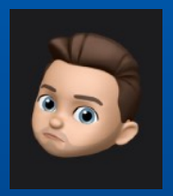
Time Machine - Back to the Future

A New guide to Old backups



James Bousfield
Operations Manager & Infrastructure Consultant
Twitter - @jbousfield53
MacAdmins - @jcbousfield

https://qrgo.page.link/qmAbt

moof IT

Firstly - Why bother?

This is entirely based on our experience and requirements, but MOST (not all) 3rd party backup tools do the following when backing up data:

- Store Data in a proprietary format
- Data Can often be compressed
- Can be VERY expensive when you start storing large quantities of data
- Restoration of data can be difficult or very time consuming

moof IT

By storing in a proprietary format, the software is required to understand the data before you can even begin the restore

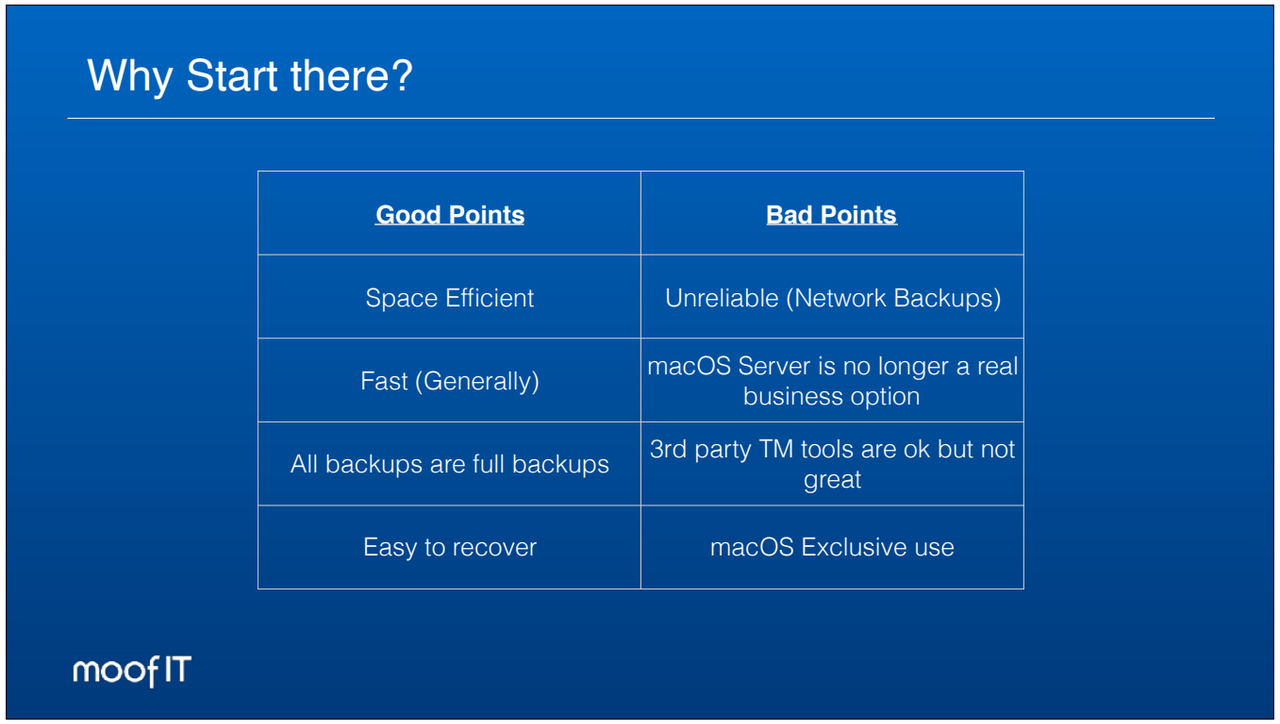
Compression of data is great for storage, but when restoring it usually makes the process much longer than if it was stored in plain format.

Some 3rd parties become extremely expensive when you get above a few TB's, ~£300-£400 Per month per TB backed up to their service.

Where is your data stored? On what format? Deep Storage or linked tapes?

Where we started

Time machine



We all know that time machine has its own issues but the idea of time machine is great… space efficient, generally fast, can be encrypted and every backup is stored as a “full” backup. The upside of all of this is that the data is easy to restore.

The Apple Server app no longer supports Time machine over networks, however this can be completed in the System Preferences Pane now. 3rd Party tools have replicated the service but they aren’t overly reliable.

The ideal backup solution would be available for use on all platforms and be accessible from any device that is authorised to access the solution.

Security Considerations before we start

- Data should always be encrypted at all points
  - Transit - Over SSH
  - At Rest - Luks SHA-256 Encrypted Virtual Disks,
  - Dont use standard ports for access

moofIT

SSH - uses both asymmetric and symmetric cryptographic algorithms to ensure that the data is securely encrypted and transferred at optimal speeds

Encryption - The type of encryption available depends on the host you are installing on. We use Linux and there are lots more options are also available

If publishing over the internet ensure you use SSH Keys and known hosts rather than Username and Passwords. Restrict your ability to log in directly to the console to a physical keyboard attached or from a known safe IP (Head office) NEVER use port 22 for SSH for any public facing services, if you have username/password authorisation allowed over the internet.

