Transferring files from/to cloud storage

This document walks through some of the basics of using the command line program Rclone allowing to synchronize files between the cluster (compute and login nodes) and external cloud storage systems. Rclone has a rich set of features and can be used with many different cloud systems including Dropbox, ownCloud, Google Drive, etc. However, here we will consider Seafile as it is a cloud platform the cloud service at the LUIS is based on.

Rclone, which provides cloud equivalents to the unix commands like `ls`, `cp`, `mkdir`, `rm`, `rsync`, etc. is installed on the login and transfer nodes.

Please note: If you plan to migrate a large amount of data, it is recommended to use the transer node of the cluster, as it is connected to the university's 10Gbs network and has no limitation set on the CPU-time utilized by user processes.

Please node: The cloud storage systems provided by the LUIS can be accessed directly from the computing nodes.

Configuring cloud endpoint for Seafile

Note that the Rclone configuration steps in this subsection need to be completed only once for each cloud storage endpoint.

Since Rclone does not currently support Seafile authentication over SSO, we will assume that you have configured your LUIS “Cloud-Seafile” storage for access via WebDAV. If you have not done so already, follow the instructions in the section “Zugriff über WebDAV” on the service documentation page.

However, if you want to access your LUIS Projekt-Seafile, you would only need your user credentials (please refer to https://www.luis.uni-hannover.de/en/services/storage-systems/dateiservice/services/speichersysteme/dateiservice/projektablage/first-steps-and-basic-facts#c18022).

NOTE: if you are configuring your “Projekt-Seafile”, enter 43 or `seafile` above and follow respective instructions.

A list of all possible storage providers can be found here or by running `rclone help backends`.

In the next two steps enter `https://seafile.cloud.uni-hannover.de/dav` as the URL of LUIS “Cloud-Seafile” and `other` for the option `vendor`:

```
Name of the Webdav site/service/software you are using
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value
....
5 / Other site/service or software
  \ "other"
vendor> other
```

Next you will be prompted to enter your WebDAV username and password:

```
User name.
Enter a string value. Press Enter for the default ("").
user> username@uni-hannover.de

Password.
y) Yes type in my own password
g) Generate random password
n) No leave this optional password blank (default)
y/g/n> y

Enter the password:
password:
Confirm the password:
password:
```

Leave the next 3 parameters blank(default).

You will finally get a summary, where you should type y if everything is OK and then q to finish the configuration:

```
--------------------
[cloud-luis]
type = webdav
url = https://seafile.cloud.uni-hannover.de/dav
vendor = other
user = username@uni-hannover.de
pass = *** ENCRYPTED ***
--------------------
y) Yes this is OK (default)
e) Edit this remote
d) Delete this remote
y/e/d> y

Current remotes:

Name                 Type
====                 ====
cloud-luis           webdav

e) Edit existing remote
n) New remote
```

```
d) Delete remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
e/n/d/r/c/s/q>
e/n/d/r/c/s/q> q
```

Rclone usage examples

Note: You can use TAB key for completing Rclone commands and their options.

The command shows all Rclone *remotes* you have confugured:

```
username@transfer$ rclone listremotes
cloud-luis:
project-luis:
mycloud:
```

Navigating objects in the cloud storage

The following displays top level directories (or buckets) on the cloud which has been configured as the rclone *remote* `mycloud`:

```
username@transfer$ rclone lsd mycloud:
```

Note the colon at the end of the name of *remote*.

To list all files in the path `mydir` on the cloud:

```
username@transfer$ rclone ls mycloud:mydir
```

Note that by default `ls` recursively lists contents of directories. Use the option `–max-depth N` to stop the recursion at the level `N`.

If you want to get more information about files like their size, modification time and path, run:

```
username@transfer$ rclone lsl mycloud:
```

The `tree` subcommand lists recursively the *remote* contents in a tree format. The option `-C` colorizes the output:

```
username@transfer$ rclone tree -C mycloud:
/
├── dir1
│   ├── dir1_2
│   │   └── file2
│   ├── myfile1
```

```
│   └── myfile5
└── dir3
    ├── dir3_1
    │   └── myfile3
    └── file4
```

To selectively display files and directories, with the commands `ls` and `lsl`, you can apply global options such as `--exclude/--include`, `--filter`, `--min-size`, etc. More information can be found here.

