1. 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。

1.1 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

1.2 方法

1.2.1 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
```

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
```

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
```

1.2.2 1. 設定

```
rclone config # 対話的に設定を行う rclone config redacted
```

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`

1.2.3.3 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。

1.2.4 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。

1.2.5 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 . --include "*.[pdf|md|txt]" proton:/docs/
```

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
```

1.2.6 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。

1.2.7 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。

1.2.8 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。

1.2.9 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。

1.2.10 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。

1.2.11 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。

1.2.12 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-emp
```

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。

1.2.13 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。

```
rclone copy /local/path remote:path --include "*.jpg"
```

1.2.14 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 s
```

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

1.2.15 13. ダウンロードリンク

```
rclone link protondrive:path/to/folder --expire 7d --p
```

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

```
rclone lsf protondrive:path/to/directory | while read
```

- 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
```

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

```
rclone lsf protondrive:path/to/parent_folder -R | grep
```

1.2.16 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. `proton-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> proton-encrypt Storage> 13
```

構成ファイルには、以下の設定が追加されます。

```
[proton-encrypt] type = crypt remote = remote:path pas
```

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> proton-remote:share # remote:pa
```

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 proton-remote:share rclone
```

チェック

```
rclone cryptcheck remote:path encryptedremote:path --o
```

復号化

```
rclone cryptdecode encryptedremote: encryptedfilename1
```

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 remot
```

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/
```

1.2.17 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_
```

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
```

1.2.18 16. ハッシュ化

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

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 md5su
```

1.2.19 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 cop
```

1.2.20 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 nf
```

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: # relPa
```

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 cachi
```

1.2.21 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:priva
```

1.2.22 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
```

1.2.23 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 "$HOME/bin". ln -s "$(r
```

```
# If you skipped step 1: git annex initremote MyRemote
```

1.2.24 キャッシュ

```
Storage> cache
```

1.2.25 圧縮

```
Storage> compress
```

1.2.26 連結 (Combine)

1.2.27 FTP/CIFS

1.2.28 HTTP

1.2.29 SMB

1.2.30 SSH/SFTP

1.2.31 WebDAV

1.2.32 In Memory

1.2.33 Localdisk

1.2.34 MISC

- Streaming

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

- GUI

```
rclone rcd --rc-web-gui --rc-user x --rc-pass x rclone
```

1.2.35 Third party tools

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

1.2.36 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=$(rcl
```

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

- https://www.reddit.com/r/rclone/comments/1d2pm6p/rclone_bloc share_id=E7Wk9x_YtWoDoTOjBkvSE&utm_content=1&utm_m

1.2.37 Proxy

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

1.2.38 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

1.3 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/

1.3.39 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? ▷➡️視覚核▷➡️