Tom Bridge

Partner at Technolutionary LLC

Author of Munki-in-a-Box

Co-host of MacAdmins.org Podcast

Backing up Cloud Servers, and Remote Storage

The Tao of Backup

What are we going to cover?

- Servers and Storage in the Cloud need backup
- Service provider backup is good, but you need more
- What types of backups?
  - work-in-progress
  - archiving closed dockets
  - real-time synchro for two live locations

Build vs. Buy

- rsync vs. …
  - Sync software (e.g., Synchronize Pro, BitTorrent Sync)
  - Time Machine
  - CrashPlan (and others)
- If off the shelf, use the off the shelf
- If needs to be customized, or you don’t want the overhead of more software, then customize

Automation vs. User Interaction

- Choose the level of automation vs. user interaction
  - Automatic: Cron/launchd
  - Convenient: Password-built-in applet
  - Security: Password-required applet
- How safe is your process from outsiders potentially disrupting or hijacking it?
  - Don't store passwords in clear text
  - Limit your backup servers to only allow specific IPs or users to send data to specific locations
  - Consider security at all times

Automation vs. User Interaction

- Automation comes with increased risk of…
  - Simple mistakes having dire side effects
  - Not noticing a problem
- User interaction has risk of people not doing what they committed to do
- Lesser of two evils; choose and manage

Reminder:
Sync is NOT a Backup Unto Itself

Command Line Primer

- Construction of a command:
  - command
  - options & switches
  - source and target
- Capitalization matters (case sensitivity)
- Read and understand the tools, using man pages
- Choice of editors don’t insert unhelpful formatting
  - nano, vi, etc…
- Version and compatibility of tools

rsync

- Does a "differential" backup where only the chunks of files that are different are transferred
- Extremely efficient for copying large trees of files when only a few modifications have been made
- Not generally more effective for binary files such as DMGs, zip files, etc, because these tend to change the entire file if something inside is different
- Works locally and to remote servers (can work over SSH if remote server doesn't have rsync installed)

rsync in OS X

- Default OS X version old and deprecated: 2.6.9
- Current rsync version is 3.1.2
- As of 10.11 El Capitan, /usr/bin/rsync cannot be replaced or modified (due to SIP)
  - … and it's still 2.6.9

Obtaining Updated rsync

- Download new rsync to /usr/local/bin/rsync:

```
curl -O http://rsync.samba.org/ftp/rsync/
rsync-3.1.2.tar.gz
tar -xzvf rsync-3.1.2.tar.gz
rm rsync-3.1.2.tar.gz
cd rsync-3.1.2
./prepare-source
./configure
make
sudo make install
```

- Compare:

```
/usr/local/bin/rsync --version
/usr/bin/rsync --version
```

man rsync

```
NAME
  rsync - faster, flexible replacement for rcp

SYNOPSIS
  rsync [OPTION]... SRC [SRC]... DEST
  rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST
  rsync [OPTION]... SRC [SRC]... [USER@]HOST::DEST
  rsync [OPTION]... SRC [SRC]... rsync://[USER@]HOST[:PORT]/DEST
  rsync [OPTION]... SRC
  rsync [OPTION]... [USER@]HOST:SRC [DEST]
  rsync [OPTION]... [USER@]HOST::SRC [DEST]
  rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]
```

Common rsync options

- **`-P`**: show partial progress bar
- **`-h`**: human-readable numbers
- **`-a`**: "archive" mode copies nearly everything as-is
- **`-v`**: verbosity
- **`-z`**: compression
- `rsync -Phavz source dest`
  - Shows progress, human readable file sizes, copies mostly everything, shows every file being copied, and compresses for lower bandwidth

Basic rsync usage

- `rsync -a SOURCE DESTINATION`
- `rsync -a /path/to/source /path/to/dest`
  - Will copy the folder "source" to "dest"
- `rsync -a source username@destination:/path/to/server/folder/`

Classic rsync

- Copy content of "files" to destination on server:

```
rsync -avz  /path/to/files/  user:password-
if-you-want-saved@host:/path/to/target/
```

