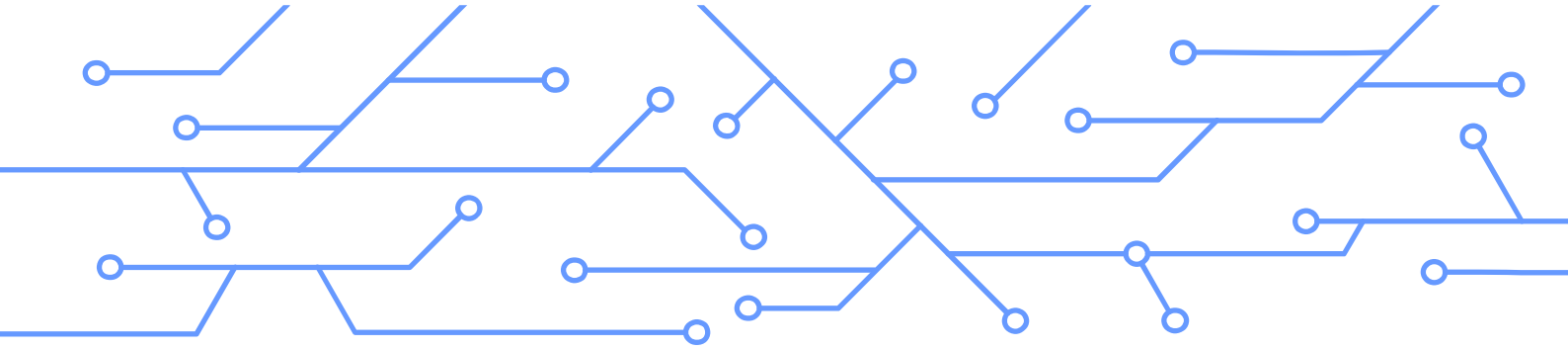

Exploring the rsync Utility

As we learned in the article, Using rsync to Sync Local and Remote Systems, rsync is a powerful utility for transferring files and creating backup operations. You can set up recurring file transfers for specific sets of files or entire directories and synchronize local and remote servers.

In this tutorial, we will explore:

The -a Flag

The -a (-archive) flag is an alias for a collection of other flags, -rlpgoD. If we break it down, each flag does the following:

- **`r - Recursive`**
  This tells rsync to copy directories recursively, meaning it will copy the directory, its contents and the contents of any subdirectories.
- **`l - Transfer any symlinks encountered`**
  If there are any symbolic links in the source files, rsync will recreate the link on the destination server rather than copying the file or folder that the link points to.
- **`t - Preserve time stamps`**
  rsync will transfer modification times along with the files and update them on the destination server.

Warning:

If -t is missing, the next transfer will behave as if it were used and all files will be updated, not just those with a modification time stamp.

- **`p - Preserve permissions`**
  rsync will update the destination file permissions to be the same as the source file permissions.
- **`g - Preserve groups`**
  This flag causes rsync to keep the group of the destination file the same as the source file..
- **`o - Preserve ownership`**

This keeps the owner of the destination file the same as the owner of the source file.

- **`D - Preserve block and character devices`**
  This will allow the destination system to recreate the character and block device information from the source system.

More Common rsync Flags

- **`v - Verbose`**
  By default, rsync works silently, by including the -v flag you will get information about what files are being transferred and a summary at the end.
- **`P - Combination of -partial -progress`**
  –partial will tell rsync to keep a partially transferred file if the transfer is interrupted. This will make a subsequent transfer of the rest of the file much faster.
- **–progress** tells rsync to print information showing the progress of a transfer.
- **`z - Compress`**
  rsync will compress any data from files sent to the destination machine. This flag is useful for slow links.
- **`H - Hard links`**
  This will tell rsync to recreate hard links on the remote system to match the links on the local system. Without using -H, hard links will be treated as regular files.
- **`e - -rsh=COMMAND`**
  This allows you to choose an alternative remote shell program to use for communication. rsync is typically configured to use ssh, but you may prefer to use rsh on a local network.
- **`n - Dry run`**
  A useful tool to tell rsync not to do any transfers, but report actions that would have been taken. This is a great way to check your commands prior to making a live transfer.

Flags in Action

Now that we've seen some of the more common flags used with rsync, let's take a look at a few examples showing how they work.

Note:

When moving files from one location to another, it is important to use a trailing slash (/) at the end of your first argument. If this is not used, your files will copy over as a subdirectory of the destination directory. You can see an example of this in our article Using rsync to Sync Local and Remote Systems.

Make a Backup of a Folder Without cp

First, create a backup directory:

```
mkdir /home/user1/backups
```

Then run rsync to move the data from the folder of your choice into the backup directory you just created:

```
rsync -avHP /home/user1/public_html/media /home/user1/backups/
```

Compress Files

You can reduce the network transfer adding the -z option if you are transferring files which have not yet been compressed, like text files.

The command would look like this:

```
rsync -az {source destination}
```

Using the P Flag

The -P flag can be very helpful, it is a combination of the flags –progress and –partial. –progress gives you a progress bar for the transfers and –partial allows you to resume interrupted transfers.

Your command:

```
rsync -azP {source destination}
```

Output:

sending incremental file list

```
./

file1

          0 100%    0.00kB/s    0:00:00 (xfer#1, to-check=99/101)

file10

          0 100%    0.00kB/s    0:00:00 (xfer#2, to-check=98/101)

file100

          0 100%    0.00kB/s    0:00:00 (xfer#3, to-check=97/101)

file11

          0 100%    0.00kB/s    0:00:00 (xfer#4, to-check=96/101)

. . .
```

If we update the modification time on some of the files, you can see that rsync intelligently re-copies only the changed files

```
touch liquid/file{1..10}

rsync -azP {source destination}

sending incremental file list

file1

          0 100%    0.00kB/s    0:00:00 (xfer#1, to-check=99/101)

file10

          0 100%    0.00kB/s    0:00:00 (xfer#2, to-check=98/101)

file2

          0 100%    0.00kB/s    0:00:00 (xfer#3, to-check=87/101)

file3

          0 100%    0.00kB/s    0:00:00 (xfer#4, to-check=76/101)

. . .
```

Warning:

If files are removed from the source, it is necessary to delete them from the destination directory in order to keep both directories in sync. By default, rsync does not delete anything from the destination directory.

You can change this behavior using –delete. To prevent data loss, use –dry-run to test before using –delete.

```
rsync -a --delete {source} {destination}
```

Exclude Certain Files

You can also exclude certain files or directories located inside a directory you are syncing by using –exclude=. The command will look like this:

```
rsync -a --exclude={pattern_to_exclude source destination}
```

Override that exclusion for files that match a different pattern by using the –include= option.

Type the command:

```
rsync -a --exclude={pattern_to_exclude} --include={pattern_to_included
source destination}
```