Disaster recovery (Backup)

Geethike Jayasinghe/ LEARN

Introduction to Backups

- Backup and recovery is essential. Failure to have verified backup and recovery procedures puts your data at risk of loss. Users often only learn this lesson after critical information they require is permanently lost. Attempting to recover from data loss can be both time consuming and extremely difficult. So learn from others mistakes, and ensure beforehand that you have a system in place that protects your data and suits your needs.

Some key points about Backups:

Before deciding on a backup and recovery strategy you have to ask the following questions:

1. **Why?** - Why are you protecting yourself against disaster? Does it matter if you lose data? What losses will you suffer ($$$)?
2. **What?** - What are you going to backup? Your entire hard drive or just some of the data?
3. **When? -** When is the best time to backup your system? How often will you perform a backup? When will you use full backups and incremental backups.
4. **Where? -** Where will the backups be stored? On-site? Off-Site? Cloud?
5. **Medium? -** Attached storage (usb stick, usb hard drive, tape drive), backup server? OSI layers).

Types of Backup:

There are many methods to provide backup and recovery; choosing the best process for you or your business will have to take several factors in to account.

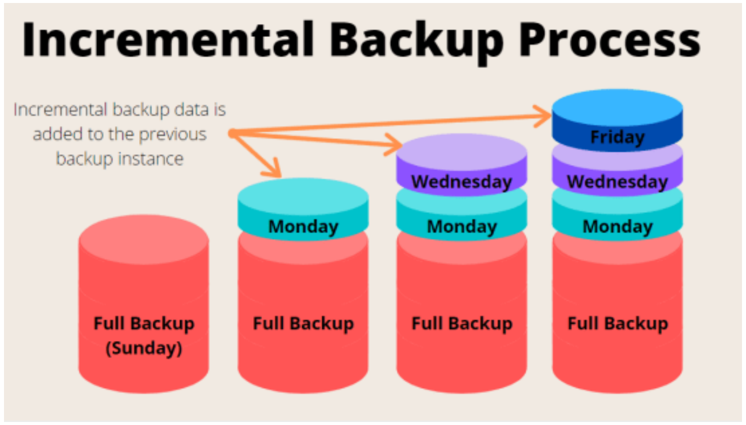
1. **Recovery time objective (RTO):** How fast should data be recovered? Can you continue to operate if data recovery is not recovered for a day, a week, etc?
2. **Recovery point objective (RPO):** How much data can be lost. Can you lose two hours, two days or two weeks of data?

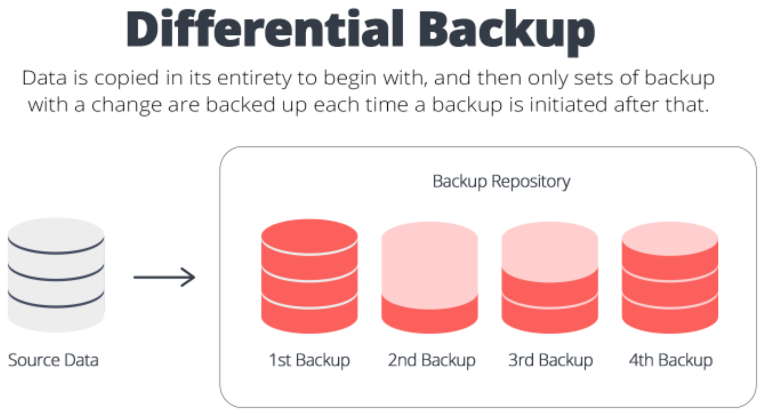
example:

if you can withstand losing one week of data then a weekly backup would be sufficient, but if you can only withstand losing one day then you would need to employ a nightly backup (or a variation)

Types of Backup:

1. **Full:** A full backup backs up all the files in the back up target.
2. **Incremental:** An incremental backup backs up all the files that have changed since the last backup.
3. **Differential:** A differential backup backs up all the files that have changed since the last full backup.




Differential Backup
Data is copied in its entirety to begin with, and then only sets of backup with a change are backed up each time a backup is initiated after that.
Backup Repository
Source Data
1st Backup
2nd Backup
3rd Backup
4th Backup

Backup Methods

Depending on your budget and specific RTO and RPO you can choose from manual, local automated, or remote automated.

1. **Manual -** Manual backup would be initiated on a schedule by the user and is the most common method for home users to backup their files. This method is also the least reliable.

