SimplStor

NETWORK STORAGE SOLUTIONS BY ZEROWAIT

White Paper

Rsync Data Replication

Date: August 30, 2017

Data Replication

Many SimplStor customers are interested in using replication technology to maintain multiple copies of data, either onsite or at a remote location. Customers who are familiar with Network Appliance Data ONTAP storage systems are familiar with 'snapmirror'. While 'snapmirror' itself is a proprietary technology, the open source utility 'rsync' can provide similar results and comparable replication speeds on Linux and Unix systems. This white paper describes how to setup 'rsync' to provide replicated data between multiple Linux systems. We will also describe a method of synchronizing backups of NetApp volumes.

Rsync Background[1]

Rsync was first announced and released in 1996 by Andrew Tridgell and is currently maintained as part of the Samba Project (https://rsync.samba.org). Rsync is a file transfer program capable of efficient remote update via a fast differencing algorithm. Rsync is commonly found on Unix and Linux systems and functions as both a file synchronization and file transfer program. The rsync algorithm uses a type of delta encoding at the pseudo-block level that minimizes the amount of data that is transferred, and additionally can use compression and encryption of the data stream.

Rsync is typically used for synchronizing files and directories between two different systems. For example, if the command `rsync local-file user@remote-host:remote-file` is run, rsync will use SSH to connect as `user` to `remote-host`. Once connected, it will invoke the remote host's rsync and then the two programs will determine what parts of the file need to be transferred over the connection. Rsync can also be used between two directories on the same system, providing a local incremental copy command.

Rsync can be used to either "push" or "pull" files between systems, so where this is controlled is up to the implementer. Rsync will use SSH as the underlying remote shell protocol or is can be configured as a daemon service.

Enterprise Linux

Nearly all modern Linux and Unix systems include Rsync as either a standard component or an easily obtained option. If needed, it can be compiled and installed from source (https://rsync.samba.org/download.html), but this is generally not necessary. On Enterprise Linux (RHEL and CentOS) it can be installed with 'yum'.

```
    # yum install rsync
```

[1] https://en.wikipedia.org/wiki/Rsync

Both EL6 and EL7 utilize version 3.x of rsync. It is recommended that rsync v3.x be used on all systems as this includes the addition of an incremental recursion scan (which helps with large transfers), ACL support, extended attribute support, filename character-set conversion, and other desirable features. The command syntax described below is independent of the version and should be universal across most Linux and Unix systems.

The general syntax of an rsync command is:

```
        $ rsync [OPTIONS] [SOURCE] [DESTINATION]
```

The SOURCE and DESTINATION specifications can include remote hosts and authentication parameters as well as folder and file specifications. Examples are:

```
        relative/path/to/folder
        /path/to/folder
        /path/to/folder/file
        hostname:/path/to/folder
        user@hostname:/path/to/folder
```

There are many options that can be specified when using rsync. Please refer to the online documentation (`man rsync`) for details. A summary of options and usage can be obtained by entering “`rsync --help`” (see appendix below).

Some typical options are:

Archive mode:

```
        $ rsync -a [SOURCE] [DESTINATION]
```

Archive mode and compress (used for network transfers):

```
        $ rsync -ac [SOURCE] [DESTINATION]
```

Archive, compress and provide a verbose listing and summary of files being transferred:

```
        $ rsync -acv [SOURCE] [DESTINATION]
```

Rsync “archive” mode will recursively copy files and directories while maintaining links, device and special files, timestamps, permissions and user/group ownerships.

Rsync will utilize secure shell as an underlying secure transport over the network and the use of public key authentication is assumed in the descriptions below. If you do not know how to create and use ssh keys, see the simple example in the appendix at the end of this paper.

Synchronizing Folders

In its simplest form, rsync can be used by any individual user to synchronize the contents of folders between systems. In fact, rsync can be used to synchronize folders on the same system, creating a local backup. Files that are readable by the user on the source system will be copied to the target system. Files on the target system may have a different ownership if these commands are run by a non-root user.

