Rclone Backup to Box for Cluster

Markus G. S. Weiss

2025/05/05

Contents

1 Introduction

This tutorial explains how to configure `rclone` on your cluster to back up `/mfs/io/groups/sterling/mf` to a Box directory named `cluster-backup`, with subfolders for `daily`, `archive`, and `logs`, and how to schedule it via `cron`. Users in the sterling group only need to run the commands in Sections 3–5 and 7. The scripts are maintained centrally under `/mfs/io/groups/sterling/setup`.

2 Prerequisites

- `rclone` (v1.38 or later) installed on both the cluster and your desktop (with a browser).
- Confirm `rclone` versions match:

```
/mfs/io/groups/sterling/software-tools/rclone/rclone-v1
   .69.1-linux-amd64/rclone version
rclone version
```

- A Box Enterprise SSO account.
- Shell access to the cluster with `cron` available.

Tip

Before running any live syncs, you can test with `-dry-run` to see what would transfer or delete without affecting Box:

```
/mfs/io/groups/sterling/software-tools/rclone/rclone-v1
   .69.1-linux-amd64/rclone sync \
  /mfs/io/groups/sterling/mfshome/$USER box:cluster-
     backup/daily \
  --dry-run --fast-list --checksum
```

3 Configure the Box remote with offline authorization

On the cluster, run `rclone` using its full path:

```
/mfs/io/groups/sterling/software-tools/rclone/rclone-v1.69.1-
   linux-amd64/rclone config
```

Press Enter to accept each default (shown in `<angle brackets>`):

```
No remotes found, make a new one?
n/s/q> n

name> box
Storage> box
client_id> <leave blank>
client_secret> <leave blank>
box_config_file> <leave blank>
access_token> <leave blank>

box_sub_type>
  1 / user
  2 / enterprise
box_sub_type> 2

Edit advanced config?
y/n> n

Use web browser to automatically authenticate?
y/n> n
```

1. **Copy** the printed `rclone authorize` command, switch to your local machine, paste and run it. Complete the OAuth flow in your browser to obtain a long token string.

2. Back on the cluster, paste only that token at:

   ```
   config_token> xxxxxxxxxxxxxxxx...xxx
   ```

3. When asked, keep the remote:

   ```
   Keep this "box" remote?
   y) y
   ```

4. Verify:

```
/mfs/io/groups/sterling/software-tools/rclone/rclone-v1
   .69.1-linux-amd64/rclone lsd box:
```

4 Create the Box folder hierarchy

Run once on the cluster using the full `rclone` path:

```
# Parent backup folder
/mfs/io/groups/sterling/software-tools/rclone/rclone-v1.69.1-
   linux-amd64/rclone mkdir box:cluster-backup

# Subfolders
/mfs/io/groups/sterling/software-tools/rclone/rclone-v1.69.1-
   linux-amd64/rclone mkdir box:cluster-backup/daily
/mfs/io/groups/sterling/software-tools/rclone/rclone-v1.69.1-
   linux-amd64/rclone mkdir box:cluster-backup/archive
/mfs/io/groups/sterling/software-tools/rclone/rclone-v1.69.1-
   linux-amd64/rclone mkdir box:cluster-backup/logs
```

Verify:

```
/mfs/io/groups/sterling/software-tools/rclone/rclone-v1.69.1-
   linux-amd64/rclone lsd box:cluster-backup
```

5 Prepare the local environment

On the cluster, create a directory for logs:

```
mkdir -p ~/logs
```

6 Reference scripts

Sterling group members *do not* need to modify these; they live in `/mfs/io/groups/sterling/setup`.

