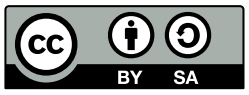
The Leading Open Source Backup Solution

Bacula® Console and Operators Guide

Kern Sibbald

December 3, 2024

This manual documents Bacula Community Edition 15.0.2 (21 March 2024)



2024-12-03

version 15.0.2

Bacula Community

Contents

Chapter 1

Bacula Enterprise Console

The Console (sometimes called the *User Agent*) is a program that allows the user or the System Administrator, to interact with the Director daemon while the daemon is running.

The current Console comes in two versions: a shell interface (TTY style), and a QT GUI interface (BAT). Both permit the administrator or authorized users to interact with Bacula. You can determine the status of a particular job, examine the contents of the Catalog as well as perform certain tape manipulations with the Console program.

Since the Console program interacts with the Bacula Director through the network, your Console and Director programs do not necessarily need to run on the same machine.

In fact, a certain minimal knowledge of the Console program is needed in order for Bacula to be able to write on more than one tape, because when Bacula requests a new tape, it waits until the user, via the Console program, indicates that the new tape is mounted.

1.1 Console Configuration

When the Console starts, it reads a standard Bacula configuration file named bconsole.conf or bat.conf in the case of the BAT QT Console version from the current directory unless you specify the **-c** command line option (see below). This file allows default configuration of the Console, and at the current time, the only Resource Record defined is the Director resource, which gives the Console the name and address of the Director. For more information on configuration of the Console program, please see the **Console Configuration** chapter (chapter 28 page 359) of the Bacula Community Edition Main manual.

1.2 Running the Console Program

The Bacula **Console** program can be run with the following options:

```
Usage: bconsole [-s] [-c config_file] [-d debug_level]
       -c <file>   set configuration file to file
       -dnn        set debug level to nn
       -n          no conio
       -s          no signals
       -u <nn>     set command execution timeout to <nn> seconds
       -t          test - read configuration and exit
       -?          print this message.
```

After launching the Console program (bconsole), it will prompt you for the next command

with an asterisk (*). Generally, for all commands, you can simply enter the command name and the Console program will prompt you for the necessary arguments. Alternatively, in most cases, you may enter the command followed by arguments. The general format is:

```
<command> <keyword1>[=<argument1>] <keyword2>[=<argument2>] ...
```

where command is one of the commands listed below; keyword is one of the keywords listed below (usually followed by an argument); and argument is the value. The command may be abbreviated to the shortest unique form. If two commands have the same starting letters, the one that will be selected is the one that appears first in the `help` listing. If you want the second command, simply spell out the full command. None of the keywords following the command may be abbreviated.

For example:

```
list files jobid=23
```

will list all files saved for JobId 23. Or:

```
show pools
```

will display all the `Pool` resource records.

The maximum command line length is limited to 511 characters, so if you are scripting the console, you may need to take some care to limit the line length.

1.3 Stopping the Console Program

Normally, you simply enter `quit` or `exit` and the `Console` program will terminate. However, it waits until the `Director` acknowledges the command. If the `Director` is already doing a lengthy command (e.g. `prune`), it may take some time. If you want to immediately terminate the `Console` program, enter the `.quit` command.

There is currently no way to interrupt a `Console` command once issued (i.e. `Ctrl-C` does not work). However, if you are at a prompt that is asking you to select one of several possibilities and you would like to abort the command, you can enter a period (`.`), and in most cases, you will either be returned to the main command prompt or if appropriate the previous prompt (in the case of nested prompts). In a few places such as where it is asking for a Volume name, the period will be taken to be the Volume name. In that case, you will most likely be able to cancel at the next prompt.

1.4 Alphabetic List of Console Keywords

Unless otherwise specified, each of the following keywords takes an argument, which is specified after the keyword following an equal sign. For example:

```
jobid=536
```

all Permitted on the `status` and `show` commands to specify all components or resources respectively.

frompool Permitted on the `update volume` command to specify that the volume specified on the command line should be updated with pool parameters.

allfrompool=<pool> Permitted on the `update` command to specify that all Volumes in the pool (specified on the command line) should be updated.

fromallpools Permitted on the `update` command to specify that all Volumes in all pools should be updated.

before Used in the `restore` command.

bootstrap Used in the `restore` command.

catalog Allowed in the `use` command to specify the catalog name to be used.

catalogs Used in the `show` command. Takes no arguments.

client | fd

clients Used in the `show`, `list`, and `llist` commands. Takes no arguments.

counters Used in the `show` command. Takes no arguments.

current Used in the `restore` command. Takes no argument.

days Used to define the number of days the `list nextvol` command should consider when looking for jobs to be run. The `days` keyword can also be used on the `status dir` command so that it will display jobs scheduled for the number of days you want.

devices Used in the `show` command. Takes no arguments.

dir | director

directors Used in the `show` command. Takes no arguments.

directory Used in the `restore` command. Its argument specifies the directory to be restored.

enabled This keyword can appear on the `update volume` as well as the `update slots` commands, and can allows one of the following arguments: yes, true, no, false, archived, 0, 1, 2. Where 0 corresponds to no or false, 1 corresponds to yes or true, and 2 corresponds to archived. Archived volumes will not be used, nor will the Media record in the catalog be pruned. Volumes that are not enabled, will not be used for backup or restore.

done Used in the `restore` command. Takes no argument.

file Used in the `restore` command.

files Used in the `list` and `llist` commands. Takes no arguments.

fileset

filesets Used in the `show` command. Takes no arguments.

help Used in the `show` command. Takes no arguments.

jobs Used in the `show`, `list` and `llist` commands. Takes no arguments.

jobmedia Used in the `list` and `llist` commands. Takes no arguments.

jobtotals Used in the `list` and `llist` commands. Takes no arguments.

jobid The JobId is the numeric jobid that is printed in the Job Report output. It is the index of the database record for the given job. While it is unique for all the existing Job records in the catalog database, the same JobId can be reused once a Job is removed from the catalog. Probably you will refer specific Jobs that ran using their numeric JobId.

job | jobname The `job` or `jobname` keyword refers to the name you specified in the Job resource, and hence it refers to any number of Jobs that ran. It is typically useful if you want to list all jobs of a particular name.

level

listing Permitted on the `estimate` command. Takes no argument.

limit

messages Used in the `show` command. Takes no arguments.

media Used in the `list` and `llist` commands. Takes no arguments.

name Used in the `list object` commands. Can specify an Object name.

nextvol | nextvolume Used in the `list` and `llist` commands. Takes no arguments.

object Used in the `list` commands. Takes no arguments.

objectid Used in the `list` and `restore` commands. Takes an ObjectId as argument.

order Used in the `list` to sort records. Can take "ASC" or "DESC" as argument.

on Takes no keyword.

off Takes no keyword.

pool

pools Used in the `show`, `list`, and `llist` commands. Takes no arguments.

select Used in the `restore` command. Takes no argument.

limit Used in the `setbandwidth` command. Takes integer in `KB/s` unit.

storages Used in the `show` command. Takes no arguments.

schedules Used in the `show` command. Takes no arguments.

sd | store | storage

tag Used in the `list` command. Takes no arguments.

ujobid The `ujobid` is a unique job identification that is printed in the Job Report output. At the current time, it consists of the Job name (from the **Name** directive for the job) appended with the date and time the job was run. This keyword is useful if you want to completely identify the Job instance run.

volume

volumes Used in the `list` and `llist` commands. Takes no arguments.

where Used in the `restore` command.

yes Used in the `restore` command. Takes no argument.

1.5 Alphabetic List of Console Commands

The following commands are currently implemented:

add [pool=<pool-name> storage=<storage> jobid=<JobId>] This command is used to add Volumes to an existing Pool. That is, it creates the Volume name in the catalog and inserts into the Pool in the catalog, but does not attempt to access the physical Volume. Once added, Bacula expects that Volume to exist and to be labeled. This command is not normally used since Bacula will automatically do the equivalent when Volumes are labeled. However, there may be times when you have removed a Volume from the catalog and want to later add it back.

Normally, the `label` command is used rather than this command because the `label` command labels the physical media (tape, disk, DVD, ...) and does the equivalent of the `add` command. The `add` command affects only the Catalog and not the physical media (data on Volumes). The physical media must exist and be labeled before use (usually with the `label` command). This command can, however, be useful if you wish to add a number of Volumes to the Pool that will be physically labeled at a later time. It can also be useful if you are importing a tape from another site. Please see the `label` command below for the list of legal characters in a Volume name.

autodisplay on/off This command accepts `on` or `off` as an argument, and turns auto-display of messages on or off respectively. The default for the console program is `off`, which means that you will be notified when there are console messages pending, but they will not automatically be displayed.

When autodisplay is turned off, you must explicitly retrieve the messages with the `messages` command. When `autodisplay` is turned on, the messages will be displayed on the console as they are received.

automount on/off This command accepts `on` or `off` as the argument, and turns auto-mounting of the Volume after a `label` command on or off respectively. The default is `on`. If `automount` is turned off, you must explicitly `mount` tape Volumes after a `label` command to use it.

cancel [jobid=<number> job=<job-name> ujobid=<unique-jobid>] | [inactive client=<client-name> storage=<storage-name>] This command is used to cancel a job and accepts `jobid=nnn` or `job=xxx` as an argument where `nnn` is replaced by the JobId and `xxx` is replaced by the job name. If you do not specify a keyword, the `Console` program will prompt you with the names of all the active jobs allowing you to choose one.

Once a Job is marked to be canceled, it may take a bit of time (generally within a minute but up to two hours) before the Job actually terminates, depending on what operations it is doing. Don't be surprised that you receive a Job not found message. That just means that one of the three daemons had already canceled the job. Messages numbered in the 1000's are from the Director, 2000's are from the File daemon and 3000's from the Storage daemon.

If the Job is still active on the Storage Daemon and/or the File Daemon, but not on the Director, it is possible to stop the Job with the **inactive** option of the `cancel` command.

cloud [storage=<sd-name> volume=<vol-name> allpools allfrompool pool=<p-name> mediatype=<type-name> drive=<num> slots=<num>] [truncate | prune | list | upload] The `cloud bconsole` command allows you to do a number of things with cloud volumes.

The `truncate` command will attempt to truncate the local cache for the specified Volume.

The `prune` command will attempt to prune the local cache for the specified Volume. Bacula will respect the **CacheRetention** directive volume attribute to determine if the cache can be truncated or not. Only parts that are uploaded to the cloud will be deleted from the cache.

The `upload` command will attempt to upload the specified Volumes.

The `list` command will list volumes stored in the Cloud. If a volume name is specified, the command will list all parts for the given volume.

create [pool=<pool-name>] This command is not normally used as the Pool records are automatically created by the `Director` when it starts based on what it finds in the conf file. If needed, this command can be to create a `Pool` record in the database using the `Pool` resource record defined in the `Director`'s configuration file. So in a sense, this command simply transfers the information from the `Pool` resource in the configuration file into the `Catalog`. Normally this command is done automatically for you when the `Director` starts providing the `Pool` is referenced within a `Job` resource. If you use this command on an existing `Pool`, it will automatically update the `Catalog` to have the same information as the `Pool` resource. After creating a `Pool`, you will most likely use the `label` command to label one or more volumes and add their names to the Media database.

When starting a Job, if Bacula determines that there is no Pool record in the database, but there is a Pool resource of the appropriate name, it will create it for you. If you want the Pool record to appear in the database immediately, simply use this command to force it to be created.

delete [volume=<vol-name> pool=<pool-name> job jobid=<id> object [objectid=id]] The delete command is used to delete a Volume, Pool Job, Object, Snapshot or Client record from the Catalog as well as all associated catalog Volume records that were created. This command operates only on the Catalog database and has no effect on the actual data written to a Volume. This command can be dangerous and we strongly recommend that you do not use it unless you know what you are doing.

If the keyword Volume appears on the command line, the named Volume will be deleted from the Catalog, if the keyword Pool appears on the command line, a Pool will be deleted, if the keyword Object appears on the command line, an Object and all of its associated records (File) will be deleted from the catalog, and if the keyword Job appears on the command line, a Job and all of its associated records (File and JobMedia) will be deleted from the catalog. The full form of this command is:

```
delete pool=<pool-name>
```

or

```
delete volume=<volume-name> pool=<pool-name>
```

or

```
delete JobId=<job-id> JobId=<job-id2> ...
```

or

```
delete JobId=n,m,o-r,t ...
```

or

```
delete client=xxx-fd
```

or

```
delete snapshot
```

or

```
delete object [objectid=id [all]] [category=obj\_cat] [type=obj\_type] [name=obj\_name] [uuid=obj\_uuid] [sour
```

The first form deletes a Pool record from the catalog database. The second form deletes a Volume record from the specified pool in the Catalog database. The third form deletes the specified Job record from the catalog database. The fourth form deletes JobId records for JobIds n, m, o, p, q, r and t. Where each one of the n,m,... is, of course, a number. That is a delete jobid accepts lists and ranges of jobids.

The deletion of a Client record will prune all job records associated with the Client. This command is possible only of the Client resource is no longer defined in the Director configuration file.

The deletion of a Snapshot record can be done with the sixth command.

The deletion of an Object records can be done with the last command. Different filters can be used along with 'delete object' command: category, type, name, uuid or source. Passing specific object id can be also used as a filter to delete many objects at once. Following command will delete all objects records with the same category, type, name, uuid and source as object with given id.

```
    delete object objectid=1 all
```

disable job<job-name> This command permits you to disable a Job for automatic scheduling. The job may have been previously enabled with the Job resource **Enabled** directive or using the console enable command. The next time the Director is restarted, the Enable/Disable state will be set to the value in the Job resource (default enabled) as defined in the bacula-dir.conf file.

disable job all This command permits you to disable all Jobs for automatic scheduling. The next time the `Director` is restarted, the Enable/Disable state will be set to the value in the `Job` resource (default enabled) as defined in the `bacula-dir.conf` file.

enable job<job-name> This command permits you to enable a `Job` for automatic scheduling. The job may have been previously disabled with the `Job` resource **Enabled** directive or using the console `disable` command. The next time the `Director` is restarted, the Enable/Disable state will be set to the value in the `Job` resource (default enabled) as defined in the `bacula-dir.conf` file.

enable job all This command permits you to enable all Jobs for automatic scheduling. It does not enable jobs which have **Disabled** directive directive. The next time the `Director` is restarted, the Enable/Disable state will be set to the value in the `Job` resource (default enabled) as defined in the `bacula-dir.conf` file.

estimate Using this command, you can get an idea how many files will be backed up, or if you are unsure about your Include statements in your `FileSet`, you can test them without doing an actual backup. The default is to assume a Full backup. However, you can override this by specifying a level=Incremental or level=Differential on the command line. A `Job` name must be specified or you will be prompted for one, and optionally a `Client` and `FileSet` may be specified on the command line. It then contacts the client which computes the number of files and bytes that would be backed up. Please note that this is an estimate calculated from the number of blocks in the file rather than by reading the actual bytes. As such, the estimated backup size will generally be larger than an actual backup.

The `estimate` command can use the accurate code to detect changes and give a better estimation. You can set the accurate behavior on command line using accurate=yes/no or use the `Job` setting as default value.

Optionally you may specify the keyword listing in which case, all the files to be backed up will be listed. Note, it could take quite some time to display them if the backup is large. The full form is:

```
estimate job=<job-name> listing client=<client-name> accurate=<yes/no>
         fileset=<fileset-name> level=<level-name>
```

Specification of the job is sufficient, but you can also override the client, fileset, accurate and/or level by specifying them on the `estimate` command line.

As an example, you might do:

```
@output /tmp/listing
estimate job=NightlySave listing level=Incremental
@output
```

which will do a full listing of all files to be backed up for the Job **NightlySave** during an Incremental save and put it in the file `/tmp/listing`. Note, the byte estimate provided by this command is based on the file size contained in the directory item. This can give wildly incorrect estimates of the actual storage used if there are sparse files on your systems. Sparse files are often found on 64 bit systems for certain system files. The size that is returned is the size Bacula will backup if the sparse option is not specified in the `FileSet`. There is currently no way to get an estimate of the real file size that would be found should the sparse option be enabled.

exit This command terminates the `Console` program.

gui Invoke the non-interactive gui mode.

```
gui [on|off]
```

help This command displays the list of commands available.

label This command is used to label physical volumes. The full form of this command is:

```
label storage=<storage-name> volume=<volume-name>
      slot=<slot>
```

If you leave out any part, you will be prompted for it. The media type is automatically taken from the Storage resource definition that you supply. Once the necessary information is obtained, the Console program contacts the specified Storage daemon and requests that the Volume be labeled. If the Volume labeling is successful, the Console program will create a Volume record in the appropriate Pool.

The Volume name is restricted to letters, numbers, and the special characters hyphen (-), underscore (_), colon (:), and period (.). All other characters including a space are invalid. This restriction is to ensure good readability of Volume names to reduce operator errors.

