



Network Backup with Bacula How-To

Create an autonomous backup solution for your
computer network using practical, hands-on recipes

Eugene Pankov

[PACKT]
PUBLISHING



Network Backup with Bacula How-To

Create an autonomous backup solution for your computer network using practical, hands-on recipes

Eugene Pankov



BIRMINGHAM - MUMBAI

Network Backup with Bacula How-To

Copyright © 2012 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: October 2012

Production Reference: 1191012

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84951-984-7

www.packtpub.com

Credits

Author

Eugene Pankov

Project Coordinator

Shraddha Bagadia

Reviewer

Wassim C. Zaarour

Proofreader

Aaron Nash

Acquisition Editor

Mary Nadar

Production Coordinator

Prachali Bhiwandkar

Commissioning Editor

Priyanka Shah

Cover Work

Prachali Bhiwandkar

Technical Editor

Prashant Salvi

Cover Image

Conidon Miranda

About the Author

Eugene Pankov is a Lead Software Architect and Developer at Linology Networks LLC. For over a decade, he has been providing software design and development services, as well as Linux server setup and maintenance for a large variety of companies, from web hosting to sports telemetry. Eugene is the founder and main developer of the Ajenti open source web server administration panel, which is used by nearly 10,000 people worldwide at the time of writing this book. He has successfully completed a large variety of projects, from Django websites to Android games.

About the Reviewer

Wassim Zaarour, is a Systems and Network Engineer with 7 years of experience in the ICT field, and has worked on various technologies and systems. Wassim works for a leading company in the managed services and ICT department of the EMEA region.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- ▶ Fully searchable across every book published by Packt
- ▶ Copy and paste, print and bookmark content
- ▶ On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	1
Network Backup with Bacula How-To	5
Getting started with Bacula (Must know)	5
Simple directory backup (Must know)	9
Scheduling backups (Must know)	13
Backing up MySQL server (Should know)	14
Backing up PostgreSQL server (Become an expert)	18
Backing up OpenLDAP server (Become an expert)	20
Backing up mailboxes (Should know)	21
Backing up Zarafa (Become an expert)	23
Backups behind a firewall (Become an expert)	26
Backing up a Windows client (Should know)	30
Multiple storage servers (Should know)	33
Setting up from scratch	36
Appendix: Mini tips	42

Preface

Network Backup with Bacula How-To is a practical guide to setting up the Bacula backup system using a number of sequential recipes, ranging from the simplest to the most intricate ones, which will give you the knowledge needed to build a self-contained backup infrastructure. This book contains instructions on the setup and configuration of various Bacula components as well as information to help you painlessly integrate your Windows and Linux workstations into the backup system. The included recipes teach ways to back up machines running different operating systems and server software and to create multi-server backup storage configurations. With Network Backup with Bacula How-To, you will learn everything you need to create an autonomous backup solution for your computer network.

What this book covers

Getting started with Bacula (Must know), explains how to set up the simplest Bacula Director and Bacula Storage installations, and how the bconsole utility can be used to monitor and interact with daemons.

Simple directory backup (Must know), explains how to set up a simple backup of a single directory from a Unix machine to the previously configured Bacula Director.

Scheduling backups (Must know), teaches you how to employ different scheduling strategies for your backups.

Backing up MySQL server (Should know), explains how to perform an online backup of MyISAM tables and write-locked backup of InnoDB tables from the MySQL database server.

Backing up PostgreSQL server (Become an expert), explains how to back up databases from a PostgreSQL server installation.

Backing up OpenLDAP server (Become an expert), explains how to perform a backup of the OpenLDAP Active Directory server.

Backing up mailboxes (Should know), explains how to back up Maildir and mbox format mailboxes from Postfix and other mail servers.

Backing up Zarafa (Become an expert), explains how to back up using the Zarafa groupware servers.

Backups behind a firewall (Become an expert), explains how to set up backups when the Director and client are separated by a firewall.

Backing up a Windows client (Should know), explains how to back up a Windows client.

Multiple storage servers (Should know), explains how to set up multiple storage servers for different backup scenarios.

Setting up from scratch, provides an example to perform a backup from scratch.

Appendix: Mini tips, provides some tips that will be useful while performing backups.

What you need for this book

A Linux-powered machine. We suggest that you use a virtual machine similar to VirtualBox, so you will be free to experiment and not afraid to break things.

Who this book is for

Junior network administrators familiar with Linux or BSD, who are looking for a simple and consistent network backup solution. It's assumed that you will be able to set up and configure a Linux or BSD server, and are familiar with local network layouts and routing.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "We can include other contexts through the use of the `include` directive."

A block of code is set as follows:

```
[default]
exten => s,1,Dial(Zap/1|30)
exten => s,2,VoiceMail(u100)
exten => s,102,VoiceMail(b100)
exten => i,1,VoiceMail(s0)
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
[default]
exten => s,1,Dial(Zap/1|30)
exten => s,2,VoiceMail(u100)
exten => s,102,VoiceMail(b100)
exten => i,1,VoiceMail(s0)
```

Any command-line input or output is written as follows:

```
# cp /usr/src/asterisk-addons/configs/cdr_mysql.conf.sample
   /etc/asterisk/cdr_mysql.conf
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "clicking the **Next** button moves you to the next screen".



Warnings or important notes appear in a box like this.



Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title via the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on www.packtpub.com or e-mail suggest@packtpub.com.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/support>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

Network Backup with Bacula How-To

Welcome to Network Backup with Bacula How-To. This book contains instructions on the setup and configuration of various Bacula components as well as information to help you painlessly integrate your Windows and Linux workstations into the backup system. You will learn how to set up Bacula Director and Storage, how to back up Linux and Windows clients, ways to overcome limitations implied by firewall protection, specific cases of backup for MySQL, OpenLDAP, Postfix, and many others, as well as different scheduling strategies and various other tips.

Getting started with Bacula (Must know)

This recipe explains how to set up the simplest Bacula Director and Bacula Storage installations and how the `bconsole` utility can be used to monitor and interact with daemons.

Getting ready

You will need root access to a BSD or Linux machine where you will be installing the Bacula server. This recipe is using Debian Linux 6.0 as an example. The machine is assumed to have an IP address of 10.10.1.100.

How to do it...

To install Bacula Director and Bacula Storage, perform the following steps:

1. Log in to the target machine using SSH client or a physical console.
2. Install the Bacula Director and Bacula Storage daemons plus the `bconsole` utility, using the package manager of your choice as follows (this example uses APT):

```
$ apt-get install bacula-director-sqlite bacula-sd-sqlite bacula-console
```

3. Now, replace the contents of the `/etc/bacula/bacula-dir.conf` file with the following code:

```
Director {
    Name = debian-dir
    DIRport = 9101
    QueryFile = "/etc/bacula/scripts/query.sql"
    WorkingDirectory = "/var/lib/bacula"
    PidDirectory = "/var/run/bacula"
    Maximum Concurrent Jobs = 1
    Password = "password-dir"
    DirAddress = 10.10.1.100
}

Storage {
    Name = LocalStorage
    Address = 10.10.1.100
    SDPort = 9103
    Password = "password-sd"
    Device = FileStorage
    Media Type = File
}

Job {
    Name = DefaultJob
    Type = Backup
    Messages = Standard
    Pool = Default
    Client = DefaultClient
    Fileset = DefaultFileset
    Storage = LocalStorage
}

Client {
    Name = DefaultClient
```

```
Address = 10.10.1.100
Password = ""
Catalog = DefaultCatalog
}

Fileset {
    Name = DefaultFileset
}

Catalog {
    Name = DefaultCatalog
    dbname = "bacula"; dbuser = ""; dbpassword = ""
}
```

The highlighted lines contain information you might want to alter, such as passwords and network addresses.

4. Replace the `bacula-sd.conf` file with a new one that has the following code:

```
Storage {
    Name = LocalStorage
    SDPort = 9103
    WorkingDirectory = "/var/lib/bacula"
    Pid Directory = "/var/run/bacula"
    Maximum Concurrent Jobs = 20
    SDAddress = 10.10.1.100
}

Director {
    Name = debian-dir
    Password = "password-sd"
}

Device {
    Name = FileStorage
    Media Type = File
    Archive Device = /tmp/test-backups
    LabelMedia = yes;
    Random Access = Yes;
    AutomaticMount = yes;
    RemovableMedia = no;
    AlwaysOpen = no;
}

Messages {
    Name = Standard
    director = debian-dir = all
}
```


5. And finally, the `bconsole.conf` file would look as follows:

```
Director {  
    Name = localhost-dir  
    DIRport = 9101  
    address = 10.10.1.100  
    Password = "password-dir"  
}
```

Note that appropriate passwords should match in both files.

6. Now restart the daemons as follows so the new configuration is applied:

```
$ /etc/init.d/bacula-director restart  
$ /etc/init.d/bacula-sd restart
```

7. Create a directory for backup storage mentioned in the `bacula-sd.conf` file as follows:

```
$ mkdir /tmp/test-backups
```

Now you should be able to use the `bconsole` utility to inspect the status of Director and Storage using the `status director` and `status storage` commands as follows:

```
$ bconsole  
Connecting to Director 10.10.1.100:9101  
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)  
Enter a period to cancel a command.  
*status director  
debian-dir Version: 5.0.2 (28 April 2010) i486-pc-linux-gnu debian  
squeeze/sid  
Daemon started 07-Jun-12 13:54, 0 Jobs run since started.  
  Heap: heap=245,760 smbytes=48,038 max_bytes=48,940 bufs=109 max_  
bufs=119  
No Scheduled Jobs.  
====  
Running Jobs:  
No Jobs running.  
====  
*status storage  
Automatically selected Storage: LocalStorage  
Connecting to Storage daemon LocalStorage at 10.10.1.100:9103  
  
LocalStorage Version: 5.0.2
```

```
...
Running Jobs:
No Jobs running.
Device status:
Device "FileStorage" (/tmp/test-backups) is not open.
*
```

How it works...

A typical Bacula system consists of three independently running daemons:

- ▶ **Storage daemon (SD):** This daemon receives backups from Director and stores them into the storage (files or tapes)
- ▶ **File daemon (FD):** This daemon collects the files from client machines and sends them to the Director
- ▶ **Director daemon:** This daemon performs job scheduling and negotiates data transfers between Storage and File daemons

In this setup, we installed Director and Storage daemons on the same machine and configured a file-based storage at `/tmp/test-backups` in our filesystem.

The `bconsole` utility is used to connect to the Director daemon, pass commands to it, and receive messages from it. When we requested the status of Storage with the `status storage` command, Director forwarded the request to Storage. Therefore, it's only important to have a direct connection from console to Director.



Replace the sample passwords of Director, Storage and the `bconsole` utility. Each component can reside on a separate machine as long as connectivity between them is possible. Don't forget to always use fully qualified domain names or non-local IP addresses when specifying remote components.

Simple directory backup (Must know)

This recipe explains how to set up a simple backup of a single directory from a UNIX machine to the previously configured Bacula Director.

Getting ready

You will need a Bacula Director machine configured as per the *Getting started with Bacula* recipe, and a client machine that will be backed up. In this recipe we assume the client machine to have an IP address as 10.10.1.101.

How to do it...

To back up a single directory, perform the following steps:

1. Log in to the client machine and install the Bacula file daemon as follows:
2. Now replace the contents of the `/etc/bacula/bacula-fd.conf` file with the following configuration:

```
Director {
    Name = debian-dir
    Password = "password-fd"
}

FileDaemon {
    Name = debian-fd
    FDport = 9102
    WorkingDirectory = /var/lib/bacula
    Pid Directory = /var/run/bacula
    Maximum Concurrent Jobs = 20
    FDAddress = 10.10.1.101
}

Messages {
    Name = Standard
    director = debian-dir = all, !skipped, !restored
}
```

3. Restart the daemon as follows to apply updated configuration:

```
$ /etc/init.d/bacula-fd restart
```

Log in to the Director machine and add the following changes to `/etc/bacula/bacula-dir.conf`:

1. Replace the fake client with an actual definition as follows:

```
Client {
    Name = DefaultClient
    Address = 10.10.1.101
    Password = "password-fd"
    Catalog = DefaultCatalog
}
```

Note that passwords in the Director and Client configurations should match.

2. Replace the empty FileSet with a specification of the /home/user directory as follows:

```
Fileset {  
    Name = DefaultFileset  
    Include {  
        Options {  
            signature = MD5  
        }  
        File = /home/user  
    }  
}
```

3. Set the DefaultJob to be an incremental backup job as follows:

```
Job {  
    Name = DefaultJob  
    Type = Backup  
    Level = Incremental  
    Messages = Standard  
    Pool = Default  
    Client = DefaultClient  
    Fileset = DefaultFileset  
    Storage = LocalStorage  
}
```

4. Restart the daemon as follows to apply updated configuration:

```
$ /etc/init.d/bacula-director restart
```

5. Run bconsole and issue a command to create a labeled Volume for backup storage, using the label command as follows:

```
$ bconsole  
Connecting to Director 10.10.1.100:9101  
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)  
Enter a period to cancel a command.  
*label  
Automatically selected Catalog: DefaultCatalog  
...  
Enter new Volume name: Test  
...  
Select the Pool (1-3): 1  
Sending label command for Volume "Test" Slot 0 ...  
3000 OK label.VolBytes=191 DVD=0 Volume="Test"  
Device="FileStorage" (/tmp/test-backups)  
Catalog record for Volume "Test", Slot 0 successfully created.
```

6. Now that the all components are configured and a volume is created, we can start the backup job using the `run` command as follows:

```
$ bconsole
Connecting to Director 10.10.1.100:9101
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)
Enter a period to cancel a command.
*run DefaultJob
Automatically selected Catalog: DefaultCatalog
Using Catalog "DefaultCatalog"
Run Backup job
JobName:   DefaultJob
Level:     Incremental
Client:    DefaultClient
FileSet:   DefaultFileset
Pool:      Default (From Job resource)
Storage:   LocalStorage (From Job resource)
When:      2012-06-08 08:05:50
Priority: 10
OK to run? (yes/mod/no): yes
Job queued. JobId=4
You have messages.
*messages
...
08-Jun 08:05 debian-dirJobId 4: Start Backup JobId 4, Job=DefaultJ
ob.2012-06-08                                _08.05.52_10
08-Jun 08:05 debian-dirJobId 4: Using Device "FileStorage"
...
Last Volume Bytes:      5,112 (5.112 KB)
Non-fatal FD errors:    0
SD Errors:              0
FD termination status:  OK
SD termination status:  OK
Termination:           Backup OK
```

The termination status indicates that the job was completed successfully.

How it works...

The backup process is regulated by the Director daemon. First, the File (client) daemon on the client machine is notified of the job's start. Then, the File daemon connects to the Storage and sends the compressed backup data. The transfer is monitored by Director. Once the Storage confirms that backup has been stored, the job is considered finished.

Backups are organized into volumes (which may be files or tapes depending on the type of storage). In this particular setup, the backups are stored into `Test` volume located in the `/tmp/test-backups/Test` file.

The Storage daemon, which receives backups from Director, stores them into the storage (files or tapes).

There's more...

There are a number of handy `bconsole` commands available, such as the following:

- ▶ `messages`: This command lists pending messages sent to you by Director.
- ▶ `cancel`: This command will cancel a running job immediately.
- ▶ `list` and `show`: These commands can be used to inspect various Bacula objects. For example, the `list jobs` command will list the latest submitted jobs and the `show job 1` command will display the status of the job 1

Scheduling backups (Must know)

In this recipe, you'll learn to employ different scheduling strategies for your backups.

Getting ready

You will need a Bacula Director machine with a configured client and backup job.

How to do it...

To schedule a backup, perform the following steps:

1. Log in to the Director machine.
2. Create a `Schedule` object in the `/etc/bacula/bacula-dir.conf` file as follows:

```
Schedule {  
    Name = "Weekly"  
    Run = sun at 3:00  
}
```

3. Locate the `Job` definition and link it with the newly added `Schedule` as follows:

```
Job {  
    ...  
    Schedule = "Weekly"  
}
```

4. Restart the daemon to apply the updated configuration:

```
$ /etc/init.d/bacula-director restart
```

Now, the job will be automatically started at 3:00 every Sunday.

There's more...

There are lots of additional scheduling keywords available. Some examples are as follows:

- ▶ `on 1 at 2:00`: These scheduling keywords run the scheduled backups every 1st day of the month at given time
- ▶ `hourly at 0:15`: These scheduling keywords run the backup after 15 minutes every hour
- ▶ All keywords may be ranged, for example, `mon-sat at 14:00`
- ▶ Other keywords include: `w00`, `w01` (weeks of the year), `1st sun`, `4th mon` (numbered day-of-week each month), `daily`, `weekly`, `monthly`
- ▶ Some job parameters can be overridden in the `Run` directive as follows:

```
Run = Level=Incremental Pool=Default mon-fri at 1:00
```
- ▶ `Schedule` may contain more than one `Run` directive; they will all be executed at the respective time:

```
Schedule {  
    Name = "WeeklyCycle"  
    Run = Level=Full sun at 5:00  
    Run = Level=Incremental mon-sat at 5:00  
}
```

Backing up MySQL server (Should know)

This recipe explains how to perform an online backup of MyISAM tables and a write-locked backup of InnoDB tables from the MySQL database server.

Getting ready

You will need a Bacula Director machine with a configured client and the Bacula File daemon running on the machine containing MySQL server.

How to do it...

To make an online back up of MyISAM tables, perform the following steps:

1. Log in to the client machine.
2. Create a dedicated MySQL user for Bacula with access and lock permissions on all required databases and tables as follows:

```
$ mysql -p
Enter password:
...
mysql> CREATE USER 'bacula'@'localhost' IDENTIFIED BY 'bacula-
password';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT SELECT, RELOAD, LOCK TABLES ON *.* TO
'bacula'@'localhost' IDENTIFIED BY 'bacula-password';
Query OK, 0 rows affected (0.00 sec)
```

3. Create the following script in the `/usr/local/bin/mysqlhotcopyall` file, which will dump the databases for backup:

```
#!/bin/bash
DBLIST="test mysql" # databases for backing up
DBDIR=/var/lib/bacula/mysql
UP=" --user=$1 --password=$2"
LOGFILE=/var/log/backup.log
mkdir $DBDIR
for DATABASE in $DBLIST
do
mysqlhotcopy $UP $DATABASE ${DBDIR} --allowold >> ${LOGFILE}
done
```

4. Make the script executable by executing the following command:
5. Now on the Director machine, create a new Job and FileSet as follows:

```
Job {
    Name = MySQLJob
    Type = Backup
    Level = Incremental
    Messages = Standard
    Pool = Default
    Client = DefaultClient
```



```
Fileset = MySQLFileset
Storage = LocalStorage
ClientRunBeforeJob = "/usr/local/bin/mysqlhotcopyall bacula
    bacula-password"
ClientRunAfterJob = "/bin/rm -rf /var/lib/bacula/mysql"
}

Fileset {
    Name = MySQLFileset
    Include {
        Options {
            signature = MD5
        }
        File = /var/lib/bacula/mysql
    }
}
```

Make sure to use the right password for bacula MySQL user here.

6. Restart the daemon as follows to apply the updated configuration:
`$ /etc/init.d/bacula-director restart`
7. Now test the backup using `bconsole` as follows:

```
$ bconsole
Connecting to Director 10.10.1.100:9101
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)
Enter a period to cancel a command.
*run MySQLJob yes
Job queued. JobId=9
*messages
...
Termination:          Backup OK
```

The output of the `bconsole` command is shown in the following screenshot:

```

Machine View Devices Help
SD Files Written:      0
FD Bytes Written:      0 (0 B)
SD Bytes Written:      0 (0 B)
Rate:                  0.0 KB/s
Software Compression:  None
USS:                   no
Encryption:            no
Accurate:              no
Volume name(s):
Volume Session Id:     5
Volume Session Time:   1346225778
Last Volume Bytes:     6,018 (6.018 KB)
Non-fatal FD errors:   0
SD Errors:             0
FD termination status: OK
SD termination status: OK
Termination:          Backup OK

29-Aug 03:50 debian-dir JobId 6: Begin pruning Jobs older than 6 months .
29-Aug 03:50 debian-dir JobId 6: No Jobs found to prune.
29-Aug 03:50 debian-dir JobId 6: Begin pruning Jobs.
29-Aug 03:50 debian-dir JobId 6: No Files found to prune.
29-Aug 03:50 debian-dir JobId 6: End auto prune.

*_

```

How it works...

We're using the `ClientRunBeforeJob` and `ClientRunAfterJob` directives to run a script that creates a snapshot of the databases to the `/var/lib/bacula/mysql` folder. Then, this folder is backed up. The `mysqlhotcopy` command will perform a write lock on the tables, so during the backup, database will operate in read-only mode.

There's more...

If any of your tables run on InnoDB engine, change the backup script to use `mysqldump` instead of `mysqlhotcopy` as follows:

```

#!/bin/bash
DBLIST="test mysql" # databases for backing up
DBDIR=/var/lib/bacula/mysql
UP=" -u $1 --password=$2"
LOGFILE=/var/log/backup.log
mkdir $DBDIR
for DATABASE in $DBLIST
do
    mysqldump $UP $DATABASE >> ${DBDIR}/${DATABASE}
done

```

Backing up PostgreSQL server (Become an expert)

This recipe explains how to back up databases from a PostgreSQL server installation.

Getting ready

You will need a Bacula Director machine with a configured client and the Bacula File daemon running on the machine containing PostgreSQL server.

How to do it...

To back up a database from a PostgreSQL server installation, perform the following steps:

1. Log in to the client machine.
2. Create the following script in the `/usr/local/bin/pgsqlcopyall` file:

```
#!/bin/bash
pg_dumpall -U postgres > /var/lib/pgsql-dump/data.sql
```
3. If your PostgreSQL security policy prevents password-less local logon, adjust the `pg_dumpall` call to include the password.
4. Make the script executable and create a folder for storing dump:

```
$ chmod +x /usr/local/bin/pgsqlcopyall
$ mkdir /var/lib/pgsql-dump
$ chown postgres: /var/lib/pgsql-dump
```
5. Now on the Director machine, create a new Job and FileSet as follows:

```
Job {
    Name = PgSQLJob
    Type = Backup
    Level = Incremental
    Messages = Standard
    Pool = Default
    Client = DefaultClient
    Fileset = PgSQLFileset
    Storage = LocalStorage
    ClientRunBeforeJob = "su - postgres -c /usr/local/bin/
        pgsqlcopyall"
    ClientRunAfterJob = "/bin/rm -rf /var/lib/pgsql-dump/*"
}

