How To Use Rsync to Sync Local and Remote Directories

Linux BasicsBackups

-
- By Justin Ellingwood

  Last Validated onNovember 18, 2020 Originally Published onSeptember 10, 2013 3.1m views

Introduction

Rsync, which stands for “remote sync”, is a remote and local file synchronization tool. It uses an algorithm that minimizes the amount of data copied by only moving the portions of files that have changed.

In this guide, we will cover the basic usage of this powerful utility.

What Is Rsync?

Rsync is a very flexible network-enabled syncing tool. Due to its ubiquity on Linux and Unix-like systems and its popularity as a tool for system scripts, it is included on most Linux distributions by default.

Basic Syntax

The basic syntax of `rsync` is very straightforward, and operates in a way that is similar to ssh, scp, and cp.

We will create two test directories and some test files with the following commands:

```
cd ~
mkdir dir1
mkdir dir2
touch dir1/file{1..100}
```

We now have a directory called `dir1` with 100 empty files in it.

```
ls dir1
```

```
Output
file1    file18  file27  file36  file45  file54  file63  file72  file81  file90
file10   file19  file28  file37  file46  file55  file64  file73  file82  file91
file100  file2   file29  file38  file47  file56  file65  file74  file83  file92
file11   file20  file3   file39  file48  file57  file66  file75  file84  file93
file12   file21  file30  file4   file49  file58  file67  file76  file85  file94
file13   file22  file31  file40  file5   file59  file68  file77  file86  file95
file14   file23  file32  file41  file50  file6   file69  file78  file87  file96
file15   file24  file33  file42  file51  file60  file7   file79  file88  file97
file16   file25  file34  file43  file52  file61  file70  file8   file89  file98
```

```
file17   file26  file35  file44  file53  file62  file71  file80  file9   file99
```

We also have an empty directory called `dir2`.

To sync the contents of `dir1` to `dir2` on the same system, type:

```
rsync -r dir1/ dir2
```

The `-r` option means recursive, which is necessary for directory syncing.

We could also use the `-a` flag instead:

```
rsync -a dir1/ dir2
```

The `-a` option is a combination flag. It stands for “archive” and syncs recursively and preserves symbolic links, special and device files, modification times, group, owner, and permissions. It is more commonly used than `-r` and is usually what you want to use.

An Important Note

You may have noticed that there is a trailing slash (`/`) at the end of the first argument in the above commands:

```
rsync -a dir1/ dir2
```

This is necessary to mean “the contents of `dir1`”. The alternative, without the trailing slash, would place `dir1`, including the directory, within `dir2`. This would create a hierarchy that looks like:

```
~/dir2/dir1/[files]
```

Always double-check your arguments before executing an rsync command. Rsync provides a method for doing this by passing the `-n` or `--dry-run` options. The `-v` flag (for verbose) is also necessary to get the appropriate output:

```
rsync -anv dir1/ dir2
```

```
Output
sending incremental file list
./
file1
file10
file100
file11
file12
file13
file14
file15
file16
file17
```

```
file18
. . .
```

Compare this output to the output we get when we remove the trailing slash:

```
rsync -anv dir1 dir2
```

```
Output
sending incremental file list
dir1/
dir1/file1
dir1/file10
dir1/file100
dir1/file11
dir1/file12
dir1/file13
dir1/file14
dir1/file15
dir1/file16
dir1/file17
dir1/file18
. . .
```

You can see here that the directory itself is transferred.

How To Use Rsync to Sync with a Remote System

Syncing to a remote system is trivial if you have SSH access to the remote machine and `rsync` installed on both sides. Once you have SSH access verified between the two machines, you can sync the `dir1` folder from earlier to a remote computer by using this syntax (note that we *want* to transfer the actual directory in this case, so we omit the trailing slash):

```
rsync -a ~/dir1 username@remote_host:destination_directory
```

This is called a “push” operation because it pushes a directory from the local system to a remote system. The opposite operation is “pull”. It is used to sync a remote directory to the local system. If the `dir1` were on the remote system instead of our local system, the syntax would be:

```
rsync -a username@remote_host:/home/username/dir1 place_to_sync_on_local_machine
```

Like `cp` and similar tools, the source is always the first argument, and the destination is always the second.

Useful Options for Rsync

Rsync provides many options for altering the default behavior of the utility. We have already discussed some of the more necessary flags.

If you are transferring files that have not already been compressed, like text files, you can reduce the network transfer by adding compression with the `-z` option:

```
rsync -az source destination
```

The `-P` flag is very helpful. It combines the flags `--progress` and `--partial`. The first of these gives you a progress bar for the transfers and the second allows you to resume interrupted transfers:

```
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 run the command again, we will get a shorter output, because no changes have been made. This illustrates rsync's ability to use modification times to determine if changes have been made.

```
rsync -azP source destination
```

```
Output
sending incremental file list
sent 818 bytes received 12 bytes 1660.00 bytes/sec
total size is 0 speedup is 0.00
```

We can update the modification time on some of the files and see that rsync intelligently re-copies only the changed files:

```
touch dir1/file{1..10}
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)
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)
. . .
```

In order to keep two directories truly in sync, it is necessary to delete files from the destination directory if they are removed from the source. By default, rsync does not delete anything from the destination directory.

We can change this behavior with the `--delete` option. Before using this option, use the `--dry-run` option and do testing to prevent data loss:

```
rsync -a --delete source destination
```

If you wish to exclude certain files or directories located inside a directory you are syncing, you can do so by specifying them in a comma-separated list following the `--exclude=` option:

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

If we have specified a pattern to exclude, we can override that exclusion for files that match a different pattern by using the `--include=` option.

```
rsync -a --exclude=pattern_to_exclude --include=pattern_to_include source destination
```

Finally, rsync's `--backup` option can be used to store backups of important files. It is used in conjunction with the `--backup-dir` option, which specifies the directory where the backup files should be stored.

```
rsync -a --delete --backup --backup-dir=/path/to/backups /path/to/source destination
```

Conclusion

Rsync can simplify file transfers over networked connections and add robustness to local directory syncing. The flexibility of rsync makes it a good option for many different file-level operations.

A mastery of rsync allows you to design complex backup operations and obtain fine-grained control over what is transferred and how.

https://www.digitalocean.com/community/tutorials/how-to-use-rsync-to-sync-local-and-remote-directories