Lars Wirzenius
(liw@liw.fi)

Backups with Obnam

Obnam Version - 1.19
Version - 2016.620

Version 2016.620– – Document LaTeXed – 17th January 2016
[git] ▪ Branch: 1.19 @ 66f7715 ▪ Release: 1.19 (2016-01-17)

Contents

Page

1 **Introduction** 5
This manual 5

2 **README FIRST: A quick tour of Obnam** 7
Configuration 7
Initial backup 8
Incremental backups 8
Multiple clients in one repository 8
Removing old generations . 8
Restoring data 8
Using encryption 9

3 **You know you should** 10
Why backup? 10
Backup concepts 10
Backup strategies 12
Backups and security 13
Backup storage media considerations 14

4 **Installing Obnam** 16
Debian 16
Other systems 16

5 **Backing up** 17
Your first backup 17
Your second backup 18
Choosing what to backup, and what not to backup 18
Storing backups remotely . 19
URL syntax 20
Pull backups 21
Configuration files: a quick intro 21
When your precious data is very large 22
De-duplication 23
De-duplication and safety against checksum collisions 24
Locking 25
Consistency of live data . . 26

6 **Restoring from backups** 27
Oh no! It's all FUSEd together 27
Restoring without FUSE . . 28
An actual example of a restoration 29
Practice makes restores painless 29

7 **Forgetting old backup generations** 30
Choosing a schedule for forgetting generations . . 31

8 **Verifying backups** 32

9 **Sharing a repository between multiple clients** 34

10 **Using encryption** 36
You don't admit to being a spy, so isn't encryption unnecessary? 36
How Obnam encryption works 37
Setting up Obnam to use encryption 38
Checking if a repository uses encryption 39
Managing encryption keys in a repository 39

11 **Other stuff** 40
k4dirstat cache files 40

12 Case studies 41

13 Troubleshooting 42
Turning on full logging . . . 42
Reporting problems ("bugs") 42

14 Obnam configuration files and settings 44
Where is my configuration? 44
Configuration file syntax . . 45
Checking what my configuration is 46
Finding out all the configuration settings 46

15 The backup repository internals 47
Repository file permissions . 47

16 Obnam options 48
--version 48
-h 48
--help 48
--output=FILE 48
-r REPOSITORY 48
--repository=REPOSITORY 48
--client-name=CLIENT-NAME 49
--trace=TRACE 49
--quiet 49
--no-quiet 49
--verbose 49
--no-verbose 49
--pretend 49
--dry-run 49
--no-act 49
--no-pretend 50
--no-dry-run 50
--no-no-act 50
--lock-timeout=TIMEOUT . 50
--compress-with=PROGRAM 50
--root=ROOT 50
--testing-fail-matching=REGEXP 50
--warn-age=AGE 50
--critical-age=AGE 50
--to=TO 50
--generation=GENERATION 50
--keep=KEEP 51
--verify-randomly=N 51

17 Backing up 52
--exclude=EXCLUDE . . . 52
--exclude-caches 52
--no-exclude-caches 52
--one-file-system 52
--no-one-file-system 52
--checkpoint=SIZE 52
--de-duplicate=MODE . . . 52
--leave-checkpoints 53
--no-leave-checkpoints . . . 53
--small-files-in-btree 53
--no-small-files-in-btree . . . 53

18 Encryption 54
--encrypt-with=ENCRYPT-WITH 54
--keyid=KEYID 54
--weak-random 54
--no-weak-random 54
--symmetric-key-bits=SYMMETRIC-KEY-BITS 54

19 Integrity checking (fsck) 55
--fsck-fix 55
--no-fsck-fix 55
--fsck-ignore-chunks 55
--no-fsck-ignore-chunks . . . 55
--fsck-ignore-client=NAME . 55
--fsck-last-generation-only . 55
--no-fsck-last-generation-only 55
--fsck-skip-generations . . . 55
--no-fsck-skip-generations . . 56
--fsck-skip-dirs 56
--no-fsck-skip-dirs 56
--fsck-skip-files 56
--no-fsck-skip-files 56
--fsck-skip-per-client-b-trees 56
--no-fsck-skip-per-client-b-trees 56
--fsck-skip-shared-b-trees . . 56
--no-fsck-skip-shared-b-trees 56

20 Logging 57
--log=FILE 57
--log-level=LEVEL 57
--log-max=SIZE 57
--log-keep=N 57
--log-mode=MODE 57

21 Mounting with FUSE **58**
--viewmode=MODE 58
--fuse-opt=FUSE 58

22 Performance **59**
--dump-memory-profile=METHOD 59
--memory-dump-interval=SECONDS 59

23 Performance tweaking **60**
--node-size=SIZE 60
--chunk-size=SIZE 60
--upload-queue-size=SIZE . 60
--lru-size=SIZE 60
--idpath-depth=IDPATH-DEPTH 60
--idpath-bits=IDPATH-BITS 60
--idpath-skip=IDPATH-SKIP 60
--chunkids-per-group=NUM 61

24 SSH/SFTP **62**
--ssh-key=FILENAME . . . 62
--strict-ssh-host-keys 62
--no-strict-ssh-host-keys . . 62
--ssh-known-hosts=FILENAME 62
--pure-paramiko 62
--no-pure-paramiko 62

25 Performance tuning **63**
Introduction 63
Measurements 63
Discussion 65
Running Obnam under the Python profiler 65

26 The inbuilt help **67**
The help file 67
Options: 68
Backing up: 69
Configuration files and settings: 70
Development of Obnam itself: 70
Encryption: 70
Integrity checking (fsck): 71
Logging: 71
Mounting with FUSE: 71
Peformance: 71
Performance tweaking: 72
SSH/SFTP: 72
The 'man' page 72
Making backups 74
Verifying backups . . . 75
URL syntax 75
Generation specifications 76
Policy for keeping and removing backup generations 76
Using encryption . . . 77
Configuration files . . . 77
Multiple clients and locking 78
OPTIONS 78
Backing up 80
Configuration files and settings 81
Development of Obnam itself 81
Encryption 81
Integrity checking (fsck) 81
Logging 82
Mounting with FUSE . 82
Peformance 82
Performance tweaking 82
SSH/SFTP 83
EXIT STATUS 83
ENVIRONMENT . . . 84
FILES 84
EXAMPLE 84
SEE ALSO 85

27 Errors - code and names **86**
By error code 86
By name 87

28 See Also **91**

29 Legal stuff **92**

30 Supporting Obnam development **93**

31 News **94**

Glossary **118**

Index of commands **120**

General Index **123**

Chapter 1

Introduction

... backups? did someone talk about backups? I'm sure I heard someone mention backups here somewhere. Backups! BACKUPS! BACKUPS ARE AWESOME!

That's a direct quote from my IRC history. I find backups quite interesting, particularly from an implementation point of view, and I may sometimes obsess about them a little bit. This is why I've written my own backup software. It's called Obnam. This is its manual.

I'm unusual: most people find backups boring at best, and tedious most of the time. When I talk with people about backups, the usual reaction is "um, I know I should". There are a lot of reasons for this. One is that backups are a lot like insurance: you have to spend time, effort, and money up front to have any use for them. Another is that the whole topic is scary: you have to think about when things go wrong, and that puts people off. A third reason is that while there are lots of backup tools and methods, it's not always easy to choose between them.

This manual is for the Obnam program, but it tries to be useful to everyone thinking about backups.

This manual

This manual has been written in "LaTeX" to benefit from its superior markup and features, for further information please see http://en.wikipedia.org/wiki/LaTeX. It has almost the same layout as the "official" OBNAM manual at http://obnam.org/ and uses the same sources. It is a "work-in-progress" and is not complete, but still being added to.

This means that

is what you say to the programme

and

what the programme says to you

Sharon Kimble.
17th January 2016

Chapter 2

README FIRST: A quick tour of Obnam

You probably only need to read this chapter.

This chapter gives a quick introduction to the most important parts of Obnam. The rest of the book is basically a verbose version of this chapter. You should start by reading this chapter, then pretend you've read the rest, and everyone will look at you in awe at cocktail parties. I promise, nobody else will have read the rest of the book either, so there's no risk of getting caught.

Configuration

Obnam does not require a configuration file, and you can configure everything using command line options. You can, however, use a configuration file: save it as '~/.obnam.conf' and make it have content like this:

```
[config]
repository = sftp://your.server/home/youruser/backups/
log = /home/liw/obnam.log
```

The examples below assume you have created a configuration file, so that options do not need to be repeated every time.

You probably want to enable the 'log' setting, so that if there is a problem, you can find out all the information available to fix it from the log file.

Initial backup

Your first backup will be pretty big, and will take a long time. A long backup may crash, but that is not a problem: Obnam makes a checkpoint[1] every one hundred megabytes or so.

obnam backup $HOME

Incremental backups

When you've made your initial, full backup (possibly in stages), you can back up any changes simply by running Obnam again:

obnam backup $HOME

This will back up all new files, and any changed files. It will also record which files have been deleted since the previous backup.

You can run Obnam as often as you like. Only the changes from the previous run are backed up.

Multiple clients in one repository

You can backup multiple clients to a single repository by providing the option --client-name=<identifier> when running the program. Backup sets will be kept separate, but data de-duplication will happen across all the sets.

Removing old generations

Eventually your backup repository will grow so big you'll want to remove some old generations. The Obnam operation is called forget:

obnam forget --keep=30d

This would keep one backup from each of the last thirty calendar days, counting from the newest backup (not current time). If you've backed up several times during a day, only the latest generation from that day is kept.

Any data that is part of a generation that is to be kept will remain in the repository. Any data that exists only in those generations that is to be forgotten gets removed.

Restoring data

You will hopefully never need this, but the whole point of having backups is to restore data in case of a disaster.

obnam restore --to=/var/tmp/my-recovery $HOME

[1] Not yet defined

The above command will restore your entire home directory to '/var/tmp/my-recovery', from the latest backup generation. If you only need some particular directory or file, you can specify that instead:

```
obnam restore --to=/var/tmp/my-recover $HOME/Archive/receipts
```

If you can't remember the name of the file you need, use 'obnam ls':

```
obnam ls > /var/tmp/my-recovery.list
```

This will output the contents of the backup generation, in a format similar to 'ls -lAR'. Save it into a file and browse that. (It's a fairly slow command, so it's comfortable to save to a file.)

See also An actual example of a restoration.

Using encryption

Obnam can use the GnuPG program to encrypt the backup. To enable this, you need to have or create a PGP key, and then configure Obnam to use it:

```
[config]
encrypt-with = CAFEBABE
```

Here, 'CAFEBABE' is the key identifier[2] for your key, as reported by GnuPG. You need to have 'gpg-agent' or equivalent software configured, for now, because Obnam has no way to ask for or configure the passphrase.

After this, Obnam will automatically encrypt and decrypt data.

Note that if you encrypt your backups, you'll want to back up your GPG key in some other way. You can't restore any files from the obnam backup without it, so you can't rely on the same obnam backup to back up the GPG key itself. Back up your passphrase-encrypted GPG key somewhere else, and make sure you have a passphrase strong enough to stand up to offline brute-force attacks. Remember that if you lose access to your GPG key, your entire backup becomes useless.

If you enable encryption after making backups, you need to start over with a new repository.

You can't mix encrypted and unencrypted backups in the same repository.

(There are a bunch of Obnam commands for administering encryption. You won't need them, unless you share the same repository with several machines. In that case, you should read the manual page.)

See also Setting up Obnam to use encryption.

[2] Not yet defined

Chapter 3

You know you should

This chapter is philosophical and theoretical about backups. It discusses why you should back up, various concepts around backups, what kinds of things you should think about when setting up backups and what to do in the long term (verification, etc). It also discusses some assumptions Obnam makes and some constraints it imposes.

Why backup?

Backup concepts

This section covers core concepts in backups, and defines some terminology used in this book.

Live data[3] is the data you work with or keep. It's the files on your hard drive: the documents you write, the photos you save, the unfinished novels you wish you'd finish.

Most live data is precious[4] in that you'll be upset if you lose it. Some live data is not precious: your web browser cache probably isn't, for example. This distinction can let you limit the amount of data you need to back up, which can significantly reduce your backup costs.

A backup[5] is a spare copy of your live data. If you lose some or all of your live data, you can get it back ("restore[6]") from your backup. The backup copy is, by practical necessity, older than your live data, but if you made the backup recently enough, you won't lose much.

[3] all the data you have

[4] all the data you care about, see also Backup concepts

[5] a separate safe copy of your live data that will remain intact even if the primary copy gets destroyed deleted or wrongly modified

[6] retrieving data from a backup repository

Sometimes it's useful to have more than one old backup copy of your live data. You can have a sequence of backups, made at different times, giving you a backup history[7]. Each copy of your live data in your backup history is a generation[8]. This lets you retrieve a file you deleted a long time ago, but didn't realise you needed until now. If you only keep one backup version, you can't get it back, but if you keep, say, a daily backup for a month, you have a month to realise you need it, before it's lost forever.

The place your backups are stored is the backup repository[9]. You can use many kinds of backup media[10] for backup storage: hard drives, tapes, optical disks (DVD-R, DVD-RW, etc), USB flash drives, online storage, etc. Each type of medium has different characteristics: size, speed, convenience, reliability, price, which you'll need to balance for a backup solution that's reasonable for you.

You may need multiple backup repositories or media, with one of them located off-site[11], away from where your computers normally live. Otherwise, if your house burns down, you'll lose all your backups too.

You need to verify[12] that your backups work. It would be awkward to go to the effort and expense of making backups and then not be able to restore your data when you need to. You may even want to test your disaster recovery[13] by pretending that all your computer stuff is gone, except for the backup media. Can you still recover? You'll want to do this periodically, to make sure your backup system keeps working.

There is a very large variety of backup tools[14]. They can be very simple and manual: you can copy files to a USB drive using your file manager, once in a blue moon. They can also be very complex: enterprise backup products that cost huge amounts of money and come with a multi-day training package for your sysadmin team, and which require that team to function properly.

You'll need to define a backup strategy[15] to tie everything together: what live data to back up, to what medium, using what tools, what kind of backup history to keep, and how to verify that they work.

[7] all the backup generations

[8] a backup in a series of backups of the same live data, to give historical insight

[9] the location where your backups are stored

[10] where a backup repository is stored

[11] a backup repository stored physically far away from the live data

[12] making sure a backup system works and that data actually can be restored from backups and that the backups have not become corrupted

[13] what you do when something goes wrong

[14] FIXME - not defined

[15] a plan for how to make sure your data is safe even if the dinosaurs return in space ships to re-take world now that the ice age is over

Backup strategies

You've set up a backup repository, and you have been backing up to it every day for a month now: your backup history is getting long enough to be useful. Can you be happy now?

Welcome to the world of threat modelling. Backups are about insurance, of mitigating small and large disasters, but disasters can strike backups as well. When are you so safe, that no disaster can affect you?

There is always a bigger disaster waiting to happen. If you backup to a USB drive on your work desk, and someone breaks in and steals both your computer and the USB drive, the backups did you no good.

You fix that by having two USB drives, and you keep one with your computer and the other in a bank vault. That's pretty safe, unless there's an earth quake that destroys both your home and the bank.

You fix that by renting online storage space from another country. That's quite good, except there's a bug in the operating system that you use, which happens to be the same operating system the storage provider uses, and hackers happen to break into both your and their systems, wiping all files.

You fix that by hiring a 3D printer that prints slabs of concrete on which your data is encoded using QR codes. You're safe until there's a meteorite hits Earth and destroys the entire civilisation.

You fix that by sending out satellites with copies of your data, into stable orbits around all nine planets (Pluto is too a planet!) in the solar system. Your data is safe, even though you yourself are dead from the meteorite, until the Sun goes supernova and destroys everything in the system.

There is always a bigger disaster. You have to decide which ones are likely enough that you want to consider them, and also decide what the acceptable costs are for protecting against them.

A short list of scenarios for thinking about threats:

- What if you lose your computer?
- What if you lose your home and all of its contents?
- What if the area in which you live is destroyed?
- What if you have to flee your country?

These questions do not cover everything, but they're a start. For each one, think about:

- Can you live with your loss of data? If you don't restore your data, does it cause a loss of memories, or some inconvenience in your daily life, or will it make it nearly impossible to go back to living and working normally? What data do you care most about?

- How much is it worth to you to get your data back, and how fast do you want that to happen? How much are you willing to invest money and effort to do the initial backup, and to continue backing up over time? And for restores, how much are you willing to pay for that? Is it better for you to spend less on backups, even if that makes restores slower, more expensive, and more effort? Or is the inverse true?

The threat modelling here is about safety against accidents and natural disasters. Threat modelling against attacks and enemies is similar, but also different, and will be the topic of the next episode in the adventures of Bac-Kup.

Backups and security

You're not the only one who cares about your data. A variety of governments, corporations, criminals, and overly curious snoopers are probably also interested. (It's sometimes hard to tell them apart.) They might be interested to find evidence against you, blackmail you, or just curious about what you're talking about with your other friends.

They might be interested in your data from a statistical point of view, and don't particularly care about you specifically. Or they might be interested only in you.

Instead of reading your files and email, or looking at your photos and videos, they might be interested in preventing your access to them, or to destroy your data. They might even want to corrupt your data, perhaps by planting child porn in your photo archive.

You protect your computer as well as you can to prevent these and other bad things from happening. You need to protect your backups with equal care.

If you back up to a USB drive, you should probably make the drive be encrypted. Likewise, if you back up to online storage. There are many forms of encryption, and I'm unqualified to give advice on this, but any of the common, modern ones should suffice except for quite determined attackers.

Instead of, or in addition to, encryption, you could ensure the physical security of your backup storage. Keep the USB drive in a safe, perhaps, or a safe deposit box.

The multiple backups you need to protect yourself against earthquakes, floods, and roving gangs of tricycle-riding clowns, are also useful against attackers. They might corrupt your live data, and the backups at your home, but probably won't be able to touch the USB drive encased in concrete and buried in the ground at a secret place only you know about.

The other side of the coin is that you might want to, or need to, ensure others do have access to your backed up data. For example, if the clown gang kidnaps you, your spouse might need access to your backups to be able to contact your MI6 handler to ask them to rescue you. Arranging safe access to (some) backups is an interesting problem to which there are various solutions. You could give your spouse the encryption passphrase, or give the passphrase to a trusted friend or your lawyer. You could also use something like http://www.digital-scurf.org/software/libgfshare to escrow encryption keys more safely.

Backup storage media considerations

This section discusses possibilities for backup storage media, and their various characteristics, and how to choose the suitable one for oneself.

There are a lot of different possible storage media. Perhaps the most important ones are -

- Magnetic tapes of various kinds.
- Hard drives: internal vs external, spinning magnetic surfaces vs SSDs vs memory sticks.
- Optical disks: CD, DVD, Blu-ray.
- Online storage of various kinds.
- Paper.

We'll skip the more exotic or unusual forms, such as microfilm.

Magnetic tapes

Magnetic tapes are traditionally probably the most common form of backup storage. They can be cheap per gigabyte, but tend to require a fairly hefty initial investment in the tape drive. Much backup terminology comes from tape drives: full backup[16] vs incremental backup[17], especially. Obnam doesn't support tape drives at all.

Hard drives

Hard drives are a common modern alternative to tapes, especially for those who do not wish to pay for a tape drive. Hard drives have the benefit of every bit of backup being accessible at the same speed as any other bit, making finding a particular old file easier and faster. This also enables snapshot backups[18], which is the model Obnam uses.

[16] a fresh backup of all precious live data

[17] a backup of any changes (new files, modified files, deletions) compared to a previous backup generation (either the previous full backup or the previous incremental backup); usually, you can't remove a full backup without removing all of the incremental backups that depend on it

[18] an alternative to full/incremental backups where every backup generation is effectively a full backup of all the precious live data and can be restored and removed as easily as any other generation

Different types of hard drives have different characteristics for reliability, speed, and price, and they may fluctuate fairly quickly from week to week and year to year. We won't go into detailed comparisons of all the options. From Obnam's point of view, anything that can look like a hard drive (spinning rust, SSD, USB flash memory stick, or online storage) is useable for storing backups, as long as it is re-writable.

Optical disks

Optical disks, particularly the kind that are write-once and can't be updated, can be used for backup storage, but they tend to be best for full backups that are stored for long periods of time, perhaps archived permanently, rather than for a actively used backup repository. Alternatively, they can be used as a kind of tape backup, where each tape is only ever used once. Obnam does not support optical drives as backup storage.

Paper

Paper likewise works better for archival purposes, and only for fairly small amounts of data. However, a backup printed on good paper with archival ink can last decades, even centuries, and is a good option for small, but very precious data. As an example, personal financial records, secret encryption keys, and love letters from your spouse. These can be printed either normally (preferably in a font that is easy to OCR), or using two-dimensional barcode (e.g, QR). Obnam doesn't support these, either.

Obnam only works with hard drives, and anything that can simulate a read/writable hard drive, such as online storage. By an amazing co-incidence, this seems to be sufficient for most people.

Chapter 4

Installing Obnam

This chapter explains how to install Obnam. It is not a very extensive set of instructions, yet. In particular, it really only caters to Debian users. Instructions for other systems would be very much welcome.