To create a new directory `mydir` on the *remote* `mycloud`, type:

```
username@transfer$ rclone mkdir mycloud:dir1/mydir
```

Note that in case of Seafile cloud storage, you can not remove/create top level directories(also called Libraries) using Rclone.

Copying & synchronizing data

To copy a file called `myfile.txt` from your local directory to a subdirectory `dir1` on the cloud:

```
username@transfer$ rclone copy myfile.txt mycloud:dir1
```

The following will transfer the contents of your `$BIGWORK/mydir` directory to the subdirectory `dir1/mydir` on the cloud storage:

```
username@transfer$ rclone copy --progress $BIGWORK/mydir mycloud:dir1/mydir
```

If the destination directory (`mycloud:dir1/mydir` in the example above) does not exist, Rclone will create it. Files that already exist at the destination are skipped. If you want to skip files that are newer at the destination use the global flag `--update`, which ensures that the latest version of the files is available in the cloud.

Note: To speed up copying a directory containing a large number of small files, the directory should be transferred as a compressed tarball archive file (see the section on working with large datasets). This can also help you to overcome the limitation imposed by some cloud storage providers(e.g. Google Drive) on the number of simultaneously transferred files.

As long as the network and storage systems(remote/local) can handle it, you may improve the overall transfer rates by increasing the values for these two Rclone global options:

```
--transfers=N (Number of file transfers to be run in parallel. default N=4)
--drive-chunk-size=SIZE (Transfer chunk size in kilobytes. Must a power of 2
>= 256k. default SIZE=8192)
```

The parameter `--drive-chunk-size` might be useful only when transferring large files.

If you would like your destination storage (remote or local) to have exactly the same content as the source, use the `sync` subcommand instead. Below is an example to sync the contents of your cloud directory `mycloud:dir1/mydir` (the source) to the `$BIGWORK/mydir` (the destination) directory at

the cluster:

```
username@transfer$ rclone sync mycloud:dir1/mydir $BIGWORK/mydir
```

Contrary to the `copy` subcommand, if files are removed from the source, synchronizing the source and destination will delete files from the destination as well. Copying will never delete files in the destination.

WARNING: as the command can cause data loss at the destination, it is recommended to always test it first with the `--dry-run` flag to see exactly what would be copied and deleted.

Additional flags can be used similarly to the `copy` subcommand.

Deleting objects from the cloud storage

The command removes the directory `mydir` and all its contents from the `dir1` at the *remote* `mycloud:`

```
username@transfer$ rclone purge mycloud:dir1/mydir
```

If you need to selectively delete files from the cloud use the `delete` subcommand instead. For example, the following will remove all files larger than 500MB from the `dir1/mydir` directory:

```
username@transfer$ rclone delete --min-size 500M mycloud:dir1/mydir
```

To remove files older than 60 days:

```
username@transfer$ rclone delete --min-age 60d mycloud:dir1
```

To see other Rclone global flags, execute `rclone help flags`. More information on how to filter objects is available here.

You may first want to check what would be deleted with the `--dry-run` flag or use the `--interactive` option.

Note that the `delete` removes only files keeping the directory structure in the *remote* unchanged. Adding the option `--rmdirs` will remove empty sub-directories along with files.

Creating a cron job

If you wish to periodically run rclone, you can achieve this with a cron job. To create a cron job, modify your current crontab using the `crontab -e` command. Once in the crontab editor you can input your rclone commands. Below is an example cron job executing the `rclone sync` every 20 minutes:

```
*/20 * * * * /bin/rclone sync <myremote>:<mydata> </local/path/to/mydata>
```

After you exit from the editor, the modified crontab will be installed automatically. To list all your

current cron jobs invoke `crontab -l`.

Note: the cron job will only be executed on the machine you created it on. Therefore, the recommended way to work with cron jobs is to manage them on a fixed machine. A good candidate would be the transfer node.

Further Reading

- Rclone command line reference
- Rclone supported cloud storage systems
- Rclone configuration for various cloud storage

From:
https://docs.cluster.uni-hannover.de/ - **Cluster Docs**

Permanent link:
https://docs.cluster.uni-hannover.de/doku.php/guide/cloud_storage_rclone

Last update: **2026/08/31 08:25**