**How To Use Rsync to Sync Local and Remote Directories**

**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}

Copy

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

* ls dir1

Copy

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

Copy

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

We could also use the -a flag instead:

* rsync -a dir1/ dir2

Copy

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

Copy

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]

Copy

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

Copy

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

Copy

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

Copy

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

Copy

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

Copy

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

Copy

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

Copy

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

Copy

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

Copy

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

Copy

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

Copy

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

Copy

**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.

rsync command in Linux with Examples

**rsync**or remote synchronization is a software utility for Unix-Like systems that efficiently sync files and directories between two hosts or machines. One of them being the source or the local-host from which the files will be synced, the other one being the remote-host, on which synchronization will take place. There are basically two ways in which *rsync*can copy/sync data:

* Copying/syncing to/from another host over any remote shell like *ssh*, *rsh*.
* Copying/Syncing through rsync daemon using TCP.

Rsync is famous for its **delta-transfer algorithm**, in which it copies only the differences between the source files present in the local-host and the existing files in the destination or the remote host.

**Example:**

rsync local-file user@remote-host:remote-file

**What Happens here:** Rsync will first use SSH to connect as user to remote-host and will ask for user's password. Once connected, it will invoke the remote host’s rsync and then the two programs will determine what parts of the local-file needs to be copied so that the remote file matches the local one. Please note the following behavior of rsync:

* Files that do not exist on the remote-host are copied.
* Files that have been updated will be synced, rsync will copy only the changed parts of files to the remote host.
* File that is exactly the same are not copied to the remote host at all.

**Syntax of rsync:**

rsync [options] source [destination]

**Options:**

* **-a, –archive:** This is equivalent to using **-rlptgoD**. Archive mode includes all the necessary options like copying files recursively, preserving almost everything (like symbolic links, file permissions, user & group ownership and timestamps).
* -a, --archive

**Note:** The *archive* mode does not preserve hard links, because finding multiply-linked files is expensive. A *-H*options must be explicitly specified for hard links.

* **-v, –verbose:** By default, *rsync*works silently. A single *-v* will give us information about what files are being transferred and a brief summary about the data transferred at the end. Two *-v* options will give us information on the status of **delta-transmission** and on what files are up to date so as to be skipped and slightly more information at the end. More than two -v options are generally used for debugging rsync.
* -v, --verbose
* **-h, –human-readable format:** Outputs in a human readable format.
* -h, --human-readable format
* **-z, –compress:** Compress file data during the transfer
* -z, --compress

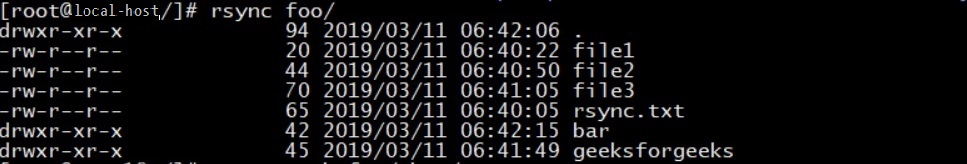
**Examples:**

 **Using rsync as a list command:** If only the source path is specified, the contents of the source are listed in an output format similar to ls -l.

rsync foo/

The above command will list the files and directories present in the directory *foo*.

**Output:**

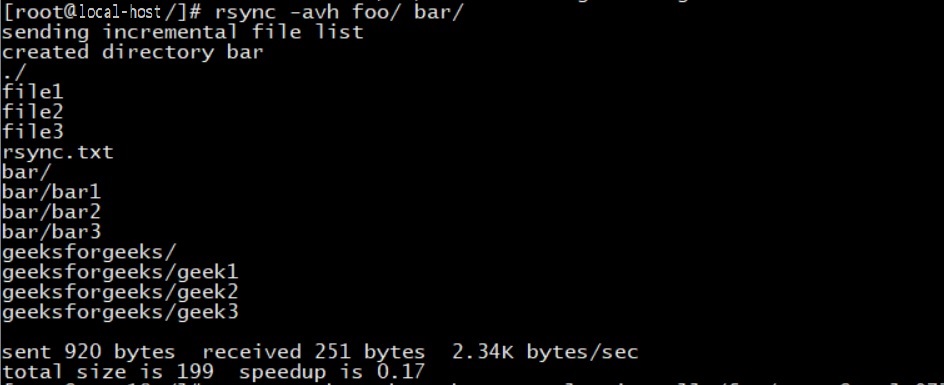


 **Copy/Sync files and directory locally:** If neither the source or destination path specifies a remote host, the rsync commands behave as a copy command.