Debian

It is easiest to install Obnam on a Debian system. If you're running Debian 'jessie' or a later release, Obnam is included and you can just install it -

sudo apt-get install obnam

There may be a newer version of Obnam on the author's site. The rest of this section explains how to install from there.

Add the following line to your '/etc/apt/sources.list' file -

deb http://code.liw.fi/debian wheezy main

Then run the following commands -

sudo apt-get update
sudo apt-get install obnam

The commands will complain that the PGP key used to sign the archive is not known to apt. You can either ignore this, or add the key from `http://code.liw.fi/apt.asc` to your key, after suitable verification.

Other systems

For other systems, you need to install from sources. See the "README" file in the source tree for instructions.

Chapter 5

Backing up

This chapter discusses the various aspects of making backups with Obnam.

Your first backup

Let's make a backup! To walk through the examples in this directory, you need to have some live data to backup. The examples use specific filenames for this. You'll need to adapt the examples to your own files. The examples assume your home directory is '/home/tomjon', and that you have a directory called 'Documents' in your home directory for your documents. Further, it assumes you have a USB drive mounted at '/media/backups', and that you will be using a directory 'tomjon-repo' on that drive as the backup repository.

With those assumptions, here's how you would backup your documents:

```
obnam backup -r /media/backups/tomjon-repo ~/Documents
```

That's all. It will take a little while, if you have a lot of documents, but eventually it'll look something like this:

```
Backed up 11 files (of 11 found), uploaded 97.7 KiB in 0s at 647.2 KiB/s
average speed
```

This tells you that Obnam found a total of eleven files, of which it backed up all eleven. The files contained a total of about a hundred kilobytes of data, and that the upload speed for that data was over six hundred kilobytes per second. The actual units are using IEC prefixes, which are base-2, to avoid ambiguity. See "Wikipedia on kibibytes" [19] for more information.

Your first backup run should probably be quite small to see that all settings are right without having to wait a long time. You may want to choose a small directory to start with, instead of your entire home directory.

[19] Wikipedia on kibibytes https://en.wikipedia.org/wiki/Kibibyte

Your second backup

Once you've run your first backup, you'll want to run a second one. It's done the same way:

```
obnam backup -r /media/backups/tomjon-repo ~/Documents
```

Note that you don't need to tell Obnam whether you want a full backup or an incremental backup. Obnam makes each backup generation be a snapshot of the data at the time of the backup, and doesn't make a difference between full and incremental backups. Each backup generation is equal to each other backup generation. This doesn't mean that each generation will store all the data separately. Obnam makes sure each new generation only backs up data that isn't already in the repository. Obnam finds that data in any file in any previous generation, amongst all the clients sharing the same repository.

We'll later cover how to remove backup generations, and you'll learn that Obnam can remove any generation, even if it shares some of the data with other generations, without those other generations losing any data.

After you've done your second backup generation, you'll want to see the generations you have:

```
obnam generations -r /media/backups/tomjon-repo
```

```
2 2014-02-05 23:13:50 .. 2014-02-05 23:13:50 (14 files, 100000 bytes)
5 2014-02-05 23:42:08 .. 2014-02-05 23:42:08 (14 files, 100000 bytes)
```

This lists two generations, which have the identifiers 2 and 5. Note that generation identifiers are not necessarily a simple sequence like 1, 2, 3. This is due to how some of the internal data structures of Obnam are implemented, and not because its author in any way thinks it's fun to confuse people.

The two time stamps for each generation are when the backup run started and when it ended. In addition, for each generation is a count of files in that generation (total, not just new or changed files), and the total number of bytes of file content data they have.

Choosing what to backup, and what not to backup

Obnam needs to be told what to back up, by giving it a list of directories, known as backup roots. In the examples in this chapter so far, we've used the directory '~/Documents' (that is, the directory 'Documents' in your home directory) as the backup root[20]. There can be multiple backup root[21]:

```
obnam -r /media/backups/tomjon-repo ~/Documents ~/Photos
```

[20] a directory that is to be backed up, including all files in it, and all its subdirectories
[21] a directory that is to be backed up, including all files in it, and all its subdirectories

Everything in the backup root directories gets backed up – unless it's explicitly excluded. There are several ways to exclude things from backups:

- The '--exclude' setting uses regular expressions that match the full pathname of each file or directory: if the pathname matches, the file or directory is not backed up. In fact, Obnam pretends it doesn't exist. If a directory matches, then any files and sub-directories also get excluded. This can be used, for example, to exclude all MP3 files ("--exclude='.mp3").
- The '--exclude-caches' setting excludes directories that contain a special "cache tag" file called 'CACHEDIR.TAG', that starts with a specific sequence of bytes. Such a tag file can be created in, for example, a Firefox or other web browser cache directory. Those files are usually not important to back up, and tagging the directory can be easier than constructing a regular expression for '--exclude'.
- The '--one-file-system' setting excludes any mount points and the contents of the mounted filesystem. This is useful for skipping, for example, virtual filesystems such as '/proc', remote filesystems mounted over NFS, and Obnam repositories mounted with 'obnam mount' (which we'll cover in the next chapter).

In general it is better to back up too much rather than too little. You should also make sure you know what is and isn't backed up. The '--pretend' option tells Obnam to run a backup, except it doesn't change anything in the backup repository, so it's quite fast. This way you can see what would be backed up, and tweak exclusions as needed.

Storing backups remotely

You probably want to store at least one backup remotely, or "off site". Obnam can make backups over a network, using the SFTP protocol (part of SSH). You need the following to achieve this:

- A server that you can access over SFTP. This can be a server you own, a virtual machine ("VPS") you rent, or some other arrangement. You could, for example, have a machine at a friends' place, in exchange for having one of their machines at your place, so that you both can backup remotely.
- An ssh-key for logging into the server. You can log in using passwords too, but it is quite cumbersome.
- Enough disk space on the server to hold your backups.

Obnam only uses the SFTP part of the SSH connection to access the server. To test whether it will work, you can run this command:

```
sftp USER@SERVER
```

Change ‘USER@SERVER‘ to be your actual user and address for your server. This should say something like ‘Connected to localhost.‘ and you should be able to run the ‘ls -la‘ command to see a directory list of files on the remote side.

Once all of that is set up correctly, to get Obnam to use the server as a backup repository, you only need to give an SFTP URL:

obnam -r sftp://USER@SERVER/ /my-precious-backups

For details on SFTP URLs, see the next section.

URL syntax

Whenever obnam accepts a URL, it can be either a local pathname, or an SFTP URL. An SFTP URL has the following form:

sftp://[user@]domain[:port]/path

where ‘domain‘ is a normal Internet domain name for a server, ‘user‘ is your username on that server, ‘port‘ is an optional numeric port number, and ‘path‘ is a pathname on the server side. Like **bzr**(1), but unlike the SFTP URL standard, the pathname is absolute, unless it starts with ‘/ /‘ in which case it is relative to the user’s home directory on the server.

Examples:

- ‘sftp://liw@backups.pieni.net/ /backup-repo‘ is the directory ‘backup-repo‘ in the home directory of the user ‘liw‘ on the server ‘backups.pieni.net‘. Note that we don’t need to know the absolute path of the home directory.
- ‘sftp://root@my.server.example.com/home‘ is the directory ‘/home‘ (note absolute path) on the server ‘my.server.example.com‘, and the ‘root‘ user is used to access the server.
- ‘sftp://foo.example.com:12765/anti-clown-society‘ is the directory ‘/anti-clown-society‘ on the server ‘foo.example.com‘, accessed via the port 12765.

You can use SFTP URLs for the repository, or the live data (‘–root‘), but note that due to limitations in the protocol, and its implementation in the paramiko library, some things will not work very well for accessing live data over SFTP. Most importantly, the handling of of hardlinks is rather suboptimal. For live data access, you should not end the URL with / / and should append a dot at the end in this special case.

Pull backups

Obnam can also access the live data over SFTP, instead of via the local filesystem. This means you can run Obnam on, say, your desktop machine to backup your server, or on your laptop to backup your phone (assuming you can get an SSH server installed on your phone). Sometimes it is not possible to install Obnam on the machine where the live data resides, and then it is useful to do a pull backup instead: you run Obnam on a different machine, and read the live data over the SFTP protocol.

To do this, you specify the live data location (the 'root' setting, or as a command line argument to 'obnam backup') using an SFTP URL. You should also set the client name explicitly. Otherwise Obnam will use the hostname of the machine on which it runs as the name, and this can be highly confusing: the client name is 'my-laptop' and the server is 'down-with-clowns' and Obnam will store the backups as if the data belongs to 'my-laptop'.

It gets worse if you backup your laptop as well to the same backup repository. Then Obnam will store both the server and the laptop backups using the same client name, resulting in much confusion to everyone.

Example:

```
obnam backup -r /mnt/backups sftp://server.example.com/home –client-name=server.example.com
```

Configuration files: a quick intro

By this time you may have noticed that Obnam has a number of configurable settings you can tweak in a number of ways. Doing it on the command line is always possible, but then you get quite long command lines. You can also put them into a configuration file.

Every command line option Obnam knows can be set in a configuration file. Later in this manual there is a whole chapter that covers all the details of configuration files, and all the various settings you can use. For now, we'll give a quick introduction.

An Obnam configuration looks like this:

```
[config]
repository = /media/backup/tomjon-repo
root = /home/liw/Documents, /home/liw/Photos
exclude = \.mp3
exclude-caches = yes
one-file-system = no
```

This form of configuration file is commonly known as an "INI file", from Microsoft Windows '.INI' files. All the Obnam settings go into a section titles '[config]', and each setting has the same name as the command line option, but without the double dash prefix. Thus, it's '--exclude' on the command line and 'exclude' in the configuration file.

Some settings can have multiple values, such as 'exclude' and 'root'. The values are comma separated. If there's a lot of values, you can split them on multiple lines, where the second and later lines are indented by space or TAB characters.

That should get you started, and you can reference the "Obnam configuration files and settings" chapter for all the details.

When your precious data is very large

When your precious data is very large, the first backup may a very long time. Ditto, if you get a lot of new precious data for a later backup. In these cases, you may need to be very patient, and just let the backup take its time, or you may choose to start small and add to the backups a bit at a time.

The patient option is easy: you tell Obnam to backup everything, set it running, and wait until it's done, even if it takes hours or days. If the backup terminates prematurely, e.g., because of a network link going down, you won't have to start from scratch thanks to Obnam's checkpoint support. Every gigabyte or so (by default) Obnam stops a backup run to create a checkpoint generation. If the backup later crashes, you can just re-run Obnam and it will pick up from the latest checkpoint. This is all fully automatic, you don't need to do anything for it to happen. See the '--checkpoint' setting for choosing how often the checkpoints should happen.

Note that if Obnam doesn't get to finish, and you have to re-start it, the scanning starts over from the beginning. The checkpoint generation does not contain enough state for Obnam to continue scanning from the latest file in the checkpoint: it'd be very complicated state, and easily invalidated by filesystem changes. Instead, Obnam scans all files, but most files will hopefully not have changed since the checkpoint was made, so the scanning should be relatively fast.

The only problem with the patient option is that your most precious data doesn't get backed up while all your large, but less precious data is being backed up. For example, you may have a large amount of downloaded videos of conference presentations, which are nice, but not hugely important. While those get backed up, your own documents do not get backed up.

You can work around this by initially excluding everything except the most precious data. When that is backed up, you gradually reduce the excludes, re-running the backup, until you've backed up everything. As an example, your first backup might have the following configuration:

```
obnam backup -r /media/backups/tomjon-repo ~ --exclude ~/Downloads
```

This would exclude all downloaded files. The next backup run might exclude only video files:

```
obnam backup -r /media/backups/tomjon-repo ~ --exclude ~/Downloads/'.*mp4'
```

After this, you might reduce excludes to allow a few videos, such as those whose name starts with a specific letter:

```
obnam backup -r /media/backups/tomjon-repo ~ --exclude ~/Downloads/'[b-zB-Z].*mp4'
```

Continue allowing more and more videos until they've all been backed up.

De-duplication

Obnam de-duplicates[22] the data it backs up, across all files in all generations for all clients sharing the repository. It does this by breaking up all file data into bits called chunks. Every time Obnam reads a file and gets a chunk together, it looks into the backup repository to see if an identical chunk already exists. If it does, Obnam doesn't need to upload the chunk, saving space, bandwidth, and time.

De-duplication in Obnam is useful in several situations:

- When you have two identical files, obviously. They might have different names, and be in different directories, but contain the same data.
- When a file keeps growing, but all the new data is added at the end. This is typical for log files, for example. If the leading chunks are unmodified, only the new data needs to be backed up.
- When a file or directory is renamed or moved. If you decide that the English name for the 'Photos' directory is annoying and you want to use the Finnish 'Valokuvat' instead, you can rename that in an instant. However, without de-duplication, you then have to backup all your photos again.
- When all of a team works on the same things, and everyone has copies of the same files, the backup repository only needs one copy of each file, rather than one per team member.

De-duplication in Obnam isn't perfect. The granularity of finding duplicate data is quite coarse (see the '--chunk-size' setting), and so Obnam often doesn't find duplication when it exists, when the changes are small.

De-duplication isn't useful in the following scenarios:

[22] this is a specialised data compression technique for eliminating duplicate copies of repeating data

- A file changes such that things move around within the file. The (current) Obnam de-duplication is based on non-overlapping chunks from the beginning of a file. If some data is inserted, Obnam won't notice that the chunks have shifted around. This can happen, for example, for disk or ISO images.
- Files with duplicate data that is not on a chunk boundary. For example, emails with large attachments. Each email recipient gets different 'Received' headers, which shifts the body and attachments by different amounts. As a result, Obnam won't notice the duplication.
- Data in compressed files, such as '.zip' or '.tar.xz' files. Obnam doesn't know about the file compression, and only sees the compressed version of the data. Thus, Obnam won'd de-duplicate it.

A future version of Obnam will hopefully improve the de-duplication algorithms. If you see this optimistic paragraph in a version of Obnam released in 2017 or later, please notify the maintainers. Thank you.

De-duplication and safety against checksum collisions

This is a bit of a scary topic, but it would be dishonest to not discuss it at all. Feel free to come back to this section later.

Obnam uses the MD5 checksum algorithm for recognising duplicate data chunks. MD5 has a reputation for being unsafe: people have constructed files that are different, but result in the same MD5 checksum. This is true. MD5 is not considered safe for security critical applications.

Every checksum algorithm can have collisions. Changing Obnam to use, say, SHA1, SHA2, or the as new SHA3 algorithm would not remove the chance of collisions. It would reduce the chance of accidental collisions, but the chance of those is already so small with MD5 that it can be disregarded. Or put in another way, if you care about the chance of accidental MD5 collisions, you should be caring about accidental SHA1, SHA2, or SHA3 collisions as well.

Apart from accidental collisions, there are two cases where you should worry about checksum collisions (regardless of algorithm).

First, if you have an enemy who wishes to corrupt your backed up data, they may replace some of the backed up data with other data that has the same checksum. This way, when you restore, your data is corrupted without Obnam noticing.

Second, if you're into researching checksum collisions, you're likely to have files that cause checksum collisions, and in that case, if you restore after a catastrophe, you probably want to get the files back intact, rather having Obnam confuse one with the other.

To deal with these situations, Obnam has three de-duplication modes, set using the '--deduplicate' setting:

- The default mode, ‘fatalist‘, assumes checksum collisions do not happen. This is a reasonable compromise between performance, safety, and security for most people.
- The ‘verify‘ mode assumes checksum collisions do happen, and verifies that the already backed up chunk is identical to the chunk to be backed up, by comparing the actual data. Doing this requires downloading the chunk from the backup repository, which can be quite slow, since checksums will often match. This is a useful mode if you have very fast access to the backup repository, and want to de-duplicate, such as when the backup repository is on a locally connected hard drive.
- The ‘never‘ mode turns off de-duplication completely. This is useful if you’re worried about checksum collisions, and do not require de-duplication.

There is, unfortunately, no way to get both de-duplication that is invulnerable to checksum collision and is fast even when accessing the backup repository is slow. The only way to be invulnerable is to compare the data, and if downloading the data from the repository is slow, then the comparison will take significant time.

Locking

Multiple clients can share a repository, and to prevent them from trampling on each other, they lock parts of the repository while working. The "Sharing a repository between multiple clients" chapter will discuss this in more detail.

If Obnam terminates abruptly, even if there’s only one client ever using the repository, the lock may stay around and prevent that one client for making new backups. The termination may be due to the network connection breaking, or due to a bug in Obnam. It can also happen if Obnam is interrupted by the user before it’s finished.

The Obnam command ‘force-lock‘ deals with this situation. It is dangerous, though. If you force open a lock that is in active use by a running Obnam instance, there will likely to be some stepping of toes. The result may, in extreme cases, even result in repository corruption[23]. So be careful.

If you’ve decided you can safely do it, this is an example of how to do it:

```
obnam -r /media/backups/tomjon-repo force-lock
```

Note that some of the locks are per-client, to prevent you from accidentally running Obnam twice for the same client, which would result in standing on your own toes: kind of impressive, but uncomfortable and not recommended.

[23] unwanted modification to (backup) data

If you need to force open a lock for specific client, you can specify the client name explicitly:

```
obnam --client-name magrat -r /media/backups/tomjon-repo force-lock
```

Consistency of live data

Making a backup can take a good while. While the backup is running, the filesystem may change. This leads to the snapshot of data Obnam presents as a backup generation being internally inconsistent. For example, before a backup you might have two files, A and B, which need to be kept in sync. You run a backup, and while it's happening, you change A, and then B. However, you're unlucky, and Obnam manages to backup A before you save your changes, and B after you save changes to that. The backup generation now has versions of A and B that are not synchronised. This is bad.

This can be dealt with in various ways, depending on the circumstances. Here's a few:

- The Logical Volume Manager (LVM) provides snapshots. You can set up your backups so that they first create a snapshot of each logical volume to be backed up, run the backup, and delete the snapshot afterwards. This prevents anyone from modifying the files in the snapshot, but allows normal use to continue while the backup happens.
- A similar thing can be done using the btrfs filesystem and its subvolumes.
- You can shut down the system, reboot it into single user mode, and run the backup, before rebooting back into normal mode. This is not a good way to do it, but it is the safest way to get a consistent snapshot of the filesystem.

Note that filesystem level snapshots can't really guarantee a consistent view of the live data. An application may be in the middle of writing a file, or set of files, when the snapshot is being made. To some extent this indicates an application bug, but knowing that doesn't let you sleep better.

Usually, though, most systems have enough idle time that a consistent backup snapshot can happen during that time. For a laptop, for example, a backup can be run while the user is elsewhere, instead of actively using the machine.

Part of your backup verification suite should check that the data in a backup generation is internally consistent, if that can be done. Otherwise, you'll either have to analyse the applications you use, or trust they're not too buggy.

If you didn't understand this section, don't worry and be happy and sleep well.

Chapter 6

Restoring from backups

The worst has happened! Your cat got confused between its litter box and your hard drive! Your goat deleted your most important document ever! Woe be you!

Let's stay calm. This is why you have backups. There's no need for exclamation marks. Take a deep breath, have a cup of tea, and all will be well.

There's two different approaches for restoring data with Obnam. One relies on a FUSE filesystem, which is a very nice piece of technology that allows Obnam to let you view your backups as just another directory. It is the preferred way, but it is not always available, so Obnam also provides a more primitive, less easy to use method.

Oh no! It's all FUSEd together

The 'obnam mount' command lets you look at your backups as if they were just another directory. This requires that you have FUSE setup. See the installation chapter for details on that. Most modern Linux desktops have this out of the box.

```
mkdir ~/backups
obnam mount --to ~/backups
```

Run the above command, and then look at the '~/backups' directory. You'll see something like this:

```
ls -l  /backups
total 12
drwxr-xr-x 24 root root 4096 Feb 11 21:41 2
drwxr-xr-x 24 root root 4096 Feb 11 21:41 5
lrwxr-xr-x 24 root root 4096 Feb 11 21:41 latest -> 5
```

Each directory in '~/backups' is a backup generation, named after the generation identifier Obnam invents. The 'latest' symbolic link points at the latest generation.

After this, you can restore a single file very easily:

```
cp ~/backups/latest/home/tomjon/Documents/iloveyou.txt ~/restored.txt
```

You can copy any files you want from the '~/backups' directory, from any generation, or all of them if you want to. You can look at files directly, before copying them out, too.

```
less ~/backups/2/home/tomjon/Documents/iloveyou.txt
```

This is an easy way to make sure you find the right version instead of just the latest one.

You can't delete anything in '~/backups'. That directory is read-only, and you can't accidentally, or on purpose, delete or modify anything there. This is intentional: the 'obnam mount' command is meant to be a safe way for you to look at your backups, not something you need to be careful about.

Once you're done looking at your backups, you can un-mount the repository:

```
fusermount -u ~/backups
```

In addition to doing these things from the command line, you can, of course, use your favorite file manager (graphical or textual) to look at your backed up files. The mounting and un-mounting (depending on your desktop setup) may need to be done on the command line.

