How to Use the Linux rsync Command

Link: https://www.hostinger.com/tutorials/how-to-use-rsync

Copying files from one device to another can be a cumbersome task. Fortunately, you can simplify this process on Linux using the **rsync** command.

rsync, short for remote sync, lets you transfer and synchronize files or folders between local devices and remote Linux-based servers. Whether you're a pro or just getting started, mastering the **rsync** command can streamline your Linux file management.

This article will delve deep into the **rsync** command and how it works. We'll also demonstrate how to use the **rsync** command through practical examples.

What Is rsync?

rsync is a powerful and versatile [Linux command](#) for transferring and synchronizing files between local and remote devices. Unlike traditional copy commands, **rsync** uses a delta-transfer algorithm to only transmit the differences between the source and destination files. This approach drastically reduces bandwidth usage and speeds up transfers.

rsync also has robust features for transferring files to a backup server and mirroring tasks. It preserves file attributes and supports secure transfers over SSH, making it suitable for both local and remote file transfers.

How Does rsync Work?

This section will explore various **rsync** options and basic syntax for different purposes.

rsync Options and Parameters

rsync has numerous command line options, parameters, and configuration files to tailor its behavior. Here are some commonly used ones:

- **-v or –verbose** – Increase verbosity, providing more detailed output during the transfer.
- **-a or –archive** – Archive mode, which includes recursive copying and preserving file permissions, timestamps, [symbolic links](#), and device files.
- **-r or –recursive** – Recursively copy directories.
- **–delete** – Delete excluded files from the destination directory.
- **–exclude=PATTERN** – Exclude files or directories matching the specified pattern.
- **–include=PATTERN** – Include files or directories matching the specified pattern.
- **-z or –compress** – Compress file data during the transfer to reduce bandwidth usage.
- **-s or –sparse** – Generate a summary of synchronized files and directories, including sparse files, after a sync operation.
- **–dry-run** – Perform a trial run without making any actual changes.
- **–temp-dir** – Specify a directory to store temporary files.
- **-u or –update** – Skip files on the destination side that are newer than the source files so only older files are updated.

- **-h or –human-readable** – Output numbers in a human-readable format.
- **-i or –itemize-changes** – Output a list of changes made during the transfer.
- **–progress** – Show progress during the transfer.
- **–stats** – Provides file transfer stats after it is complete.
- **-e or –rsh=COMMAND** – Specify which remote shell to use.
- **–bwlimit=RATE** – Limit the bandwidth to increase network efficiency.
- **-P or –partial –progress** – Keep partially transferred files and show progress.

For a comprehensive list of all available **rsync** options, run the following command:

```
man rsync
```

You will see detailed information about each option and parameter.

Basic Syntax

The basic syntax of an **rsync** command is as follows:

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

- **[OPTIONS]** – This is the section where you can include **rsync** options. You can add more than one option.
- **SOURCE** – This is the source directory or file you want to copy or synchronize. Specify the path to the source data here.
- **DESTINATION** – The destination directory where the source data will be copied or synchronized. Specify the path to the destination directory or file here.

Basic Syntax for Remote Shell

When using **rsync** to transfer data from a local computer to a Linux virtual private server (VPS), communication relies on the **rsync** daemon. The **rsync** syntax for the remote shell is as follows:

```
rsync [OPTIONS] -e "SSH_COMMAND" SOURCE DESTINATION
```

The **-e** option is used to specify the remote shell. In most cases, you'll use **ssh** to connect to the remote host using the **rsync** remote update protocol.

Let's explore two common scenarios.

Use the following command to pull data from a remote system to your local machine:

```
rsync -avz -e ssh user@remote_host:/path/to/source/ /path/to/local/destination/
```

Use the following command to push data from your local file system to a remote directory using the **CVS** protocol:

```
rsync -avz /path/to/local/source/ user@remote_host:/path/to/remote/destination/
```

How to Check the rsync Version

rsync is typically included by default in many <u>Linux distributions</u>. Let's check whether **rsync** is already installed on your system.

For Windows users working with <u>VPS Hosting</u>, <u>use PuTTY SSH</u> to log in. If you're using macOS or Linux, access Terminal.

Once logged in, execute the command below:

```
rsync --version
```

You'll receive an output similar to the following:

```
rsync version 3.2.7 protocol version 31
```

How to Install rsync

If **rsync** isn't pre-installed on your local or remote machine, go ahead and install it manually. Here are the installation commands for different operating systems:

For Debian-based distributions, including Ubuntu:

```
sudo apt-get install rsync
```

For Fedora-based distributions, such as CentOS:

```
sudo dnf install rsync
```

For macOS:

```
brew install rsync
```

How to Use rsync Commands

Before learning to use **rsync**, let's prepare two test directories named **original** and **duplicate**. The **original** directory will contain three sample files, while the **duplicate** directory will start out empty.

To create these directories, follow these commands:

```
cd
mkdir original
mkdir duplicate
```

Next, create three sample files inside the **original** folder:

```
touch original/file{1..3}
```

To make sure all the sample files are created, list all the files in the **original** directory and observe the file system using this command:

```
rsync original/
```

Most Common rsync Commands

One of the most essential use cases for **rsync** is to replicate data between two directories within the same system. To do this, use the following command:

```
rsync original/* duplicate/
```

The contents inside the **original** directory will be mirrored in the **duplicate** directory. If you add a new file or update existing files in the **original** directory, only the new or changed files will be transferred. However, if the **duplicate** folder doesn't exist, it will result in an error.

