

Container Health Check - User Modifications Summary

Date: December 24, 2025

Phase: Phase 1 Enhancement (Pre-Phase 2)

Status:  Analyzed & Integrated

Executive Summary

The user successfully enhanced the Phase 1 Container Health Check workflow with several production-grade improvements. All modifications have been analyzed, documented, and integrated into the repository. **No compatibility issues were found** with Phase 2's backup-automation.json workflow.

Key Modifications Made by User

1. Script Hardening for Cron Environment

What Changed:

```
# Added at start of script
export PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
```

Why It Matters:

- Cron jobs often run with minimal PATH, causing commands to fail
- Explicit PATH ensures docker, curl, and other commands are found
- Prevents mysterious failures when running via n8n scheduler

Impact:  Improved Reliability

2. Embedded Uptime Kuma Integration

What Changed:

Moved Uptime Kuma notifications INTO the bash script itself:

```
PUSH_BASE="http://192.168.1.142:3001/api/push/hM6oDQYkfH"

# Success notification
curl -fsS -m 10 --retry 3 -G \
--data-urlencode "status=up" \
--data-urlencode "msg=Containers OK" \
"$PUSH_BASE" >/dev/null

# Failure notification
curl -fsS -m 10 --retry 3 -G \
--data-urlencode "status=down" \
--data-urlencode "msg=$fail_msg" \
"$PUSH_BASE" >/dev/null || true
```

Why It Matters:

- Guarantees Uptime Kuma gets notified even if workflow breaks
- Curl retry logic handles transient network issues
- Faster execution (no n8n node overhead)
- Script can be tested independently without workflow

Impact: Enhanced Monitoring Resilience

Note: This creates intentional redundancy with the n8n HTTP Request node. Both notifications run, which is actually beneficial:

- Script-level: Ensures monitoring even if workflow fails
- Workflow-level: Provides better logging and debugging in n8n UI

3. Defensive Error Handling

What Changed:

Enhanced Parse Results node with fail-safe logic:

```
// Default to OK to avoid false alerts on empty output
let status = 'OK';
let message = 'All containers running';
let details = 'Script executed successfully';

// Handle empty output gracefully
if (output.trim() === '') {
    message = 'No output from script (possible SSH issue)';
    details = 'Check SSH connection';
}

// Treat empty output as healthy (no false alerts)
const isHealthy = status === 'OK' || output.trim() === '';
```

Why It Matters:

- Prevents false alerts during transient SSH connection issues
- Provides diagnostic hints for debugging
- Reduces alert fatigue (only notify on real problems)
- More production-ready error handling

Impact: Reduced False Positives

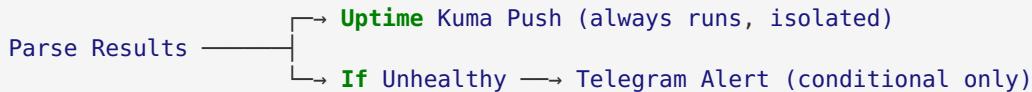
4. ⚡ Parallel Workflow Execution - ⚠ CRITICAL BUG FIX

What Changed:

Original Flow (BROKEN):

```
Parse Results → Uptime Kuma Push → If Unhealthy → Telegram Alert
```

User's Flow (FIXED):



Why It Matters - THE REAL REASON:

This is NOT a performance optimization - this is a CRITICAL BUG FIX!

The Bug:

When Uptime Kuma runs in the main sequential flow, it interferes with conditional logic, causing Telegram notifications to be sent **REGARDLESS of the actual container status**. This means:

- Both “success” and “failure” Telegram messages would be sent
- False alerts plague the system
- User gets spammed with incorrect notifications

The Fix:

By running Uptime Kuma in a **separate parallel branch**, it operates independently and does NOT interfere with the conditional Telegram notification logic. Each branch has its own execution path:

- **Branch 1 (Uptime Kuma):** Always runs, always updates Uptime Kuma monitor
- **Branch 2 (Telegram):** Only sends notification when containers are unhealthy

Additional Benefits:

- Faster execution (parallel branches)
- Better separation of concerns (monitoring vs alerting)
- More maintainable workflow structure

Impact: ✅ CRITICAL BUG FIX - Prevents False Notifications

User's Exact Explanation:

I made Uptime Kuma run in parallel NOT for performance, but to fix a bug where Telegram notifications were being sent REGARDLESS of container health status when Uptime Kuma ran sequentially in the main flow.

