Linux - How to use rClone

> Last verified: October 2026 against the rclone documentation (current release v1.75.1); the filter and symlink behaviour was tested with rclone v1.75.1.

Here's how to get rclone doing what rsync was doing, but with real parallelism.

1. Install it

```
sudo -v ; curl https://rclone.org/install.sh | sudo bash
```

or via a package manager (`dnf install rclone`, `apt install rclone`, `brew install rclone`, etc.). The rclone docs note that distribution packages are often out of date, so the install script or the official binaries are the better way to get the current version.

2. Configure a remote

Rclone needs a "remote" config pointing at your server. For an SSH target the **SFTP backend** is the simple option: it works over your existing SSH setup, no extra service needed.

```
rclone config
```

Walk through the prompts:

- `n` for new remote
- name it, e.g. `myserver`
- type: `sftp`
- host: `server.example.com`
- user: `root`
- port: 22 (default)
- leave the rest default, use SSH agent or key auth if you have it set up

Or skip the wizard and write it directly to `~/.config/rclone/rclone.conf`:

```
[myserver]
type = sftp
host = server.example.com
```

```
user = root
```

Or use the inline command to create the config file

Run this single command (replace YOUR.SERVER.IP with the real IP or hostname):

```
rclone config create myserver sftp host=YOUR.SERVER.IP user=root pass=$(rclone obscure 'YOUR_PASSWORD')
```

After it finishes, test the connection (`lsd` lists only the top-level directories; `ls` would list every file on the server recursively):

```
rclone lsd myserver:/
```

3. Run the sync

```
rclone sync /path/to/source/ myserver:/path/to/dest/ \
  --progress \
  --transfers=32 \
  --checkers=32
```

Key flags, and why:

- **`sync`** — makes the destination match the source, including **deleting** extra files on the destination (like `rsync -a --delete`). Use `copy` instead if you don't want deletions, and try `--dry-run` first.
- **`--transfers=32`** — number of files transferred in parallel. This is the big one for many-small-files workloads; rsync can't do this natively. Tune based on file count/CPU — 16–64 is a common range.
- **`--checkers=32`** — parallel workers for comparing file existence/size/hash before transfer.
- **`--progress`** — live stats, like rsync's `-P`.

4. Useful additions

```
rclone sync /path/to/source/ myserver:/path/to/dest/ \
  --progress \
  --transfers=32 \
  --checkers=32 \
  --stats=5s \
  --stats-one-line \
  --exclude ".git/**" \
```

```
  --dry-run
```

- `--dry-run` — test first, see what would change without touching anything
- `--stats=5s --stats-one-line` — periodic compact progress instead of a wall of text
- `--exclude` — same idea as rsync's exclude patterns
- `--checksum` — only if you need content-based comparison instead of size+mtime (same cost tradeoff as rsync's `-c`)

Pulling a Synology share to a local disk, with settings for a LAN and for a WAN link. To skip Synology's system folders (such as `@eaDir` and `#recycle`), exclude the folder *and its contents* with `/**`; a bare `--exclude "#recycle"` only matches a file with that name, so the folder's contents would still be copied.

```
#LAN
rclone sync myserver:/mnt/volume1/data_syno /mnt/volume1/data/ \
  --exclude "@*" \
  --exclude "@*/**" \
  --exclude "#recycle/**" \
  --multi-thread-streams=16 \
  --buffer-size=128M \
  --transfers=16 \
  --progress \
  --checkers=32 \
  --sftp-concurrency=128 \
  --log-level=INFO \
  --stats=10s
```

```
#WAN
rclone sync myserver:/mnt/volume1/data_syno /mnt/volume1/data/ \
  --exclude "@*" \
  --exclude "@*/**" \
  --exclude "#recycle/**" \
  --multi-thread-streams=4 \
  --buffer-size=64M \
  --transfers=8 \
  --progress \
  --checkers=16 \
  --sftp-concurrency=64 \
  --bwlimit=20M \
  --log-level=INFO \
  --stats=30s
```

Each continued line must end with a backslash `\`. `--fast-list` is left out because the SFTP backend does not support the ListR feature it relies on, so it has no effect there. `--sftp-concurrency=64` is already the default.