To synchronize files and create a new folder simultaneously, use this command instead:

```
rsync original/ duplicate/
```

How to Use rsync Commands With Subdirectories

To synchronize folders and subdirectories between two locations, use this **rsync** copy directory command:

```
rsync -r original/*/ duplicate/
```

To synchronize a specific subdirectory, type the command below:

```
rsync -r original/subdirectory_name/ duplicate/
```

Replace **subdirectory_name** with the name of the subfolder you want to synchronize.

You may want to exclude a particular subdirectory from synchronization. In this case, enter the following command to do it:

```
rsync -r --exclude=subdirectory_name original/ duplicate/
```

How to Synchronize Files

To sync or update files between two folders, use this command:

```
rsync -av original/ duplicate/
```

To copy the files from the **original** directory to a remote server, enter this command:

```
rsync -av -e ssh original/ username@remote_host:/path/to/destination/
```

Replace **username, remote_host,** and **/path/to/destination/** with the appropriate values.

How to Combine rsync Commands

As you become more familiar with **rsync**, let's explore its capability to combine multiple commands for complex file management tasks.

You can combine synchronization and exclusion features to achieve precise results.

The example below shows how you can synchronize all files from the **original rsync** directory while excluding **TXT** files:

```
rsync -av --exclude='*.txt' original/ duplicate/
```

Combine the **-r** option with synchronization commands to ensure that **rsync** directories and their contents are recursively synchronized.

```
rsync -av -r original/ duplicate/
```

Before synchronizing an actual **rsync** folder, you can use the **–dry-run** option to preview the changes **rsync** would make without making any actual modifications.

```
rsync -av --dry-run original/ duplicate/
```

Other Options for rsync Commands

The **–delete** option allows you to delete files from the destination directory that no longer exist in the source directory. To use this option, include it in your **rsync** command like this:

```
rsync -av --delete original/ duplicate/
```

rsync supports synchronizing specified files or file types using patterns and wildcards. For example, to only synchronize **TXT** files, enter:

```
rsync -av original/*.txt duplicate/
```

You can also exclude files based on specific patterns in their names. To exclude a file named **example.txt**, type the following command:

```
rsync -av --exclude=example.txt original/ duplicate/
```

Combine the **–include** and **–exclude** options to include multiple files or directories while excluding others. Here's an example to include files beginning with the letter **L** and exclude all the other files:

```
rsync -av --include='L*' --exclude='*' original/ duplicate/
```

To limit synchronization to files below a specific size, use the **–max-size** option followed by the size limit. The **rsync** command to only synchronize files smaller than **10 MB** is as follows:

```
rsync -av --max-size=10M original/ duplicate/
```

How to Add a Progress Bar

Monitoring synchronization progress can be helpful, especially for large file transfers. **rsync** allows you to include a progress bar using the **–progress** option. Here's the command you can employ:

```
rsync -av --progress original/ duplicate/
```

The output will look something like this:

```
file1.txt
    5,120,000 100%   50.00MB/s 0:00:00 (xfr#1, to-chk=2/3)
file2.txt
    5,345,678 100%   55.67MB/s 0:00:00 (xfr#2, to-chk=1/3)
```

To add a progress bar and keep partially transferred files instead of deleting them upon interruption, use the **-P** option:

```
rsync -av -P original/ duplicate/
```

How to Create an rsync Backup

Lastly, **rsync** provides a convenient way to create backup files using the **–backup** option. This option lets you back up files to a server, preventing overwriting during synchronization.

To create a remote backup and specify its directory, use the following command:

```
rsync -av --backup --backup-dir=/path/to/backup/ original/ duplicate/
```

When executed, the **rsync** backup option generates an incremental file list and appends a tilde (~) to the original file name, such as **important.txt.**

Conclusion

rsync is a powerful remote synchronization, data transfer, and file mirroring tool. In this guide, we've covered everything you need to get started with the tool, from installation to practical **rsync** examples you can apply via the command line. Mastering **rsync** will enhance your Linux file management, making it more efficient and reliable.

Discover Other Linux Commands for Server Management

How to Check Disk Space on Linux

How to Transfer Data With Curl Command

How to Calculate Process Execution With Time Command

How to Transfer Files Using Scp Command

How to Monitor Changes With Watch Command

How to Shutdown and Restart the Server

How to List Services in Linux

How to Write and Display to File With Tee Command

rsync FAQ

This section will answer the most common questions about **rsync.**

What Operating Systems Are Compatible With rsync?

rsync is primarily designed for Unix-like operating systems, including Linux and macOS. However, it can also be used on Windows systems with the help of third-party **rsync** client applications like Cygwin or Windows Subsystem for Linux (WSL). This makes **rsync** a versatile choice for file synchronization across various operating systems.

How Does rsync Differ From Other File Transfer Methods?

Instead of transferring entire file systems, **rsync** only sends the differences between destination and source files, reducing bandwidth usage. It can work over secure SSH connections, offer flexible file compression, and resume interrupted transfers. It's particularly handy when dealing with a large number of files in a remote system.

Are There Any Limitations or Drawbacks to Using rsync?

While **rsync** is a powerful tool, it has some limitations. First, it may not be suitable for real-time synchronization as it operates in batch mode. Additionally, it doesn't provide native encryption, as users often rely on SSH for secure transfers. Lastly, **rsync** can be complex for beginners, requiring a learning curve to master its extensive options.

Revision #1
Created 5 July 2024 21:23:33 by Administrador
Updated 5 July 2024 21:28:48 by Administrador