Fileset {
    Name = PgSQLFileset
```

```

Include {
  Options {
    signature = MD5
  }
  File = /var/lib/pgsql-dump
}
}

```

6. Restart the daemon as follows to apply the updated configuration:

```
$ /etc/init.d/bacula-director restart
```

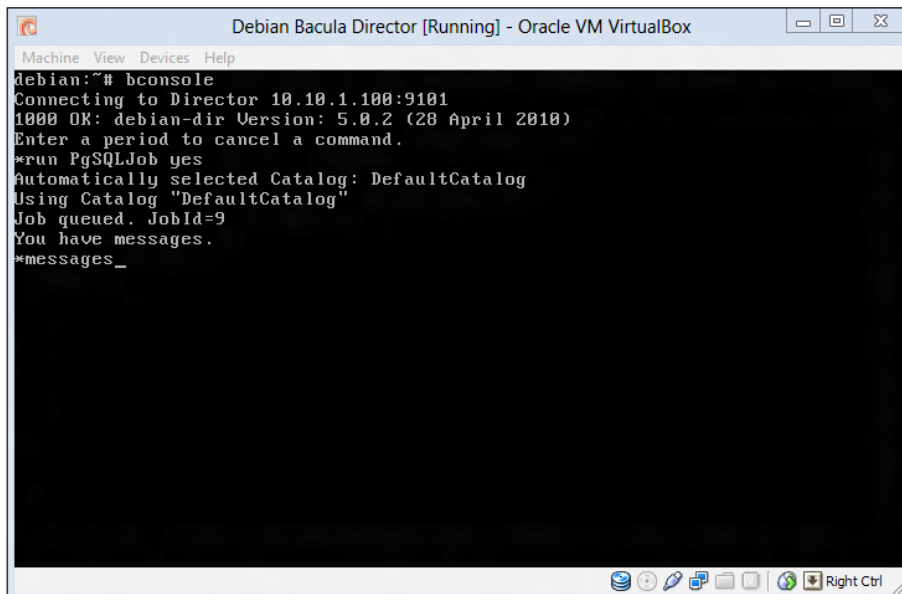
Now test the backup using `bconsole` as follows:

```

$ bconsole
Connecting to Director 10.10.1.100:9101
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)
Enter a period to cancel a command.
*run PgSQLJob yes
Job queued. JobId=10
*messages
...
Termination:           Backup OK

```

The output of the `bconsole` command is shown in the following screenshot:



How it works...

We're using `ClientRunBeforeJob` and `ClientRunAfterJob` directives to run a script that takes a complete SQL dump of the PostgreSQL databases to `/var/lib/pgsql-dump` folder. Then, that folder is being backed up by Bacula.

Backing up OpenLDAP server (Become an expert)

This recipe explains how to perform backup of OpenLDAP Active Directory server

Getting ready

You will need a Bacula Director machine with a configured client and the Bacula File daemon running on the machine containing PostgreSQL server.

How to do it...

1. Log in to the client machine.
2. Create the following script in the `/usr/local/bin/ldapcopyall` file:

```
#!/bin/bash
slapcat -b <suffix>> /var/lib/ldap-dump/<suffix>.ldif
Repeat the last line for every LDAP suffix of your server.
```

If you only have a single suffix available, you may omit the `-b` option:

```
#!/bin/bash
slapcat>/var/lib/ldap-dump/ldap.ldif
```

3. Make the script executable and create a directory for storing dump as follows:

```
$ chmod +x /usr/local/bin/ldapcopyall
$ mkdir /var/lib/ldap-dump
```

4. Now on the Director machine, create a new `Job` and `FileSet` as follows:

```
Job {
    Name = LDAPJob
    Type = Backup
    Level = Incremental
    Messages = Standard
    Pool = Default
    Client = DefaultClient
    Fileset = LDAPFileset
    Storage = LocalStorage
    ClientRunBeforeJob = "/usr/local/bin/ldapcopyall"
```

```

        ClientRunAfterJob = "/bin/rm -rf /var/lib/ldap-dump/*"
    }

    Fileset {
        Name = LDAPFileset
        Include {
            Options {
                signature = MD5
            }
            File = /var/lib/ldap-dump
        }
    }
}

```

- Restart the daemon as follows to apply the updated configuration:

```
$ /etc/init.d/bacula-director restart
```

Now test the backup using `bconsole` as follows:

```

$ bconsole
Connecting to Director 10.10.1.100:9101
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)
Enter a period to cancel a command.
*run LDAPJob yes
Job queued. JobId=12
*messages
...
Termination:          Backup OK

```

How it works...

We're using the `ClientRunBeforeJob` and `ClientRunAfterJob` directives to run a script that takes a complete LDAP database dump to `/var/lib/ldap-dump` folder. Then, that folder is being backed up by Bacula. After the job is completed, dump folder is wiped.

Backing up mailboxes (Should know)

This recipe explains how to back up Maildir and mbox format mailboxes from Postfix and other mail servers.

Getting ready

You will need a Bacula Director machine with a configured client and a mail server with the Bacula File daemon running.

How to do it...

To back up mailboxes, perform the following steps:

1. Log in to the client machine.
2. First, you'll need to determine the type of mailbox storage configured (Maildir or /var/mail).



The storage type depends on your MDA (mail delivery agent) and its configuration.

Look for following files:

- ▶ /var/mail/<username> or /home/<username>/mail indicates /var/mail storage.
- ▶ /home/<username>/Maildir indicates Maildir storage type.

3. Now on the Director machine, create a new Job and FileSet as follows:

```
Job {
    Name = MailJob
    Type = Backup
    Level = Incremental
    Messages = Standard
    Pool = Default
    Client = DefaultClient
    Fileset = MailFileset
    Storage = LocalStorage
}
For Maildir:
Fileset {
    Name = MailFileset
    Include {
        Options {
            signature = MD5
        }
        File = /home/alice/Maildir
        File = /home/bob/Maildir
    }
}
For /var/mail:
Fileset {
    Name = MailFileset
    Include {
```

```
Options {  
    signature = MD5  
}  
File = /var/mail/alice  
File = /home/alice/Mail  
File = /var/mail/bob  
File = /home/bob/Mail  
}}
```

4. Restart the daemon as follows to apply updated configuration:

```
$ /etc/init.d/bacula-director restart
```

5. Now test the backup using `bconsole` as follows:

```
$ bconsole  
Connecting to Director 10.10.1.100:9101  
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)  
Enter a period to cancel a command.  
*run MailJob yes  
Job queued. JobId=14  
*messages  
...  
Termination: Backup OK
```

How it works...

Backing up mail is straightforward. We determine the location of users' mailboxes and store them into the backup volume.

There's more...

This setup isn't only applicable to Postfix. In fact, you can use this configuration with any MDA that uses Maildir or `/var/mail` (mbox) mailbox format.

Backing up Zarafa (Become an expert)

This recipe explains how to back up using the Zarafa groupware servers.

Getting ready

You will need a Bacula Director machine with a configured client and a Zarafa groupware server (either Community or Business edition), with the Bacula File daemon running. Backup will require additional free space of roughly same size as your mailboxes.

How to do it...

To back up Zarafa, perform the following steps:

1. Log in to the client machine.
2. Create a script named `zarafacopyall` and stored at `/usr/local/bin/zarafacopyall` as follows:

```
#!/bin/bash
if [ $1 = "Full" ]; then
echo Doing full backup
rm -rf /var/lib/zarafa-dump/*
fi
zarafa-backup -avJ -o /var/lib/zarafa-dump
```

The script will be given a backup type as the first argument and will either add new layer of incremental backup files or create a new backup from scratch.

3. Make the script executable and create a directory for storing dump as follows:

```
$ chmod +x /usr/local/bin/zarafacopyall
$ mkdir /var/lib/zarafa-dump
```

4. Now on the Director machine, create a new Job, FileSet, and Schedule as follows:

```
Job {
    Name = ZarafaJob
    Type = Backup
    Level = Incremental
    Messages = Standard
    Pool = Default
    Client = DefaultClient
    Fileset = ZarafaFileset
    Storage = LocalStorage
    Schedule = ZarafaSchedule
    ClientRunBeforeJob = "/usr/local/bin/zarafacopyall %1"
}

Fileset {
    Name = ZarafaFileset
    Include {
        Options {
            signature = MD5
        }
        File = /var/lib/zarafa-dump
    }
}
```

```

}

Schedule {
    Name = ZarafaSchedule
    Run = Level=Full 1st Saturday at 11:59
    Run = Level=Incremental Monday-Friday at 23:05
}

```

5. Restart the daemon as follows to apply the updated configuration:

```
$ /etc/init.d/bacula-director restart
```

6. Now test the backup using `bconsole` as follows:

```

$ bconsole
Connecting to Director 10.10.1.100:9101
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)
Enter a period to cancel a command.
*run ZarafaJob yes
Job queued. JobId=17
*messages
...
Termination:          Backup OK

```

How it works...

We use the `zarafa-backup` utility bundled with the Zarafa server to create backup packages. By default, `zarafa-backup` will add a new incremental set of files after each run. As a result of daily backup, the folder can grow enormously big, so we will purge the incremental backup stack monthly and initiate a full from-scratch backup during the first weekend of the month.

The `%1` substitute in the `ClientRunBeforeJob` directive will be replaced by a backup level specified either as Full, Differential, or Incremental.

There's more...

Our setup will skip backup of the Junk mail folders for the sake of disk space, but you can enable it by removing the `-J` option from the `zarafa-backup` call.

You may also limit backup to specified subset of Zarafa users. To achieve this, use the modified `/usr/local/bin/zarafacopyall` script as follows:

```
#!/bin/bash
if [ $1 = "Full" ]; then
echo Doing full backup
rm -rf /var/lib/zarafa-dump/*
fi

zarafa-backup -vJ -u alice -o /var/lib/zarafa-dump
zarafa-backup -vJ -u bob -o /var/lib/zarafa-dump
```

Backups behind a firewall (Become an expert)

This recipe explains how to set up backups when the Director and client are separated by a firewall.

Getting ready

You will need a Bacula Director machine and a client machine located in different subnetworks. In this recipe we suppose the Director machine resides in subnet 10.10.1.0/24 with IP address 10.10.1.100, and the client machine resides in subnet 10.10.2.0/24 with IP address 10.10.2.101.

How to do it...

To perform a backup when the Director and client machine are separated by a firewall, perform the following steps:

1. Log in to the Director machine.
2. First, we must use a **Fully Qualified Domain Name (FQDN)** for our Storage daemon. Add this domain name to `/etc/hosts` as follows:

```
$ echo "10.10.1.100 local-sd" >> /etc/hosts
```

3. Change the Storage (`/etc/bacula/bacula-sd.conf`) configuration to use it, as follows:

```
Storage {
    Name = LocalStorage
    SDPort = 9103
    WorkingDirectory = "/var/lib/bacula"
    Pid Directory = "/var/run/bacula"
```

```
Maximum Concurrent Jobs = 20
SDAddress = local-sd
}
```

4. Generate an SSH public key using the `ssh-keygen` utility, using default settings as follows:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx root@debian
Examine your newly generated public key:
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAA... root@debian
```

Copy the whole line from `ssh-rsa` to `root@debian`.

5. Now log in to the client machine and append the copied line to the `~/.ssh/authorized_keys` file, prepending the `no-pty` token to the line as follows:


```
$ echo "no-pty ssh-rsa AAAA... root@debian" >> ~/.ssh/authorized_keys
```

6. Add the Storage FQDN as follows:

```
$ echo "127.0.0.1 local-sd" >> /etc/hosts
```

7. Coming back to the Director, move the keys to a location accessible by Bacula as follows:

```
$ mv ~/.ssh/id_rsa* /etc/bacula
$ chown bacula: /etc/bacula/id_rsa*
```

Test the SSH connection as follows:

```
$ su bacula -s /bin/bash -c "ssh -i /etc/bacula/id_rsa
root@10.10.2.101"
PTY allocation request failed on channel 0
```

You should see the **PTY allocation request failed** line.

8. Create a script that will maintain the tunnel in the `/usr/local/bin/bacula-tunnel` file as follows:

```
#!/bin/bash
CLIENT=$2
ADDR=10.10.1.100

case "$1" in
    start)
        ssh -fnCN2 -o PreferredAuthentications=publickey -i /
etc/bacula/id_rsa -R 9101:$ADDR:9101 -R 9103:$ADDR:9103 -L
9102:localhost:9102 root@$CLIENT > /dev/null 2>&1
        exit $?
        ;;

    stop)
        kill 'ps ax | grep "ssh -fnCN2 -o PreferredAuthentications
=publickey -i /etc/bacula/id_rsa" | grep "$CLIENT" | awk '{ print
$1 }''
        exit $?
        ;;

    *)
        exit 1
        ;;
esac
```

9. Create a new Client, Job, Storage, and FileSet in the `/etc/bacula/bacula-dir.conf` file as follows:

```
Storage {
    Name = LocalStorage
    Address = local-sd
    SDPort = 9103
    Password = "password-sd"
    Device = FileStorage
    Media Type = File
}

Client {
    Name = FirewallClient
    Address = localhost
    Password = "password-fd"
    Catalog = DefaultCatalog
    FD Port = 9102
}
```

```
Job {
    Name = FirewallJob
    Type = Backup
    Level = Incremental
    Messages = Standard
    Pool = Default
    Client = FirewallClient
    Fileset = FirewallFileset
    Storage = LocalStorage
    RunBeforeJob = "/usr/local/bin/bacula-tunnel start 10.10.2.101"
    RunAfterJob = "/usr/local/bin/bacula-tunnel stop 10.10.2.101"
}
Fileset {
    Name = FirewallFileset
    Include {
        Options {
            signature = MD5
        }
        File = /home
    }
}
```

10. Restart the daemons as follows to apply the updated configuration:

```
$ /etc/init.d/bacula-director restart
$ /etc/init.d/bacula-sd restart
```

11. Now test the backup using bconsole as follows:

```
$ bconsole
Connecting to Director 10.10.1.100:9101
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)
Enter a period to cancel a command.
*run FirewallJob yes
Job queued. JobId=17
*messages
...
Termination:          Backup OK
```

How it works...

We created the network tunnel between the Director and the client, which will forward Director and Storage ports to the client, and forward the client port to the Director, thus creating an illusion of a local Storage and Director for the client.

We utilize the name resolution trick with the Storage, as both the Director and the client have to connect to Storage using its full name; we create a fake domain name and map it to different network addresses on both hosts.

The %2 substitute in the `RunBeforeJob` directive (which is executed before any communications occur) will be replaced by the client address.

Backing up a Windows client (Should know)

This recipe explains how to back up a Windows client.