2. **Local automated -** Automated backups that target a hard drive or tape drive attached to the physical box being backed up fall into this category. Advanced home users and small businesses will often use this method.

3. **Remote automated -** Automated backups that target a hard drive, tape drive or virtual tape library (VTL) over the network fall into this backup. This type of backup is often used by businesses that have money they can dedicate to the process of backup. As the organization becomes more mature they may even stage the backup on multiple mediums and increase the distance between backup and production systems.

mtime, atime and ctime

Ubuntu records three different times for each file:

mtime - modification time; this value is changed when the contents of the file is changed.

note: *file system backups change atime while raw device backups will not. If you are implementing incremental or differential backups this is important*

atime - access time; the value of this is changed when the file is accessed. The atime can also change when a backup utility or script has read the file as well as when a user has reads the file.

ctime - change time; the value is updated whenever the attributes of the file change. This can be ownership or permission.

Recovery

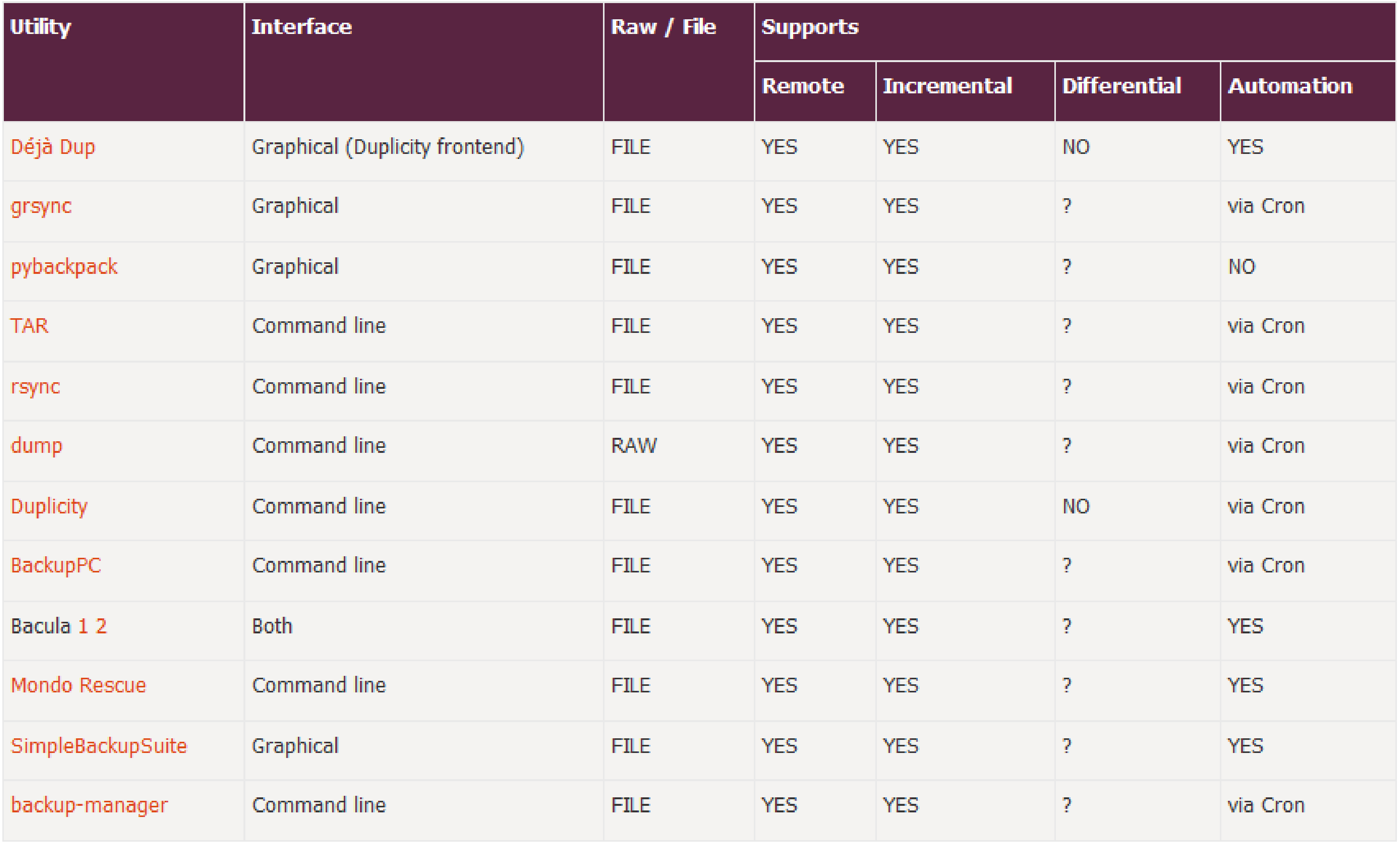
Note: *many people consider only the backup part of this process and do nothing to verify that the backup can be restored. It is very important to test that your backup process is working and that data can be recovered.*

