calomel.org

Rsync Examples, SSH, Backup @ Calomel.org

https://calomel.org/rsync_tips.html

backups over ssh and secure local backup tips and tricks

Rsync is arguably one of the most powerful tools you can have in your arsenal as a systems administrator or user. It does not matter if you take care of one system or thousands, rsync can make your life easier. You can backup files, transfer data from one system to another or make sure the data in two locations are identical. Rsync can do this for you without you worrying about what data has changed; it will take care of the details.

History of rsync...

rsync is a software application for Unix systems which synchronizes files and directories from one location to another while minimizing data transfer using delta encoding when appropriate. An important feature of rsync not found in most similar programs/protocols is that the mirroring takes place with only one transmission in each direction. rsync can copy or display directory contents and copy files, optionally using compression and recursion.

The rsync utility uses an algorithm (invented by Australian computer programmer Andrew Tridgell) for efficiently transmitting a structure (such as a file) across a communications link when the receiving computer already has a different version of the same structure.

The recipient splits its copy of the file into fixed-size non-overlapping chunks, of size S, and computes two checksums for each chunk: the MD4 hash, and a weaker 'rolling checksum'. It sends these checksums to the sender.

The sender then compares its rolling checksums with the set sent by the recipient to determine if any matches exist. If they do, it verifies the match by computing the MD4 checksum for the matching block and by comparing it with the MD4 checksum sent by the recipient.

The sender then sends the recipient those parts of its file that didn't match any of the recipient's blocks, along with assembly instructions on how to merge these blocks into the recipient's version to create a file identical to the sender's copy.

If the sender's and recipient's versions of the file have many sections in common, the utility needs to transfer relatively little data to synchronize the files. Wikipedia, rsync

How does rsync help me?

What this means in layman's terms is rsync will only copy the data that is different on the two systems. If there is no data on the target machine then rsync will simple copy the data. But, if there was a copy of the data and only 1% of the file has changed there is no need to copy the entire file again. For example, if you had a 1 gig ISO file and only 1% of the file has changed then rsync will only copy the 1% that has changed thus saving time and bandwidth. This is incredibly efficient in time savings and essential if you pay for your Internet data rate.

Lets take a look at some examples...

Using rsync instead of copy (cp)

The simplest use of rsync is to copy data from on place to another on the same machine. You can copy entire disks, directories or just files with rsync. Lets says we have a external usb disk mount at /usbdisk/ that we are going to copy data from. We want to make sure the copy can resume even if someone unplugs the usb disk or we loose power to the unit. The following command will copy the files on the usb disk from the directory /usbdisk/ to the local disk in /localdisk/.

```
rsync -av /usbdisk/ /localdisk/
```

Now, what if you lost connectivity to the usb device and the sync stopped for some reason? You would then run the same rsync command a second time. Rsync would look at the files in both locations and copy the difference. In effect it would continue the copy from the point the sync was interrupted.

Using rsync over ssh instead of scp

Lets say we wanted to copy all of the files under our local directory, "/usbdisk/" and place a copy on the machine "somemachine" under the remote directory "/backups/". The name of the user on the remote machine is "calomel". Notice we added the "-z" argument so all data will be compressed over the network. We could use the following line:

```
rsync -avz /usbdisk/ calomel@somemachine:/backups/
```

What if you wanted to delete any files on the _remote_ machine that are no longer on the source machine? We can use the same command above and simply add the "--del" argument. This will tell rsync to delete any files from "/backups/" that are no longer on the source directory, "/usbdisk/".

```
rsync -avz --del /usbdisk/ calomel@somemachine:/backups/
```

Rsync recursive copy with parent directory name

What if you wanted to not just copy all of the files under /usbdisk/, but also the /usbdisk/ directory name. Just leave off the last "/" after the "/usbdisk" directory name. This will tell rsync to copy the directory name and all files under that directory. On the remote server you will then see the directory structure /backups/usbdisk/.

```
rsync -avz /usbdisk calomel@somemachine:/backups/
```

You can also rsync multiple files or directories using a single rsync line. This is useful when coping over the network as you only need to enter your ssh password once. Here we will copy the source directories /data1 and /data2 to the remote machine "somemachine" in the /backups/ directory.

```
rsync -avz /data1 /data2 calomel@somemachine:/backups/
```

Rsync pull instead of push

You can also pull the data from a remote machine to your local box. To rsync all of the files from the same "somemachine" to our local box just reverse the target and destination of the previous example:

```
rsync -avz calomel@somemachine:/backups/ /usbdisk/
```

Pull multiple files with rsync

Here we are pulling multiple files from a remote server to our local machine. We have to use curly braces to tell rsync we want only the files "file1", "file2" and "file3." Using this method you only have to enter your ssh password once to collect all the files.

```
rsync -avz calomel@somemachine:/remote_machine/{file1,file2,file3} /local/disk/
```

Rsync when remote files or directories contain spaces

Spaces cause all sorts of problems. To rsync when you have spaces in the files and/or directories you need to use a single tick to encompass the entire string and then escape the spaces. The following line shows the remote directory name "/I Hate Spaces" and the file name "some File.avi", both contain spaces. We will be rsync'ing the file to our local directory "/current_dir/".

```
rsync -av calomel@somemachine:'/I\ Hate\ Spaces/some\ File.avi' /current_dir/
```

If you are interested, check out our Youtube download script (Wget Mp3) where we explain how to download videos and even convert audio to mp3.

