"
You have selected the following JobId: 1

Building directory tree for JobId(s) 1 ...
41 files inserted into the tree.

You are now entering file selection mode where you add (mark) and
remove (unmark) files to be restored. No files are initially added, unless
you used the "all" keyword on the command line.
Enter "done" to leave this mode.

cwd is: /
$ cd "/@amazon_rds/"
Invalid path given.
cwd is: /
$ dir
----------   0 root     root               0  1970-01-01 01:00:00  /@amazon-
↪rds/
$ estimate
41 total files; 0 marked to be restored; 0 bytes.
$ mark *
41 files marked.
$ done
Bootstrap records written to /tmp/regress/working/127.0.0.1-dir.restore.3.bsr

The Job will require the following (*=>InChanger):
   Volume(s)                 Storage(s)                SD Device(s)
===========================================================================

   TEST-2024-06-06:0         File                      FileStorage

Volumes marked with "*" are in the Autochanger.


41 files selected to be restored.

Using Catalog "MyCatalog"
Run Restore job
```

(continues on next page)

(continued from previous page)

```
JobName:         RestoreFiles
Bootstrap:       /tmp/regress/working/127.0.0.1-dir.restore.3.bsr
Where:           /
Replace:         Always
FileSet:         Full Set
Backup Client:   127.0.0.1-fd
Restore Client:  127.0.0.1-fd
Storage:         File
When:            2024-06-06 14:39:13
Catalog:         MyCatalog
Priority:        10
Plugin Options:  *None*
OK to run? (Yes/mod/no): mod
Parameters to modify:
   1: Level
   2: Storage
   3: Job
   4: FileSet
   5: Restore Client
   6: When
   7: Priority
   8: Bootstrap
   9: Where
   10: File Relocation
   11: Replace
   12: JobId
   13: Plugin Options
Select parameter to modify (1-13): 13
Automatically selected : amazon-rds: region="us-east-1" access_key=
↪"AKIAQVXBC4DM4VRBK4XL" secret_key="hqvQac/ZikF6+E9AGDOwGjWPMyrz0K6zAC1eQ9KL
↪" instances="jg-pg-aws" start_instances=yes snapshots_retention=2 export_
↪bucket=j-bacula-rds export_role=arn:aws:iam::046642946265:role/service-role/
↪rds-export-role export_key=73ed9600-749e-47b5-93f0-6761f0ac9734 export_
↪delete=yes debug=6
Plugin Restore Options
Option                          Current Value        Default Value
instance_identifier:            *None*               (*none*)
from_snapshot_id:               *None*               (*none*)
instance_class:                 *None*               (*none*)
availability_zone:              *None*               (*none*)
parameter_groups:               *None*               (*none*)
vpc_security_groups_ids:        *None*               (*none*)
tags:                           *None*               (*none*)
multi_az:                       *None*               (*none*)
db_port:                        *None*               (*none*)
storage_type:                   *None*               (*none*)
storage_throughput:             *None*               (*none*)
iops:                           *None*               (*none*)
pitr_from_instance_id:          *None*               (*none*)
pitr_date:                      *None*               (*none*)
pitr_latest:                    *None*               (*none*)
parquet_db_socket:              *None*               (*none*)
```

(continues on next page)

(continued from previous page)

```
parquet_db_engine:              *None*                (*none*)
parquet_db_host:                *None*                (*none*)
parquet_db_name:                *None*                (*none*)
parquet_db_port:                *None*                (*none*)
parquet_db_user:                *None*                (*none*)
parquet_db_password:            *None*                (*none*)
parquet_save_mode:              *None*                (*none*)
access_key:                     *None*                (*none*)
secret_key:                     *None*                (*none*)
region:                         *None*                (*none*)
debug:                          *None*                (*none*)
Use above plugin configuration? (Yes/mod/no): mod
You have the following choices:
   1: instance_identifier (Set the identifier for a new restored database␣
↪instance or cluster)
   2: from_snapshot_id (Set a snapshot identifier, exisiting in AWS to␣
↪restore an instance or cluster directly from it)
   3: instance_class (Set database instance class)
   4: availability_zone (Set the destination availability zone for the␣
↪instance(s) and its/their volume(s))
   5: parameter_groups (Set the name(s) of the parameter groups that will be␣
↪applied to the restored database instance (separated by ','))
   6: vpc_security_groups_ids (Set the id(s) of the VPC Security Groups that␣
↪will be applied to the restored database instance (separated by ','))
   7: tags (Set tags to the restored instance(s) (tag1key:tag1value,
↪tag2key:tag2value...))
   8: multi_az (Configure the restored database instance as Multi AZ)
   9: db_port (Set the database port)
   10: storage_type (Set the instance storage type)
   11: storage_throughput (Set the instance storage throughput)
   12: iops (Iops value for instance storage)
   13: pitr_from_instance_id (Set the instance identifier that will reference␣
↪the instance to be used to proceed with a Point In Time Recovery restore)
   14: pitr_date (Sets the date for Point in Time Recovery restore)
   15: pitr_latest (Run a Point in Time Recovery restore using the latest␣
↪date available)
   16: parquet_db_socket (Set the socket path to use a socket based␣
↪connection to restore parquet data from the backup directly into a database)
   17: parquet_db_engine (Set the engine (postgresql or mysql) to restore␣
↪parquet data from the backup directly into a database)
   18: parquet_db_host (Set the host to connect and restore parquet data from␣
↪the backup directly into a database)
   19: parquet_db_name (Set the database name to connect and restore parquet␣
↪data from the backup directly into a database)
   20: parquet_db_port (Set the database port to connect and restore parquet␣
↪data from the backup directly into a database)
   21: parquet_db_user (Set the database user to connect and restore parquet␣
↪data from the backup directly into a database)
   22: parquet_db_password (Set the database password to connect and restore␣
↪parquet data from the backup directly into a database)
   23: parquet_save_mode (Set the restore mode for a parquet restore: Append,␣
↪Overwrite, ErrorIfExists or Ignore)
```

(continues on next page)

(continued from previous page)