It is crucial that your backups are tested by restoring them. Here are some tests you should do to ensure that you can recover from a disaster:

1. Restore many single files
2. Restore an older version of a file
3. Restore an entire folder
4. Restore an entire drive and compare the checksum

- If you do not test you may find out that nothing was being backed up when you need to restore the files in reality.

Backup Utilities



Utility	Interface	Raw / File	Supports			
			Remote	Incremental	Differential	Automation
Déjà Dup	Graphical (Duplicity frontend)	FILE	YES	YES	NO	YES
grsync	Graphical	FILE	YES	YES	?	via Cron
pybackpack	Graphical	FILE	YES	YES	?	NO
TAR	Command line	FILE	YES	YES	?	via Cron
rsync	Command line	FILE	YES	YES	?	via Cron
dump	Command line	RAW	YES	YES	?	via Cron
Duplicity	Command line	FILE	YES	YES	NO	via Cron
BackupPC	Command line	FILE	YES	YES	?	via Cron
Bacula 1 2	Both	FILE	YES	YES	?	YES
Mondo Rescue	Command line	FILE	YES	YES	?	YES
SimpleBackupSuite	Graphical	FILE	YES	YES	?	YES
backup-manager	Command line	FILE	YES	YES	?	via Cron

Bacula

Bacula is a backup program enabling you to backup, restore, and verify data across your network. There are Bacula clients for Linux, Windows, and Mac OS X - making it a cross-platform network wide solution.

https://www.bacula.org/

Bacula Overview

Bacula is made up of several components and services used to manage which files to backup and backup locations:

Bacula Director: a service that controls all backup, restore, verify, and archive operations.

Bacula Console: an application allowing communication with the Director. There are three versions of the Console:

- Text based command line version.
- Gnome based GTK+ Graphical User Interface (GUI) interface.
- wxWidgets GUI interface.

Bacula Overview

Bacula File: also known as the Bacula Client program. This application is installed on machines to be backed up, and is responsible for the data requested by the Director.

