n8n Workflow Patterns & Best Practices

This document captures the workflow patterns, conventions, and best practices learned from analyzing the user's production workflows. Use this as a reference when creating new workflows to ensure consistency.

Table of Contents

Overview

The user has successfully deployed and customized two main workflows:
- **Container Health Check** (Phase 1) - Every 5 minutes
- **Media Stack Backup Automation** (Phase 2) - Every 3 days

Both workflows demonstrate mature patterns for monitoring, alerting, and integration with external services.

⚠️ CRITICAL: Parallel Execution Architecture

> **🚨 MANDATORY PATTERN FOR ALL WORKFLOWS**
>
> **The Problem:** When Uptime Kuma notifications run in the main sequential flow, they interfere with conditional logic, causing incorrect Telegram notifications (both success AND failure messages sent regardless of actual status).
>
> **The Solution:** Uptime Kuma MUST run in a separate parallel branch, isolated from conditional Telegram notifications.
>
> **Pattern:**
> ```
> Parse Output → ┬─ Uptime Kuma (parallel, always runs)
>                └─ Check Status → Conditional Telegram (success OR failure)
> ```
>
> **This is NOT optional** - all workflows must follow this pattern to prevent false alerts!

User Modifications Analysis

Container Health Check Workflow - Key Improvements

The user made significant enhancements to the original Phase 1 workflow:

1. Script Hardening for Cron

```
# Added explicit PATH export for cron reliability
export PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
```

Rationale: Prevents issues when running via cron where PATH may be limited.

2. Embedded Uptime Kuma Integration

The user integrated Uptime Kuma push notifications directly into the bash script:

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

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

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

Benefits:
- Faster execution (no n8n node overhead)
- More resilient (curl with retries)
- Guaranteed notification even if workflow fails
- Script can be run independently for testing

Note: This creates redundancy with the n8n HTTP node, but provides better reliability.

3. Enhanced Error Handling in Parse Results

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

// Treat empty output as healthy to avoid false alerts
const isHealthy = status === 'OK' || output.trim() === '';
```

Key Insights:
- Empty SSH output shouldn't trigger alerts
- Provides helpful diagnostic messages
- Prevents alert fatigue from transient network issues

4. Parallel Execution Pattern - CRITICAL BUG FIX ⚠️

> ⚠️ **WARNING:** This is NOT just a performance optimization - this is a **CRITICAL BUG FIX** that prevents incorrect notifications!

The Bug (Sequential Execution):
When Uptime Kuma runs in the main sequential flow, it can interfere with conditional logic, causing Telegram notifications to be sent **REGARDLESS** of the actual status. This leads to false alerts.

Sequential (BROKEN):

```
Parse → Uptime Kuma → Check Status → Telegram Success AND Telegram Failure (BOTH FIRE!)
```

The Fix (Parallel Execution):

```
Parse → ┬─ Uptime Kuma (always runs, doesn't interfere)
        └─ Check Status → Telegram Success OR Telegram Failure (conditional)
```

Why This Works:
- Uptime Kuma runs in its own parallel branch (isolated from conditional logic)
- Telegram notifications have their own conditional branch with proper if/else logic
- Each branch operates independently - no interference

Implementation in n8n JSON:

```
"Parse Results": {
  "main": [
    [
      {
        "node": "Uptime Kuma Push",
        "type": "main",
        "index": 0
      },
      {
        "node": "If Unhealthy",
        "type": "main",
        "index": 0
      }
    ]
  ]
}
```

CRITICAL RULE: All monitoring workflows MUST follow this pattern to prevent incorrect notifications!

5. If Unhealthy Condition Logic

Original:

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

User's Version:

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

Rationale:
- More explicit (checks actual status value)
- Easier to debug (can see exact status in logs)
- Consistent with script output format

Credential Naming Conventions

Current Credentials in Use

SSH Credentials

Workflow	Type	ID	Name	Notes
Container Health Check	`sshPrivateKey`	`8Ct7KD05bVDLL-pnI`	"SSH Private Key account"	User's preference
Backup Automation	`sshPassword`	`1`	"SSH ollivanders.home"	Legacy/fallback

Recommendation: Both authentication methods are valid. Private key is more secure and user's preferred method.

Telegram Credentials

Workflow	ID	Name	Chat ID Method
Container Health Check	`VAPBpHoQv9dvxzNh`	"Telegram account"	Hard-coded: `7787445571`
Backup Automation	`2`	"Telegram Bot"	Variable: `$vars.TELEGRAM_CHAT_ID`

Analysis:
- Hard-coded chat ID: Simpler, works for single-user setups
- Variable-based: More flexible, better for multi-environment deployments

User Preference: Hard-coded chat ID (seen in production workflow)

Uptime Kuma Push URLs

Workflow	Push URL	Monitor ID
Container Health Check	`http://192.168.1.142:3001/api/push/hM6oDQYkfH`	Container health monitor
Backup Automation	`http://192.168.1.142:3001/api/push/6egmEoFola`	Backup monitor

Pattern: Each workflow has its own dedicated push monitor in Uptime Kuma.

Error Handling Patterns

Defensive Parsing (User's Approach)