Please note, when labeling a blank tape, Bacula will get **read I/O error** when it attempts to ensure that the tape is not already labeled. If you wish to avoid getting these messages, please write an EOF mark on your tape before attempting to label it:

```
mt rewind
mt weof
```

The label command can fail for a number of reasons:

1. The Volume name you specify is already in the Volume database.
2. The Storage daemon has a tape or other Volume already mounted on the device, in which case you must unmount the device, insert a blank tape, then do the label command.
3. The Volume in the device is already a Bacula labeled Volume. (Bacula will never relabel a Bacula labeled Volume unless it is recycled and you use the relabel command).
4. There is no Volume in the drive.

There are two ways to relabel a volume that already has a Bacula label. The brute force method is to write an end of file mark on the tape using the system mt program, something like the following:

```
mt -f /dev/st0 rewind
mt -f /dev/st0 weof
```

For a disk volume, you would manually delete the Volume.

Then you use the label command to add a new label. However, this could leave traces of the old volume in the catalog.

The preferable method to relabel a Volume is to first purge the volume, either automatically, or explicitly with the purge command, then use the relabel command described below.

If your autochanger has barcode labels, you can label all the Volumes in your autochanger one after another by using the label barcodes command. For each tape in the changer containing a barcode, Bacula will mount the tape and then label it with the same name as the barcode. An appropriate Media record will also be created in the catalog. Any barcode that begins with the same characters as specified on the "**CleaningPrefix**=xxx" directive in the Director's Pool resource, will be treated as a cleaning tape, and will not be labeled. However, an entry for the cleaning tape will be created in the catalog. For example with:

```
Pool {
     Name ...
     Cleaning Prefix = "CLN"
}
```

Any slot containing a barcode of CLNxxxx will be treated as a cleaning tape and will not be mounted. Note, the full form of the command is:

```
label storage=xxx pool=yyy slots=1-5,10 barcodes
```

list The list command lists the requested contents of the Catalog. The various fields of each record are listed on a single line. The various forms of the list command are:

```
list events

list jobs

list jobid=<id>                (list jobid id)

list jobs joberrors            (list jobs with errors)

list jobs jobstatus=f          (list jobs with jobstatus f)

list jobs limit=10 order=desc  (list the last 10 jobs)

list jobs limit=10 order=asc   (list the first 10 jobs)

list jobs client=xxx           (list jobs for client xxx)

list copies                    (list copy jobs)

list ujobid=<unique job name>  (list job with unique name)

list job=<job-name>            (list all jobs with "job-name")

list jobname=<job-name>        (same as above)

     In the above, you can add "limit=nn" to limit the output to
     nn jobs.

list joblog jobid=<id> (list job output if recorded in the catalog)

list jobmedia

list jobmedia jobid=<id>

list jobmedia job=<job-name>

list files jobid=<id>

list files job=<job-name>

     In the above, you can add type=<all|deleted> to see all file records or only
     deleted records.

list files type=<deleted|all> jobid=<id>

list pools

list clients

list jobtotals

list objects (list plugin objects)

list objects jobid=<id>

list object client=<client> (list plugin objects of specified client)

list object type=<type> (list plugin objects of specified type)

list objects category=<category> (list plugin objects of specified category)

list objects limit=10 order=desc  (list the last 10 objects)

list objects limit=10 order=asc   (list the first 10 objects)

list volumes

list volumes jobid=<id>

list volumes pool=<pool-name>

list volumes job=<job-name>

list volume=<volume-name>

list nextvolume job=<job-name>
```

```
list nextvol job=<job-name>

list nextvol job=<job-name> days=nnn
```

What most of the above commands do should be more or less obvious. In general if you do not specify all the command line arguments, the command will prompt you for what is needed.

The `list nextvol` command will print the Volume name to be used by the specified job. You should be aware that exactly what Volume will be used depends on a lot of factors including the time and what a prior job will do. It may fill a tape that is not full when you issue this command. As a consequence, this command will give you a good estimate of what Volume will be used but not a definitive answer. In addition, this command may have certain side effect because it runs through the same algorithm as a job, which means it may automatically purge or recycle a Volume. By default, the job specified must run within the next two days or no volume will be found. You can, however, use the days=nnn specification to specify up to 50 days. For example, if on Friday, you want to see what Volume will be needed on Monday, for job MyJob, you would use `list nextvol job=MyJob days=3`.

If you wish to add specialized commands that list the contents of the catalog, you can do so by adding them to the `query.sql` file. However, this takes some knowledge of programming SQL. Please see the `query` command below for additional information. See below for listing the full contents of a catalog record with the `llist` command.

As an example, the command `list pools` might produce the following output:

```
+------+---------+---------+---------+----------+-------------+
| PoId | Name    | NumVols | MaxVols | PoolType | LabelFormat |
+------+---------+---------+---------+----------+-------------+
|    1 | Default |       0 |       0 | Backup   | *           |
|    2 | Recycle |       0 |       8 | Backup   | File        |
+------+---------+---------+---------+----------+-------------+
```

As mentioned above, the `list` command lists what is in the database. Some things are put into the database immediately when Bacula starts up, but in general, most things are put in only when they are first used, which is the case for a Client as with Job records, etc.

Bacula should create a client record in the database the first time you run a job for that client. Doing a `status` will not cause a database record to be created. The client database record will be created whether or not the job fails, but it must at least start. When the Client is actually contacted, additional info from the client will be added to the client record (a `uname -a` output).

If you want to see what `Client` resources you have available in your conf file, you use the Console command `show clients`.

llist The `llist` or "long list" command takes all the same arguments that the `list` command described above does. The difference is that the `llist` command `list` the full contents of each database record selected. It does so by listing the various fields of the record vertically, with one field per line. It is possible to produce a very large number of output lines with this command.

If instead of the `list pools` as in the example above, you enter `llist pools` you might get the following output:

```
          PoolId: 1
            Name: Default
         NumVols: 0
         MaxVols: 0
         UseOnce: 0
      UseCatalog: 1
 AcceptAnyVolume: 1
    VolRetention: 1,296,000
  VolUseDuration: 86,400
     MaxVolJobs: 0
    MaxVolBytes: 0
       AutoPrune: 0
```

```
         Recycle: 1
        PoolType: Backup
     LabelFormat: *

          PoolId: 2
            Name: Recycle
         NumVols: 0
         MaxVols: 8
         UseOnce: 0
      UseCatalog: 1
 AcceptAnyVolume: 1
    VolRetention: 3,600
 VolUseDuration: 3,600
     MaxVolJobs: 1
    MaxVolBytes: 0
      AutoPrune: 0
        Recycle: 1
       PoolType: Backup
    LabelFormat: File
```

messages This command causes any pending console messages to be immediately displayed.

memory Print current memory usage.

mount The `mount` command is used to get Bacula to read a volume on a physical device. It is a way to tell Bacula that you have mounted a tape and that Bacula should examine the tape. This command is normally used only after there was no Volume in a drive and Bacula requests you to mount a new Volume or when you have specifically unmounted a Volume with the `unmount` console command, which causes Bacula to close the drive. If you have an autoloader, the `mount` command will not cause Bacula to operate the autoloader unless you specify a slot and possibly a drive. The various forms of the mount command are:

```
mount storage=<storage-name> [ slot=<num> ] [ drive=<num> ]
```

```
mount [ jobid=<id> | job=<job-name> ]
```

If you have specified **Automatic Mount = yes** in the Storage daemon's `Device` resource, under most circumstances, Bacula will automatically access the Volume unless you have explicitly `unmount`ed it in the Console program.

prune The `prune` command allows you to safely remove expired database records from Jobs, Volumes and Statistics. This command works only on the Catalog database and does not affect data written to Volumes. In all cases, the `prune` command applies a retention period to the specified records. You can prune expired File entries from Job records; you can prune expired Job records from the database, and you can prune both expired Job and File records from specified Volumes.

```
prune files|jobs|volume|stats client=<client-name>
      volume=<volume-name>
```

For a Volume to be pruned, the **VolStatus** must be *Full*, *Used*, or *Append*, otherwise the pruning will not take place.

With the all keyword, all combination of `Client`/`Pool` present in the `Job` table will be pruned. `prune jobs all yes`

purge The `purge` command will delete associated Catalog database records from Jobs and Volumes without considering the retention period. `purge` works only on the Catalog database and does not affect data written to Volumes. This command can be dangerous because you can delete catalog records associated with current backups of files, and we recommend that you do not use it unless you know what you are doing. The permitted forms of `purge` are:

```
purge files jobid=<jobid>|job=<job-name>|client=<client-name>
```

```
purge jobs client=<client-name> (of all jobs)
```

```
purge volume|volume=<vol-name> (of all jobs)
```

For the purge command to work on Volume Catalog database records the **VolStatus** must be Append, Full, Used, or Error.

The actual data written to the Volume will be unaffected by this command unless you are using the **ActionOnPurge=Truncate** option on those Media.

To ask Bacula to truncate your Purged volumes, you need to use the following command in interactive mode or in a RunScript:

```
* purge volume action=truncate storage=File allpools

# or by default, action=all
* purge volume action storage=File pool=Default
```

This is possible to specify the volume name, the media type, the pool, the storage, etc... (see help purge) Be sure that your storage device is idle when you decide to run this command.

query This command reads a predefined SQL query from the query file (the name and location of the query file is defined with the QueryFile resource record in the Director's configuration file). You are prompted to select a query from the file, and possibly enter one or more parameters, then the command is submitted to the Catalog database SQL engine.

The following queries are currently available (version 2.2.7):

```
Available queries:
1: List up to 20 places where a File is saved regardless of the directory
2: List where the most recent copies of a file are saved
3: List last 20 Full Backups for a Client
4: List all backups for a Client after a specified time
5: List all backups for a Client
6: List Volume Attributes for a selected Volume
7: List Volumes used by selected JobId
8: List Volumes to Restore All Files
9: List Pool Attributes for a selected Pool
10: List total files/bytes by Job
11: List total files/bytes by Volume
12: List Files for a selected JobId
13: List Jobs stored on a selected MediaId
14: List Jobs stored for a given Volume name
15: List Volumes Bacula thinks are in changer
16: List Volumes likely to need replacement from age or errors
Choose a query (1-16):
```

quit This command terminates the console program. The console program sends the **quit** request to the Director and waits for acknowledgment. If the Director is busy doing a previous command for you that has not terminated, it may take some time. You may quit immediately by issuing the .quit command (i.e. quit preceded by a period).

relabel This command is used to label physical volumes. The full form of this command is:

```
relabel storage=<storage-name> oldvolume=<old-volume-name>
        volume=<newvolume-name>
```

If you leave out any part, you will be prompted for it. In order for the Volume (old-volume-name) to be relabeled, it must be in the catalog, and the volume status must be marked Purged or Recycle. This happens automatically as a result of applying retention periods, or you may explicitly purge the volume using the purge command.

Once the volume is physically relabeled, the old data previously written on the Volume is lost and cannot be recovered.

release This command is used to cause the Storage daemon to rewind (release) the current tape in the drive, and to re-read the Volume label the next time the tape is used.

```
release storage=<storage-name>
```

After a `release` command, the device is still kept open by Bacula (unless **Always Open** is set to No in the Storage Daemon's configuration) so it cannot be used by another program. However, with some tape drives, the operator can remove the current tape and to insert a different one, and when the next Job starts, Bacula will know to re-read the tape label to find out what tape is mounted. If you want to be able to use the drive with another program (e.g. `mt`), you must use the `unmount` command to cause Bacula to completely release (close) the device.

reload The `reload` command causes the Director to re-read its configuration file and apply the new values. The new values will take effect immediately for all new jobs. However, if you change schedules, be aware that the scheduler pre-schedules jobs up to two hours in advance, so any changes that are to take place during the next two hours may be delayed. Jobs that have already been scheduled to run (i.e. surpassed their requested start time) will continue with the old values. New jobs will use the new values. Each time you issue a `reload` command while jobs are running, the old config values will keptf until all jobs that were running before issuing the reload terminate, at which time the old config values will be released from memory. As a default a maximum number of 32 reload requests that can be made, which is generally sufficient. In the case that you make a very large number of reload requests, you may use the **Maximum Reload Requests** directive in the Director resource of **bacula-dir.conf** to set a larger maximum to that value you wish.

restart The `restart` command allows console users to restart a canceled, failed, or incomplete Job. For canceled and failed Jobs, the Job will restart from the beginning. For incomplete Jobs the Job will restart at the point that it was stopped either by a stop command or by some recoverable failure.

resume The `resume` command does exactly the same thing as a `restart` command, but for some users the name may be more logical because in general the `restart` command is used to resume running a Job that was incomplete.

restore The `restore` command allows you to select one or more Jobs (JobIds) to be restored using various methods. Once the JobIds are selected, the File records for those Jobs are placed in an internal Bacula directory tree, and the restore enters a file selection mode that allows you to interactively walk up and down the file tree selecting individual files to be restored. This mode is somewhat similar to the standard Unix `restore` program's interactive file selection mode.

```
restore storage=<storage-name> client=<backup-client-name> where=<path>
        pool=<pool-name> fileset=<fileset-name>
        restoreclient=<restore-client-name> objectid=<id>
        restorejob=<job-name> jobuser=<user-name> jobgroup=<group-name>
        select current all done
```

Where current, if specified, tells the `restore` command to automatically select a restore to the most current backup. If not specified, you will be prompted. The all specification tells the `restore` command to restore all files. If it is not specified, you will be prompted for the files to restore. For details of the `restore` command, please see the **Restore** chapter (chapter 30 page 373) of the Bacula Community Edition Main manual.

The client keyword initially specifies the client from which the backup was made and the client to which the restore will be make. However, if the restoreclient keyword is specified, then the restore is written to that client.

The restore job rarely needs to be specified, as bacula installations commonly only have a single restore job configured. However, for certain cases, such as a varying list of `RunScript` specifications, multiple restore jobs may be configured. The restorejob argument allows the selection of one of these jobs.

run This command allows you to schedule jobs to be run immediately. The full form of the command is:

```
run job=<job-name> client=<client-name>
    fileset=<FileSet-name>  level=<level-keyword>
    storage=<storage-name>  where=<directory-prefix>
    when=<universal-time-specification> spooldata=yes|no yes
```

Any information that is needed but not specified will be listed for selection, and before starting the job, you will be prompted to accept, reject, or modify the parameters of the job to be run, unless you have specified yes, in which case the job will be immediately sent to the scheduler.

On my system, when I enter a run command, I get the following prompt:

```
A job name must be specified.
The defined Job resources are:
     1: Matou
     2: Polymatou
     3: Rufus
     4: Minimatou
     5: Minou
     6: PmatouVerify
     7: MatouVerify
     8: RufusVerify
     9: Watchdog
Select Job resource (1-9):
```

If I then select number 5, I am prompted with:

```
Run Backup job
JobName:  Minou
FileSet:  Minou Full Set
Level:    Incremental
Client:   Minou
Storage:  DLTDrive
Pool:     Default
When:     2003-04-23 17:08:18
OK to run? (yes/mod/no):
```

If I now enter yes, the Job will be run. If I enter mod, I will be presented with the following prompt:

```
Parameters to modify:
     1: Level
     2: Storage
     3: Job
     4: FileSet
     5: Client
     6: When
     7: Pool
Select parameter to modify (1-7):
```

If you wish to start a job at a later time, you can do so by setting the When time. Use the mod option and select When (no. 6). Then enter the desired start time in `YYYY-MM-DD HH:MM:SS` format.

The spooldata argument of the run command cannot be modified through the menu and is only accessible by setting its value on the intial command line. If no spooldata flag is set, the job, storage or schedule flag is used.

setbandwidth This command is used to limit the bandwidth of a running job or a client.

```
setbandwidth limit=<nb> [ jobid=<id> | client=<cli> ]
```

setdebug This command is used to set the debug level in each daemon. The form of this command is:

```
setdebug level=nn [trace=0/1 client=<client-name> | dir | director |
          storage=<storage-name> | all | options=0cidtTlp | tags=<tags>]
```

If trace=1 is set, then tracing will be enabled, and the daemon will be placed in trace mode, which means that all debug output as set by the debug level will be directed to the file `bacula.trace` in the current directory of the daemon. Normally, tracing is needed only for Win32 clients where the debug output cannot be written to a terminal or redirected to a file. When tracing, each debug output message is appended to the trace file. You must explicitly delete the file when you are done.

If options parameter is set, the following arguments can be used to control debug functions.

0 clear debug flags

i Turn off, ignore bwrite() errors on restore on File Daemon

d Turn off decomp of BackupRead() streams on File Daemon

t Turn on timestamp in traces

T Turn off timestamp in traces

c Truncate trace file if trace file is activated

l Turn on recoding events on P() and V()

p Turn on the display of the event ring when doing a lockdump

It is now possible to use *class* of debug messages called `tags` to control the debug output of Bacula daemons.

all Display all debug messages

bvfs Display BVFS debug messages

sql Display SQL related debug messages

memory Display memory and poolmem allocation messages

scheduler Display scheduler related debug messages

```
* setdebug level=10 tags=bvfs,sql,memory
* setdebug level=10 tags=!bvfs
```

The `tags` option is composed of a list of tags, tags are separated by "," or "+" or "-" or "!". To disable a specific tag, use "-" or "!" in front of the tag.

setip Sets new client address — if authorized.

