Bacula Enterprise Configuration

Bacula Systems Documentation

Contents

Contents

The following article aims at presenting various information on how to configure Bacula Enterprise.

1 Best Practices

As a particularly reliable, scalable and powerful backup and recovery solution, Bacula Enterprise offers great flexibility and often, a variety of ways for the user to implement Backup and Restore strategies, policies and regulatory requirements to the very highest standards. Best Practice implementation is always highly recommended by **Bacula Systems**, and Bacula Enterprise's tools, features and architecture are designed to facilitate that goal. This document draws from feedback from our many customers as well as from our developers to guide you in Best Practices covering a large range of Bacula's capabilities to help you exploit your Bacula Enterprise infrastructure to its maximum potential.

Bacula Systems remains at your service if you need any help or validation of your implementation.

Tip: If you need clarification on any concepts within this document, we recommend that you read the following documents in Bacula Enterprise documentation:

- Fundamentals
- Bacula Enterprise Installation

1.1 Actions to Avoid

This chapter covers a few common mistakes that we have seen users making in our many years of experience in Customer Support. Read everything carefully and adapt your own Bacula usage according to your environment. It will save you a lot of trouble down the road.

Naming Bacula Resources

Do not rename resources after you have started production or tests. Renaming will cause side effects, and you will incur a lot of manual Catalog maintenance. This is because renamed resources will get new IDs in the database, new jobs will use those new IDs and will mix them up with older and previous used IDs for Volumes, Storage, Pools, Media Type, etc. To correct this you will need to adapt and replace the IDs inside the Catalog to prevent a mismatch.

For instance: if you change characters in names from lower to upper case while keeping the same resource names, the resource selection can become unpredictable and older resources (with the previous ID) might be used instead. So please carefully name and make modifications to resources.

The best way to avoid problems in this respect is to start with a well designed and strictly adhered to naming scheme.

Read this article about resource naming.

Permissions for Starting Bacula Daemons

When you start daemon processes with incorrect permissions the output files that are created will have the wrong permissions. Side effects include files that cannot be accessed any more the next time you start up the daemons. The same is true for the PID files that are used by the processes, Catalog dumps, log files, or bootstrap files. If they cannot be accessed/created/removed, it will create unexpected behavior.

Be sure to start your Director or Storage Daemon with user and group `bacula`.

Best practice is to use one of the provided services scripts to start and stop Bacula Daemons:

`systemctl start bacula-dir` # on systems that use systemd `service bacula-dir start` `/opt/bacula/bin/bacula-dir -t -u bacula -g bacula`

Note: It is recommended to always test the configuration syntax with the `-t` parameter prior to launch a daemon.

Modifying Scripts from Bacula Installation Packages

You are free to adapt your settings for your Bacula installation and improve scripts. You should copy existing scripts or configuration parts and choose a different name. When you update your Bacula installation packages, modified scripts may be overwritten with default files from the package maintainer and your changes would be lost.

Media Types for Different Storage Locations

You need to use unique Media Types for different Storage Daemons (SDs) and storage locations (*Archive Device* file paths) on the same SD. Please use a distinct "Archive Device" that will not be used by any other device. Exceptions are AutoChanger devices that will point to the same Archive Device. In particular for Global Endpoint Deduplication (GED) the Media Type of volumes should never be shared with the Media Type for non-deduplicating volumes.

For further reading, see Disk Backup Design (especially the conclusions at the end).

Using Media Types in Pools

The Storage Device will be identified internally via the "Media Type" when a job is processed. The Media Type needs to match, so renaming it in the Storage Device definition will mix up newly created volumes with volumes that were created with the old settings. Strong side effects will show up, as you will not be able to recycle, prune, truncate older volumes any more. A storage device might refuse to work or a backup job will request new or different volumes. Best practice is to avoid renaming "Media Type" values in the Storage definition and use either the Storage or the Storage Group directives in your pool resources, so Bacula is aware of the Storage to be used for all the media types in the Pool resource.

Note: Never rename a Media-Type for a storage device after you have already created and labeled volumes in a Pool. Alternatively create a new Pool and use a storage with a new Media Type. Add new Volumes and reconfigure your Jobs to use the new Pool instead. After Volume retention time has passed for the old pool, those Volumes can be removed, step by step. Finally remove the old Pool, when no Volumes remain inside.

Block Size of Tape Devices

You must test your tape device with `btape` before you go into production (See section sec:tapecheck). The "Minimum Block Size" should almost *never* be used. It will just waste tape space.

However, to avoid Kernel problems handling the massive request for memory, a "Maximum Block Size" of 512 KiB is the maximum we advise to use.

Do not change those settings afterwards, otherwise your media will become incompatible and you will encounter many read- and write-errors.

Example for 512k:

```
Device {
  Name = ParisTapeLibrary1-LTO6-Drive01
  MaximumBlockSize = 524288 # 512k to be tested with 64,128,256k before
  ...
}
```


Using Tape Libraries with a Barcode Reader

If you are using a Tape Autochangers with barcode reader please see sec:labeltapes.

Handle Multiple Catalogs

Multiple Catalogs can be configured with Bacula Enterprise, however we strongly recommend not to do so, nor do we support it.

Maintaining a Valid Bootstrap File

By using a **WriteBootstrap** record in each of your Director's Job resources, you can constantly maintain a file that will enable you to recover the state of your system as of the last backup without having the **Bacula** catalog. This permits you to more easily recover from a disaster that destroys your **Bacula** catalog.

When a Job resource has a **WriteBootstrap** record, **Bacula** will maintain the designated file (normally on another system but mounted by NFS) with up to date information necessary to restore your system. For example, in my Director's configuration file, I have the following record:

```
Write Bootstrap = "/mnt/deuter/files/backup/client-name.bsr"
```

where I replace **client-name** by the actual name of the client that is being backed up. Thus, **Bacula** automatically maintains one file for each of my clients. The necessary bootstrap information is appended to this file during each **Incremental** backup, and the file is totally rewritten during each **Full** backup.

