

Phase 2 Setup Guide: Backup Automation Migration

Overview

Phase 2 migrates your backup script from cron to n8n (running every 3 days) with **significant improvements** to prevent data corruption and service interruptions.

Phase 2 Goals

1. **Migrate backup automation** from cron to n8n
2. **Implement hot backup strategy** (no container stopping!)
3. **Add backup verification** to detect failures early
4. **Implement retention policy** (keep last 5 backups from 2 weeks)
5. **Enable comprehensive monitoring** with Telegram + Uptime Kuma
6. **Prevent Plex database corruption** through SQLite-aware backups

Key Improvements Over Old Backup Script

Feature	Old Script (Cron)	New Script (n8n)
Schedule	 Weekly (Sundays)	 Every 3 days
Services	 6 services	 11 services (9 local + 2 remote)
Container Status	 Stops ALL containers	 Hot backup (keeps running)
Plex Database	 Risked corruption	 SQLite-aware backup
Service Downtime	 ~5-10 minutes weekly	 Zero downtime
Verification	 None	 Size checks after each backup
Retention	 Manual cleanup	 Auto-keep last 5 backups (2 weeks)
Notifications	 Uptime Kuma only	 Telegram + Uptime Kuma
Error Details	 Minimal	 Full logs in Telegram
Logging	 Basic	 Timestamped, detailed

Why Hot Backups Are Better

The Problem with the Old Approach

The original `backup_media_stack.sh` script used this dangerous pattern:

```
# DON'T DO THIS!
docker compose down      # Stop containers
tar -czf backup.tar.gz ... # Backup files
docker compose up -d       # Restart containers
```

This caused issues:

1. **Plex Database Corruption:** Stopping Plex while it's writing to the SQLite database can corrupt it
2. **Service Interruption:** ~5-10 minutes of downtime every Sunday at midnight
3. **Lost Metadata:** If backup fails, containers stay down until manual intervention
4. **User Experience:** Family/friends can't use Plex during backup window

The Hot Backup Solution

The new script backs up files **while containers are running**:

For Plex (Critical - SQLite Database)

```
# Use SQLite's built-in backup command
sqlite3 plex_database.db ".backup backup_file.db"
```

This uses SQLite's official backup API that:

- Handles Write-Ahead Logging (WAL) properly
- Creates consistent snapshots even during writes
- Prevents corruption by using proper locking

For Other Services (Sonarr, Radarr, etc.)

```
# Use tar with excludes for cache/logs
tar -czf backup.tar.gz --exclude='*/logs/*' /path/to/config
```

These services store data in ways that are safe to copy while running:

- Configuration files (rarely change during backup)
- Small SQLite databases (less critical than Plex)
- No risk of corruption from hot backup

Workflow Architecture: Why Parallel Execution Matters

CRITICAL: Understanding the Notification Bug Fix

 **THIS IS NOT OPTIONAL - Understanding this prevents incorrect notifications!**

Both the Container Health Check (Phase 1) and Backup Automation (Phase 2) workflows use a **parallel execution pattern** that is critical to prevent a serious bug.

The Bug (Sequential Execution)

If Uptime Kuma notifications run in the main sequential flow, they **interfere with conditional logic**:

X BROKEN FLOW:
 Execute Script → Parse Output → **Uptime** Kuma → Check Status → Telegram
RESULT: Both success AND failure Telegram messages are sent!

What Happens:

- Uptime Kuma runs and passes data forward
- The conditional check receives data that has been “tainted” by the Uptime Kuma node
- Telegram notifications fire REGARDLESS of actual status
- You get false alerts constantly

The Fix (Parallel Execution)

By running Uptime Kuma in a **separate parallel branch**, it operates independently:

✓ CORRECT FLOW:
 Execute → Parse ———
 ↗ **Uptime** Kuma Push (always runs, isolated)
 ↗ Check Status → Telegram (conditional only)

What Happens:

- Parse Output sends data to TWO independent branches
- **Branch 1:** Uptime Kuma always runs, updates monitoring status
- **Branch 2:** Conditional check only sends Telegram if needed
- No interference between branches - each operates independently

Why This Pattern Is Mandatory

Without parallel execution:

