Basic information

Currently (11/02/2025) I have:
15+15+15+20+20 = 85GB of total free space.

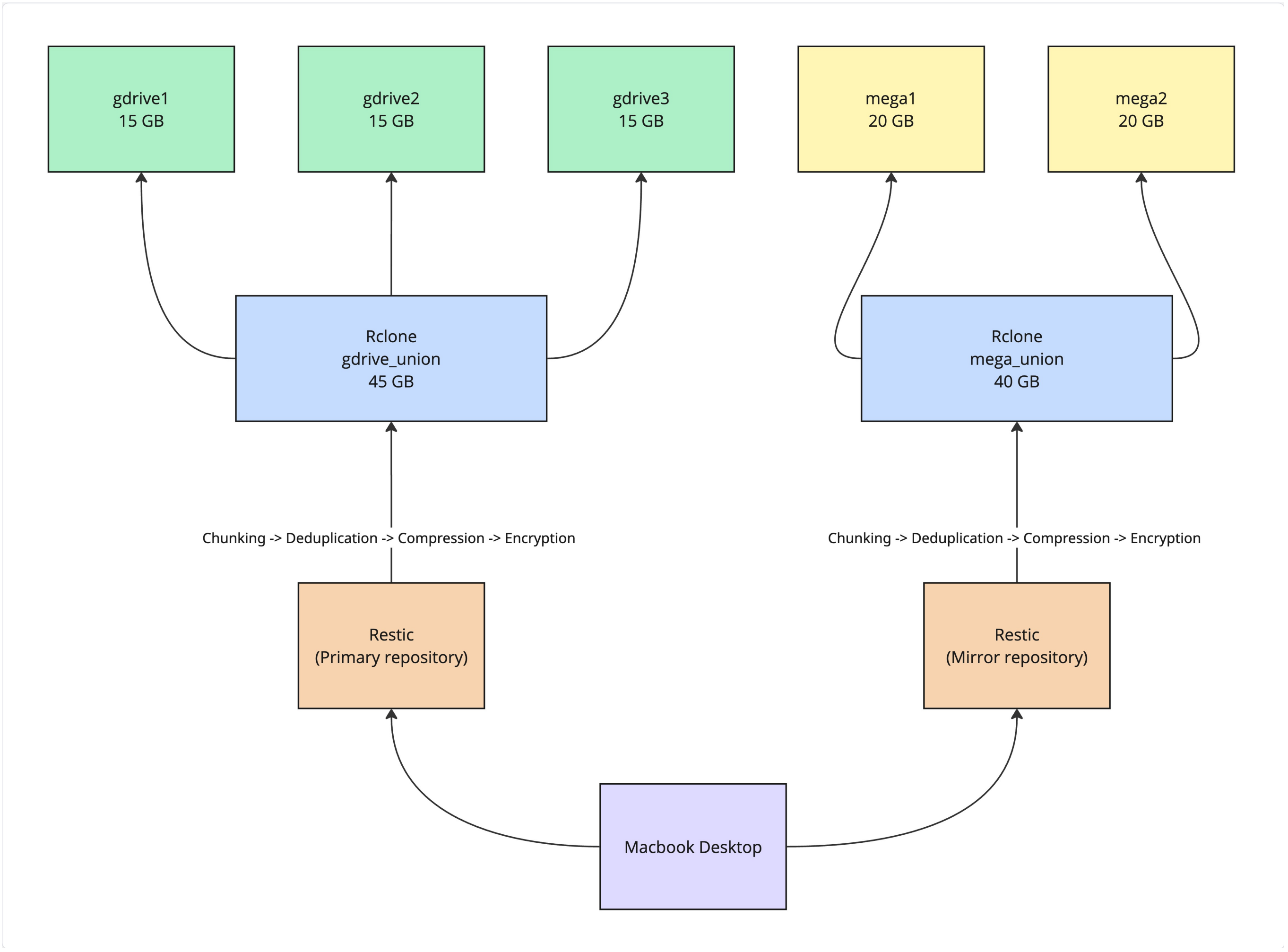
Of which:
15+15+15 = 45GB are on Google Drive
20+20 = 40GB are on MEGA

Google Drive acts as the primary repository.
MEGA acts as the secondary repository and is a mirror of the primary.

