rclone Cloud file sync

START HERE

`rclone config`
Set up a remote

`rclone ls <remote>:<dir>`
List files

`rclone lsd <remote>:`
List top folders

`rclone check <src> <dst>`
Compare two sides

`rclone copy <src> <remote>:<dir> -P`
Upload, keep extras

`rclone mount <remote>: <dir>`
Browse as a folder

`rclone sync <src> <dst> --dry-run`
Preview a mirror

`rclone about <remote>:`
Show quota

Source local or remote → `copy / sync / move` → **rclone** filters, checks → `backend API` → **Destination** remote:path

CONCEPTS

Paths are remote:path. A path without a colon is local. Test any sync with --dry-run first.

sync makes the destination identical to the source and DELETES extra files there. Swapped arguments delete files you wanted to keep.

Config lives in rclone.conf (find it with rclone config file). It holds tokens and secrets: keep it private.

REMOTES AND CONFIG

`rclone config`
Interactive remote setup

`rclone listremotes`
List configured remotes

`rclone config show <remote>`
Show a remote's settings

`rclone config file`
Print config file path

`rclone config create <name> <type> [key=value]`
Create a remote non-interactively

`rclone config delete <name>`
Remove a remote

`rclone obscure <password>`
Obscure a password for the config

`rclone authorize <type>`
Get a token on a machine with a browser
Paste the result into the headless machine's config

`rclone --config <file> ...`
Use another config file

`rclone help backends`
List supported storage types

COPY, SYNC AND MOVE

`rclone copy <src> <dst> -P`
Copy new/changed files, never delete

`rclone sync <src> <dst> -P`
Mirror src to dst, deleting extras
Always run with --dry-run first; deletes happen on the destination

`rclone move <src> <dst>`
Copy then delete from source

`rclone copyto <src> <dst>`
Copy one file, renaming it

COPY, SYNC AND MOVE (cont.)

`rclone moveto <src> <dst>`
Move or rename one file

`rclone copyurl <url> <dst>`
Download a URL straight to a remote

`rclone rcat <remote>:<file>`
Upload from stdin

`rclone cat <remote>:<file>`
Print a remote file

`rclone bisync <path1> <path2> --resync`
First two-way sync run
bisync is newer; use --resync once, then run without it

`rclone bisync <path1> <path2>`
Two-way sync

`rclone sync <src> <dst> --create-empty-src-dirs`
Keep empty directories

LISTING AND INFO

`rclone ls <remote>:<dir>`
Size and path of every file

`rclone lsd <remote>:<dir>`
List directories only

`rclone lsl <remote>:<dir>`
Add modification times

`rclone lsf <remote>:<dir>`
Plain names, script friendly

`rclone lsf -R --files-only <remote>:`
Recursive file names

`rclone lsjson <remote>:<dir>`
List as JSON

`rclone tree <remote>:<dir>`
Tree view

`rclone size <remote>:<dir>`
Total size and file count

`rclone ncdu <remote>:`
Interactive disk usage

`rclone about <remote>:`
Quota and free space

`rclone link <remote>:<file>`
Create a public link

VERIFY AND CLEAN UP

`rclone check <src> <dst>`
Compare sizes and hashes

`rclone check <src> <dst> --one-way`
Only check source files exist

VERIFY AND CLEAN UP (cont.)

`rclone check <src> <dst> --size-only`
Compare sizes only

`rclone md5sum <remote>:<dir>`
Print MD5 sums

`rclone hashsum <hash> <remote>:<dir>`
Print any supported hash

`rclone delete <remote>:<dir>`
Delete files, keep directories

`rclone deletefile <remote>:<file>`
Delete one file

`rclone purge <remote>:<dir>`
Delete directory and everything in it
No filters apply; purge cannot be undone on most remotes

`rclone mkdir <remote>:<dir>`
Create a directory or bucket

FLAGS

`-n / --dry-run`
Show what would happen

`-P / --progress`
Live transfer stats

`-v / -vv`
More logging

`-i / --interactive`
Ask before each change

`--transfers <n>`
Parallel file transfers

`--bwlimit <rate>`
Limit bandwidth (e.g. 5M)

`-u / --update`
Skip files newer on destination

`--ignore-existing`
Never overwrite existing files

`-c / --checksum`
Compare by hash, not mod-time

`--backup-dir <dir>`
Move overwritten/deleted files aside
Must be on the same remote as the destination

`--max-delete <n>`
Abort if sync would delete more

`--log-file <file> --log-level INFO`
Log to a file

`--files-from <file>`
Transfer only listed paths

FILTERS

`--include "<glob>"`
Only matching files
Order matters: first matching rule wins; add --exclude "" last*

`--exclude "<glob>"`
Skip matching files

`--exclude-from <file>`
Read exclude patterns from file

`--filter "+ <glob>"`
Include rule in filter syntax

`--filter-from <file>`
Rules in a file (+ / - lines)

`--min-size <size>`
Skip smaller files

`--max-size <size>`
Skip bigger files

`--max-age <dur>`
Only files newer than e.g. 7d

`--min-age <dur>`
Only files older than e.g. 1h

MOUNT AND SERVE

`rclone mount <remote>: <dir>`
Mount as a filesystem

`rclone mount <remote>: <dir> --vfs-cache-mode writes`
Make writes reliable
Without a cache mode, many programs cannot write files

`rclone mount ... --daemon`
Run in the background

`fusermount -u <dir>`
Unmount on Linux

RECIPES

Safe mirror to the cloud

```
rclone sync ~/docs remote:docs --dry-run
rclone sync ~/docs remote:docs -P --max-delete 20 \
  --backup-dir remote:docs-trash/$(date +%F)
rclone check ~/docs remote:docs --one-way
```

Backup only some files

```
rclone copy ~/proj remote:proj -P \
  --exclude ".git/**" --exclude "node_modules/**" \
  --max-age 30d
```

Mount and unmount

```
mkdir -p ~/cloud
rclone mount remote: ~/cloud --vfs-cache-mode writes --daemon
# ... use the files ...
fusermount -u ~/cloud
```

Cron-friendly upload with log

```
rclone copy /srv/backup remote:backup \
  --log-file ~/rclone.log --log-level INFO --bwlimit 2M
```