- **X** False Telegram alerts (both success AND failure)
- **X** Notification spam
- **X** Loss of trust in monitoring system
- **X** Alert fatigue

With parallel execution:

- **✓** Uptime Kuma always gets status updates
- **✓** Telegram only notifies when needed
- **✓** Clean separation of concerns
- **✓** Reliable monitoring

How to Verify in n8n

When you import the workflow, look at the connections in the n8n editor:

1. Find the “Parse Backup Output” node
2. You should see TWO arrows coming out:
 - One to “Uptime Kuma Push”
 - One to “Check Backup Status”
3. These run in parallel, not sequentially

If you see them running sequentially (one after another), the workflow is broken!

Key Takeaway

This parallel execution pattern MUST be used in all future workflows that combine status monitoring (Uptime Kuma) with conditional notifications (Telegram). It's a critical architectural pattern, not just a performance optimization.

✓ Prerequisites

Before starting Phase 2, ensure Phase 1 is complete:

- [] n8n is running at `http://192.168.1.142:5678`
- [] PostgreSQL backend is healthy
- [] Git repository initialized at `/home/ubuntu/homelab-automation/`
- [] Telegram bot configured and working
- [] Container health monitoring workflow is active
- [] SSH credentials configured in n8n

Verify Phase 1:

```
cd /home/ubuntu/homelab-automation
git status # Should show clean working tree
docker ps | grep n8n # Should show n8n running
curl http://192.168.1.142:5678 # Should respond
```

🚀 Installation Steps

Step 1: Copy Improved Backup Script to Server

The improved script is already in your Git repository. Now copy it to the server location:

```
# SSH into ollivanders.home (if not already there)
ssh user@192.168.1.142

# Create scripts directory if it doesn't exist
sudo mkdir -p /home/ubuntu/homelab-automation/scripts

# The script should already be at:
ls -lh /home/ubuntu/homelab-automation/scripts/backup_media_stack_improved.sh

# Verify it's executable
chmod +x /home/ubuntu/homelab-automation/scripts/backup_media_stack_improved.sh
```

Step 2: Install SQLite3 (Required for Plex Hot Backup)

```
# On RHEL 10.1
sudo dnf install sqlite -y

# Verify installation
sqlite3 --version
# Expected output: 3.x.x or higher
```

Why SQLite3? The script uses `sqlite3` command to safely backup Plex's database while it's running. Without this, the script falls back to rsync (less safe).

Step 3: Test Backup Script Manually

IMPORTANT: Test the script manually before automating it!

```
# Run the improved backup script
sudo /home/ubuntu/homelab-automation/scripts/backup_media_stack_improved.sh

# Expected output:
# [TIMESTAMP] Starting Media Stack Hot Backup
# [TIMESTAMP] Checking service status (containers will remain running)...
# [TIMESTAMP] Backing up Plex database (hot backup)...
# [TIMESTAMP] ✓ Plex database hot backup successful
# [TIMESTAMP] Backing up Plex...
# ... (more backup operations)
# [TIMESTAMP] ✓ All backups completed successfully!
```

Check the log file:

```
cat /mnt/server/logs/backup_improved.log
```

Verify backups were created:

```
ls -lh /mnt/server/backup/media-stack/
# Should see files like:
# plex_2025-12-24.tar.gz
# sonarr_2025-12-24.tar.gz
# radarr_2025-12-24.tar.gz
# overseerr_2025-12-24.tar.gz
# tautulli_2025-12-24.tar.gz
# heimdall_2025-12-24.tar.gz
# scrypted_2025-12-24.tar.gz
# uptime-kuma_2025-12-24.tar.gz
# frigate_2025-12-24.tar.gz
# sabnzbd_2025-12-24.tar.gz
# nginx-proxy-manager_2025-12-24.tar.gz
```

Check backup sizes are reasonable:

```
du -sh /mnt/server/backup/media-stack/*
# Plex should be largest (10GB+ typical)
# Others typically 50MB-500MB each
```

Step 4: Import n8n Workflow

1. **Open n8n:** Navigate to `http://192.168.1.142:5678`
2. **Login** with your credentials
3. **Click “Add Workflow”** (+ icon in top right)
4. **Click the “ : ” menu** → Select “Import from File”
5. **Upload:** `/home/ubuntu/homelab-automation/workflows/backup-automation.json`
6. **Rename** the workflow if desired (default: “Media Stack Backup Automation”)

Step 5: Configure Workflow Settings

The workflow needs a few configurations:

A. Configure SSH Credentials

The workflow references SSH credentials as `"SSH ollivanders.home"`.