6.1 A) backup.sh

```
#!/usr/bin/env bash

#
   ------------------------------------------------------------------------

# Script: backup.sh
# Description:
#   Backs up local data to remote storage via rclone.
#   - Daily incremental backups
#   - Weekly snapshots (Sundays)
#   - Prunes local logs older than 30 days
#   - Prunes remote snapshots older than 28 days
#   - Uploads logs to remote
# Usage:
#   backup.sh  (override settings via environment variables as
    needed)
#
# Configuration (env overrides):
#   DATA_DIR           Local directory to back up (default: /
   mfs/.../$USER)
#   REMOTE_ROOT        Remote root for backups (default: box:
   cluster-backup)
#   RCLONE_BIN         Path to rclone binary (default: rclone-
   v1.69.1)
#   LOG_DIR            Directory for local logs (default:
   $HOME/logs)
#
# Author: Markus G. S. Weiss
# Date:   2025-05-05
#
   ------------------------------------------------------------------------

set -euo pipefail

# --- Configuration (override via env if desired)
   ------------------------------
: "${DATA_DIR:=/mfs/io/groups/sterling/mfshome/$USER}"
: "${REMOTE_ROOT:=box:cluster-backup}"
```

```
: "${RCLONE_BIN:=/mfs/io/groups/sterling/software-tools/rclone
   /rclone-v1.69.1-linux-amd64/rclone}"
: "${LOG_DIR:=$HOME/logs}"

DATE_STR=$(date +%F)
LOCK_FILE="$HOME/.backup_${USER}.lock"

# Common rclone options
RCLONE_OPTS="--dry-run --fast-list --checksum --log-level
   WARNING"

# Retry settings
MAX_RETRIES=3
RETRY_DELAY=10

# Snapshot retention (days)
REMOTE_RETENTION_DAYS=28

# --- Setup
   ------------------------------------------------------------------

# Ensure log directory exists
mkdir -p "$LOG_DIR"

# Prevent overlapping runs
exec 200>"$LOCK_FILE"
flock -n 200 || {
  echo "[$(date '+%F %T')] Another backup is already running.
     Exiting." >> "$LOG_DIR/backup-$DATE_STR.log"
  exit 1
}

# --- Utility: retry wrapper
   ---------------------------------------------------
retry() {
  local n=1 cmd="$*"
  until eval "$cmd"; do
    if (( n >= MAX_RETRIES )); then
      echo "[$(date '+%F %T')] ERROR: Command failed after
         $MAX_RETRIES attempts: $cmd" >> "$LOG_DIR/backup-
         $DATE_STR.log"
```

```
      return 1
    fi
    echo "[$(date '+%F %T')] WARN: Command failed (attempt $n/
       $MAX_RETRIES). Retrying in $RETRY_DELAY s..." >> "
       $LOG_DIR/backup-$DATE_STR.log"
    sleep $RETRY_DELAY
    ((n++))
  done
}

# --- 1) Prune local logs older than N days
  -----------------------------------
prune_local_logs() {
  local retention_days=30 logf="$LOG_DIR/backup-$DATE_STR.log"
  echo "[$(date '+%F %T')] Pruning local logs older than
     $retention_days days..." >> "$logf"
  find "$LOG_DIR" -type f -name '*.log' -mtime +
     $retention_days -delete
  echo "[$(date '+%F %T')] Pruning local logs completed." >> "
     $logf"
}

# --- 2) Prune old remote snapshots
  -------------------------------------------
prune_remote_snapshots() {
  local logf="$LOG_DIR/backup-$DATE_STR.log"
  echo "[$(date '+%F %T')] Pruning remote snapshots older than
      $REMOTE_RETENTION_DAYS days..." >> "$logf"
  retry "$RCLONE_BIN delete '$REMOTE_ROOT/archive' --min-age $
     {REMOTE_RETENTION_DAYS}d $RCLONE_OPTS" >> "$logf"
  echo "[$(date '+%F %T')] Pruned remote snapshots." >> "$logf
     "
}

# --- 3) Daily incremental backup
  -----------------------------------------------
backup_daily() {
  local src="$DATA_DIR" dest="$REMOTE_ROOT/daily" logf="
     $LOG_DIR/backup-$DATE_STR.log"
  echo "[$(date '+%F %T')] Starting daily backup from $src to
     $dest..." >> "$logf"
```

```
  retry "$RCLONE_BIN sync '$src' '$dest' $RCLONE_OPTS --log-
     file '$logf'"
  echo "[$(date '+%F %T')] Daily backup completed." >> "$logf"
}

# --- 4) Weekly snapshot (Sundays)
   ------------------------------------------
snapshot_weekly() {
  if [[ "$(date +%u)" == "7" ]]; then
    local src="$DATA_DIR" dest="$REMOTE_ROOT/archive/$DATE_STR
       " logf="$LOG_DIR/snapshot-$DATE_STR.log"
    echo "[$(date '+%F %T')] Starting weekly snapshot from
       $src to $dest..." >> "$logf"
    retry "$RCLONE_BIN copy '$src' '$dest' $RCLONE_OPTS --log-
       file '$logf'"
    echo "[$(date '+%F %T')] Weekly snapshot completed." >> "
       $logf"
  fi
}

# --- 5) Upload logs
   ------------------------------------------------------------
upload_logs() {
  local src="$LOG_DIR" dest="$REMOTE_ROOT/logs" logf="$LOG_DIR
     /backup-$DATE_STR.log"
  echo "[$(date '+%F %T')] Uploading logs from $src to $dest
     ..." >> "$logf"
  retry "$RCLONE_BIN sync '$src' '$dest' $RCLONE_OPTS --log-
     file '$logf'"
  echo "[$(date '+%F %T')] Log upload completed." >> "$logf"
}

# --- Main
   ------------------------------------------------------------------------


main() {
  prune_local_logs
  prune_remote_snapshots
  backup_daily
  snapshot_weekly
  upload_logs
```

```
  echo "[$(date '+%F %T')] Script finished successfully." >> "
     $LOG_DIR/backup-$DATE_STR.log"
}

main "$@"

# ---Log Rotation (optional)
   ------------------------------------------------
# For home-directory logs, add ~/.config/logrotate/backup:
# $HOME/logs/*.log {
#   daily
#   rotate 30
#   compress
#   missingok
#   notifempty
#   copytruncate
# }
```