```
   24: access_key (Set a different access key to access to the destination)
   25: secret_key (Set a different secret key to access to the destination)
   26: region (Set the destination region)
   27: debug (Change debug level)
Select parameter to modify (1-27): 17
Please enter a value for parquet_db_engine: mysql
Plugin Restore Options
Option                          Current Value        Default Value
instance_identifier:            *None*               (*none*)
from_snapshot_id:               *None*               (*none*)
instance_class:                 *None*               (*none*)
availability_zone:              *None*               (*none*)
parameter_groups:               *None*               (*none*)
vpc_security_groups_ids:        *None*               (*none*)
tags:                           *None*               (*none*)
multi_az:                       *None*               (*none*)
db_port:                        *None*               (*none*)
storage_type:                   *None*               (*none*)
storage_throughput:             *None*               (*none*)
iops:                           *None*               (*none*)
pitr_from_instance_id:          *None*               (*none*)
pitr_date:                      *None*               (*none*)
pitr_latest:                    *None*               (*none*)
parquet_db_socket:              *None*               (*none*)
parquet_db_engine:              mysql                (*none*)
parquet_db_host:                *None*               (*none*)
parquet_db_name:                *None*               (*none*)
parquet_db_port:                *None*               (*none*)
parquet_db_user:                *None*               (*none*)
parquet_db_password:            *None*               (*none*)
parquet_save_mode:              *None*               (*none*)
access_key:                     *None*               (*none*)
secret_key:                     *None*               (*none*)
region:                         *None*               (*none*)
debug:                          *None*               (*none*)
Use above plugin configuration? (Yes/mod/no): mod
You have the following choices:
   1: instance_identifier (Set the identifier for a new restored database␣
↪instance or cluster)
   2: from_snapshot_id (Set a snapshot identifier, exisiting in AWS to␣
↪restore an instance or cluster directly from it)
   3: instance_class (Set database instance class)
   4: availability_zone (Set the destination availability zone for the␣
↪instance(s) and its/their volume(s))
   5: parameter_groups (Set the name(s) of the parameter groups that will be␣
↪applied to the restored database instance (separated by ','))
   6: vpc_security_groups_ids (Set the id(s) of the VPC Security Groups that␣
↪will be applied to the restored database instance (separated by ','))
   7: tags (Set tags to the restored instance(s) (tag1key:tag1value,
↪tag2key:tag2value...))
   8: multi_az (Configure the restored database instance as Multi AZ)
   9: db_port (Set the database port)
```

(continues on next page)

(continued from previous page)