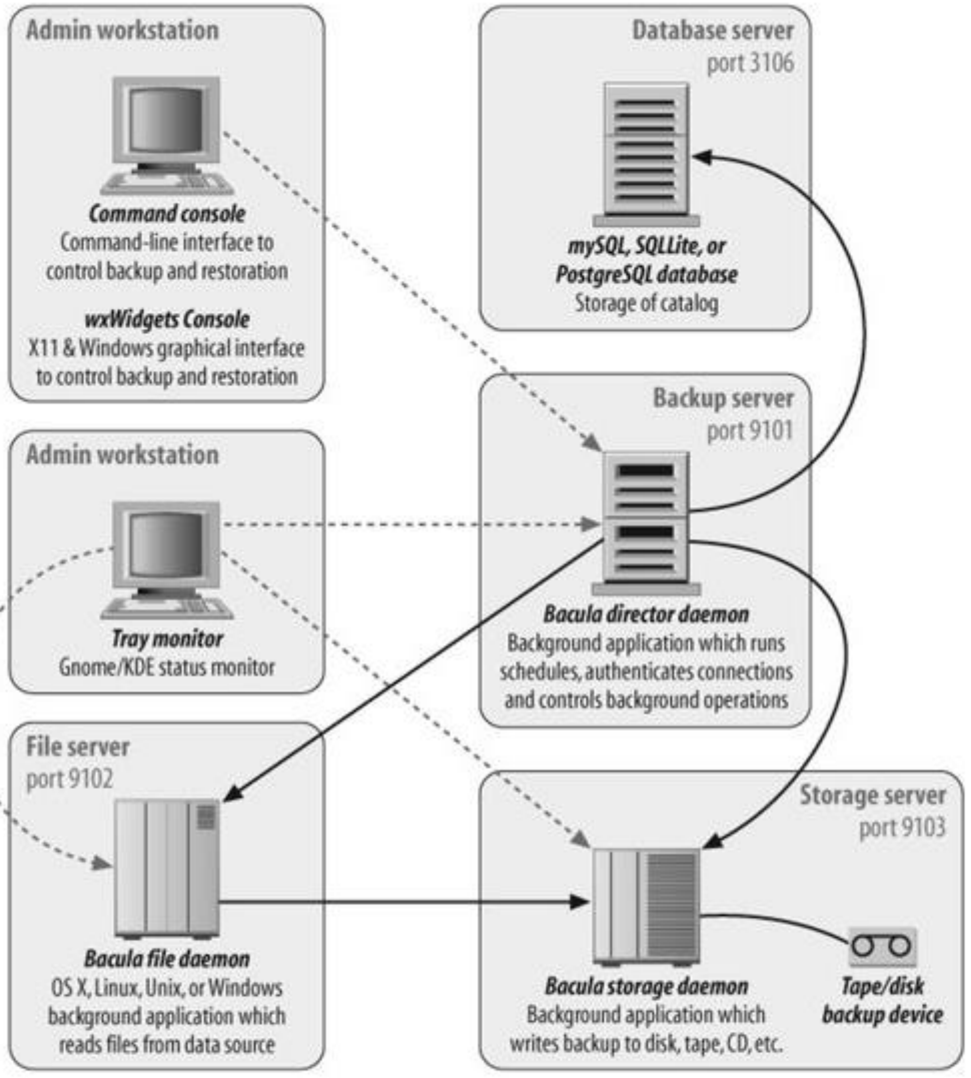
Bacula Storage: the programs that perform the storage and recovery of data to the physical media.

Bacula Catalog: is responsible for maintaining the file indexes and volume databases for all files backed up, enabling quick location and restoration of archived files. The Catalog supports three different databases MySQL, PostgreSQL, and SQLite.

Bacula Monitor: allows the monitoring of the Director, File daemons, and Storage daemons. Currently the Monitor is only available as a GTK+ GUI application.

These services and applications can be run on multiple servers and clients, or they can be installed on one machine if backing up a single disk or volume.

Bacula Overview


Admin workstation
Command console
Command-line interface to control backup and restoration
wxWidgets Console
X11 & Windows graphical interface to control backup and restoration
Database server
port 3106
mySQL, SQLLite, or PostgreSQL database
Storage of catalog
Admin workstation
Tray monitor
Gnome/KDE status monitor
Backup server
port 9101
Bacula director daemon
Background application which runs schedules, authenticates connections and controls background operations
File server
port 9102
Bacula file daemon
OS X, Linux, Unix, or Windows background application which reads files from data source
Storage server
port 9103
Bacula storage daemon
Background application which writes backup to disk, tape, CD, etc.
Tape/disk backup device

Bacula Overview

Director

The Bacula director is the application at the heart of a Bacula deployment: it manages media pool definition, scheduling, dependency tracking, access control, and reporting. It is responsible for nearly all policy-based configuration. Most changes to Bacula configuration occur in the director's configuration file.

Database server

Another essential piece of the Bacula design philosophy is the catalogan index of backed-up file storage locationswhich is kept in a relational database and accessed by SQL queries and updates. Currently, Bacula supports three databases: SQLite, MySQL, and PostgreSQL.

Storage daemon

The Bacula storage daemon manages interaction with media used to store backup data and is the only part of Bacula that actually communicates with volumes (such as tape, disk, or DVD-ROM) used for backups. All other Bacula components use the storage daemon's TCP/IP interface to communicate with the storage daemon and are unaware of the actual methods used to store and retrieve data. The storage daemon manages mounting and unmounting of storage volumes, labeling tapes (using OCR barcodes if available), and management of automatic tape loaders/libraries (ATLs).

Bacula Overview

File daemon

The file daemon actually transfers the client's data to the storage server. It must be installed on each machine that is to be backed up. This piece communicates with the director to determine what client data is to be backed up and which storage daemon is to be used, and then transmits the data directly to the selected storage daemon, including metadata about the files it is sending (such as file size, access control parameters, and ownership information) as well as file contents.

