RCLONE

rclone は強力なクラウドストレージ管理ツールです。以下に、最もよく使用されるrcloneコマンドとその説明をまとめました。

用例

- encrypt files/folders and its name, backup => Rclone
- ignore filter, on-demand, password, zip compress => Need scripting
- chunker -> crypt -> union
- combine -> hasher -> crypt -> drive

方法

0. 基本

次の構文が基本形となります。

```
rclone <command> <remote_bucket>:<remote> ...
```

ドキュメントにおける `remote` とは、遠隔のクラウド上のデータいずれかの対象を指します。　基本のコマンド

```
rclone mkdir remote:path # 新しいディレクトリをクラウド上に作成します。
rclone lsl remote:path # ファイルのサイズ、変更日時とともに詳細なリストを表示します。
rclone copy src:path dest:path # src -> dest へコピーします。
```

活用例　ほとんどのコマンドに役立つ重要なフラグ:

```
  -n, --dry-run         Do a trial run with no permanent changes
  -i, --interactive     Enable interactive mode
  -v, --verbose count   Print lots more stuff (repeat for more)
  -vv
```

その他のフラグ一覧はこちら。　シェル補完を設定します。

```
sudo rclone genautocomplete fish
```

1. 設定

```
rclone config # 対話的に設定を行う
rclone config redacted # 現在の設定状況を確認できます
```

新しいリモートを追加したり、既存の設定を変更したりするための対話式セットアップを開始します。

設定先: `~/.config/rclone/rclone.conf`

注意

- `~/config/rclone/rclone.conf`設定ファイルは、バックアップをとり厳重に取り扱ってください。
- 紛失した場合暗号化済ファイルも元に戻せず、第三者に流失した場合、セキュリティ脅威となります。
- 後述するコンフィグファイルの暗号化も必要に応じて行うとよいでしょう。

2. リスト表示

```
rclone ls remote:path
```

指定したパスのファイルとディレクトリを一覧表示します。

```
rclone lsd remote:path
```

ディレクトリのみを一覧表示します。

3. コピー (アップロード)

```
rclone copy /local/path remote:path --progress
```

ローカルからリモートへコピーします。逆も可能です。

```
# 既存のファイルを上書きせず、新しいファイルのみをコピーします。
rclone copy --ignore-existing /local/path remote:path --dry-run
```

```
rclone copy . --include "*.[pdf|md|txt]" mega1:/docs/ -v
rclone copy . --include "*.zip" mega1:/releases/ -v

rclone copy --create-empty-src-dirs . mega1:/  # ディレクトリ構造を維持
```

ツリーの統合にはcombineを使いましょう

```
rclone tree s3:files
rclone tree drive:important/files

[remote]
type = combine
upstreams = images=s3:imagesbucket files=drive:important/files
```

4. 同期

```
rclone sync /local/path remote:path
```

ソースの状態をデスティネーションに完全に反映させます。

5. 移動

```
rclone move /local/path remote:path
```

ファイルを移動（コピー後に元のファイルを削除）します。

6. 削除

```
rclone delete remote:path/file
```

指定したファイルを削除します。

```
rclone purge remote:path
```

指定したディレクトリとその中身をすべて削除します。

```
rclone dedupe remote:path/file --dedupe-mode newest
```

重複ファイルのみをすべて削除します。
同一のファイルを削除し、最新のファイルを保持します。

7. チェック

```
rclone check /local/path remote:path
```

2つの場所のファイルを比較し、違いを報告します。

8. サイズ確認

```
rclone size remote:path
```

指定したパスの合計サイズと、ファイルとディレクトリの数を表示します。

9. マウント

```
rclone mount remote:path /local/mountpoint
```

リモートストレージをローカルファイルシステムにマウントします。

10. 双方向同期

```
rclone bisync /local/path remote:path
```

2つの場所間で双方向同期を行います。

```
rclone bisync remote1:path1 remote2:path2 --create-empty-src-dirs --compare
size,modtime,checksum --slow-hash-sync-only --resilient -MvP --drive-skip-gdocs --fix-
case --resync --dry-run
```

Triggerなしの自動双方向同期は現時点ではありません。
しかし、CloudCross , InsyncやOdriveなどの外部ソリューションを使用できます。

11. フィルタリング

すべてのコマンドで `--include`, `--exclude` フラグを使用してファイルをフィルタリングできます。
検索のように使うことができます。

```
rclone ls --include "*.pdf" remote:
rclone copy /local/path remote:path --include "*.jpg" --exclude "*.tmp"
```