Case 1: Synchronizing folders between two systems: This command will assure that the contents of a local folder called /images/store are replicated on a remote system to a folder relative to the users home directory.

```
$ rsync -ac --delete /images/store/ user@remote.company.com:images/prod
```

Because the source folder designation ends with a "/", only the contents of the /image/store folder are copied, not the "store" folder itself. If the "/" is left off, then the target would create images/prod/store/ as well as all the files. The "--delete" option assures that the target folder does not contain any extraneous files that are not present in the source folder. Leaving the "--delete" option off will result in copying all files from the source, but extra files (from previous synchronizations or copied in other ways) may be present on the target.

Case 2: Copy a remote users home directory to the local system, but do not erasing any local files.

```
$ rsync -ac hostinguser@hostingservice.net:/home/user   /data/host
```

This command will recreate the "user" folder and all of its contents in the local /data/host folder. If the local /data/host/user folder contains additional files, they will remain in addition to the files copied from the remote system. Files in the /data/host/user folder will be owned by the user who invoked the command. If this command is run as the "root" user, the UID and GID ownership of the files from the source system will be retained.

Backing Up Client Systems

Backing up complete client systems is just a special case of synchronizing folders. When you use rsync to synchronize whole systems you need to be aware of how UIDs and GIDs are handled between systems, and you need to take care in how you specify the target location so that files do not get improperly intermingled. Use of the "root" account is best suited to this type of operation since it will retain the correct ownership on the backup system.

Case 3: "Pull" a backup of the /home directories on a remote system to a local folder.

```
# rsync -ac --delete  host1:/home  /data/host1
```

As long as this is run from the “root” account, it will create an exact copy of the remote /home directory in /data/host1/home, retaining the proper UID and GID ownership of files.

Case 4: “Push” a backup of the /home directories on a local system to a remote backup host. This performs the same backup as the previous case except that it is “pushed” from the client.

```
# rsync -ac --delete /home  backuphost:/data/host1
```

Note that if the “host1” directory does not yet exist on the backuphost, it will be created for you. As above, the resulting backup will be in /data/host1/home.

Case 5: “Push” a backup of the /mnt directory on the local system, but do not descend or cross into any file system mount points (-x).

```
# rsync -ac -x --delete /mnt  backuphost:/data/host1
```

Backing Up Remote Mounts

Some systems may not be able to run rsync for one reason or another, but if you can mount the source system so that it appears as a folder on the target you can use rsync to synchronize the mounted files to a local directory. A good example of this would be creating a backup of a NetApp Filer Volume on a SimplStor system. Since the algorithm used to compare files by rsync is based on differences in pseudo-blocks, this method is comparable in efficiency to NetApp’s SnapMirror. If you utilize a SimplStor file system that incorporates its own snapshots, you can also simulate SnapMirror snapshots.

Case 6: In our example, we will mount the remote systems volumes on a local mount point called /mnt/filer1/. Using NFSv4, all of the exported volumes on filer1 can be mounted to this single mount point with one entry in the /etc/fstab file:

```
filer1:/vol   /mnt/filer1    nfs     vers=4   0 0
```

The resulting folders in the /mnt/filer1 directory will look something like this:

```
# ls -l /mnt/filer1/
d---------.  6 root root  4096 Dec  2  2014 root
d---------.  3 root root  4096 Apr 11 18:30 vol1
drwxrwxrwx. 30 root 1003 20480 Aug 21 06:16 vol2
d---------.  6 root root  4096 Sep 30  2016 vol3
```

We can now use rsync to synchronize a copy of these mount points from one directory to another. We use the “--exclude” option to prevent the copying of the NetApp .snapshot directory to the archive location.

```
      # rsync -a --delete --exclude .snapshot /mnt/filer1/root  /data/filer1/
      # rsync -a --delete --exclude .snapshot /mnt/filer1/vol1  /data/filer1/
      # rsync -a --delete --exclude .snapshot /mnt/filer1/vol2  /data/filer1/
      # rsync -a --delete --exclude .snapshot /mnt/filer1/vol3  /data/filer1/
```

Or

```
      # rsync -a --delete --exclude .snapshot /mnt/filer1/  /data/filer1/
```

These commands will result in equivalent results. If you are planning to run these via cron however, doing these individually will provide more flexibility in scheduling.