1. In n8n, go to **“Credentials”** (left sidebar)
2. Click **“+ Add Credential”**
3. Select **“SSH”**
4. Fill in:
 - **Name:** `SSH ollivanders.home`
 - **Host:** `192.168.1.142`
 - **Port:** `22`
 - **Username:** Your SSH username (e.g., `user`)
 - **Password:** Your SSH password
5. Click **“Save”**

B. Configure Telegram Variable (Optional)

If you want to use Telegram notifications:

1. In the workflow editor, go to **“Variables”** tab
2. Add variable:
 - **Name:** `TELEGRAM_CHAT_ID`
 - **Value:** Your Telegram chat ID (from Phase 1)
3. **Alternative:** Edit the Telegram nodes directly and hardcode your chat ID

C. Update Uptime Kuma Push URL (If Needed)

The workflow uses this Uptime Kuma push URL:

```
http://192.168.1.142:3001/api/push/6egmEoFola
```

If your URL is different:

1. Check your Uptime Kuma monitor for the correct push URL
2. Edit each “Uptime Kuma” node in the workflow
3. Update the `url` parameter

Step 6: Test the Workflow

Before activating it on a schedule, test it manually:

1. **In n8n workflow editor**, click **“Execute Workflow”** button (top right)
2. **Watch the execution:**
 - “Weekly Backup Schedule” → skipped (manual execution)
 - “Uptime Kuma - Backup Start” → should succeed
 - “Execute Backup Script” → should run script via SSH
 - “Check Backup Result” → should route based on exit code
 - “Uptime Kuma - Success/Failure” → should notify
 - “Parse Backup Output” → should extract stats
 - “Telegram - Success/Failure” → should send message
3. **Check Telegram:** You should receive a detailed backup notification

4. Check Uptime Kuma: `http://192.168.1.142:3001` should show backup status updated

If execution fails:

- Check the error message in n8n
 - Verify SSH credentials are correct
 - Verify backup script path is correct
 - Check SSH connectivity: `ssh user@192.168.1.142 'ls -la /home/ubuntu/homelab-automation/scripts/'`
-

Migration: Disable Old Cron Job

Once you've verified the n8n workflow works, disable the old cron job to prevent conflicts:

```
# SSH into ollivanders.home
ssh user@192.168.1.142

# Edit crontab
crontab -e

# Find this line:
# 0 0 * * 0 /bin/bash -lc '/mnt/server/tools/backup_and_restore/media_stack/
backup_media_stack.sh ...'

# Comment it out by adding # at the beginning:
# 0 0 * * 0 /bin/bash -lc '/mnt/server/tools/backup_and_restore/media_stack/
backup_media_stack.sh ...'

# Save and exit (Ctrl+O, Enter, Ctrl+X in nano)
```

Verify cron job is disabled:

```
crontab -l | grep backup_media_stack
# Should show the line commented out with #
```

Activation: Enable n8n Workflow

1. In n8n, open the “Media Stack Backup Automation” workflow
2. Toggle the “Active” switch in the top right (should turn green)
3. Verify schedule: Click on “Every 3 Days Backup Schedule” node
 - Should show: `0 0 */3 * *` (Every 3 days at midnight)
 - Timezone: Should match your server (America/New_York)

The workflow is now active! It will run automatically every 3 days at midnight.