- Copy folder "source" to server as "dest", showing progress in human readable format:

```
rsync -Phavz /path/to/source user@host:/path/
to/dest
```

Power of Script

- The real power comes with stringing multiple commands together
- Allows for no dark-time between task 1 completing and the beginning of task 2
- Allows for the accurate repetition of task n, for as many boring times as you need it to be done

Reminder: Good Habits

- When scripting, use some form of version mgmt
  - name/number your scripts as you develop them
  - keeping light notes about what each version adds
  - source control - upload to GitHub or BitBucket
- Output should help you track
  - have your script create embed a header line in any output files, quoting the script version
- Alternatively … Breadcrumbs
  - Have scripts create (or check for, causing different outcomes) hidden ‘breadcrumb’ files (empty, but fixed name starts with a period)

Reminder: Good Habits

- Add or change as few things as possible (1 or 2) when editing or building a script, lest you break A while building B
- Consider embedding occasional screen output lines through your script, to provide some feedback if your script hangs or fails midway through
- Use version/source control frequently so you can always revert to previous revisions, especially if making big changes

“touch”

- `touch /path/of/interest/.last_rsync_time`
- Will either create (or modify the last modified time stamp on) a file
- Can be used to mark files to backup on next run

Choices

- When choosing how to best protect the source data, consider the behavior of it.
- How fast does it change, vs. how long do you need it to stay still?
- It’s not about backup, it’s about restore
  - Is the restore accurate and useable

Think Photo

- You want to create a perfect "image" of the dataset
- Think about a photograph of a large, active park
- While you use iPhone's panoramic panning, people walk in the direction of your pan.
  - you now end up with the same person in the picture 3 or 4 times
  - if walking the other way, you might end up with just a partial, unrecognizable blur

Files Blur Too

- Same with live files such as:
  - large PST file or database
  - log files
  - large media file
- If the index to captured at block 1 is out of date before you're at block n, the restored file may not work
- Solution: always use application-aware tools or commands to 'dump' a database or shadow file, then backup that flat / static file.

Real Examples

- MySQL / PostgreSQL is a good example of this "blur"
- Live log files are another example
- Backups like CrashPlan, Time Machine, or sync applications won't handle properly

MySQL: Backing it up

Backup every day, keep for a week

```
#!/bin/bash

# Clean up older backups, anything older than 7 days
find /Users/backup/ -mtime +7 -delete

# Get all databases
DUMPNAME=/Users/backup/fulldump-`date "+%Y%m%d"`.sql
/usr/bin/mysqldump -u root --
password="paswordgoeshere" --all-databases > $
{DUMPNAME}
ln -sf ${DUMPNAME} /Users/backup/latest-fulldump.sql
```

And, the rest of the site?

```
# Get the site
SITENAME=/Users/backup/www_site-`date "+%Y%m
%d"`.tar.gz
tar czvf ${SITENAME} /Library/WebServer/Documents/
db.companydomain.com
ln -sf ${SITENAME} /Users/backup/latest-site.tar.gz

# Get local configs
USRLOCALNAME=/backup/usr_local-`date "+%Y%m%d"`.tar.gz
tar czvf ${USRLOCALNAME} /usr/local
ln -sf ${USRLOCALNAME} /backup/latest-usrlocal.tar.gz
```

Behavior at Both Ends

- Identifying the desired behavior at the source … and the target
    - Example: Move the daily backups to a weekly set
- Use pattern (regex) matching and how to select the right source data
- Beware of the rogue loop with the best of intentions but the simplest of syntax errors
- Test regex filter syntax in a non-destructive way

Moving Data

- You may want to script moving data at source, or destination
  - Example: Move some data in a source path, while leaving others (move out files older than 30 days)
- Do you want to MOVE files (remove from source) vs. COPY (leave at source)?
- Do you want to segregate or notate files at the source, if choosing to leave them on the volume (move into a ‘processed’ folder, or change filename)?