```
// Always provide fallback values
const output = $input.item.json.stdout || '';
const lines = output.split('\n').filter(l => l.trim());

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

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

Key Principles

1. **Always provide defaults** - Never leave variables undefined
2. **Handle empty output gracefully** - Don't trigger false alerts
3. **Provide diagnostic hints** - Help future debugging
4. **Fail-safe rather than fail-loud** - Prefer warning over alert for transient issues

---

Notification Patterns

Telegram Message Format

Health Check Alert (User's Pattern)

```
⚠️ *Container Health Alert*

*Status:* {{ $json.status }}
*Issue:* {{ $json.message }}

*Details:* {{ $json.details }}

*Time:* {{ $json.timestamp }}
*Server:* ollivanders.home (192.168.1.142)
```

Style Guide:
- Use emojis for visual categorization (⚠️ for warnings, ✅ for success)
- Bold headers with Markdown (`*Header:*`)
- Include timestamp and server identification
- Provide actionable details

Backup Notifications (Established Pattern)

Success:

```
✅ *Media Stack Backup - SUCCESS*

📅 Date: {{ $now.format('yyyy-MM-dd HH:mm:ss') }}
⏱️ Duration: {{ duration }} seconds
💾 Backup Size: {{ size }}
📁 Location: /mnt/server/backup/media-stack/

*Services Backed Up (11 total):*
...

✓ All backups verified
✓ Retention policy applied (2 weeks)
```

Failure:

```
⚠️ *Media Stack Backup - FAILED*

📅 Date: {{ $now.format('yyyy-MM-dd HH:mm:ss') }}
❌ Exit Code: {{ exitCode }}

*Error Output:*
```

{{ last_20_lines_of_error }}

```
⚠️ Please investigate immediately!
📋 Check logs: /mnt/server/logs/backup_improved.log
```

Common Elements:
- Status indicator in title (SUCCESS/FAILED)

- Timestamp (formatted consistently)
- Relevant metrics (duration, size, exit code)
- Actionable next steps for failures
- Use of code blocks for log output

Uptime Kuma Integration

Dual Integration Pattern

The user employs a **redundant notification strategy**:

1. **Script-level notification** (bash script with curl)
2. **Workflow-level notification** (n8n HTTP Request node)

Why Both?

```
# In script - runs regardless of workflow state
curl -fsS -m 10 --retry 3 -G \
  --data-urlencode "status=up" \
  --data-urlencode "msg=Containers OK" \
  "$PUSH_BASE" >/dev/null
```

```
// In workflow - provides workflow-level visibility
{
  "node": "Uptime Kuma Push",
  "parameters": {
    "url": "=http://192.168.1.142:3001/api/push/hM6oDQYkfH",
    "queryParameters": {
      "status": "={{ $json.uptimeKumaStatus }}",
      "msg": "={{ $json.uptimeKumaMsg }}"
    }
  }
}
```

Benefits:
- Script-level ensures monitoring even if workflow breaks
- Workflow-level provides better logging and debugging
- Curl with retries handles transient network issues

Push URL Structure

```
http://192.168.1.142:3001/api/push/[MONITOR_ID]?status=[up|down]&msg=[message]
```

Parameters:
- `status`: `up` or `down`
- `msg`: URL-encoded status message
- `ping`: Optional, defaults to current timestamp