Execute remote shell command to rsync files

It is important to note rsync can also execute commands on the remote machine to help you generate a list of files copy. The shell command is expanded by your remote shell before rsync is called.

The following line will run the find command on the remote machine in the video directory and rsync all "avi" files it finds to our machine in the /download directory.

```
rsync -avR calomel@somemachine:'`find /data/video -name "*.[avi]"`' /download/
```

Pull data from a remote machine to local server using ssh

The following command will pull the data from "remote_machine" in /stuff/data/ and place it on the local system in /BACKUP/remote_machine/. The arguments "-avx" will set archive mode (-a) equivalent to -rlptgoD, be verbose (-v) and will not cross file system boundaries (-x) like NFS or samba. The timeout command makes sure rsync will not hang if the remote system is unreachable after 30 seconds. We will be deleting any files on the target directory (/BACKUP/remote_machine/) that are _not_ found in the source directory (data/). If you do not want to allow rsync to delete any files then take out "--delete-excluded".

The directory structure of the target machine will look like /BACKUP/remote_machine/data/ and this is considered a non-relative path option. Notice /stuff/ is not in the path.

```
rsync -avx --timeout=30 --delete-excluded backupuser@remote_machine:/stuff/data/
/BACKUP/remote_machine/
```

If you wanted the target directory structure to be relative you can add the argument "-R". The directory structure would then look like /BACKUP/remote_machine/stuff/data/ as the sync path name starts / on the source machine. The command with "-R" added looks like:

```
rsync -Ravx --timeout=30 --delete-excluded
backupuser@remote_machine:/stuff/data/ /BACKUP/remote_machine/
```

Incremental backups using rsync

The following example will make an incremental backup of the directory /data/working/ and put a copy of any file that changes into a dated directory in /BACKUP/ . This can be used to keep a daily backup tree of any changed files and not have to overwrite the previous days files. Note that this method does need to copy the entire file if it changes as the new files are made in the directory named under current day.

```
rsync --backup --backup-dir=`date +%Y.%m.%d` -a /data/working/ /BACKUP/
```

If you have a file under /BACKUP/ called file1 and that file has changed on the source machine in /data/working/ then a new directory will be made. The incremental dir would be named `date +%Y.%m. %d` or the numerical values for YEAR.MONTH.DAY and put under /BACKUP/. The changed data "file1" would be put under that directory.

Using rsync through ssh with keys? Validate that only rsync can be run

Lets say you need to setup an automated rsync between two machines. Lets also say that you are using ssh with ssh keys to authenticate and you only want rsync used. This mean you do not want the ssh connection used to bring up an interactive shell terminal.

The following script can be put anywhere, but we will put it in /usr/local/bin and called it **validate_rsync.sh** and make sure it is executable by running "chmod 755 /usr/local/bin/validate_rsync.sh". Then put your ssh key in ~/.ssh/id_dsa.pub from user1@machine1 into the ~/.ssh/authorized_keys file on user2@machine2. Now, at the beginning of the ssh key in ~/.ssh/authorized_keys on user2@machine2 add the string, **command="/usr/local/bin/validate_rsync.sh"** . This will mean that any machine using the ssh key from user1@machine1 will run the script validate_rsync.sh and will only be allow to execute rsync.

```
## Calomel.org -- validate_rsync.sh
#
#!/bin/sh
case "$SSH_ORIGINAL_COMMAND" in
*\&*)
echo "Rejected"
;;
*\(*)
echo "Rejected"
;;
*\{*)
echo "Rejected"
;;
*\;*)
echo "Rejected"
;;
*\>*)
echo "Rejected"
;;
*\`*)
echo "Rejected"
;;
*\|*)
echo "Rejected"
;;
rsync\ --server*)
$SSH_ORIGINAL_COMMAND
;;
*)
echo "Rejected"
;;
esac
```

Questions?

Do you have more examples using rsync?Yes we do. Check on the Calomel.org Home Page for instructions on automated server and user based backups. We go through the options of declaring backup schedules, ignoring specified file types and compressing long term archives.

How can I change the encryption method of ssh which rsync is using?Using the "--rsh" directive you can specify any encryption algorithm you want to. Lets use the "arcfour" encryption as it is the least CPU intensive and works well for very slow systems like 386's and Sun's SPARC 20's.

```
rsync -avx --timeout=30 --delete-excluded --rsh="ssh -c arcfour -l backupuser"
remote_machine:/stuff/data/ /BACKUP/remote_machine/
```

What should I be careful with when using Rsync?

1. Be careful with the --delete command. If your using a source dir that is not mounted (nfs,cifs,etc) but the mount for the dir is still there then you will sync your blank dir. All remote files will be deleted.
2. Be careful with slashes after the dir names on the source. A slash after the dir name compared to no slash after the dir name will do 2 totally different things.
3. Running out of memory with the older version of Rsync when doing a huge amount of files. Use the newer version due to its ability to do incremental file lists.