```
   10: storage_type (Set the instance storage type)
   11: storage_throughput (Set the instance storage throughput)
   12: iops (Iops value for instance storage)
   13: pitr_from_instance_id (Set the instance identifier that will reference␣
↪the instance to be used to proceed with a Point In Time Recovery restore)
   14: pitr_date (Sets the date for Point in Time Recovery restore)
   15: pitr_latest (Run a Point in Time Recovery restore using the latest␣
↪date available)
   16: parquet_db_socket (Set the socket path to use a socket based␣
↪connection to restore parquet data from the backup directly into a database)
   17: parquet_db_engine (Set the engine (postgresql or mysql) to restore␣
↪parquet data from the backup directly into a database)
   18: parquet_db_host (Set the host to connect and restore parquet data from␣
↪the backup directly into a database)
   19: parquet_db_name (Set the database name to connect and restore parquet␣
↪data from the backup directly into a database)
   20: parquet_db_port (Set the database port to connect and restore parquet␣
↪data from the backup directly into a database)
   21: parquet_db_user (Set the database user to connect and restore parquet␣
↪data from the backup directly into a database)
   22: parquet_db_password (Set the database password to connect and restore␣
↪parquet data from the backup directly into a database)
   23: parquet_save_mode (Set the restore mode for a parquet restore: Append,␣
↪Overwrite, ErrorIfExists or Ignore)
   24: access_key (Set a different access key to access to the destination)
   25: secret_key (Set a different secret key to access to the destination)
   26: region (Set the destination region)
   27: debug (Change debug level)
Select parameter to modify (1-27): 19
Please enter a value for parquet_db_name: test
Plugin Restore Options
Option                          Current Value        Default Value
instance_identifier:            *None*               (*none*)
from_snapshot_id:               *None*               (*none*)
instance_class:                 *None*               (*none*)
availability_zone:              *None*               (*none*)
parameter_groups:               *None*               (*none*)
vpc_security_groups_ids:        *None*               (*none*)
tags:                           *None*               (*none*)
multi_az:                       *None*               (*none*)
db_port:                        *None*               (*none*)
storage_type:                   *None*               (*none*)
storage_throughput:             *None*               (*none*)
iops:                           *None*               (*none*)
pitr_from_instance_id:          *None*               (*none*)
pitr_date:                      *None*               (*none*)
pitr_latest:                    *None*               (*none*)
parquet_db_socket:              *None*               (*none*)
parquet_db_engine:              mysql                (*none*)
parquet_db_host:                *None*               (*none*)
parquet_db_name:                test                 (*none*)
parquet_db_port:                *None*               (*none*)
```

(continues on next page)

(continued from previous page)

```
parquet_db_user:                *None*                  (*none*)
parquet_db_password:            *None*                  (*none*)
parquet_save_mode:              *None*                  (*none*)
access_key:                     *None*                  (*none*)
secret_key:                     *None*                  (*none*)
region:                         *None*                  (*none*)
debug:                          *None*                  (*none*)
Use above plugin configuration? (Yes/mod/no): mod
You have the following choices:
   1: instance_identifier (Set the identifier for a new restored database␣
↪instance or cluster)
   2: from_snapshot_id (Set a snapshot identifier, exisiting in AWS to␣
↪restore an instance or cluster directly from it)
   3: instance_class (Set database instance class)
   4: availability_zone (Set the destination availability zone for the␣
↪instance(s) and its/their volume(s))
   5: parameter_groups (Set the name(s) of the parameter groups that will be␣
↪applied to the restored database instance (separated by ','))
   6: vpc_security_groups_ids (Set the id(s) of the VPC Security Groups that␣
↪will be applied to the restored database instance (separated by ','))
   7: tags (Set tags to the restored instance(s) (tag1key:tag1value,
↪tag2key:tag2value...))
   8: multi_az (Configure the restored database instance as Multi AZ)
   9: db_port (Set the database port)
   10: storage_type (Set the instance storage type)
   11: storage_throughput (Set the instance storage throughput)
   12: iops (Iops value for instance storage)
   13: pitr_from_instance_id (Set the instance identifier that will reference␣
↪the instance to be used to proceed with a Point In Time Recovery restore)
   14: pitr_date (Sets the date for Point in Time Recovery restore)
   15: pitr_latest (Run a Point in Time Recovery restore using the latest␣
↪date available)
   16: parquet_db_socket (Set the socket path to use a socket based␣
↪connection to restore parquet data from the backup directly into a database)
   17: parquet_db_engine (Set the engine (postgresql or mysql) to restore␣
↪parquet data from the backup directly into a database)
   18: parquet_db_host (Set the host to connect and restore parquet data from␣
↪the backup directly into a database)
   19: parquet_db_name (Set the database name to connect and restore parquet␣
↪data from the backup directly into a database)
   20: parquet_db_port (Set the database port to connect and restore parquet␣
↪data from the backup directly into a database)
   21: parquet_db_user (Set the database user to connect and restore parquet␣
↪data from the backup directly into a database)
   22: parquet_db_password (Set the database password to connect and restore␣
↪parquet data from the backup directly into a database)
   23: parquet_save_mode (Set the restore mode for a parquet restore: Append,␣
↪Overwrite, ErrorIfExists or Ignore)
   24: access_key (Set a different access key to access to the destination)
   25: secret_key (Set a different secret key to access to the destination)
   26: region (Set the destination region)
   27: debug (Change debug level)
```

(continues on next page)

(continued from previous page)