12. 同期

`--dry-run` フラグを使用すると、実際の変更を行わずに操作をシミュレートできます。

```
rclone sync /local/path remote:path --dry-run
rclone sync --interactive /local/path remote:path # 削除前に確認を表示

rclone sync /local/path remote:path --dry-run --bwlimit 1M --update --exclude "*.mp4" --
retries 3
```

ignore fileを指定して無視するには　`--drive-use-trash`：ファイルを削除する際にゴミ箱に移動させる

13. ダウンロードリンク

```
rclone link mega1:path/to/folder --expire 7d --password mysecretpassword
```

- If you want to get links for multiple files, you can use a loop in bash. Here's an example:

```
rclone ls mega1:path/to/directory | while read -r file; do
  echo "$file: $(rclone link mega1:path/to/directory/"$file")"
done > download_links.txt
```

- If you want to generate a random password, you can use the openssl command in combination with rclone:

```
password=$(openssl rand -base64 12)
link=$(rclone link mega1:path/to/folder --expire 7d --password "$password")
echo "Link: $link"
echo "Password: $password"
```

- If you want to do this for multiple folders, you can use a loop:

```
set path share/
  set remote mega1:$path
  set date (date '+%FT%T.%3N%Z' --utc)

  rclone lsf --include '*.pdf' $remote | while read -l file
          set id null
          set name $file
          set password (openssl rand -base64 12)
          set link (rclone link $remote$file --expire 7d)

          jq -n --arg id "$id" \
                    --arg name "$name" \
                    --arg path "$path" \
                    --arg link "$link" \
                    --arg date "$date" \
                    --arg password "$password" \
                    '{
                  id: $id,
                  name: $name,
                  path: $path,
```

```
                link: $link,
                date: $date,
                password: $password
              }' >> index.json
  end
```

14. 暗号化

remote全体を暗号化および復号化します。これは、暗号化用remoteを新たに作成して通信をWrapすることで成立します。

名前入力は任意(e.g. `mega1-encrypt`)、ストレージ種別設定で`Encrypt/Decrypt a remote`を選択することで、設定できます。

```
name> mega1-encrypt

Storage> 13
```

構成ファイルには、以下の設定が追加されます。

```
[mega1-encrypt]
type = crypt
remote = remote:path
password = *** ENCRYPTED ***
password2 = *** ENCRYPTED ***
```

passwordが生成されます。
必要に応じてsalt, pepper付加して暗号化します。　passwordは、手動で生成・管理することも可能です。

```
openssl rand -base64 36
```

```
Storage> crypt
remote> mega1:share # remote:path
file_encryption> 1 # 1:ファイルタイプ, 2:ファイル名のみ, 3:オフ
directory_name_encryption> 1 # 1:すべてのフォルダ名, 2:オフ
generate_random_password> g # パスワードを自動生成
Bits> 128 # Password strength in bits
password or phrase for salt > n # 2h番目のパスワードを自動生成
```

暗号化が完了するとパスワードを変更できません。
後は通常のアップロード手順と同じように利用できます。

```
# Uploading
# rclone mkdir mega1:share
rclone -q copy tmp.json mega1-encrypt:share

# Bulk upload
fd 'mailchimp' --full-path | parallel rclone -q copy {} mega1-encrypt:share

# Show list
rclone ls mega1:share # for encrypted-paths
rclone ls mega1-encrypt:share # for origin-paths

# Error the wrong remote name
# 2024/09/02 18:57:46 Failed to create file system for "mega1-encrypt:": didn't find
section in config file

# Error: the dir not exist
```

```
# 2024/09/02 18:57:54 Failed to create file system for "mega1-encrypt:": failed to make
remote "mega1-encrypt:share" to wrap: didn't find section in config file

# Error: traffic blocked
# 2024/09/02 19:03:57.474276 ERROR RESTY 422 POST https://mail.proton.me/api/auth/v4: We
are detecting potentially abusive traffic coming from your network and have temporarily
blocked logins. If you believe this is in error, please contact us here:
https://proton.me/support/appeal-abuse (Code=2028, Status=422)
```