Example: Moving Older Files

Moving files that are more than 180 days (only) from one location to another.

```
# Create list of affected files -- find all the
files that are 180 days ago or more
$ find . -type f -ctime +180 -print > ${TMPLIST}

# Sync to the other box (need to define variables)
#   {EXCH_USER}  ${EXCH_HOST}   ${TARGET_DIR}
$ rsync -avz --files-from=${TMPLIST} ./ $
{EXCH_USER}@${EXCH_HOST}:${TARGET_DIR}/

# Delete after successful sync
# Dangerous -- be careful
$ cat ${TMPLIST}|xargs rm -f
```

rsync’s Clean Up Flags

Another way to do the clean up is to use rsync’s built-in clean up flags such as:

```
-e  flags of interest...
   --remove-source-files   sender removes synchronized files (non-dir)
   --delete                delete extraneous files from dest dirs
   --delete-before         receiver deletes before transfer (default)
   --delete-during         receiver deletes during xfer, not before
   --delete-after          receiver deletes after transfer, not before
   --delete-excluded       also delete excluded files from dest dirs
```

rsync’s Clean Up Flags

So, if you want to move backups from one machine to another, you might do:

```
rsync -avz -e --remove-source-files  --delete-after
/path/to/backupfiles     user:password-if-you-want-it-
saved@host:/path/to/targetbackupfiles
```

Excluding Files

- Sometimes, there are files you don't want:
  - OS Specific files (e.g., `.DS_STORE`)
  - Installation specific (e.g., configuration paths)
  - Temp or Cache files
  - Source control files (.git, .hg)
- rsync flags to the rescue, exclude any file with 'dir1':

```
rsync --exclude 'dir1'
```

- 6 rsync Examples to Exclude Files and Folders

  `http://www.thegeekstuff.com/2011/01/rsync-exclude-files-and-folders/`

Additional Considerations

- Outcome / confirmation and documentation of a script's completion
- Add logging?
- Do you want to append a date/time stamped summary line to a fixed-path log file?
- Add reporting or notification?
- Do you want the script to email you?

Real Examples

- Kerio Example:
    - Backing up a live server
    - Having an easy fail over, albeit manual
- Web Server (e.g., WordPress)
    - Backing up a live server
    - Database, Files, Config

Scripting the Total Solution

- Automating rsync and mysql
- Scheduling (cron or whatever)
- Logs
- Notifications and issues
- Email a report

Trust and Verify

- Now that you’ve built a script or application to move the data, you’re not wrong to want to trust it. That’s not a bad thing - but trust AND VERIFY.
- Take the time for periodic validation that you can successfully recover from the hypothetical threat that you seek protection from
- Establish what ‘normal’ behavior looks like (time to run, size of output, etc.) and determine how you’ll detect off-nominal developments
- TEST A FULL RESTORE!

Test, before going live

- Test regex filter syntax in a non-destructive way
- Regex Testing Tools
  - http://regexr.com
  - http://ryanswanson.com/regexp/#start
  - http://www.regexpal.com
- Additional Regex tools:
  - http://www.cheatography.com/davechild/cheat-sheets/regular-expressions/
  - http://regexlib.com/CheatSheet.aspx
  - http://www.virtuosimedia.com/dev/php/37-tested-php-perl-and-javascript-regular-expressions
  - http://code.tutsplus.com/tutorials/8-regular-expressions-you-should-know--net-6149

Incremental and Differential

- Incremental
  - Monday: Full backup
  - Tuesday: backs up changes between Mon and T
  - Wed: backs up changes between T and W
  - Thurs: backs up changes between W and Th.
- Differential
  - Monday: Full
  - Tues: backs up changes between M and T
  - Wed: backs up changes between M and W
  - Thur: backs up changes between M and Th.

Incremental vs. Differential

- Incremental allows one restore from the date of latest incremental to get everything.
- Differential requires you restore Monday then Tues then Wed then Thurs.

Questions?

Tom Bridge
tom@technolutionary.com

Special Thanks to:
Sean Costello
Background Backup