If you are starting off in the middle of a cycle (i.e. with Incremental backups) rather than at the beginning (with a Full backup), the **bootstrap** file will not be immediately valid as it must always have the information from a Full backup as the first record. If you wish to synchronize your bootstrap file immediately, you can do so by running a `restore` command for the client and selecting a full restore, but when the `restore` command asks for confirmation to run the restore Job, you simply reply no, then copy the bootstrap file that was written to the location specified on the **Write Bootstrap** record. The restore bootstrap file can be found in **restore.bsr** in the working directory that you defined. In the example given below for the client **rufus**, my input is shown in bold. Note, the JobId output has been partially truncated to fit on the page here:

```
(in the Console program)
*restore
First you select one or more JobIds that contain files
to be restored. You will then be presented several methods
of specifying the JobIds. Then you will be allowed to
select which files from those JobIds are to be restored.
To select the JobIds, you have the following choices:
     1: List last 20 Jobs run
     2: List Jobs where a given File is saved
     3: Enter list of JobIds to select
     4: Enter SQL list command
     5: Select the most recent backup for a client
     6: Cancel
Select item: (1-6): 5
The defined Client resources are:
     1: Minimatou
     2: Rufus
```

(continues on next page)

(continued from previous page)

```
     3: Timmy
Select Client (File daemon) resource (1-3): 2
The defined Fileset resources are:
     1: Other Files
Item 1 selected automatically.
 +-------+------+-------+---------+---------+------+-------+------------+
 | JobId | Levl | Files | StrtTim | VolName | File | SesId | VolSesTime |
 +-------+------+-------+---------+---------+------+-------+------------+
 | 2     | F    | 84    | ...     | test1   | 0    | 1     | 1035645259 |
 +-------+------+-------+---------+---------+------+-------+------------+
 You have selected the following JobId: 2
 Building directory tree for JobId 2 ...
 The defined Storage resources are:
      1: File
 Item 1 selected automatically.
 You are now entering file selection mode where you add and
 remove files to be restored. All files are initially added.
 Enter "done" to leave this mode.
 cwd is: /
 $ done
 84 files selected to restore.
 Run Restore job
 JobName: kernsrestore
 Bootstrap: /home/user1/bacula/working/restore.bsr
 Where: /tmp/bacula-restores
 Fileset: Other Files
 Client: Rufus
 Storage: File
 JobId: *None*
 OK to run? (yes/mod/no): no
 quit
 (in a shell window)
 cp ../working/restore.bsr /mnt/deuter/files/backup/rufus.bsr
```

1.2 Security Considerations

- Only the File daemon needs to run with root permission (so that it can access all files). As a consequence, you may run your Director, Storage daemon, and MySQL or PostgreSQL database server as non-root processes. There are **-u** and the **-g** options that allow you to specify a userid and groupid on the command line to be used after **Bacula** starts.
- You should protect the **Bacula** port addresses (normally 9101, 9102, and 9103) from outside access by a firewall or other means of protection to prevent unauthorized use of your daemons.
- You should ensure that the configuration files are not world readable since they contain passwords that allow access to the daemons. Anyone who can access the Director using a console program can restore any file from a backup Volume.
- You should protect your Catalog database. Please note that the **Bacula** setup procedure leaves the database open to anyone. At a minimum, you should assign the user `bacula` a userid and add it to your Director's configuration file in the appropriate Catalog resource.

- If you use the make_catalog_backup script provided by **Bacula**, remember that you should take care when supplying passwords on the command line.

1.3 Concurrent Jobs

The concurrent job runs can be managed by modifying the **Maximum Concurrent Jobs** in the Director's configuration file (**bacula-dir.conf**) in the Director, Job, Client, and Storage resources.

Additionally the File daemon, and the Storage daemon each have their own **Maximum Concurrent Jobs** directive that sets the overall maximum number of concurrent jobs the daemon will run.

For example, if you want two different jobs to run simultaneously backing up the same Client to the same Storage device, they will run concurrently only if you have set **Maximum Concurrent Jobs** greater than one in the Director resource, the Client resource, and the Storage resource in **bacula-dir.conf**.

Below is a stripped down **bacula-dir.conf** file showing you the four places where the the file must be modified to allow the same job **NightlySave** to run up to four times concurrently. The change to the Job resource is not necessary if you want different Jobs to run at the same time, which is the normal case.

```
#
# Bacula Director Configuration file -- bacula-dir.conf
#
Director {
    Name = rufus-dir
    Maximum Concurrent Jobs = 4
    ...
}
Job {
    Name = "NightlySave"
    Maximum Concurrent Jobs = 4
    Client = rufus-fd
    Storage = File
    ...
}
Client {
    Name = rufus-fd
    Maximum Concurrent Jobs = 4
    ...
}
Storage {
    Name = File
    Maximum Concurrent Jobs = 4
    ...
}
```


1.4 Software Compression

For software compression, Bacula backup Jobs can be configured to use **ZIP** (levels 1 to 9, default is 6), **LZO** (the level LZ01X) or **ZSTD** (levels: zstd1, zstd3, zstd10 and zstd19) compression.

This is done on a file by file basis by the File Daemon, before network transit. **LZO** provides much faster compression and decompression speed but lower compression ratios than **GZIP**.

Refer to the Bacula Enterprise Main manual, Configuring the Director chapter, The Fileset resource section, `compression=...`, configured within an Options resource, for more information.

ZLIB-compressed data written by a 64-bit machine may not always uncompress correctly on a 32-bit machine.

