Rclone VFS and MergerFS Setup

This guide is for advanced users only. Ultra.cc is not responsible for any data loss or application errors due to this setup. We do not provide official support for Rclone or MergerFS. You may visit our #community-support channel on our Discord server for assistance.

Please make yourself aware of the Ultra.cc Fair Usage Policy. It is very important *not* to mount your Cloud storage to any of the premade folders. Do *not* download directly to a Rclone mount from a torrent or nzbget client. Both will create massive instability for both you and everyone else on your server. *Always follow the documentation and create a new folder for mounting. It is your responsibility to ensure usage is within acceptable limits.*

Please do not mount to any of the default directories such as:
downloads
files
media
bin
.apps
.config
www
/homexx/username/
or any pre-created directory found on your Ultra.cc Slot

Introduction

This section will teach you how to set up a Rclone VFS mount and MergerFS on Ultra.cc slots, and it assumes the following:

- You have a working Rclone setup, especially correctly configured remotes of your preferred cloud storage provider. In this tutorial, we'll be using Google Drive. If you use another cloud storage provider, change the flags that are appropriate to your setup and visit Rclone documentation for more information.
- You have the appropriate knowledge of setting up and running your own systemd services.
- You are comfortable working in CLI, compiling from sources and setting up cron jobs.

The workflow of the setup is as follows:

- There are 2 folders, one local and the Rclone mount named Mount.
- These 2 folders are merged via MergerFS and is mounted to another folder. All apps such as Plex, Radarr, Sonarr, and such will be pointed to this folder.

- When you copy a file to MergerFS, this will be copied to Local First. Directory structures will be retained.
- A Rclone move script moves all the contents inside Local every 19:00 CET, retaining the directory structures.
- Applications that are pointed to MergerFS wouldn't know the difference.

The Pros of this setup are as follows:

- New files will be immediately available in Plex and have faster loading times due to it being available locally.
- Uploads are less prone to errors than moving files directly via Rclone mount.
- It's essentially a "set it and forget it" setup.

And, the cons of this setup are as follows:

- New files will not be available on Google Drive until you run the Rclone move script.
- There will be 3 points of failure on this setup, Rclone, apps that are connected to the MergerFS folder and MergerFS.
- Monitoring functions such as Plex's "Update my library automatically" will not work for mounts. You may need to set your application to periodically scan the mount.

> Before we proceed, it is imperative to stop all Rclone/plexdrive processes and stop all the apps that are connected to your Rclone mount before proceeding.

Preparation

Install and Configure Rclone

Install and configure Rclone if you haven't already. Refer here for more information: Installation, Configuration & Usage of rclone

Install mergerfs

- Run the command given below in your terminal. This should automatically install MergerFS to your slot.
- Select 1 or 2 when prompted. We recommend selecting `2` when prompted by the installer.

```
bash <( wget -qO- https://scripts.ultra.cc/main/MergerFS-Rclone/Installer%20Scripts/mergerfs-install.sh)
```

- To confirm if the installation is completed, do `which mergerfs`

```
ultradocs@pollux:~$ which mergerfs
/home/ultradocs/bin/mergerfs
```

Change Application Settings

Change your application settings according to rclone Optimizations for Apps.

Setup the Work Flow using a Script

What does the script do?

- The script will set up the workflow for you.
- It will create the following directories on your service:
  - `~/Stuff/Mount`
  - `~/.tmp-rclone`
  - `~/Stuff/Local/Downloads/torrents`
  - `~/Stuff/Local/Downloads/usenet`
  - `~/MergerFS/Movies`
  - `~/MergerFS/TV Shows`
  - `~/scripts`
- The following systemd services will be created by it to mount rclone-vfs and mergerfs respectively:
  - `~/.config/systemd/user/rclone-vfs.service`
  - `~/.config/systemd/user/mergerfs.service`
- Finally, it will also save the rclone-upload script and set a cronjob to run it in your crontab.
- You will be given a choice between two kinds of rclone-upload scripts. One will be a normal script that will only upload your data and the other will also send a discord notification.
- In the case that you chooose the `Discord-Notification` type, make sure that you have a `Discord Webhook URL` ready for it.
- The `rclone-upload.sh` script will be saved in the following location:
  - `~/scripts/rclone-upload.sh`
- The `rclone-upload.sh` script will be set to move data to your cloud drive daily at 19:00 CET and optionally also send a discord notification upon completion.