rsync -avh foo/ bar/

The above command will copy/sync all the files and directories present in directory foo to directory *bar*. If the destination directory is not present (here bar), rsync automatically creates one and copies all the data in it.

**Output:**



 **Rsync using ssh:** There are two different ways for rsync to contact a remote system:

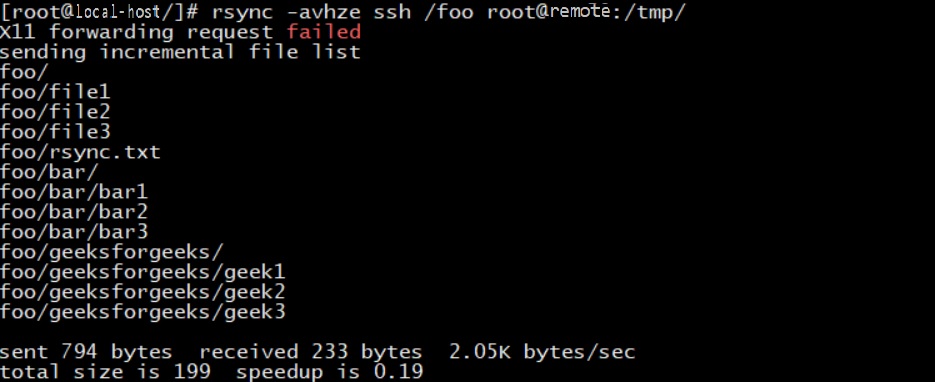
* Using a remote-shell program as the transport(such as ssh(Secure Shell) or rsh(Remote Shell)).
* Contacting an rsync daemon directly via TCP.

Here we will be discussing rsync over ssh.

rsync -avhze ssh /foo user@remote-host:/tmp/

To specify the type of protocol to be used, -e option is used.

**Output:**

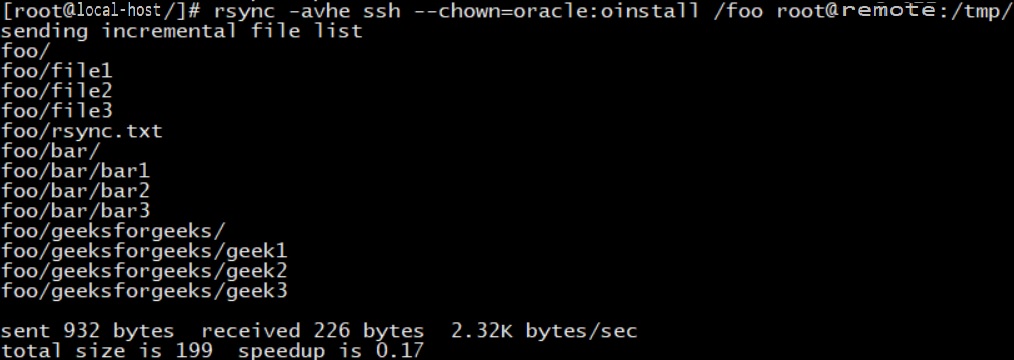


 **Rsync with particular file permissions:**If we want to sync files to the local or remote host with the permissions of the files being changed. The following command must be used.

rsync -avhe ssh --chown=USER:GROUP /foo user@remote-host:/tmp/

The above command will sync all the files present in directory ***/foo*** with the files present in directory ***/tmp*** in the remote-host with all the files owned by USER with group GROUP.