Restoring without FUSE

When 'obnam mount' isn't available, you can do restores directly with just Obnam. Use 'obnam generations' and 'obnam ls' to find the right generation to restore, and then run a command like this:

```
obnam restore --to /tmp/tomjon-restored /home/tomjon/Documents
```

This would restore just the indicated directory. If you don't tell Obnam what to restore, it'll restore everything in the latest generation. You can choose a different generation with '--generation':

```
obnam restore --to /tmp/tomjon-restored --generation 2
```

Note that you can't restore to a directory that already exists. This is to prevent you from accidentally overwriting your live data with restored files. If you do want to replace your live data with restored files, you should restore to a temporary location first, and then move the files to where you want them to be.

An actual example of a restoration

I had a corrupted gnus file, and this is how I restored it from backup.

```
obnam --config=/home/foobar/cron/conf/obnam.conf generations> /home/foobar/cron/upload/obgen.txt
```

This copies all generations for the main obnam backup to obgen.txt, and this is part of that file.

```
1207586 2014-08-25 08:00:43 .. 2014-08-25 08:08:24 (385163 files, 175029819657 bytes)
1208367 2014-08-25 12:00:42 .. 2014-08-25 12:08:31 (385965 files, 175057598863 bytes)
1209313 2014-08-25 16:00:12 .. 2014-08-25 16:07:33 (386537 files, 175076976590 bytes)
1210254 2014-08-25 20:00:15 .. 2014-08-25 20:09:41 (386896 files, 175086483254 bytes)
```

And I decided to restore from generation 1208367.

This is the actual restore command.

```
obnam --config=/home/foobar/cron/conf/obnam.conf --generation=1208367 restore /home/foobar/News/rss/nnrss.el --to=/home/foobar/cron/upload/
```

This restores 'nnrss.el' to /home/foobar/cron/upload/ from where I was able to copy it back to its proper place in /home/foobar/News/rss/

Obviously you replace your user-name for foobar.

Practice makes restores painless

You should practice doing restores. This makes you trust your backups more, and lets you be calmer if disaster were to strike. (In fancier terms, you should test your disaster recovery plan.)

Do a trial restore of a few files, or all files, until you're sure you know how to do that. Then do it again, from time to time, to be sure your backups still work. It's much less frightening to do a real restore, when data has actually gone missing, if you've done it before.

In extreme cases, particularly if you're an Obnam developer, you perhaps format your hard drive and then do complete restore, just so you know you can. If you're not an Obnam developer, this is perhaps a bit extreme: at least use a separate hard drive instead of your normal one.

Chapter 7

Forgetting old backup generations

Every time you make a backup, your backup repository grows in size. To avoid overflowing all available storage, you need to get rid of some old backups. That's a bit of a dilemma, of course: you make backups in order to not lose data and now you have to do exactly that.

Obnam uses the term "forget" about removing a backup generation. You can specify which generation to remove manually, by generation identifier, or you can have a schedule to forget them automatically.

To forget a specific generation:

```
obnam forget 2
```

(This example assumes you have a configuration file that Obnam finds automatically, and that you don't need to specify things like repository location or encryption on the command line.)

You can forget any individual generation. Thanks to the way Obnam treats every generation as an independent snapshot (except it's not really a full backup every time), you don't have to worry about the distinctions between a full and incremental backup.

Forgetting backups manually is tedious, and you probably want to use a schedule to have Obnam automatically pick the generations to forget. A common type of schedule is something like this:

- keep one backup for each day for the past week
- keep one backup for each week for the past three months
- keep one backup for each month for the past two years
- keep one backup for each year for the past fifty seven years

Obnam uses the '--keep' setting to specify a schedule. The setting for the above schedule would look like this:

```
--keep 7d,15w,24m,57y
```

This isn't an exact match, due to the unfortunate ambiguity of how long a month is in weeks, but it's close enough. The setting "7d" is interpreted as "the last backup of each calendar day for the last seven days on which backups were made". Similarly for the other parts of the schedule. See the Obnam configuration files and settings chapter for exact details.

The schedule picks a set of generations to keep. Everything else gets forgotten.

Choosing a schedule for forgetting generations

The schedule for retiring backup generations is a bit of a guessing game, just like backups in general. If you could reliably tell the future, you'd know all the disasters that threaten your data, and you could backup only the things that would otherwise be lost in the future.

In this reality, alas, you have to guess. You need to think about what risks you're facing (or your data is), and how much backup storage space you're willing to spend on protecting against them.

- Are you afraid of your hard drive suddenly failing in a very spectacular way, such as by catching fire or being stolen? If so, you really only need one very recent backup to cover against that.
- Do you worry about your hard drive, or filesystem, or your applicatin programs, or you yourself, slowly corrupting your data over a longer period of time? How long will it take you to find that out? You need a backup history that lasts longer than it takes for you to detect slow corruption.
- Likewise with accidental deletion of files. How long will it take for you to notice? That's how long the backup history should be, at minimum.

There's other criteria as well. For example, would you like to see, in fifty years, how your files are laid out today? If so, you need a fifty-year-old backup, plus perhaps a backup from each year, if you want to compare how things were each year in between. With increasing storage space and nice de-duplication features, this isn't quite as expensive as it might be.

There is no one schedule that fits everyones needs and wants. You have to decide for youself. That's why the default in Obnam is to keep everything forever. It's not Obnam's duty to decide that you should not keep this or that backup generation.

Chapter 8

Verifying backups

It's 9 in the evening. Do you know if your backups work? Do you know when you last made a successful backup of all of your data? Do you know whether you can restore from that backup? If not, how well can you sleep?

You should verify your backups, and do it regularly, not just when you first set up the backup system. Verification means doing whatever you need to do to ensure all of your precious data has been backed up and can be correctly restored from the backups.

The simplest way to do that is to restore all your data, and compare it with your live data, and note any differences. That requires you have enough free disk space to restore everything, but it's almost the only way to be really sure.

It's also a great way to ensure the restoring actually works. If you don't test that, don't expect it'll work when needed.

If you have the disk space to do a complete restore, doing so is a great way to exercise your disaster recovery process in general. Here's one way of doing it:

- On your main computer, do a backup.
- On a second computer, perhaps borrowed for this, restore all your data, without using your main computer at all.
- Start using the restored data as your live data. Do real work, and do all the things you normally do. Pretend your main computer was eaten by your pet shark.
- If you notice something missing, or being corrupt, or being too old, get the files from your main computer, and fix your backup process so that the next time you won't have that problem.

How often should you do that? That, again, depends on how you feel about your data, and how much you trust your backup tools and processes. If it's really important that you can recover from a disaster, you need to verify more frequently. If data loss is merely inconvenient and not life-changingly disastrous, you can verify less often.

In addition to restoring data, Obnam provides two other ways to verify your backups:

- 'obnam verify' is like 'obnam restore', except it compares the backed up data with live data, and reports any differences. This requires you to trust that Obnam does the verification correctly.
- 'obnam mount' lets you access your backed up data as if it were just a directory. You can then use any tool you trust to compare the backed up data with live data. This is very much like doing a restore, since the comparison tool will have to extract all the data and metadata from the backup; it just doesn't write it out.

Both of these approaches have the problem that they compare a backup with live data, and the live data may have changed after the backup was made. You need to verify all differences manually, and if the live data changes frequently, there can be a large number of wrong reports.

Chapter 9

Sharing a repository between multiple clients

Obnam lets you backup several computers to the same repository. Each client is identified by a name, which defaults to the system hostname: the name you get when you run the 'hostname' command. You can also set the name explicitly, using the '--client-name' setting in Obnam.

All the clients sharing a repository share the file content data (the chunks), and can de-duplicate across clients. Each client has its own backup generations, and those are fully independent from other clients. You can, for example, forget any generations you want for one client, and it doesn't affect the generations or any backed up data from any other client.

Obnam takes care of locking automatically so you can run Obnam on each client without having to arrange it so that you only run it on one client at a time.

A caveat of sharing a repository is that any client has access to all chunks, and can delete any other client from the repository. This means you should only share a repository amongst clients in the same security domain: all clients should be trusted equally. If one client gets hacked, then the intruder has access to all the data in the repository, and can delete the backups of all the clients using that repository.

To share a repository amongst clients you need to do the following:

- Set a unique name for each client. It needs to be unique within the repository.
- Arrange for each client to have access to the repository.

That's all.

To see what clients are using a repository, use this:

obnam clients

There is currently no way to remove a client from a repository, unless you're using encryption. This is to be considered a bug in Obnam, and will be fixed at a future time. After that, a time machine will be developed so that this paragraph will have never existed.

Chapter 10

Using encryption

Obnam allows you to encrypt your backups. This chapter discusses why and how to do that.

You don't admit to being a spy, so isn't encryption unnecessary?

You're not the only one who cares about your data. A variety of governments, corporations, criminals, and overly curious snoopers und lookenpeepers may also be interested. (It's sometimes hard to tell them apart.) They might be interested in it to data in order to find evidence against you, blackmail you, or just curious about what you're talking about with your other friends.

They might be interested in your data from a statistical point of view, and don't particularly care about your specifically. Or they might be interested only in you.

Instead of reading your files and email, or looking at your photos and videos, they might be interested in preventing your access to them, or to destroy your data. They might even want to corrupt your data, perhaps by planting child porn in your photo archive.

You protect your computer as well as you can to prevent these and other bad things from happening. You need to protect your backups with equal care.

If you back up to a USB drive, you should probably make the drive be encrypted. Likewise, if you back up to online storage. There are many forms of encryption, and I'm unqualified to give advice on this, but any of the common, modern ones should suffice except for quite determined attackers.

Instead of, or in addition to, encryption, you could ensure the physical security of your backup storage. Keep the USB drive in a safe, perhaps, or a safe deposit box.

The multiple backups you need to protect yourself against earthquakes, floods, and roving gangs of tricycle-riding clowns, are also useful against attackers. They might corrupt your live data, and the backups at your home, but probably won't be able to touch the USB drive encased in concrete and buried in the ground at a secret place only you know about.

The other side of the coin is that you might want to, or need to, ensure others do have access to your backed up data. For example, if the clown gang kidnaps you, your spouse might need access to you backups to be able to contact your MI6 handler to ask them to rescue you. Arranging safe access to (some) backups is an interesting problem to which there are various solutions. You could give your spouse the encryption passphrase, or give the passphrase to a trusted friend or your lawyer. You could also use something like libgfshare to escrow encryption keys more safely.

How Obnam encryption works

An Obnam repository contains several directories, for different types of data.

- A per-client directory for each client, for data that is only relevant to that client, such as the generations to that client.
- A directory for the list of clients.
- A directory for all the chunks of file content data, plus additional directories used for de-duplicating chunks.

The per-client directory is encrypted so that only that client can access it. This means that only the client itself can see its generations, and the files in each generation.

The shared directories (client list, chunks) is encrypted so that all clients can use them. This allows clients to share chunks, so that de-duplication works across all clients.

This encryption scheme assumes that all clients sharing a repository trust each other, and that it's OK for them to be able to read all the chunk data they want. The encryption does not protect siblings from reading each others e-mail from the backup repository, for example, but it does protect them against their parents, if the parents don't have a suitable encryption key.

In addition to the encryptions for client you can add additional keys. These keys can also access the backup repository. For example, the parents' key might be added to the repository so that if need be, they could restore any child's data, even if the child had lost their own encryption key.

In a corporate setting, the backup administrator key might be added so that the administrator can, for example, verify the integrity of the repository, or to access data of an employee who has won the lottery and isn't currently available due to bad Internet access to the Moon.

Such additional keys can be added either for any one client, or to all clients.

Setting up Obnam to use encryption

Obnam uses PGP keys, specifically the GNU Privacy Guard (GnuPG, gpg) implementation of them. To use encrypted backups, you need to first create a PGP key pair for yourself. See the GnuPG documentation for instructions.

Once you have a working GnuPG setup and a key pair (consisting of a public key and a secret key), you need to find the key identifier for them. Run the following command and pick your key from the list.

```
gpg --list-keys
```

The output will look something like this:

```
pub 4096R/5E8511F9 2009-07-22
uid Lars Wirzenius <liw@liw.fi>
sub 2048R/9BE35AE6 2011-08-05
```

That's the output for one key; there may be many keys. The key identifier is on the line staring with 'pub', in the second column after the slash. Above, it's 5E8511F9.

In the rest of the examples in this chapter, we'll assume your key identifier is CAFEFACE.

To set up encryption, use the '--encrypt-with' setting:

```
[config]
encrypt-with = CAFEFACE
```

That's all.

Note that a repository should be fully encrypted or not encrypted at all, and that you can't switch afterwards. If you change your mind about whether to use encryption at all, you'll need to start a new repository. All clients sharing a repository need to be using encryption, or else none of them may use encryption. If you mix encryption or cleartext backups, the error messages may prove to be confusing.

Obnam will automatically encrypt all the files it writes to the backup repository, and de-crypt them when needed. As long as you only have one encryption key for each client, and don't add additional keys, Obnam will take care of adding the right keys to the right places automatically.

Checking if a repository uses encryption

There is no direct way with Obnam to check if a repository uses encryption. However, you can check that manually: if your repository contains the file ‘clientlist/key‘, the repository is encrypted.

Managing encryption keys in a repository

This section discusses how to manage encryption keys in a repository, how to add additional keys for each toplevel, and how to change keys for a client. It also shows how to check what keys are being used, and what access each key has.

Chapter 11

Other stuff

This chapter discusses topics that do not warrant a chapter of their own, such as compressing backups and running Obnam from cron.

k4dirstat cache files

"k4dirstat" is a utility for visualising the disk space used by a directory tree. Obnam's "kdirstat" command can be used to produce a listing of the contents of a generation in a format which can be read by k4dirstat using "File", "Read Cache File" from the k4dirstat menu. e.g.

obnam kdirstat --client CLIENT --generation GENID > CLIENT.kdirstat.cache

gzip -v9 CLIENT.kdirstat.cache # OPTIONAL

"CLIENT.kdirstat.cache[.gz]" can now be read by "k4dirstat".

https://bitbucket.org/jeromerobert/k4dirstat/wiki/Home

Chapter 12

Case studies

This chapter goes through, in some detail, some typical use cases for backups. For case, it discusses the data being backed up, and explains choices of backup strategy, storage, etc. Some case studies:

- Single laptop user, typical data of documents, photos, music, backing up to a USB hard drive.
- A VPS or similar server, with web pages, e-mail, and maybe other data, backed up to another server.
- A small company with a number of laptops and desktops, a local file server, a rented co-lo server, backing up to a rented server in a co-lo and to a rotated set of large USB drives.
- Restoring from a complete disaster, where all local computers and storage media are destroyed, but there is an off-site backup that is intact.

Chapter 13

Troubleshooting

This chapter discusses how to debug problems with Obnam. It covers things such as log files, various levels of logging and tracing, and common problems with Obnam use. It also explains what things go where in an Obnam backup repository.

Turning on full logging

Obnam can write a log file. There are several options controlling that. Knowing these can help get out the most information when there's a problem that needs to be investigated.

- '–log=obnam.log' tells Obnam where to log. The log is a simple text file.
- '–log-level=debug' tells Obnam to log at the most detailed level. The default level is 'info', which excludes most debug information.
- '–trace=obnamlib –trace=larch' tells Obnam to log additional debug information. The two arguments match all filenames in Obnam and the Larch library Obnam uses. This additional information is mostly useful to someone who can read and understand the program source code.

Note that these settings can make log files be quite large, in the order of tens of megabytes. The size depends on how many files and how much data your live data has.

Reporting problems ("bugs")

If you have a problem with Obnam, and you want to report it (please do!), including the following information is helpful and makes it easier to figure out what the problem is.

- You should report problems to the ‘obnam-support@obnam.org‘ mailing list. This is a publically archived mailing list where various people help others use Obnam.
- What is the problem? What did you try to achieve? What actually happened?
- The version of Obnam and Larch you’re using, and how you installed it.
  - On Debian, run ‘dpkg -l obnam python-larch‘ on the command line and include the output.
- The exact command line you used. Copy-paste it instead of typing it again into the mail. Sometimes the problem can be hidden if you don’t copy the command line exactly. Also, copying by typing is boring, and we should avoid boring things in life.
- If there’s an error message, copy-paste that into the mail.
- The output of ‘obnam –dump-config‘, which includes the full configuration. Include it as an attachment to your mail to ‘obnam-support‘. If you have some secret information, such as filenames or hostnames, you can replace those with XXXX.
- If you can reproduce the problem while running with ‘–log-level=debug‘, ‘–log=obnam.log‘ and ‘–trace=obnamlib –trace=larch‘ options, include a suitable amount from the end of the log file. The suitable amount may depend on the situation, but if you give the last two hundred lines, and it’s not enough, we’ll ask for more. Again, feel free to replace any sensitive filenames, etc, with XXXX.
- The output of the ‘env‘ command, in the same terminal window in which you ran Obnam. (Again, as an attachment.)
- If your bug is about performance, please run Obnam under profiling, and attach the profiling file. To run Obnam under profiling, install the Python profile (‘python-profiler‘ package in Debian/Ubuntu), and set the OBNAM_PROFILE environment variable to the name of the file with the profiling output (that’s the file you should send by mail). For example:
  OBNAM_PROFILE=obnam.prof obnam backup
  would run the backup under the profiler, and write the result to ‘obnam.prof‘.

Thank you for your help in making Obnam better.

Chapter 14

Obnam configuration files and settings

This chapter discusses Obnam configuration files: where they are, what they contain, and how they are used.

Where is my configuration?

Obnam looks for its configuration files in a number of places:

- /etc/obnam.conf
- /etc/obnam/*.conf
- ~/.obnam.conf
- ~/.config/obnam/*.conf

Note that in '/etc/obnam' and '~/.config/obnam', all files that have a '.conf' suffix are loaded, in "asciibetical" order, which is like alphabetical, but based on character codes rather than what humans understand, but unlike alphabetical isn't dependent on the language being used.

Any files in the list above may or may not exist. If it exists, it is read, and then the next file is read. A setting in one file can be overridden by a later file, if it is set there as well. For example, '/etc/obnam.conf' might set 'log-level' to 'INFO', but '~/.obnam.conf' may then set it to 'DEBUG', if a user wants more detailed log files.

The Obnam configuration files in '/etc' apply to everyone who runs Obnam on that machine. This is important: they are not just for when 'root' runs Obnam.

If you want to have several Obnam configurations, for example for different backup repositories, you need to name or place the files so they aren't on the list above. For example:

- /etc/obnam/system-backup.profile
- ~/.config/obnam/online.profile
- ~/.config/obnam/usbdrive.profile

You would then need to specify that file for Obnam to use it:

```
obnam --config ~/.config/obnam/usbdrive.profile
```

If you want to not be affected by any configuration files, except the ones you specify explicitly, you need to also use the '--no-default-config' option:

```
obnam --no-default-config --config ~/.obnam-is-fun.conf
```

Command line options override values from configuration files.

Configuration file syntax

Obnam configuration files use the INI file syntax, specifically the variant implemented by the Python ConfigParser library. They look like this:

```
[config]
log-level = debug
log = /var/log/obnam.log
encrypt-with = CAFEBEEF
root = /
one-file-system = yes
```

Names of configuration variables are the same as the corresponding command line options. If '--foo' is the option, then the variable in the file is 'foo'. Any command line option '--foo=bar' can be used in a configuration file as 'foo = bar'. There's are exceptions to this ('--no-default-config', '--config', '--help', and a few others), but they're all things you wouldn't put in a configuration file anyway.

Every option, or setting, has a type. Mostly this doesn't matter, unless you give it a value that isn't suitable. The two important exceptions to this are:

- Boolean or yes/no or on/off settings. For example, '--exclude-caches' is a setting that is either turned on (when the option is used) or off (when it's not used). For every boolean setting '--foo', there is an option '--no-foo'. In a configuration files, 'foo' is turned on by setting it to 'yes' or 'true', and off by setting it to 'no' or 'false'.
- Some settings can be lists of values, such as '--exclude'. You can use '--exclude' as many times as you want, each time a new exclusion pattern is added, rather than replacing the previous patterns. In a configuration file, you would write all the values at once, separated by commas and optional spaces: for example, 'exclude = foo, bar, baz'. In a configuration file, the previous list of values is replaced entirely rather than added to.

For a more detailed explanation of Obnam configuration file syntax, see the cliapp (5) manual page on your system, or cliapp man page on the WWW.

Checking what my configuration is

Obnam can read configuration files from a number of places, and it can be tricky to figure out what the actual configuration is. The --dump-config option helps here.

obnam --config ~/.obnam.fun --exclude-caches --dump-config

The option will tell Obnam to write out (to the standard output) a configuration file that captures every setting, and reporting the value that it would have if '--dump-config' werent used.

This is a good way to see what the current settings are and also as a starting point if you want to make a configuration file from scratch.

Finding out all the configuration settings

These can be found in The help file and The 'man' page.

Chapter 15

The backup repository internals

This chapter describes what the Obnam backup repository looks like. Unless you're interested in that, you can skip that entirely.

For now, look at the Obnam website at http://obnam.org/development/.