Run the Script

- It is absolutely necessary that you know the rclone remote name which you set up in Installation, Configuration & Usage of rclone before running this script.
- You can confirm your rclone remote name by executing the following SSH command:

```
rclone listremotes
```

- Optionally, you should also have a Discord Webhook URL ready if you wish to receive discord notifications from `rclone-upload.sh`.
- To execute the rclone-mergerfs workflow setup script, run the SSH command given below:

```
bash <(wget -qO- https://scripts.ultra.cc/main/MergerFS-Rclone/rclone-mergerfs.sh)
```

- Confirm that rclone-vfs and mergerfs are set up correctly by executing the following SSH command:

```
ls ~/MergerFS
```

- It should list files from your cloud drive.

Set Sonarr Correctly

- To take advantage of your new MergerFS work flow, you must set up Sonarr correctly.
- As soon as you login to Sonarr, go to `Settings` and click on the cog that says `Show Advanced`.

Enable Hardlinks

- Go to `Settings -> Media Management`
- Check `Use Hardlinks Instead of Copy` under `Importing`
- Click on `Save Changes` in the top-left.

Set Root Folder

- Go to `Settings -> Media Management`
- Click on `Add Root Folder` under `Root Folders`
- Browse to `/home/your_username/MergerFS/TV Shows`
- Click `Ok`.
- All Series in Sonarr should have `/home/your_username/MergerFS/TV Shows` as their `Root Folder`.
  - You can have other Root Folders, but they all should be inside `/home/your_username/MergerFS`.

Remote Path Mapping

- Go to `Settings -> Download Clients`
- Ensure that you have already set up your Download Clients! Click here for the Guide.
- Click on the `+` button to the right-hand side under `Remote Path Mappings`.
- Set up the Remote Path Mapping as given below:

```
Host: {username}.{servername}.usbx.me

Remote Path: /home/your_username/Stuff/Local/Downloads

Local Path: /home/your_username/MergerFS/Downloads
```

- Click on Save.

Set Radarr Correctly

- To take advantage of your new MergerFS work flow, you must set up Radarr correctly.
- As soon as you login to Radarr, go to `Settings` and click on the cog that says `Show Advanced`.

Enable Hardlinks

- Go to `Settings -> Media Management`
- Check `Use Hardlinks Instead of Copy` under `Importing`
- Click on `Save Changes` in the top-left.

Set Root Folder

- Go to `Settings -> Media Management`
- Click on `Add Root Folder` under `Root Folders`

- Browse to `/home/your_username/MergerFS/Movies`
- Click `Ok`.
- All Movies in Radarr should have `/home/your_username/MergerFS/Movies` as their `Root Folder`.
  - You can have other Root Folders, but they all should be inside `/home/your_username/MergerFS`.

Remote Path Mapping

- Go to `Settings -> Download Clients`
- Ensure that you have already set up your Download Clients! Click here for the Guide.
- Click on the `+` button to the right-hand side under `Remote Path Mappings`.
- Set up the Remote Path Mapping as given below:

```
Host: {username}.{servername}.usbx.me

Remote Path: /home/your_username/Stuff/Local/Downloads

Local Path: /home/your_username/MergerFS/Downloads
```

- Click on Save.

Connect Sonarr & Radarr to Plex / Emby

Plex Media Server

- In Plex Media Server, go to `Settings -> Network` and set `Secure Connections` to `Preferred`. Then, click on `Save Changes`.
- In both Sonarr & Radarr, go to `Settings -> Connect` and click on the `+` button.
- Choose `Plex Media Server`.
- Set it up with the details given below:

```
Name : Anything as per your preference.

Notification Triggers: Check `On Download`, `On Upgrade`, `On Rename`.

Host: 172.17.0.1

Port: 16825 (The port of your Plex Media Server, visible in the Control Panel)

Use SSL: Unchecked

Update Library: Checked
```

- Click on `Authenticate with Plex.tv`.
- A browser pop-up window will open. Login using your Plex account.
- After authenticationg with Plex.tv, the `Auth Token` field will be filled automatically.
- Click on `Test` and then `Save`.

Emby

- Login to your Emby instance.
- Go to `Manage Emby Server -> Advanced -> Api Keys`

- Click on `+ New Api Key` and add the App name as `Arrs`. Then click `ok`.
- In both Sonarr & Radarr, go to `Settings -> Connect` and click on the `+` button.
- Choose `Emby`.
- Set it up with the details given below:

```
Name : Anything as per your preference.

Notification Triggers: Check `On Download`, `On Upgrade`, `On Rename`.

Host: 172.17.0.1

Port: Emby's port as given in your Ultra Control Panel.

Use SSL: Unchecked

API Key: Paste the one which was created earlier.

Update Library: Checked
```

- Click on `Test` and then `Save`.