Case 7: An alternate approach would be to create an rsync copy of one of the snapshots that is available on the NetApp volume. This would exclude the potential impact of file changes on an active NetApp volume.

```
      # rsync -a --delete /mnt/filer1/vol1/.snapshot/nightly.0/  \
                          /data/filer1/vol1
```

This command would create an rsync copy of the nightly.0 snapshot of the /vol/vol1 volume of the NetApp filer.

Apendix 1: OpenSSH public key authentication setup

OpenSSH keys are used to authenticate an ssh connection from a source system to a target system. A public key pair is created on the source system, then the public half of the key is appended to the authorize_keys file in the desired account on the target system. The command steps to accomplish this are summarized below. While a passphrase will provide significant security advantages, you must also understand how to use “keychains” in order to use passphrase protected keys for automated operations. For simplicity, we use a blank (null) passphrase in the steps below.

1) Create the public key pair:

```
[source]$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/myuser/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/myuser/.ssh/id_dsa.
Your public key has been saved in /home/myuser/.ssh/id_dsa.pub.
The key fingerprint is:
36:43:39:20:3b:64:f5:f3:f8:04:d9:74:22:05:a3:04 myuser@source.host.net
The key's randomart image is:
+--[ DSA 1024]----+
|    Eo+ +o+ .    |
|   o + + B o     |
|    o . O .      |
|     . . *       |
|        S o      |
|       . =       |
|          .      |
|                 |
|                 |
+-----------------+
[source]$ ls -l ~/.ssh
total 12
-rw-------. 1 myuser myuser  668 Sep  1 19:37 id_dsa
-rw-r--r--. 1 myuser myuser  610 Sep  1 19:37 id_dsa.pub
-rw-r--r--. 1 myuser myuser 1517 Mar  3  2015 known_hosts
```

2) Now copy the public key to the desired account on the target system and append it to the authorized_keys file in the ~/.ssh directory:

```
[source]$ cat ~/.ssh/id_dsa.pub | ssh remuser@target.host.net \
          "cat >> .ssh/authorized_keys"
```

```
The authenticity of host 'target.host.net (fe80::250:56ff:fea3:4eca%ens160)'
can't be established.
ECDSA key fingerprint is 32:bc:a7:e2:34:06:7c:ad:b8:c5:c3:83:d4:1f:55:72.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'target.host.net,fe80::250:56ff:fea3:4eca%ens160'
(ECDSA) to the list of known hosts.
remuser@target.host.net's password:
[source]$
```

3) You will only need to accept the authenticity of the target host once on the first connection. After providing the password for the remote user account, the public key is appended to the target authorized_keys file and subsequent ssh connections will be authenticated by the keys.

```
[source]$ ssh remuser@target.host.net
[target]$
```

