Automation

Automation script and tips

- **How to replace Cron by SystemD**
- **□□ BookStack Backup Guide (Using systemd, with Auto-Cleanup)**

How to replace Cron by SystemD

Use `systemd` to schedule automated tasks

? Overview

On modern Linux systems, **systemd timers** can fully replace cron jobs. Timers give you:

- Better logging (`journalctl`)
- Service dependencies (`After=`, `Requires=`)
- Reliable scheduling after boot
- Random delays to avoid load spikes
- Clear control (`start`, `stop`, `enable`, `status`)

This guide explains how to create a scheduled task using systemd.

---

1. Create a systemd Service

The service defines **what** will run — usually a script or command.

Example file:
`/etc/systemd/system/myscript.service`

```
[Unit]
Description=Run my custom script

[Service]
Type=oneshot
ExecStart=/usr/local/bin/myscript.sh
```

Make your script executable:

```
chmod +x /usr/local/bin/myscript.sh
```

---

2. Create a systemd Timer

The timer defines **when** the service runs.

File:
`/etc/systemd/system/myscript.timer`

```
[Unit]
Description=Run my script every day at 03:00

[Timer]
OnCalendar=03:00
Persistent=true

[Install]
WantedBy=timers.target
```

`Persistent=true` ensures missed runs (e.g., machine off) are executed at the next boot.

---

3. Enable and Start the Timer

Reload systemd and activate the timer:

```
systemctl daemon-reload
systemctl enable --now myscript.timer
```

---

4. Verify the Timer is Working

List active timers:

```
systemctl list-timers
```

View logs for the executed service:

```
journalctl -u myscript.service
```

---

5. Useful Scheduling Examples

Run Daily at 3 AM

```
OnCalendar=03:00
```

Run Every 15 Minutes

```
OnCalendar=*:0/15
```

Run Every Monday at 09:00

```
OnCalendar=Mon 09:00
```

Run on Reboot (after 5 minutes)

```
OnBootSec=5min
```

Run Hourly

```
OnCalendar=hourly
```

Run Monthly

```
OnCalendar=monthly
```

6. Example: Run a Script Every 30 Minutes

`/etc/systemd/system/healthcheck.service`

```
[Unit]
Description=Health check script

[Service]
Type=oneshot
ExecStart=/usr/local/bin/healthcheck.sh
```

`/etc/systemd/system/healthcheck.timer`

```
[Unit]
Description=Run health check every 30 minutes

[Timer]
OnCalendar=*:0/30

[Install]
WantedBy=timers.target
```

Enable:

```
systemctl enable --now healthcheck.timer
```

---

7. Troubleshooting

Check Timer Status

```
systemctl status myscript.timer
```

Check Service Logs

```
journalctl -u myscript.service
```

Manually Trigger the Task

```
systemctl start myscript.service
```

? Summary

systemd timers are a robust and modern alternative to classic cron:

- More control
- Better logging
- Cleaner dependency management
- Easier to maintain

? BookStack Backup Guide (Using systemd, with Auto-Cleanup)

This document explains how to back up a BookStack instance using a **backup script**, a **systemd service**, and a **systemd timer**, plus an **automatic cleanup timer** that deletes old backups.

It's work only for bookstack in docker if you have a bare metal install you nee to modify the script

No cron is used — fully systemd-native.

? What Gets Backed Up?

A full BookStack backup requires:

1. **Database dump** (MySQL/MariaDB) → contains all pages, books, users, settings
2. **Application files** → BookStack codebase
3. **Upload files** → images + attachments
4. **Storage files** → internal uploads, custom icons, etc.