**Output:**



**Note:**The user and group must already be created in the remote-host.

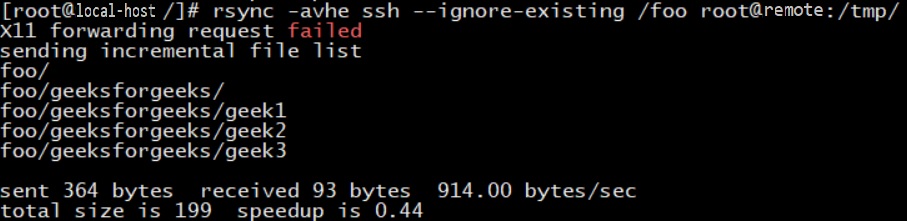
 **Rsync with --ignore-existing-files:** We can also skip the already existing files on the destination. This can generally be used when we are performing backups using the **–link-dest** option, while continuing a backup run that got interrupted.

rsync --ignore-existing -avhe /foo user@remote-host:/tmp/

So any files that do not exist on the destination will be copied over. Here I have deleted the geeksforgeeks folder from the directory foo, so it should copy only the geeksforgeeks directory.

**Note:**This does not ignore existing directories, or nothing would get done. Even if there are some changes in a file in the local host, it still would not be synced if its present on the remote host.

**Output:**

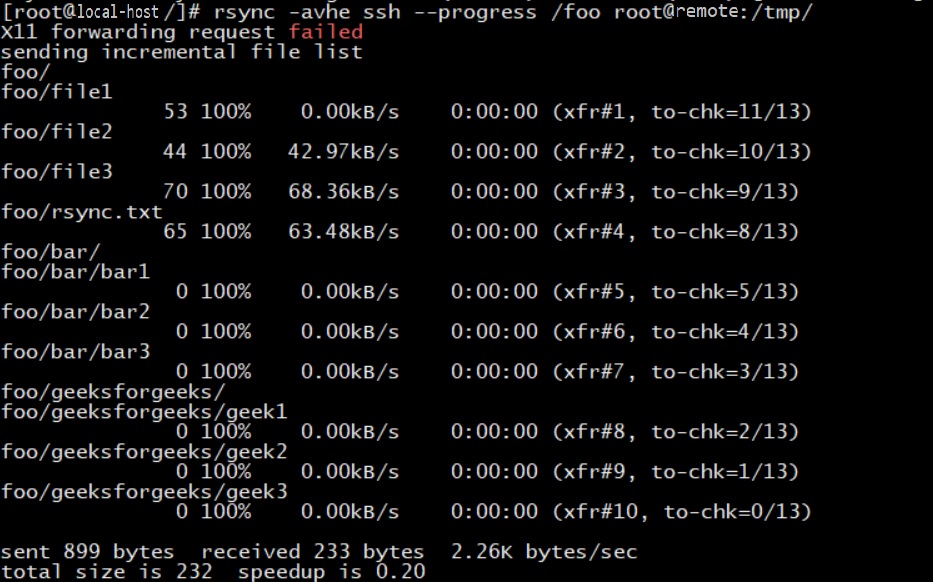


 **Show progress during transfer:** To show the progress while transferring the data from local-host to remote-host, we can use -–progress option.

rsync -avhe ssh --progress /foo user@remote-host:/tmp/

When the transfer completes for a particular file, rsync outputs a summary line as shown below.

**Output:**



In the above image, if we look at the file /foo/file2, it tells us that

* The file was 44 bytes long in total.
* The average rate of transfer for the whole file was 42.97 kilobytes per second over the 0:00:00 seconds that it took to complete.
* It was the second transfer during the current *rsync*session.
* There are 10 more files for the remote-host to check (to see if they are up-to-date or not) remaining out of the 13 total files in the file-list.