A console is authorized to use the `SetIP` command only if it has a `Console` resource definition in both the Director and the Console. In addition, if the console name, provided on the **Name** directive, must be the same as a Client name, the user of that console is permitted to use the `SetIP` command to change the **Address** directive in the Director's `Client` resource to the IP address of the Console. This permits portables or other machines using DHCP (non-fixed IP addresses) to "notify" the Director of their current IP address.

show The `show` command will list the `Director`'s resource records as defined in the Director's configuration file (normally `bacula-dir.conf`). This command is used mainly for debugging purposes by developers. The following keywords are accepted on the `show` command line:catalogs, clients, counters, devices, directors filesets, jobs, messages, pools, schedules, storages, all, help. Please don't confuse this command with the `list`, which displays the contents of the catalog.

sqlquery The `sqlquery` command puts the Console program into SQL query mode where each line you enter is concatenated to the previous line until a semicolon (;) is seen. The semicolon terminates the command, which is then passed directly to the SQL database engine. When the output from the SQL engine is displayed, the formation of a new SQL command begins. To terminate SQL query mode and return to the Console command prompt, you enter a period (.) in column 1.

Using this command, you can query the SQL catalog database directly. Note you should really know what you are doing otherwise you could damage the catalog database. See the `query` command above for simpler and safer way of entering SQL queries.

Depending on what database engine you are using (`MySQL` or `PostgreSQL`), you will have somewhat different SQL commands available. For more detailed information, please refer to the `MySQL`, `PostgreSQL` documentation.

status This command will display the status of all components. For the director, it will display the next jobs that are scheduled during the next 24 hours as well as the status of currently running jobs. For the Storage Daemon, you will have drive status or autochanger content. The File Daemon will give you information about current jobs like average speed or file accounting. The full form of this command is:

```
status [all | dir=<dir-name> | director [days=nnn] |
       client=<client-name> | [slots] storage=<storage-name>] | network [bytes=<nb>]
```

If you do a `status dir`, the console will list any currently running jobs, a summary of all jobs scheduled to be run in the next 24 hours, and a listing of the last ten terminated jobs with their statuses. The scheduled jobs summary will include the Volume name to be used. You should be aware of two things: 1. to obtain the volume name, the code goes through the same code that will be used when the job runs, but it does not do pruning nor recycling of Volumes; 2. The Volume listed is at best a guess. The Volume actually used may be different because of the time difference (more durations may expire when the job runs) and another job could completely fill the Volume requiring a new one.

In the Running Jobs listing, you may find the following types of information:

```
2507 Catalog MatouVerify.2004-03-13_05.05.02 is waiting execution
5349 Full    CatalogBackup.2004-03-13_01.10.00 is waiting for higher
             priority jobs to finish
5348 Differe Minou.2004-03-13_01.05.09 is waiting on max Storage jobs
5343 Full    Rufus.2004-03-13_01.05.04 is running
```

Looking at the above listing from bottom to top, obviously JobId 5343 (Rufus) is running. JobId 5348 (Minou) is waiting for JobId 5343 to finish because it is using the Storage resource, hence the "waiting on max Storage jobs". JobId 5349 has a lower priority than all the other jobs so it is waiting for higher priority jobs to finish, and finally, JobId 2507 (MatouVerify) is waiting because only one job can run at a time, hence it is simply "waiting execution"

If you do a `status dir`, it will by default list the first occurrence of all jobs that are scheduled today and tomorrow. If you wish to see the jobs that are scheduled in the next three days (e.g. on Friday you want to see the first occurrence of what tapes are scheduled to be used on Friday, the weekend, and Monday), you can add the days=3 option. Note, a days=0 shows the first occurrence of jobs scheduled today only. If you have multiple run statements, the first occurrence of each run statement for the job will be displayed for the period specified.

If your job seems to be blocked, you can get a general idea of the problem by doing a `status dir`, but you can most often get a much more specific indication of the problem by doing a `status storage=xxx`. For example, on an idle test system, when we do `status storage=File`, we get:

```
status storage=File
Connecting to Storage daemon File at 192.168.68.112:8103

rufus-sd Version: 1.39.6 (24 March 2006) i686-pc-linux-gnu redhat (Stentz)
Daemon started 26-Mar-06 11:06, 0 Jobs run since started.

Running Jobs:
No Jobs running.
====

Jobs waiting to reserve a drive:
====

Terminated Jobs:
 JobId  Level    Files          Bytes Status   Finished        Name
===================================================================
    59  Full         234      4,417,599 OK       15-Jan-06 11:54 kernsave
====

Device status:
Autochanger "DDS-4-changer" with devices:
   "DDS-4" (/dev/nst0)
Device "DDS-4" (/dev/nst0) is mounted with Volume="TestVolume002"
Pool="*unknown*"
    Slot 2 is loaded in drive 0.
    Total Bytes Read=0 Blocks Read=0 Bytes/block=0
    Positioned at File=0 Block=0

Device "DVD-Writer" (/dev/hdc) is not open.
Device "File" (/tmp) is not open.
```

```
====

In Use Volume status:
====
```

Now, what this tells us is that no jobs are running and that none of the devices are in use. Now, if we unmount the autochanger, which will not be used in this example, and then start a Job that uses the File device, the job will block. When we re-issue the status storage command, we get for the Device status:

```
status storage=File
...
Device status:
Autochanger "DDS-4-changer" with devices:
   "DDS-4" (/dev/nst0)
Device "DDS-4" (/dev/nst0) is not open.
    Device is BLOCKED. User unmounted.
    Drive 0 is not loaded.

Device "DVD-Writer" (/dev/hdc) is not open.
Device "File" (/tmp) is not open.
    Device is BLOCKED waiting for media.
====
...
```

Now, here it should be clear that if a job were running that wanted to use the Autochanger (with two devices), it would block because the user unmounted the device. The real problem for the Job I started using the "File" device is that the device is blocked waiting for media — that is Bacula needs you to label a Volume.

If you enter status storage, Bacula will prompt you with a list of the storage resources. When you select one, the Storage daemon will be requested to do a status. However, note that the Storage daemon will do a status of all the devices it has, and not just of the one you requested. In the current version of Bacula, when you enter the status storage command, it prompts you only with a subset of all the storage resources that the Director considers to be in different Storage daemons. In other words, it attempts to remove duplicate storage definitions. This can be a bit confusing at first, but can vastly simplify the promt listing if you have defined a large number of storage resources.

If you prefer to see the full list of all storage resources, simply add the keyword select to the command such as: status select storage and you will get a prompt that includes all storage resources even if they reference the same storage daemon.

If you enter status network, Bacula will prompt you with a list of the storage resources and a list of the client resources. Then, Bacula will test the network throughput between the two selected daemons.

```
*status network
The defined Client resources are:
     1: zog82-fd
     2: zog8-fd
Select Client (File daemon) resource (1-2): 2
Automatically selected Storage: File1
Connecting to Storage File1 at zog8:8103
Connecting to Client zog8-fd at zog8:8102
Running network test between Client=zog8-fd and Storage=File1 with 52.42 MB ...
2000 OK bytes=52428800 duration=48ms write_speed=1.088 GB/s
2000 OK bytes=52428800 duration=56ms read_speed=933.8 MB/s
```

tag The tag command will add, delete or list the tags associated with catalog records such as Clients, Jobs, Volumes or Objects. The command accepts all parameters in command line.

```
*tag add name="#test1" client=zog8-fd
1000 Tag added

*tag
Available Tag operations:
     1: Add
     2: Delete
     3: List
```

```
Select Tag operation (1-3): 1
Available Tag target:
     1: Client
     2: Job
     3: Volume
Select Tag target (1-3): 1
Automatically selected Client: zog8-fd
Enter the Tag value: test1
1000 Tag added
*tag
Available Tag operations:
     1: Add
     2: Delete
     3: List
Select Tag operation (1-3): 3
Available Tag target:
     1: Client
     2: Job
     3: Volume
Select Tag target (1-3): 1
Automatically selected Client: zog8-fd
+--------+----------+---------+
| tag    | clientid | client  |
+--------+----------+---------+
| test1  |        1 | zog8-fd |
| #test1 |        1 | zog8-fd |
+--------+----------+---------+
```

time Prints the current time.

trace Turn on/off trace to file.

umount For old-time Unix guys. See the unmount command for full details.

unmount This command causes the indicated Bacula Storage daemon to unmount the specified device. The forms of the command are the same as the mount command:

```
unmount storage=<storage-name> [ drive=<num> ]
```

```
unmount [ jobid=<id> | job=<job-name> ]
```

Once you unmount a storage device, Bacula will no longer be able to use it until you issue a mount command for that device. If Bacula needs to access that device, it will block and issue mount requests periodically to the operator.

If the device you are unmounting is an autochanger, it will unload the drive you have specified on the command line. If no drive is specified, it will assume drive 1.

update This command will update the catalog for either a specific Pool record, a Volume record, or the Slots in an autochanger with barcode capability. In the case of updating a Pool record, the new information will be automatically taken from the corresponding Director's configuration resource record. It can be used to increase the maximum number of volumes permitted or to set a maximum number of volumes. The following main keywords may be specified: media, volume, pool, slots, stats.

In the case of updating a Volume, you will be prompted for which value you wish to change. The following Volume parameters may be changed:

```
Volume Status
Volume Retention Period
Volume Use Duration
Maximum Volume Jobs
Maximum Volume Files
Maximum Volume Bytes
Recycle Flag
Recycle Pool
Slot
InChanger Flag
Pool
Volume Files
Volume from Pool
All Volumes from Pool
All Volumes from all Pools
```

For slots update slots, Bacula will obtain a list of slots and their barcodes from the Storage daemon, and for each barcode found, it will automatically update the slot in the catalog Media record to correspond to the new value. This is very useful if you have moved cassettes in the magazine, or if you have removed the magazine and inserted a different one. As the slot of each Volume is updated, the InChanger flag for that Volume will also be set, and any other Volumes in the Pool that were last mounted on the same Storage device will have their InChanger flag turned off. This permits Bacula to know what magazine (tape holder) is currently in the autochanger.

If you do not have barcodes, you can accomplish the same thing in version 1.33 and later by using the update slots scan command. The scan keyword tells Bacula to physically mount each tape and to read its VolumeName.

For Pool update pool, Bacula will move the Volume record from its existing pool to the pool specified.

For Volume from Pool, All Volumes from Pool and All Volumes from all Pools, the following values are updated from the Pool record: Recycle, RecyclePool, VolRetention, VolUseDuration, MaxVolJobs, MaxVolFiles, and MaxVolBytes. (RecyclePool feature is available with Bacula 2.1.4 or higher.)

The full form of the update command with all command line arguments is:

```
update volume=xxx pool=yyy slots volstatus=xxx VolRetention=ddd
       VolUse=ddd MaxVolJobs=nnn MaxVolBytes=nnn Recycle=yes|no
       slot=nnn enabled=n recyclepool=zzz actiononpurge=xxx
```

```
update volume=xxx frompool
```

```
update volume allfrompool=xxx
```

```
update volume fromallpools
```

use This command allows you to specify which Catalog database to use. Normally, you will be using only one database so this will be done automatically. In the case that you are using more than one database, you can use this command to switch from one to another.

```
use [catalog=name-of-catalog]
```

var This command takes a string or quoted string and does variable expansion on it the same way variable expansion is done on the **LabelFormat** string. Thus, for the most part, you can test your LabelFormat strings. The difference between the var command and the actual LabelFormat process is that during the var command, no job is running so "dummy" values are used in place of Job specific variables. Generally, however, you will get a good idea of what is going to happen in the real case.

version The command prints the Director's version.

wait The wait command causes the Director to pause until there are no jobs running. This command is useful in a batch situation such as regression testing where you wish to start a job and wait until that job completes before continuing. This command now has the following options:

```
wait [jobid=nn] [jobuid=unique id] [job=job name]
```

If specified with a specific JobId, the wait command will wait for that particular job to terminate before continuing.

1.6 Special dot Commands

There is a list of commands that are prefixed with a period (.). These commands are intended to be used either by batch programs or graphical user interface front-ends. They are not normally used by interactive users. Once GUI development begins, this list will be considerably expanded. The following is the list of dot commands:

```
.api
.backups job=xxx      list backups for specified job
.clients              list all client names
.catalogs
.defaults client=xxx fileset=yyy  list defaults for specified client
.die                  cause the Director to segment fault (for debugging)
.dir                  when in tree mode prints the equivalent to the dir command,
                      but with fields separated by commas rather than spaces.
.dump
.exit                 quit
.events               list record custom events
.filesets             list all fileset names
.help                 help command output
.jobs                 list all job names
.estimate
.jlist                list catalog objects in JSON format (see list command)
.levels               list all levels
.messages             get quick messages
.msgs                 return any queued messages
.pools                list all pool names
.quit                 quit
.putfile
.schedule
.sql
.status               get status output
.storage              return storage resource names
.volstatus
.media
.mediatypes
.locations
.actiononpurge
.bvfs_lsdirs
.bvfs_lsfiles   [allfiles] [jobid=xxx] [pathid=xxx] [path=str]
.bvfs_get_volumes
.bvfs_update
.bvfs_get_jobids
.bvfs_get_jobs
.bvfs_get_bootstrap
.bvfs_get_fileindex
.bvfs_versions
.bvfs_get_delta
.bvfs_restore
.bvfs_cleanup
.bvfs_decode_lstat
.bvfs_clear_cache
.bvfs_update_fv
.bvfs_delete_fileid
.setuid
.ls
.types                list job types
.query
.tags                 list tags
```

1.7 Special At (@) Commands

Normally, all commands entered to the Console program are immediately forwarded to the Director, which may be on another machine, to be executed. However, there is a small list of `at` commands, all beginning with an at character (`@`), that will not be sent to the Director, but rather interpreted by the Console program directly. Note, these commands are implemented only in the tty console program and not in the BAT Console. These commands are:

@input <filename> Read and execute the commands contained in the file specified.

@output <filename> w/a Send all following output to the filename specified either overwriting the file (w) or appending to the file (a). To redirect the output to the terminal, simply enter `@output` without a filename specification. WARNING: be careful not to overwrite a valid file. A typical example during a regression test might be:

```
@output /dev/null
commands ...
@output
```

@tee <filename> w/a Send all subsequent output to both the specified file and the terminal. It is turned off by specifying `@tall`, `@tee` or `@output` without a filename.

@tall <filename> w/a Send all subsequent input and output to both the specified file and the terminal. It is turned off by specifying `@tall`, `@tee` or `@output` without a filename.

@sleep <seconds> Sleep the specified number of seconds.

@time Print the current time and date.

@version Print the console's version.

@quit quit

@exit quit

@# anything Comment

@help Get the list of every special `@` commands.

@separator <char> When using `bconsole` with readline, you can set the command separator to one of those characters to write commands who require multiple input on one line, or to put multiple commands on a single line.

```
!$%&'()*+,-/:;<>?[]^`{|}~
```

Note, if you use a semicolon (;) as a separator character, which is common, you will not be able to use the `sql` command, which requires each command to be terminated by a semicolon.

1.8 Running the Console from a Shell Script

You can automate many Console tasks by running the console program from a shell script. For example, if you have created a file containing the following commands:

```
./bconsole -c ./bconsole.conf <<END_OF_DATA
unmount storage=DDS-4
quit
END_OF_DATA
```

when that file is executed, it will unmount the current DDS-4 storage device. You might want to run this command during a Job by using the `RunBeforeJob` or `RunAfterJob` records.

It is also possible to run the Console program from file input where the file contains the commands as follows:

```
./bconsole -c ./bconsole.conf <filename
```

where the file named filename contains any set of console commands.

As a real example, the following script is part of the Bacula regression tests. It labels a volume (a disk volume), runs a backup, then does a restore of the files saved.

```
bin/bconsole -c bin/bconsole.conf <<END_OF_DATA
@output /dev/null
messages
@output /tmp/log1.out
label volume=TestVolume001
run job=Client1 yes
wait
messages
@#
@# now do a restore
@#
@output /tmp/log2.out
restore current all
yes
wait
messages
@output
quit
END_OF_DATA
```

The output from the backup is directed to /tmp/log1.out and the output from the restore is directed to /tmp/log2.out. To ensure that the backup and restore ran correctly, the output files are checked with:

```
grep "^ *Termination: *Backup OK" /tmp/log1.out
backupstat=$?
grep "^ *Termination: *Restore OK" /tmp/log2.out
restorestat=$?
```

1.9 Adding Volumes to a Pool

If you have used the label command to label a Volume, it will be automatically added to the Pool, and you will not need to add any media to the pool.

Alternatively, you may choose to add a number of Volumes to the pool without labeling them. At a later time when the Volume is requested by Bacula you will need to label it.

Before adding a volume, you must know the following information:

1. The name of the Pool (normally "Default")
2. The Media Type as specified in the Storage resource in the Director's configuration file (e.g. "DLT8000")
3. The number and names of the Volumes you wish to create.

