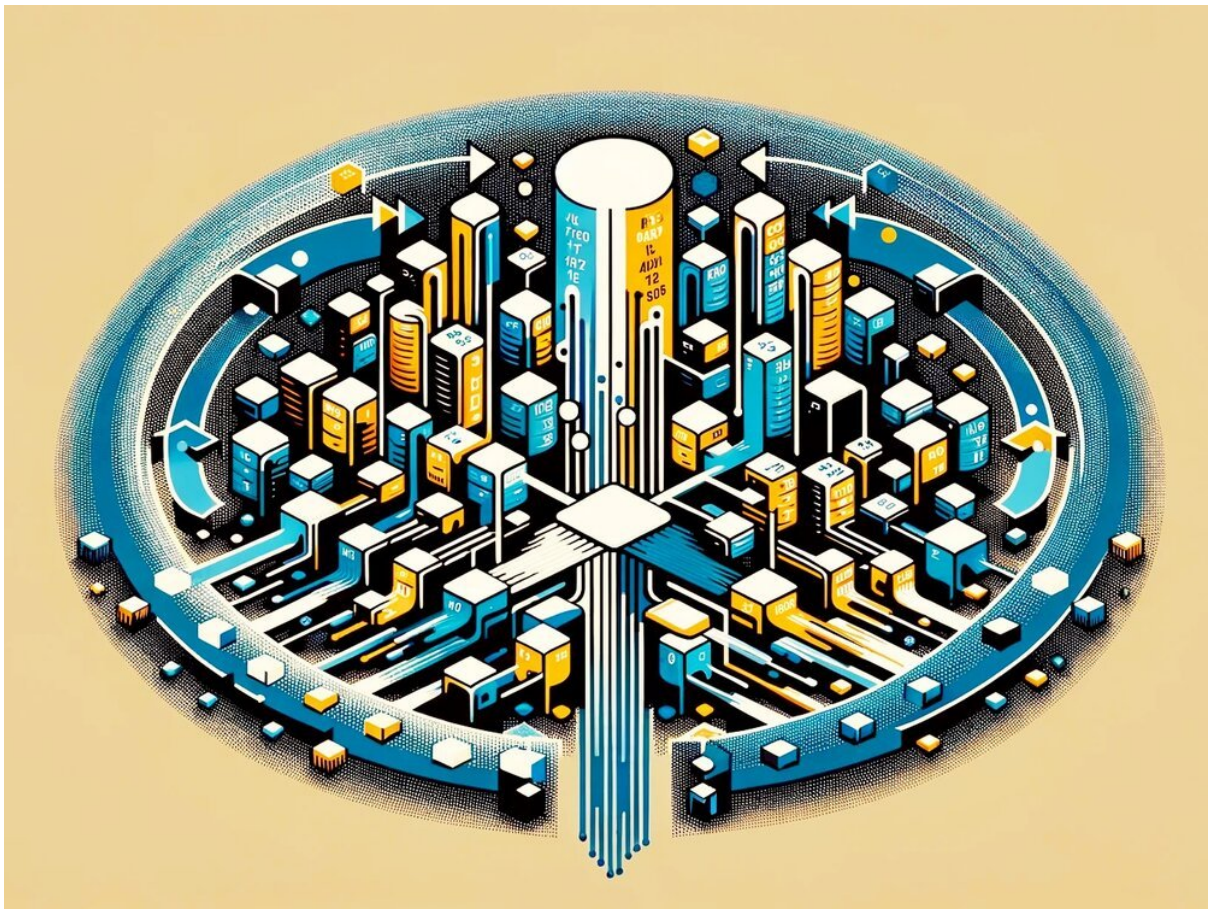
Backup Rotation Scheme

Rotate your backups with 'rsync-backup-rotator'

