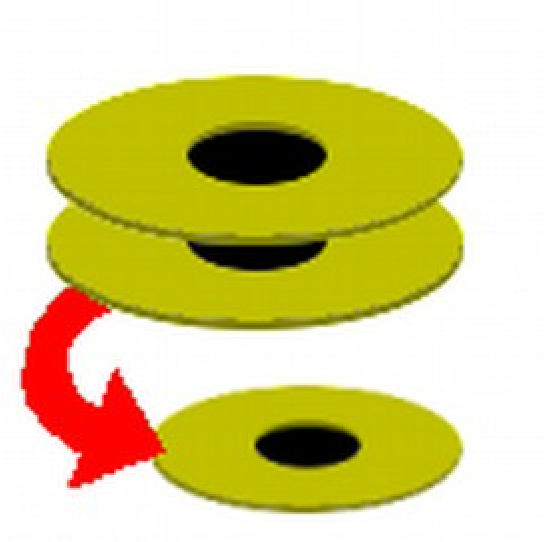
Incremental Backups with Dirvish

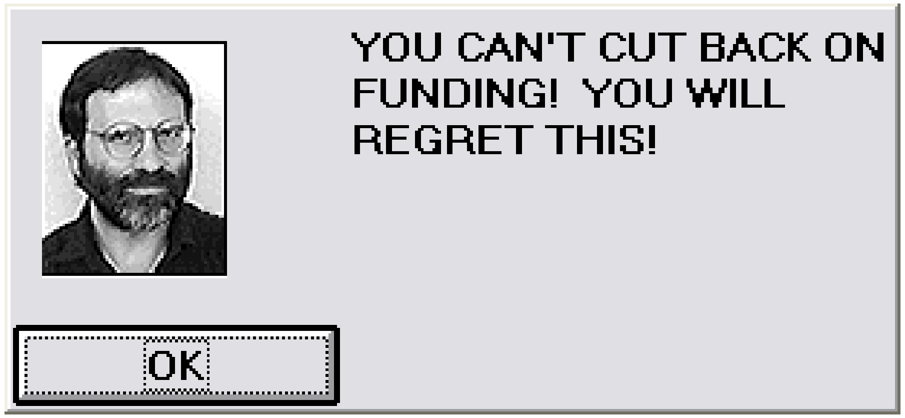


Tom Ryder
tom@sanctum.geek.nz
https://sanctum.geek.nz/

People and organisations learning how backups work go through distinct **stages**...

Stage 0: nO bAcKuPs

- Accidents happen
- Newbies toast filesystems
- Drives fail
- Laptops get stolen
- Servers get hacked



YOU CAN'T CUT BACK ON FUNDING! YOU WILL REGRET THIS!

Stage 1: Manual local backups

- Each night (when you remember), you copy all your stuff from one drive onto another:

```
$ cp -r /home/me /mnt/backup
```

- 100 times better than nothing, but...
- Boring
- Error-prone

Stage 2: Automated local backups

- You recruit `cron(8)` to run that command for you:

```
$ grep backup /etc/crontab
0 0 * * *  root  cp -r /home/me /mnt/backup
```

- systemd timers work too; pick your poison
- Much more fire-and-forget
- Send the errors somewhere useful!

Stage 3: Differential remote backups

- Network destination, preferably a different building
- Copying gigabytes of data every night is slow (and maybe expensive)
- A lot of the data will be the same as last time
- Use `rsync` to copy *only the changed parts*

```
0 0 * * *  root  rsync -a /home/me srv::bak
```

Stage 4: Incremental backups

- You need a file as it was *three days ago*, and not last night
- You only have last night’s backup; the prior state is gone forever!
- Keep backups for every day of the week:

```
0 0 * * 1  root  rsync -a /home/me srv::bak/mon
0 0 * * 2  root  rsync -a /home/me srv::bak/tue
0 0 * * 3  root  rsync -a /home/me srv::bak/wed
...
```

Stage 5: Deduplicated backups

- Backups are important, but disk space is finite
- Incremental backups fill up space fast
- Lots of **redundancy** for files that don’t change
- Store *only the changed files*
- Git does it this way, too

- But how do you get the data *out*?
- How are backups *represented*?

School of hard links

- Filesystems like ext4 support **hard links:**

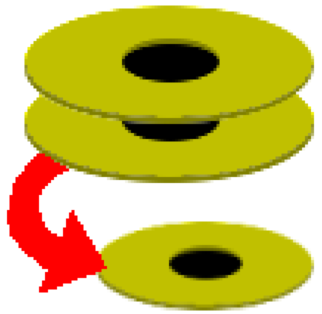
  ```
  $ ln name1 name2
  ```

  Note: no `-s` flag this time: not a **symbolic link**
  - Some similarities, though
- Two filesystem names point to the **same data**, specifically, the same **inode**
- Neither an “original” nor a “copy”
- `rsync` understands them (`-H` option)

So, since your kernel image doesn’t change day-to-day, why not store just one copy, and hard link all references to it?