For example, to add media to a Pool, you would issue the following commands to the console program:

```
*add
Enter name of Pool to add Volumes to: Default
Enter the Media Type: DLT8000
Enter number of Media volumes to create. Max=1000: 10
Enter base volume name: Save
Enter the starting number: 1
10 Volumes created in pool Default
*
```

To see what you have added, enter:

```
*list media pool=Default
+-------+----------+---------+---------+-------+------------------+
| MedId | VolumeNa | MediaTyp| VolStat | Bytes | LastWritten      |
+-------+----------+---------+---------+-------+------------------+
|    11 | Save0001 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    12 | Save0002 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    13 | Save0003 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    14 | Save0004 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    15 | Save0005 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    16 | Save0006 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    17 | Save0007 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    18 | Save0008 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    19 | Save0009 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    20 | Save0010 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
+-------+----------+---------+---------+-------+------------------+
*
```

Notice that the console program automatically appended a number to the base Volume name that you specify (Save in this case). If you don't want it to append a number, you can simply answer 0 (zero) to the question "Enter number of Media volumes to create. Max=1000:", and in this case, it will create a single Volume with the exact name you specify.

Chapter 2

Baculum API and Web GUI Tools

This chapter presents the Bacula Web based interface that has been added to the Bacula project for version 7.0 and later.

2.1 New Features in 11.0.0

2.1.1 Upgrade to 11.0.0

Baculum 9.x and older does not work with Bacula 11.0 because of the new catalog format in Bacula 11.0.0.

Baculum 11.0 works with Bacula $<$ 11.0 but two file list specific functions does not work:

- searching jobs by filename in the restore wizard
- the file list on the finished backup job page

Fully supported are the following relations:

- Baculum 9.x with Bacula Director $<$ 11.0
- Baculum 11.0 with Bacula Director $>=$ 11.0

2.1.2 Multi-user interface improvements

There have been added new functions and improvements to the multi-user interface and restricted access.

The Security page has new tabs:

- Console ACLs
- OAuth2 clients
- API hosts

These new tabs help to configure OAuth2 accounts, create restricted Bacula Console for users and create API hosts. They ease the process of creating users with a restricted Bacula resources access.

2.1.3 Add searching jobs by filename in the restore wizard

In the restore wizard now is possible to select job to restore by filename of file stored in backups. There is also possible to limit results to specific path.

2.1.4 Show more detailed job file list

The job file list now displays file details like: file attributes, UID, GID, size, mtime and information if the file record for saved or deleted file.

2.1.5 Add graphs to job view page

On the job view page, new pie and bar graphs for selected job are available.

2.1.6 Implement graphical status storage

On the storage status page are available two new types of the status (raw and graphical). The graphical status page is modern and refreshed asynchronously.

2.1.7 Add Russian translations

2.1.8 Global messages log window

There has been added new window to browse Bacula logs in a friendly way.

2.1.9 Job status weather

Add the job status weather on job list page to express current job condition.

2.1.10 Restore wizard improvements

In the restore wizard has been added listing and browsing names encoded in non-UTF encoding.

2.1.11 New API endpoints

- /oauth2/clients
- /oauth2/clients/client_id
- /jobs/files

2.1.12 New parameters in API endpoints

- /jobs/jobid/files - 'details' parameter
- /storages/show - 'output' parameter
- /storages/storageid/show - 'output' parameter

Note Upgrade from Baculum 9.6 to Baculum 11.0 is fully supported both for installations using binary packages and using source archives.

2.2 New Features in 9.6.0

2.2.1 Upgrade to 9.6.4

There has been introduced new way of managing users. If you use default Basic authetication - no additional change is required. If you use custom auth method realized by web server - no additional change is requred. If it is LDAP auth please consider switching to new LDAP auth method in Baculum Web.

Import users

Importing Basic users from default user file to Baculum Web is done automatically. If you would like to import LDAP users to Baculum Web, you can use import option available on the Security page.

New dependency

There has been added a new dependency. It is PHP LDAP module. There is required to install this module (php-ldap package) after upgrade. Appropriate information about this dependency is displayed on the Baculum Web page. If you install Baculum using binary packages, no additional action is needed because this dependency will be installed automatically.

2.2.2 Support for commands that can take a long time

Some commands executed in bconsole can take a long time. They are for example: label tape volumes in autochanger, update slots without barcodes, estimate job command. To Baculum has been added support for these long time taking actions. Previously if command took time longer than 30 seconds, the request timed out.

Note On the Baculum API side has been changed way of sending requests for the above command. More details about it you can find in the Baculum API documentation.

2.2.3 Support for SELinux

There are new SELinux policy modules for Baculum API and Baculum Web. They can be applied manually or by installing new rpm packages available in Baculum repositories for CentOS and Fedora:

- baculum-api-selinux
- baculum-web-selinux

2.2.4 Graphical client status

In Baculum Web on the client page is available new graphical client client.

2.2.5 Graphical running job status

In Baculum Web on the running job page is available new graphical running job status that shows detailed information about current job. For backup job type the status also displays file and byte progress bars which base on estimated values.

2.2.6 Capability to start, stop and restart components

In Baculum API and Baculum Web are now available actions to start, stop and restart Bacula components. Actions can be defined by users and they can be executed from Baculum Web interface or directly by sending requests to Baculum API.

2.2.7 Statistics configuration

Statistics resource provides new feature in Bacula that enables saving Bacula component statistics to external databases such as Graphite or CSV file. The statistics are now configurable on the Baculum interface.

2.2.8 New graph types

In Baculum Web on graph page are available new graph types:

- Job size / hour
- Job size / day
- Average job size / hour
- Average job size / day
- Job files / time
- Job files / hour
- Job files / day
- Average job files / hour
- Average job files / day
- Job count / hour
- Job count / day
- Job duration / time
- Average job speed / Time
- Jobs status / day

2.2.9 Support for new directives

New Director directives support:

- Director resource:
  - CommCompression
- Job resource:
  - VirtualFullBackupPool

  - MaxVirtualFullInterval
  - BackupsToKeep
  - DeleteConsolidatedJobs
- JobDefs resource:
  - VirtualFullBackupPool
  - MaxVirtualFullInterval
  - BackupsToKeep
  - DeleteConsolidatedJobs
- Console resource:
  - RestoreClientAcl
  - BackupClientAcl
  - DirectoryAcl

New Storage Daemon directives support:

- Storage resource:
  - CommCompression
- Device resource:
  - WormCommand
  - Cloud
- Cloud resource:
  - Name
  - Description
  - Driver
  - HostName
  - BucketName
  - AccessKey
  - SecretKey
  - Region
  - Protocol
  - UriStyle
  - TruncateCache
  - Upload
  - MaximumConcurrentUploads
  - MaximumConcurrentDownloads
  - MaximumUploadBandwidth
  - MaximumDownloadBandwidth

New File Daemon directives support:

- FileDaemon resource:
  - CommCompression

New Console directives support:

- Console resource:
  - CommCompression

2.2.10 Changes in API endpoints

Please note that in Baculum API have been changed endpoints with access to API panel and with access OAuth2 tokens. New endpoints are follow:

- / - Baculum API panel
- /oauth/authorize - to authorize in authorization server
- /oauth/token - to get token from authorization server

For backward compatibility previous panel and OAuth2 endpoints are still available, but they will be removed in the future.

Note For future versions Baculum API users, who use OAuth2 authorization and call API by own scripts, need to switch in theris scripts old OAuth2 endpoints to the new ones. Users, who install Baculum API from source archive, have to update the web server configuration to support new endpoints. Users, who install Baculum from binary packages, do not need to take any additional action.

Note Starting from version 9.6.0 there has been finished support for old API endpoints that do not contain version 'v1' in paths.

2.2.11 New API functions

In Baculum API are added the following new functions:

- label volume with barcodes
- label volume without barcode
- update slots with barcodes
- update slots without barcodes
- start, stop and restart Bacula components
- status client
- set bandwidth limit for client
- set bandwidth limit for job
- list job files

2.2.12 New Web controls

The Baculum Web are added new controls to support:

- password directives with show/hide option
- speed type directives
- multiple the same Console ACL directives

2.2.13 Miscellaneous improvements

Miscellaneous improvements in Baculum API and Baculum Web:

- add option to show size unit values as decimal or binary bytes

- add version number to API and Web
- restore wizard improvements to use restore file browsers on different screen sizes and on mobile devices
- add on status client page client and job bandwidth limit setting
- add list job files tab to the job history page
- add job history list on job page

2.3 Base Features

Baculum provides the following base features:

- Running Bacula jobs (backup, restore, verify...).
- Baculum API with OAuth2 authorization and HTTP Basic authentication.
- Baculum Web GUI - modern mobile-friendly web interface.
- Configuring Bacula on local and remote hosts.
- Monitoring Bacula service status.
- Bacula console available via a Web window.
- Multi-user interface.
- Support for customized and restricted consoles (Console ACL function).
- Volume management.
- User friendly graphs and metrics.
- Basic storage daemon operations (mount, umount, release, ...).
- Easy to use configuration and restore wizards.
- Live AJAX based statuses.

To try Baculum features without installation, please visit the Baculum online demo page available at the following address:

https://baculum.app

2.4 General Requirements

Environment for Baculum Web installation should have following components installed:

- A web server - with URL rewrite module loaded. Baculum Web has been tested with Apache and Lighttpd web servers.
- PHP 5.4.0 or higher with following modules installed:
  - PHP cURL module
  - PHP DOM module
  - PHP JSON module
  - PHP LDAP module

Environment for Baculum API installation should have following components installed:

- A web server - with URL rewrite module loaded. Baculum has been tested with Apache and Lighttpd web servers.
- PHP 5.4.0 or higher with following modules installed:
  - PHP PDO support - depending on your catalog database: PDO PostgreSQL or PDO MySQL. Note, in case using MySQL database there is required to use MySQL native driver. It is php-mysqlnd for PHP, not php-mysql.
  - PHP BCMath module
  - PHP DOM module
  - PHP JSON module
- A working Bacula bconsole - configured Bacula text based console
- Access to the Bacula Catalog database (local or remote)
- In case using config module, read and write access to Bacula configuration files

With installation from binary packages (deb, rpm) all requirements will be automatically installed as packages dependencies.

2.5 Installation Baculum API from rpm binary packages

Note Before start using Baculum API and Baculum Web version 9.0.0 and later please backup your Bacula configuration in safe place. It is specially important because on first save config action the Bacula configuration is joined into one file per Bacula component.

For rpm binary there are the following packages:

- baculum-api - main Baculum API package with application files
- baculum-api-httpd - Apache web server configuration files for Baculum API
- baculum-api-lighttpd - Lighttpd web server configuration files for Baculum API
- baculum-api-selinux - SELinux policy module for Baculum API
- baculum-common - Common files for Baculum API and Baculum Web
- baculum-web - main Baculum Web package with application files
- baculum-web-httpd - Apache web server configuration files for Baculum Web
- baculum-web-lighttpd - Lighttpd web server configuration files for Baculum Web
- baculum-web-selinux - SELinux policy module for Baculum Web

2.5.1 Add the Baculum rpm repository

To add the Baculum repository, first you must import the Baculum public key:

```
rpm --import http://www.bacula.org/downloads/baculum/baculum.pub
```

Once the key is imported, the next step is to add the repository definition. First you must create the following file:

```
/etc/yum.repos.d/baculum.repo
```

For CentOS 7 the **baculum.repo** file should have the following content:

For Bacula Director <= 9.6

```
[baculumrepo]
name=Baculum CentOS repository
baseurl=http://www.bacula.org/downloads/baculum/stable/centos
gpgcheck=1
enabled=1
```

For Bacula Director >= 11.0

```
[baculumrepo]
name=Baculum CentOS repository
baseurl=http://www.bacula.org/downloads/baculum/stable-11/centos
gpgcheck=1
enabled=1
```

For CentOS 8 the **baculum.repo** file should have the following content:

For Bacula Director <= 9.6

```
[baculumrepo]
name=Baculum CentOS repository
baseurl=http://www.bacula.org/downloads/baculum/stable/centos8
gpgcheck=1
enabled=1
```

For Bacula Director >= 11.0

```
[baculumrepo]
name=Baculum CentOS repository
baseurl=http://www.bacula.org/downloads/baculum/stable-11/centos8
gpgcheck=1
enabled=1
```

For Fedora 33 the **baculum.repo** file should have the following content:

For Bacula Director <= 9.6

```
[baculumrepo]
name=Baculum Fedora repository
baseurl=http://www.bacula.org/downloads/baculum/stable/fedora33
gpgcheck=1
enabled=1
```

For Bacula Director >= 11.0

```
[baculumrepo]
name=Baculum Fedora repository
baseurl=http://www.bacula.org/downloads/baculum/stable-11/fedora33
gpgcheck=1
enabled=1
```

For Fedora 34 the **baculum.repo** file should have the following content:

For Bacula Director >= 11.0

```
[baculumrepo]
name=Baculum Fedora repository
baseurl=http://www.bacula.org/downloads/baculum/stable-11/fedora34
gpgcheck=1
enabled=1
```

2.5.2 Installation for Apache

Install Baculum API for the Apache web server as follows:

```
yum install baculum-common baculum-api baculum-api-httpd
```

Restart your Apache web server:

```
systemctl restart httpd
```

2.5.3 Installation for Lighttpd

Installation on system with access via Lighttpd is as follows:

```
yum install baculum-common baculum-api baculum-api-lighttpd
```

Please note that in case CentOS distribution the Lighttpd web server is available in the distribution packages after enabling the EPEL repository.

Start Baculum API as application using the Lighttpd web server:

```
systemctl start baculum-api-lighttpd
```

2.5.4 SELinux support

To enable Baculum API support for SELinux is needed to install the following binary package:

```
yum install baculum-api-selinux
```

2.5.5 Access to bconsole via sudo for Apache and Lighttpd

Baculum API requires access to Bconsole and to Bacula JSON programs. To configure Bconsole sudo access and the Bacula JSON programs access there can use following entries in newly created Baculum sudoers.d file (usually in path /etc/sudoers.d/baculum):

Note, please define sudo for the Bacula JSON programs only when you are going use Bacula configuration module in Baculum.

In case default Apache user, the file contents must be:

```
Defaults:apache !requiretty
apache  ALL=NOPASSWD:  /usr/sbin/bconsole
apache  ALL=NOPASSWD:  /usr/sbin/bdirjson
apache  ALL=NOPASSWD:  /usr/sbin/bsdjson
apache  ALL=NOPASSWD:  /usr/sbin/bfdjson
apache  ALL=NOPASSWD:  /usr/sbin/bbconsjson
```

In case default Lighttpd user the file contents must be:

```
Defaults:lighttpd !requiretty
lighttpd  ALL=NOPASSWD:  /usr/sbin/bconsole
lighttpd  ALL=NOPASSWD:  /usr/sbin/bdirjson
lighttpd  ALL=NOPASSWD:  /usr/sbin/bsdjson
lighttpd  ALL=NOPASSWD:  /usr/sbin/bfdjson
lighttpd  ALL=NOPASSWD:  /usr/sbin/bbconsjson
```

2.6 Installation Baculum API from deb binary packages

Note Before start using Baculum API and Baculum Web version 9.0.0 and later please backup your Bacula configuration in safe place. It is specially important because on first save config action the Bacula configuration is joined into one file per Bacula component.