Make it executable:

```
chmod +x /mfs/io/groups/sterling/setup/backup.sh
```

6.2 B) cronscript

```
#
   ------------------------------------------------------------------------

# Crontab: sterling's backup jobs
# Description:
#   Runs the master backup.sh every day, with all pruning and
   log-uploads
#   handled internally in that script.
#
# Author: Markus G. S. Weiss
# Date:   2025-05-05
#
   ------------------------------------------------------------------------


SHELL=/bin/bash
```

```
PATH=/usr/local/bin:/usr/bin:/bin
MAILTO=$USER@utdallas.edu

# Run backup.sh daily at 02:00
0 2 * * * /mfs/io/groups/sterling/setup/backup.sh
```

7 Install the cron job

On the cluster, install the pre-written cron script:

```
crontab /mfs/io/groups/sterling/setup/cronscript
```

Verify:

```
crontab -l
```

8 Monitoring & Maintenance

- **View logs (live tail):**

  ```
  tail -f ~/logs/backup-$(date +%F).log
  ```

- **Clean up local logs older than 30 days:**

  ```
  find ~/logs -type f -mtime +30 -delete
  ```

- **Test restores:**

  ```
  /mfs/io/groups/sterling/software-tools/rclone/rclone-v1
     .69.1-linux-amd64/rclone copy \
    box:cluster-backup/daily/path/to/file /tmp && \
    diff /tmp/file /mfs/io/groups/sterling/mfshome/$USER/
       path/to/file
  ```

- **Error notifications:** Cron will email stderr/stdout to `$USER@yourdomain.com`. For advanced alerting, grep logs for `ERROR` and pipe to mail or integrate with Slack.

9 Additional Notes

- **Security & permissions:** Do *not* check `~/.config/rclone/rclone.conf` into any shared repositories—it contains tokens.
- **Data encryption:** Consider using an rclone `crypt` wrapper for encryption at rest.
- **API rate limits:** Box enforces API quotas. Tweak `-transfers`, `-checkers`, or add `-tpslimit 3` if you hit rate-limit errors.
- **Network/firewall:** Ensure outbound HTTPS (port 443) is open. If behind a proxy, set `HTTPS_PROXY` or use `-proxy`.
- **Monthly or quarterly snapshots:** Extend the weekly logic with checks like:

```
if [[ "$(date +%d)" == "01" ]]; then
  ... # monthly archive
fi
```

- **Upstream docs:** Official rclone Box backend documentation: https://rclone.org/box/

10 Summary

In this tutorial, you have:

- **Configured** the Box remote on a headless cluster node via offline authorization.
- **Created** a clear Box folder hierarchy (`cluster-backup/{daily,archive,logs}`) for organized storage.
- **Prepared** a local log directory and referenced centrally maintained backup and cron scripts.
- **Written** a robust `backup.sh` that performs daily incremental syncs, weekly snapshots, and log uploads.
- **Scheduled** the backup using a `crontab`, including log rotation and snapshot cleanup.
- **Implemented** monitoring, restore procedures, and maintenance routines (log pruning, error alerts).

- **Added** best-practice notes on dry-runs, version checks, security, API-rate limits, and firewall considerations.

Great work! Your cluster's home directory is now automatically and safely backed up to Box every night, with versioning, logs, and the tools for easy maintenance and recovery.