```
Select parameter to modify (1-27): 21
Please enter a value for parquet_db_user: root
Plugin Restore Options
Option                          Current Value        Default Value
instance_identifier:            *None*               (*none*)
from_snapshot_id:               *None*               (*none*)
instance_class:                 *None*               (*none*)
availability_zone:              *None*               (*none*)
parameter_groups:               *None*               (*none*)
vpc_security_groups_ids:        *None*               (*none*)
tags:                           *None*               (*none*)
multi_az:                       *None*               (*none*)
db_port:                        *None*               (*none*)
storage_type:                   *None*               (*none*)
storage_throughput:             *None*               (*none*)
iops:                           *None*               (*none*)
pitr_from_instance_id:          *None*               (*none*)
pitr_date:                      *None*               (*none*)
pitr_latest:                    *None*               (*none*)
parquet_db_socket:              *None*               (*none*)
parquet_db_engine:              mysql                (*none*)
parquet_db_host:                *None*               (*none*)
parquet_db_name:                test                 (*none*)
parquet_db_port:                *None*               (*none*)
parquet_db_user:                root                 (*none*)
parquet_db_password:            *None*               (*none*)
parquet_save_mode:              *None*               (*none*)
access_key:                     *None*               (*none*)
secret_key:                     *None*               (*none*)
region:                         *None*               (*none*)
debug:                          *None*               (*none*)
Use above plugin configuration? (Yes/mod/no): mod
You have the following choices:
   1: instance_identifier (Set the identifier for a new restored database␣
↪instance or cluster)
   2: from_snapshot_id (Set a snapshot identifier, exisiting in AWS to␣
↪restore an instance or cluster directly from it)
   3: instance_class (Set database instance class)
   4: availability_zone (Set the destination availability zone for the␣
↪instance(s) and its/their volume(s))
   5: parameter_groups (Set the name(s) of the parameter groups that will be␣
↪applied to the restored database instance (separated by ','))
   6: vpc_security_groups_ids (Set the id(s) of the VPC Security Groups that␣
↪will be applied to the restored database instance (separated by ','))
   7: tags (Set tags to the restored instance(s) (tag1key:tag1value,
↪tag2key:tag2value...))
   8: multi_az (Configure the restored database instance as Multi AZ)
   9: db_port (Set the database port)
   10: storage_type (Set the instance storage type)
   11: storage_throughput (Set the instance storage throughput)
   12: iops (Iops value for instance storage)
   13: pitr_from_instance_id (Set the instance identifier that will reference␣
```

(continues on next page)

(continued from previous page)

```
↪the instance to be used to proceed with a Point In Time Recovery restore)
   14: pitr_date (Sets the date for Point in Time Recovery restore)
   15: pitr_latest (Run a Point in Time Recovery restore using the latest␣
↪date available)
   16: parquet_db_socket (Set the socket path to use a socket based␣
↪connection to restore parquet data from the backup directly into a database)
   17: parquet_db_engine (Set the engine (postgresql or mysql) to restore␣
↪parquet data from the backup directly into a database)
   18: parquet_db_host (Set the host to connect and restore parquet data from␣
↪the backup directly into a database)
   19: parquet_db_name (Set the database name to connect and restore parquet␣
↪data from the backup directly into a database)
   20: parquet_db_port (Set the database port to connect and restore parquet␣
↪data from the backup directly into a database)
   21: parquet_db_user (Set the database user to connect and restore parquet␣
↪data from the backup directly into a database)
   22: parquet_db_password (Set the database password to connect and restore␣
↪parquet data from the backup directly into a database)
   23: parquet_save_mode (Set the restore mode for a parquet restore: Append,␣
↪Overwrite, ErrorIfExists or Ignore)
   24: access_key (Set a different access key to access to the destination)
   25: secret_key (Set a different secret key to access to the destination)
   26: region (Set the destination region)
   27: debug (Change debug level)
Select parameter to modify (1-27): 22
Please enter a value for parquet_db_password: root
Plugin Restore Options
Option                          Current Value        Default Value
instance_identifier:            *None*               (*none*)
from_snapshot_id:               *None*               (*none*)
instance_class:                 *None*               (*none*)
availability_zone:              *None*               (*none*)
parameter_groups:               *None*               (*none*)
vpc_security_groups_ids:        *None*               (*none*)
tags:                           *None*               (*none*)
multi_az:                       *None*               (*none*)
db_port:                        *None*               (*none*)
storage_type:                   *None*               (*none*)
storage_throughput:             *None*               (*none*)
iops:                           *None*               (*none*)
pitr_from_instance_id:          *None*               (*none*)
pitr_date:                      *None*               (*none*)
pitr_latest:                    *None*               (*none*)
parquet_db_socket:              *None*               (*none*)
parquet_db_engine:              mysql                (*none*)
parquet_db_host:                *None*               (*none*)
parquet_db_name:                test                 (*none*)
parquet_db_port:                *None*               (*none*)
parquet_db_user:                root                 (*none*)
parquet_db_password:            root                 (*none*)
parquet_save_mode:              *None*               (*none*)
access_key:                     *None*               (*none*)
```

(continues on next page)

(continued from previous page)