Disable Compression for Specific Storage Resource

The Fileset compression configuration can be overridden by the `AllowCompression = no` directive configured on the Storage resource definition. It means a Fileset with `compression = ...` directive will not compress the data backed up to a Storage resource that has configured `AllowCompression = no` (the default is `yes`). This way, usage of software compression with tape drives can be avoided.

Refer to the Configuring the Director chapter, the Storage resource section, `AllowCompression` directive for more information.

Fileset Example

Software compression can be achieved by means of the **compression=...** directives in the Fileset resource. The following example configures GZIP1 compression for the `/home` directory in the "MyFileset" Fileset:

```
Fileset {
   Name = "MyFileset"
   Include {
      Options {
         wildfile = "*.gz"
         exclude = yes
      }
      Options {
         compression = GZIP1
         signature = SHA1
      }
      File = /home
   }
}
```

It is strongly recommended to run tests with Jobs/Filesets and some possible compression configurations with a set of current file types and assess which kind of compression is the most suitable for your environment. The **bconsole** `estimate` command gives the number of bytes that would be backed up with no compression, as it does not actually process data.

1.5 Testing and Monitoring

Testing Backup and Restore

Verify Configuration

After you have created a new Job, it is strongly recommended to test it. Run the backup and then restore its data as a test to see that everything is correctly configured and thus can be documented in your disaster recovery procedures.

Important information can be gathered during this test:

- How long does the Job take?
- What is the load on the client, the network and the Storage Daemon?
- Does the Job run successfully?
- Can the data be restored?

How to Test

In order to test your backup Job, from `bconsole` simply type:

```
run job=MyNewJob
```

and follow the on-screen options.

Once the Job has finished successfully, type again into the console:

```
restore
```

Note:

- It is very important to run a test backup and restore to see the impact on the SD/network but also to ask for help if issues arise during such tests.
- You can also use a Verify Job to verify your backup

Config Files, Version Control and Test before Reload/Restart

You will surely make a lot of modifications over time to your configuration files in `/opt/bacula/etc`, that's why it is very good practice to:

- include this folder in your backup policy
- and/or put this folder under a revision control system of your choice eg. CVS/SVN/GIT/Mercurial

This will not only ensure the integrity of your configuration, especially if there are several system administrators involved in the backup process, but also allow to revert changes to a known good state.

Backup Job Configuration

Here is a simple Fileset/Job definition to do the configuration backup

```
Fileset {
   Name = "DisasterRecovery-fs"
   Include {
      Options {
         signature = SHA1
      }
      File = /opt/bacula/working/bacula.sql # where the Bacula catalog dump␣
↪goes
      File = /opt/bacula/etc
      # you can add other files like keys, content of /etc to make this␣
↪Fileset
      # more complete and adapted to your environment
   }
}

Job {
   Name = "DisasterRecovery-job"
   Type = "Backup"
   Client = "baculaServer"   # change to the name of the fd running on the␣
↪Bacula DIR
   Fileset = "DisasterRecovery-fs"
   JobDefs = "Default-jd"
   Level = "Full"  # full backup is preferable
   Messages = "Standard"
   Pool = "DisasterRecovery-pool" # the pool we just defined to hold all␣
↪config and catalog dumps
   Priority = 15 # Adjust to your priorities so this job runs last, after all␣
↪jobs of your backup window
   Runscript {
      Command = "/opt/bacula/scripts/make_catalog_backup bacula bacula"
      RunsOnClient = no
      RunsWhen = Before
   }
   Runscript {
      Command = "/opt/bacula/scripts/delete_catalog_backup"
      RunsOnClient = no
      RunsWhen = After
   }
   Schedule = "NightAfterJobs"
   Storage = "OnDisk"
   WriteBootstrap = "/opt/bacula/bsr/catalog-backup.bsr" # important, can be␣
↪sent by email
}
```

Check for Configuration Errors

After each modification, always check the configuration to avoid issues when reloading/restarting the Bacula Director process:

```
# /opt/bacula/bin/bacula-dir -t -u bacula -g bacula
```

And correct errors that are displayed or contact Support.

Note: The Director **will not start** if there are parsing errors in your configuration. Changed configuration will bot be applied if you use `reload` in your **bconsole** while there are errors in the configuration. Test your configuration and modifications. You might use the `breload` Bacula Enterprise command which will test your configuration, make a compressed archive and reload the director configuration as well. It is a convenient way to ensure you reload the configuration only if it is well formed.

Testing Tape Autochanger/Drives

See TestTapeDriveDeviceBtapeUtility.

Testing Tape Autochanger for the Best Performance

Testing the tape autochanger is a very important task in every Bacula setup and it should be done before running any production backup because it will:

- identify any connectivity, hardware or configuration issues preventing data to be backed up efficiently and safely
- assure the best performance of Bacula Enterprise working with the tested tape autochanger
- confirm the best settings so no other changes will be required that would make the overall Bacula Enterprise configuration more complex.

Testing a Single Device or Tape Autochanger

Tests should be performed with the `btape` utility to verify the Autochanger configuration and if `m` and `mtx` commands are running correctly. Preferably, the `btape` tests are done before going to production. Additionally, a special `speed` test will perform raw data and random data tests on your tape device with your current configuration to test the performance of your device.

Performing a Writing/Reading Device Check

Before running the `btape` tests, a working SD configuration to connect to the Tape Library must be functional. Please refer to vendor and operating system manuals in order to install this. The Storage Daemon connected to the Tape Library should be shutdown before running the `btape` test, so no backup jobs will interfere. As an example, the first tape drive (LTO-drive0) will be used for this test.

Note: A blank and empty tape that has not been used before with other third part vendor or legacy backup tools, or which has been blanked manually must be used for these tests.