Bacula Overview

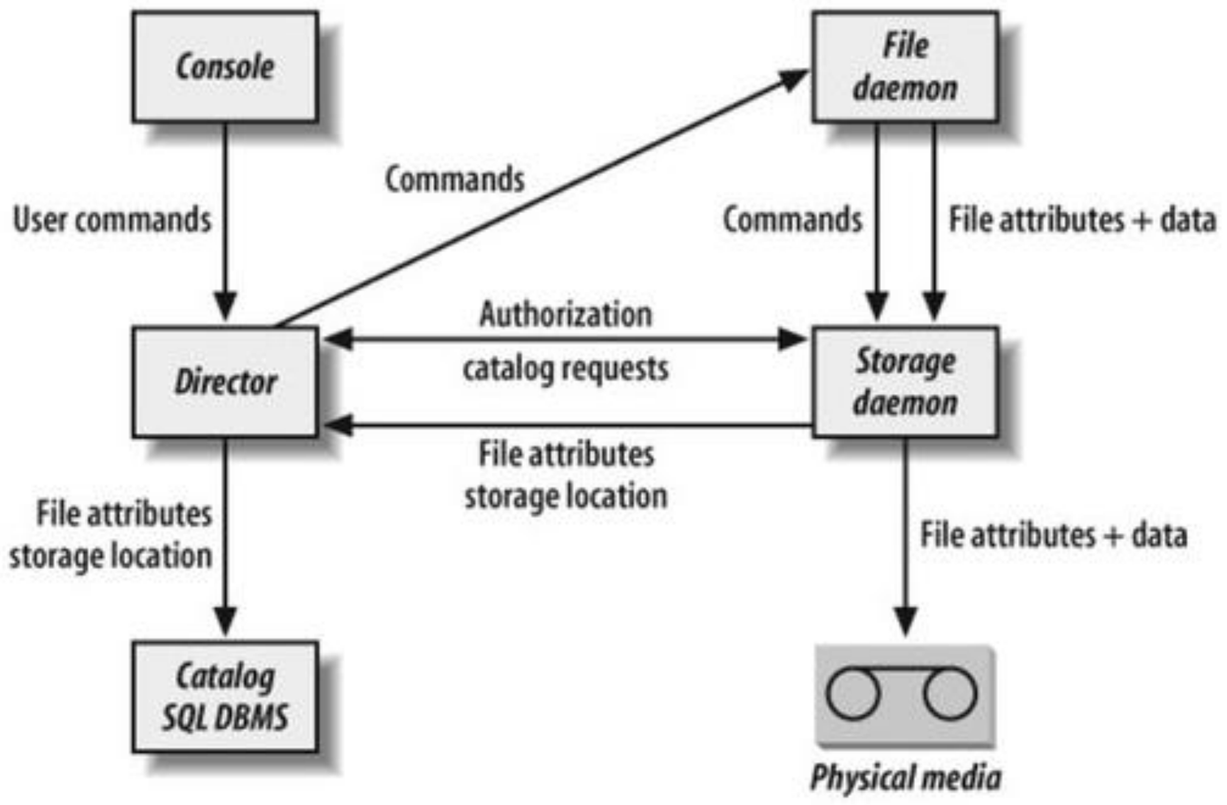
Administrative console

The administrative console provides the operator user interface for job management, message handling, and status information. Volume management operations (like manually labeling a new storage volume) are also handled from the console.

The console comes in multiple flavors. A line-mode console is available on all platforms, is the most well supported, and is the way most administrators choose to interact with Bacula. A GUI is available on systems with the wxWidgets libraries; the current GUI is a graphical wrapper over the line-mode interface, which makes interactive file restoration somewhat easier and provides tab-completion and instant interactive help for commands. The line-mode console is often preferred, largely because it is easy to operate over low-bandwidth connections, making remote administration easier. The Bacula console does not have to run on the same machine as the director.