Repository file permissions

Obnam sets the permissions of all files it creates in the repository such that only the owner of the files can read or write them. (Technically, 0600 for files, 0700 for directories.)

This is to prevent backups from leaking because someone else has read access to the repository. There is no setting in Obnam to control this.

Chapter 16

Obnam options

Obnam command line syntax consists of a **command** possibly followed by arguments. The 'arguments' are listed below.

--version

Show the program's version number and then exits.

-h

Shows the help message and then exits.

--help

Shows the help message and then exits.

--output=FILE

Write the output to FILE, instead of standard output.

-r REPOSITORY

The name of the backup repository.

--repository=REPOSITORY

The name of the backup repository.

--client-name=CLIENT-NAME

The name of the client (e.g. london).

--trace=TRACE

Add to the filename patters for which trace debugging, logging happens.

--quiet

Be silent.

--no-quiet

--verbose

Be verbose.

--no-verbose

--pretend

Does not actually change anything, just a "dummy" run. Can be useful for testing things out before you go live.

--dry-run

Does not actually change anything, just a "dummy" run. Can be useful for testing things out before you go live.

--no-act

Does not actually change anything (works with backup, forget and restore only, and may only simulate approximately real behaviour).

--no-pretend

--no-dry-run

--no-no-act

--lock-timeout=TIMEOUT

When locking in the backup repository, wait TIMEOUT seconds for an existing lock to go away before giving up.

--compress-with=PROGRAM

Use PROGRAM to compress repository with (one of none, deflate).

--root=ROOT

What to backup.

--testing-fail-matching=REGEXP

Development testing helper: simulate failures during backup for files that match the given regular expressions.

--warn-age=AGE

For nagios-last-backup-age: maximum age (by default in hours) for the most recent backup before status is warning. Accepts one character unit specifier (s,m,h,d for seconds, minutes, hours, and days).

--critical-age=AGE

For nagios-last-backup-age: maximum age (by default in hours) for the most recent backup before statis is critical. Accepts one char unit specifier (s,m,h,d for seconds, minutes, hours, and days.)

--to=TO

Where to restore to.

--generation=GENERATION

Which generation to restore.

--keep=KEEP

Your policy for what generations to keep when forgetting.

--verify-randomly=N

Verify N files randomly from the backup (default is zero, meaning everything).

Chapter 17

Backing up

--exclude=EXCLUDE

Regular expression for pathnames to exclude from a backup (can be used multiple times).

--exclude-caches

Exclude directories (and their subdirs) that contain a CACHEDIR.TAG file.

--no-exclude-caches

--one-file-system

Exclude directories (and their subdirs) that are in a different filesystem

--no-one-file-system

--checkpoint=SIZE

Make a checkpoint after a given SIZE (default is 1073741824).

--de-duplicate=MODE

Find duplicate data in backed up data and store it only once; three modes are available: never de-duplicate, verify that no hash collisions happen, or (the default) fatalistically accept the risk of collisions.

--leave-checkpoints

Leave checkpoint generations at the end of a successful backup run.

--no-leave-checkpoints

--small-files-in-btree

Put the contents of small files directly into the per-client B-tree, instead of separate chunk files; do not use this as it is quite bad for performance.

--no-small-files-in-btree

Chapter 18

Encryption

--encrypt-with=ENCRYPT-WITH

PGP key with which to encrypt data in the backup repository.

--keyid=KEYID

PGP key id to add to/remove from the backup repository.

--weak-random

Use /dev/urandom instead of /dev/random to generate symmetric keys.

--no-weak-random

--symmetric-key-bits=SYMMETRIC-KEY-BITS

The size of symmetric key, in bits.

Chapter 19

Integrity checking (fsck)

--fsck-fix

Should fsck try to fix problems?

--no-fsck-fix

--fsck-ignore-chunks

Ignore chunks when checking repository integrity, (assume all chunks exist and are correct).

--no-fsck-ignore-chunks

--fsck-ignore-client=NAME

Do not check the repository data for cient NAME.

--fsck-last-generation-only

Check only the last generation for each client.

--no-fsck-last-generation-only

--fsck-skip-generations

Do not check any generations.

--no-fsck-skip-generations

--fsck-skip-dirs

Do not check anything about directories and their files.

--no-fsck-skip-dirs

--fsck-skip-files

Do not check anything about files.

--no-fsck-skip-files

--fsck-skip-per-client-b-trees

Do not check per-client B-trees.

--no-fsck-skip-per-client-b-trees

--fsck-skip-shared-b-trees

Do not check shared B-trees.

--no-fsck-skip-shared-b-trees

Chapter 20

Logging

--log=FILE

Write log entries to FILE (default is to not write log files at all); use "syslog" to log to system log, or "none" to disable logging.

--log-level=LEVEL

Log at LEVEL, one of debug, info, warning, error, critical, fatal (default: info).

--log-max=SIZE

Rotate logs larger than SIZE, zero for never (default: 0).

--log-keep=N

Keep the last N logs (default is 10).

--log-mode=MODE

Set permissions of new log files to MODE (octal; default 0600).

Chapter 21

Mounting with FUSE

--viewmode=MODE

"Single" directly mount specified generation, or "Multiple" mount all generations as separate directories.

--fuse-opt=FUSE

Options to pass directly to Fuse.

Chapter 22

Performance

--dump-memory-profile=METHOD

Make memory profiling dumps using METHOD, which is one of: none, simple, meliae, or heapy (default: simple).

--memory-dump-interval=SECONDS

Make memory profiling dumps at least SECONDS apart.

Chapter 23

Performance tweaking

--node-size=SIZE

Size of B-tree nodes on disk; only affects new B-trees so you may need to delete a client or repository to change this for existing repositories (default: 262144).

--chunk-size=SIZE

Size of chunks of file data backed up (default: 1048576).

--upload-queue-size=SIZE

Length of upload queue for B-tree nodes (default: 128).

--lru-size=SIZE

Size of LRU cache for B-tree nodes (default: 256).

--idpath-depth=IDPATH-DEPTH

Depth of chunk id mapping.

--idpath-bits=IDPATH-BITS

Chunk id level size.

--idpath-skip=IDPATH-SKIP

Chunk id mapping lowest bits skip.

--chunkids-per-group=NUM

Encode NUM chunk ids per group (default: 1024).

Chapter 24

SSH/SFTP

--ssh-key=FILENAME

Use FILENAME as the ssh RSA private key for sftp access (default is using keys known to ssh-agent).

--strict-ssh-host-keys

Require that the ssh host key must be known and correct to be accepted; default is to accept unknown keys.

--no-strict-ssh-host-keys

--ssh-known-hosts=FILENAME

Filename of the user's known hosts file (default: /root/.ssh/known_hosts).

--pure-paramiko

Do not use openssh even if available, use paramiko only instead.

--no-pure-paramiko

Chapter 25

Performance tuning

This chapter discusses ways to tune Obnam performance for various situations. It covers the various options that can affect CPU and memory consumptions, as well as ways to experiment to find a good set of values.

Introduction

Obnam has a number of options for performance tuning. See the manual page for all the details. Below is an adapted excerpt of e-mails written by Lionel Bouton of how to test various values to find a good set for your situation. See the list archive for the e-mails: first and second.

Measurements

Tuning **lru-size** and/or **upload-queue-size** can make a significant difference in performance.

Here follows some test results for this situation:

- Data to backup stored on a btrfs volume on SSD: 155000 files, 3.66GiB.
- Local system: 64 bit Linux, Python 2.7.5, OpenSSH 6.6p1 with hpn patches.
- Local CPU: Intel(R) Core(TM) i5-3317U CPU @ 1.70GHz (mostly idle).
- Remote system: 64bit Linux, OpenSSH 6.6p1 with hpn patches, repository data on ext4 on standard SATA 7200rpm disk, large memory (everything should fit in memory, only writes should hit disk).
- Very minimal changes in the data backed up during tests so successive backups only check for differences and backup transfers nearly no content.
- Backup over WiFi (1ms RTT, max speed over sftp 3MB/s).

I use this command line without any configuration file:

obnam -r sftp://obnam@SERVER/~/repo --compress-with=
deflate --client-name=CLIENT backup DIR

During testing I added

--lru-size <l> --upload-queue-size <q>

with different <l> and <q> values.

The resident memory of the Obnam process grows steadily (probably filling caches) until it hits a pretty stable ceiling (cache full or nothing new to put in cache) during the backup. It raises again rapidly at the very end (during commits/unlock/...). The value reported below is obtained either through the RES column reported by the htop utility or the RSS column reported by "ps aux" and is the max witnessed near the end of the backup.

Each combination was tested at least twice unless it was considered not interesting after the first run. Timing seems consistent enough given the systems involved (the system hosting the repository is often busy) and memory usage is very consistent across runs.

Default values as fetched from __init__.py are: l=256, q=128.

Conditions	**Time**	**Memory**	**Number of runs**
default values	22m21s - 24m51s	~260M	2
l=10000, q=default	13m45s - 15m03s	~332M	2
l=default, q=250	08m23s - 10m29s	~278M	5
l=default, q=350	02m42s - 02m49s	272-276M	2
l=default, q=400	02m13s - 02m18s	268-272M	3
l=default, q=500	02m10s - 02m16s	267-272M	3
l=default, q=512	02m13s - 02m14s	265-269M	2
l=512, q=512	01m55s - 02m06s	322-326M	3
l=768, q=512	01m55s - 01m58s	397-418M	3
l=1024, q=512	01m53s - 01m55s	403-418M	3
l=2048, q=512	01m55s - 01m59s	408-410M	3
l=4096, q=512	~01m58s	~419M	1
l=default, q=600	02m14s - 02m26s	269-272M	4
l=default, q=750	02m13s - 02m15s	266-272M	2
l=default, q=1000	02m19s - 02m20s	~266M	2
l=default, q=10000	02m23s - 02m35s	~266M	2

So in my configuration, when nearly no data changes between backups, **--lru-size=1024 --upload-queue-size=512** is at least 11x faster than the default configuration.

Discussion

--upload-queue-size seems to have the greatest effect without any adverse effect (memory usage remains at the same level). For a little extra boost with a small impact on memory usage, I can increase **--lru-size** to 1024.

Note that Obnam was using 100% of the CPU for most of the time in the fastest configuration, replacing --verbose with --quiet didn't change the running time.

Please note that the ideal settings for my backup configuration might differ from the ones for yours. You might get even better results after tuning of your own.

These parameters have a nice behaviour for tuning: **upload-queue-size** doesn't seem to have much drawback if at all when being increased (it begins to give signs of slowing down obnam at 10000 here but it might be the performance variance inherent in my configuration) and increasing **lru-size** only increases memory usage a bit without slowing things noticeably after reaching the ideal spot.

A good rule of thumb seems to try increasing one of these parameters by 2x or 4x and keep going at it until performance stops increasing.

Running Obnam under the Python profiler

A **profiler** is a program that measures where another program spends its time. This can be very useful for finding out why the other program is slow.

Obnam can easily be run under the Python profiler. You need to have the profiler installed. Check with your operating system or Python installation how to achieve that. To see if you have it installed, run the following command on the command line:

```
python -c 'import cProfile'
```

If this outputs nothing, all is well. If it outputs an error such as the following, you have not got the profiler installed:

```
Traceback (most recent call last):
File "<string>", line 1, in <module>
ImportError: No module named cProfiler
```

Once you have the profiler installed, run Obnam like this:

```
OBNAM_PROFILE=backup.prof obnam bsckup
```

This will cause the profiling data to be written to the file 'backup.prof'. You can do this for any Obnam command, and write it to any file.

The profiling data is in binary form. Obnam comes with a little helper program to transform it to a human-readable form:

```
obnam-viewprof backup.prof | less
```

If you run the above command, you'll see that the humans to whom this is readable are programmers and circus clowns. If you can understand the output, great! If not, it's still useful to send that to the Obnam developers to report a performance problem.

Chapter 26

The inbuilt help

The help file

If you run 'obnam –help' this will be displayed - Usage: obnam [options] add-key [CLIENT-NAME]...
obnam [options] backup [DIRECTORY | URL]...
obnam [options] client-keys
obnam [options] clients
obnam [options] diff [GENERATION1] GENERATION2
obnam [options] dump-repo
obnam [options] force-lock
obnam [options] forget [GENERATION]...
obnam [options] fsck
obnam [options] generations
obnam [options] genids
obnam [options] help
obnam [options] help-all
obnam [options] kdirstat [FILE]...
obnam [options] list-keys
obnam [options] list-toplevels obnam [options] ls [FILE]...
obnam [options] mount [ROOT]
obnam [options] nagios-last-backup-age
obnam [options] remove-client [CLIENT-NAME]...
obnam [options] remove-key [CLIENT-NAME]...
obnam [options] restore [DIRECTORY]...
obnam [options] verify [DIRECTORY]...
* obnam add-key: Add a key to the repository.
* obnam backup: Backup data to repository.
* obnam client-keys: List clients and their keys in the repository.
* obnam clients: List clients using the repository.
* obnam diff: Show difference between two generations.
* obnam dump-repo: Dump (some) data structures from a repository.
* obnam force-lock: Force a locked repository to be open.

* obnam forget: Forget (remove) specified backup generations.
* obnam fsck: Verify internal consistency of backup repository.
* obnam generations: List backup generations for client.
* obnam genids: List generation ids for client.
* obnam help: Print help.
* obnam help-all: Print help, including hidden subcommands.
* obnam kdirstat: List contents of a generation in kdirstat cache format.
* obnam list-keys: List keys and the repository toplevels they're used in.
* obnam list-toplevels: List repository toplevel directories and their keys.
* obnam ls: List contents of a generation.
* obnam mount: Mount a backup repository as a FUSE filesystem.
* obnam nagios-last-backup-age: Check if the most recent generation is recent enough.
* obnam remove-client: Remove client and its key from repository.
* obnam remove-key: Remove a key from the repository.
* obnam restore: Restore some or all files from a generation.
* obnam verify: Verify that live data and backed up data match.

Options:

--version	show program's version number and exit
-h, --help	show this help message and exit
--generate-manpage=TEMPLATE	fill in manual page TEMPLATE
--output=FILE	write output to FILE, instead of standard output
-r URL, --repository=URL	name of backup repository (can be pathname or supported URL)
--client-name=CLIENT-NAME	name of client (defaults to hostname)
--quiet, --silent	be silent: show only error messages, no progress updates
--no-quiet, --no-silent	
--verbose	be verbose: tell the user more of what is going on and generally make sure the user is aware of what is happening or at least that something is happening and also make sure their screen is getting updates frequently and that there is changes happening all the time so they do not get bored and that they in fact get frustrated by getting distracted by so many updates that they will move into the Gobi desert to live under a rock
--no-verbose	
--pretend, --dry-run, --no-act	do not actually change anything (works with backup, forget and restore only, and may only simulate approximately real behavior)
--no-pretend, --no-dry-run, --no-no-act	

--lock-timeout=TIMEOUT	when locking in the backup repository, wait TIMEOUT seconds for an existing lock to go away before giving up
--repository-format=FORMAT	what format to use for new repositories? one of "6", "simple"
--compress-with=PROGRAM	use PROGRAM to compress repository with (one of none, deflate)
--dump-repo-file-metadata	dump metadata about files?
--no-dump-repo-file-metadata	
--warn-age=AGE	for nagios-last-backup-age: maximum age (by default in hours) for the most recent backup before status is warning. Accepts one char unit specifier (s,m,h,d for seconds, minutes, hours, and days.
--critical-age=AGE	for nagios-last-backup-age: maximum age (by default in hours) for the most recent backup before statis is critical. Accepts one char unit specifier (s,m,h,d for seconds, minutes, hours, and days.
--to=TO	where to restore or FUSE mount
--generation=GENERATION	which generation to restore
--always-restore-setuid	restore setuid/setgid bits in restored files, even if not root or backed up file had different owner than user running restore
--no-always-restore-setuid	
--keep=KEEP	policy for what generations to keep when forgetting
--verify-randomly=N	verify N files randomly from the backup (default is zero, meaning everything)

Backing up:

--root=URL	what to backup
--exclude=EXCLUDE	regular expression for pathnames to exclude from backup (can be used multiple times)
--exclude-from=FILE	read exclude patterns from FILE
--exclude-caches	exclude directories (and their subdirs) that contain a CACHEDIR.TAG file (see http://www.brynosaurus.com/cachedir/spec.html for what it needs to contain, and http://liw.fi/cachedir/ for a helper tool)
--no-exclude-caches	
--include=INCLUDE	regular expression for pathnames to include from backup even if it matches an exclude rule (can be used multiple times)
--one-file-system	exclude directories (and their subdirs) that are in a different filesystem
--no-one-file-system	

--checkpoint=SIZE	make a checkpoint after a given SIZE
--deduplicate=MODE	find duplicate data in backed up data and store it only once; three modes are available: never de- duplicate, verify that no hash collisions happen, or (the default) fatalistically accept the risk of collisions
--leave-checkpoints	leave checkpoint generations at the end of a successful backup run
--no-leave-checkpoints	
--small-files-in-btree	put contents of small files directly into the per- client B-tree, instead of separate chunk files; do not use this as it is quite bad for performance
--no-small-files-in-btree	

Configuration files and settings:

--dump-setting-names	write out all names of settings and quit
--dump-config	write out the entire current configuration
--no-default-configs	clear list of configuration files to read
--config=FILE	add FILE to config files
--list-config-files	list all possible config files
--help-all	show all options

Development of Obnam itself:

--trace=TRACE	add to filename patters for which trace debugging logging happens
--pretend-time=TIMESTAMP	pretend it is TIMESTAMP (YYYY-MM-DD HH:MM:SS); this is only useful for testing purposes
--crash-limit=COUNTER	artificially crash the program after COUNTER files written to the repository; this is useful for crash testing the application, and should not be enabled for real use; set to 0 to disable (disabled by default)
--sftp-delay=SFTP-DELAY	add an artificial delay (in milliseconds) to all SFTP transfers
--testing-fail-matching=REGEXP	development testing helper: simulate failures during backup for files that match the given regular expressions

Encryption:

--encrypt-with=ENCRYPT-WITH	PGP key with which to encrypt data in the backup repository
--keyid=KEYID	PGP key id to add to/remove from the backup repository
--weak-random	use /dev/urandom instead of /dev/random to generate symmetric keys
--no-weak-random	

--key-details	show additional user IDs for all keys
--no-key-details	
--symmetric-key-bits=BITS	size of symmetric key, in bits

Integrity checking (fsck):

--fsck-fix	should fsck try to fix problems? Implies --fsck-rm-unused
--no-fsck-fix	
--fsck-rm-unused	should fsck remove unused chunks?
--no-fsck-rm-unused	
--fsck-ignore-chunks	ignore chunks when checking repository integrity (assume all chunks exist and are correct)
--no-fsck-ignore-chunks	
--fsck-ignore-client=NAME	do not check repository data for client NAME
--fsck-last-generation-only	check only the last generation for each client
--no-fsck-last-generation-only	
--fsck-skip-generations	do not check any generations
--no-fsck-skip-generations	
--fsck-skip-dirs	do not check anything about directories and their files
--no-fsck-skip-dirs	
--fsck-skip-files	do not check anything about files
--no-fsck-skip-files	
--fsck-skip-per-client-b-trees	do not check per-client B-trees
--no-fsck-skip-per-client-b-trees	
--fsck-skip-shared-b-trees	do not check shared B-trees
--no-fsck-skip-shared-b-trees	

Logging:

--log=FILE	write log entries to FILE (default is to not write log files at all); use "syslog" to log to system log, or "none" to disable logging
--log-level=LEVEL	log at LEVEL, one of debug, info, warning, error, critical, fatal (default: info)
--log-max=SIZE	rotate logs larger than SIZE, zero for never (default: 25000000)
--log-keep=N	keep last N logs (5)
--log-mode=MODE	set permissions of new log files to MODE (octal; default 0600)

Mounting with FUSE:

--fuse-opt=FUSE	options to pass directly to Fuse

Peformance:

--dump-memory-profile=METHOD	make memory profiling dumps using METHOD, which is one of: none, simple, meliae, or heapy (default: simple)
--memory-dump-interval=SECONDS	make memory profiling dumps at least SECONDS apart

Performance tweaking:

--node-size=SIZE	size of B-tree nodes on disk; only affects new B-trees so you may need to delete a client or repository to change this for existing repositories
--chunk-size=SIZE	size of chunks of file data backed up
--upload-queue-size=SIZE	length of upload queue for B-tree nodes
--lru-size=SIZE	size of LRU cache for B-tree nodes
--idpath-depth=IDPATH-DEPTH	depth of chunk id mapping
--idpath-bits=IDPATH-BITS	chunk id level size
--idpath-skip=IDPATH-SKIP	chunk id mapping lowest bits skip
--chunkids-per-group=NUM	encode NUM chunk ids per group

SSH/SFTP:

--ssh-key=FILENAME	use FILENAME as the ssh RSA private key for sftp access (default is using keys known to ssh-agent)
--strict-ssh-host-keys	DEPRECATED, use --ssh-host-keys-check instead
--no-strict-ssh-host-keys	
--ssh-host-keys-check=VALUE	If "yes", require that the ssh host key must be known and correct to be accepted. If "no", do not require that. If "ask", the user is interactively asked to accept new hosts. The default ("ssh-config") is to rely on the settings of the underlying SSH client
--ssh-known-hosts=FILENAME	filename of the user's known hosts file
--ssh-command=EXECUTABLE	alternative executable to be used instead of "ssh" (full path is allowed, no arguments may be added)
--pure-paramiko	do not use openssh even if available, use paramiko only instead
--no-pure-paramiko	

The 'man' page

OBNAM(1) - General Commands Manual - OBNAM(1)

NAME - obnam - make, restore, and manipulate backups.