The commands used below are examples only, you will need to adapt them to your current settings and environment.

```
systemctl stop bacula-sd
```

Then, the status of the tape library should be reviewed:

```
mtx -f /dev/tape/by-serial/changer-device status
```

The tape drive to test, for example the "Data Transfer Element 0", should be empty. If not, please unload it with `mtx` or your tape library interface. Then, a tape should be loaded in the Tape Drive (Drive 0, index=0 in the `Device{}` resource in `bacula-sd.conf` or in the related resource) that will be tested with a new tape media (eg. from slot 22 in the command below):

```
mtx -f /dev/tape/by-serial/changer load 22 0
```

Then, `btape` should be run:

```
/opt/bacula/bin/btape -v -c /opt/bacula/etc/bacula-sd.conf LTO-drive0
```

where "LTO-drive0" is the name of the tape drive to test in your SD configuration, at drive index 0 in this example.

The actual testing routine can be started with

```
test
```

It will create some output which needs to be carefully assessed.

If any error happens with this first test, it must be fixed. Once fixed, please continue with the next step, performing a speed check (The `file_size` parameter is important in order to write a file bigger than **Maximum File Size** defined in the `Device{}` resource).

```
speed file_size=20
```

Two directives can be fine tuned in a device configuration so it leads to a faster writing process to a tape drive.

Maximum File Size

For LTO-5 tapes, a value between 8GiB and 12GiB is indicated and between 8GiB and 24GiB for LTO-6; in general, with higher capacity, higer speed tape drives, the file sizes will be increased. The larger this logical file size is, the longer it will take to restore a single file as the tape will need to read more unneeded data before reaching the desired single file. However, backups will be faster as less End-of-File marks need to be written.

Maximum Block Size

The following values should be tested: 128K, 256K, 512K. The largest value should not be *exceeded* in order to avoid kernel problems. For LTO-5 and up, 256K and 512K are usually appropriate.

Note:

- If the **Maximum Block Size** settings changes to a lower value after tapes have been written with production data, then your media will become incompatible and *I/O-ERRORS* will happen.
- **Minimum Block Size** should *never* be used, as it will just waste tape space. If you think you should use it, please contact us.

- Organize your tests through pairs of directives, for example **Minimum File Size = 8GiB**; **Maximum Block Size = 131072**, **Maximum File Size = 8GiB**; **Maximum Block Size = 262144**, etc. (8GiB-524288, 12GiB-131072, 12GiB-262144, 12GiB-524288, etc.)
- When the test plan is defined, both Directives should be changed accordingly in the `Device{}` Resource of the Storage Daemon configuration file for the chosen drive **and all other drives that can share media**.

  Once modified, please run the **btape** `speed` tests according the **Minimum File Size of your selected pair. After each test with a pair you will need to restart the **btape** utility to load the configuration changes. Please find this example with a **Maximum File Size** given in GiB:

```
speed file_size=20
```

- The `file_size` value should be much larger than the `Maximum file Size` **directive**. In order to find the best settings for a production environment, it will be necessary to observe reported throughput of the "Test with zero data and Bacula block structure" and the following "Test with random data, should give the minimum throughput" from all test runs.

  Find more details about the **btape** command in our Bacula Enterprise Utility programs in your download area or by contacting us.

1.6 Reporting Issues

The Bacula Systems Support Team is available for your questions and for any configuration or product issue. Please do not hesitate to open a support ticket with us.

The more detailed your ticket is, the faster the Bacula support engineers can help you.

Keep it to one issue per ticket. If you have several questions or problems, open separate tickets for each of them.

Issue Description

In case you encounter an issue, it is important to open a support case with Bacula Systems Support Services with as much relevant information as possible about the erroneous Job or the issue itself. For a Job in error:

- Configuration of the Job, ideally a bsys_report from the Director and the Storage Daemon if impacted. Please see bsys_report Generation Instructions below.
- The complete joblog (backup and restore) with timestamps
- A description of how this Job behaved in the past
- The impact of this error on your backup production infrastructure

Minimum Logs to Attach to Issue

In order to display the complete joblog, from **bconsole** simply type:

```
@tee /tmp/joblog-XXXX
llist joblog jobid=XXXX
@tee
```

- Where XXXX is the jobid of the job in error.
- You can now open a support case and simply attach or copy/paste the content of the file `/tmp/joblog-XXXX`.
- If you use a plugin, there are possibly complementary logs in a dedicated `/opt/bacula/working` subdirectory.
- If you experience a crash of a Bacula service (Director/File Daemon/Storage Daemon), there are `.lockdump` and `.traceback` files located in `/opt/bacula/working` that are important for our development team to analyze to find the root cause of the issue. Please add them to the ticket.
- These lockdump and traceback files are generated only if the typical platform debugger, typically **gdb** is installed and can attach to a crashing process. Both is not, by default, the case on a reasonably hardened system, so it may be necessary to install and configure some dependencies.

BSys_report Generation

The bsys_report generator script will produce a report of your Bacula configuration and machine set up. The bsys_report may be generated in one of two ways:

- BWeb Management Suite (Web GUI)
- Command Line (shell prompt)

From BWeb Management Suite

From the BWeb Main Menu, click `Configuration -> BWeb Configuration -> Generate BSYS Report` (in the right pane of the page).

Note: If you have SELinux set to enforce, note that BWeb cannot generate bsys_report, thus you must use the bsys_report generator presented in the next chapter: *Linux Shell Command Line*.

From the Linux Shell Command Line

We update this bsys_report script frequently, so please be sure to download a current one from this link:

```
curl -o bsys_report.tar.gz https://www.baculasystems.com/ml/bsys_report/bsys_
↪report.tar.gz
```

or if `curl` is not installed

```
wget https://www.baculasystems.com/ml/bsys_report/bsys_report.tar.gz
```

Unpack it:

```
tar xvzf bsys_report.tar.gz
```

Set it executable:


```
chmod +x bsys_report.pl
```

As either the bacula or root user on your director (DIR) and storage daemon (SD) machines run the command:

```
./bsys_report.pl
```

Then submit the generated report files in a support ticket so we can review them and better understand your Bacula environment.

