Linux - How to use rClone

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

1. Install it

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

or via package manager (`dnf install rclone`, `apt install rclone`, `brew install rclone`, etc.)

2. Configure a remote

Rclone needs a "remote" config pointing at your server. Two good options for an SSH target:

Option A: SFTP backend (works over your existing SSH setup, no extra service needed)

```
rclone config
```

Walk through the prompts:

- `n` for new remote
- name it, e.g. `sfl004`
- type: `sftp`
- host: `sfl-lin-004`
- 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`:

```
[sfl004]
type = sftp
host = sfl-lin-004
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 'PAssword1')
```

After it finishes, test the connection:

```
rclone ls myserver:/
```

3. Run the sync

```
rclone sync /opt/kiwi/rev/ sfl004:/opt/kiwi/rev/ \
  --progress \
  --transfers=32 \
  --checkers=32
```

Key flags, and why:

- **`sync`** — makes destination match source (like `rsync -a`, deletes extras on dest). Use `copy` instead if you don't want deletions.
- **`--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 /opt/kiwi/rev/ sfl004:/opt/kiwi/rev/ \
  --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`)

```
#LAN
rclone sync sfl004:/mnt/volume1/data_syno /mnt/volume1/data/ \
--exclude="@*" /
--exclude="#recycle" /
--multi-thread-streams=16 /
--buffer-size=128M /
--transfers=16 /
--progress /
--checkers=32 /
--sftp-concurrency=128 /
--fast-list /
--log-level=INFO /
--stats=10s
#WAN
rclone sync sfl004:/mnt/volume1/data_syno /mnt/volume1/data/ \
--exclude "@*" /
--exclude "#recycle" /
--multi-thread-streams=4 /
--buffer-size=64M /
--transfers=8 /
--progress /
--checkers=16 /
--sftp-concurrency=64 /
--fast-list /
--bwlimit=20M /
--log-level=INFO /
--stats=30s
```

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 /opt/kiwi/rev/ sfl004:/opt/kiwi/rev/ \
  --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 /opt/kiwi/rev/ sfl004:/opt/kiwi/rev/ \
  --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 default symlink behavior differs from rsync's.

By default, rclone **skips symlinks entirely** (doesn't follow them, doesn't recreate them, doesn't error) unless you tell it what to do. Your options:

Copy symlinks as symlinks (equivalent to rsync's `-l` / part of `-a`) — this is almost certainly what you want if rsync's `-a` was working fine for you before:

```
--links
```

This recreates the symlink itself on the destination (SFTP backend supports this). Broken symlinks are preserved as-is, not treated as errors.

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.

Skip symlinks silently — this is the default, no flag needed, but worth being explicit if you want it documented in scripts:

```
--skip-links
```

For your case, matching rsync's `-a` behavior, you want `--links`:

```
rclone sync /opt/kiwi/rev/ sfl004:/opt/kiwi/rev/ \
  --progress \
  --transfers=32 \
  --checkers=32 \
  --exclude-from excludes.txt \
  --links
```

One gotcha: with `--links` on the **SFTP backend specifically**, symlink support depends on the remote SSH server's SFTP subsystem supporting symlink creation (most do — OpenSSH's does). Worth doing a `--dry-run -vv` first to confirm rclone reports handling them the way you expect, especially if any of your symlinks point outside `/opt/kiwi/rev/` or are broken/dangling.

Note Worthy

Fix: add the host key to known_hosts

```
ssh-keyscan sfl-lin-004.onling.com >> ~/.ssh/known_hosts
```

Or just SSH to it manually once and accept the fingerprint interactively:

```
ssh root@sfl-lin-004.onling.com
```

(type `yes` when prompted, then exit)

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

Revision #4
Created 8 August 2026 03:04:00 by Steve Ling
Updated 8 August 2026 05:55:24 by Steve Ling