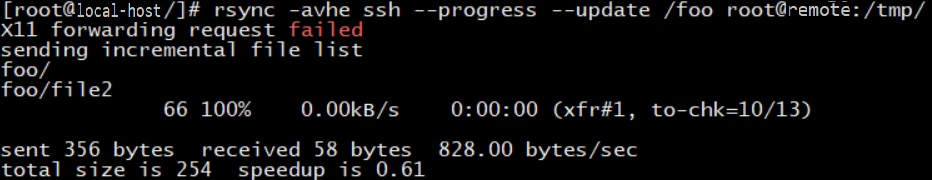
 **Update the remote only if there is a newer version is on the local filesystem:**If we want to copy files over the remote-host that have been updated more recently on the local filesystem. It is done with *–update* flag. The behavior is now like this:

* Files that do not exist on the remote-host are copied.
* Files that exist on both local and remote but have a newer timestamp on the local-host are copied to remote-host. (Conversely, files that have an older timestamp are not copied).

Here, I made some changes in *file1*and *file2*, but the changes in *file2*were done recently. So only, *file2*will get synced.

rsync -avhe ssh --progress --update /foo root@remote-host:/tmp/

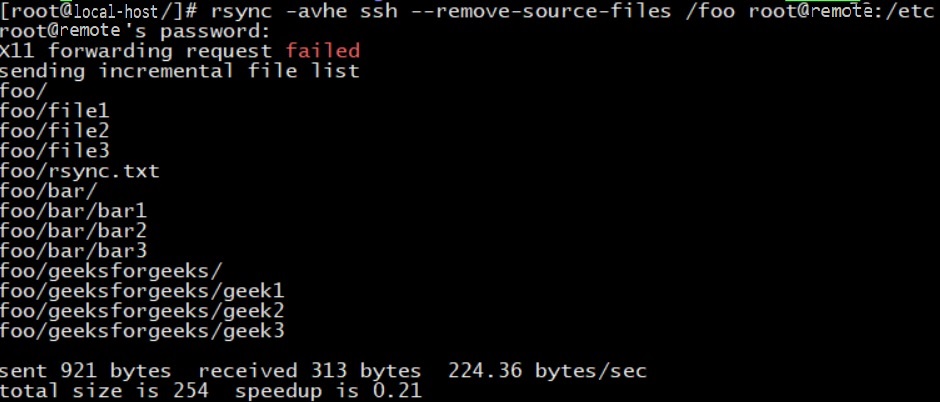
**Output:**



 **Automatically delete files from local-host after successful transferring:**Now, suppose we have a web server and a backup server and we created a daily backup and synced it with our backup server and then we don’t want to keep the local copy of the backup in our web server. So, instead of deleting it manually after successful transfer, we can use the --remove-source-files flag to automatically delete the files from the web server.

rsync -avhe ssh --remove-source-files /foo user@backup-server:/tmp

**Output:**

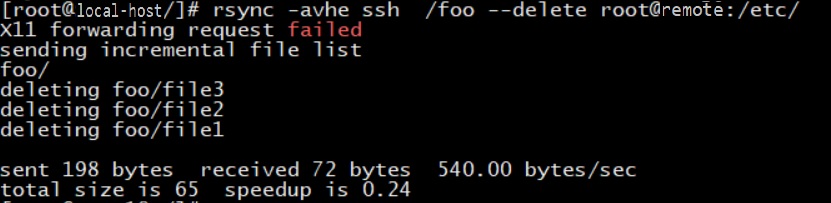


**Note:**This will delete only the files and not the directories.

 **Delete the files that have been deleted on the local-host:**If there are some files that are deleted on the local-host and we want that to be updated on the remote host as well, then we need to use the --delete option.

rsync -avhe ssh /foo --delete user@remote-host:/tmp/

**Output:**



So, here *file1*, *file2*, *file3*were deleted on the local-host, and as can be seen, are updated in the remote-host as well.

**Note:** rsync does not delete the files automatically to sync the directories present on both sides.

 **Performing a Dry run with rsync:**A Dry run makes rsync perform a trial run that doesn’t make any changes and displays almost the same output as a real run would do. It is generally used with the -v, –verbose and/or -i, –itemize-changes options so as to see what an rsync command would do before one actually runs it.

rsync -avhe ssh --dry-run --chown=USER:GROUP /foo user@remote-host:/

**Output:**