Note: More information about bsys_report can be obtained by running it with the –help command-line switch:

```
./bsys_report.pl --help
```

Bsys Report Collector Script

It can be helpful in large Bacula environments to get all the reports needed by the Bacula Systems Support team to help understand your environment and to diagnose issues. Manually uploading the bsys_report.pl script to the Director, and each SD, then running the script on each server and finally downloading the results from each server can be tedious and time consuming.

The *get_bsys_reports.py* script is a Python script which allows a Bacula administrator to quickly retrieve bsys reports from the Director server and one, several, or all of your Storage servers with one command.

When the script is finished downloading all the bsys reports, they are tarred up into one single file so they may be sent to the Support team easily.

The latest version of this script may be found on Github here: https://github.com/Bacula-Systems

```
waa@x1carbon get_bsys_reports (main)$ ./get_bsys_reports.py --dir speedy-file -m TestMask

- Script starting...
  - Option '--dir' provided on command line. Will attempt to get report from the Director.

  - The following Storage/Autochanger resource was provided on the command line: speedy-file
    - Checking validity of provided Storage/Autochanger resource.

  - Getting list of all Storage/Autochanger resources from the Director.
    - Found the following Storage/Autochanger resources configured in the Direcor: None, speedy-file, speedy-syn, mysql-FileChgr1
      - Storage "speedy-file" is valid.

  - Determining IP address for Storage resource and creating unique host list.
    - Director: bacula-dir, Address: x1carbon.revpol.com
      - x1carbon.revpol.com is not an IP address. Attempting to resolve...
        - FQDN/host x1carbon.revpol.com = 10.1.1.4
        - Adding Director's IP address 10.1.1.4 to list of hosts to retrieve report from.
    - Storage: speedy-file, Address: 10.1.1.4
      - 10.1.1.4 is an IP address
      - IP address for Storage "speedy-file" (10.1.1.4) already in list. Skipping...

  - Attempting to retrieve report from server with IP address: 10.1.1.4
    - Working on host: 10.1.1.4 (Director)
      - Setting up ssh connection
        - Done
      - Uploading ./bsys_report.pl to 10.1.1.4:/tmp
        - Done
      - Running 10.1.1.4:/tmp/20220523163211_bsys_report.pl -o /tmp
        - Done
      - Retrieving report 10.1.1.4:/tmp/bsys_report.2022-05-23.x1carbon.3009883.gz
        - Done

  - Only one bsys report retrieved.
    - Not creating a tarball of one bsys report file.
    - Prepending bsys report filename with ticket mask "TestMask"

  - The one bsys report file is here: /tmp/all_bsys_reports-nbf25yfl/TestMask_Director-10.1.1.4-bsys_report.2022-05-23.x1carbon.3009883.gz
- Script complete.
```

Fig. 1: Example output of the script getting a report just from the Director server. Notice that the *-m* command line option was used to automatically prepend a ticket mask(or a company name) to the name of the tar file, or in this case, the name of the one report retrieved.

1.7 Catalog Database Choice

We recommend a **PostgreSQL** backend to run Bacula Enterprise. Please follow the advised settings you find in the psql-tuning to obtain the best performance and security for your catalog.

To change the catalog password, please run these commands:

```
# su - postgres
$ psql
postgres=# alter user bacula with password'your_new_password';
postgres=# \q
```

Then you must put the new password in `/opt/bacula/etc/bacula-dir.conf` as shown below:

```
Catalog {
  Name = MyCatalog
  DBname = "bacula"
  DBuser = "bacula"
  DBpassword = "your_new_password"
}
```

Restart bacula-dir service:

```
systemctl restart bacula-dir
```

1.8 Items to Implement Before Production

We recommend you take your time before implementing a production a Bacula backup system since Bacula is a rather complex program, and if you make a mistake, you may suddenly find that you cannot restore your files in case of a disaster. This is especially true if you have not previously used a major backup product.

If you follow the instructions in this section, you will have covered most of the major problems that can occur. It goes without saying that if you ever find that we have left out an important point, please inform us, so that we can document it to the benefit of everyone.

Critical Items

The following assumes that you have installed Bacula, you more or less understand it, and that you have set up a basic production configuration. If you haven't done the above, please do so and then come back here. The following is a sort of checklist that points with perhaps a brief explanation of why you should do it. In most cases, you will find the details elsewhere in the documentation. The order is more or less the order you would use in setting up a production system (if you already are in production, use the checklist anyway).

- Test your tape drive for compatibility with Bacula by using the **test** command of the `btape` utility (see the btape section).
- Better than doing the above is to walk through the nine steps in the of the Tape Testing section. It may take you a bit of time, but it will eliminate surprises.
- Test the end of tape handling of your tape drive by using the **fill** command of the `btape` utility (see the btape section).

- Do at least one restore of files. If you backup multiple OS types (Linux, Solaris, MacOS, FreeBSD, Windows, etc.), restore files from each system type. The Restoring Files section shows you how.
- Write a bootstrap file to a separate system for each backup job. The **Write Bootstrap** directive is described in the Director Configuration section, and more details are available in the Bootstrap File section. Also, the default **bacula-dir.conf** comes with a **Write Bootstrap** directive defined. This allows you to recover the state of your system as of the last backup.
- Backup your catalog. An example of this is found in the default **bacula-dir.conf** file. The backup script is installed by default and should handle any database, though you may want to make your own local modifications. See also Backing Up Your Bacula Database - Security Considerations for more information.
- Write a bootstrap file for the catalog. An example of this is found in the default **bacula-dir.conf** file. This will allow you to quickly restore your catalog in the event it is wiped out – otherwise it is many excruciating hours of work.
- Make a copy of the **bacula-dir.conf**, **bacula-sd.conf**, and **bacula-fd.conf** files that you are using on your server. Put it in a safe place (on another machine) as these files can be difficult to reconstruct if your server dies.
- Prepare tools and documentation in case of disaster recovery! See DisasterRecovery.
- Bacula assumes all filenames are in UTF-8 format. This is important when saving the filenames to the catalog. For Windows machine, Bacula will automatically convert from Unicode to UTF-8, but on Unix, Linux, *BSD, and MacOS X machines, you must explicitly ensure that your locale is set properly. Typically this means that the **LANG** environment variable must end in **.UTF-8**. A full example is **en_US.UTF-8**. The exact syntax may vary a bit from OS to OS, and exactly how you define it will also vary. On most modern Windows machines, you can edit the configuration files with **notepad** and choose output encoding UTF-8.