Set your Download Clients Correctly

> In both Sonarr & Radarr, you must click on the cog that says Show Advanced in Settings. Some required fields for the Download client settings will not be visible otherwise.

- Before proceeding with this secion, set up your download clients normally in both Sonarr and Radarr. Click here for that Guide.
- Sonarr and Radarr both must add their downloads to directories inside `/home/your_username/Stuff/Local/Downloads`.
- To achieve this, you will have to make changes in your download clients.

Torrent Clients

rTorrent

- In Sonarr/Radarr itself, go to `Settings -> Download Clients`.
- Click on the rTorrent download client.
- Locate the `Directory` setting and set it to: `/home/your_username/Stuff/Local/Downloads/torrents`
- Click on `Save`.

Deluge

- In Deluge, it's default download path will have to changed. Therefore, it is not recommmended to use Deluge in this work flow.
- Login to your Deluge instance.
- Click on `Preferences` and go to `Downloads`
- Set the `Download to:` folder to `/home/your_username/Stuff/Local/Downloads/torrents`
- Click on `Apply`, and then `Ok`.

qBittorrent

- Login to your qBittorrent instance.
- Go to `Options -> Downloads`.
- Under `Saving Management`, set the following settings:

```
Default Torrent Management Mode: Automatic

When Torrent Category changed: Relocate torrent
```

- Click on `Save`.
- For Sonarr
  - Find the `tv-sonarr` category. If your Sonarr is using a different category with qBittorrent, edit that one.
  - Right-click on it and choose `Edit category..`
  - Set the `Save path:` to `/home/your_username/Stuff/Local/Downloads/torrents`
- For Radarr
  - Find the `radarr` category. If your Radarr is using a different category with qBittorrent, edit that one.
  - Right-click on it and choose `Edit category..`
  - Set the `Save path:` to `/home/your_username/Stuff/Local/Downloads/torrents`

Transmission

- In Sonarr/Radarr itself, go to `Settings -> Download Clients`.
- Click on the Transmission download client.
- Locate the `Directory` setting and set it to: `/home/your_username/Stuff/Local/Downloads/torrents`
- Click on `Save`.

Usenet Clients

- You can optionally create two more directories to better sort your Series and Movies.
- To do so, run the following SSH commands:

```
mkdir -p ~/Stuff/Local/Downloads/usenet/Sonarr

mkdir -p ~/Stuff/Local/Downloads/usenet/Radarr
```

NZBGet

- For Radarr
  - By default, Radarr sets the NZBGet download client to use the `Movies` category.
  - You must change the `DestDir` of this category in your NZBGet instance. If you are using a different category, change the `DestDir` of that one.
  - Login to your NZBGet instance.
  - Go to `CATEGORIES` and find the `Movies` category.
  - Set the `DestDir` to `/home/your_username/Stuff/Local/Downloads/usenet/Radarr`
  - Click on `Save all changes`.
- For Sonarr
  - By default, Sonarr sets the NZBGet download client to use the `tv` category. However, this is not already present in NZBGet and often has to be added.
  - You must change the `DestDir` of the category in your NZBGet instance. If you are using a different category, change the `DestDir` of that one.
  - Login to your NZBGet instance.
  - Go to `CATEGORIES` and find the `tv` category.

    - Set the `DestDir` to `/home/your_username/Stuff/Local/Downloads/usenet/Sonarr`
    - Click on `Save all changes`.

SABnzbd

- For Radarr
    - By default, Radarr sets the SABnzbd download client to use the `movies` category.
    - You must change the `Folder/Path` of this category in your SABnzbd instance. If you are using a different category, change the `Folder/Path` of that one.
    - Login to your SABnzbd instance.
    - Go to `Config -> Categories` and find the `movies` category.
    - Set the `Folder/Path` to `/home/your_username/Stuff/Local/Downloads/usenet/Radarr`
    - Click on `Save`.
- For Sonarr
    - By default, Sonarr sets the SABnzbd download client to use the `tv` category.
    - You must change the `Folder/Path` of this category in your SABnzbd instance. If you are using a different category, change the `Folder/Path` of that one.
    - Login to your SABnzbd instance.
    - Go to `Config -> Categories` and find the `tv` category.
    - Set the `Folder/Path` to `/home/your_username/Stuff/Local/Downloads/usenet/Sonarr`
    - Click on `Save`.