For deb binary there are the following packages:

- baculum-api - main Baculum API package with application files
- baculum-api-apache2 - Apache web server configuration files for Baculum API
- baculum-api-lighttpd - Lighttpd web server configuration files for Baculum API
- baculum-common - Common files for Baculum API and Baculum Web
- baculum-web - main Baculum Web package with application files
- baculum-web-apache2 - Apache web server configuration files for Baculum Web
- baculum-web-lighttpd - Lighttpd web server configuration files for Baculum Web

2.6.1 Add the Baculum deb repository

To add the Baculum repository, first import the Baculum public key:

```
wget -qO - http://www.bacula.org/downloads/baculum/baculum.pub | apt-key add -
```

Once the key is imported, the next step is to create a new baculum file:

```
/etc/apt/sources.list.d/baculum.list
```

For Debian 9 Stretch the **baculum.list** file should have the following content:

For Bacula Director <= 9.6

```
deb http://www.bacula.org/downloads/baculum/stable/debian stretch main
deb-src http://www.bacula.org/downloads/baculum/stable/debian stretch main
```

For Bacula Director >= 11.0

```
deb http://www.bacula.org/downloads/baculum/stable-11/debian stretch main
deb-src http://www.bacula.org/downloads/baculum/stable-11/debian stretch main
```

For Debian 10 Buster the **baculum.list** file should have the following content:

For Bacula Director <= 9.6

```
deb http://www.bacula.org/downloads/baculum/stable/debian buster main
deb-src http://www.bacula.org/downloads/baculum/stable/debian buster main
```

For Bacula Director >= 11.0

```
deb http://www.bacula.org/downloads/baculum/stable-11/debian buster main
deb-src http://www.bacula.org/downloads/baculum/stable-11/debian buster main
```

For Debian 11 Bullseye the **baculum.list** file should have the following content:

For Bacula Director >= 11.0

```
deb http://www.bacula.org/downloads/baculum/stable-11/debian bullseye main
deb-src http://www.bacula.org/downloads/baculum/stable-11/debian bullseye main
```

For Ubuntu 18.04 Bionic the **baculum.list** file should have the following content:

For Bacula Director <= 9.6

```
deb [ arch=amd64 ] http://www.bacula.org/downloads/baculum/stable/ubuntu bionic main
deb-src http://www.bacula.org/downloads/baculum/stable/ubuntu bionic main
```

For Bacula Director >= 11.0

```
deb [ arch=amd64 ] http://www.bacula.org/downloads/baculum/stable-11/ubuntu bionic main
deb-src http://www.bacula.org/downloads/baculum/stable-11/ubuntu bionic main
```

For Ubuntu 20.04 Focal the **baculum.list** file should have the following content:

For Bacula Director <= 9.6

```
deb [ arch=amd64 ] http://www.bacula.org/downloads/baculum/stable/ubuntu focal main
deb-src http://www.bacula.org/downloads/baculum/stable/ubuntu focal main
```

For Bacula Director >= 11.0

```
deb [ arch=amd64 ] http://www.bacula.org/downloads/baculum/stable-11/ubuntu focal main
deb-src http://www.bacula.org/downloads/baculum/stable-11/ubuntu focal main
```

After adding repository definition, please refresh repository indexes:

```
apt-get update
```

2.6.2 Installation for Apache

To install Baculum API access via Apache web server by using apt packages manager use the command:

```
apt-get install baculum-common baculum-api baculum-api-apache2
```

Next you must enable mod_rewrite module for Apache, with the following command:

```
a2enmod rewrite
```

and include Baculum VirtualHost definition in the Apache configuration with:

```
a2ensite baculum-api
```

Then restart your Apache server with:

```
systemctl restart apache2
```

2.6.3 Installation for Lighttpd

Example installation with access via Lighttpd web server looks following:

```
apt-get install baculum-common baculum-api baculum-api-lighttpd
```

Start Baculum API as application available through Lighttpd web server:

```
systemctl start baculum-api-lighttpd
```

2.6.4 Access to bconsole via sudo for Apache and Lighttpd

Baculum API requires access to Bconsole and to the Bacula JSON programs. To configure Bconsole sudo access, we strongly recommend that you create a Baculum sudoers.d file, which should be in /etc/sudoers.d/baculum:

Note, please define sudo for the Bacula JSON programs only when you are going use Bacula configuration module in Baculum.

Both for Apache and Lighttpd user the file contents can be:

```
Defaults:www-data !requiretty
www-data  ALL=NOPASSWD:  /usr/sbin/bconsole
www-data  ALL=NOPASSWD:  /usr/sbin/bdirjson
www-data  ALL=NOPASSWD:  /usr/sbin/bsdjson
www-data  ALL=NOPASSWD:  /usr/sbin/bfdjson
www-data  ALL=NOPASSWD:  /usr/sbin/bbconsjson
```

2.7 Debugging your First Baculum API Login

At each step of the initial login to Baculum, the screen will have a test button, that will allow you to check if your parameters were correctly entered. If not, you will see error message on the wizard page. You can also get additional detail by examining the Apache error log, that is usually found at:

```
/var/log/httpd/baculum-api-error.log
```

If you use Lighttpd thento get additional detail you can check:

```
/var/log/lighttpd/baculum-api-error.log
```

In addition, special debug output is placed by Baculum in the file:

```
/usr/share/baculum/htdocs/protected/API/Logs/baculum-api.log
```

The debug you can enable in file:

```
/usr/share/baculum/htdocs/protected/API/Config/api.conf
```

by switching in [api] section option debug to "1".

With the information in those two files, you can usually quickly find and correct most problems.

2.8 Installation Baculum Web from rpm binary packages

2.8.1 Installation for Apache

Install Baculum Web for the Apache web server as follows:

```
yum install baculum-common baculum-web baculum-web-httpd
```

Restart your Apache web server:

```
systemctl restart httpd
```

2.8.2 Installation for Lighttpd

Installation on system with access via Lighttpd is as follows

```
yum install baculum-common baculum-web baculum-web-lighttpd
```

Please note that in case CentOS distribution the Lighttpd web server is available in the distribution packages after enabling the EPEL repository.

Start Baculum as application using the Lighttpd web server:

```
systemctl start baculum-web-lighttpd
```

2.8.3 SELinux support

To enable Baculum Web support for SELinux is needed to install the following binary package:

```
yum install baculum-web-selinux
```

2.9 Installation Baculum Web from deb binary packages

2.9.1 Installation for Apache

To install Baculum Web access via Apache web server by using apt packages manager use the command:

```
apt-get install baculum-common baculum-web baculum-web-apache2
```

Next you must enable mod_rewrite module for Apache, with the following command:

```
a2enmod rewrite
```

and include Baculum VirtualHost definition in the Apache configuration with:

```
a2ensite baculum-web
```

The restart your Apache server with:

```
systemctl restart apache2
```

2.9.2 Installation for Lighttpd

Example installation with access via Lighttpd web server looks following:

```
apt-get install baculum-common baculum-web baculum-web-lighttpd
```

Start Baculum Web as application available through Lighttpd web server:

```
systemctl start baculum-web-lighttpd
```

2.10 Running Baculum API and Web for the First Time

2.10.1 Running Baculum API

Access to Baculum API from web browser: **http://localhost:9096**

First time login: **admin**

First time password: **admin**

2.10.2 Running Baculum Web

Access to Baculum Web from web browser: **http://localhost:9095**

First time login: **admin**

First time password: **admin**

2.10.3 Installation wizards

Installation with HTTP Basic authentication

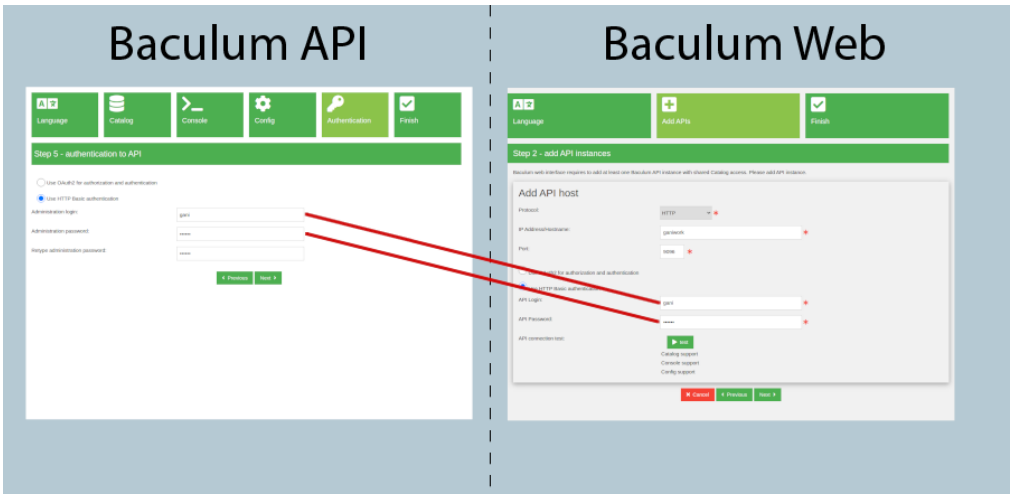


Figure 2.1: Baculum installation with HTTP Basic authentication

Installation with OAuth2 authorization

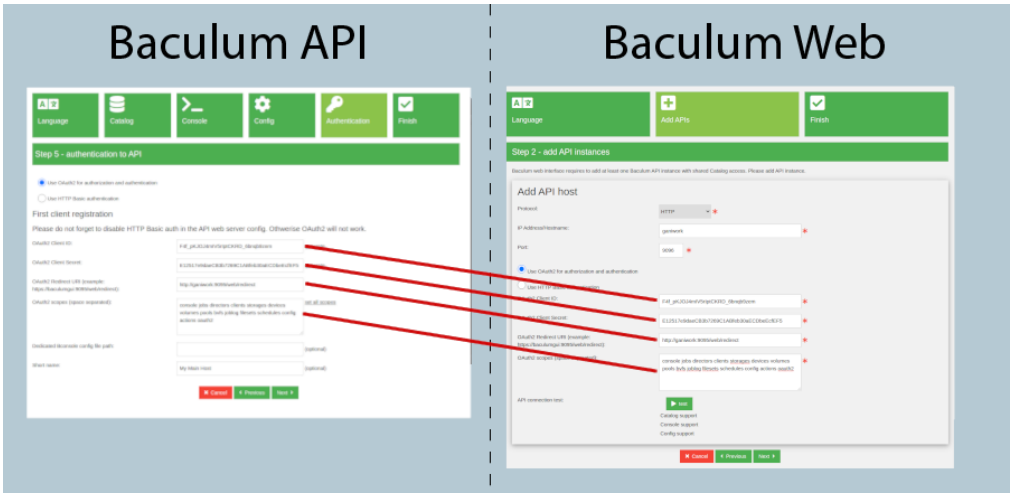


Figure 2.2: Baculum installation with HTTP Basic authentication

2.10.4 Configuring Bacula

During first use the Bacula configuration feature when 'Save' button is clicked, there can be visible an permission error similar to this error:

```
Error 1000: Internal error.
[Warning] file_put_contents(/etc/bacula/bacula-dir.conf): failed to open stream:
Permission denied (@line 56 in file
/usr/share/baculum/htdocs/protected/Common/Class/ConfigBacula.php).
```

It means that the Baculum API web server user does not have permission to write Bacula configuration file. To solve it, please setup permissions for Bacula configuration files to allow web server user have write access.

2.11 Baculum API documentation

2.11.1 API version 1

The documentation for Baculum API version 1 is placed at the following link:

https://www.bacula.org/downloads/baculum/baculum-api-v1/

2.11.2 API version 2

The documentation for Baculum API version 2 is placed at the following link:

https://www.bacula.org/downloads/baculum/baculum-api-v2/

Changes in API version 2:

- Send request body parameters to the API in JSON format instead of POST form parameters.
- Drop using the create[] and update[] surrounds in the POST and PUT request body parameters.
- Move the /api/v1/status/{director|storage|client}/ endpoints to: /api/v2/clients/{clientid}/status, /api/v2/storages/{storageid}/status, /api/v2/directors/{director_name}/status.
- Unify /jobs/{jobid}/files endpoint output for detailed and normal modes.
- Stop using OAuth2 'status' scope.
- API version 1 is still possible to use and it is preserved. Nothing changes here.
- New and modern API admin panel.

Example usage

Example request to run a job when the OAuth2 authorization is enabled:

```
curl \
  -X POST \
  'https://baculum-api:9096/api/v2/jobs/run' \
```

```
  -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer 6bd7c887e768efb9a031d0545aa0552de0140fe2' \
  --data-raw '{"name": "BackupClient1", "level": "F", "client": "darkstar-fd","storage": "UP","poo
```

where:

- API instance address is: baculum-api
- OAuth2 access token is: 6bd7c887e768efb9a031d0545aa0552de0140fe2

Example request to run a job when the Basic authentication is enabled:

```
curl \
  -X POST \
  'https://baculum-api:9096/api/v2/jobs/run' \
  --basic \
  -u 'myuser:mypass' \
  -H 'Content-Type: application/json' \
  --data-raw '{"name": "BackupClient1", "level": "F", "client": "darkstar-fd","storage": "UP","poo
```

where:

- API instance address is: baculum-api
- Basic user is: myuser
- Basic user password is: mypass

2.12 Installation from the source tar file

There is possible to install Baculum from the source bacula-gui tar archive. To install please unpack the bacula-gui source archive and go to this unpacked directory:

```
cd bacula-gui-9.6.0/baculum/
```

Then please prepare the Baculum API and the Baculum Web files depending on used distribution in the way described below.

In this description the document root directory for web files is path: **/var/www/baculum**. This location can be changed during installation in the **WWWDIR** parameter value.

2.12.1 Manual installation on rpm-based Linux distributions

To prepare all Baculum files to installation please execute from the source files path the following command:

```
make build DESTDIR=/tmp/baculum-files WWWDIR=/var/www/baculum
```

After executing above command, the directory **/tmp/baculum-files** should contain all required files ready to copy to destination system paths.

Install the Baculum API and the Baculum Web dependencies:

```
yum install httpd php php-common php-pdo php-mysqlnd php-pgsql php-bcmath php-json php-xml
```

Copy to the destination path all Baculum web type files:

```
cp -R /tmp/baculum-files/var/www/baculum/ /var/www
```

Copy the web server configuration files:

```
cp /tmp/baculum-files/etc/httpd/conf.d/baculum-*conf /etc/httpd/conf.d/
```

Copy the HTTP Basic authentication files:

```
cp /tmp/baculum-files/etc/baculum/Config-api-apache/baculum.users /var/www/baculum/protected
cp /tmp/baculum-files/etc/baculum/Config-web-apache/baculum.users /var/www/baculum/protected
```

Copy translation files:

```
cp --remove-destination /tmp/baculum-files/usr/share/locale/en/LC_MESSAGES/baculum-api.mo /v
cp --remove-destination /tmp/baculum-files/usr/share/locale/pl/LC_MESSAGES/baculum-api.mo /v
cp --remove-destination /tmp/baculum-files/usr/share/locale/pt/LC_MESSAGES/baculum-api.mo /v
cp --remove-destination /tmp/baculum-files/usr/share/locale/ru/LC_MESSAGES/baculum-api.mo /v
cp --remove-destination /tmp/baculum-files/usr/share/locale/en/LC_MESSAGES/baculum-web.mo /v
cp --remove-destination /tmp/baculum-files/usr/share/locale/pl/LC_MESSAGES/baculum-web.mo /v
cp --remove-destination /tmp/baculum-files/usr/share/locale/pt/LC_MESSAGES/baculum-web.mo /v
cp --remove-destination /tmp/baculum-files/usr/share/locale/ja/LC_MESSAGES/baculum-web.mo /v
cp --remove-destination /tmp/baculum-files/usr/share/locale/ru/LC_MESSAGES/baculum-web.mo /v
```

Set recursively owner and group for files and directories:

```
chown -R apache:apache /var/www/baculum
```

Prepare and install the SELinux module (if SELinux is used in operating system):

```
yum install selinux-policy-devel
```

```
make -C examples/selinux/ -f /usr/share/selinux/devel/Makefile baculum-api.pp
make -C examples/selinux/ -f /usr/share/selinux/devel/Makefile baculum-web.pp
install -D -m 644 examples/selinux/baculum-api.pp /usr/share/selinux/packages/baculum-api/ba
install -D -m 644 examples/selinux/baculum-web.pp /usr/share/selinux/packages/baculum-web/ba
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/baculum/protected/API/Config(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/baculum/protected/API/Logs(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/baculum/protected/Web/Config(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/baculum/protected/Web/Logs(/.*)?'
semanage fcontext -a -t httpd_cache_t '/var/www/baculum/assets(/.*)?'
semanage fcontext -a -t httpd_cache_t '/var/www/baculum/protected/runtime(/.*)?'
restorecon -i -R '/var/www/baculum/protected/API/Config' '/var/www/baculum/protected/API/Log
semodule -i /usr/share/selinux/packages/baculum-api/baculum-api.pp
semodule -i /usr/share/selinux/packages/baculum-web/baculum-web.pp
```

Start the Apache web server

```
systemctl start httpd
```

2.12.2 Manual installation on deb-based Linux distributions

To prepare all Baculum files to installation please execute from the source files path the following command:

```
make build DESTDIR=/tmp/baculum-files SAMPLETYPE=deb-template HTTPDNAME=apache2 HTTPDSITECONF=site
```

After executing above command, the directory **/tmp/baculum-files** should contain all required files ready to copy to destination system paths.

Install the Baculum Web and the Baculum API dependencies:

```
apt-get install apache2 libapache2-mod-php php-bcmath php-cgi php-mysql php-pgsql php-json php-xml
```

Copy to the destination path all Baculum web type files:

```
cp -R /tmp/baculum-files/var/www/baculum/ /var/www
```

Copy the web server configuration files:

```
cp /tmp/baculum-files/etc/apache2/sites-available/baculum-*conf /etc/apache2/sites-available/
```

Copy the HTTP Basic authentication files:

```
cp /tmp/baculum-files/etc/baculum/Config-api-apache/baculum.users /var/www/baculum/protected/API/C
cp /tmp/baculum-files/etc/baculum/Config-web-apache/baculum.users /var/www/baculum/protected/Web/C
```

Copy translation files:

```
cp --remove-destination /tmp/baculum-files/usr/share/locale/en/LC_MESSAGES/baculum-api.mo /var/www
cp --remove-destination /tmp/baculum-files/usr/share/locale/pl/LC_MESSAGES/baculum-api.mo /var/www
cp --remove-destination /tmp/baculum-files/usr/share/locale/pt/LC_MESSAGES/baculum-api.mo /var/www
cp --remove-destination /tmp/baculum-files/usr/share/locale/ru/LC_MESSAGES/baculum-api.mo /var/www
cp --remove-destination /tmp/baculum-files/usr/share/locale/en/LC_MESSAGES/baculum-web.mo /var/www
cp --remove-destination /tmp/baculum-files/usr/share/locale/pl/LC_MESSAGES/baculum-web.mo /var/www
cp --remove-destination /tmp/baculum-files/usr/share/locale/pt/LC_MESSAGES/baculum-web.mo /var/www
cp --remove-destination /tmp/baculum-files/usr/share/locale/ja/LC_MESSAGES/baculum-web.mo /var/www
cp --remove-destination /tmp/baculum-files/usr/share/locale/ru/LC_MESSAGES/baculum-web.mo /var/www
```

Set recursively owner and group for files and directories:

```
chown -R www-data:www-data /var/www/baculum
```

Enable the web server Baculum API and Baculum Web sites:

```
a2ensite baculum-api
a2ensite baculum-web
```

Enable the URL rewrite module for the Apache web server:

```
a2enmod rewrite
```

Start the Apache web server

```
systemctl start apache2
```

2.12.3 Validating manual installation

To check manually installed the Baculum API and the Baculum Web files please use a script, which checks whether files and directories exist in proper paths, if they have set valid permissions, ownership and other requirements.