Monitoring and Validation

Check Workflow Execution History

1. In n8n, go to “Executions” (left sidebar)

2. Find “Media Stack Backup Automation” executions
3. Click on any execution to see:
 - Which nodes ran successfully (green)
 - Which nodes failed (red)
 - Output data from each node
 - Duration of execution

Check Backup Files

Every few days, verify backups were created:

```
# Check latest backups
ls -lt /mnt/server/backup/media-stack/ | head -10

# Should see files with dates from every 3 days, like:
# plex_2025-12-24.tar.gz
# plex_2025-12-21.tar.gz
# plex_2025-12-18.tar.gz
# sonarr_2025-12-24.tar.gz
# ...
```

Check Logs

```
# View latest backup log
tail -100 /mnt/server/logs/backup_improved.log

# Look for:
# - "✓ All backups completed successfully!"
# - Exit code: 0
# - Backup summary statistics
```

Verify Retention Policy

After 2 weeks, verify old backups are being deleted:

```
# Count backups per service
ls /mnt/server/backup/media-stack/plex_*.tar.gz | wc -l
# Should show maximum of 5 files (2 weeks worth at every-3-days frequency)

# Verify all 11 services have backups
ls /mnt/server/backup/media-stack/ | grep -E "plex|sonarr|radarr|overseerr|tautulli|
heimdall|scrypted|uptime-kuma|frigate|sabnzbd|nginx-proxy-manager"
```

SSH Access Requirements

Important: The backup script requires passwordless SSH access to remote hosts:

- **Sabnzbd:** dainja@192.168.4.99
- **Nginx Proxy Manager:** dainja@192.168.1.236

Set up SSH keys if not already configured:

```
# From ollivanders.home (192.168.1.142), set up SSH keys
ssh-copy-id dainja@192.168.4.99
ssh-copy-id dainja@192.168.1.236

# Test connections
ssh dainja@192.168.4.99 'echo "Sabnzbd SSH works"'
ssh dainja@192.168.1.236 'echo "NPM SSH works"'
```

Troubleshooting

Issue: Backup Script Fails with “Permission Denied”

Symptom: Workflow execution shows “Permission denied” error

Solution:

```
# Ensure script is executable
chmod +x /home/ubuntu/homelab-automation/scripts/backup_media_stack_improved.sh

# Ensure backup directory has proper permissions
sudo mkdir -p /mnt/server/backup/media-stack
sudo chown -R $USER:$USER /mnt/server/backup/media-stack
```

Issue: SQLite3 Not Found

Symptom: Log shows “⚠ SQLite3 not found, using rsync fallback”

Solution:

```
# Install SQLite3
sudo dnf install sqlite -y

# Verify installation
sqlite3 --version
```

Issue: Remote Sabnzbd Backup Fails

Symptom: Error like “ssh: connect to host 192.168.4.99 port 22: Connection refused”

Solution:

```
# Test SSH connection manually
ssh dainja@192.168.4.99 'echo "SSH works"'

# If that fails, check:
# 1. SSH service is running on remote host
# 2. Firewall allows SSH (port 22)
# 3. SSH keys are set up (or password authentication enabled)

# Add SSH key for passwordless authentication (recommended)
ssh-copy-id dainja@192.168.4.99
```

Issue: Backup Size is Suspiciously Small

Symptom: Backup file is < 100KB or much smaller than expected

Solution:

```
# Check the log for verification errors
grep "backup too small" /mnt/server/logs/backup_improved.log

# Manually check directory sizes
du -sh /mnt/server/plex
du -sh /mnt/server/sonarr
# etc.

# The backup should be roughly the same size as the source directories
```

Issue: Telegram Notifications Not Working

Symptom: Workflow succeeds but no Telegram message received

Solution:

1. Verify Telegram bot token is correct in n8n credentials
2. Verify chat ID is correct (check `$vars.TELEGRAM_CHAT_ID`)
3. Test Telegram node separately in n8n
4. Check if bot is blocked or removed from chat

Issue: Uptime Kuma Not Updating

Symptom: Uptime Kuma status doesn't change after backup runs

Solution:

1. Check Uptime Kuma is running: `docker ps | grep uptime-kuma`
2. Verify push URL is correct: `http://192.168.1.142:3001/api/push/6egmEoFola`
3. Test the URL manually:

```
bash
curl "http://192.168.1.142:3001/api/push/6egmEoFola?status=up&msg=Test"
```



Backup Restoration Procedure

If you ever need to restore from a backup:

Restore Plex

```
# 1. Stop Plex container
cd /mnt/server/plex
docker compose down

# 2. Backup current config (just in case)
mv config config.backup.$(date +%Y%m%d)

# 3. Extract backup
cd /mnt/server/backup/media-stack
tar -xzf plex_2025-12-22.tar.gz -C /

# 4. Restart Plex
cd /mnt/server/plex
docker compose up -d

# 5. Verify Plex is working
docker logs plex --tail 100
```

Restore Other Services (Sonarr, Radarr, etc.)

```
# Same process as Plex, just adjust paths:
cd /mnt/server/sonarr
docker compose down
# ... extract backup ...
docker compose up -d
```

Restore Remote Sabnzbd

```
# SSH into remote host
ssh dainja@192.168.4.99

# Stop Sabnzbd
cd /home/dainja/sabnzbd
docker compose down

# Copy backup from main server
scp user@192.168.1.142:/mnt/server/backup/media-stack/sabnzbd_2025-12-22.tar.gz /tmp/

# Extract backup
tar -xzf /tmp/sabnzbd_2025-12-22.tar.gz -C /

# Restart Sabnzbd
docker compose up -d
```

Restore Remote Nginx Proxy Manager

```
# SSH into remote host
ssh dainja@192.168.1.236

# Stop Nginx Proxy Manager
cd /opt/npm
docker compose down

# Copy backup from main server
scp user@192.168.1.142:/mnt/server/backup/media-stack/nginx-proxy-manager_2025-12-22.tar.gz /tmp/

# Extract backup
tar -xzf /tmp/nginx-proxy-manager_2025-12-22.tar.gz -C /

# Restart Nginx Proxy Manager
docker compose up -d
```

Restore New Local Services (Heimdall, Scrypted, Uptime Kuma, Frigate)

```
# Same process as other local services, just adjust paths:
# Example for Heimdall:
cd /mnt/server/tools/heimdall
docker compose down
tar -xzf /mnt/server/backup/media-stack/heimdall_2025-12-22.tar.gz -C /
docker compose up -d

# Repeat for scrypted, uptime-kuma, and frigate
```

Best Practices for Restoration

1. **Always test backups** periodically by restoring to a test environment
2. **Keep multiple backup copies** (our retention policy keeps 2 weeks / 5 backups)
3. **Document your restoration steps** specific to your setup
4. **Take a snapshot before restoring** if using VM/LVM snapshots
5. **Verify SSH access** to remote hosts before attempting remote restores



Next Steps

Congratulations! Phase 2 is now complete. Your backups run every 3 days with hot backup strategies.

Phase 3 Preview: Docker Update Automation

The next phase will migrate `docker_weekly_pull.sh` to n8n:

- Automated image updates every Sunday at 2 AM
- Update verification before applying
- Rollback capability if updates fail
- Detailed change notifications
- Downtime minimization strategies

Phase 4 Preview: Advanced Monitoring

Future enhancements:

- Disk space monitoring and alerts
 - Performance metrics dashboards
 - Predictive failure detection
 - Automated remediation for common issues
-



Additional Resources

- [n8n Documentation](https://docs.n8n.io/) (<https://docs.n8n.io/>)
 - [SQLite Backup Documentation](https://www.sqlite.org/backup.html) (<https://www.sqlite.org/backup.html>)
 - [Plex Database Management](https://support.plex.tv/articles/201539237-backing-up-plex-media-server-data/) (<https://support.plex.tv/articles/201539237-backing-up-plex-media-server-data/>)
 - [Uptime Kuma API Documentation](https://github.com/louislam/uptime-kuma/wiki/API) (<https://github.com/louislam/uptime-kuma/wiki/API>)
-



Getting Help

If you encounter issues:

1. **Check logs:** `/mnt/server/logs/backup_improved.log`
 2. **Check n8n execution history:** Shows detailed error messages
 3. **Review this guide:** Many issues are covered in Troubleshooting
 4. **Check GitHub Issues:** [homelab-automation repository]
-

Phase 2 Setup Complete! 🎉

Your backup system is now:

- Automated via n8n
- Using hot backup strategies
- Monitored with Telegram + Uptime Kuma
- Verified and retained properly
- Preventing Plex database corruption