Enter Dirvish

- Wrapper for `rsync(1)`'s options
- Stores backup sets in **vaults**
- Uses hard links for deduplication
- Backup is complete at filesystem level
  - Easy to explore (`cd`, `ls`, `find`...)
  - Easy to restore (`cp`, `mv`, `rsync`...)
  - Sparing on space



Dirvish vault structure 1/3

```
$ ls /bak/pc
20190708  20190709  dirvish
```

Dirvish vault structure 2/3

```
$ ls /bak/pc
20190708  20190709  dirvish
```

```
$ cat /bak/pc/dirvish/default.conf
tree: /
```

Dirvish vault structure 3/3

```
$ ls /bak/pc
20190708  20190709  dirvish
```

```
$ cat /bak/pc/dirvish/default.conf
tree: /
```

```
$ ls -i /bak/pc/20190708/tree/etc/hostname
10486452 /bak/pc/20190708/tree/etc/hostname

$ ls -i /bak/pc/20190709/tree/etc/hostname
10486452 /bak/pc/20190709/tree/etc/hostname
```

Dirvish config 1/4

```
$ cat /etc/dirvish.conf
bank:
    /bak
```

Dirvish config 2/4

```
$ cat /etc/dirvish.conf
bank:
    /bak
exclude:
    - /dev/
    - /proc/
    - /sys/
```

Dirvish config 3/4

```
$ cat /etc/dirvish.conf
bank:
    /bak
exclude:
    - /dev/
    - /proc/
    - /sys/
expire-default: +7 days
```

Dirvish config 4/4

```
$ cat /etc/dirvish.conf
bank:
    /bak
exclude:
    - /dev/
    - /proc/
    - /sys/
expire-default: +7 days
Runall:
    pc 5:30
```

Dirvish backup schedules 1/2

“I want to take backups every day. I want those backups to get deleted after they’re a week old, *except* the ones taken each Friday, which I want to keep for a month, and the ones taken on the first of the month, which I want to keep for a year. Can I automate that without writing code?”

Dirvish backup schedules 2/2

```
expire-default: +7 days
expire-rule:
    * * * * fri  +1 months
    * * 1 * *    +12 months
```

Bonus stage: Encrypted 1/2

- Ideally, encrypt at the block level
  - LUKS or dm-crypt
  - BitLocker
- Send over a trusted network or an authenticated, encrypted tunnel, to your machine in a secure location
- Transparency is a big win
- It’s just easier

Bonus stage: Encrypted 2/2

- If the backup server is managed by `$EVILCORP`, you might need something like **Duplicity**
- **Goal:** The remote server *never sees your plaintext data*
- Uses your **GnuPG** key pair
- Still incremental!
- Still deduplicated!
- Still verifiable!

The Tao of Backup: 1/7

- **Coverage:** Don’t exclude files without a really good reason.
  - “Not enough disk space” is a bad reason—get bigger disks!
  - “Not important” isn’t great, either—why do you have it in the first place?
  - **Your time** is always worth more than disk space, or the dollars to get it

The Tao of Backup: 2/7

- **Frequency:** Back stuff up with a frequency that reflects your work on that stuff.
  - Daily tends to be a happy medium.
  - Best to include weekends as well!

The Tao of Backup: 3/7

- **Separation:** Keep backups in different physical locations.
  - The more important the data, the more copies there should be, and the further apart they should be.
  - A good method for most technical people’s personal files is to back up to a local device and then (encrypted!) to cloud storage.

The Tao of Backup: 4/7

- **History:** Keep old backups as long as practical
  - This is the *raison d’être* for incremental backups
  - Disk space is a concern here
  - Deduplication can help a *lot*, depending on the data
  - Decide on a retention *schedule*. Dirvish is good at this, but other systems can do it too.
  - Don’t do this manually

The Tao of Backup: 5/7

- **Testing:** Restore the backup and reinstate the data in place
  - Does it work?
  - How long does it take?
  - Is the data the same?
    - Validity checks at backup time are helpful, but it’s not the same thing
  - Time-intensive, and a tough sell to management...

The Tao of Backup: 6/7

- **Security**: Worth backing up, likely sensitive!
  - Where is it? (Physically!) “*There is no cloud...*”
  - How can it be retrieved?
  - Who can retrieve it?
  - Which systems see it in plaintext? When?

The Tao of Backup: 7/7

- **Integrity:** Does the backup contain the data intended, bit-for-bit?
  - Verifying integrity at *backup* time
  - Verifying integrity at *restore* time
  - Plain old checksums are a good start
  - Checksum-based systems like Git can help, too

Tom’s addendum to the Tao

Monitoring!

- Send errors somewhere that someone will ***actually read them*** so that they get fixed
- Notify when the job doesn’t run *correctly*
- Notify when the job hasn’t run *at all*
  - *Quis custodiet ipsos custodes?*
- Use system mail, Dirvish hooks, and Nagios

Questions?

- **Dirvish:** http://dirvish.org/
- **Duplicity:** http://duplicity.nongnu.org/
- **Tao of Backup:** http://www.taobackup.com/

Email: tom@sanctum.geek.nz
Website: https://sanctum.geek.nz/
Twitter: @tejrnz