```
secret_key:                     *None*              (*none*)
region:                         *None*              (*none*)
debug:                          *None*              (*none*)
Use above plugin configuration? (Yes/mod/no): mod
You have the following choices:
   1: instance_identifier (Set the identifier for a new restored database␣
↪instance or cluster)
   2: from_snapshot_id (Set a snapshot identifier, exisiting in AWS to␣
↪restore an instance or cluster directly from it)
   3: instance_class (Set database instance class)
   4: availability_zone (Set the destination availability zone for the␣
↪instance(s) and its/their volume(s))
   5: parameter_groups (Set the name(s) of the parameter groups that will be␣
↪applied to the restored database instance (separated by ','))
   6: vpc_security_groups_ids (Set the id(s) of the VPC Security Groups that␣
↪will be applied to the restored database instance (separated by ','))
   7: tags (Set tags to the restored instance(s) (tag1key:tag1value,
↪tag2key:tag2value...))
   8: multi_az (Configure the restored database instance as Multi AZ)
   9: db_port (Set the database port)
   10: storage_type (Set the instance storage type)
   11: storage_throughput (Set the instance storage throughput)
   12: iops (Iops value for instance storage)
   13: pitr_from_instance_id (Set the instance identifier that will reference␣
↪the instance to be used to proceed with a Point In Time Recovery restore)
   14: pitr_date (Sets the date for Point in Time Recovery restore)
   15: pitr_latest (Run a Point in Time Recovery restore using the latest␣
↪date available)
   16: parquet_db_socket (Set the socket path to use a socket based␣
↪connection to restore parquet data from the backup directly into a database)
   17: parquet_db_engine (Set the engine (postgresql or mysql) to restore␣
↪parquet data from the backup directly into a database)
   18: parquet_db_host (Set the host to connect and restore parquet data from␣
↪the backup directly into a database)
   19: parquet_db_name (Set the database name to connect and restore parquet␣
↪data from the backup directly into a database)
   20: parquet_db_port (Set the database port to connect and restore parquet␣
↪data from the backup directly into a database)
   21: parquet_db_user (Set the database user to connect and restore parquet␣
↪data from the backup directly into a database)
   22: parquet_db_password (Set the database password to connect and restore␣
↪parquet data from the backup directly into a database)
   23: parquet_save_mode (Set the restore mode for a parquet restore: Append,␣
↪Overwrite, ErrorIfExists or Ignore)
   24: access_key (Set a different access key to access to the destination)
   25: secret_key (Set a different secret key to access to the destination)
   26: region (Set the destination region)
   27: debug (Change debug level)
Select parameter to modify (1-27): 23
Please enter a value for parquet_save_mode: Overwrite
Plugin Restore Options
Option                          Current Value       Default Value
```

(continues on next page)

(continued from previous page)

```
instance_identifier:           *None*              (*none*)
from_snapshot_id:              *None*              (*none*)
instance_class:                *None*              (*none*)
availability_zone:             *None*              (*none*)
parameter_groups:              *None*              (*none*)
vpc_security_groups_ids:       *None*              (*none*)
tags:                          *None*              (*none*)
multi_az:                      *None*              (*none*)
db_port:                       *None*              (*none*)
storage_type:                  *None*              (*none*)
storage_throughput:            *None*              (*none*)
iops:                          *None*              (*none*)
pitr_from_instance_id:         *None*              (*none*)
pitr_date:                     *None*              (*none*)
pitr_latest:                   *None*              (*none*)
parquet_db_socket:             *None*              (*none*)
parquet_db_engine:             mysql               (*none*)
parquet_db_host:               *None*              (*none*)
parquet_db_name:               test                (*none*)
parquet_db_port:               *None*              (*none*)
parquet_db_user:               root                (*none*)
parquet_db_password:           root                (*none*)
parquet_save_mode:             Overwrite           (*none*)
access_key:                    *None*              (*none*)
secret_key:                    *None*              (*none*)
region:                        *None*              (*none*)
debug:                         *None*              (*none*)
Use above plugin configuration? (Yes/mod/no): yes
Run Restore job
JobName:         RestoreFiles
Bootstrap:       /tmp/regress/working/127.0.0.1-dir.restore.3.bsr
Where:           /
Replace:         Always
FileSet:         Full Set
Backup Client:   127.0.0.1-fd
Restore Client:  127.0.0.1-fd
Storage:         File
When:            2024-06-06 14:39:13
Catalog:         MyCatalog
Priority:        10
Plugin Options:  User specified
OK to run? (Yes/mod/no): yes
Job queued. JobId=4
```

Listing 18: **Restore job result**

```
list joblog jobid=4
*list joblog jobid=4
| 127.0.0.1-dir JobId 4: Start Restore Job RestoreFiles.2024-06-06_14.39.14_
↪13                          |
| 127.0.0.1-dir JobId 4: Restoring files from JobId(s) 1                    ␣
```

(continues on next page)

(continued from previous page)

