|  |
| --- |
| Leveraging rsync  A Local-cache Alternative to Time Machine |

Abstract

Systems fail.

This document covers two alternatives available to macOS users to make strategic backups.

A. The macOS feature: Time Machine.

Affords to the most comprehensive backup strategy – at a system-wide scope (p2).

B. The canonical ‘nix utility/filter rsync.

Affords the most flexible backup strategy – scope is not constrained (p5).

The rsync filter can optionally be launched at user-customizable intervals, using the ‘nix canonical filter crontab, while the macOS Time Machine launches at fixed intervals.

Both of the above approaches accommodate ad hoc launches of a backup.

Both of the above approaches use the concept of DELTAS to conserve disk space.

NB

Out of scope for this document is IntelliJ’s local history feature, which operates on a more granular scale as compared to the other two backup approaches. IntelliJ backups have fundamentally the same objective as the other two back mechanisms (but operate in a very limited arena).

ref <https://www.jetbrains.com/help/idea/local-history.html#restore-changes-from-local-history>



A. Time Machine

Can be used profitably but is ideally designed to operate by leveraging an external device as its backing store. This obviates the risk of data loss in the event of an irretrievable system fail.

<https://support.apple.com/en-us/HT201250>

For Time Machine, I use as backup storage hardware, a *fast* USB 3.0 flash drive medium. The storage capacity of the drive is 127 GB, and the peak transfer rate is 350 MB/sec.

The host computer’s SSD hard drive has a maximum capacity of 250 GB, and currently has used just 27% of that capacity (67 GB).

So, for the time being I have a safety factor of 1.7. With the delta approach that Time Machine adopts, it is not likely I will run out of back up capacity shortly before I double SSD usage (1.7 x 67GB ≈ 114 GB).

Excluding the backing up of Mac OS’ system files saves 1 GB.



Time Machine | Periodicity OF

Unlike crontab the backup intervals for Time Machine are fixed. You must keep your backup medium mounted (attached) to the host (MacBook) at all times, which can be difficult at times.

Graphical user interface, application

Description automatically generated



Time Machine | In Flight

Below are two views of the state of the Time Machine in-flight, followed by a sanity check on the host SSD’s (transient) characteristics.

Table

Description automatically generated with medium confidence

Graphical user interface, application, Teams

Description automatically generated

Reports 2.08 GB of the flash drive’s capacity has been consumed.

Text

Description automatically generated

Reports 1.8 GB of the flash drive’s capacity has been consumed.

The discrepancy in the values of the two reported measurements (above) is not due to the size difference between Gibibyte v Gigabyte.

Graphical user interface, application

Description automatically generated

B. Local Cache | rsync

Line 1787

The crux of the rsync filter’s caching logic is as follows.

a. Synchronize changes made in the (implied) current directory – the name of which is

dynamically stipulated by the value of the canonical ßash shell variable ($1) – with

the explicitly-designated (must be accessible) caching directory, in this case:

/var/tmp/exchange/

b. Stipulate the non-remote rsh shell program (in preference to the default/remote ssh shell program).

c. Show progress, but send any complaints to the OS canonical null device:

2>/dev/null

Text

Description automatically generated

NB Consider adding this logic after line 1780 to assure that the directory exists:

mkdir -p /var/tmp/exchange/

If the directory exists, nothing happens; if it does not, creating it avoids an error.

**Legend**

**-a** Archive recursively

**-v** Verbose

**-z** Compress file data during transfer

**-h** Human-readable numeral output

**-e** Choose another remote shell program for communication

between local/remote copies of rsync than default (ssh)

A local network can use the rsh remote shell program

**--progress**

show progress during file transfer

ref <https://linux.die.net/man/1/rsync>­­­­



C. In Situ | rsync

**One**

Initially, each directory/file contains the same data.

The modified data transfers from left-to-right – from the $HOME/Desktop/code/code file, into the /var/tmp/exchange/code/code file.

A screenshot of a computer

Description automatically generated with medium confidence



**Two**

A change/addition to the desktop file – $HOME/Desktop/code/code – has been made.

Text

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated



**Three**

The depo() ßash function is coded to find the named the artifact(s) – stipulated by the $1 variable –

in the current directory.

Because we want to transfer the file code (which is inside the directory ~/Desktop/code/) that is where we navigate to make the invocation of the depo() ßash function.

Text

Description automatically generated



**Four**

The change/addition is reflected in the cached file.

A screenshot of a computer

Description automatically generated



­­­­­

foldersize | A Bash function, uses du to list the size-on-disk of folders/files

A screenshot of a computer

Description automatically generated with medium confidence



caption | A Bash function, imprints a Terminal with a title

A screenshot of a computer

Description automatically generated with medium confidence