Data is sent first from the Mac to the primary repo and then from the Mac to the secondary repo.
Data is deduplicated and encrypted with Restic.
The encryption key is saved in `<your-password-manager>`.

Each repository has 21.451 GiB of occupied space.
In the future I plan to expand the space further and add data from external HDDs, but email management cannot work with too much data to save so I have to find a better solution.

System Diagram



PHASE 1: Account Creation

FIRST ACCOUNT (GOOGLE 15GB):

Name: `<BackupAccount01>`
Date of Birth: `<DD/MM/YYYY>`
Gender: `<Gender>`
Login Email: `<your-backup-email>@passinbox.com`
Password: Saved in a secure note in `<your-password-manager>`

SECOND ACCOUNT (GOOGLE 15GB):

I didn't create a new account. I used one I already had `<your-email-1>@gmail.com` (Password saved in `<your-password-manager>`)

THIRD ACCOUNT (GOOGLE 15GB):

I didn't create a new account. I used one I already had `<your-email-2>@gmail.com`
(Password saved in `<your-password-manager>`)

FOURTH ACCOUNT (MEGA 20GB)

(mega1 on Rclone)

First Name: `<BackupAccount02>`
Last Name: `<BackupAccount02>`
Email: `<your-backup-email>@passinbox.com`
Password: saved in `<your-password-manager>` in a secure note
Recovery Key: saved in `<your-password-manager>` in a secure note

FIFTH ACCOUNT (MEGA 20GB)

(mega2 on Rclone)

First Name: `<BackupAccount02>`
Last Name: `<BackupAccount02>`
Email: `<your-email-2>@gmail.com`
Password: saved in `<your-password-manager>` in a secure note
Recovery Key: saved in `<your-password-manager>` in a secure note

PHASE 2: Linking Accounts to Rclone

Configuration of `<BackupAccount01>` / `<BackupAccount02>` / `<BackupAccount03>` to Rclone.
All the Google account configurations are the same.
MEGA configurations change slightly but are very similar.

Input:

```
<your-username> ~/Desktop $ rclone config
```

Expected output:

```
2025/11/02 13:11:33 NOTICE: Config file "/Users/<your-
username>/.config/rclone/rclone.conf" not found - using defaults
No remotes found, make a new one?
n) New remote
s) Set configuration password
q) Quit config
n/s/q>
```

Input:

```
n
```

Expected output:

```
Enter name for new remote.
name>
```

Input:

```
gdrive1 # or gdrive2 or gdrive3 or mega1 or mega2
```

Expected output:

```
Option Storage.
Type of storage to configure.
Choose a number from below, or type in your own value.
```

Input:

```
22 # Google Drive or 35 for MEGA
```

client_id> and **client_secret>**

Input:

Leave empty, add later if needed. Required for higher rate limits. Takes about 15 minutes to set up.

scope>

Input:

```
1
```

service_account_file> Leave empty, not needed

Expected output:

```
Edit advanced config?
y/n>
```

Input:

```
n
```

Expected output:

```
Use web browser to automatically authenticate rclone with remote?
 * Say Y if the machine running rclone has a web browser you can use
 * Say N if running rclone on a (remote) machine without web browser access
If not sure try Y. If Y failed, try N.

y) Yes (default)
n) No
y/n>
```

Input:

```
y # Browser will open for Google authentication
```

Expected output:

```
Configure this as a Shared Drive (Team Drive)?

y) Yes
n) No (default)
y/n>
```

Input:

```
n
```

Expected output:

```
Configuration complete.
Options:
- type: drive
- scope: drive
- token: {"access_token":"<access-
token>","token_type":"Bearer","refresh_token":"<refresh-
token>","expiry":"2025-11-02T14:25:26.877282+01:00","expires_in":3599}
- team_drive:
Keep this "gdrive1" remote?
y) Yes this is OK (default)
e) Edit this remote
d) Delete this remote
y/e/d>
```