? 1. Install the Backup Script

Create the script at:

```
/usr/local/bin/bookstack-backup.sh
```

Script content

> Update `DB_NAME`, `DB_USER`, `DB_PASS`, `BACKUP_DIR`, and `BOOKSTACK_DIR` to match your setup.

```
#!/usr/bin/env bash
set -euo pipefail

# === CONFIG ===
BACKUP_DIR="/backup/bookstack"

# Docker container + DB creds
DB_CONTAINER="bookstack-db"
DB_NAME="bookstack"
DB_USER="bookstack"
DB_PASS="YOUR_DB_PASSWORD"

# Host paths for bind mounts (adjust to yours)
APP_DIR="/opt/bookstack/app"            # optional if you want the app code
UPLOADS_DIR="/opt/bookstack/uploads"
STORAGE_DIR="/opt/bookstack/storage"

DATE=$(date +"%Y-%m-%d_%H-%M-%S")
TARGET_DIR="${BACKUP_DIR}/${DATE}"

mkdir -p "$TARGET_DIR"

echo "[*] $(date) - Dumping database from container ${DB_CONTAINER}..."
docker exec "${DB_CONTAINER}" \
  sh -c "mysqldump -u\"${DB_USER}\" -p\"${DB_PASS}\" \"${DB_NAME}\"" \
  > "${TARGET_DIR}/db.sql"

echo "[*] $(date) - Copying BookStack upload/storage data..."

# Only copy app dir if it exists (in case you don't map it)
if [ -d "$APP_DIR" ]; then
  rsync -a "$APP_DIR" "${TARGET_DIR}/app"
fi

rsync -a "$UPLOADS_DIR" "${TARGET_DIR}/uploads"
rsync -a "$STORAGE_DIR" "${TARGET_DIR}/storage"

echo "[*] $(date) - Creating archive..."
cd "$BACKUP_DIR"
tar -czf "bookstack_${DATE}.tar.gz" "$DATE"
```

```
echo "[*] $(date) - Cleaning up temp folder..."
rm -rf "$TARGET_DIR"

echo "[+] $(date) - Backup complete: ${BACKUP_DIR}/bookstack_${DATE}.tar.gz"
```

Make it executable:

```
chmod +x /usr/local/bin/bookstack-backup.sh
```

?? 2. systemd Service (Backup Runner)

Create:

```
/etc/systemd/system/bookstack-backup.service
```

With:

```
[Unit]
Description=BookStack backup
Wants=network-online.target
After=network-online.target mariadb.service mysql.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/bookstack-backup.sh
User=root
Group=root

# Optional: keep system responsive
Nice=10
IOSchedulingClass=best-effort
IOSchedulingPriority=7

[Install]
WantedBy=multi-user.target
```

Test it manually:

```
systemctl daemon-reload
systemctl start bookstack-backup.service
journalctl -u bookstack-backup.service -e
```

? 3. systemd Timer (Daily Backup at 02:00)

Create:

```
/etc/systemd/system/bookstack-backup.timer
```

With:

```
[Unit]
Description=Daily BookStack backup at 02:00

[Timer]
OnCalendar=*-*-* 02:00:00
Persistent=true
Unit=bookstack-backup.service

[Install]
WantedBy=timers.target
```

Enable the timer:

```
systemctl daemon-reload
systemctl enable --now bookstack-backup.timer
```

Check timers:

```
systemctl list-timers | grep bookstack
```

? 4. Auto-Delete Old Backups (Cleanup Timer)

This step adds:

- a **cleanup script** that deletes backup files older than **30 days**
- a **systemd service** to run the script
- a **systemd timer** to schedule cleanup (default: daily at 03:00)

You can adjust the retention days and schedule as needed.

4.1 Cleanup Script

Create:

```
/usr/local/bin/bookstack-backup-cleanup.sh
```

With:

```
#!/usr/bin/env bash
set -euo pipefail

# === CONFIG ===
BACKUP_DIR="/backup/bookstack"
RETENTION_DAYS=30

echo "[*] $(date) - Cleaning up BookStack backups older than ${RETENTION_DAYS} days in
${BACKUP_DIR}..."

if [ ! -d "$BACKUP_DIR" ]; then
  echo "[!] Backup directory ${BACKUP_DIR} does not exist, nothing to clean."
  exit 0
fi

# Delete .tar.gz archives older than RETENTION_DAYS
find "$BACKUP_DIR" -maxdepth 1 -type f -name "bookstack_*.tar.gz" -mtime +$RETENTION_DAYS -
print -delete

# Optionally clean leftover dated directories (should normally not exist)
find "$BACKUP_DIR" -maxdepth 1 -type d -regex '.*/[0-9-]\{10\}_[0-9:-]\{8\}' -mtime
+$RETENTION_DAYS -print -exec rm -rf {} +

echo "[+] $(date) - Cleanup complete."
```

Make it executable:

```
chmod +x /usr/local/bin/bookstack-backup-cleanup.sh
```

> To change retention, edit `RETENTION_DAYS=30` (for example, to 7 for one week).