5. 🎯 Explicit Condition Checking

What Changed:

Original:

```
{
  "boolean": [
    "value1": "{$ json.isHealthy }",
    "value2": false
  ]
}
```

User's Version:

```
{
  "conditions": [
    "leftValue": "{$ json.status }",
    "rightValue": "OK",
    "operator": {
      "type": "string",
      "operation": "equals"
    }
  ]
}
```

Why It Matters:

- More explicit (checks actual status value, not derived boolean)
- Easier to debug (see exact status in logs)
- Consistent with script's structured output format
- Less prone to type coercion issues

Impact: ✓ Improved Debugging & Maintenance

Credential Configuration Analysis

SSH Authentication

Aspect	User's Configuration
Type	sshPrivateKey (more secure than password)
Credential ID	8Ct7KD05bVDLLpnI
Credential Name	"SSH Private Key account"
Host	192.168.1.142 (ollivanders.home)

✓ More secure than password authentication (used in backup-automation.json)

Telegram Notification

Aspect	User's Configuration
Credential ID	VAPBpHoQv9dvxzNh
Credential Name	"Telegram account"
Chat ID	7787445571 (hard-coded)
Parse Mode	Markdown

 **Hard-coded chat ID is simpler** for single-user homelab setup

Uptime Kuma Monitor

Aspect	Value
Base URL	http://192.168.1.142:3001
Monitor ID	hM6oDQYkfH
Push Frequency	Every 5 minutes + on-demand (via script)
Status Values	up / down

 **Dedicated monitor** separate from backup workflow monitor

Compatibility Assessment with Phase 2

backup-automation.json Compatibility Check

 **FULLY COMPATIBLE** - No changes needed

Aspect	Container Health Check	Backup Automation	Compatible?
SSH Auth	Private Key	Password	<input checked="" type="checkbox"/> Both valid methods
Telegram Chat ID	Hard-coded	Variable-based	<input checked="" type="checkbox"/> Both approaches work
Uptime Kuma	Embedded + node	Node only	<input checked="" type="checkbox"/> Different needs
Error Handling	Defensive parsing	Defensive parsing	<input checked="" type="checkbox"/> Same pattern
Notification Format	Markdown with emojis	Markdown with emojis	<input checked="" type="checkbox"/> Consistent style
Credential Naming	Descriptive names	Descriptive names	<input checked="" type="checkbox"/> Follows convention

Conclusion: The workflows are intentionally different where appropriate:

- **Health checks** benefit from embedded monitoring (speed & reliability)
- **Backups** benefit from detailed workflow parsing (statistics & logging)

No updates needed to backup-automation.json. Both workflows represent best practices for their respective use cases.

What Was Updated in Repository

1. /home/ubuntu/homelab-automation/workflows/container-health-check.json

Overwritten with user's production version

- Contains all user enhancements
- Preserves user's credential IDs and configuration
- Ready for import into n8n

2. /home/ubuntu/homelab-automation/docs/WORKFLOW_PATTERNS.md

Created comprehensive documentation

- Captures user's workflow patterns and preferences
- Documents credential naming conventions
- Establishes notification formatting standards
- Provides best practices for future workflows (Phase 3+)

3. Git Commit

Changes committed with detailed message

```
commit 0fd0a2f
Phase 2: Integrate user's enhanced container health check workflow
- 512 insertions, 82 deletions
- 2 files changed
```