Expected output:

```
Current remotes:

Name                 Type
====                 ====
gdrive1              drive

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>
```

Input:

```
q
```

PHASE 3: Testing Connections

Test First Repository (gdrive1)

Input:

```
rclone lsd gdrive1:
echo "Test rclone" > test.txt
rclone copy test.txt gdrive1:/
rclone ls gdrive1:/
```

Expected output:

```
       12 test.txt
```

Input:

```
rm test.txt
rclone delete gdrive1:/test.txt
```

Test Second Repository (gdrive2)

Input:

```
rclone lsd gdrive2:
echo "Test rclone" > test.txt
rclone copy test.txt gdrive2:/
rclone ls gdrive2:/
```

Expected output:

```
       12 test.txt
```

Input:

```
rm test.txt
rclone delete gdrive2:/test.txt
```

Test Third Repository (gdrive3)

Input:

```
rclone lsd gdrive3:
echo "Test rclone" > test.txt
rclone copy test.txt gdrive3:/
rclone ls gdrive3:/
```

Expected output:

```
       12 test.txt
```

Input:

```
rm test.txt
rclone delete gdrive3:/test.txt
```

Test Fourth Repository (mega1)

Input:

```
rclone lsd mega1:
echo "Test rclone" > test.txt
rclone copy test.txt mega1:/
rclone ls mega1:/
```

Expected output:

```
       12 test.txt
```

Input:

```
rm test.txt
rclone delete mega1:/test.txt
```

Test Fifth Repository (mega2)

Input:

```
rclone lsd mega2:
echo "Test rclone" > test.txt
rclone copy test.txt mega2:/
rclone ls mega2:/
```

Expected output:

```
       12 test.txt
```

Input:

```
rm test.txt
rclone delete mega2:/test.txt
```

PHASE 4: Union of Repositories

Union of Google Repos (Primary)

rclone union "gdrive_union" combines:
├── gdrive1: (15GB)
├── gdrive2: (15GB)
└── gdrive3: (15GB) = A SINGLE 45GB storage for Restic

Input:

```
rclone config create gdrive_union union upstreams "gdrive1: gdrive2:
gdrive3:"
```

Test to Verify Correct Creation:

Input:

```
echo "Test rclone" > test.txt
rclone copy test.txt gdrive_union:/
rclone ls gdrive_union:/
rm test.txt
rclone delete gdrive_union:/test.txt
```

Union of MEGA Repos (Mirror)

Input:

```
rclone config create mega_union union upstreams "mega1: mega2:"
```

Test to Verify Correct Creation:

Input:

```
echo "Test rclone" > test.txt
rclone copy test.txt mega_union:/
rclone ls mega_union:/
rm test.txt
rclone delete mega_union:/test.txt
```

PHASE 5: Initializing Restic Repositories

Generate Password for Restic:

The password is saved in `<your-password-manager>` as `Restic-Rclone` It is 30 characters long. It is used for both repos for simplicity (Primary and Secondary). Saved in `<your-password-manager>` both as login and as secure note.

Initialize Primary Repo

Input:

```
restic -r rclone:gdrive_union:/restic-backup init
```

Expected output:

```
created restic repository <repo-id> at rclone:gdrive_union:/restic-backup
```

Test to Verify Restic Created the Necessary Structure:

Input:

```
rclone lsd gdrive_union:/restic-backup
```

Expected output:

```
-1 2025-11-02 17:45:53        -1 data
-1 2025-11-02 17:45:53        -1 index
-1 2025-11-02 17:45:53        -1 keys
-1 2025-11-02 17:45:53        -1 locks
-1 2025-11-02 17:45:53        -1 snapshots
```

Initialize Mirror Repo

Input:

```
restic -r rclone:mega_union:/restic-backup init
```

Test to Verify Restic Created the Necessary Structure:

Input:

```
rclone lsd mega_union:/restic-backup
```

Expected output:

```
-1 2025-11-02 17:45:53        -1 data
-1 2025-11-02 17:45:53        -1 index
-1 2025-11-02 17:45:53        -1 keys
-1 2025-11-02 17:45:53        -1 locks
-1 2025-11-02 17:45:53        -1 snapshots
```

PHASE 6: Testing with 100MB of Generated Data

Generate a Structure with Synthetic but Realistic Data

Input:

```
mkdir -p ~/Desktop/restic-test/{documents,photos,videos}
```

```
# Create documents
dd if=/dev/urandom of=~/Desktop/restic-test/documents/file1.txt bs=1m
count=1
dd if=/dev/urandom of=~/Desktop/restic-test/documents/file2.txt bs=1m
count=2
dd if=/dev/urandom of=~/Desktop/restic-test/documents/file3.txt bs=1m
count=3

# Create photos
dd if=/dev/urandom of=~/Desktop/restic-test/photos/photo1.jpg bs=1m count=10
dd if=/dev/urandom of=~/Desktop/restic-test/photos/photo2.jpg bs=1m count=10
dd if=/dev/urandom of=~/Desktop/restic-test/photos/photo3.jpg bs=1m count=10

# Create videos
dd if=/dev/urandom of=~/Desktop/restic-test/videos/video1.mp4 bs=1m count=20
dd if=/dev/urandom of=~/Desktop/restic-test/videos/video2.mp4 bs=1m count=20
dd if=/dev/urandom of=~/Desktop/restic-test/videos/video3.mp4 bs=1m count=24

# Check created structure
tree ~/Desktop/restic-test

# Verify it's 100MB
du -sh ~/Desktop/restic-test
```

Send Everything with Restic to Primary Repo

An Ookla speed test shows upload speed is 130 Mbps. Let's see how long it takes to send 100 MB of data

Input:

```
restic -r rclone:gdrive_union:/restic-backup backup ~/Desktop/restic-test
```

Output:

```
processed 9 files, 100.000 MiB in 1:02
```

Send Everything with Restic to Secondary Repo to Test MEGA

Input:

```
restic -r rclone:mega_union:/restic-backup backup ~/Desktop/restic-test
```

Output:

```
processed 9 files, 100.000 MiB in 0:37
```

Try to Recover a File from Primary Repo and a Folder from Secondary Repo

Input:

```
restic -r rclone:gdrive_union:/restic-backup restore latest --target
~/Desktop/restic-restore-test --include /Users/<your-
username>/Desktop/restic-test/photos/photo1.jpg
```

Verify byte by byte:

Input:

```
diff -r ~/Desktop/restic-test/ ~/Desktop/restic-restore-mega/Users/<your-
username>/Desktop/restic-test/
```

Delete Tests from Google

Find the snapshot to delete:

Input:

```
restic -r rclone:gdrive_union:/restic-backup snapshots
```

Delete the snapshot:

Input:

```
restic -r rclone:gdrive_union:/restic-backup forget <snapshot-id> --prune
```

Delete Tests from MEGA:

Find the snapshot to delete:

Input:

```
restic -r rclone:mega_union:/restic-backup snapshots
```

Delete the snapshot:

Input:

```
restic -r rclone:mega_union:/restic-backup forget <snapshot-id> --prune
```

Delete test folders from desktop:

Input:

```
rm -rf ~/Desktop/restic-test
rm -rf ~/Desktop/restic-restore-test
rm -rf ~/Desktop/restic-restore-mega
```

PHASE 7: Sending Complete Backup

Send to Primary Repo

Input:

```
restic -r rclone:gdrive_union:/restic-backup backup ~/Desktop
```

Output:

```
processed 48627 files, 21.451 GiB in 46:52
snapshot <snapshot-id> saved
```

Send to MEGA Mirror

Input:

```
restic -r rclone:mega_union:/restic-backup backup ~/Desktop
```

Output:

```
processed 48627 files, 21.451 GiB in 1:00:01
snapshot <snapshot-id> saved
```