SYNOPSIS - obnam makes, restores, manipulates, and otherwise deals with backups. It can store backups on a local disk or to a server via sftp. Every backup generation looks like a fresh snapshot, but is really incremental: the user does not need to worry whether it's a full backup or not. Only changed data is backed up, and if a chunk of data is already backed up in another file, that data is re-used.

The place where backed up data is placed is called the backup repository. A repository may be, for example, a directory on an sftp server, or a directory on a USB hard disk. A single repository may contain backups from several clients. Their data will intermingle as if they were using separate repositories, but if one client backs up a file, the others may re-use the data.

obnam command line syntax consists of a command possibly followed by arguments. The commands are listed below.

- **backup** makes a new backup. The first time it is run, it makes a full backup, after that an incremental one.
- **restore** is the opposite of a backup. It copies backed up data from the backup repository to a target directory. You can restore everything in a generation, or just selected files.
- **clients** lists the clients that are backed up to the repository.
- **generations** lists every backup generation for a given client, plus some metadata about the generation.
- **genids** lists the identifier for every backup generation for a given client. No other information is shown. This can be useful for scripting.
- **ls** lists the contents of a given generation, similar to ls -lAR.
- **kdirstat** lists the contents of a given generation, in a format which is compatible with the kdirstat cache file format, which can then be used to visualise the contents of a backup.
- **verify** compares data in the backup with actual user data, and makes sure they are identical. It is most useful to run immediately after a backup, to check that it actually worked. It can be run at any time, but if the user data has changed, verification fails even though the backup is OK.
- **forget** removes backup generations that are no longer wanted, so that they don't use disk space. Note that after a backup generation is removed the data can't be restored anymore. You can either specify the generations to remove by listing them on the command line, or use the --keep option to specify a policy for what to keep (everything else will be removed).
- **fsck** checks the internal consistency of the backup repository. It verifies that all clients, generations, directories, files, and all file contents still exists in the backup repository. It may take quite a long time to run.
- **force-lock** removes a lock file for a client in the repository. You should only force a lock if you are sure no-one is accessing that client's data in the repository. A dangling lock might happen, for example, if obnam loses its network connection to the backup repository.

- **client-keys** lists the encryption key associated with each client.
- **list-keys** lists the keys that can access the repository, and which toplevel directories each key can access. Some of the toplevel directories are shared between clients, others are specific to a client.
- **list-toplevels** is like list-keys, but lists toplevels and which keys can access them.
- **add-key** adds an encryption key to the repository. By default, the key is added only to the shared toplevel directories, but it can also be added to specific clients: list the names of the clients on the command line. The key is given with the --keyid option. Whoever has access to the secret key corresponding to the key id can access the backup repository (the shared toplevels plus specified clients).
- **remove-key** removes a key from the shared toplevel directories, plus any clients specified on the command line.
- **nagios-last-backup-age** is a check that exits with non-zero return if a backup age exceeds a certain threshold. It is suitable for use as a check plugin for nagios. Thresholds can be given the --warn-age and --critical-age options.
- **diff** compares two generations and lists files differing between them. Every output line will be prefixed either by a plus sign (+) for files that were added, a minus sign (-) for files that have been removed or an asterisk (*) for files that have changed. If only one generation ID is specified on the command line that generation will be compared with its direct predecessor. If two IDs have been specified, all changes between those two generations will be listed.
- **mount** makes the backup repository available via a read-only FUSE filesystem. Each backup generation is visible as a subdirectory, named after the generation id. This means you can look at backed up data using normal tools, such as your GUI file manager, or command line tools such as ls(1), diff(1), and cp(1). You can't make new backups with the mount subcommand, but you can restore data easily.

You need to have the FUSE utilities and have permission to use FUSE for this to work. The details will vary between operating systems; in Debian, install the package fuse and add yourself to the fuse group (you may need to log out and back in again).

Making backups

When you run a backup, obnam uploads data into the backup repository. The data is divided into chunks, and if a chunk already exists in the backup repository, it is not uploaded again. This allows obnam to deal with files that have been changed or renamed since the previous backup run. It also allows several backup clients to avoid uploading the same data. If, for example, everyone in the office has a copy of the same sales brochures, only one copy needs to be stored in the backup repository.

Every backup run is a generation. In addition, obnam will make checkpoint generations every now and then. These are exactly like normal generations, but are not guaranteed to be a complete snapshot of the live data. If the backup run needs to be aborted in the middle, the next backup run can continue from the latest checkpoint, avoiding the need to start over from scratch.

If one backup run drops a backup root directory, the older generations will still keep it: nothing changes in the old generations just because there is a new one. If the root was dropped by mistake, it can be added back and the next backup run will re-use the existing data in the backup repository, and will only back up the file metadata (filenames, permissions, etc).

Verifying backups

What good is a backup system you cannot rely on? How can you rely on something you cannot test? The obnam verify command checks that data in the backup repository matches actual user data. It retrieves one or more files from the repository and compares them to the user data. This is essentially the same as doing a restore, then comparing restored files with the original files using cmp(1), but easier to use.

By default, verification happens on all files. You can also specify the files to be verified by listing them on the command line. You should specify the full paths to the files, not relative to the current directory.

The output lists files that fail verification for some reason. If you verify everything, it is likely that some files (e.g., parent directories of backup root) may have changed without it being a problem. Note that you will need to specify the whole path to the files or directories to be verified, not relative to the backup root. You still need to specify at least one of the backup roots on the command line or via the --root option so that obnam will find the filesystem, in case it is a remote one.

URL syntax

Whenever obnam accepts a URL, it can be either a local pathname, or an sftp URL. An sftp URL has the following form:

sftp://[user@]domain[:port]/path

where domain is a normal Internet domain name for a server, user is your username on that server, port is an optional port number, and path is a pathname on the server side. Like bzr(1), but unlike the sftp URL standard, the pathname is absolute, unless it starts with / / in which case it is relative to the user's home directory on the server.

See the EXAMPLE section for examples of URLs.

You can use sftp URLs for the repository, or the live data (root), but note that due to limitations in the protocol, and its implementation in the paramiko library, some things will not work very well for accessing live data over sftp. Most importantly, the handling of of hardlinks is rather suboptimal. For live data access, you should not end the URL with / / and should append a dot at the end in this special case.

Generation specifications

When not using the latest generation, you will need to specify which one you need. This will be done with the –generation option, which takes a generation specification as its argument. The specification is either the word latest, meaning the latest generation (also the default), or a number. See the generations command to see what generations are available, and what their numbers are.

Policy for keeping and removing backup generations

The forget command can follow a policy to automatically keep some and remove other backup generations. The policy is set with the --keep=POLICY option.

POLICY is comma-separated list of rules. Each rule consists of a count and a time period. The time periods are h, d, w, m, and y, for hour, day, week, month, and year.

A policy of 30d means to keep the latest backup for each day when a backup was made, and keep the last 30 such backups. Any backup matched by any policy rule is kept, and any backups in between will be removed, as will any backups older than the oldest kept backup.

As an example, assume backups are taken every hour, on the hour: at 00:00, 01:00, 02:00, and so on, until 23:00. If the forget command is run at 23:15, with the above policy, it will keep the backup taken at 23:00 on each day, and remove every other backup that day. It will also remove backups older than 30 days.

If backups are made every other day, at noon, forget would keep the 30 last backups, or 60 days worth of backups, with the above policy.

Note that obnam will only inspect timestamps in the backup repository, and does not care what the actual current time is. This means that if you stop making new backups, the existing ones won't be removed automatically. In essence, obnam pretends the current time is just after the latest backup when forget is run.

The rules can be given in any order, but will be sorted to ascending order of time period before applied. (It is an error to give two rules for the same period.) A backup generation is kept if it matches any rule.

For example, assume the same backup frequency as above, but a policy of 30d,52w. This will keep the newest daily backup for each day for thirty days, and the newest weekly backup for 52 weeks. Because the hourly backups will be removed daily, before they have a chance to get saved by a weekly rule, the effect is that the 23:00 o'clock backup for each day is saved for a month, and the 23:00 backup on Sundays is saved for a year.

If, instead, you use a policy of 72h,30d,52w, obnam would keep the last 72 hourly backups, and the last backup of each calendar day for 30 days, and the last backup of each calendar week for 52 weeks. If the backup frequency was once per day, obnam would keep the backup of each calendar hour for which a backup was made, for 72 such backups. In other words, it would effectively keep the last 72 daily backups.

Sound confusing? Just think how confused the developer was when writing the code.

If no policy is given, forget will keep everything.

A typical policy might be 72h,7d,5w,12m, which means: keep the last 72 hourly backups, the last 7 daily backups, the last 5 weekly backups and the last 12 monthly backups. If the backups are systematically run on an hourly basis, this will mean keeping hourly backups for three days, daily backups for a week, weekly backups for a month, and monthly backups for a year.

The way the policy works is a bit complicated. Run forget with the --pretend option to make sure you're removing the right ones.

Using encryption

obnam can encrypt all the data it writes to the backup repository. It uses gpg(1) to do the encryption. You need to create a key pair using gpg --gen-key (or use an existing one), and then tell obnam about it using the --encrypt-with option.

Configuration files

obnam will look for configuration files in a number of locations. See the FILES section for a list. All these files together are treated as one big file with the contents of all files concatenated.

The files are in INI format, and only the [config] section is used (any other sections are ignored).

The long names of options are used as keys for configuration variables. Any setting that can be set from the command line can be set in a configuration file, in the [config] section.

For example, the options in the following command line:

```
obnam --repository=/backup --exclude='.wav$' backup
```

could be replaced with the following configuration file:

```
[config]
repository: /backup
exclude: .wav$
```

(You can use either foo=value or foo: value syntax in the files.)

The only unusual thing about the files is the way options that can be used many times are expressed. All values are put in a single logical line, separated by commas (and optionally spaces as well). For example:

```
[config]
exclude = foo, bar, .mp3$
```

The above has three values for the exclude option: any files that contain the words foo or bar anywhere in the fully qualified pathname, or files with names ending with a period and mp3 (because the exclusions are regular expressions).

A long logical line can be broken into several physical ones, by starting a new line at white space, and indenting the continuation lines:

```
[config]
exclude = foo,
bar,
.mp3$
```

The above example adds three exclusion patterns.

Multiple clients and locking

obnam supports sharing a repository between multiple clients. The clients can share the file contents (chunks), so that if client A backs up a large file, and client B has the same file, then B does not need to upload the large file to the repository a second time. For this to work without confusion, the clients use a simple locking protocol that allows only one client at a time to modify the shared data structures. Locks do not prevent read-only access at the same time: this allows you to restore while someone else is backing up.

Sometimes a read-only operation happens to access a data structure at the same time as it is being modified. This can result in a crash. It will not result in corrupt data, or incorrect restores. However, you may need to restart the read-only operation after a crash.

OPTIONS

--always-restore-setuid	restore setuid/setgid bits in restored files, even if not root or backed up file had different owner than user running restore
--client-name=CLIENT-NAME	name of client (defaults to hostname)
--compress-with=PROGRAM	use PROGRAM to compress repository with (one of none, deflate)
--critical-age=AGE	for nagios-last-backup-age: maximum age (by default in hours) for the most recent backup before statis is critical. Accepts one char unit specifier (s,m,h,d for seconds, minutes, hours, and days.
--dump-repo-file-metadata	dump metadata about files?
--generate-manpage=TEMPLATE	SUPPRESSHELP
--generation=GENERATION	which generation to restore
-h, --help	show this help message and exit
--keep=KEEP	policy for what generations to keep when forgetting
--lock-timeout=TIMEOUT	when locking in the backup repository, wait TIMEOUT seconds for an existing lock to go away before giving up
--no-always-restore-setuid	
--no-dump-repo-file-metadata	
--no-pretend, --no-dry-run, --no-no-act	
--no-quiet, --no-silent	
--no-verbose	
--output=FILE	write output to FILE, instead of standard output
--pretend, --dry-run, --no-act	do not actually change anything (works with backup, forget and restore only, and may only simulate approximately real behavior)
--quiet, --silent	be silent: show only error messages, no progress updates
-r, --repository=URL	name of backup repository (can be pathname or supported URL)
--repository-format=FORMAT	what format to use for new repositories? one of "6", "simple"
--to=TO	where to restore or FUSE mount

--verbose	be verbose: tell the user more of what is going on and generally make sure the user is aware of what is happening or at least that something is happening and also make sure their screen is getting updates frequently and that there is changes happening all the time so they do not get bored and that they in fact get frustrated by getting distracted by so many updates that they will move into the Gobi desert to live under a rock
--verify-randomly=N	verify N files randomly from the backup (default is zero, meaning everything)
--version	show program's version number and exit
--warn-age=AGE	for nagios-last-backup-age: maximum age (by default in hours) for the most recent backup before status is warning. Accepts one char unit specifier (s,m,h,d for seconds, minutes, hours, and days.

Backing up

--checkpoint=SIZE	make a checkpoint after a given SIZE
--deduplicate=MODE	find duplicate data in backed up data and store it only once; three modes are available: never de-duplicate, verify that no hash collisions happen, or (the default) fatalistically accept the risk of collisions
--exclude=EXCLUDE	regular expression for pathnames to exclude from backup (can be used multiple times)
--exclude-caches	exclude directories (and their subdirs) that contain a CACHEDIR.TAG file (see http://www.brynosaurus.com/cachedir/spec.html for what it needs to contain, and http://liw.fi/cachedir/ for a helper tool)
--exclude-from=FILE	read exclude patterns from FILE
--include=INCLUDE	regular expression for pathnames to include from backup even if it matches an exclude rule (can be used multiple times)
--leave-checkpoints	leave checkpoint generations at the end of a successful backup run
--no-exclude-caches	
--no-leave-checkpoints	
--no-one-file-system	
--no-small-files-in-btree	
--one-file-system	exclude directories (and their subdirs) that are in a different filesystem
--root=URL	what to backup

--small-files-in-btree	put contents of small files directly into the per-client B-tree, instead of separate chunk files; do not use this as it is quite bad for performance

Configuration files and settings

--config=FILE	add FILE to config files
--dump-config	write out the entire current configuration
--dump-setting-names	SUPPRESSHELP
--help-all	show all options
--list-config-files	SUPPRESSHELP
--no-default-configs	clear list of configuration files to read

Development of Obnam itself

--crash-limit=COUNTER	artificially crash the program after COUNTER files written to the repository; this is useful for crash testing the application, and should not be enabled for real use; set to 0 to disable (disabled by default)
--pretend-time=TIMESTAMP	pretend it is TIMESTAMP (YYYY-MM-DD HH:MM:SS); this is only useful for testing purposes
--sftp-delay=SFTP-DELAY	add an artificial delay (in milliseconds) to all SFTP transfers
--testing-fail-matching=REGEXP	development testing helper: simulate failures during backup for files that match the given regular expressions
--trace=TRACE	add to filename patters for which trace debugging logging happens

Encryption

--encrypt-with=ENCRYPT-WITH	PGP key with which to encrypt data in the backup repository
--key-details	show additional user IDs for all keys
--keyid=KEYID	PGP key id to add to/remove from the backup repository
--no-key-details	
--no-weak-random	
--symmetric-key-bits=BITS	size of symmetric key, in bits
--weak-random	use /dev/urandom instead of /dev/random to generate symmetric keys

Integrity checking (fsck)

--fsck-fix	should fsck try to fix problems? Implies --fsck-rm-unused

--fsck-ignore-chunks	ignore chunks when checking repository integrity (assume all chunks exist and are correct)
--fsck-ignore-client=NAME	do not check repository data for client NAME
--fsck-last-generation-only	check only the last generation for each client
--fsck-rm-unused	should fsck remove unused chunks?
--fsck-skip-dirs	do not check anything about directories and their files
--fsck-skip-files	do not check anything about files
--fsck-skip-generations	do not check any generations
--fsck-skip-per-client-b-trees	do not check per-client B-trees
--fsck-skip-shared-b-trees	do not check shared B-trees
--no-fsck-fix	
--no-fsck-ignore-chunks	
--no-fsck-last-generation-only	
--no-fsck-rm-unused	
--no-fsck-skip-dirs	
--no-fsck-skip-files	
--no-fsck-skip-generations	
--no-fsck-skip-per-client-b-trees	
--no-fsck-skip-shared-b-trees	

Logging

--log=FILE	write log entries to FILE (default is to not write log files at all); use "syslog" to log to system log, or "none" to disable logging
--log-keep=N	keep last N logs (10)
--log-level=LEVEL	log at LEVEL, one of debug, info, warning, error, critical, fatal (default: info)
--log-max=SIZE	rotate logs larger than SIZE, zero for never (default: 0)
--log-mode=MODE	set permissions of new log files to MODE (octal; default 0600)

Mounting with FUSE

--fuse-opt=FUSE	options to pass directly to Fuse

Peformance

--dump-memory-profile=METHOD	make memory profiling dumps using METHOD, which is one of: none, simple, meliae, or heapy (default: simple)
--memory-dump-interval=SECONDS	make memory profiling dumps at least SECONDS apart

Performance tweaking

--chunk-size=SIZE	size of chunks of file data backed up

--chunkids-per-group=NUM	encode NUM chunk ids per group
--idpath-bits=IDPATH-BITS	chunk id level size
--idpath-depth=IDPATH-DEPTH	depth of chunk id mapping
--idpath-skip=IDPATH-SKIP	chunk id mapping lowest bits skip
--lru-size=SIZE	size of LRU cache for B-tree nodes
--node-size=SIZE	size of B-tree nodes on disk; only affects new B-trees so you may need to delete a client or repository to change this for existing repositories
--upload-queue-size=SIZE	length of upload queue for B-tree nodes

SSH/SFTP

--no-pure-paramiko	
--no-strict-ssh-host-keys	
--pure-paramiko	do not use openssh even if available, use paramiko only instead
--ssh-command=EXECUTABLE	alternative executable to be used instead of "ssh" (full path is allowed, no arguments may be added)
--ssh-host-keys-check=VALUE	If "yes", require that the ssh host key must be known and correct to be accepted. If "no", do not require that. If "ask", the user is interactively asked to accept new hosts. The default ("ssh-config") is to rely on the settings of the underlying SSH client
--ssh-key=FILENAME	use FILENAME as the ssh RSA private key for sftp access (default is using keys known to ssh-agent)
--ssh-known-hosts=FILENAME	filename of the user's known hosts file
--strict-ssh-host-keys	DEPRECATED, use –ssh-host-keys-check instead
Option values	The SIZE value in options mentioned above specifies a size in bytes, with optional suffixes to indicate kilobytes (k), kibibytes (Ki), megabytes (M), mebibyts (Mi), gigabytes (G), gibibytes (Gi), terabytes (T), tibibytes (Ti). The suffixes are case-insensitive.

EXIT STATUS

obnam will exit with zero if everything went well, and non-zero otherwise.

ENVIRONMENT

obnam will pass on the environment it gets from its parent, without modification. It does not obey any unusual environment variables, but it does obey the usual ones when running external programs, creating temporary files, etc.

FILES

/etc/obnam.conf
/etc/obnam/*.conf
/.obnam.conf
/.config/obnam/*.conf

Configuration files for obnam. It is not an error for any or all of the files to not exist.