```
↪                          |
| 127.0.0.1-dir JobId 4: Connected to Storage "File" at 127.0.0.1:8103 with␣
↪TLS                         |
| 127.0.0.1-dir JobId 4: Using Device "FileStorage" to read.                ␣
↪                          |
| 127.0.0.1-dir JobId 4: Connected to Client "127.0.0.1-fd" at 127.0.0.1:8102␣
↪with TLS                  |
| 127.0.0.1-fd JobId 4: Connected to Storage at 127.0.0.1:8103 with TLS      ␣
↪                          |
| 127.0.0.1-sd JobId 4: Ready to read from volume "TEST-2024-06-06:0" on File␣
↪device "FileStorage" (/tmp/regress/tmp). |
| 127.0.0.1-sd JobId 4: Forward spacing Volume "TEST-2024-06-06:0" to␣
↪addr=247                        |
| 127.0.0.1-sd JobId 4: Elapsed time=00:00:01, Transfer rate=159.2 K Bytes/
↪second                       |
| 127.0.0.1-fd JobId 4: amazon-rds: Plugin log of this job available in: /tmp/
↪regress/working/amazon-rds/amazon-rds-debug-0.log |
| 127.0.0.1-fd JobId 4: amazon-rds: Jar Version: 1.0.0                      ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Starting backend restore process        ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: No port provided. Using default port for␣
↪mysql:3306                |
| 127.0.0.1-fd JobId 4: amazon-rds: Instance identifier was not set         ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: From snapshot id was not set            ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Pitr latest : false                     ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Pitr date was not set                   ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Pitr from instance id was not set       ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Parquet DB Name: test                   ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Parquet DB User: root                   ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:cdimages with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:cdimages restored                 ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:fileevents with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:fileevents restored               ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:filemedia with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:filemedia restored                ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:restoreobject with 0␣
↪records to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:restoreobject restored            ␣
```

(continues on next page)

(continued from previous page)

```
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:events with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:events restored                    ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:malwaremd5 with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:malwaremd5 restored                ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:mediatype with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:mediatype restored                 ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:file with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:file restored                      ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:pool with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:pool restored                      ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:metaattachment with 0␣
↪records to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:metaattachment restored            ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:location with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:location restored                  ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:status with 28 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:status restored                    ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:counters with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:counters restored                  ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:storage with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:storage restored                   ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:unsavedfiles with 0␣
↪records to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:unsavedfiles restored              ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:path with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:path restored                      ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:client with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:client restored                    ␣
```

(continues on next page)

(continued from previous page)

```
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:jobmedia with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:jobmedia restored                 ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:basefiles with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:basefiles restored                ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:tagclient with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:tagclient restored                ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:log with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:log restored                      ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:malwaresha256 with 0␣
↪records to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:malwaresha256 restored            ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:tagobject with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:tagobject restored                ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:snapshot with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:snapshot restored                 ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:object with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:object restored                   ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:tagjob with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:tagjob restored                   ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:tagmedia with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:tagmedia restored                 ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:device with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:device restored                   ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:jobhisto with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:jobhisto restored                 ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:pathhierarchy with 0␣
↪records to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:pathhierarchy restored            ␣
```

(continues on next page)

(continued from previous page)

```
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:media with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:media restored                    ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:pathvisibility with 0␣
↪records to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:pathvisibility restored           ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:job with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:job restored                      ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:fileset with 0 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:fileset restored                  ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:metaemail with 0 records␣
↪to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:metaemail restored                ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:locationlog with 0␣
↪records to database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:locationlog restored              ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: Restoring table:version with 1 records to␣
↪database: test in mode:overwrite |
| 127.0.0.1-fd JobId 4: amazon-rds: Table:version restored                  ␣
↪                          |
| 127.0.0.1-fd JobId 4: amazon-rds: No more items to restore. Restore ended  ␣
↪                          |
| 127.0.0.1-dir JobId 4: Bacula 127.0.0.1-dir 18.0.3 (22May24):
Build OS:               x86_64-pc-linux-gnu ubuntu 22.04
JobId:                  4
Job:                    RestoreFiles.2024-06-06_14.39.14_13
Restore Client:         "127.0.0.1-fd" 18.0.3 (22May24) x86_64-pc-linux-gnu,
↪ubuntu,22.04
Where:                  /
Replace:                Always
Start time:             06-jun-2024 14:39:16
End time:               06-jun-2024 14:39:27
Elapsed time:           11 secs
Files Expected:         41
Files Restored:         41
Bytes Restored:         118,684 (118.6 KB)
Rate:                   10.8 KB/s
FD Errors:              0
FD termination status:  OK
SD termination status:  OK
Termination:            Restore OK |
| 127.0.0.1-dir JobId 4: Begin pruning Jobs older than 6 months .           ␣
↪                          |
```

(continues on next page)

(continued from previous page)