PHASE 8: Verifying Storage Space

Check how much space is left on each account:

Input:

```
rclone about gdrive1:

rclone about gdrive2:

rclone about gdrive3:

rclone about mega1:

rclone about mega2:

rclone about mega_union:

rclone about gdrive_union:
```

PHASE 9: Checking Backup Integrity

Primary repo:
Input:

```
restic -r rclone:gdrive_union:/restic-backup check
```

Secondary repo:
Input:

```
restic -r rclone:mega_union:/restic-backup check
```

```
```

PHASE 10: Backup Automation

Adding Password to macOS Keychain

Input:

```
security add-generic-password \
  -a "$USER" \
  -s "restic-backup" \
  -w
```

Test to see if it saved it. Should print it in clear text:

Input:

```
security find-generic-password -a "$USER" -s "restic-backup" -w
```

Script 1: backup-desktop.sh

Create a script to perform the primary backup:

Input:

```
nvim ~/.local/share/restic-backup/Scripts/backup-desktop.sh
```

Script content:

```
#!/bin/bash

#============================================================================
===
# RESTIC BACKUP AUTOMATION SCRIPT
# Automatic backup of ~/Desktop to Google Drive and MEGA
#============================================================================
===

# CONFIGURATION
BACKUP_PATH="$HOME/Desktop"
```

```
REPO_PRIMARY="rclone:gdrive_union:/restic-backup"
REPO_MIRROR="rclone:mega_union:/restic-backup"
KEYCHAIN_SERVICE="restic-backup"
LOG_DIR="$HOME/.local/share/restic-backup/logs"
DESKTOP="$HOME/Desktop"

# Retention Policy
KEEP_DAILY=7
KEEP_WEEKLY=4
KEEP_MONTHLY=6

# Colors for terminal output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m' # No Color

# Timestamp for log
TIMESTAMP=$(date +"%Y%m%d-%H%M%S")
LOG_FILE="$LOG_DIR/backup-$TIMESTAMP.log"

#============================================================================
===
# UTILITY FUNCTIONS
#============================================================================
===

# Function for logging (print to screen AND save to file)
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Function for macOS notifications
notify() {
    osascript -e "display notification \"$2\" with title \"$1\""
}

# Function to copy log to desktop in case of error
copy_log_to_desktop() {
    cp "$LOG_FILE" "$DESKTOP/backup-error-$(date +%Y%m%d).log"
    log "⚠  Log copied to Desktop for visibility"
}

#============================================================================
===
# BACKUP START
```

```
#============================================================================
===

log "=========================================="
log "STARTING RESTIC BACKUP"
log "=========================================="
log "Path to backup: $BACKUP_PATH"
log "Primary repository: $REPO_PRIMARY"
log "Mirror repository: $REPO_MIRROR"

# Retrieve password from Keychain
log ""
log "Retrieving password from Keychain..."
export RESTIC_PASSWORD=$(security find-generic-password -a "$USER" -s
"$KEYCHAIN_SERVICE" -w 2>&1)

if [ $? -ne 0 ]; then
    log "❌ ERROR: Unable to retrieve password from Keychain"
    notify "Backup Failed" "Error retrieving password from Keychain"
    copy_log_to_desktop
    exit 1
fi

log "✅ Password retrieved successfully"

# Variables to track backup status
GDRIVE_SUCCESS=false
MEGA_SUCCESS=false

#============================================================================
===
# BACKUP TO GOOGLE DRIVE (Primary)
#============================================================================
===

log ""
log "=========================================="
log "BACKUP TO GOOGLE DRIVE (Primary)"
log "=========================================="

START_TIME=$(date +%s)

restic -r "$REPO_PRIMARY" backup "$BACKUP_PATH" \
    --verbose \