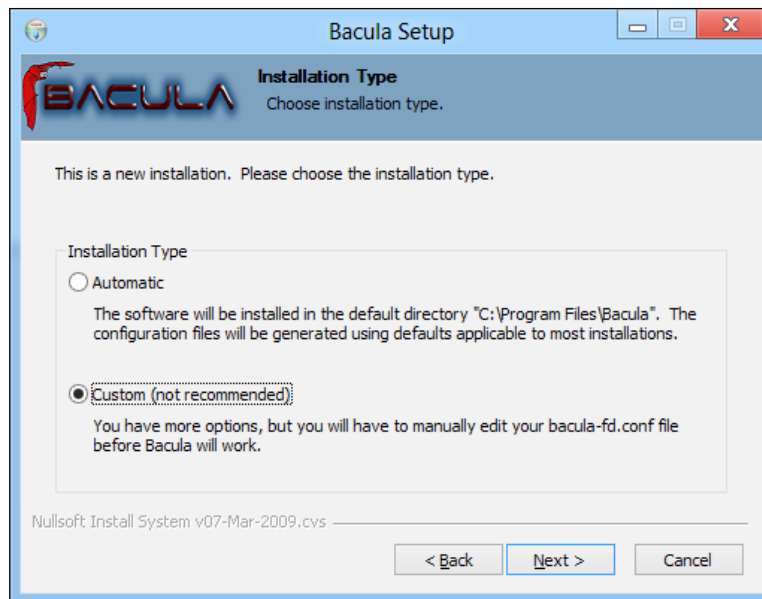
Getting ready

You will need a Bacula Director machine set up as per the *Simple Directory Backup* recipe and a client machine running Microsoft Windows.

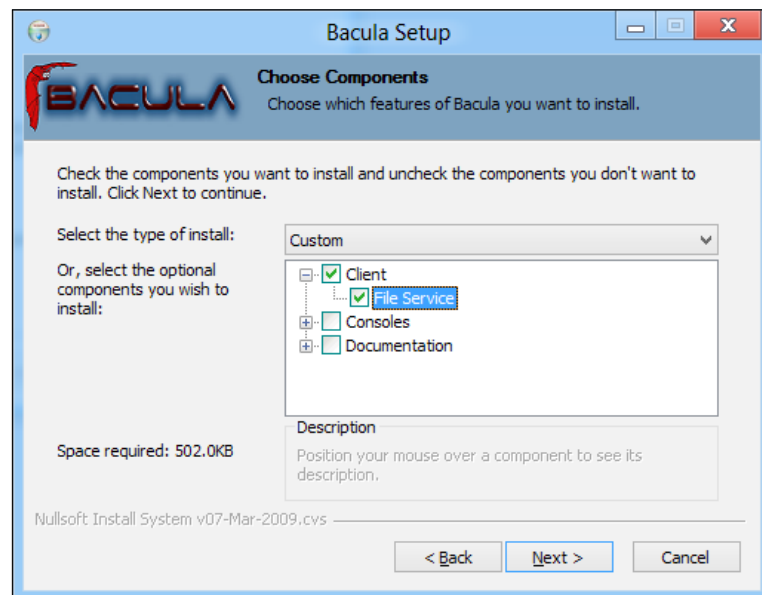
How to do it...

To back up a Windows client, perform the following steps:

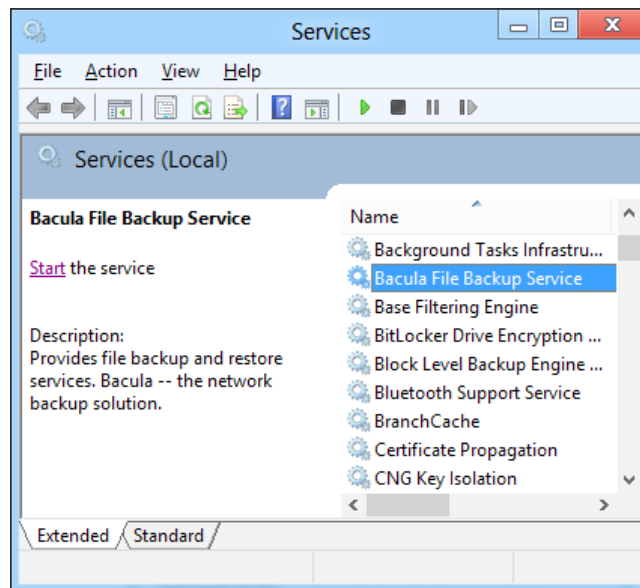
1. Download the Bacula Client installation package for your architecture from http://sourceforge.net/projects/bacula/files/Win32_64.
2. Install it using the **Custom** installation option as shown in the following screenshot:



3. We only need the File daemon so uncheck installation components other than **File Service** before proceeding, as shown in the following screenshot:



4. On the succeeding screens, enter a name and password for your client (we used `debian-fd` and `password-fd` in previous recipes), and director name (`debian-dir`).
5. After the installation is complete, locate the configuration file (`C:\Program Files\Bacula\bacula-fd.conf` if you used default installation path) and remove the `Restricted Director` section of it.
6. Now press `Win + R` and type in the `services.msc` command to launch the **Services** management console. Click on `Bacula File Backup Service` and click on the **Start** to launch it as shown in the following screenshot:



Now you should be able to verify client connectivity using the `bconsole` command from the Director machine as follows:

```
$ bconsole
Connecting to Director 10.10.1.100:9101
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)
Enter a period to cancel a command.
*status client
Automatically selected Client: DefaultClient
Connecting to Client DefaultClient at 10.10.1.101:9102

debian-fd Version: 5.2.10 (28 June 2012)  VSS Linux Cross-compile
Win32
```

```
Daemon started 05-Jul-12 12:51. Jobs: run=0 running=0.
Microsoft Professional (build 8400), 64-bit
Heap: heap=0 smbytes=16,428 max_bytes=16,523 bufs=50 max_bufs=51
Sizeof: boffset_t=8 size_t=4 debug=0 trace=1
Running Jobs:
Director connected at: 05-Jul-12 12:52
No Jobs running.
====

Terminated Jobs:
====
*
```

Multiple storage servers (Should know)

This recipe explains how to set up multiple storage servers for different backup scenarios.

Getting ready

You will need a Bacula Director machine set up as per the *Simple directory backup* recipe and two machines to be used as Storage (10.10.1.101 and 10.10.1.102 in this recipe).

How to do it...

To set up multiple storage servers, perform the following steps:

1. Log in to the first Storage machine and install the Bacula Storage daemon as follows:

```
$ apt-get install bacula-sd-sqlite3
```
2. Replace the configuration file `/etc/bacula/bacula-sd.conf` with the following content:

```
Storage {
    Name = LocalStorage
    SDPort = 9103
    WorkingDirectory = "/var/lib/bacula"
    Pid Directory = "/var/run/bacula"
    Maximum Concurrent Jobs = 20
    SDAddress = 10.10.1.101
}
```

```

Director {
    Name = debian-dir
    Password = "password-sd"
}

Device {
    Name = FileStorage
    Media Type = File
    Archive Device = /var/test-backups
    LabelMedia = yes;
    Random Access = Yes;
    AutomaticMount = yes;
    RemovableMedia = no;
    AlwaysOpen = no;
}

Messages {
    Name = Standard
    director = debian-dir = all
}

```

3. Restart the daemon as follows to apply the changes:

```
$ /etc/init.d/bacula-sd restart
```

4. Repeat the same for the second Storage machine.

Now log in to the Director server and add Storage specifications to the `/etc/bacula/bacula-dir.conf` file as follows:

```

Storage {
    Name = StorageA
    Address = 10.10.1.101
    SDPort = 9103
    Password = "password-sd"
    Device = FileStorage
    Media Type = File
}

Storage {
    Name = StorageB
    Address = 10.10.1.102
    SDPort = 9103
    Password = "password-sd"
    Device = FileStorage
    Media Type = File
}

```

Make sure you put in the correct addresses and passwords.