Bacula Overview

Information flow in Bacula



bacula-dir.conf

```
Director {                            # define myself
  Name = what-dir
  DIRport = 9101                # where we listen for UA connections
  QueryFile = "/etc/bacula/query.sql"
  WorkingDirectory = "/var/bacula"
  PidDirectory = "/var/run"
  Maximum Concurrent Jobs = 1
  Password = "BCONSOLE PASSWORD"         # Console password
  Messages = Daemon
}
```

Job Definitions

Job definitions let you set up common configuration data for many jobs at once.

```
JobDefs {
  Name = "DefaultJob"
  Type = Backup
  Level = Incremental
  Schedule = "MonthlyCycle"
  Storage = LTO-1
  Messages = Standard
  Pool = Monthly
  Priority = 20
}

JobDefs {
  Name = "DefaultGenie"
  Type = Backup
  Level = Incremental
  Schedule = "MonthlyCycle"
  Storage = LTO-1
  Messages = Standard
  Pool = Genie
  Priority = 10
}
```

Create a backup job:

A Backup job, which must have a unique name, defines the details of which Client and which data should be backed up.

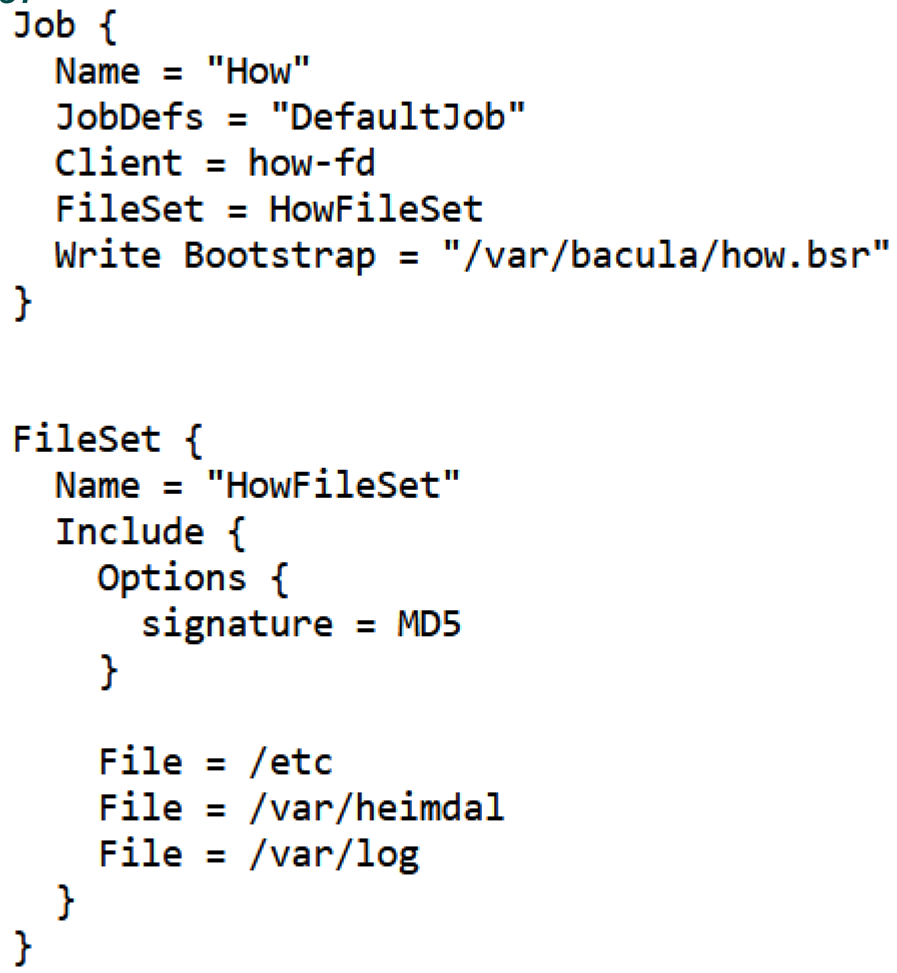
```
Job {
  Name = "What"
  JobDefs = "DefaultJob"
  Client = what-fd
  FileSet = WhatFileSet
  Write Bootstrap = "/var/bacula/what.bsr"
}

FileSet {
  Name = "WhatFileSet"
  Include {
    Options {
      signature = MD5
    }

    File = /etc
    File = /var/bacula
  }
}
```

Create a backup job:

A Backup job, which must have a unique name, defines the details of which Client and which data should be backed up.



```
Job {
  Name = "How"
  JobDefs = "DefaultJob"
  Client = how-fd
  FileSet = HowFileSet
  Write Bootstrap = "/var/bacula/how.bsr"
}


FileSet {
  Name = "HowFileSet"
  Include {
    Options {
      signature = MD5
    }

    File = /etc
    File = /var/heimdal
    File = /var/log
  }
}
```