Recommended Items

Although these items may not be critical, they are recommended and will help you avoid problems.

- Read the Fundamentals section.
- Learn what each of the main Bacula Utility Programs does (described in the Tape Volume Management chapter and in the dbcheck chapter).
- Set up reasonable retention periods so that your catalog does not grow to be too big. See the following three sections:
  - Recycling your Volumes,
  - Basic Volume Management,
  - Using Pools to Manage Volumes.

2 Customizing the Configuration Files

When each of the **Bacula** programs starts, it reads a configuration file specified on the command line or the default **bacula-dir.conf**, **bacula-fd.conf**, **bacula-sd.conf**, or **console.conf** for the Director, the File Daemon, the Storage Daemon, and the Console program respectively.

Each service (Director, Client, Storage, Console) has its own configuration file containing a set of Resource definitions. These resources are very similar from one service to another, but may contain different directives (records) depending on the service. For example, in the Director's resource file, the **Director** resource defines the name of the Director, a number of global Director parameters and his password. In the File Daemon configuration file, the **Director** resource specifies which Directors are permitted to use the File Daemon.

Before running **Bacula** for the first time, you must customize the configuration files for each daemon. Default configuration files will have been created by the installation process, but you will need to modify them to correspond to your system.

2.1 Detailed Information for Each Daemon

The details of each Resource and the directives permitted therein are described in the following chapters.

The following configuration files must be defined:

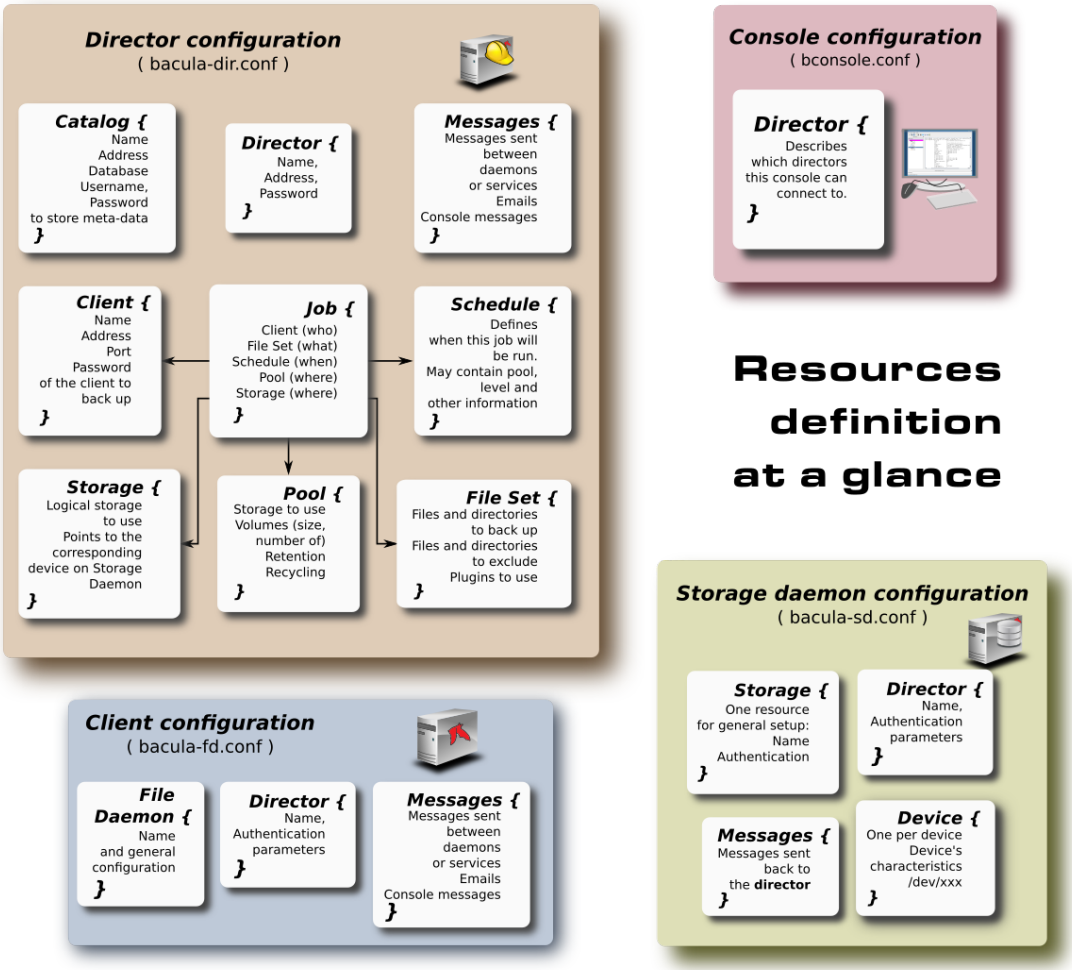
- Console – to define the resources for the Console program (user interface to the Director). It defines which Directors are available so that you may interact with them.
- Director – to define the resources necessary for the Director. You define all the Clients and Storage daemons that you use in this configuration file.
- File Daemon – to define the resources for each client to be backed up. That is, you will have a separate Client resource file on each machine that runs a File daemon.
- Storage – to define the resources to be used by each Storage daemon. Normally, you will have a single Storage daemon that controls your tape drive or tape drives. However, if you have tape drives on several machines, you will have at least one Storage daemon per machine.