    2>&1 | tee -a "$LOG_FILE"
```

```bash
if [ ${PIPESTATUS[0]} -eq 0 ]; then
    END_TIME=$(date +%s)
    ELAPSED=$((END_TIME - START_TIME))
    log "✅ Google Drive backup completed in ${ELAPSED}s"
    GDRIVE_SUCCESS=true

    # Apply retention policy
    log ""
    log "Applying retention policy on Google Drive..."
    log "   - Keep daily: $KEEP_DAILY"
    log "   - Keep weekly: $KEEP_WEEKLY"
    log "   - Keep monthly: $KEEP_MONTHLY"

    restic -r "$REPO_PRIMARY" forget \
        --keep-daily $KEEP_DAILY \
        --keep-weekly $KEEP_WEEKLY \
        --keep-monthly $KEEP_MONTHLY \
        --prune \
        2>&1 | tee -a "$LOG_FILE"

    if [ $? -eq 0 ]; then
        log "✅ Retention policy applied on Google Drive"
    else
        log "⚠ Warning: Issues applying retention policy on Google Drive"
    fi
else
    log "❌ ERROR: Google Drive backup failed"
    GDRIVE_SUCCESS=false
fi

#============================================================================
===
# BACKUP TO MEGA (Mirror)
#============================================================================
===

log ""
log "=========================================="
log "BACKUP TO MEGA (Mirror)"
log "=========================================="

START_TIME=$(date +%s)

restic -r "$REPO_MIRROR" backup "$BACKUP_PATH" \
    --verbose \
    2>&1 | tee -a "$LOG_FILE"
```

```
if [ ${PIPESTATUS[0]} -eq 0 ]; then
    END_TIME=$(date +%s)
    ELAPSED=$((END_TIME - START_TIME))
    log "✅ MEGA backup completed in ${ELAPSED}s"
    MEGA_SUCCESS=true

    # Apply retention policy
    log ""
    log "Applying retention policy on MEGA..."
    log "   - Keep daily: $KEEP_DAILY"
    log "   - Keep weekly: $KEEP_WEEKLY"
    log "   - Keep monthly: $KEEP_MONTHLY"

    restic -r "$REPO_MIRROR" forget \
        --keep-daily $KEEP_DAILY \
        --keep-weekly $KEEP_WEEKLY \
        --keep-monthly $KEEP_MONTHLY \
        --prune \
        2>&1 | tee -a "$LOG_FILE"

    if [ $? -eq 0 ]; then
        log "✅ Retention policy applied on MEGA"
    else
        log "⚠  Warning: Issues applying retention policy on MEGA"
    fi
else
    log "❌ ERROR: MEGA backup failed"
    MEGA_SUCCESS=false
fi

#===========================================================================
===
# FINAL SUMMARY AND NOTIFICATIONS
#===========================================================================
===

log ""
log "=========================================="
log "BACKUP SUMMARY"
log "=========================================="
log "Google Drive (Primary): $([ "$GDRIVE_SUCCESS" = true ] && echo "✅ OK"
|| echo "❌ FAILED")"
log "MEGA (Mirror): $([ "$MEGA_SUCCESS" = true ] && echo "✅ OK" || echo "❌
FAILED")"
log "Log saved in: $LOG_FILE"
log "============================================"

# Handle notifications and exit code based on result
if [ "$GDRIVE_SUCCESS" = true ] && [ "$MEGA_SUCCESS" = true ]; then
    # ✅ COMPLETE SUCCESS
    log "✅ BACKUP COMPLETED SUCCESSFULLY!"
    notify "✅ Backup Completed" "All backups completed successfully"
    exit 0

elif [ "$GDRIVE_SUCCESS" = true ] || [ "$MEGA_SUCCESS" = true ]; then
    # ⚠ PARTIAL SUCCESS
    if [ "$GDRIVE_SUCCESS" = false ]; then
        log "⚠ PARTIAL BACKUP: Google Drive failed, MEGA completed"
        notify "⚠ Partial Backup" "Google Drive failed, MEGA completed
successfully"
    else
        log "⚠ PARTIAL BACKUP: MEGA failed, Google Drive completed"
        notify "⚠ Partial Backup" "MEGA failed, Google Drive completed
successfully"
    fi
    copy_log_to_desktop
    exit 1

else
    # 🔴 TOTAL ERROR
    log "🔴 TOTAL ERROR: Both backups failed"
    notify "🔴 Backup Failed" "ERROR: Both backups failed!"
    copy_log_to_desktop
    exit 2
fi
```