EXAMPLE

To back up your home directory to a server:

```
obnam backup --repository sftp://your.server/ /backups $HOME
```

To restore your latest backup from the server:

```
obnam restore --repository sftp://your.server/ /backups --to /var/tmp/my.home.dir
```

To restore just one file or directory:

```
obnam restore --repository sftp://your.server/ /backups --to /var/tmp/my.home.dir $HOME/myfile.txt
```

Alternatively, mount the backup repository using the FUSE filesystem (note that the --to option is necessary):

```
mkdir my-repo
obnam mount --repository sftp://your.server/ /backups -to my-repo
cp my-repo/latest/$HOME/myfile.txt
fusermount -u my-repo
```

To check that the backup worked:

```
obnam verify --repository sftp://your.server/ /backups /path/to/file
```

To remove old backups, keeping the newest backup for each day for ten years:

```
obnam forget --repository sftp://your.server/ /backups -keep 3650d
```

To verify that the backup repository is OK:

```
obnam fsck --repository sftp://your.server/ /backups
```

To view the backed up files in the backup repository using FUSE:

```
obnam mount --to my-fuse
ls -lh my-fuse
fusermount -u my-fuse
```

SEE ALSO

obnam comes with a manual in HTML and PDF forms. See /usr/share/doc/obnam if you have Obnam installed system-wide, or in the subdirectory manual in the source tree.

cliapp(5)

Chapter 27

Errors - code and names

By error code

R018FCX - ToplevelIsFileError
R01F56X - RepositorySettingMissingError
R02C17X - HardlinkError
R0B15DX - RepositoryGenerationDoesNotExist
R0BE94X - RepositoryClientNotLocked
R0C79EX - GpgError
R0F22CX - URLSchemeAlreadyRegisteredError
R0FC21X - SetMetadataError
R169C6X - MissingFilterError
R173AEX - NoFilterTagError
R1A025X - RepositoryClientKeyNotAllowed
R1CA00X - ClientDoesNotExistError
R22E66X - SizeSyntaxError
R24424X - RepositoryClientDoesNotExist
R283A6X - UnitNameError
R2FA37X - WrongNumberOfGenerationSettingsError
R338F2X - BackupRootMissingError
R3B42AX - WrongNumberOfGenerationsForVerify
R3E151X - RepositoryFileDoesNotExistInGeneration
R3E1C1X - RestoreTargetNotEmpty
R41CE6X - RepositoryClientAlreadyExists
R43272X - RepositoryChunkDoesNotExist
R45B50X - DuplicatePeriodError
R47416X - WrongHostKeyError
R4C3BCX - BackupErrors
R57207X - RepositoryClientGenerationUnfinished
R5914DX - InvalidPortError
R5F98AX - NoHostKeyError
R681AEX - LockFail
R6A098X - RepositoryGenerationKeyNotAllowed

R6C1C8X - RepositoryClientListNotLocked
R6EAF2X - RepositoryClientLockingFailed
R7137EX - BagIdNotSetError
R79699X - RepositoryFileKeyNotAllowed
R79ED6X - BackupRootDoesNotExist
R7B8D0X - FileNotFoundError
R826A1X - UnknownVFSError
R8AAC1X - NoHostKeyOfWantedTypeError
R8F974X - RepositoryChunkIndexesLockingFailed
R91CA1X - ShowFirstGenerationError
R9808DX - ForgetPolicySyntaxError
RA4F35X - RootIsNotADirectory
RA5942X - WrongNumberOfGenerationsForDiffError
RA7D64X - UnknownRepositoryFormatWanted
RA881CX - RepositoryChunkContentNotInIndexes
RA920EX - NotARepository
RABC26X - FuseModuleNotFoundError
RB1048X - RepositoryClientListLockingFailed
RB4324X - GAImmutableError
RB8E98X - WrongURLSchemeError
RB927BX - SeparatorError
RBF6DDX - RepositoryAccessError
RCB0CAX - KeyAuthenticationError
RCE08AX - ObnamIOError
RCEF5CX - MallocError
RD5FA4X - ObnamSystemError
RD6259X - RestoreErrors
RDF30DX - Fail
RE187FX - RepositoryChunkIndexesNotLocked
REFB32X - RepositoryClientHasNoGenerations
RF4EFDX - UnknownRepositoryFormat

By name

BackupErrors - R4C3BCX - There were errors during the backup
BackupRootDoesNotExist - R79ED6X - Backup root does not exist or is not a directory: root
BackupRootMissingError - R338F2X - No backup roots specified
BagIdNotSetError - R7137EX - Bag id not set: cannot append a blob (programming error)
ClientDoesNotExistError - R1CA00X - Client client does not exist in repository repo
DuplicatePeriodError - R45B50X - Forget policy may not duplicate period (period): policy
Fail - RDF30DX - filename: reason
FileNotFoundError - R7B8D0X - FUSE: File not found: filename
ForgetPolicySyntaxError - R9808DX - Forget policy syntax error: policy

FuseModuleNotFoundError - RABC26X - Failed to load module "fuse", try installing python-fuse

GAImmutableError - RB4324X - Attempt to modify an immutable GADirectory

GpgError - R0C79EX - gpg failed with exit code returncode: stderr

HardlinkError - R02C17X - Cannot hardlink on SFTP; sorry
This is due to a limitation in the Python paramiko library that Obnam uses for SSH/SFTP access.

InvalidPortError - R5914DX - Invalid port number port in url: error

KeyAuthenticationError - RCB0CAX - Can't authenticate to SSH server using key

LockFail - R681AEX - Couldn't create lock lock_name: reason

MallocError - RCEF5CX - malloc out of memory while calling function

MissingFilterError - R169C6X - Unknown filter tag: tagname

NoFilterTagError - R173AEX - No filter tag found

NoHostKeyError - R5F98AX - No known host key for hostname

NoHostKeyOfWantedTypeError - R8AAC1X - No known type key_type host key for hostname

NotARepository - RA920EX - url does not seem to be an Obnam repository

ObnamIOError - RCE08AX - I/O error: filename: errno: strerror

ObnamSystemError - RD5FA4X - System error: filename: errno: strerror

RepositoryAccessError - RBF6DDX - Repository does not exist or cannot be accessed: error

RepositoryChunkContentNotInIndexes - RA881CX - Repository chunk indexes do not contain content

RepositoryChunkDoesNotExist - R43272X - Repository doesn't contain chunk chunk_id. It is expected at filename

RepositoryChunkIndexesLockingFailed - R8F974X - Repository chunk indexes are already locked

RepositoryChunkIndexesNotLocked - RE187FX - Repository chunk indexes are not locked

RepositoryClientAlreadyExists - R41CE6X - Repository client client_name already exists

RepositoryClientDoesNotExist - R24424X - Repository client client_name does not exist

RepositoryClientGenerationUnfinished - R57207X - Cannot start new generation for client_name: previous one is not finished yet (programming error)

RepositoryClientHasNoGenerations - REFB32X - Client client_name has no generations

RepositoryClientKeyNotAllowed - R1A025X - Client client_name uses repository format format which does not allow the key key_name to be use for clients

RepositoryClientListLockingFailed - RB1048X - Repository client list could not be locked

RepositoryClientListNotLocked - R6C1C8X - Repository client list is not locked

RepositoryClientLockingFailed - R6EAF2X - Repository client client_name could not be locked

RepositoryClientNotLocked - R0BE94X - Repository client client_name is not locked

RepositoryFileDoesNotExistInGeneration - R3E151X - Client client_name, generation genspec does not have file filename

RepositoryFileKeyNotAllowed - R79699X - Client client_name uses repository format format which does not allow the key key_name to be use for files

RepositoryGenerationDoesNotExist - R0B15DX - Cannot find requested generation gen_id!r for client client_name

RepositoryGenerationKeyNotAllowed - R6A098X - Client client_name uses repository format format which does not allow the key key_name to be used for generations

RepositorySettingMissingError - R01F56X - No –repository setting. You need to specify it on the command line or a configuration file

RestoreErrors - RD6259X - There were errors when restoring
See previous error messages for details.

RestoreTargetNotEmpty - R3E1C1X - The restore –to directory (to) is not empty.

RootIsNotADirectory - RA4F35X - baseurl is not a directory, but a VFS root must be a directory

SeparatorError - RB927BX - Forget policy must have rules separated by commas, see position position: policy

SetMetadataError - R0FC21X - filename: Couldn't set metadata metadata: errno: strerror

ShowFirstGenerationError - R91CA1X - Can't show first generation. Use 'obnam ls' instead

SizeSyntaxError - R22E66X - "size" is not a valid size

ToplevelIsFileError - R018FCX - File at repository root: filename

URLSchemeAlreadyRegisteredError - R0F22CX - VFS URL scheme scheme already registered

UnitNameError - R283A6X - "unit" is not a valid unit

UnknownRepositoryFormat - RF4EFDX - Unknown format format at url

UnknownRepositoryFormatWanted - RA7D64X - Unknown format format requested

UnknownVFSError - R826A1X - Unknown VFS type: url

WrongHostKeyError - R47416X - SSH server hostname offered wrong public key
Note that this may due to an obsolete host key in your "known hosts" file. If so, use "ssh-key -R" to remove it. However, it can also be a sign that someone is trying to hijack your connection to your server, and you should be careful.

WrongNumberOfGenerationSettingsError - R2FA37X - The restore command wants exactly one generation option

WrongNumberOfGenerationsForDiffError - RA5942X - Need one or two generations

WrongNumberOfGenerationsForVerify - R3B42AX - verify must be given exactly one generation

WrongURLSchemeError - RB8E98X - SftpFS used with non-sftp URL: url

Chapter 28

See Also

This chapter gives pointers to more information about Obnam, backups, and related things. For the time being, this is a very short list, but suggestions for things to add to it are very much welcome.

- Obnam home page: http://obnam.org
  - There are short tutorials, download links, an FAQ, contact information, etc, here.
- Lars Wirzenius, interesting blog tags.
  - http://blog.liw.fi/tag/backups/
  - http://blog.liw.fi/tag/obnam/
- Cache directory tag standard: http://www.bford.info/cachedir/
  - http://liw.fi/cachedir/ is a utility to manage the tag files

Chapter 29

Legal stuff

Chapter 30

Supporting Obnam development

Obnam is free software: you get full access to the source code, you can modify the software as you wish, and you can distribute copies of the software in its original or modified form. It is also free of charge.

One of the goals of Obnam is to make sure everyone has access to nice backup software, and are not beholden to anyone else for that software. You can use Obnam, and store your backups anywhere that suits you, and the Obnam developers have no say in that.

However, Obnam development requires some resources. Obnam is primarily developed by Lars Wirzenius, its original author (hi!), in his free time. If you would like to help support Obnam development, here's a list of things you could do:

- Send fixes and improvements, either to code or documentation.
- Donate something to the author. See `http://obnam.org/donate/` for suggestions.
- Hire the author to do some Obnam development. Contact him privately by e-mail `liw@liw.fi`.

Note that any of these are optional. If you like Obnam, and are happy just using it, that's completely OK.

Chapter 31

News

This file summarizes changes between releases of Obnam.

NOTE: Obnam has an EXPERIMENTAL repository format under development, called 'green-albatross'. It is NOT meant for real use. It is likely to change in incompatible ways without warning. Do not use it unless you're willing to lose your backup.

Version 1.19, released 2016-01-15

Bug fixes:

* Backup no longer ignores a closed SSH connection. This means it won't keep trying to use it, forever. Instead, it crashes and terminates the backup.

* The Paramiko SSH implementation, which Obnam uses, changed the interface to the 'prefetch' method in its 1.16 version. Obnam can now deal with either variant of the method. Found and reported by Kyle Manna, who provided a patch that Lars Wirzenius rewrote to be backwards compatible to older versions of Paramiko.

Improvements to the manual:

* The manual now has an appendix listing all Obnam errors, with codes and explanations. This will need to be updated manually from time to time.

* The manual now has sections on turning on full debug logging and reporting problems.

Improvements to functionality:

* The output of 'obnam generations' now show time zone. Lars Wirzenius implemented based on suggestion by Limdi.

Version 1.18.2, released 2015-11-15

Bug fixes:

* The ‘–exclude-caches‘ option now works correctly again. Prevoiusly it would exclude a cache directory, but would scan through and back up the contents of the cache directory. As a result, the backup generation would be much bigger, but have hidden files, not visible in the output of ‘obnam ls‘. To fix that, remove any generations made with Obnam 1.18 or 1.18.1.

This will affect other exclusions as well.

Version 1.18.1, released 2015-11-06

Bug fixes:

* The ‘–quiet‘ option now disables the statistics reports at the end of a backup run.

Version 1.18, released 2015-11-04

Bug fixes:

* William Boughton fixed parsing for sftp URLs with IPv6 addresses. Previously, ‘sftp://[::1]‘ would be interpreted by Obnam as an address ‘[‘ followed by the port ‘:1]‘, but now it is correctly interpreted as the adddress ‘::1‘ and no explicit port.

* Ian Campbell fixed a bug in the kdirstat plugin, improving the handling of unknown file types.

* Lars Wirzenius changed the ‘scan_tree‘ code to not be recursive, to avoid problems with directory trees that are deeper than Python’s call stack limit allows.

Minor changes:

* Lars Wirzenius added support for a multiline progress message during backup. Version 0.24 or newer of ‘ttystatus‘ is needed for this, but Obnam will work with an older version by displaying the same single-line progress message as before.

* Ben Boeckel added the ‘–gnupghome‘ setting so that Obnam can be configured to use a separate GnuPG (gpg) configuration directory.

* Henri Sivonen improved the compression code to not compress if the result would be larger.

Version 1.17, released 2015-09-12

* Luká Poláek added the ‘–fsck-skip-checksums‘ setting to greatly speed up ‘obnam fsck‘.

* Lars Wirzenius fixed a bug that caused Obnam to sometimes back up the parent of the backup live data root. In other words, if running ‘obnam backup /HOME/important‘, then Obnam might backup the whole of the home directory, instead of just the important subdirectory.

Version 1.16, released 2015-09-06

* Fixed another typo in a variable name ("netloc"), found by Benedikt Neuffer.

* Fixed a lot of missing module imports, unnecessary module imports, and other minor bugs and style issues found by pylint. Pylint now gets run automatically by the test suite.

This includes a fix in ‘exclude_pathnames_plugin.py‘ to add a missing import and fix variable namaes, by Diane Trout. A similar fix was also contributed by Mesar Hameed.

* Luká Poláek fixed an unlocking problem when GnuPG fails during an Obnam run. The lock should now be removed rather than left behind.

Version 1.15, released 2015-08-19

* Fixed a typo in a variable name ("netloc"), found by Dirk.

Version 1.14, released 2015-08-14

Bug fixes:

* Since 1.9, Obnam has had trouble with sftp URLs for backup roots, particularly for URLs specifying the server’s root directory. Dennis Jacobfeuerborn found the reason: the backup plugin was treating URLs as filenames. This should now be fixed.

Version 1.13, released 2015-08-01

Bug fixes:

* Luká Poláek found and fixed a repository corruption problem: if ‘obnam forget‘ was interrupted at the wrong moment, it might remove a chunk, but not the reference to it. This would case a future run of ‘obnam forget‘ to crash due to a missing chunk (error code R43272X). ‘obnam forget‘ will now ignore such a missing chunk, since it would’ve deleted it anyway.

Lars Wirzenius then changed things so that chunk files are only removed once references to the chunks have been committed.

Improvements:

* ‘obnam forget‘ now commits changes after each generation it has removed. This means that if the operation is committed, less work is lost. Suggested by Luká Poláek, re-implemented by Lars Wirzenius.

Version 1.12, released 2015-07-08

Bug fixes:

* Steven Monai reported that using '–one-file-system' would crash, and it turned out to be a missing import.

* Jan Niggemann reported that '–exclude-caches' no longer worked. This was due to a bug introduced when the option was moved to its own plugin (for cleaner code). The bug was masked by another bug, in the Yarn test suite. Both bugs have now been fixed.

Improvements:

* Jan Niggemann translated the Obnam manpage to German. Due to cliapp not supporting other languages than English yet, the manual page lacks option descriptions.

Version 1.11, released 2015-07-02

* The 1.10 release failed to correctly include the Green Albatross code, due to a missing line in 'setup.py'. This has been fixed.

Version 1.10, released 2015-07-01

Major bug fixes:

* Lars Wirzenius fixed the 'obnam backup' command to lock the whole repository, the same way as 'obnam forget' does, when it removes checkpoint generations. This means that during checkpoint removal, no other client can make a backup, which is unfortunate. To avoid that, set 'leave-checkpoints = yes' in the configuration. That will prevent 'obnam backup' from removing checkpoints.

Minor new features:

* Lars Wirzenius added the 'obnam list-formats' command to list all repository formats.

* The default value for the 'upload-queue-size' setting is now 1024, chosen based on some benchmarking made by Lars Wirzenius to balance speed and memory use.

* An EXPERIMENTAL new repository format, 'green-albatross', as been introduced. It is not ready for actual use, and is only added so that its code doesn't diverge far from the main line of development.

* Teemu Hukkanen reported that the Synology NAS device returns EACCES instead of ENOENT when user tries to remove a non-existent file. Obnam now copes with either error code.

Minor fixes:

* 'python setup.py build' no longer formats the manual page into plain text. This is now done in 'python setup.py docs' instead. The latter is an optional build step, and probably only works on Debian.

* ‘obnam restore –to=DIR‘ now requires that the directory ‘DIR‘ either doesn’t exist, or it is empty when the restore starts. This is to prevent users from restore on top of a running system.

Version 1.9, released 2015-03-22

New features:

* James Vasile changed Obnam so it can backup an individual file, instead of an entire directory.

* James Vasile added the ‘–include‘ option to Obnam, allowing one to include files that would otherwise be excluded (see ‘–exclude‘).

* Carlo Teubner changed ‘obnam fsck‘ to remove unused chunks, if the ‘–fsck-fix‘ or ‘–fsck-rm-unused‘ settings are used. He also made it not check for unused chunks when it’s useless to do so, because of various ‘–fsck-skip‘ settings are used.

* A start of a French translation of the manual by pedrito2.

* Ian Cambell provided a new Obnam command, ‘obnam kdirstat‘, which makes the KDE ‘k4dirstat‘ utility be able to show graphically which parts of a backup generation use most space.

* Lars Wirzenius added the ‘simple‘ repository format, which is for demonstration only. It is much too simplistic to be used for real.

Minor changes:

* The manual page and ‘obnam –help‘ are now clearer that the ‘–root‘ setting and command line arguments to ‘obnam backup‘ can be SFTP URLs. Thanks to Simone Piccardi for reporting the issue.

* David Fries filled in the displayed file permission mode bits.

* Grammar and typo fixes for the obnam.1 manual page, from Jean Jordaan.

* Tom Chiverton suggested a clarification to the manual page for "obnam mount" to say that each generation is a subdirectory.

* David Fries changed restore to set the group ownership if possible even when not root. No warnings are issued if the attempt fails.

* Jan Niggemann added a little to the German translation of the Obnam manual.

* Lars Wirzenius added the path to the error message about a missing chunk (R43272X).

* Lars Wirzenius made the message at the end of a backup report more statistics about transfers during the backup.

Bug fixes:

* The Obnam SFTP plugin would loop infinitely if it lost the connection to the SSH server while creating a temporary file. Itamar Turner-Trauring provided a fix for this.

* Will Dyson fixed a bug about locking while removing checkpoint generations.

* Michel Alexandre Salim fixed a Python 2.6 compatibility problem in the unit tests (use of 'assertRaises' as a context manager).

* Lars Kruse fixed a bug with backing up of overlapping backup roots (e.g., / and /boot), given a test case by Adrien Clerc.

* Thomas Eschenbacher fixed a bug in the format 6 repository code that would crash when there is an obscure problem and a B-tree code can't be found in the tree.

* Tom Chiverton pointed out that the manual page was using "obnam restore" instead of "obnam mount" in an example for "obnam mount".

* The yarn test suite now runs FUSE tests ('obnam mount') when 'fusermount' is available, rather than checking for membership in the group 'fuse'. The latter is a Debianism (fixed in Debian 'jessie').

* Thomas Waldmann noticed that 'obnam verify' didn't notice that a file had new data, when the modification time was the same. Obnam now notices this.

* Thomas Waldmann fixed many typos and minor bugs in the source code.

* Laurence Perkins reported that the Tahoe-LAFS SFTP server returned some 'stat' fields as None. Fixed to change those to be 0 instead.

* Lars Wirzenius fixed double-downloading of chunks during restores.

Version 1.8, released 2014-05-13

* The error message has been improved for when setting metadata (owner, permission, and similar) of a restored file fails.

* 'obnam force-lock' now works even when the client running it is not in the client list.

Security issues:

* Joey Hess found a problem in 'obnam restore': restored files would be created with quite liberal default permissions, which would be set to the backed-up permissions later. This could allow a snooper to read files they shouldn't be. This has been fixed now by using restrictive default permissions. A workaround for older versions is to create a directory, set its permissions to 0700, and restore to a subdirectory of that directory.

Bug fixes:

* ‘–help‘ output no longer shows the default value of any options. It was shown only for a few options anyway. The proper way to see the current settings is with the ‘–dump-config‘ option. The bug that was fixed that the generated manual page no longer contains values that are specific to the machine doing the generation, such as the hostname as the default value for ‘–client-name‘. Reported by SanskritFritz.

* When a file was backed up, and later excluded with ‘–exclude‘, Obnam wouldn't remove it from the new backups. Now it does. Bug fixed by Anssi Hannula, though his patch got changed because it no longer applied.

* When restoring extended attributes not in the user namespace (named like ‘user.foo‘) Obnam now ignores them, instead of trying to set them and crashing.

* When restoring from a directory that is not a repository, the error message is now clearer.

* Obnam would previously allow the backup root to be a symbolic link pointing at a directory. However, this only worked for backups. No other operations would work and would only see the symbolic link, not the directory it pointed at. Obnam now gives an error message even for the backup.

* Obnam no longer excludes files named ‘syslog‘ or ‘none‘, if the setting ‘–log=none‘ or ‘–log=syslog‘ is used.

Version 1.7.4, released 2014-03-31

* The manual is now dual-licensed under GNU GPL v3 or later, and Creative Commons CC-BY-SA 4.0.

* The 1.7.3 release never went out. Let's pretend it wasn't even tagged in git, and everyone will be happy.