5. Restart the daemon as follows to apply the changes:

```
$ /etc/init.d/bacula-director restart
```

6. Now you should be able to access both Storage servers from the Director as follows:

```
$ bconsole
Connecting to Director 10.10.1.100:9101
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)
Enter a period to cancel a command.
*status storage
The defined Storage resources are:
    1: StorageA
    2: StorageB
Select Storage resource (1-2): 1
Connecting to Storage daemon StorageA at 10.10.1.101:9103

LocalStorage Version: 5.0.2 (28 April 2010) i486-pc-linux-gnu
debian squeeze/sid

Running Jobs:
No Jobs running.

*status storage
The defined Storage resources are:
    1: StorageA
    2: StorageB
Select Storage resource (1-2): 2
Connecting to Storage daemon StorageB at 10.10.1.102:9103

LocalStorage Version: 5.0.2 (28 April 2010) i486-pc-linux-gnu
debian squeeze/sid

Running Jobs:
No Jobs running.

*
```

There's more...

To use a specific Storage for a job, add the `Storage` directive to the job definition as follows:

```
Job {
    Name = DefaultJob
    Type = Backup
    Level = Incremental
    Messages = Standard
    Pool = Default
    Client = DefaultClient
    Fileset = DefaultFileset
    Storage = StorageA
}
```

Setting up from scratch

This recipe provides an example to perform a backup from scratch.

Getting ready

To start following this example, you need two machines with Debian 6.0 installed. For installation, use either CD 1 or netinstall images available at <http://www.debian.org/distrib/>.

In case you are unsure about which options to use during the installation process, let's look at the main points.

Be sure to properly configure your network card. In this example we suppose that the first machine (Director) has an IP address 10.10.1.100 and the second one (Client) has IP address 10.10.1.101. If your network administrator requires that you use addresses assigned by the DHCP server (automatic network configuration), either ask your administrator to configure their DHCP server so it assigns a static predefined address to your machines, or use FQDN instead.

When choosing installation components, be sure to select both the **Standard System** and **SSH Server** options so you can access your system remotely later.

How to do it...

1. After the installation is complete and the system has rebooted, log in to the root shell (either physically or remotely) and install the Bacula Director components as follows:

```
$ apt-get update
$ apt-get install -y bacula-director-sqlite3 bacula-sd-sqlite3
bacula-console
```

2. Create a folder to store backups as follows:

```
$ mkdir /var/backups
$ chown bacula: /var/backups -R
```

3. Now replace the configuration file `/etc/bacula/bacula-director.conf` with the following content:

```
Director {
    Name = debian-dir
    DIRport = 9101
    QueryFile = "/etc/bacula/scripts/query.sql"
    WorkingDirectory = "/var/lib/bacula"
    PidDirectory = "/var/run/bacula"
    Maximum Concurrent Jobs = 1
    Password = "password-dir"
    DirAddress = 10.10.1.100
}

Storage {
    Name = debian-sd
    Address = 10.10.1.100
    SDPort = 9103
    Password = "password-sd"
    Device = FileStorage
    Media Type = File
}

Job {
    Name = DefaultJob
    Type = Backup
    Level = Incremental
    Messages = Standard
    Pool = Default
    Client = DefaultClient
    Fileset = DefaultFileset
    Storage = debian-sd
}

Client {
    Name = DefaultClient
    Address = 10.10.1.101
    Password = "password-fd"
    Catalog = DefaultCatalog
    FD Port = 9102
}
```

```

Fileset {
    Name = DefaultFileset
    Include {
        Options {
            signature = MD5
        }
        File = /home
    }
}

Catalog {
    Name = DefaultCatalog
    dbname = "bacula"; dbuser = ""; dbpassword = ""
}

Messages {
    Name = Standard
    console = all, !skipped, !saved
    append = "/var/lib/bacula/log" = all, !skipped
    catalog = all
}

Pool {
    Name = Default
    Pool Type = Backup
    Recycle = yes
    AutoPrune = yes
    Volume Retention = 365 days
}

Pool {
    Name = Scratch
    Pool Type = Backup
}

```

If you're unsure of what text editor is available on your system, use the following command to find out :

```
$ sensible-editor /etc/bacula/bacula-director.conf
```

4. Replace the `/etc/bacula/bacula-sd.conf` file's code with the following code:

```

Storage {
    Name = debian-sd
    SDPort = 9103
    WorkingDirectory = "/var/lib/bacula"
    Pid Directory = "/var/run/bacula"
}

```

```
Maximum Concurrent Jobs = 20
SDAddress = 10.10.1.100
}

Director {
    Name = debian-dir
    Password = "password-sd"
}

Device {
    Name = FileStorage
    Media Type = File
    Archive Device = /var/backups
    LabelMedia = yes;
    Random Access = Yes;
    AutomaticMount = yes;
    RemovableMedia = no;
    AlwaysOpen = no;
}

Messages {
    Name = Standard
    director = debian-dir = all
}
```

5. And replace the `/etc/bacula/bconsole.conf` file's code with the following code:

```
Director {
    Name = localhost-dir
    DIRport = 9101
    address = 10.10.1.100
    Password = "password-dir"
}
```

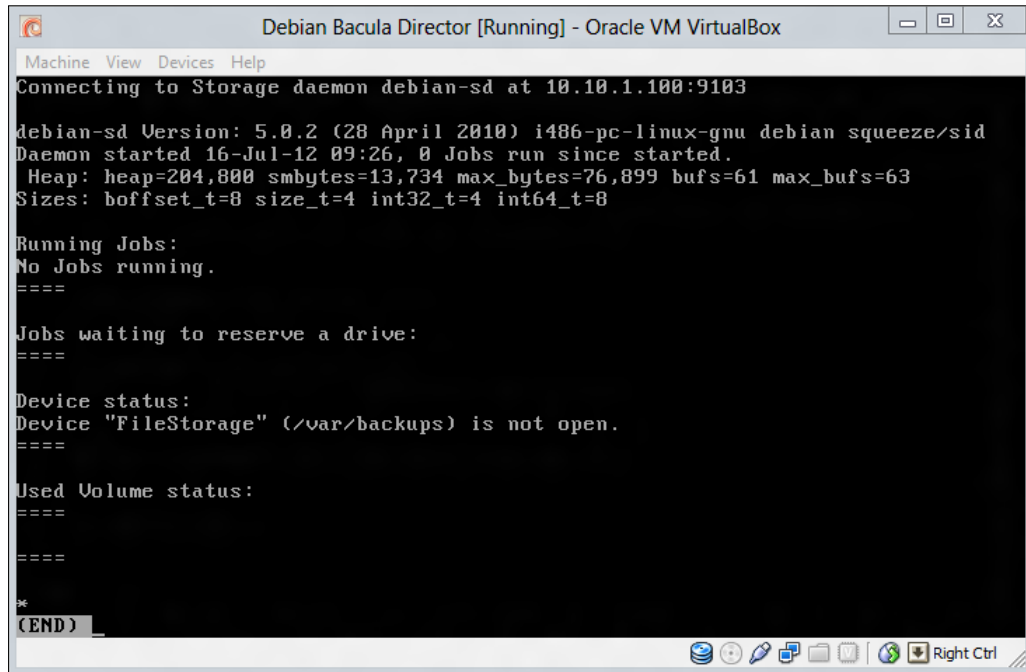
Now restart daemons as follows to apply the new configuration:

```
$ /etc/init.d/bacula-director restart
$ /etc/init.d/bacula-sd restart
```

6. Test the connectivity using `bconsole` as follows:

```
$ bconsole
Connecting to Director 10.10.1.100:9101
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)
Enter a period to cancel a command.
*status storage
```