Schedule backup job:

```
#
# When to do the backups, full backup on first sunday of the month,
#  differential (i.e. incremental since full) every other sunday,
#  and incremental backups other days


# This schedule does a monthly full, and then incrementals
Schedule {
  Name = "MonthlyCycle"
  # Run Full backup on first day of each month
  Run = Level=Full on 1 at 1:05
  # Run incremental on other days
  Run = Level=Incremental on 2-31 at 1:05
  # Run Archive full backup on 1st day of each quarter.
  Run = Level=Full Pool=Archive on jan 1 at 1:00
  Run = Level=Full Pool=Archive on apr 1 at 1:00
  Run = Level=Full Pool=Archive on jul 1 at 1:00
  Run = Level=Full Pool=Archive on oct 1 at 1:00
}
```

Adding client definitions:

```
# Client (File Services) to backup
Client {
  Name = what-fd
  Address = what.isds.duke.edu
  FDPort = 9102
  Catalog = MyCatalog
  Password = "WHAT BACULA-FD PASSWORD"          # password for FileDaemon
  File Retention = 100 days
  Job Retention = 6 months
  AutoPrune = yes                     # Prune expired Jobs/Files
}

# which client definition
Client {
  Name = which-fd
  Address = which.isds.duke.edu
  FDPort = 9102
  Catalog = MyCatalog
  Password = "WHICH BACULA-FD PASSWORD"
  File Retention = 100 days
  Job Retention = 6 months
  AutoPrune = yes                     # Prune expired Jobs/Files
}
```

Adding pool definitions:

```
Pool {
  Name = Genie
  Pool Type = Backup
  Recycle = yes                       # Bacula can automatically recycle Volumes
  Recycle Oldest Volume = yes           # Recycle oldest Volumes first
  AutoPrune = yes                     # Prune expired volumes
  Volume Retention = 128 days         # Approx 4 months
  Accept Any Volume = yes             # write on any volume in the pool
  Volume Use Duration = 30 days         # Allow writing only for 30 days
  Label Format = Genie-
}
```

Adding Storage definitions:

```
Storage {
  Name = LTO-1
# Do not use "localhost" here
  Address = what.isds.duke.edu                # N.B. Use a fully qualified name here
  SDPort = 9103
  Password = "LTO-1 STORAGE DAEMON ON WHAT PASSWORD"
  Device = LTO-1
  Media Type = LTO-1
  Autochanger = yes
}
```

Adding Restore definitions:

```
# Standard Restore template, to be changed by Console program
Job {
  Name = "RestoreFiles"
  Type = Restore
  Client=what-fd
  FileSet="WhatFileSet"
  Storage = LTO-1
  Pool = Monthly
  Messages = Standard
  Where = /tmp/bacula-restores
}
```

Bacula Storage Daemon Configuration file:

```
Storage {                             # definition of myself
  Name = what-sd
  SDPort = 9103                  # Director's port
  WorkingDirectory = "/var/bacula"
  Pid Directory = "/var/run"
  Maximum Concurrent Jobs = 20
}

#
# List Directors who are permitted to contact Storage daemon
#
Director {
  Name = what-dir
  Password = "LTO-1 STORAGE DAEMON ON WHAT PASSWORD"
}
```

Bacula Storage Daemon Configuration file:

```
#
# Devices supported by this Storage daemon
# To connect, the Director's bacula-dir.conf must have the
#  same Name and MediaType.
#

Device {
  Name = FileStorage
  Media Type = File
  Archive Device = /tmp/bacula
  Label media = yes;                    # lets Bacula label unlabeled media
  Random Access = Yes;
  AutomaticMount = yes;                # when device opened, read it
  RemovableMedia = no;
  AlwaysOpen = no;
}
```

Bacula File Daemon Configuration file:

```
#
# List Directors who are permitted to contact this File daemon
#
Director {
  Name = what-dir
  Password = "WHAT BACULA-FD PASSWORD"
}

#
# "Global" File daemon configuration specifications
#
FileDaemon {                         # this is me
  Name = what-fd
  FDport = 9102                 # where we listen for the director
  WorkingDirectory = /var/bacula
  Pid Directory = /var/run
  Maximum Concurrent Jobs = 20
}
```

Thank You

Geethike Jayasinghe/LEARN

Email: geethike@learn.ac.lk