```
| 127.0.0.1-dir JobId 4: No Jobs found to prune.                              ␣
↪                          |
| 127.0.0.1-dir JobId 4: Begin pruning Files.                                 ␣
↪                          |
| 127.0.0.1-dir JobId 4: No Files found to prune.                             ␣
↪                          |
| 127.0.0.1-dir JobId 4: End auto prune.                                      ␣
↪                          |
```

7.3 List and Query

Enterprise

> Bacula Enterprise Only
>
> This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

It is possible to list database instances or clusters using the bconsole `.ls` command or through a `.query` command.

Any Fileset parameter can be employed in the `plugin=""` value to filter the results, similar to how the backup operates.

Below, there are several examples:

List instances:

Listing 19: **List example**

```
.ls plugin="amazon-rds: instances=jg-pg-aws region=us-east-1 access_
↪key=AKIAQTESTKEY12134g secret_key=m23480ahpqwre894qwrffsfdSecretExample" ␣
↪client=127.0.0.1-fd path=/
Connecting to Client 127.0.0.1-fd at 127.0.0.1:9102
-rw-r-----   1 nobody   nobody                 3223 2025-02-24 18:19:57  /
↪@amazon-rds/jg-pg-aws/jg-pg-aws.i.json
2000 OK estimate files=1 bytes=3,223
```

Query instances:

Listing 20: **Query example: List all instances**

```
.query plugin="amazon-rds: region=us-east-1 access_key=AKIAQTESTKEY12134g␣
↪secret_key=m23480ahpqwre894qwrffsfdSecretExample" client=127.0.0.1-fd␣
↪parameter="instance"
instance=jg-pg-aws
instanceId=jg-pg-aws
jg-pg-aws.dbName=bacula
jg-pg-aws.status=starting
jg-pg-aws.createTime=2024-03-28 10:27:41
jg-pg-aws.latestRestorableTime=2024-06-04 11:14:35
```


Note

The previous query command can be used to retrieve the `latestRestorableTime`, which is useful for any Point-In-Time recovery operation.

List snapshots of a given instance:

Listing 21: **Query example: Get all snapshots of a given instance**

```
*.query client=127.0.0.1-fd plugin="amazon-rds: region=us-east-1 access_
↪key=AKIAQTESTKEY12134g secret_key=m23480ahpqwre894qwrffsfdSecretExample␣
↪instances=jg-pg-aws" parameter=snapshot
snapshot=plugintest-20240527-114110-03
snapshotId=plugintest-20240527-114110-03
plugintest-20240527-114110-03.instanceId=jg-pg-aws
plugintest-20240527-114110-03.status=available
plugintest-20240527-114110-03.createTime=2024-05-27 09:41:25
snapshot=plugintest-20240527-124713-03
snapshotId=plugintest-20240527-124713-03
plugintest-20240527-124713-03.instanceId=jg-pg-aws
plugintest-20240527-124713-03.status=available
plugintest-20240527-124713-03.createTime=2024-05-27 10:48:05
snapshot=rds:jg-pg-aws-2024-05-16-12-30
snapshotId=rds:jg-pg-aws-2024-05-16-12-30
rds:jg-pg-aws-2024-05-16-12-30.instanceId=jg-pg-aws
rds:jg-pg-aws-2024-05-16-12-30.status=available
rds:jg-pg-aws-2024-05-16-12-30.createTime=2024-05-16 12:30:54
snapshot=rds:jg-pg-aws-2024-05-17-12-30
snapshotId=rds:jg-pg-aws-2024-05-17-12-30
rds:jg-pg-aws-2024-05-17-12-30.instanceId=jg-pg-aws
rds:jg-pg-aws-2024-05-17-12-30.status=available
rds:jg-pg-aws-2024-05-17-12-30.createTime=2024-05-17 12:31:00
snapshot=rds:jg-pg-aws-2024-05-24-16-02
snapshotId=rds:jg-pg-aws-2024-05-24-16-02
rds:jg-pg-aws-2024-05-24-16-02.instanceId=jg-pg-aws
rds:jg-pg-aws-2024-05-24-16-02.status=available
rds:jg-pg-aws-2024-05-24-16-02.createTime=2024-05-24 16:02:49
snapshot=rds:jg-pg-aws-2024-05-25-12-30
snapshotId=rds:jg-pg-aws-2024-05-25-12-30
rds:jg-pg-aws-2024-05-25-12-30.instanceId=jg-pg-aws
rds:jg-pg-aws-2024-05-25-12-30.status=available
rds:jg-pg-aws-2024-05-25-12-30.createTime=2024-05-25 12:30:44
snapshot=rds:jg-pg-aws-2024-05-26-12-30
snapshotId=rds:jg-pg-aws-2024-05-26-12-30
rds:jg-pg-aws-2024-05-26-12-30.instanceId=jg-pg-aws
rds:jg-pg-aws-2024-05-26-12-30.status=available
rds:jg-pg-aws-2024-05-26-12-30.createTime=2024-05-26 12:30:38
snapshot=rds:jg-pg-aws-2024-05-27-09-41
snapshotId=rds:jg-pg-aws-2024-05-27-09-41
rds:jg-pg-aws-2024-05-27-09-41.instanceId=jg-pg-aws
rds:jg-pg-aws-2024-05-27-09-41.status=available
rds:jg-pg-aws-2024-05-27-09-41.createTime=2024-05-27 09:41:25
```

(continues on next page)

(continued from previous page)