What it does:

- Retrieves password from macOS Keychain
- Backup of `~/Desktop` to Google Drive (primary)
- Backup of `~/Desktop` to MEGA (mirror)
- Automatically applies retention policy (7 daily, 4 weekly, 6 monthly)
- Creates detailed log in `~/.local/share/restic-backup/logs/backup-YYYYMMDD-HHMMSS.log`
- macOS notification at the end (success or error)
- In case of error: copies log to Desktop as `backup-error-YYYYMMDD.log`

Exit codes:

- `0` = Complete success (both backups ok)

- `1` = Partial success (one failed)
- `2` = Total error (both failed)

Script 2: verify-repositories.sh

```bash
#!/bin/bash

#===========================================================================
===
# RESTIC REPOSITORY VERIFICATION SCRIPT
# Verify integrity of Google Drive and MEGA repositories
#===========================================================================
===

# CONFIGURATION
REPO_PRIMARY="rclone:gdrive_union:/restic-backup"
REPO_MIRROR="rclone:mega_union:/restic-backup"
KEYCHAIN_SERVICE="restic-backup"
LOG_DIR="$HOME/.local/share/restic-backup/logs"

# Timestamp for log
TIMESTAMP=$(date +"%Y%m%d")
LOG_FILE="$LOG_DIR/verify-$TIMESTAMP.log"

#===========================================================================
===
# UTILITY FUNCTIONS
#===========================================================================
===

# Function for logging (print to screen AND save to file)
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Function for macOS notifications
notify() {
    osascript -e "display notification \"$2\" with title \"$1\""
}

#===========================================================================
===
# VERIFICATION START
#===========================================================================
===
log "=========================================="
log "STARTING REPOSITORY INTEGRITY VERIFICATION"
log "=========================================="
log "Primary repository: $REPO_PRIMARY"
log "Mirror repository: $REPO_MIRROR"

# Retrieve password from Keychain
log ""
log "Retrieving password from Keychain..."
export RESTIC_PASSWORD=$(security find-generic-password -a "$USER" -s
"$KEYCHAIN_SERVICE" -w 2>&1)

if [ $? -ne 0 ]; then
    log "❌ ERROR: Unable to retrieve password from Keychain"
    notify "Verification Failed" "Error retrieving password from Keychain"
    exit 1
fi

log "✅ Password retrieved successfully"

# Variables to track verification status
GDRIVE_CHECK_OK=false
MEGA_CHECK_OK=false

#============================================================================
===
# VERIFY GOOGLE DRIVE REPOSITORY
#============================================================================
===

log ""
log "=========================================="
log "VERIFYING GOOGLE DRIVE REPOSITORY"
log "=========================================="

START_TIME=$(date +%s)

restic -r "$REPO_PRIMARY" check 2>&1 | tee -a "$LOG_FILE"

if [ ${PIPESTATUS[0]} -eq 0 ]; then
    END_TIME=$(date +%s)
    ELAPSED=$((END_TIME - START_TIME))
    log "✅ Google Drive repository: INTACT (verified in ${ELAPSED}s)"
    GDRIVE_CHECK_OK=true
else
```

```
    log "❌ Google Drive repository: ERRORS DETECTED"
    GDRIVE_CHECK_OK=false
fi

#=============================================================================
===
# VERIFY MEGA REPOSITORY
#=============================================================================
===

log ""
log "=========================================="
log "VERIFYING MEGA REPOSITORY"
log "=========================================="

START_TIME=$(date +%s)

restic -r "$REPO_MIRROR" check 2>&1 | tee -a "$LOG_FILE"

if [ ${PIPESTATUS[0]} -eq 0 ]; then
    END_TIME=$(date +%s)
    ELAPSED=$((END_TIME - START_TIME))
    log "✅ MEGA repository: INTACT (verified in ${ELAPSED}s)"
    MEGA_CHECK_OK=true
else
    log "❌ MEGA repository: ERRORS DETECTED"
    MEGA_CHECK_OK=false
fi

#=============================================================================
===
# FINAL SUMMARY AND NOTIFICATIONS
#=============================================================================
===

log ""
log "=========================================="
log "VERIFICATION SUMMARY"
log "=========================================="
log "Google Drive: $([ "$GDRIVE_CHECK_OK" = true ] && echo "✅ INTACT" ||
echo "❌ ERRORS")"
log "MEGA: $([ "$MEGA_CHECK_OK" = true ] && echo "✅ INTACT" || echo "❌
ERRORS")"
log "Log saved in: $LOG_FILE"
log "=========================================="
```

```
# Final notifications based on result
if [ "$GDRIVE_CHECK_OK" = true ] && [ "$MEGA_CHECK_OK" = true ]; then
    log "✅ VERIFICATION COMPLETED: All repositories are intact"
    notify "✅ Verification Completed" "All repositories are intact"
    exit 0
elif [ "$GDRIVE_CHECK_OK" = true ] || [ "$MEGA_CHECK_OK" = true ]; then
    log "⚠ WARNING: Some repositories have errors"
    notify "⚠ Verification: Partial Errors" "Check log for details"
    exit 1
else
    log "🔴 ERROR: Both repositories have errors"
    notify "🔴 Verification: Critical Errors" "Both repositories have
errors!"
    exit 2
fi
```