2.2 Character Sets

Bacula is designed to handle most character sets of the world, US ASCII, German, French, Chinese, etc. However, it does this by encoding everything in UTF-8, and it expects all configuration files (including those read on Win32 machines) to be in UTF-8 format. UTF-8 is typically the default on Linux machines, but not on all Unix machines, nor on Windows, so you must take some care to ensure that your locale is set properly before starting **Bacula**.

To ensure that **Bacula** configuration files can be correctly read including foreign characters the **LANG** environment variable must end in **.UTF-8**. An full example is **en_US.UTF-8**. The exact syntax may vary a bit from 0S to OS, and exactly how you define it will also vary. On most newer Win32 machines, you can use **notepad** to edit the conf files, then choose output encoding UTF-8.

Bacula assumes that all filenames are in UTF-8 format on Linux and Unix machines. On Win32 they are in Unicode (UTF-16), and will be automatically converted to UTF-8 format.



2.3 Resource Directive Format

Although, you won't need to know the details of all the directives, a basic knowledge of **Bacula** resource directives is essential. Each directive contained within the resource (within the braces) is composed of a keyword followed by an equal sign (=) followed by one or more values. The keywords must be one of the known **Bacula** resource record keywords, and it may be composed of upper or lower case characters and spaces.

Each resource definition MUST contain a Name directive, and may optionally contain a Description directive. The Name directive is used to uniquely identify the resource. The Description directive is (will be) used during display of the Resource to provide easier human recognition. For example:

```
Director {
    Name = "MyDir"
    Description = "Main Bacula Director"
    WorkingDirectory = "$HOME/bacula/bin/working"
}
```

Defines the Director resource with the name "MyDir" and a working directory **$HOME/bacula/bin/working**. In general, if you want spaces in a name to the right of the first equal sign (=), you must enclose that name within double quotes. Otherwise quotes are not generally necessary because once defined, quoted strings and unquoted strings are all equal.

2.4 Comments

When reading the configuration file, blank lines are ignored and everything after a hash sign (#) until the end of the line is taken to be a comment. A semicolon (;) is a logical end of line, and anything after the semicolon is considered as the next statement. If a statement appears on a line by itself, a semicolon is not necessary to terminate it, so generally in the examples in this manual, you will not see many semicolons.

2.5 Upper/Lower Case and Spaces

Case (upper/lower) and spaces are totally ignored in the resource directive keywords (the part before the equal sign).

Within the keyword (i.e. before the equal sign), spaces are not significant. Thus the keywords: **name**, **Name**, and **N a m e** are all identical.

Spaces after the equal sign and before the first character of the value are ignored.

In general, spaces within a value are significant (not ignored), and if the value is a name, you must enclose the name in double quotes for the spaces to be accepted. Names may contain up to 127 characters. Currently, a name may contain any `ASCII` character. Within a quoted string, any character following a backslash (\) is taken as itself (handy for inserting backslashes and double quotes (")).

Please note, however, that **Bacula** resource names as well as certain other names (e.g. Volume names) must contain only letters (including `ISO` accented letters), numbers, and a few special characters (space, underscore, dash, dot, colon). All other characters and punctuation are invalid.

2.6 Including Other Configuration Files

If you wish to break your configuration file into smaller pieces, you can do so by including other files using the syntax **@filename** where **filename** is the full path and filename of another file. The @filename specification can be given anywhere a primitive token would appear.

If you wish include all files in a specific directory, you can use the following:

```
# Include subfiles associated with configuration of clients.
# They define the bulk of the Clients, Jobs, and Filesets.
# Remember to "reload" the Director after adding a client file.
# @|"sh -c 'for f in /etc/bacula/clientdefs/*.conf; do echo @${f} ; done'"
```

2.7 Recognized Primitive Data Types

When parsing the resource directives, **Bacula** classifies the data according to the types listed below. The first time you read this, it may appear a bit overwhelming, but in reality, it is all pretty logical and straightforward.

name A keyword or name consisting of alphanumeric characters, including the hyphen, underscore, and dollar characters. The first character of a **name** must be a letter. A name has a maximum length currently set to 127 bytes. Typically keywords appear on the left side of an equal (i.e. they are **Bacula** keywords – i.e. Resource names or directive names). Keywords may not be quoted.

name-string A name-string is similar to a name, except that the name may be quoted and can thus contain additional characters including spaces. Name strings are limited to 127 characters in length. Name strings are typically used on the right side of an equal (i.e. they are values to be associated with a keyword).

string A quoted string containing virtually any character including spaces, or a non-quoted string. A string may be of any length. Strings are typically values that correspond to filenames, directories, or system command names. A backslash () turns the next character into itself, so to include a double quote in a string, you precede the double quote with a backslash. Likewise to include a backslash.

directory A directory is either a quoted or non-quoted string. A directory will be passed to your standard shell for expansion when it is scanned. Thus constructs such as **$HOME** are interpreted to be their correct values.

password This is a **Bacula** password and it is stored internally in MD5 hashed format.

integer A 32 bit integer value. It may be positive or negative.

positive integer A 32 bit positive integer value.

long integer A 64 bit integer value. Typically these are values such as bytes that can exceed 4 billion and thus require a 64 bit value.