To check files for installation with Apache web server:

```
/tmp/baculum-files/baculum-install-checker.sh -a
```

To check files for installation with Lighttpd web server:

```
/tmp/baculum-files/baculum-install-checker.sh -l
```

2.13 OAuth2 authorization

In Baculum API you can setup OAuth2 for authorization and authentication.

To get an access token there is used **Authorization Code Grant** flow. Authorization and access token URLs are as follow:

```
Authorization URL: /oauth/authorize
Access Token URL: /oauth/token
```

Baculum API does not use refresh tokens. After expiration token the client application has to re-authorize again.

Default expiration time for authorization code is 7 seconds, for access token 120 seconds. These values can be changed in:

```
/usr/share/baculum/htdocs/protected/Common/Class/OAuth2.php
```

in constants AUTHORIZATION_ID_EXPIRES_TIME and ACCESS_TOKEN_EXPIRES_TIME.

Default OAuth2 callback URL in Baculum Web is following:

```
https://baculumgui:9095/web/redirect
```

2.13.1 Before running OAuth2

Important note before using OAuth2

When you decide to use OAuth2, you must change default HTTP Basic authentication setting. Otherwise OAuth2 will not work. It is all about enabling OAuth2 acces to /api/ endpoints but still keeping the HTTP Basic protection for the Baculum API panel pages.

For Apache this change consists in replacing in the Baculum API Apache config the Directory tag /usr/share/baculum/htdocs into Location tag /panel

#

```
# NOTE: When you use OAuth2 then change this Directory section
# From: <Directory /usr/share/baculum/htdocs>
#            ...section body...
#       </Directory>
# To:   <Location /panel>
#            ...section body...
#       </Location>
#
<Directory /usr/share/baculum/htdocs>
#<Location /panel>
        AuthType Basic
        AuthName "Baculum Auth"
        AuthUserFile /usr/share/baculum/htdocs/protected/API/Config/baculum.users
        Require valid-user
#</Location>
</Directory>
```

For Lighttpd this change consists in uncommenting in the Baculum API Lighttpd config lines as shown in the comment below.

```
#
# Uncomment this line and closing braces below when you use OAuth2
#
#$HTTP["url"] =~ "^/panel.*$" {
        auth.backend = "htpasswd"
        auth.backend.htpasswd.userfile = "/usr/share/baculum/htdocs/protected/API/Config/baculum.u
        auth.require = ( "/" => (
                "method" => "basic",
                "realm" => "Baculum Auth",
                "require" => "valid-user"
                )
        )
#}
```

2.14 Multi-user interface

Baculum enables access to Bacula resources for defined users, where every user uses own resources (Jobs, Clients, FileSets ...etc.). These resources are assigned to users by the Bacula Restricted Consoles and then they are used by Baculum.

To setup this multi-user interface there is needed to enable the OAuth2 authorization on Baculum API hosts, that are used by the Baculum Web interface. There is also necessary to configure in the Director the Console resources for users and Bconsole config files dedicated for them.

Note Since Baculum version 9.6.6.1 all the process described below can be done directly from the Baculum Web interface.

Minimal Console resource configuration can look as below. These **CommandAcl** values in the configuration are required to proper working all available functions for normal Baculum users (run job, restore backup, cancel job, delete job and others).

```
Console {
  Name = "Limited User 144"
  Password = "A6cTimESfLs7xPNOMC/ein92BF4="
  JobAcl = "BackupCatalog"
```

```
  JobAcl = "RestoreFiles"
  ClientAcl = "myhost-fd"
  StorageAcl = "File1"
  PoolAcl = "File"
  CommandAcl = "gui"
  CommandAcl = ".api"
  CommandAcl = ".jobs"
  CommandAcl = ".ls"
  CommandAcl = ".client"
  CommandAcl = ".fileset"
  CommandAcl = ".pool"
  CommandAcl = ".status"
  CommandAcl = ".storage"
  CommandAcl = ".bvfs_get_jobids"
  CommandAcl = ".bvfs_update"
  CommandAcl = ".bvfs_lsdirs"
  CommandAcl = ".bvfs_lsfiles"
  CommandAcl = ".bvfs_versions"
  CommandAcl = ".bvfs_restore"
  CommandAcl = ".bvfs_cleanup"
  CommandAcl = "restore"
  CommandAcl = "show"
  CommandAcl = "estimate"
  CommandAcl = "run"
  CommandAcl = "delete"
  CommandAcl = "cancel"
  FilesetAcl = "Full Set"
  CatalogAcl = "MyCatalog"
  WhereAcl = "/tmp/restore"
}
```

Assigning Restricted Consoles to users is realized during configuring OAuth2 accounts on API hosts panel as on attached screenshot.

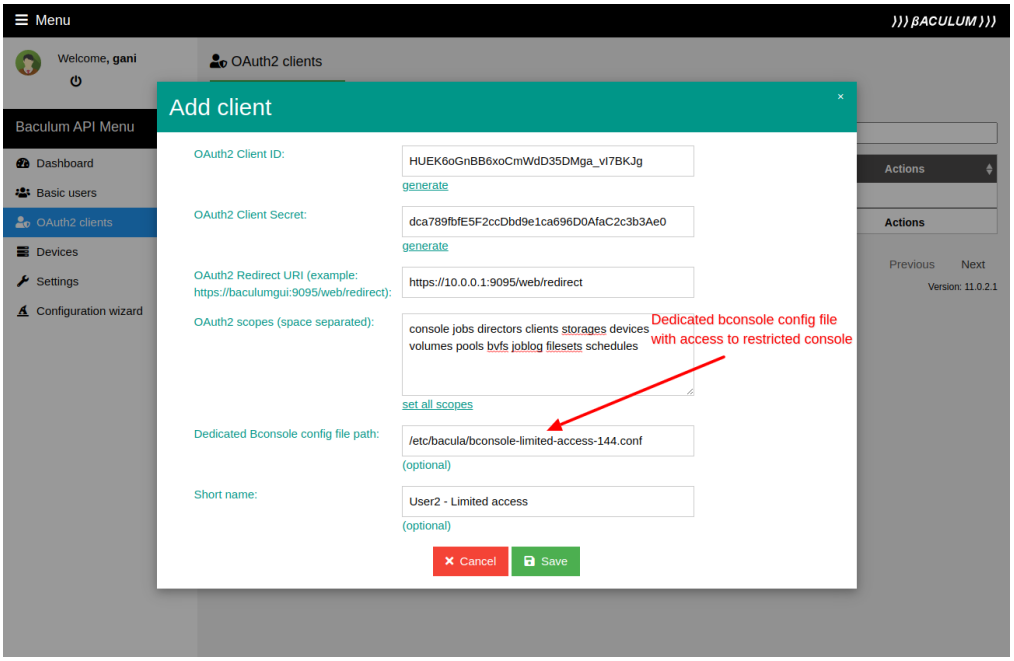


Figure 2.3: Add OAuth2 user account

Please note that in the OAuth2 scopes there is not "config" scope defined, because normal users

don't have access to configuring Bacula resources by Baculum API hosts.

Once the OAuth2 accounts with assigned dedicated consoles are done, now you can connect Baculum Web to those new Baculum API OAuth2 accounts.

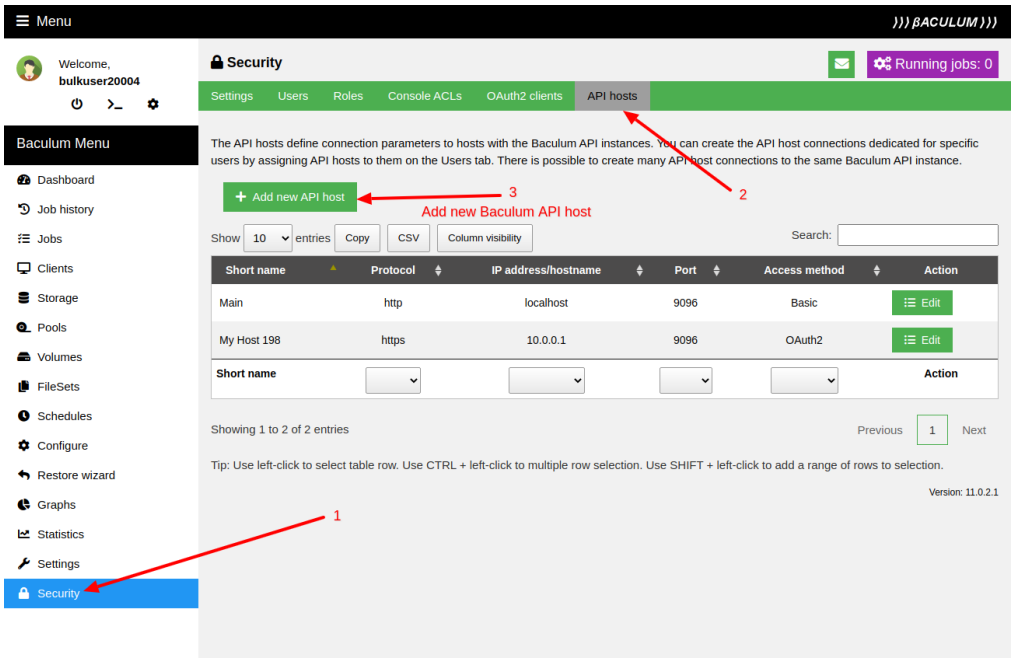


Figure 2.4: Add new API host

At the end there is needed to create Baculum Web users and assign them to appropriate OAuth2 user accounts on API hosts.

Images below show example access to Baculum API hosts by a normal user and by an admin user. In this setting the administrator is able to manage Host A, Host B, Host C and the normal user is able to use Host B only.

Note Since version 11.0.2.2 there is possible to assign multiple API hosts to one user account.

Normal (non-admin) user interface can look as on screenshot:

2.14.1 Multi-user interface setup in steps

Below you can find list with steps needed to setup the Baculum multi-user interface:

1. Install Baculum API.
2. Change the Baculum API web server config file to support OAuth2 and restart the web server.
3. Go to the Baculum API installation wizard and fill needed forms on wizard steps.
4. In "Authentication" step please select OAuth2 authorization option and define there administrator OAuth2 client account. It is an account without any dedicated Bconsole config file path (full bconsole access) and with all scopes.
5. Finish the Baculum API wizard.
6. Install Baculum Web. Go to the installation wizard and fill forms on each wizard step.