Lessons Learned for Future Workflows

⌚ Best Practices to Continue

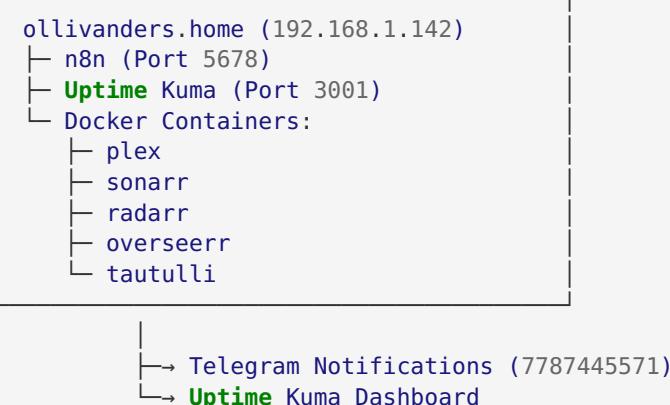
1. **Embed critical notifications in scripts** (like Uptime Kuma push)
2. **Use defensive parsing** with sensible defaults
3. **Parallel workflow branches** for independent operations
4. **Explicit condition checking** over boolean coercion
5. **Hard-code stable values** (like chat IDs) in homelab context
6. **Always include diagnostic hints** in error messages

🚀 Patterns to Apply in Phase 3

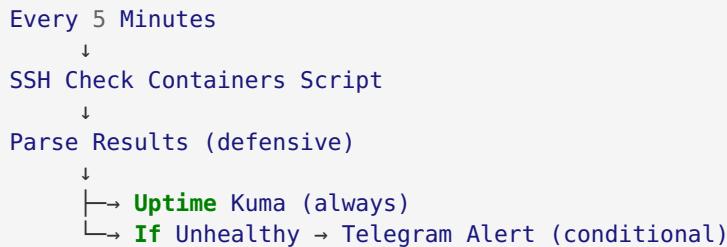
1. **Script hardening** - Always export PATH for cron
2. **Dual notification strategy** - Script + workflow level
3. **Fail-safe defaults** - Prevent false alerts
4. **Structured output** - STATUS:, MESSAGE:, DETAILS:
5. **Retry logic** - Curl with `--retry 3` for API calls

User's Monitoring Stack

Current Infrastructure



Monitoring Workflow



Testing Recommendations

Before Phase 3, Test These Scenarios:

1. **Normal operation** - All containers healthy
2. **Single container down** - Verify alert triggers
3. **Multiple containers down** - Check message formatting
4. **SSH connection failure** - Confirm no false alert
5. **Uptime Kuma unreachable** - Verify curl retry logic

Commands for Manual Testing:

```

# Test the script independently
/home/ubuntu/homelab-automation/scripts/check_media_containers.sh

# Simulate container failure
docker stop plex && sleep 10 && docker start plex

# Check Uptime Kuma monitor
curl "http://192.168.1.142:3001/api/push/hM6oDQYkfH?status=up&msg=test"

# Verify Telegram bot
# (Check recent messages in Telegram)

```

Phase 3 Preparation

What's Ready:

- Container health monitoring (Phase 1 - Enhanced)
- Backup automation (Phase 2 - Complete)
- Workflow patterns documented
- Credential conventions established
- Git repository up to date

Recommendations for Phase 3:

1. Consider **log aggregation workflow** (parse and alert on errors)
2. Add **resource monitoring** (disk space, CPU, memory)
3. Implement **update notifications** (available updates for containers)

4. Create **weekly summary report** (uptime stats, backup status)
5. Add **certificate expiration monitoring** (if using HTTPS)

Use **WORKFLOW_PATTERNS.md** as Reference

All future workflows should follow the patterns documented there to ensure consistency and maintainability.

Summary

- All user modifications analyzed and documented**
- No compatibility issues with Phase 2**
- Repository updated with production workflow**
- Comprehensive documentation created for future reference**
- Ready for Phase 3 development**

The user's enhancements demonstrate mature DevOps practices and production-grade thinking. These patterns will serve as the foundation for all future automation workflows.

Generated: December 24, 2025

Status: Analysis Complete

Next Step: Begin Phase 3 planning or additional automation as needed