```
snapshot=rds:jg-pg-aws-2024-05-27-10-47
snapshotId=rds:jg-pg-aws-2024-05-27-10-47
rds:jg-pg-aws-2024-05-27-10-47.instanceId=jg-pg-aws
rds:jg-pg-aws-2024-05-27-10-47.status=available
rds:jg-pg-aws-2024-05-27-10-47.createTime=2024-05-27 10:48:05
snapshot=rds:jg-pg-aws-2024-06-03-11-41
snapshotId=rds:jg-pg-aws-2024-06-03-11-41
rds:jg-pg-aws-2024-06-03-11-41.instanceId=jg-pg-aws
rds:jg-pg-aws-2024-06-03-11-41.status=available
rds:jg-pg-aws-2024-06-03-11-41.createTime=2024-06-03 11:41:04
```

Note

The previous query command can be used to retrieve the snapshot ids currently available in the cloud. They can be used to restore the instance to that particular state by using the `from_snapshot_id` restore parameter.

8 Limitations

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The following article presents limitations of Amazon RDS Plugin.

When the export to S3 function is not used, Amazon RDS plugin will not store actual data in Bacula volumes, as it will merely function as a Snapshot manager within the AWS cloud.

In instances where a new Amazon RDS instance or cluster is desired for restoring purposes, the source Snapshot must be present in the Amazon RDS service to be selected for use.

Point-in-Time Recovery function is also based on the existing Amazon RDS configuration and previous snapshots. To use this feature, is necessary to enable the Amazon RDS "Automated Backups" function and establish a retention period, even if other snapshots are managed through Bacula Enterprise Amazon RDS Plugin. With that function enabled, RDS service will upload transaction logs for database instances to Amazon S3 every five minutes. To see the most recent restorable time for a database instance, one can employ the `.query` command with the `instance` parameter command and examine the value returned in the `LatestRestorableTime` field.

The export function to S3, and consequently, the backup with the plugin using Apache Parquet format, contains **only the data** of the protected databases. Indexes and any additional information are excluded from both the export and the backup stored in Bacula.

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.

8.1 Cloud Costs

As you may already be aware, storing data in the cloud incurs additional expenses.

Data transfer needs to be considered as well. While uploading data is typically free or incurs minimal costs, downloading it is usually not free, and charges will apply based on each operation and the volume of data transferred.

Amazon employs a pricing model for each of its storage tiers. Additionally, costs will differ depending on the region you use. More information are to be found here:

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

Exporting an Amazon RDS backup to S3 and subsequently downloading it will result in certain charges. Keep it in mind and schedule this operation according to a frequency that aligns with your needs and convenience.

9 Troubleshooting

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

In this article, there are suggested solutions to common situations that can cause trouble during the usage of the Amazon RDS Plugin.

9.1 Out of Memory

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

If you ever face *OutOfMemory* errors of the Java daemon (which can be found in the `amazon-ec2-debug.err` file), you are very likely using a high level of concurrency through internal `concurrent_threads` parameter and/or parallel jobs.

To address this issue, you may consider the following options:

a) Reduce `concurrent_threads` parameter.

b) Reduce the number of jobs running in parallel.

c) If neither of the above is feasible, you should increase the JVM memory (you may also need to increase the underlying host physical memory)

To increase the JVM memory, you will need to:

Create the following file: `/opt/bacula/etc/amazon_rds_backend.conf`.

Below, an example of the contents:

Listing 22: **Limits file contents**

```
AMAZON_RDS_JVM_MIN=2G
AMAZON_RDS_JVM_MAX=8G
```

Those values will define the MIN (`AMAZON_RDS_JVM_MIN`) and MAX (`AMAZON_RDS_JVM_MAX`) memory values assigned to the JVM Heap size. Exercise caution if you are running jobs in parallel, as large values and several jobs at once could quickly consume all available memory on your host.

The `/opt/bacula/etc/AMAZON_RDS_backend.conf` will remain unchanged during package upgrades, ensuring that your memory configurations are retained.

9.2 Throttling

Enterprise

Bacula Enterprise Only

This solution is **only** available for Bacula Enterprise. For subscription inquiries, please reach out to sales@baculasystems.com.

The API usage in the Bacula Enterprise Amazon RDS Plugin is generally less intensive compared to certain other plugins, as the operations (such as make snapshot, remove snapshot, list instances) are of a higher level, and the number of these operations is not expected to reach the boundaries. Nevertheless, it is advisable to review limits of this service at the following link: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Limits.html

If you ever experience throttling, the recommended solution is to reduce concurrency with one of the different strategies mentioned in the *Best Practices* section.