Willem L. Middelkoop
Dec. 15, 2023

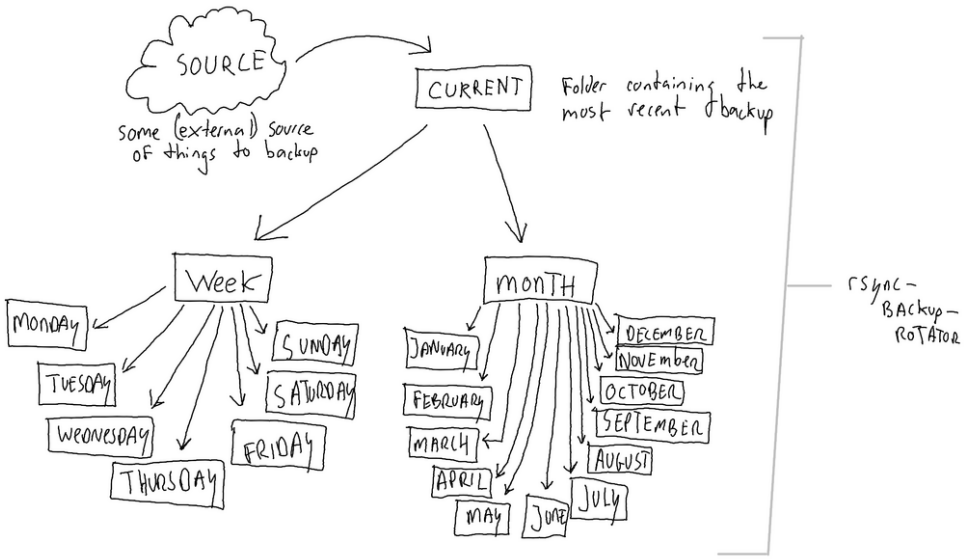


In today's digital age, safeguarding your data is paramount. Simply creating a copy of your files may not be enough as they can get corrupted, overwritten or blocked by ransomware. Having multiple, time-rotated (and ideally, offsite) backups is a stronger defense. I created a new tool, rsync-backup-rotator, to help you with this.

Backup Rotation Scheme

Backup rotation is a strategy used in data management where multiple backup copies are created and stored at different intervals, rather than relying on a single backup copy. This method is particularly useful because it mitigates various risks associated with data loss. For instance, if a single backup copy gets corrupted, overwritten, or compromised (e.g., by ransomware), all data since the last backup could be lost. By rotating backups,

you create multiple recovery points, allowing you to restore data from different moments in time. This approach provides a more comprehensive safety net, as it protects against both recent data loss and more long-term issues.



Rotating backup to enable data recovery from multiple points in time - enabling time traveling for data!

The **'rsync-backup-rotator'** tool embraces this concept of backup rotation. It automates the process of creating and managing these rotated backups. Specifically, the tool uses a central 'current' folder that contains the latest backup data. Based on user-defined settings, it then rotates this data into different folders. For daily backups, it creates and stores copies in subfolders named after each weekday, within a 'week' folder. This means there's a separate backup for each day, like 'Monday', 'Tuesday', and so on. For monthly backups, on the first day of each month, it creates a snapshot in a corresponding monthly subfolder within a 'month' folder, like 'January', 'February', etc. This system ensures that users have a series of time-stamped backups, providing flexibility and security in their data restoration options.

Why rsync?

Rsync is a tool that makes backups both efficient and effective. It works by only updating the parts of files that have changed since the last backup, rather than copying everything over again. This approach significantly reduces the amount of data being transferred, saving on wear and tear for hard drives and SSDs, and reducing costs for cloud storage since fewer data changes mean less I/O operations. Essentially, rsync ensures that backups are quick and light on resources, making it a smart choice for regular data backup and rotation.

Rsync is a standard, non-commercial tool built into many operating systems like GNU/Linux, BSD variants, and macOS, making it easily accessible without extra installations or purchases. Trusted by millions worldwide, its robust and efficient data backup capabilities have established it as a reliable and widely-used industry standard.

Creating Backups with Rsync

To create a single copy using rsync, you can use the command **‘rsync -arvz SOURCE TARGET‘**. In an earlier post, I explained how to use rsync for making backups, which you can find here: How to Use Rsync to Make Backups. It's important to carefully decide what you want to backup, whether it's a simple documents folder or an entire installation. The **‘rsync-backup-rotator‘** tool, as discussed in this blog post, operates under the assumption that you have a functional copy mechanism in place. This mechanism should maintain an up-to-date copy of your files in a folder named ”current”, located in a path accessible by the tool, thereby enabling efficient rotation.

The tool: rsync-backup-rotator

Like my watches, here and here, I prefer my tools to be timeless, self explanatory, simple and dedicated to their job. The **rsync-backup-rotator** was created with this same design ethos.