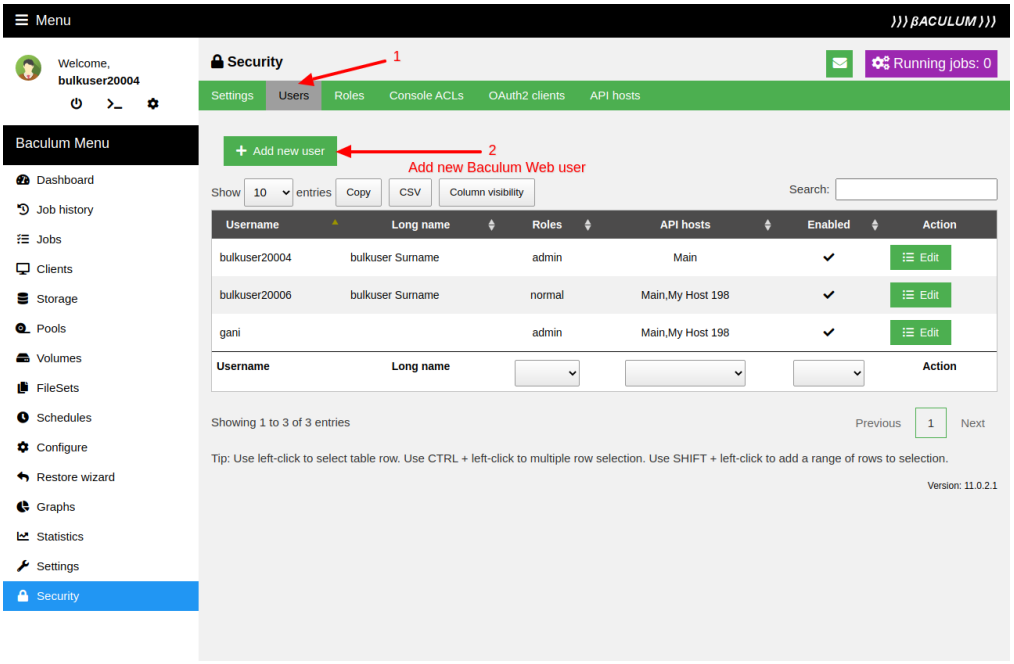


Figure 2.5: Define user to API host

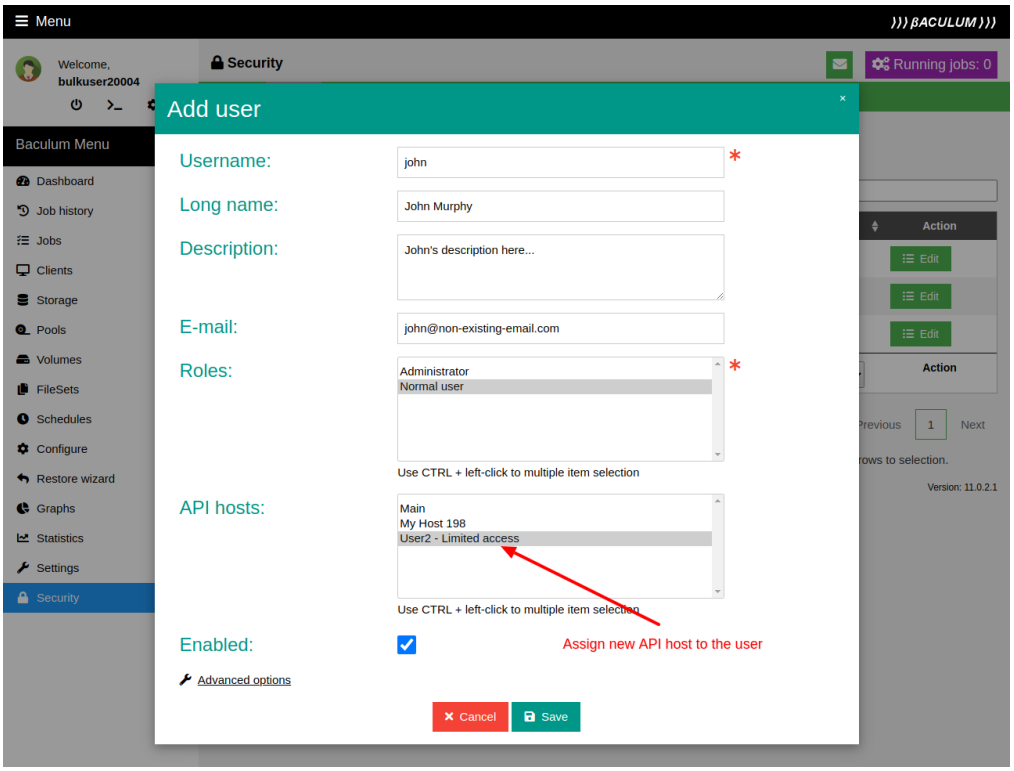


Figure 2.6: Assing user to API host

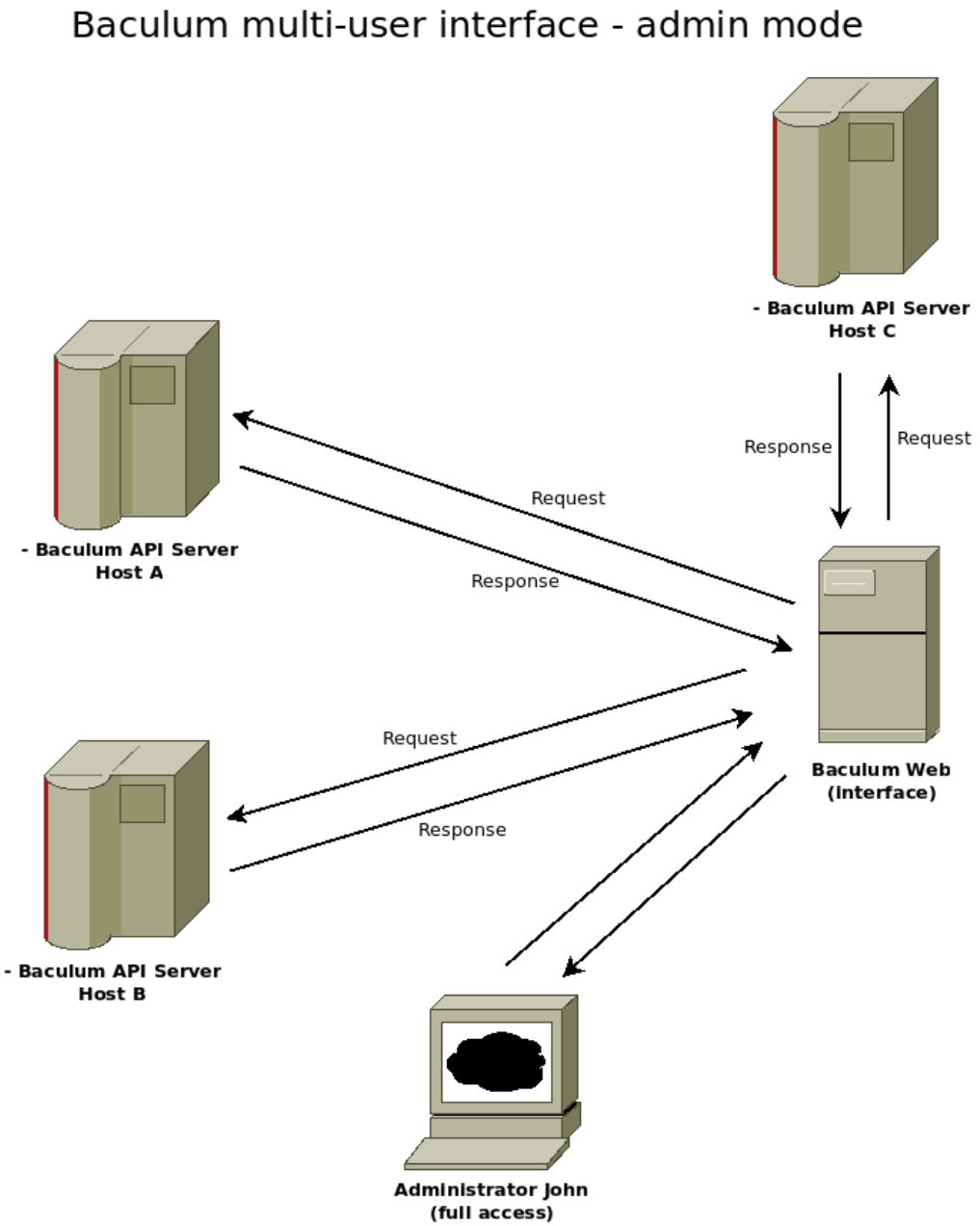


Figure 2.7: Multi user interface - admin user

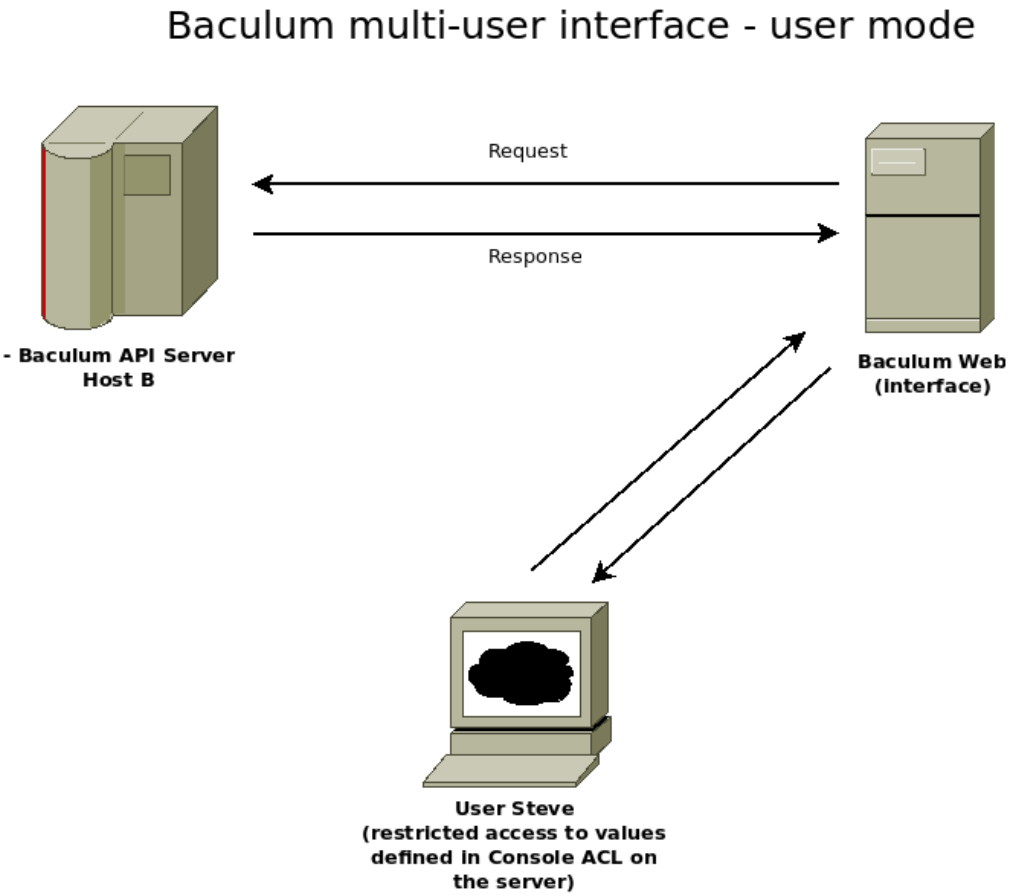


Figure 2.8: Multi user interface - normal user

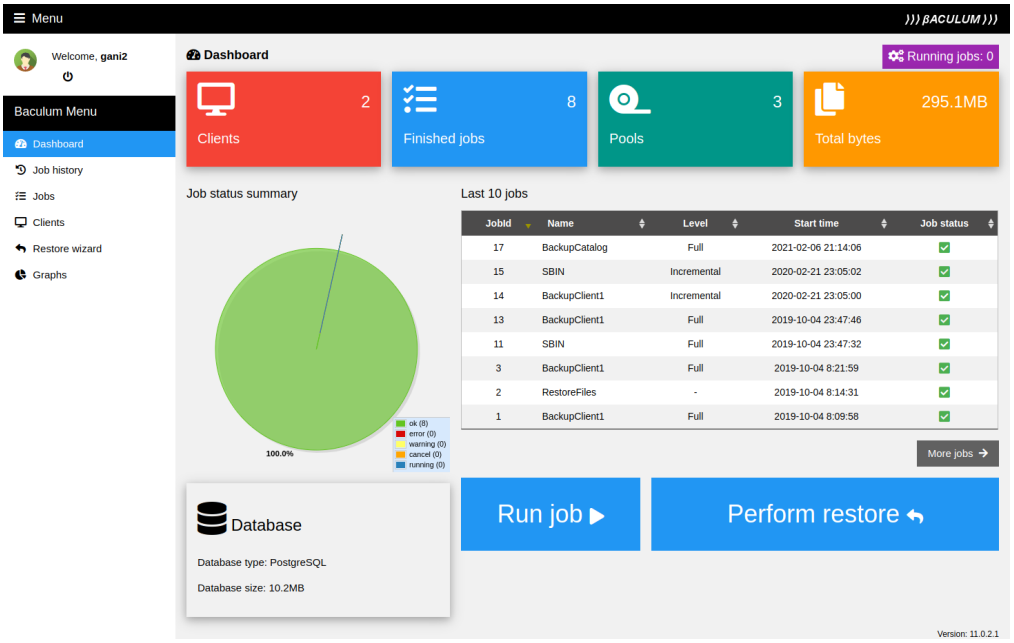


Figure 2.9: Normal user dashboard

7. In the "Add API host" step please select OAuth2 authorization option and please provide the administrator OAuth2 account parameters defined previously in the Baculum API installation wizard.
8. Finish the Baculum Web wizard.
9. Define in the Bacula Director configuration dedicated Consoles for users. You can do it on the Baculum interface or manually in the Director configuration file. At the end please reload the Director configuration.
10. In Baculum API panel on the "OAuth2 clients" page create new OAuth2 client accounts, each account for separate user or group of users. For each OAuth2 account in field "Dedicated Bconsole config file path" define separate Bconsole config file with access only to selected restricted console (the bconsole configuration files must be prepared manually before this step). In scopes field please be careful to not set "config" or "actions" scopes to regular user if it isn't your intention.
11. In Baculum Web on the "Security" page in "API hosts" tab please click "Add API host" button and add there created in previous point all OAuth2 client accounts.
12. Go to the "Users" tab on the "Security" page, create new users and assign API hosts to them.

Note Since Baculum version 9.6.6.1 if your use the OAuth2 account with 'oauth2' scope you can perform all above steps, that are doing on API side (add OAuth2 client, create dedicated Bconsole configuration file), directly in the Baculum Web interface on the Security page.

2.15 Autochanger management

From version 11.0.2.1 Baculum provides an interface to manage tape autochangers. This function enables to do the following tasks:

- Load tape from slot to tape drive (with and without mounting tape).
- Unload tape from tape drive to slot.
- Update slots using barcordes.
- Update slots reading labels directly from volumes (for changers without barcode reader).
- Move tape(s) to import/export slots.
- Release single import/export slot.
- Release all import/export slots at once.

To manage autochanger there is needed to define autochanger tape drives and autochanger device setting on dedicated Device page for that available in the Baculum API panel. After that the autochanger management should be available on the Baculum Web side on the Storage page after selecting the Storage resource associated with the autochanger device.