Set your Media Server Applications Correctly

Plex Media Server

- Your `Movies` library should point to the following folder: `/home/your_username/MergerFS/Movies`.
- Your `TV Shows` library should point to the following folder: `/home/your_username/MergerFS/TV Shows`.
- Any other library folders should also be inside `/home/your_username/MergerFS`.
- Official Plex support article on Editing Libraries.

Emby

- Your `Movies` library should point to the following folder: `/home/your_username/MergerFS/Movies`.
- Your `TV Shows` library should point to the following folder: `/home/your_username/MergerFS/TV Shows`.
- Any other library folders should also be inside `/home/your_username/MergerFS`.

JellyFin

- Your `Movies` library should point to the following folder: `/home/your_username/MergerFS/Movies`.
- Your `TV Shows` library should point to the following folder: `/home/your_username/MergerFS/TV Shows`.
- Any other library folders should also be inside `/home/your_username/MergerFS`.

Helpful Information for managing this new Work Flow

- Log locations:

```
rclone-vfs mount: ~/scripts/rclone_vfs_mount.log
mergerfs mount: ~/scripts/mergerfs_mount.log
rclone-upload script: ~/scripts/rclone-upload.log
```

- Systemd commands:

```
Restart Mounts: systemctl --user restart rclone-vfs.service mergerfs.service
Stop Mounts: systemctl --user stop rclone-vfs.service mergerfs.service
Disable mounts: systemctl --user disable --now rclone-vfs.service mergerfs.service
```

Understanding the rclone-upload Script

- If you followed everything properly thus far, you have a script installed that moves all data from `~/Stuff/Local` to your cloud drive excluding `~/Stuff/Local/Downloads` at every `19:00` CET.
- The script is saved at the following location: `~/scripts/rclone-upload.sh`
- Depending on your choice, it also optionally sends a Discord notification.
- In the case that you wish to execute the script manually, it can be done with the following SSH command: `bash ~/scripts/rclone-upload.sh`
- The log of the script is saved at the following location: `~/scripts/rclone-upload.log`
- Both of the scripts are discussed below individually.

Normal rclone-upload.sh

- Example:

```
#! /bin/bash


lock_file="$HOME/scripts/rclone-upload.lock"


trap 'rm -f "$lock_file"; exit 0' SIGINT SIGTERM
if [ -e "$lock_file" ]
then
    echo "Rclone upload script is already running."
    exit
else
    rm "$HOME"/scripts/rclone-upload.log
```

```
    touch "$lock_file"
    "$HOME"/bin/rclone move "$HOME"/Stuff/Local/ your_remote_name: \
        --config="$HOME"/.config/rclone/rclone.conf \
        --exclude "Downloads/**" \
        --drive-chunk-size 64M \
        --tpslimit 5 \
        -vvv \
        --drive-stop-on-upload-limit \
        --delete-empty-src-dirs \
        --bwlimit=8M \
        --use-mmap \
        --transfers=2 \
        --checkers=4 \
        --log-file "$HOME"/scripts/rclone-upload.log
    rm -f "$lock_file"
    trap - SIGINT SIGTERM
    exit
fi
```

- It is executed at every `19:00` CET by a cronjob saved in your crontab.