Rsync

- Native to macOS / Many Linux Distro's
- Generally quick
- Merges differences
- Clone of Data in place

-r	Recurse entire folder structure
-h	Output numbers in human readable format
-l	Follow links and grab that data too
-H	Preserve any encountered hard links

```
rsync  -rhlH $source $destination
```

moof IT

All macOS and OSX has had rsync included since its included in the UNIX layer of the OS.
The speed of the data transfer is dependent on the connection rate between the source and destination and can be affected by the storage used on the destination.
When used alone RSync compares files in place with files you are transferring to ensure the data is the same in both places.

This would replicate the folder to another location on the same device, not great as a backup as the device would still be the only source of this information

Completing the backup over a network / internet

- SSH

-e	Execute Rsync over “protocol”

```
rsync  -rhlH -e “ssh -p $sshport” $source $username@$server:/$path/
```

moof IT

If you wanna run over the internet or to a remote destination

This would complete a backup from a device to another device on the same network or to a remote IP/DNS name over the internet if you have configured it to receive information

Whats wrong with this?
A replica has its uses but wouldn’t be classed as a backup, other than the fact that the data exists in a second location

Adding the Time Machine Elements

- Date Versioning

```
folder=$(date +%Y-%m%d-%H%M)
```

- Hard linking (Space Efficiency)

```
ssh -p $port $UserName@$Server "rm $Path/latest"
ssh -p $port $username@$Server "ln -s $path/$folder $path/latest”
```

```
rsync --progress -rhlH -e "ssh -p $port" —link-dest=$path/latest "$source"
$username@$server:$/path/$folder
```

moof IT

To make it a “proper” backup we need to start adding some of the bits we were trying to replicate from the good bits from time machine.

So data versioning would mean that there is multiple restore points and Hard linking will add the ability to only backup a file once, unless it has been changed in any way. This would add a couple of the points we were looking for. When done together, this comes into the script like so:

Making the backup Faster

- Adding the Size Only Check
- - -sizeonly

- Dont backup the non required files (.afp_deleted etc)
- --filter='merge /Library/Backups/exclusions.txt'

```
folder=$(date +%Y-%m%d-%H%M)

rsync --progress --size-only -rhlH --filter='merge /Library/Backups/exclusions.txt' -e "ssh -p $port"
--link-dest=$path/latest "$source" $username@$Server:$path/$folder
```

moof IT

Typical Exclusions would be something like:
usr
var
.Trashes
.TemporaryItems
MacintoshHD
.afpDeleted*

Adding some integrity checks

- Check the Server Fingerprint matches the known_hosts file:

```
if [[ $(ssh-keyscan -p $port -t ecdsa-sha2-nistp256 $Server 2>/dev/null | awk
'{print $3}') == “$fingerprint Key” ]]
then
    writelog "Server fingerprint check: PASSED"
else
    writelog "##### ERROR ##### Server fingerprint mismatch.  This isn't the
server you are looking for...  Exiting with error code 1"
    exit 1
fi
```

moof IT

To build on the security of things, we would generally like to check the fingerprint of the server to ensure that it hasn’t been changed or that someone is trying to mimic the server to steal all the information. By using the SHA-256 key, the time it would take to break a SHA key is 10 * 3.92 * 10^56 minutes.

Completed Script

```
################################################################################
####                        Check server is the right one                   ####
################################################################################

if [[ $(ssh-keyscan -p $port -t ecdsa-sha2-nistp256 $Server 2>/dev/null | awk '{print $3}') == "$fingerprint" ]]
then
    writelog "Server fingerprint check: PASSED"
else
    writelog "##### ERROR ##### Server fingerprint mismatch.  This isn't the server you are looking for...  Exiting with error
code 1"
    exit 1
fi

################################################################################
####                             Run the backup                             ####
################################################################################


writelog "Running sync"
rsync --size-only -rhlH --filter='merge /Library/Backups/exclusions.txt' -e "ssh -p $port" --link-dest=$path/latest "$source"
$username@$Server:$path/$folder


writelog "Sync complete, creating latest symlink"
ssh -p $port $username@$Server "rm $path/latest"
ssh -p $port $username@$Server "ln -s $path/$folder $path/latest"
```

Working from the top down:

Finger print check… does it match, Yes —> Move on to the script

No —> Bin it off and throw a Jedi error

Start the backup - run with size only, dont backup the rubbish files using this txt file for reference, backup over ssh using this port, link to the “Latest” backup, from this

What about Windows?

Rsync can be installed using some 3rd party tools:

- Libssh2
- OpenSSL

https://github.com/gilbertchen/acrosync-library

moof IT

Thanks

Q & A

https://qrgo.page.link/qmAbt