チェック

```
rclone cryptcheck remote:path encryptedremote:path --one-way
# 宛先内にない追加のファイルがあることを意味します

rclone cryptcheck remote:path encryptedremote:path  --combined
# Result : =は同一，-は宛先でのみ，+はソースでのみ，*は存在していたが異なる，!はエラー
```

復号化

```
rclone cryptdecode encryptedremote: encryptedfilename1 encryptedfilename2

rclone cryptdecode --reverse encryptedremote: filename1 filename2
```

暗号化されていないファイル名を返します。　さらに、構成暗号化を必要に応じて行います。

```
>rclone config
Current remotes:

e) Edit existing remote
n) New remote
d) Delete remote
s) Set configuration password *
```

リモート全体を暗号化として設定することは可能ですが、クラウド ストレージ プロバイダーは、暗号化されたルート フォルダーを処理できない場合があります

メタデータがサポート　テスト方法:

```
rclone -v copy /your/local/file/here/ remotename:/the/remote/location/
rclone lsf remotename:/
```

15. 細分化 / 結合

大きなファイルを小さなチャンクに透過的に分割します　dg

```
[overlay]
type = chunker
remote = remote:bucket
chunk_size = 100M
hash_type = md5
```

rcloneがファイルのアップロードを開始すると、チャンカーはファイルサイズをチェックします。
複合ファイルのダウンロードが要求された場合、チャンカーは透過的に データチャンクを順番に連結して組み立てます。 `--chunker-fail-hard` コマンドを中止するフラグを設定することができます。 `--min-size` 多くのファイルがチャンクのしきい値を下回っている場合、不要なAPI呼び出しを回避するために用います。スペースの無駄が発生するため、`union` で集約します。
問題は、ハッシュdbをrcloneを使用するマシン全体で維持する必要があることです。

```
[remote]
type = union
upstreams = remote1:dir1 remote2:dir2 remote3:dir3
```

16. ハッシュ化

```
[Hasher1]
type = hasher
remote = myRemote:path
hashes = md5
max_age = off

[Hasher2]
type = hasher
remote = /local/path
hashes = dropbox,sha1
max_age = 24h
```

これは本質的に、「暗号化」のアプローチの逆です(ソースを再暗号化するのではなく, これはdestを復号化します)
これは、ソースがない場合でも機能することを意味します(2つの異なるクリプトリモコンを異なるパスワードで比較する場合など)。
`--hasher-max-age 0` でデータベース機能を使用せずにHasherを使用することは可能です。

パス内のすべてのオブジェクトのmd5sumファイルを生成します。

```
rclone hashsum MD5 remote:path # 更新を強制
rclone md5sum remote:path  # 上記と同値

rclone hashsum sha256 remote:path -f --max-age 1 h # 1時間以内のパスのSHA256
```

17. 難読化

cryptがremoteデータを暗号化するのに対し、obscureはパスワードを暗号化(難読化)します。

```
rclone obscure [pass]
rclone obscure [salt]

rclone copyto file.txt.bin file.txt -v \
--crypt-remote=lc: \
--crypt-filename-encryption=off \
--crypt-directory-name-encryption=false \
--crypt-password=[obscured pass] \
--crypt-password2=[obscured salt]
```

18. マウント

FUSEでエクスプローラに表示可能な、見せかけのファイルシステムを作成します。

```
rclone nfsmount remote:path/to/files * # 自動
rclone nfsmount remote:path/to/files X: # 特定のドライブ文字
rclone nfsmount remote:path/to/files C:\mount : # relPath
rclone nfsmount remote:path/to/files \\cloud\remote # network drive


#バックグラウンドモードで実行する場合、ユーザーは手動でマウントを停止する必要があります:
# Linux
fusermount -u /path/to/local/mount
# OS X
umount /path/to/local/mount
```

上り速度は保証されない（バックグラウンドでストリーミング処理する）ため、
専ら閲覧・下り用途に用いるべきです。

```
rclone nfsmount remote:path/to/files \mount X: # relPath
```

rcloneメタデータストアで非常に役立ちます。リモートがこれらをサポートしていない場合でも、ファイルの作成、変更、またはアクセス時間を格納します。

信頼性がより必要な場合、VFS File caching を検討してください。これは、変更と同時にファイルに書き込みます。

```
--cache-dir string                    Directory rclone will use for caching.
--vfs-cache-mode CacheMode            Cache mode off|minimal|writes|full (default off)
--vfs-cache-max-age duration          Max time since last access of objects in the cache
(default 1h0m0s)
--vfs-cache-max-size SizeSuffix       Max total size of objects in the cache (default
off)
--vfs-cache-min-free-space SizeSuffix Target minimum free space on the disk containing
the cache (default off)
--vfs-cache-poll-interval duration    Interval to poll the cache for stale objects
(default 1m0s)
--vfs-write-back duration             Time to writeback files after last use when using
cache (default 5s)
```