The output of the `bconsole` command would look like the following screenshot:



```
Debian Bacula Director [Running] - Oracle VM VirtualBox
Machine View Devices Help
Connecting to Storage daemon debian-sd at 10.10.1.100:9103

debian-sd Version: 5.0.2 (28 April 2010) i486-pc-linux-gnu debian squeeze/sid
Daemon started 16-Jul-12 09:26, 0 Jobs run since started.
Heap: heap=204,800 smbytes=13,734 max_bytes=76,899 bufs=61 max_bufs=63
Sizes: boffset_t=8 size_t=4 int32_t=4 int64_t=8

Running Jobs:
No Jobs running.
====

Jobs waiting to reserve a drive:
====

Device status:
Device "FileStorage" (/var/backups) is not open.
====

Used Volume status:
====

*
(END)
```

7. Use the `label` command to create a new volume for backup storage as follows:

```
$ bconsole
Connecting to Director 10.10.1.100:9101
1000 OK: debian-dir Version: 5.0.2 (28 April 2010)
Enter a period to cancel a command.
*label
Automatically selected Catalog: DefaultCatalog
Using Catalog "DefaultCatalog"
Automatically selected Storage: debian-sd
Enter new Volume name: Test
Defined Pools:
    1: Default
    2: Scratch
Select the Pool (1-2): 1
```

```
Connecting to Storage daemon debian-sd at 10.10.1.100:9103 ...
Sending label command for Volume "Test" Slot 0 ...
3000 OK label. VolBytes=188 DVD=0 Volume="Test"
Device="FileStorage" (/var/backups)
```

8. The configuration for Director is now complete. Log in to the client machine and install the Bacula Client daemon as follows:

```
$ apt-get update
$ apt-get install -y bacula-fd
```

9. Replace the configuration file `/etc/bacula/bacula-fd` with the following content:

```
Director {
    Name = debian-dir
    Password = "password-fd"
}

FileDaemon {
    Name = debian-fd
    FDport = 9102
    WorkingDirectory = /var/lib/bacula
    Pid Directory = /var/run/bacula
    Maximum Concurrent Jobs = 20
    FDAddress = 10.10.1.101
}

Messages {
    Name = Standard
    director = debian-dir = all, !skipped, !restored
}
```

10. Restart the daemon as follows to apply the changes:

```
$ /etc/init.d/bacula-fd restart
```

11. Test your setup by issuing the `run` command in the Director `bconsole` and watching the messages as the backup proceeds.

Appendix: Mini tips

This section provides some tips that would be useful while performing backups.

Receiving e-mail notifications

It's possible to have Bacula Director send you an e-mail each time a backup job has completed or a storage device (tape drive) needs manual operation:

1. For this, replace the Messages section of your `/etc/bacula/bacula-dir.conf` file with the following code:

```
Messages {
    Name = Standard
    mailcommand = "bsmtp -h <mailhost>:<mailport>
                  -f \"\\(Bacula\\) %r\"ss
                  -s \"Bacula: %t %e of %c %l\" %r"s
    operatorcommand = "bsmtp -h <mailhost>:<mailport>
                      -f \"\\(Bacula\\) %r\"
                      -s \"Bacula: Operator request for %j\" %r"
    Mail = <email> = all, !skipped, !terminate
    append = "/var/lib/bacula/log" = all, !skipped, !terminate
    operator = <email> = mount
    console = all, !skipped, !saved
    catalog = all
}
```

Be sure to replace `<mailhost>` with the SMTP server hostname, `<mailport>` with the SMTP server port (default 25) and `<email>` with your e-mail address.

2. Restart the daemon as follows to apply changes:

```
$ /etc/init.d/bacula-director restart
```

Now you should be receiving e-mails for every completed job or device maintenance request.

Concurrent jobs

By default, Bacula only allows a single job to be executed at any given time.

Before modifying your configuration to allow multiple concurrent jobs, you should consider how this will affect performance – for example, performing two simultaneous backups to a single tape drive takes more time than running them one after another.

Bacula allows precise control of job concurrency. The `Maximum Concurrent Jobs` directive can be applied to each of the `Director`, `Job`, `Client` and `Storage` objects.

Adding this directive as follows to `Director` will limit the overall maximum number of concurrent jobs:

```
Director {  
    ...  
    Maximum Concurrent Jobs = 4  
    ...  
}
```

When added inside the `Job` object, this directive allows the `Job` to be run simultaneously on different clients:

```
Job {  
    ...  
    Maximum Concurrent Jobs = 4  
    ...  
}
```

For the `Client`, it will limit number of jobs that read a file from this client:

```
Client {  
    ...  
    Maximum Concurrent Jobs = 4  
    ...  
}
```

And in `Storage`, it limits the amount of jobs that concurrently write into the storage:

```
Storage {  
    ...  
    Maximum Concurrent Jobs = 4  
    ...  
}
```

After modifying your configuration file, don't forget to restart the daemon as follows to apply changes:

```
$ /etc/init.d/bacula-director restart
```

Handling holidays

In some situations you may want to skip backups on holidays – for example, to avoid manual tape changing since you will be absent on that day.

To avoid modifying the `config` file each time, create the following script, which will check for holidays:

```
#!/bin/sh
HOLIDAY_FILE="/etc/holidays"
DATE='date +%F'

if [ -z 'cat $HOLIDAY_FILE | grep ^$DATE' ]; then
{
    echo "Not a holiday.";
}
else
{
    exit 1;
}
Fi
```

Save the script, make it executable, and create a file `/etc/holidays` containing a list of vacation dates, as follows:

```
#Sundays
2012-07-15
2012-07-22
2012-07-29
```



Thank you for buying Network Backup with Bacula How-To

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: www.packtpub.com.

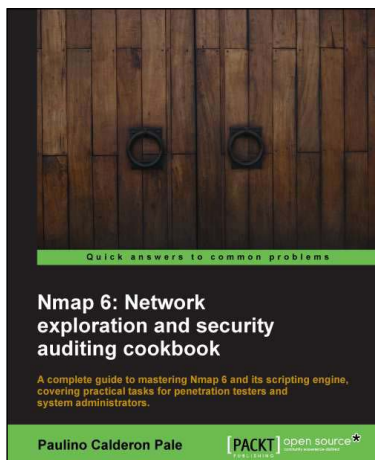
About Packt Open Source

In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.



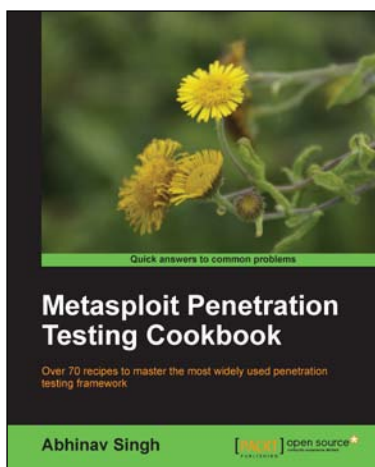
Nmap 6: Network exploration and security auditing Cookbook

ISBN: 978-1-84951-748-5

Paperback: 288 pages

A complete guide to mastering Nmap 6 and its scripting engine, covering practical tasks for penetration testers and system administrators

1. Master the power of Nmap 6
2. Learn how the Nmap Scripting Engine works and develop your own scripts!
3. 100% practical tasks, relevant and explained step-by-step with exact commands and optional arguments description



Metasploit Penetration Testing Cookbook

ISBN: 978-1-84951-742-3

Paperback: 268 pages

Over 70 recipes to master the most widely used penetration testing framework

1. More than 80 recipes/practical tasks that will escalate the reader's knowledge from beginner to an advanced level
2. Special focus on the latest operating systems, exploits, and penetration testing techniques
3. Detailed analysis of third party tools based on the Metasploit framework to enhance the penetration testing experience

Please check www.PacktPub.com for information on our titles