Best Practices

1. Script Design

- ✅ Always use `set -euo pipefail` for bash scripts
- ✅ Export PATH explicitly when running via cron
- ✅ Use structured output (STATUS:, MESSAGE:, DETAILS:)
- ✅ Include retry logic for external API calls (curl)
- ✅ Provide exit codes that workflow can interpret

2. Workflow Design

- ⚠️ **CRITICAL:** Uptime Kuma MUST run in parallel branch to prevent notification interference
- ✅ Use parallel branches for independent operations (monitoring vs. conditional alerts)
- ✅ Always parse script output with defensive code
- ✅ Provide meaningful node names for debugging
- ✅ Include notes on nodes explaining their purpose
- ✅ Use dedicated Uptime Kuma monitors per workflow
- ⚠️ **Never put Uptime Kuma in the main sequential flow before conditional notifications**

3. Error Handling

- ✅ Default to non-alarming states for transient issues
- ✅ Distinguish between “no data” and “bad data”
- ✅ Provide diagnostic hints in error messages
- ✅ Include relevant context (timestamps, server names)

4. Notifications

- ✅ Send success notifications for long-running tasks (backups)
- ✅ Send failure notifications for all critical operations
- ✅ Include actionable information (log paths, commands)
- ✅ Use consistent formatting across workflows
- ✅ Add emojis for quick visual scanning

5. Credentials

- ✅ Use descriptive names (“SSH ollivanders.home”, not “SSH1”)
- ✅ Prefer SSH keys over passwords where possible
- ✅ Keep credential IDs consistent across related workflows
- ✅ Document which credentials each workflow requires

6. Testing

- ✅ Test scripts independently before workflow integration
- ✅ Test failure scenarios (stopped containers, missing files)
- ✅ Verify Uptime Kuma receives status updates
- ✅ Confirm Telegram messages are properly formatted
- ✅ Check logs to ensure proper parsing

Workflow-Specific Notes

Container Health Check

- **Frequency:** Every 5 minutes (`*/5 * * * *`)
- **Monitored Containers:** plex, sonarr, radarr, overseerr, tautulli
- **Uptime Kuma ID:** `hM6oDQYkfH`
- **Notification Strategy:** Only on failure (reduces noise)
- **Special Handling:** Checks both State.Status and State.Health

Backup Automation

- **Frequency:** Every 3 days at midnight (`0 0 */3 * *`)
- **Backed Up Services:** 11 total (9 local + 2 remote)
- **Uptime Kuma ID:** `6egmEoFola`
- **Notification Strategy:** Always (success and failure)
- **Special Features:** Parses output for statistics, 2-week retention

---

Future Considerations

For Phase 3 and Beyond

1. **Centralized Configuration**
   - Consider using n8n environment variables for:
     - Telegram chat ID
     - Uptime Kuma base URL
     - Server hostnames
2. **Workflow Templates**
   - Create reusable sub-workflows for common patterns:
     - Telegram notification formatting
     - Uptime Kuma status updates
     - Script output parsing
3. **Monitoring Enhancements**
   - Add workflow execution history tracking
   - Implement alert suppression for known issues
   - Create dashboard for workflow status overview
4. **Documentation**
   - Keep this document updated with each new workflow
   - Document any new patterns or improvements
   - Note any changes to infrastructure or credentials

Compatibility Notes

backup-automation.json Compatibility

✅ **Compatible** - No changes needed

Analysis:
- Uses different SSH authentication (password vs. key) - Both valid
- Uses Telegram variables instead of hard-coded ID - Also valid
- Different Uptime Kuma monitor IDs - Expected (separate monitors)
- Same parsing patterns and error handling - Consistent

Recommendation: Keep both workflows as-is. The slight differences reflect appropriate adaptations for their different purposes:
- Health checks benefit from embedded monitoring (fast execution)
- Backups benefit from detailed workflow-level parsing (statistics)

Version History

- **2025-12-24** - Initial documentation based on Phase 1 & Phase 2 workflows
- Created from analysis of user's production Container Health Check workflow
- Captures patterns for future workflow development

Last Updated: December 24, 2025
Analyzed Workflows: container-health-check.json, backup-automation.json
Status: Phase 2 Complete