Apendix 2: rsync options

```
rsync  version 3.0.9  protocol version 30
Copyright (C) 1996-2011 by Andrew Tridgell, Wayne Davison, and others.
Web site: http://rsync.samba.org/
Capabilities:
    64-bit files, 64-bit inums, 64-bit timestamps, 64-bit long ints,
    socketpairs, hardlinks, symlinks, IPv6, batchfiles, inplace,
    append, ACLs, xattrs, iconv, symtimes

rsync comes with ABSOLUTELY NO WARRANTY.  This is free software, and you
are welcome to redistribute it under certain conditions.  See the GNU
General Public Licence for details.

rsync is a file transfer program capable of efficient remote update
via a fast differencing algorithm.

Usage: rsync [OPTION]... SRC [SRC]... DEST
  or   rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST
  or   rsync [OPTION]... SRC [SRC]... [USER@]HOST::DEST
  or   rsync [OPTION]... SRC [SRC]... rsync://[USER@]HOST[:PORT]/DEST
  or   rsync [OPTION]... [USER@]HOST:SRC [DEST]
  or   rsync [OPTION]... [USER@]HOST::SRC [DEST]
  or   rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]
The ':' usages connect via remote shell, while '::' & 'rsync://' usages connect
to an rsync daemon, and require SRC or DEST to start with a module name.

Options
 -v, --verbose               increase verbosity
 -q, --quiet                 suppress non-error messages
     --no-motd               suppress daemon-mode MOTD (see manpage caveat)
 -c, --checksum              skip based on checksum, not mod-time & size
 -a, --archive               archive mode; equals -rlptgoD (no -H,-A,-X)
     --no-OPTION             turn off an implied OPTION (e.g. --no-D)
 -r, --recursive             recurse into directories
 -R, --relative              use relative path names
     --no-implied-dirs       don't send implied dirs with --relative
 -b, --backup                make backups (see --suffix & --backup-dir)
     --backup-dir=DIR        make backups into hierarchy based in DIR
     --suffix=SUFFIX         set backup suffix (default ~ w/o --backup-dir)
 -u, --update                skip files that are newer on the receiver
     --inplace               update destination files in-place (SEE MAN PAGE)
     --append                append data onto shorter files
     --append-verify         like --append, but with old data in file checksum
 -d, --dirs                  transfer directories without recursing
 -l, --links                 copy symlinks as symlinks
 -L, --copy-links            transform symlink into referent file/dir
     --copy-unsafe-links     only "unsafe" symlinks are transformed
     --safe-links            ignore symlinks that point outside the source tree
 -k, --copy-dirlinks         transform symlink to a dir into referent dir
 -K, --keep-dirlinks         treat symlinked dir on receiver as dir
 -H, --hard-links            preserve hard links
 -p, --perms                 preserve permissions
 -E, --executability         preserve the file's executability
```