Bug fixes:

* Obnam FUSE got another bug fix from Valery Yundin, to fix a bug I introduced in 1.7. Reading big files via ‘obnam mount‘ should now work better.

* Fix count of backed up files. It used to always count directories. Reported by Alberto Fuentes as Debian bug [742384](https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=742384).

* ‘obnam diff latest‘ would fail due to a programming error. Reported by Junyx.

Version 1.7.2, released 2014-03-22

Bug fixes:

* Fix another bug in the FUSE plugin's file reading code, found during the release process of 1.7.2.

Version 1.7.1, released 2014-03-22

* The 'dump-repo' command now outputs JSON instead of YAML. The dependency on PyYAML is no longer.

Bug fixes:

* Nemo Inis found a bug in the FUSE plugin ('obnam mount'), where Obnam would return the wrong data when the program reading the file didn't read the whole file from the beginning in one read(2) system call.

* The test suite now skips tests that require use of extended attributes in the 'user' namespace. This should allow the test suite to be run on more build servers run by various distributions.

Version 1.7, released 2014-03-15

WARNING: This release has had fairly large parts of the internals re-written. There shouldn't be any externally visible changes due to that, but there is a chance of bugs. Be careful. Make a copy of your backup repository before upgrading, if you can.

* The 'convert5to6' subcommand has been removed. If you need to convert from a pre-1.0 backup repository, and haven't done so yet, please use Obnam version 1.6.1 or earlier to do so.

* A new 'backup-finished' hook is provided by the backup plugin, so that other plugins may do processing at the end of a backup, such as report the successful backup to a monitoring system. Patch by Enrico Tröger.

* The FUSE plugin can now refresh its view, by having the user read the '.pid' file. Patch by Valery Yundin.

* New option '–always-restore-setuid' to always restore setuid/setgid flags in permissions, even if the restore is not being run by 'root' or the owner of the files (as recorded in the backup).

* New option '–exclude-from' allows exclusion patterns to be given in a separate file (one per line), instead of in a configuration file or on the command line. Patch by Enrico Tröger.

* A start of a manual for Obnam. This will gain more content with new releases. The current versions is mainly an edited version of Lars's blog posts about backups, plus the Obnam tutorial from the Obnam homepage. See <http://code.liw.fi/obnam/manual/> for rendered versions (PDF, HTML).

* Most of the error messages Obnam produces now have a unique error code: 'ERROR: R0B15DX: Cannot find requested generation for client havelock' for example. More error messages will gain error codes in future releases. The error codes are meant to be easy to search for, and will allow error messages to be translated in the future.

* The 'obnam-benchmark' program got rewritten so that it'll do something useful, but at the same time, it is no longer useful as a general tool. It is now expected to be run from the Obnam source tree (a cloned git repository), and isn't installed anymore.

* The log file now includes information about the transfer overhead to the repository. Overhead is all the bytes that are not file content data: filenames, permission bits, extended attributes, etc, plus Obnam internal bookkeeping.

* 'obnam verify' now shows progress both based on number of files and amount of data.

Bug fixes:

* Obnam now doesn't remove chunks that are shared between clients. Previously, this would sometimes happen, because only the first client would correctly record itself as using a chunk. Now all clients do that.

* Obnam now creates a 'trustdb.gpg' in the temporary GNUPGHOME it uses during encryption operations. From version 2.0.22 (or thereabouts), 'gpg' insists on having a 'trustdb.gpg' in the GNUPGHOME it uses.

* When backing up a large file, and making a checkpoint generation in the middle of it, Obnam would say "continuing backup" after the checkpoint was finished, instead of saying the name of the file. This is now fixed.

Internal changes:

* The 'obnamlib.Error' exception class has been replaced by the 'obnamlib.ObnamError' class, which derives from the new 'obnamlib.StructuredError' class. All new exceptions will need to be derived from 'obnamlib.Error' in the future. Also, due to the way 'StructuredError' works, it is now necessary to create a new exception class for each kind of error. This gives us unique the error codes mentioned above.

* The old 'obnamlib.Repository' class is gone, and replaced with the 'obnamlib.RepositoryInterface' class, which gets implemented for each repository format (there is only one, for now, but there will be more).

Version 1.6.1, released 2013-11-30

* Fix Debian package dependencies correctly.

Version 1.6, released 2013-11-30

* Stop logging paramiko exceptions that get converted into another type of exception by the SFTP plugin in Obnam.

* ‘obnam-benchmark‘ can now use an installed version of larch. Patch by Lars Kruse.

* Obnam has been ported to FreeBSD by Itamar Turner-Trauring of HybridCluster.

* Backup progress reporting now reports scanned file data, not just backed up file data. This will hopefully be less confusing to people.

* The ‘list-keys‘, ‘client-keys‘, and ‘list-toplevels‘ commands now obey a new option, ‘–key-details‘, to show the usernames attached to each public key. Patch by Lars Kruse.

* New option ‘–ssh-command‘ to set the command Obnam runs when invoking ssh. patch by Lars Kruse.

* ‘obnam clients‘ can now be used without being an existing client. Patch by Itamar Turner-Trauring.

* New option ‘–ssh-host-keys-check‘ to better specify how SSH host keys should be checked. Patch by Itamar Turner-Trauring.

Bug fixes:

* Fix‘"obnam list-toplevels‘ so it doesn’t give an error when it’s unable to read the per-client directory of another client, when encryption is used. Fix by Lars Kruse.

* Fix the encryption plugin to give a better error message when it looks for client directories but fails to find them. Fix by Lars Kruse.

* ‘obnam list-toplevels‘ got confused when the repository contained extra files, such as "lock" (left there by a previous, crashed Obnam run). It no longer does. Fix by Lars Kruse.

* The SFTP plugin now handles another error code (EACCESS) when writing a file and the directory it should go into not existing. Patch by Armin GröSSlinger.

* Obnam’s manual page now explains about breaking long logical lines into multiple physical ones.

* The ‘/ /‘ path prefix in SFTP URLs works again, at least with sufficiently new versions of Paramiko (1.7.7.1 in Debian wheezy is OK). Reported by Lars Kruse.

* The Nagios plugin to report errors in a way Nagios expects. Patch by Martijn Grendelman.

* The Nagios plugin for Obnam now correctly handles the case where a backup repository for a client exists, but does not have a backup yet. Patch by Lars Kruse.

* ‘obnam ls‘ now handles trailing slashes in filename arguments. Reported by Biltong.

* When restoring a backup, Obnam will now continue past errors, instead of aborting with the first one. Patch by Itamar Turner-Trauring.

Version 1.5, released 2013-08-08

Bug fixes:

* Terminal progress reporting now updated only every 0.1 seconds, instead of 0.01 seconds, to reduce terminal emulator CPU usage. Reported by Neal Becker. * Empty exclude patterns are ignored. Previously, a configuration file line such as "exclude = foo, bar," (note trailing comma) would result in an empty pattern, which would match everything, and therefore nothing would be backed up. Reported by Sharon Kimble. * A FUSE plugin to access (read-only) data from the backup repository has been added. Written by Valery Yundin.

Version 1.4, released 2013-03-16

* The 'ls' command now takes filenames as (optional) arguments, instead of a list of generations. Based on patch by Damien Couroussé. * Even more detailed progress reporting during a backup. * Add –fsck-skip-generations option to tell fsck to not check any generation metadata. * The default log level is now INFO, instead of DEBUG. This is to be considered a quantum leap in the continuing rise of the maturity level of the software. (Actually, the change is there just to save some disk space and I/O for people who don't want to be involved in Obnam development and don't want to have massive log files.) * The default sizes for the 'lru-size' and 'upload-queue-size' settings have been reduced, to reduce the memory impact of Obnam. * 'obnam restore' now reports transfer statistics at the end, similarly to what 'obnam backup' does. Suggested by "S. B.".

Bug fixes:

* If listing extended attributes for a filesystem that does not support them, Obnam no longer crashes, just silently does not backup extended attributes. Which aren't there anyway. * A bug in handling stat lookup errors was fixed. Reported by Peter Palfrader. Symptom: 'AttributeError: 'exceptions.OSError' object has no attribute 'st_ino'' in an error message or log file. * A bug in a restore crashing when failing to set extended attributes on the restored file was fixed. Reported by "S. B.". * Made it clearer what is happening when unlocking the repository due to errors, and fixed it so that a failure to unlock is also an error. Reported by andrewsh. * The dependency on Larch is now for 1.20121216 or newer, since that is needed for fsck to work. * The manual page did not document the client name arguments to the 'add-key' and 'remove-key' subcommands. Reported by Lars Kruse. * Restoring symlinks as root would fail. Reported and fixed by David Fries. * Only set ssh user/port if explicitily requested, otherwise let ssh select them.

Reported by Michael Goetze, fixed by David Fries. * Fix problem with old version of paramiko and chdir. Fixed by Nick Altmann. * Fix problems with signed vs unsigned values for struct stat fields. Reported by Henning Verbeek.

Version 1.3, released 2012-12-16

* When creating files in the backup repository, Obnam tries to avoid NFS synchronisation problems by first writing a temporary file and then creating a hardlink to the actual filename. This works badly on filesystems that do not allow hard links, such as VFAT. If creating the hardlink fails, Obnam now further tries to use the 'open(2)' system call with the 'O_EXCL' flag to create the target file. This should allow things to work with both NFS and VFAT. * More detailed progress reporting during the backup. * Manual page now covers the diff subcommand. Patch by Peter Valdemar Mørch. * Speed optimisation patch for backing up files in inode numbering order, from Christophe Vu-Brugier. * A setuid or setgid bit is now not restored if Obnam is not used by root or the same user as the owner of the restored file. * Many new settings to control "obnam fsck", mainly to reduce the amount of checking being done in order to make it faster. However, fsck is has lost some features (checks), which will be added back in a future release. * More frequent fsck progress reporting. Some speed optimisations to fsck.

Bug fixes:

* Empty values for extended attributes are now backed up correctly. Previously they would cause an infinite loop. * Extended attributes without values are now ignored. This is different from attributes with empty values. Reported by Vladimir Elisseev. * An empty port number in sftp URLs is now handled correctly. Found based on report by Anton Shevtsov. * A bad performance bug when backing up full systems (starting from the filesystem root directory) has been fixed. At the beginning of each generation, Obnam removes any directories that are not part of the current backup roots. This is necessary so that if you change the backup roots, the old stuff doesn't hang around forever. However, when the backup root is the filesystem root, due to the now-fixed bug Obnam would first remove everything, and then back it up all over again. This "worked", but was quite slow. Thanks to Nix for reporting the problem. * Obnam now runs GnuPG explicitly with the "no text mode" setting, to override a "text mode" setting in the user's configuration. The files Obnam encrypts need to be treated as binary, not text files. Reported by Robin Sheat. * A shared B-tree concurrency bug has been fixed: If another instance of Obnam was modifying a shared B-tree, Obnam would crash and abort a backup, possibly leaving lock files lying around. Now a failure to look up a chunk via its checksum is ignored, and the backup continues. * Bugs in how Python OSError exceptions were being raises have been fixed. Error messages should now be somewhat clearer. * Unset or wrongly set variable "full" fixed in "obnam diff". Reported by

ROGERIO DE CARVALHO BASTOS and patched by Peter Valdemar Mørch. * Setuid and setgid bits are now restored correctly, when restore happens as root. Reported by Pavel Kokolemin. * Obnam now complains if no backup roots have been specified.

Version 1.2, released 2012-10-06

* Added a note to ‘–node-size‘ that it only affects new B-trees. Thanks, Michael Brown. * New ‘obnam diff‘ subcommand to show differences (added/removed/modified files) between two generations, by Peter Valdemar Mørch. * ‘obnam backup‘ now logs the names of files that are getting backed up at the INFO level rather than DEBUG. * The command synopsises for backup, restore, and verify commands now make it clearer that Obnam only accepts directories, not individual files, as arguments. (For now.) * The output from the ‘show‘ plugin can now be redirected with the ‘–output=FILE‘ option. Affected subcommands: ‘clients‘, ‘generations‘, ‘genids‘, ‘ls‘, ‘diff‘, ‘nagios-last-backup-age‘.

Bug fixes:

* Notify user of errors during backups. * The SFTP plugin now manages to deal with repository paths starting with ‘/ /‘ which already exist without crashing. * Character and block device nodes are now restored correctly. Thanks to Martin Dummer for the bug report. * The symmteric key for a toplevel repository directory is re-encrypted when a public key is added or removed to the toplevel using the ‘add-key‘ or ‘remove-key‘ subcommands. * Manual page typo fix. Thanks, Steve Kemp.

Version 1.1, released 2012-06-30

* Mark the ‘–small-files-in-btree‘ settings as deprecated. * Obnam now correctly checks that ‘–repository‘ is set. * Options in ‘–help‘ output are now grouped in random senseless ways rather than being in one randomly ordered group. * Manual page clarification for ‘–root‘ and ‘verify‘. Thanks, Saint Germain. * Remove outdated section from manual page explaining that there is not format conversion. Thanks, Elrond of Samba-TNG. * Added missing information about specifying a user in sftp URLs. Thanks, Joey Hess, for pointing it out. * Manual page clarification on ‘–keep‘ from Damien Couroussé. * Make ‘obnam forget‘ report which generations it would remove without ‘–pretend‘. Thanks, Neal Becker, for the suggestion.

Version 1.0, released 2012-06-01

* Fixed bug in finding duplicate files during a backup generation. Thanks to Saint Germain for reporting the problem. * Changed version number to 1.0.

Version 0.30, released 2012-05-30; a RELEASE CANDIDATE

Only bug fixes, and only in the test suite.

* Fix test case problem when ‘$TMPDIR$‘$lacks$‘$user\_xattr$‘.$Theextendedattributestestwon't succe
$Fixtestcaseproblemwhen$‘TMPDIR‘ lacks nanosecond timestamps for files. The test case now ignores such timestamps, making the test pass anyway. The timestamp accuracy is not important for this test.

Version 0.29, released 2012-05-27; a RELEASE CANDIDATE

* "obnam backup" now writes performance statistics at the end of a backup run. Search the log for "Backup performance statistics" (INFO level). * "obnam verify" now continues past the first error. Thanks to Rafa Gwiazda for requesting this. * Add an ‘obnam-viewprof‘ utility to translate Python profiling output into human readable text form. * Bug fix: If a file’s extended attributes have changed in any way, the change is now backed up. * "obnam fsck" is now a bit faster. * The shared directories in the repository are now locked only during updates, allowing more efficient concurrent backups between several computers. * Obnam now gives a better error message when a backup root is not a directory. Thanks to Edward Allcutt for reporting the error (<http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=654211>). * The output format of "obnam ls" has changed. It now has one line per file, and includes the full pathname of the file, rather mimicking the output of "ls -lAR". Thanks to Edward Allcutt for the suggestion (<http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=655095>). * A few optimizations to sftp speed. Small files are still slow.

Version 0.28, released 2012-05-10; a BETA release

* ‘force-lock‘ should now remove all locks. * Out-of-space errors in the repository now terminate the backup process. Previously, Obnam would continue, ignoring the failure to write. If you make space in the repository and restart Obnam, it will continue from the previous checkpoint. * The convert5to6 black box test now works even if run by other people than liw. * "obnam backup" now uses a single SFTP connection to the backup repository, rather than opening a new one after each checkpoint generation. Thanks to weinzwang for reporting the problem. * "obnam verify" now obeys the ‘–quiet‘ option. * "obnam backup" no longer counts chunks already in the repository in the uploaded amount of data.

Version 0.27, released 2012-04-30; a BETA release

* The repository format has again changed in an incompatible manner, so you will need to re-backup everything again. Alternatively, you can try the new ‘convert5to6‘ subcommand. See the manual page for details. Make sure you have a copy of the repository before converting, the code is new and may be buggy. * New option ‘–small-files-in-btree‘ enables Obnam to store the contents of small files in the per-client B-tree. This is not the default, at least yet, since it’s impact on real life performance is unknown, but it should make things go a bit faster for high latency repository connections.

* Some SFTP related speed optimizations. * Data filtering is now strictly stable and priority-ordered, ensuring that compression always happens before encryption etc. * Repository metadata is never filtered, so that we can be sure that in future if when we add backwards-compatibility we can detect the format without worrying about any other filtering which might occur. * Forcing of locks is now unconditional and across the entire repository. * Uses the larch 0.30 read-only mode to fix a bug where opening a B-tree rolls back changes someone else is making, even if we only use the tree to read stuff from. * "obnam backup" will now exit with a non-zero exit code if there were any errors during a backup, and the problematic files were skipped. Thanks, Peter Palfrader, for reporting the bug. * "obnam forget" is now a bit faster. * Hash collisions for filenames are now handled.

Version 0.26, released 2012-03-26; a BETA release

* Clients now lock the parts of the backup repository they're using, while making any changes, so that multiple clients can work at the same time without corrupting the repository. * Now depends on a larch 0.28, which uses journalling to avoid on-disk inconsistencies and corruption during crashes. * Compression and encryption can now be used together.

Version 0.25, released 2012-02-18; a BETA release

* Log files are now created with permissions that allow only the owner to read or write them. This fixes a privacy leak. * The 'nagios-last-backup-age' subcommand is useful for setting up Nagios (or similar systems) to check that backups get run properly. Thanks to Peter Palfrader for the patch. * Some clarification on how the forget policy works, prompted by questions from Peter Palfrader. * New settings 'ssh-known-hosts' (for choosing which file to check for known host keys), 'strict-ssh-host-keys' (for disallowing unknown host keys), and 'ssh-key' (for choosing which key file to use for SSH connections) allow better and safer use of ssh. * Checkpoints will now happen even in the middle of files (but between chunks). * The '–pretend' option now works for backups as well.

BUG FIXES:

* 'obnam ls' now shows the correct timestamps for generations. Thanks, Anders Wirzenius.

Version 0.24.1, released 2011-12-24; a BETA release

BUG FIX:

* Fix test case for file timestamps with sub-second resolution. Not all filesystems have that, so the test case has been changed to accept lack of sub-second timestamps.

Version 0.24, released 2011-12-18; a BETA release

USER VISIBLE CHANGES

* The way file timestamps (modification and access times) have changed, to fix inaccuracies introduced by the old way. Times are now stored as two integers giving full seconds and nanoseconds past the full second, instead of the weird earlier system that was imposed by Python's use of floating point for the timestamps. This causes the repository format version to be bumped, resulting in a need to start over with an empty repository. * Extended file attributes are now backed up from and restored to local filesystems. They are neither backed up, nor restored for live data accessed over SFTP. * If the '–exclude' regular expression is wrong, Obnam now gives an error message and then ignores the regexp, rather than crashing. * There is now a compression plugin, enabled with '–compress-with=gzip'. * De-duplication mode can now be chosen by the user: the new '–deduplicate' setting can be one of 'never' (fast, but uses more space); 'verify' (slow, but handles hash collisions gracefully); and 'fatalist' (fast, but lossy, if there is a hash collision). 'fatalist' is the default mode. * Restores now obey the '–dry-run' option. Thanks to Peter Palfreder for the bug report. * New option '–verify-randomly' allows you to check only a part of the backup, instead of everything. * Verify now has some progress reporting. * Forget is now much faster. * Forget now has progress reporting. It is not fast enough to do without, sorry. * Backup now removes any checkpoint generations it created during a backup run, if it succeeds without errors.

BUG FIXES:

* Now works with a repository on sshfs. Thanks to Dafydd Harries for reporting the problem. * Now depends on a newer version of the larch library, fixing a problem when the Obnam default node size changes and an existing repository has a different size. * User and group names for sftp live data are no longer queried from the local system. Instead, they're marked as unknown.

Version 0.23, released 2011-10-02; a BETA release

USER VISIBLE CHANGES:

* 'restore' now shows a progress bar. * 'fsck' now has more useful progress reporting, and does more checking, including the integrity of the contents of file content. * 'fsck' now also checks the integrity of the B-trees in the repository, so that it is not necessary to run 'fsck-larch' manually anymore. This works remotely as well, whereas 'fsck-larch' only worked on B-trees on the local filesystem. * 'force-lock' now gives a warning if the client does not exist in the repository. * Subcommands for encryption now give a warning if encryption key is not given. * The '–fsck-fix' option will now instruct 'obnam fsck' to try to fix problems found. For this release, it only means fixing B-tree missing node problems, but more will follow. * The default sizes have been changed for B-tree nodes (256 KiB) and file contents chunks

(1 MiB), based on benchmarking. * SFTP protocol use has been optimized, which should result in some more speed. This also highlights the need to change obnam so it can do uploads in the background. * If a client does not exist in the repository, ‘force-lock‘ now gives a warning to the user, rather than ignoring it silently.

DEVELOPER CHANGES:

* New ‘–sftp-delay=100‘ option can be used to simulate SFTP backups over networks with long round trip times. * ‘obnam-benchmark‘ can now use ‘–sftp-delay‘ and other changes to make it more useful.

INTERNAL CHANGES:

* Got rid of terminal status plugin. Now, the ‘Application‘ class provides a ‘ttystatus.TerminalStatus‘ instance instead, in the ‘ts‘ attribute. Other plugings are supposed to use that for progress reporting and messaging to the user. * The ‘posix_fadvise‘ system call is used only if available. This should improve Obnam’s portability a bit.