The tool is a bash script that runs without any weird dependencies or bloat on pretty much all GNU/Linux, BSD and macOS systems. Heck, you can probably get it to work on Windows, too, using the WSL. It is designed to do its job now and in the future, without any maintenance or mandatory updates.

You can download the tool here: `https://source.willem.com/rsync-backup-rotator/` There you'll find instructions and a changelog, too. The tool is distributed as free software under the GPLv3 license, intended to guarantee your freedom to use, share and change all versions of this tool.

Installation

- Download the script from `https://source.willem.com/rsync-backup-rotator/`
- Make the script executable: **‘chmod +x rsync-backup-rotator.sh‘**

Usage

To use rsync-backup-rotator, run the script with the required arguments: **./rsync-backup-rotator.sh** [**options**]

Options:

- -w: Enable backups for each weekday.
- -m: Enable backups for the first day of each month.
- --delete: Include the --delete option in rsync to remove files in the destination that are no longer in the source.
- --help: Display help information.

Examples:

- Perform a weekday backup: **‘./rsync-backup-rotator.sh /path/to/parent -w‘**
- Perform both weekday and monthly backups with deletion of removed files: **‘./rsync-backup-rotator.sh /path/to/parent -w -m --delete‘**

Conclusion

In the future I intend to release more tools and applications under the GPLv3 licence as free software. Although this is a small beginning, it is my humble attempt to give something back to the world. Stay safe, stay backed up!

```bash
#!/bin/bash
#
# rsync-backup-rotator
# Version: 1.0.0
#
# Copyright (C) 2023 Willem L. Middelkoop (mail@willem.com)
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
# For more information, visit: https://source.willem.com/rsync-backup-rotator/

# Function to show help/usage
show_help() {
    echo "Usage: $0 parent_directory [-w] [-m] [--delete] [--help]"
    echo
    echo "Options:"
    echo "  parent_directory   Mandatory. The parent directory for 'current' backups."
    echo "  -w                 Optional. Enable backups for each weekday."
    echo "  -m                 Optional. Enable backups for the first day of each month."
    echo "  --delete           Optional. Include the --delete option in rsync to remove files in t

    echo "  --help             Show this help message and exit."
    echo
    echo "Example:"
    echo "  $0 /path/to/parent -wm --delete"
}

# Function to perform rsync
do_rsync() {
    local source=$1
    local destination=$2
    # Ensure the destination directory exists
    mkdir -p "$destination"
    local rsync_opts="-arvz"
    if [ "$DELETE_ENABLED" = true ]; then
        rsync_opts="$rsync_opts --delete"
    fi
    rsync $rsync_opts "$source" "$destination"
}

# Function to handle daily backups
daily_backup() {
    if [ "$ENABLE_WEEKDAY_BACKUP" = true ]; then
        local weekday=$(date +%A)
        local dest_week="$PARENT_DIR/week/$weekday/"
        do_rsync "$SOURCE_PATH" "$dest_week"
    fi
}

# Function to handle monthly backups
monthly_backup() {
    if [ "$ENABLE_MONTHLY_BACKUP" = true ]; then
        local month=$(date +%B)
        local day=$(date +%d)
        if [ "$day" -eq 1 ]; then
            local dest_month="$PARENT_DIR/month/$month/"
            do_rsync "$SOURCE_PATH" "$dest_month"
        fi
    fi
}

# Check for --help option first
for arg in "$@"; do
    if [ "$arg" = "--help" ]; then
        show_help
        exit 0
    fi
done

# Check if a parent directory is provided
if [ $# -lt 1 ]; then
    echo "Error: No parent directory provided."
    echo
    show_help
    exit 1
fi

# Parsing command-line arguments
PARENT_DIR=$1
shift # Removing the first argument as it's the parent directory
ENABLE_WEEKDAY_BACKUP=false
ENABLE_MONTHLY_BACKUP=false
DELETE_ENABLED=false
SOURCE_PATH="$PARENT_DIR/current"

while (( "$#" )); do
    case $1 in
        -w) ENABLE_WEEKDAY_BACKUP=true;;
        -m) ENABLE_MONTHLY_BACKUP=true;;
        --delete) DELETE_ENABLED=true;;
        *) echo "Invalid option $1" >&2; exit 1;;
    esac
```