4.2 Cleanup systemd Service

Create:

```
/etc/systemd/system/bookstack-backup-cleanup.service
```

With:

```
[Unit]
Description=Cleanup old BookStack backups

[Service]
Type=oneshot
ExecStart=/usr/local/bin/bookstack-backup-cleanup.sh
User=root
Group=root

Nice=10
IOSchedulingClass=best-effort
IOSchedulingPriority=7

[Install]
WantedBy=multi-user.target
```

Test it:

```
systemctl daemon-reload
systemctl start bookstack-backup-cleanup.service
journalctl -u bookstack-backup-cleanup.service -e
```

4.3 Cleanup systemd Timer (Daily at 03:00)

Create:

```
/etc/systemd/system/bookstack-backup-cleanup.timer
```

With:

```
[Unit]
Description=Daily cleanup of old BookStack backups at 03:00

[Timer]
OnCalendar=*-*-* 03:00:00
Persistent=true
Unit=bookstack-backup-cleanup.service

[Install]
WantedBy=timers.target
```

Enable the timer:

```
systemctl daemon-reload
systemctl enable --now bookstack-backup-cleanup.timer
```

Check:

```
systemctl list-timers | grep bookstack-backup-cleanup
```

? Restore Procedure (Dockerized BookStack)

From a backup file like:

```
/backup/bookstack/bookstack_YYYY-MM-DD_HH-MM-SS.tar.gz
```

0. Stop the stack (recommended)

From the directory where your docker-compose.yml lives:

```
docker compose down
# or
docker-compose down
```

1. Extract the backup archive

```
cd /backup/bookstack
tar -xzf bookstack_YYYY-MM-DD_HH-MM-SS.tar.gz

# This creates a directory like:
# /backup/bookstack/YYYY-MM-DD_HH-MM-SS
cd YYYY-MM-DD_HH-MM-SS
ls

# You should see:
# db.sql
# uploads/
# storage/
# app/        (only if you backed app dir too)
```

2. Restore BookStack files

Adjust paths if yours are different.

```
# Restore uploads
rsync -a uploads/ /opt/bookstack/uploads/

# Restore storage
rsync -a storage/ /opt/bookstack/storage/

# Restore app code (only if you backed it and actually use APP_DIR)
if [ -d app ]; then
  rsync -a app/ /opt/bookstack/app/
fi
```

If your containers expect certain ownership/permissions, fix them now. For many setups the web user inside the container is `www-data` (uid 33), but often simple `chmod` is enough since Docker abstracts it.

Example (optional):

```
chown -R root:root /opt/bookstack
# or: chown -R 1000:1000 /opt/bookstack
```

3. Start the stack again

```
cd /path/to/your/docker-compose/
docker compose up -d
# or
docker-compose up -d
```

Wait a few seconds for the DB container (`bookstack-db`) to be healthy.

4. Restore the database inside the DB container

Set your DB variables (must match what you used in backups / docker-compose):

```
DB_CONTAINER="bookstack-db"
DB_NAME="bookstack"
DB_USER="bookstack"
DB_PASS="YOUR_DB_PASSWORD"
```

Then run:

```
cd /backup/bookstack/YYYY-MM-DD_HH-MM-SS

docker exec -e MYSQL_PWD="$DB_PASS" -i "$DB_CONTAINER" \
  mysql -u"$DB_USER" "$DB_NAME" < db.sql
```

- `-i` pipes db.sql into the container
- `MYSQL_PWD` avoids putting the password directly on the mysql command line
- if your container uses .env vars like `MYSQL_USER`, `MYSQL_PASSWORD`, `MYSQL_DATABASE`, you can map those instead of hard-coding.

5. Verify

- Open BookStack in your browser
- Check that:
  - Pages & books exist
  - Attachments and images load

- Users and settings look correct

? Recommended Extras

- Sync `/backup/bookstack` to offsite storage (S3, Minio, Backblaze, rsync to another host)
- Keep at least 7–30 daily backups depending on your risk tolerance
- If using ZFS/Btrfs:
  - Combine filesystem snapshots with this logical backup
  - Optionally snapshot the backup dataset itself

? Summary

Component	Backed Up / Managed By
Database	`mysqldump` in `bookstack-backup.sh`
App Files	`rsync` in `bookstack-backup.sh`
Uploads & Storage	Included in app directory
Backup Automation	`bookstack-backup.service` + `.timer`
Cleanup Automation	`bookstack-backup-cleanup.service` + `.timer`
Retention	`RETENTION_DAYS` in cleanup script

This setup is:

- Fully **systemd-native** (no cron)
- Includes **daily backup** + **daily cleanup**
- Easy to tweak for different times / retention periods.