Version 0.22, released 2011-08-25; a BETA release

USER VISIBLE CHANGES:

* Obnam now reports its current configuration in the log file at startup. This will hopefully remove one round of "did you use the –foo option?" questions between developers and bug reporters.

BUG FIXES:

* The repository is now unlocked on exit only if it is still locked. * A wrongly caught ‘GeneratorExit‘ is now dealt with properly. * Keyboard interrupts are logged, so they don’t show up as anonymous errors.

CHANGES RELEVANT TO DEVELOPERS ONLY:

* ‘setup.py‘ has been enhanced to work more like the old ‘Makefile‘ did: ‘clean‘ removes more artifacts. Instructions in ‘README‘ have been updated to point at ‘setup.py‘. * Compiler warning about ‘_XOPEN_SOURCE‘ re-definition fixed. * Tests are now again run during a Debian package build.

Version 0.21, released 2011-08-23; a BETA release

USER VISIBLE CHANGES:

* Obnam will now unlock the repository if there’s an error during a backup. For the most part, the ‘force-lock‘ operation should now be unnecessary, but it’s still there in case it’s useful some day.

BUG FIXES:

* Negative timestamps for files now work. Thanks to Jamil Djadala for reporting the bug. * The documentation for –checkpoint units fixed. Thanks, user weinzwang from IRC. * The connections to the repository and live data filesystem are now properly closed. This makes benchmark read/write statistics be correct.

Version 0.20.1, released 2011-08-11; a BETA release

BUG FIXES:

* More cases of Unicode strings versus plain strings in filenames over SFTP fixed. Thanks to Tapani Tarvainen.

Version 0.20, released 2011-08-09; a BETA release

BUG FIXES:

* Non-ASCII filenames over SFTP root now work. (Thanks, Tapani Tarvainen, for the reproducible bug report.) * The count of files while making a backup now counts all files found, not just those backed up. The old behavior was confusing people.

USER VISIBLE CHANGES:

* The output of 'obnam ls' now formats the columns a little prettier, so that wide values do not cause misalignment. * The error message when trying to use an encrypted repository without encryption is now better (and suggests missing encryption being the reason). Thanks, chrysn. * Obnam now supports backing up of Unix sockets.

Version 0.19, released 2011-08-03; a BETA release

INCOMPATIBILITY CHANGES:

* We now require version 0.21 of the 'larch' library, and this requires bumping the repository format. This means old backup repositories can't be used with this version, and you need to back up everything again. (Please tell me when this becomes a problem.)

BUG FIXES:

* Found one more place where a file going missing during a backup may cause a crash. * Typo in error message about on-disk formats fixed. (Thanks, Tapani Tarvainen.) * The '–trace' option works again. * 'fcntl.F_SETFL' does not seem to work on file descriptors for files owned by root that are read-only to the user running obnam. Worked around by ignoring any problems with setting the flags. * The funnest bug in this release: if no log file was specified with '–log', the current working directory was excluded from the backup.

USER VISIBLE CHANGES:

* ‘obnam(1)‘ manual page now discusses how configuration files are used. * The manual page describes problems using sftp to access live data. * The documentation for ‘–no-act‘ was clarified to say it only works for ‘forget. (Thanks, Daniel Silverstone.) * ‘obnam-benchmark‘ now has a manual page. * The backup plugin logs files it excludes, so the user can find out what’s going on. A confused user is an unhappy user.

INTERNAL STUFF:

* Tracing statements added to various parts of the code, to help debug mysterious problems. * All exceptions are derived from ‘obnamlib.AppException‘ or ‘obnamlib.Error‘, and those are derived from ‘cliapp.AppException‘, so that the user gets nicer error messages than Python stack traces. * ‘blackboxtests‘ is no longer run under fakeroot, because Debian packages are built under fakeroot, and fakeroot within fakeroot causes trouble. However, the point of running tests under fakeroot was to make sure certain kinds of bugs are caught, and since Debian package building runs the tests anyway, the test coverage is not actually diminished. * The ‘Makefile‘ has new targets ‘fast-check‘ and ‘network-tests‘. The latter runs tests over sftp to localhost.

Version 0.18, released 2011-07-20; a BETA release

* The repository format has again changed in an incompatible manner, so you will need to re-backup everything again. (If this is a problem, tell me, and I’ll consider adding backwards compatibility before 1.0 is released.) * New option ‘–exclude-caches‘ allows automatic exclusion of cache directories that are marked as such. * Obnam now makes files in the repository be read-only, so that they’re that much harder to delete by mistake. * Error message about files that can’t be backed up now mentions the correct file. * Bugfix: unreadable files and directories no longer cause the backup to fail. The problems are reported, but the backup continues. Thanks to Jeff Epler for reporting the bug. * Speed improvement from Jeff Epler for excluding files from backups. * Various other speed improvements. * Bugfix: restoring symlinks now works even if the symlink is restored before its target. Also, the permissions of the symlink (rather than its target) are now restored correctly. Thanks to Jeff Epler for an exemplary bug report. * New option ‘–one-file-system‘, from Jeff Epler. * New benchmarking tool ‘obnam-benchmark‘, which is more flexible than the old ‘run-benchmark‘. * When encrypting/decrypting data with GnuPG, temporary files are no longer used. * When verifying, ‘.../foo‘ and ‘.../foo/‘ now work the same way. * New option ‘–symmetric-key-bits‘. * The chunk directory uses more hierarchy levels, and the way chunks are stored there is now user-configurable (but you’ll get into trouble if you don’t always use the same configuration). This should speed things up a bit once the number of chunks grows very large. * New ‘–chunkids-per-group‘ option, for yet more knobs to tweak when searching for optimal performance. * Local files are now opened using ‘O_NOATIME‘ so they can be backed up without affecting timestamps. * Now uses the ‘cliapp‘ framework for writing command line

applications. The primary user-visible effect is that the manpage now has an accurate list of options. * Bugfix: Obnam now again reports VFS I/O statistics. * Bugfix: Obnam can again back up live data that is accessed using sftp. Thanks to Tapani Tarvainen for reporting the problem.

Version 0.17, released 2011-05-21; a BETA release

* This is the second BETA release. * The 'run-benchmark' script now works with the new version of 'seivot'. The only benchmark size is one gibibyte, for now, because Obnam's too slow to do big ones in reasonable time. As an aside, the benchmark script got rewritten in Python, so it can be made more flexible. * Benchmarks are run using encrypted backups. * The kernel buffer cache is dropped before each obnam run, so the benchmark result is more realistic (read: slower). * Obnam now rotates its logs. See '–log-max' and '–log-keep' options in the manual page. The default location for the log file is now ' /.cache/obnam/obnam.log' for people, and '/var/log/obnam.log' for root. * Obnam now restores sparse files correctly. * There have been some speed improvements to Obnam. * The '–repository' option now has the shorter alias '-r', since it gets used so often. * 'obnam force-lock' now merely gives an error message, instead of a Python stack trace, if the repository does not exist. * Obnam now does not crash if files go missing during a backup, or can't be read, or there are other problems with them. It will report the problem, but then continue as if it had never heard of the file. * Obnam now supports FIFO files. * Obnam now verifies checksums when it restores files. * Obnam now stores the checksum for the whole file, not just the checksum for each chunk of its contents. * Obnam's own log file is automatically excluded from backups. * Obnam now stores and restores file timestamps to full accuracy, instead of truncating them to whole seconds. * The format of the backup repository has changed in an incompatible way, and Obnam will now refuse to use an old repository. This means you will need to use an old version to restore from them, and need to re-backup everything. Sorry.

Version 0.16, released 2011-07-17; a BETA release

* This is the first BETA release. Obnam should now be feature complete for real use. Performance is lacking and there are many bugs remaining. There are no known bugs that would corrupt backed up data, or prevent its recovery. * Add encryption support. See the manual page for how to use it.

Version 0.15.1, released 2011-03-21; an ALPHA release

* Fix 'setup.py' to not import 'obnamlib', so it works when building under pbuilder on Debian. Meh.

Version 0.15, released 2011-03-21; an ALPHA release

Bugs fixed:

* Manual page GPL copyright blurb is now properly marked up as a comment. (Thanks, Joey Hess.) * README now links to python-lru correctly. (Thanks, Erik Johansson.)

Improvements and other changes:

* Filenames and directories are backed up in sorted order. This should make it easier to know how far obnam's gotten. * The location where backups are stored is now called the repository, instead of the store. Suggested by Joey Hess. * The repository and the target directory for restored data are now both created by Obnam, if they don't already exist. Suggested by Joey Hess. * Better control of logging, using the new '–trace' option. * Manual page now explains making backups a little better. * Default value for '–lru-size' reduced to 500, for great improvement in memory used, without, it seems, much decrease in speed. * 'obnam verify' now reports success explicitly. Based on question from Joey Hess. * 'obnam verify' now accepts both non-option arguments and the '–root' option. Suggested by Joey Hess. * 'obnam forget' now accepts "generation specifiers", not just numeric generation ids. This means that 'obnam forget latest' works. * I/O statistics are logged more systematically. * 'obnam force-lock' introduced, to allow breaking a lock left behind if obnam crashes. But it never does, of course. (Well, except if there's a bug, like when a file changes at the wrong moment.) * 'obnam genids' introduced, to list generation ids without any other data. The old command 'obnam generations' still works, and lists other info about each generation as well, but that's sometimes bad for scripting. * The '–dump-memory-profile' option now accepts the value 'simple', for reporting basic memory use. It has such a small impact that it's the default. * Obnam now stores the version of the on-disk format in the repository. This should allow it to handle repositories created by a different version and act suitably (hopefully without wiping all your backups).

Version 0.14, released 2010-12-29; an ALPHA release

This version is capable of backing up my laptop's home directory. It is, however, still an ALPHA release, and you should not rely on it as your sole form of backup. It is also slow. But if you're curious, now would be a good time to try it out a bit.

Bug fixes:

* 'COPYING' now contains GPL version 3, instead of 2. The code was licensed under version 3 already. (Thank you Greg Grossmeier.) * The manual page now uses '-' and '' correctly. * 'obnam forget' now actually removes data that is no longer used by any generation. * When backing up a new generation, if any of the root directories for the backup got dropped by the user, they are now also removed from the backup generation. Old generations obviously still have them. * Only the per-client B-tree forest should have multiple trees. Now this actually happens, whereas previously sometimes a very large number of new trees would be created in some forests. (What's good for rain forests is not good for saving disk space.) *

When recursing through directory trees, obnam no longer follows symlinks to directories. * obnam no longer creates a missing backup store when backing up to a local disk. It never did this when backing up via sftp. (This saves me from figuring out which of ‘store‘, ‘stor‘, and ‘sorte‘ is the real directory.)

New features and stuff:

* ‘blackboxtest‘ has been rewritten to use Python’s ‘unittest‘ framework, rather than a homegrown bad re-implementation of some of it. * ‘obnam ls‘ interprets arguments as "genspecs" rather than generation identifiers. This means ‘obnam ls latest‘ works, and now ‘latest‘ is also the default if you don’t give any spec. * ‘run-benchmarks‘ now outputs results into a git checkout of <http://braawi.org/>, an ikiwiki instance hosted by <http://www.branchable.com/>. The script also puts the results into a suitable sub-directory, adds a page for the RSS feed of benchmark results, and updates the report page that summarizes all stored results. * There is now a 100 GiB benchmark. * Clients are now called clients, instead of hosts. This terminology should be clearer. * The list of clients now stores a random integer identifier for each client (unique within the store). The identifier is used as the name of the per-client B-tree directory, rather than the hostname of the client. This should prevent a teeny tiny bit of information leakage. It also makes debugging things much harder. * Various refactorings and prettifications of the code has happened. For example, several classes have been split off from the ‘store.py‘ module. This has also resulted in much better test coverage for those classes. * The per-client trees (formerly GenerationStore, now ClientMetadataTree) have a more complicated key now: 4 parts, not 3. This makes it easier to keep separate data about files, and other data that needs to be stored per-generation, such as what the generation id is. * ‘find-duplicate-chunks‘, a tool for finding duplicate chunks of data in a files in a directory tree, was added to the tree. I have used it to find out if is worthwhile to do duplicate chunk removal at all. (It is, at least for my data.) Also, it can be used to find good values for chunk sizes for duplicate detection. * The whole way in which obnam does de-duplication got re-designed and re-implemented. This is tricky stuff, when there is more than one client. * ‘SftpFS‘ now uses a hack copied from bzrlib, to use openssh if it is available, and paramiko only if it is not. This speeds up sftp data transfers quite a bit. (Where bzrlib supports more than just openssh, we don’t, since I have no way to test the other stuff. Patches welcome.) * The way lists of chunk ids are stored for files got changed. Now we store several ids per list item, which is faster and also saves some space in the B-tree nodes. Also, it is now possible to append to the list, which means the caller does not need to first gather a list of all ids. Such a list gets quite costly when the file is quite big (e.g., in the terabyte size). * New ‘–dump-memory-profile‘ option was added to help do memory profiling with meliae or heapy have been added. (Obnam’s memory consumption finally got annoying enough that I did something about it.)

Removed stuff:

* The functional specification was badly outdated, and has been removed. I decided to stop kidding myself that I would keep it up to date. * The store design document has been removed from the store tree. The online version at <http://braawi.org/obnam/ondisk/> is the canonical version, and is actually kept up to date. * The benchmark specification has likewise been replaced with <http://braawi.org/obnam/benchmarkspec/>.

Version 0.13, released 2010-07-13; an ALPHA release

* Bug fix: a mistake in 0.12 caused checkpoints to happen after each file after the first checkpoint. Now they happen at the right intervals again. * Upload speed is now displayed during backups. * Obnam now tells the kernel that it shouldn't cache data it reads or writes. It is not likely that data being backed up is going to be needed again any time soon, so there's no point in caching it. (The posix_fadvise call is used for this.) * New –lru-size option sets size of LRU cache for nodes in memory. The obnam default is large enough to suit large backups. This uses more memory, but is faster than btree's small default of 100.

Version 0.12, released 2010-07-11; an ALPHA release

* NOTE: This version makes incompatible changes to the way data is stored on-disk. Backups made with older versions are NOT supported. Sorry. * The run-benchmark script has dropped some smaller sizes (they're too fast to be interesting), and adds a 10 GiB test size. * Various speed optimizations. Most importantly, the way file metadata (results of lstat(2)) are encoded has changed. This is the incompatible change from above. It's much faster now, though. * Preliminary support for using SFTP for the backup store added. Hasn't been used much yet, so might well be very buggy.

Version 0.11, released 2010-07-05; an ALPHA release

* Speed optimizations: - chunk identifiers are now sequential, except for the first one, or when there's a collision - chunks are now stored in a more sensible directory hierarchy (instead of one per directory, on average) - adding files to a directory in the backup store is now faster - only store a file's metadata that if it is changed * New –exclude=regexp option to exclude files based on pathnames * Obnam now makes checkpoints during backups. If a backup is aborted in the middle and then re-started, it will continue from the latest checkpoint rather than from the beginning of the previous backup run. - New option –checkpoint to set the interval between checkpoints. Defaults to 1 GiB. * Options for various B-tree settings. This is mostly useful for finding the optimal set of defaults, but may be useful in other situations for some people. - New options –chunk-group-size, –chunk-size, –node-size, –upload-queue-size. * Somewhat better progress reporting during backups.

Version 0.10, released 2010-06-29; an ALPHA release

* Rewritten from scratch. * Old NEWS file entries removed (see bzr if you're interested).

Glossary

B

backup a separate safe copy of your live data that will remain intact even if the primary copy gets destroyed deleted or wrongly modified 10

backup history all the backup generations 11

backup media where a backup repository is stored 11

backup repository the location where your backups are stored 11

backup root a directory that is to be backed up, including all files in it, and all its subdirectories 18

backup strategy a plan for how to make sure your data is safe even if the dinosaurs return in space ships to re-take world now that the ice age is over 11

backup tools 11

C

checkpoint 8

cliapp 45

corruption unwanted modification to (backup) data 25

D

de-duplicate this is a specialised data compression technique for eliminating duplicate copies of repeating data 23

disaster recovery what you do when something goes wrong 11

F

full backup a fresh backup of all precious live data 14

G

generation a backup in a series of backups of the same live data, to give historical insight 11

I

incremental backup a backup of any changes (new files, modified files, deletions) compared to a previous backup generation (either the previous full backup or the previous incremental backup); usually, you can't remove a full backup without removing all of the incremental backups that depend on it 14

K

key identifier 9

L

Live data all the data you have 10

O

off-site a backup repository stored physically far away from the live data 11

P

precious all the data you care about, see also Backup concepts 10

pull backup 21

R

restore retrieving data from a backup repository 10

root a directory that is to be backed up, including all files in it, and all its subdirectories 18

S

server 19

snapshot backups an alternative to full/incremental backups where every backup generation is effectively a full backup of all the precious live data and can be restored and removed as easily as any other generation 14

ssh-key 19

V

verify making sure a backup system works and that data actually can be restored from backups and that the backups have not become corrupted 11

Index of commands

In this index you'll find the commands which should be prefaced with "obnam"

Symbols

-
- h 48
- r REPOSITORY 48

--
- client-name=CLIENT-NAME 49
- compress-with=PROGRAM 50
- critical-age=AGE 50
- de-duplicate
  - default 24
  - fatalist 24
  - never 24
  - verify 24
- dry-run 49
- generation=GENERATION 50
- help 48
- keep=KEEP 51
- lock-timeout=TIMEOUT . . . 50
- no-act 49
- no-dry-run 50
- no-no-act 50
- no-pretend 50
- no-quiet 49
- no-verbose 49
- output=FILE 48
- pretend 49
- quiet 49
- repository=REPOSITORY . . 48
- root=ROOT 50
- testing-fail-matching=REGEXP 50
- to=TO 50
- trace=TRACE 49
- verbose 49
- verify-randomly=N 51
- version 48
- warn-age=AGE 50

B

backup **8**, **17**

Backing up
- --
  - checkpoint=SIZE 52
  - de-duplicate=MODE 52
  - exclude-caches 52
  - exclude=EXCLUDE 52
  - leave-checkpoints 53
  - no-leave-checkpoints 53
  - no-one-file-system 52
  - no-small-files-in-btree 53
  - small-files-in-btree 53

C

checkpoint 22
chunk-size 23
clients **35**
client-name 34
config 45

D

dump-config 46

E
encrypt-with 38
Encryption
--
encrypt-with=ENCRYPT-WITH 54
keyid=KEYID 54
no-weak-random 54
symmetric-key-bits=SYMMETRIC-KEY-BITS 54
weak-random 54
exclude 19
exclude-caches 19

F
force-lock 25
forget **30**
forget 8
fusermount 28

G
generations **18**

I
Integrity checking
--
fsck-fix 55
fsck-ignore-chunks 55
fsck-ignore-client=NAME 55
fsck-last-generation-only . 55
fsck-skip-dirs 56
fsck-skip-files 56
fsck-skip-generations 55
fsck-skip-per-client-b-trees 56
fsck-skip-shared-b-trees . . 56
no-fsck-fix 55
no-fsck-ignore-chunks . . . 55
no-fsck-last-generation-only 55
no-fsck-skip-dirs 56
no-fsck-skip-files 56
no-fsck-skip-generations . 56
no-fsck-skip-per-client-b-trees 56
no-fsck-skip-shared-b-trees 56

K
keep 30

L
Logging
--
log-keep=N 57
log-level=LEVEL 57
log-max=SIZE 57
log-mode=MODE 57
log=FILE 57
ls 9

M
mount 27, 33
Mounting with FUSE
--
fuse-opt=FUSE 58
viewmode=MODE 58

N
no-default-config 45

O
one-file-system 19

P
Performance
--
dump-memory-profile=METHOD 59
memory-dump-interval=SECONDS 59
Performance tweaking
--
chunk-size=SIZE 60
chunkids-per-group=NUM 61
idpath-bits=IDPATH-BITS 60
idpath-depth=IDPATH-DEPTH 60
idpath-skip=IDPATH-SKIP 60
lru-size=SIZE 60

node-size=SIZE 60
upload-queue-size=SIZE . 60
pretend 19

R
restore 9, 28

S
SSH/SFTP
--
no-pure-paramiko 62
no-strict-ssh-hosts-keys . . 62
pure-paramiko 62
ssh-key=FILENAME 62
ssh-known-hosts=FILENAME 62
strict-ssh-host-keys 62

V
verify 33

General Index

Symbols
~/.config/obnam/*.conf 44
~/.obnam.conf 44
/etc/obnam.conf 44
/etc/obnam/*.conf 44

B
Backup
  incremental 8
  initial 8
  snapshot 14, 26

C
checkpoint 8

D
de-duplicate 23

E
encryption 9, 13

G
gpg --list-keys 38

I
Incremental backup 8
Initial backup 8

K
k4dirstat 40

L
license
  Creative Commons License 92
  GNU General Public License 92

O
Obnam
  home page 91
  original author 93
  support obnam development 93
obnam.conf 7, 21
obnam.log 7

Q
Quick introduction 7