5. If SFTP itself is the bottleneck

SFTP-over-SSH still has per-file protocol overhead. For raw speed on a trusted LAN, rclone also supports:

- **`type = local`** on both ends if you mount the remote filesystem (NFS/etc.) — no protocol overhead at all
- Running an `rclone serve` daemon on the destination for a lighter native protocol

But for your use case (SSH between two Linux boxes), the SFTP backend with high `--transfers` is usually the easy win — try it and compare wall-clock time against your rsync run.

Multiple excludes

Repeat the `--exclude` flag once per pattern:

```
rclone sync /path/to/source/ myserver:/path/to/dest/ \
  --progress \
  --transfers=32 \
  --checkers=32 \
  --exclude "Revision/**" \
  --exclude "ftpDataSite/**" \
  --exclude "Backups/**"
```

The `/**` matches the folder and everything under it, anywhere in the tree. If these folders only live at the top level (not nested elsewhere), you can anchor them:

```
--exclude "/Revision/**" --exclude "/ftpDataSite/**" --exclude "/Backups/**"
```

(leading `/` means "relative to the sync root only")

If the list grows, it's cleaner to use a file instead of repeated flags:

```
# excludes.txt
Revision/**
ftpDataSite/**
Backups/**
```

```
rclone sync /path/to/source/ myserver:/path/to/dest/ \
  --progress --transfers=32 --checkers=32 \
  --exclude-from excludes.txt
```

You can test what's actually getting matched with:

```
rclone sync ... --exclude-from excludes.txt --dry-run -vv
```

Symlinks

This is the part worth being careful about — rclone's symlink behaviour differs from rsync's.

By default, rclone **does not copy symlinks**: it skips them and logs a notice for each one ("Can't follow symlink without -L/--copy-links"). Your options:

Follow symlinks and copy the actual file/directory contents (like rsync's `-L`):

```
--copy-links
```

Use this if you want the symlink target's real content transferred instead of a symlink pointer — e.g., if the destination server can't resolve the same symlink targets your source can.

Store symlinks as .rclonelink files (`-l` / `--links`):

```
--links
```

With an SFTP destination this does **not** create real symlinks. Each link is stored as a small text file named `<name>.rclonelink` that contains the link target. Copying those files back to a local disk with `--links` turns them into symlinks again. That makes it useful for backups you will restore with rclone, but not for a live copy of the tree. `--links` cannot be combined with `--copy-links`.

Skip symlinks without the notices — skipping is the default; `--skip-links` only suppresses the per-link messages, which is worth being explicit about in scripts:

```
--skip-links
```

There is no exact equivalent of rsync's `-a` (real symlinks recreated on the destination) for an SFTP destination. If the symlinks must stay real links, keep using rsync for that tree, or use `--copy-links` if copying the target contents is acceptable:

```
rclone sync /path/to/source/ myserver:/path/to/dest/ \
  --progress \
```

```
--transfers=32 \
--checkers=32 \
--exclude-from excludes.txt \
--copy-links
```

Worth doing a `--dry-run -vv` first to confirm rclone reports handling the links the way you expect, especially if any of your symlinks point outside the source folder or are broken/dangling.

Note Worthy

Fix: add the host key to known_hosts

```
ssh-keyscan server.example.com >> ~/.ssh/known_hosts
```

Or just SSH to it manually once and accept the fingerprint interactively (safer, because you can check the fingerprint):

```
ssh root@server.example.com
```

(type `yes` when prompted, then exit)

Important: the hostname in known_hosts has to match what rclone connects to.

Sources

- https://rclone.org/install/
- https://rclone.org/commands/rclone_sync/
- https://rclone.org/docs/
- https://rclone.org/local/
- https://rclone.org/sftp/
- https://rclone.org/filtering/
- https://rclone.org/overview/

Revision #2
Created 8 August 2026 03:04:00 by Admin
Updated 24 October 2026 14:10:46 by Admin