SimplStor Division: sales@simplstor.com support@simplstor.com

All rights reserved. The information contained herein is subject to change without notice. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws. SimplStor is a trademark of Zerowait Corporation. The Zerowait Logo, ZHA, and Zerowait are registered trademarks of the Zerowait Corporation. ©2017 Zerowait Corporation. Page 9

```
    --chmod=CHMOD           affect file and/or directory permissions
-A, --acls                  preserve ACLs (implies --perms)
-X, --xattrs                preserve extended attributes
-o, --owner                 preserve owner (super-user only)
-g, --group                 preserve group
    --devices               preserve device files (super-user only)
    --copy-devices          copy device contents as regular file
    --specials              preserve special files
-D                          same as --devices --specials
-t, --times                 preserve modification times
-O, --omit-dir-times        omit directories from --times
    --super                 receiver attempts super-user activities
    --fake-super            store/recover privileged attrs using xattrs
-S, --sparse                handle sparse files efficiently
-n, --dry-run               perform a trial run with no changes made
-W, --whole-file            copy files whole (without delta-xfer algorithm)
-x, --one-file-system       don't cross filesystem boundaries
-B, --block-size=SIZE       force a fixed checksum block-size
-e, --rsh=COMMAND           specify the remote shell to use
    --rsync-path=PROGRAM    specify the rsync to run on the remote machine
    --existing              skip creating new files on receiver
    --ignore-existing       skip updating files that already exist on receiver
    --remove-source-files   sender removes synchronized files (non-dirs)
    --del                   an alias for --delete-during
    --delete                delete extraneous files from destination dirs
    --delete-before         receiver deletes before transfer, not during
    --delete-during         receiver deletes during the transfer
    --delete-delay          find deletions during, delete after
    --delete-after          receiver deletes after transfer, not during
    --delete-excluded       also delete excluded files from destination dirs
    --ignore-errors         delete even if there are I/O errors
    --force                 force deletion of directories even if not empty
    --max-delete=NUM        don't delete more than NUM files
    --max-size=SIZE         don't transfer any file larger than SIZE
    --min-size=SIZE         don't transfer any file smaller than SIZE
    --partial               keep partially transferred files
    --partial-dir=DIR       put a partially transferred file into DIR
    --delay-updates         put all updated files into place at transfer's end
-m, --prune-empty-dirs      prune empty directory chains from the file-list
    --numeric-ids           don't map uid/gid values by user/group name
    --timeout=SECONDS       set I/O timeout in seconds
    --contimeout=SECONDS    set daemon connection timeout in seconds
-I, --ignore-times          don't skip files that match in size and mod-time
    --size-only             skip files that match in size
    --modify-window=NUM     compare mod-times with reduced accuracy
-T, --temp-dir=DIR          create temporary files in directory DIR
-y, --fuzzy                 find similar file for basis if no dest file
    --compare-dest=DIR      also compare destination files relative to DIR
    --copy-dest=DIR         ... and include copies of unchanged files
    --link-dest=DIR         hardlink to files in DIR when unchanged
-z, --compress              compress file data during the transfer
    --compress-level=NUM    explicitly set compression level
    --skip-compress=LIST    skip compressing files with a suffix in LIST
-C, --cvs-exclude           auto-ignore files the same way CVS does
-f, --filter=RULE           add a file-filtering RULE
-F                          same as --filter='dir-merge /.rsync-filter'
```

```
                             repeated: --filter='- .rsync-filter'
     --exclude=PATTERN       exclude files matching PATTERN
     --exclude-from=FILE     read exclude patterns from FILE
     --include=PATTERN       don't exclude files matching PATTERN
     --include-from=FILE     read include patterns from FILE
     --files-from=FILE       read list of source-file names from FILE
 -0, --from0                 all *-from/filter files are delimited by 0s
 -s, --protect-args          no space-splitting; only wildcard special-chars
     --address=ADDRESS       bind address for outgoing socket to daemon
     --port=PORT             specify double-colon alternate port number
     --sockopts=OPTIONS      specify custom TCP options
     --blocking-io           use blocking I/O for the remote shell
     --stats                 give some file-transfer stats
 -8, --8-bit-output          leave high-bit chars unescaped in output
 -h, --human-readable        output numbers in a human-readable format
     --progress              show progress during transfer
 -P                          same as --partial --progress
 -i, --itemize-changes       output a change-summary for all updates
     --out-format=FORMAT     output updates using the specified FORMAT
     --log-file=FILE         log what we're doing to the specified FILE
     --log-file-format=FMT   log updates using the specified FMT
     --password-file=FILE    read daemon-access password from FILE
     --list-only             list the files instead of copying them
     --bwlimit=KBPS          limit I/O bandwidth; KBytes per second
     --write-batch=FILE      write a batched update to FILE
     --only-write-batch=FILE like --write-batch but w/o updating destination
     --read-batch=FILE       read a batched update from FILE
     --protocol=NUM          force an older protocol version to be used
     --iconv=CONVERT_SPEC    request charset conversion of filenames
     --checksum-seed=NUM     set block/file checksum seed (advanced)
 -4, --ipv4                  prefer IPv4
 -6, --ipv6                  prefer IPv6
     --version               print version number
(-h) --help                  show this help (-h is --help only if used alone)

Use "rsync --daemon --help" to see the daemon-mode command-line options.
Please see the rsync(1) and rsyncd.conf(5) man pages for full documentation.
See http://rsync.samba.org/ for updates, bug reports, and answers
```