19. エイリアス

エイリアスを作成すると、以下２つは同一の対象を指します。

```
rclone mkdir backup:desktop
rclone mkdir mydrive:private/backup/desktop # Alias
```

22. メタデータ

settierは、リモート内のオブジェクトのストレージクラス/階層を変更します。

```
rclone --include "*.txt" settier Hot remote:path/dir
```

23. Git Annex

git-annex コンテンツを保存および取得します。

```
# Create the helper symlink in "$HOME/bin".
ln -s "$(realpath rclone)" "$HOME/bin/git-annex-remote-rclone-builtin"
```

```
# Verify the new symlink is on your PATH.
which git-annex-remote-rclone-builtin
```

```
# If you skipped step 1:
git annex initremote MyRemote type=rclone --whatelse

# If you created a symlink in step 1:
git annex initremote MyRemote type=external externaltype=rclone-builtin --whatelse
```

キャッシュ

```
Storage> cache
```

圧縮

```
Storage> compress
```

連結 (Combine)

FTP/CIFS

HTTP

SMB

SSH/SFTP

WebDAV

In Memory

Localdisk

MISC

- Streaming

```
tar | gzip | aes >s3_mount/file
```

- GUI

```
rclone rcd --rc-web-gui --rc-user x --rc-pass x
rclone rcd --rc-web-gui --rc-web-gui-no-open-browser
```

Third party tools

- https://github.com/dimitrov-adrian/RcloneTray
- https://github.com/kapitainsky/RcloneBrowser

Scripting

30分ごと(または任意の期間)に実行される“連続”同期を作成する

```
#!/bin/bash
# Get config path from rclone
config=$(rclone config file)
# Remove everything except the path
RCLONE_CONFIG="/${config#*/}"
```

```
export RCLONE_CONFIG

# Exit if rclone running
if [[ "$(pidof -x $(basename $0) -o %PPID)" ]]; then exit; fi

# Sync files to the cloud
/usr/bin/rclone sync [SOURCE_PATH] remote:[REMOTE_PATH]
    --log-file /opt/rclone_upload.log
    --verbose
```

```
*/30 * * * * /home/nl/cloudsync.sh
```

- https://www.reddit.com/r/rclone/comments/1d2pm6p/rclone_blocked_by_proton_drive_it_seems/?share_id=E7Wk9x_YtWoDoTOjBkvSE&utm_content=1&utm_medium=android_app&utm_name=androidcss&utm_source=share&utm_term=1

Proxy

- https://rclone.org/faq/#can-i-use-rclone-with-an-http-proxy

無限ストレージの活用

- github LFS , 1GB/day
- dropbox enterprise 2000 yen / month
- google drive enterprise 2500 yen / month
- google drive bussiness standard 1300x5 user / month

Summary

　rcloneは素早い処理に有効なツールですが、セキュリティ監査されておりません。
そのため、予期しないエラーや通信のスタックに対面することもあるでしょう。
足りない機能を補完するには、Cryptomatorや PGPZip、Resticなどとの併用を検討するとよいでしょう。

- https://github.com/restic/restic
- syncthing
- http://www.secfs.net/winfsp/
- https://nssm.cc/

FAQ

- https://linuxpip.org/rclone-examples#rclone-copy-examples-copying-data-between-cloud-storage-services
- https://forum.rclone.org/t/crypt-remote-hash-and-other-metadata/43885/6
- rcloneを用いてFTPによるファイルのアップロードとダウンロードをGitHub Actions上で行う | CI-CD
- GitHub - cupcakearmy/autorestic: Config driven, easy backup cli for restic.
- GitHub - xyou365/AutoRclone: AutoRclone: rclone copy/move/sync (automatically) with thousands of service accounts
- Macrium Reflect で差分バックアップと増分バックアップを使用するにはどうすればよいですか?