<yes|no> Either a **yes** or a **no**.

size A size specified as bytes. Typically, this is a floating point scientific input format followed by an optional modifier. The floating point input is stored as a 64 bit integer value. If a modifier is present, it must immediately follow the value with no intervening spaces. The following modifiers are permitted:

k 1,024 (kilobytes)

kb 1,000 (kilobytes)

m 1,048,576 (megabytes)

mb 1,000,000 (megabytes)

g 1,073,741,824 (gigabytes)

gb 1,000,000,000 (gigabytes)

time A time or duration specified in seconds. The time is stored internally as a 64 bit integer value, but it is specified in two parts: a number part and a modifier part. The number can be an integer or a floating point number. If it is entered in floating point notation, it will be rounded to the nearest integer. The modifier is mandatory and follows the number part, either with or without intervening spaces. The following modifiers are permitted:

seconds seconds

minutes minutes (60 seconds)

hours hours (3600 seconds)

days days (3600*24 seconds)

weeks weeks (3600*24*7 seconds)

months months (3600*24*30 seconds)

quarters quarters (3600*24*91 seconds)

years years (3600*24*365 seconds)

Any abbreviation of these modifiers is also permitted (i.e. **seconds** may be specified as **sec** or **s**). A specification of **m** will be taken as months.

The specification of a time may have as many number/modifier parts as you wish. For example:

```
1 week 2 days 3 hours 10 mins
1 month 2 days 30 sec
```

are valid date specifications.

speed A speed specified as data transfer rate. Typically, this is a floating point scientific input format followed by an optional modifier. The floating point input is stored as a 64-bit integer value. If a modifier is present, it must immediately follow the value with no intervening spaces. The following modifiers are permitted:

k/s kilobytes per second (1 kilobyte = 1,024 bytes)

kb/s kilobytes per second (1 kilobyte = 1,000 bytes)

kib/s kibibytes per second (1 kibibyte = 1,024 bytes)

m/s megabytes per second (1 megabyte = 1,048,576 bytes)

mb/s megabytes per second (1 megabyte = 1,000,000 bytes)

mib/s mebibytes per second (1 mebibyte = 1,048,576 bytes)

2.8 Resource Types

The table below lists all **Bacula** resource types that are currently implemented. It shows what resources must be defined for each service (daemon). The default configuration files will already contain at least one example of each permitted resource, so you need not worry about creating all these resources from scratch.

Table 1: Resource types

Resource	Director	Client	Storage	Console
Autochanger	No	No	Yes	No
Catalog	Yes	No	No	No
Client	Yes	Yes	No	No
Cloud	No	No	Yes	No
Console	Yes	No	No	Yes
Device	No	No	Yes	No
Director	Yes	Yes	Yes	Yes
Fileset	Yes	No	No	No
Job	Yes	No	No	No
JobDefs	Yes	No	No	No
Message	Yes	Yes	Yes	No
Pool	Yes	No	No	No
Schedule	Yes	No	No	No
Statistics	Yes	Yes	Yes	No
Storage	Yes	No	Yes	No

2.9 Names, Passwords and Authorization

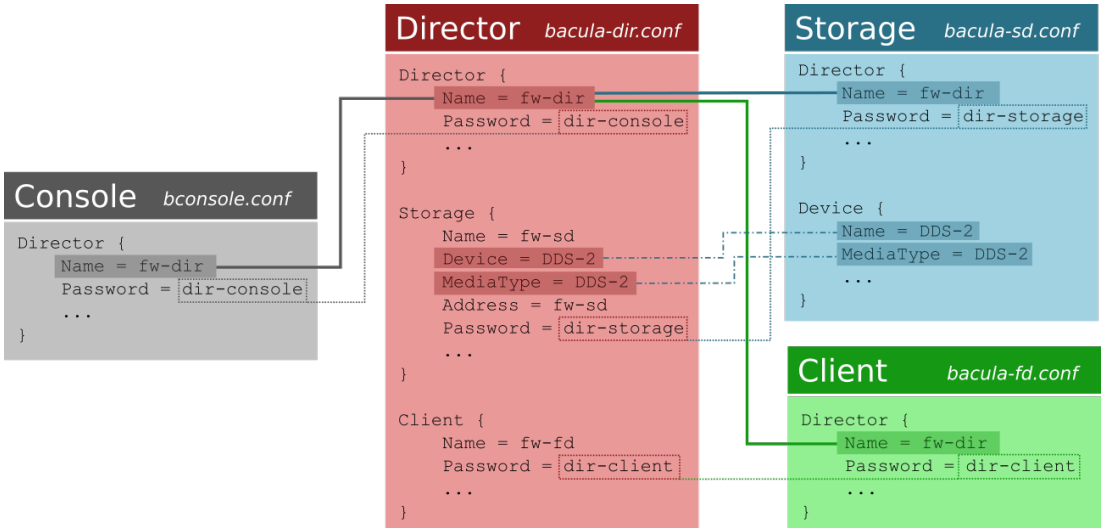
In order for one daemon to contact another daemon, it must authorize itself with a password. In most cases, the password corresponds to a particular name, so both the name and the password must match to be authorized. Passwords are plain text, any text. They are not generated by any special process; just use random text.

The default configuration files are automatically defined for correct authorization with random passwords. If you add to or modify these files, you will need to take care to keep them consistent.

Here is sort of a picture of what names/passwords in which files/Resources must match up:

In the left column, you will find the Director, Storage, and Client resources, with their names and passwords – these are all in **bacula-dir.conf**. In the right column are where the corresponding values should be found in the Console, Storage Daemon (SD), and File Daemon (FD) configuration files.

Note that the Address, **fd-sd**, that appears in the Storage resource of the Director, preceded with and asterisk in the above example, is passed to the File Daemon in symbolic form. The File Daemon then resolves it to an IP address. For this reason, you must use either an IP address or a fully qualified name. A name such as **localhost**, not being a fully qualified name, will resolve in the File daemon to the localhost of the File Daemon, which is most likely not what is desired. The password used for the File Daemon to authorize with the Storage Daemon is a temporary password unique to each Job created by the daemons and is not specified in any .conf file.



3 Catalog

4 Console

5 Director

6 File Daemon

7 Storage Daemon