What it does:

- Retrieves password from macOS Keychain
- Verifies Google Drive repository integrity (`restic check`)
- Verifies MEGA repository integrity (`restic check`)
- Creates log in `~/.local/share/restic-backup/logs/verify-YYYYMMDD.log`
- macOS notification with result

Running the Scripts

Input:

```
~/.local/share/restic-backup/Scripts/backup-desktop.sh
```

Test Integrity Check Script

Input:

```
~/.local/share/restic-backup/Scripts/verify-repositories.sh
```

Creating Automatic Jobs

Create and save the script:

Input:

```
nvim ~/Library/LaunchAgents/com.user.restic-backup.plist
```

Script content:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <!-- Job identification name -->
    <key>Label</key>
    <string>com.user.restic-backup</string>

    <!-- Command to execute -->
    <key>ProgramArguments</key>
    <array>
        <string>/bin/sh</string>
        <string>-c</string>
        <string>/Users/<your-username>/.local/share/restic-
backup/Scripts/backup-desktop.sh &amp;&amp; /Users/<your-
username>/.local/share/restic-backup/Scripts/verify-repositories.sh</string>
    </array>

    <!-- Environment variables -->
    <key>EnvironmentVariables</key>
    <dict>
        <key>PATH</key>

<string>/opt/homebrew/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin</stri
ng>
    </dict>

    <!-- When to run it: every day at 11:30 -->
    <key>StartCalendarInterval</key>
    <dict>
        <key>Hour</key>
        <integer>11</integer>
        <key>Minute</key>
        <integer>30</integer>
    </dict>

    <!-- Standard output log -->
    <key>StandardOutPath</key>
    <string>/Users/<your-username>/.local/share/restic-backup/logs/launchd-
```

```
stdout.log</string>

    <!-- Error log -->
    <key>StandardErrorPath</key>
    <string>/Users/<your-username>/.local/share/restic-backup/logs/launchd-
stderr.log</string>

    <!-- Also run at boot if it was scheduled time -->
    <key>RunAtLoad</key>
    <true/>
</dict>
</plist>
```

Load the job and check that it was loaded.

Input:

```
launchctl load ~/Library/LaunchAgents/com.user.restic-backup.plist
launchctl list | grep restic
```

Now the script will run automatically at 11:30 every day. At the end, a notification will be generated in the notification center at the top right.