rclone-upload.sh with Discord Notification

- Example:

```
#!/bin/bash

# Rclone upload script with optional Discord notification upon move completion (if something
is moved)
#
# Recommended for use via cron
# For example: */10 * * * * /path/to/rclone-upload.sh >/dev/null 2>&1
# ------------------------------------------------------------------------------

SOURCE_DIR="$HOME/Stuff/Local/"
DESTINATION_DIR="your_remote_name:"

DISCORD_WEBHOOK_URL=""
DISCORD_ICON_OVERRIDE="https://i.imgur.com/MZYwA1I.png"
DISCORD_NAME_OVERRIDE="RCLONE"
```

```
LOCK_FILE="$HOME/rclone-upload.lock"
LOG_FILE="$HOME/rclone-upload.log"

# DO NOT EDIT BELOW THIS LINE UNLESS YOU KNOW WHAT YOU'RE DOING
# ----------------------------------------------------------------------------

trap 'rm -f $LOCK_FILE; exit 0' SIGINT SIGTERM
if [ -e "$LOCK_FILE" ]
then
  echo "$0 is already running."
  exit
else
  touch "$LOCK_FILE"

  rclone_move() {
    rclone_command=$(
      "$HOME"/bin/rclone move -vP \
      --config="$HOME"/.config/rclone/rclone.conf \
      --exclude "Downloads/**" \
      --drive-chunk-size 64M \
      --use-mmap \
      --delete-empty-src-dirs \
      --log-file="$LOG_FILE" \
      --stats=9999m \
      --tpslimit=5 \
      --transfers=2 \
      --checkers=4 \
      --bwlimit=8M \
      --drive-stop-on-upload-limit \
      "$SOURCE_DIR" "$DESTINATION_DIR" 2>&1
    )
    # "--stats=9999m" mitigates early stats output
    # "2>&1" ensures error output when running via command line
    echo "$rclone_command"
  }
  rclone_move

  if [ "$DISCORD_WEBHOOK_URL" != "" ]; then

    rclone_sani_command="$(echo $rclone_command | sed 's/\x1b\[[0-9;]*[a-zA-Z]//g')" # Remove
```

```
all escape sequences

    # Notifications assume following rclone ouput:
    # Transferred:  0 / 0 Bytes, -, 0 Bytes/s, ETA - Errors:  0 Checks:  0 / 0, - Transferred:  0
/ 0, - Elapsed time:  0.0s

    transferred_amount=${rclone_sani_command#*Transferred:  }
    transferred_amount=${transferred_amount%% /*}

    send_notification() {
      output_transferred_main=${rclone_sani_command#*Transferred:  }
      output_transferred_main=${output_transferred_main% Errors*}
      output_errors=${rclone_sani_command#*Errors:  }
      output_errors=${output_errors% Checks*}
      output_checks=${rclone_sani_command#*Checks:  }
      output_checks=${output_checks% Transferred*}
      output_transferred=${rclone_sani_command##*Transferred:  }
      output_transferred=${output_transferred% Elapsed*}
      output_elapsed=${rclone_sani_command##*Elapsed time:  }

      notification_data='{
        "username": "'"$DISCORD_NAME_OVERRIDE"'",
        "avatar_url": "'"$DISCORD_ICON_OVERRIDE"'",
        "content": null,
        "embeds": [
          {
            "title": "Rclone Upload Task: Success!",
            "color": 4094126,
            "fields": [
              {
                "name": "Transferred",
                "value": "'"$output_transferred_main"'"
              },
              {
                "name": "Errors",
                "value": "'"$output_errors"'"
              },
              {
                "name": "Checks",
                "value": "'"$output_checks"'"
```

```
              },
              {
                "name": "Transferred",
                "value": "'"$output_transferred"'"
              },
              {
                "name": "Elapsed time",
                "value": "'"$output_elapsed"'"
              }
            ],
            "thumbnail": {
              "url": null
            }
          }
        ]
      }'

      /usr/bin/curl -H "Content-Type: application/json" -d "$notification_data"
$DISCORD_WEBHOOK_URL
    }

    if [ "$transferred_amount" != "0 B" ]; then
      send_notification
    fi

  fi

  rm -f "$LOCK_FILE"
  trap - SIGINT SIGTERM
  exit
fi
```

- It is executed at every `19:00` CET by a cronjob saved in your crontab.
- Optionally, you can edit the script and change the following variables as you like:
  - `DISCORD_WEBHOOK_URL` is set by the script already. In case you didn't enter it correctly or wish to change it, you can do so here.
  - `DISCORD_ICON_OVERRIDE` has a default rclone logo set. You can change this as per your preference.
  - `DISCORD_NAME_OVERRIDE` is the name of the Discord Bot being used to send the notifications. This can be set as per your preference.

Changing the time at which rclone-upload.sh is executed (Optional)

Please do not execute the upload script more than once a day. It can lead to the violation of our Terms of Service.

- As explained previously, the script is being executed daily at `19:00` CET by your crontab.
- You can edit your crontab with the following SSH command: `crontab -e`
- If this is your first time editing the crontab, it will prompt you to select a text editor:

```
Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/mcedit
  4. /usr/bin/vim.tiny

Choose 1-4 [1]:
```

- Type in `1` and then press `Enter`.
- It will show you the crontab entry for your `rclone-upload.sh` script.

```
0 19 * * * /home/your_username/scripts/rclone-upload.sh > /dev/null 2>&1
```

- You can edit the cron schedule expression as you like. Here is a simple tool that can help you find the right cron schedule expression: Crontab.guru
- Once changes are made use the key combination `Ctrl + O` and press `Enter` to save them.
- Then use the key combination `Ctrl + X` to exit the `nano` text editor.

Revision #49
Created 8 May 2022 12:55:54 by Raikiri
Updated 7 August 2024 07:42:35 by varg