2.16 Screenshots

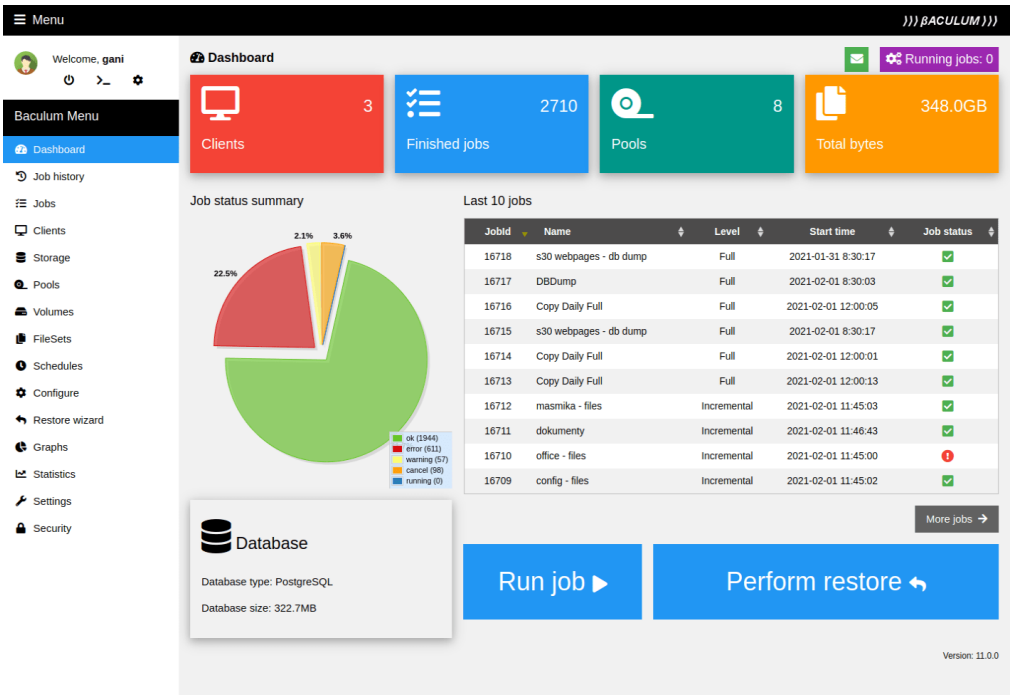


Figure 2.10: Baculum dashboard

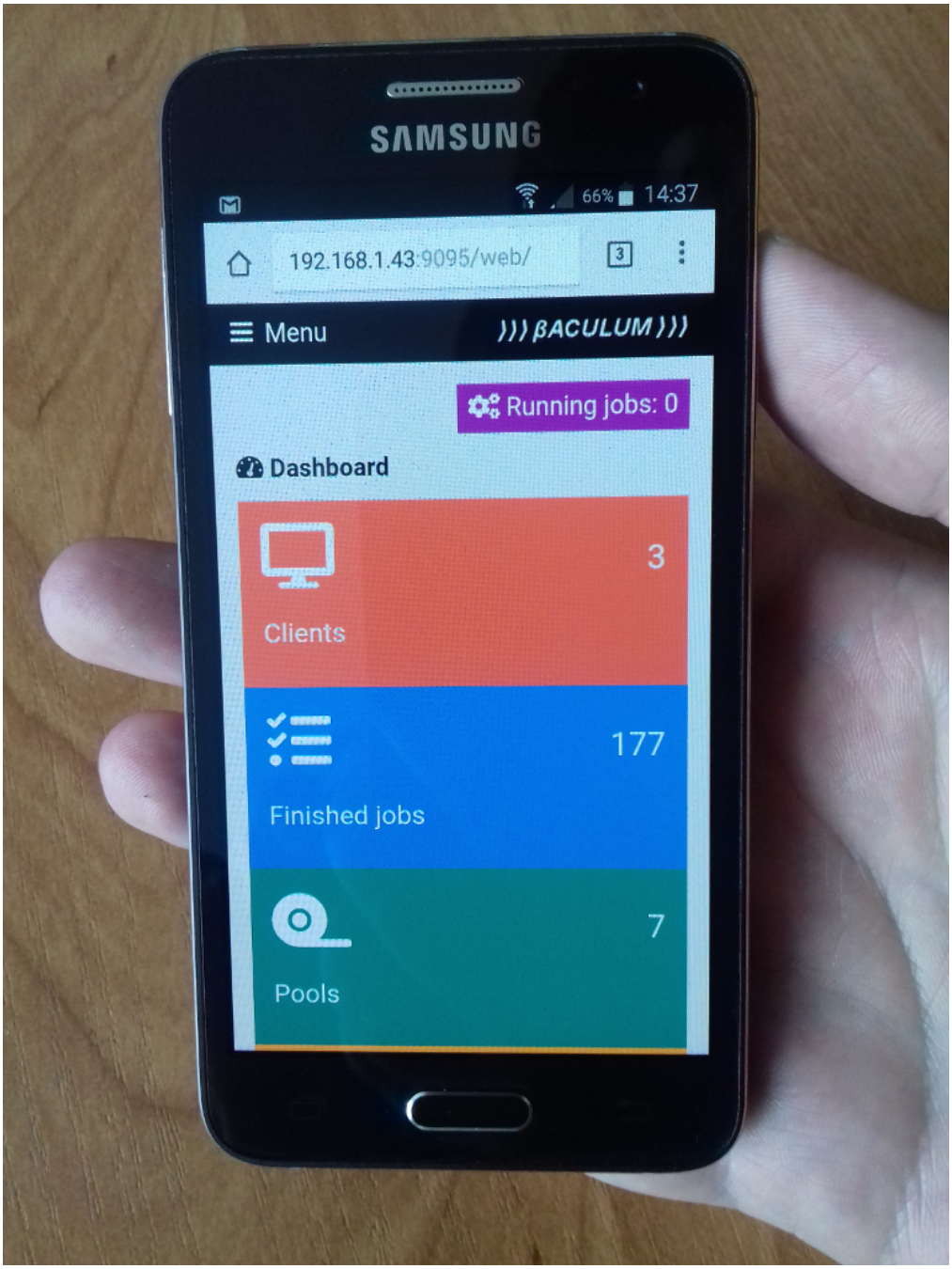


Figure 2.11: Dashboard on mobile phone

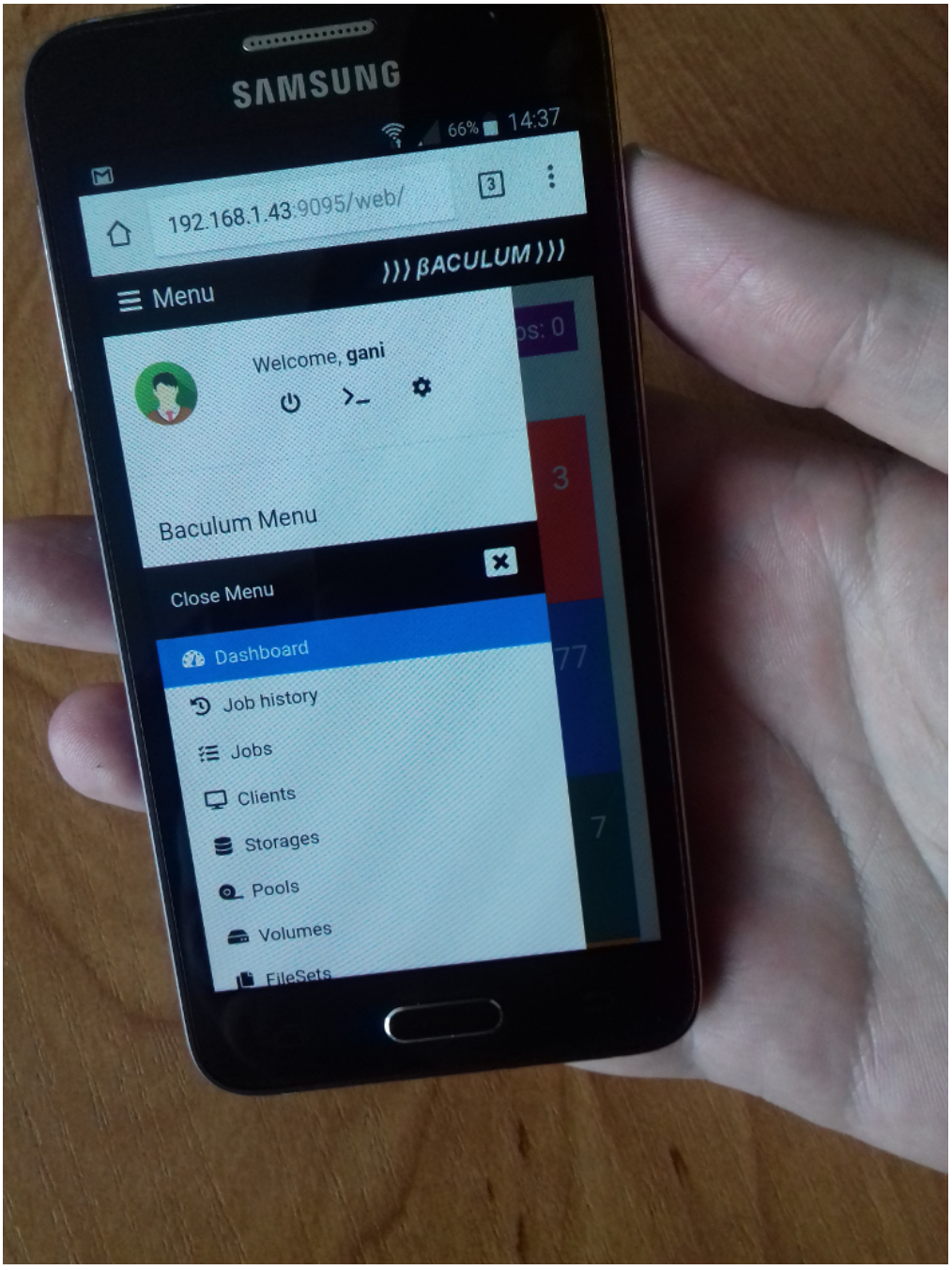


Figure 2.12: Menu on mobile phone

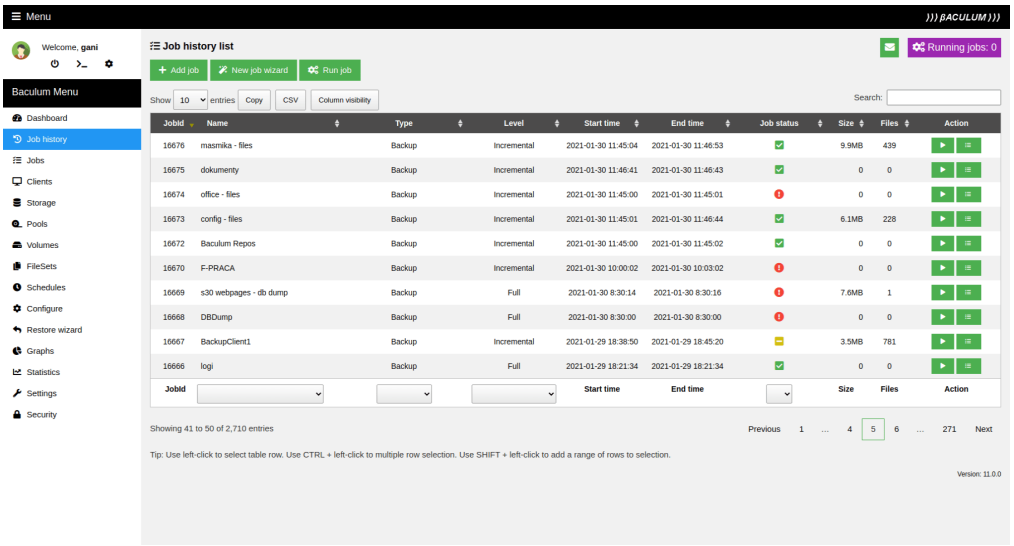


Figure 2.13: Job history list

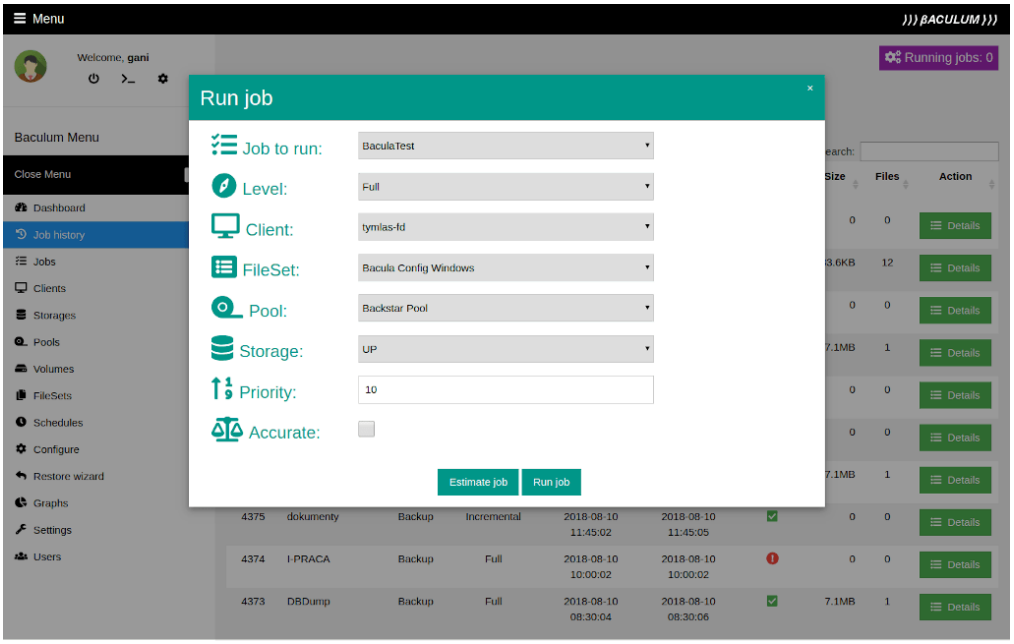


Figure 2.14: Run job window

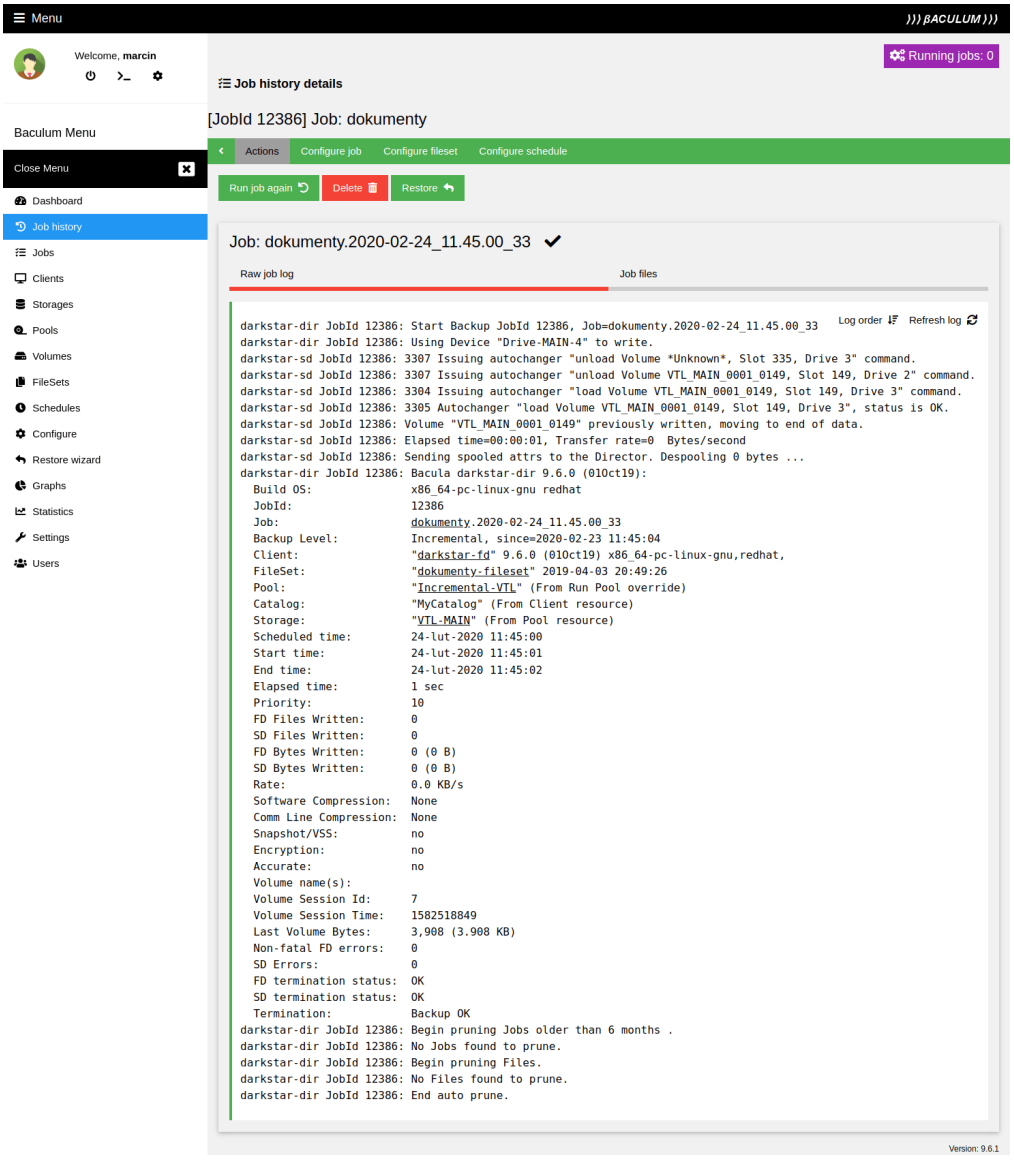


Figure 2.15: Job details

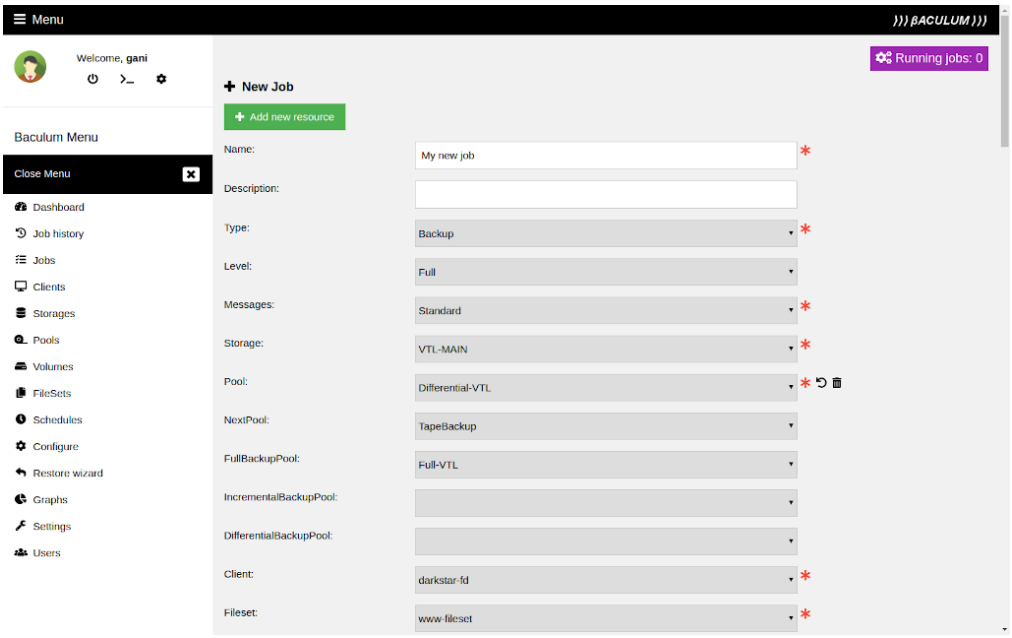


Figure 2.16: Create new job

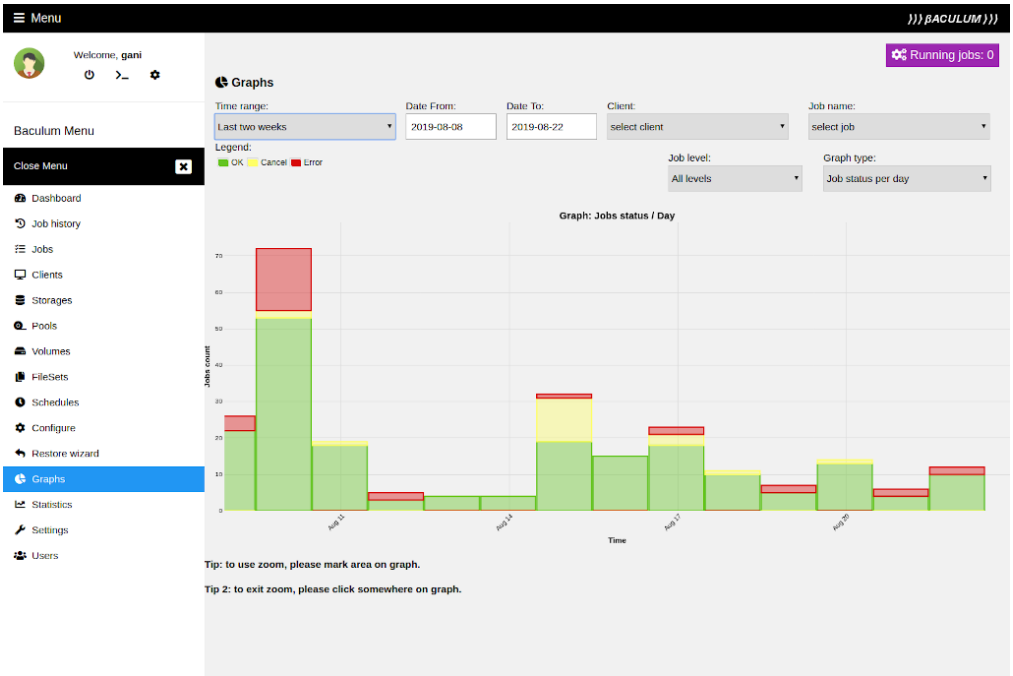


Figure 2.17: Job status graph

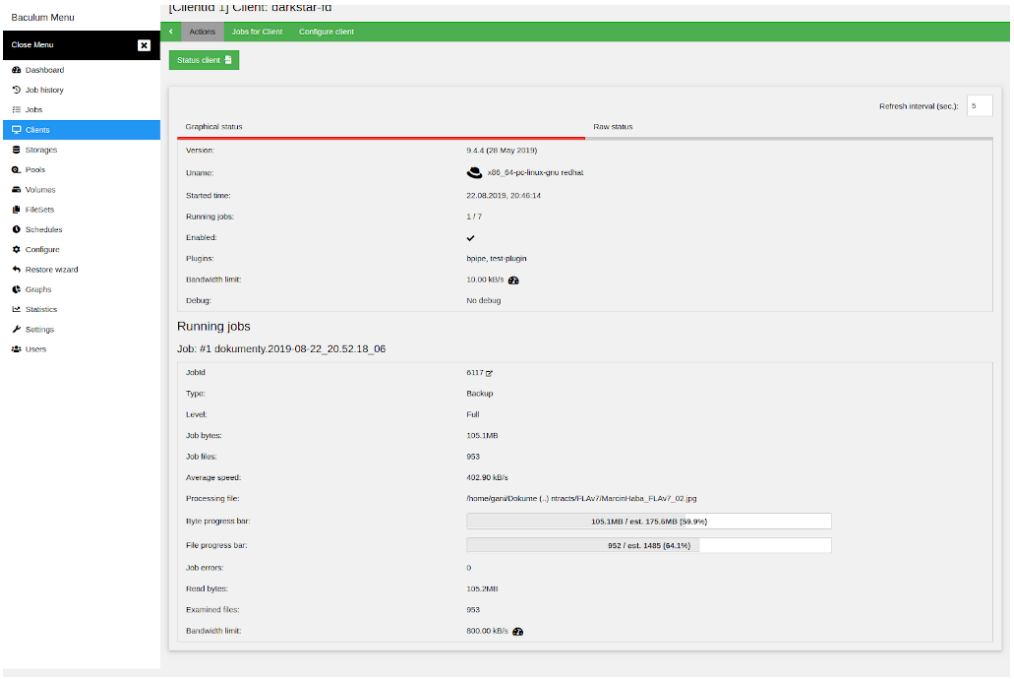


Figure 2.18: Client status

Appendices

Appendix A

Acronyms

BAT Bacula Administration Tool

Index

Symbols

A

B

C

D

E

G

H

K

L

M

P

Q

R

S

T

U

V

W