Phase 3 Setup Guide: Docker Image Update Automation

Overview

Phase 3 migrates your Docker image update automation from cron to n8n, adding comprehensive health checks, automatic rollback capabilities, and intelligent notifications. This phase ensures your containers are always running the latest images while maintaining system stability.

Phase 3 Goals

1. **Automated Updates**: Weekly Docker image updates for all 11 services
2. **Health Validation**: Pre and post-update health checks for all containers
3. **Automatic Rollback**: Intelligent rollback when updates fail health checks
4. **Detailed Notifications**: Success/failure alerts with comprehensive details
5. **Centralized Management**: All automation managed through n8n workflows

Services Covered

Local Containers (9):
- plex, sonarr, radarr, overseerr, tautulli
- heimdall
- uptime-kuma
- scrypted
- frigate

Remote Containers (2):
- sabnzbd @ 192.168.4.99 (dainja)
- nginx-proxy-manager @ 192.168.1.236 (dainja)

Understanding the Update Strategy

How Updates Work

The improved update script follows a systematic approach for each service:

1. **Pre-Update Health Check**
   - Verifies container is running
   - Checks Docker healthcheck status (if defined)
   - Documents baseline health state

2. **Image Update**
   - Pulls latest image from Docker Hub
   - Preserves old image in local cache

3. **Container Recreation**
   - Stops old container

   - Creates new container with updated image
   - Maintains all volumes and configurations

4. **Post-Update Health Check**
   - Waits 10 seconds for stabilization
   - Verifies container started successfully
   - Checks healthcheck status

5. **Automatic Rollback** (if post-update fails)
   - Attempts `docker compose restart` first
   - If that fails, container requires manual intervention
   - All rollback attempts are logged

Health Check Criteria

A container is considered “healthy” if:
- Container exists in Docker
- Container status is “running”
- If healthcheck is defined: health status is “healthy”
- If no healthcheck is defined: running status is sufficient

Rollback Limitations

Important: The rollback mechanism has limitations:
- Rollback attempts to restart containers, not revert to old images
- Old images remain in Docker cache but aren’t automatically re-deployed
- If rollback fails, manual intervention is required
- This is a safety measure to prevent automated damage to complex systems

---

Prerequisites Check

Before starting Phase 3, ensure:

Phase 1 & 2 Must Be Complete

- [ ] Git repository initialized (`/home/ubuntu/homelab-automation`)
- [ ] n8n is running and accessible
- [ ] Telegram bot configured and working
- [ ] Phase 1 & 2 workflows are active and functional
- [ ] Uptime Kuma is running with monitor created

Required Access

- [ ] SSH access to ollivanders.home (192.168.1.142)
- [ ] SSH access to remote hosts:
- [ ] dainja@192.168.4.99 (sabnzbd)
- [ ] dainja@192.168.1.236 (nginx-proxy-manager)
- [ ] SSH keys configured for passwordless authentication to remote hosts

Verify Docker Compose Locations

Ensure all Docker Compose files exist at these locations:

```
# Local projects
ls -la /mnt/server/plex/docker-compose.yml
ls -la /mnt/server/tools/heimdall/docker-compose.yml
ls -la /mnt/server/tools/uptime_kuma/docker-compose.yml
ls -la /mnt/server/scrypted/docker-compose.yml
ls -la /mnt/server/frigate/docker-compose.yml

# Remote projects (check via SSH)
ssh dainja@192.168.4.99 "ls -la /home/dainja/sabnzbd/docker-compose.yml"
ssh dainja@192.168.1.236 "ls -la /opt/npm/docker-compose.yml"
```

Uptime Kuma Monitor Setup

You need a dedicated Uptime Kuma monitor for docker updates:

1. Log into Uptime Kuma: http://192.168.1.142:3001
2. Create a new “Push” monitor:
   - **Name:** Docker Image Updates
   - **Type:** Push
   - **Heartbeat Interval:** 7 days + 1 hour (give some grace period)
3. Copy the Push URL (looks like: `http://192.168.1.142:3001/api/push/MONITOR_ID`)
4. Extract the `MONITOR_ID` from the URL
5. Update the workflow JSON with your monitor ID (see workflow configuration section)

Installation Steps

Step 1: Update Git Repository

```
# Navigate to your homelab automation directory
cd /home/ubuntu/homelab-automation

# Pull the latest Phase 3 changes
git pull origin main

# Verify Phase 3 files exist
ls -la scripts/docker_update_improved.sh
ls -la workflows/docker-update-automation.json
ls -la docs/PHASE3_SETUP_GUIDE.md
```

Step 2: Install the Improved Update Script

```
# The script should already be executable, but verify
chmod +x /home/ubuntu/homelab-automation/scripts/docker_update_improved.sh

# Create a symbolic link for easier access (optional)
sudo ln -sf /home/ubuntu/homelab-automation/scripts/docker_update_improved.sh \
    /usr/local/bin/docker_update_improved.sh
```

Step 3: Test the Script Manually

IMPORTANT: Always test the script manually before scheduling it!

```
# Test with a dry run (check if services are accessible)
/home/ubuntu/homelab-automation/scripts/docker_update_improved.sh

# Check the log output
tail -n 100 /mnt/server/logs/docker_update_improved.log
```

Expected Output:
- Pre-update health checks for all services
- Image pulls for each service
- Container recreation
- Post-update health checks
- Summary with counts and duration
- Final status: `STATUS=SUCCESS` or `STATUS=FAILURE`

What to Look For:
- ✅ All containers should be healthy before update
- ✅ Image pulls should succeed
- ✅ Containers should restart successfully
- ✅ Post-update health checks should pass
- ⚠️ Any warnings should be investigated

If Script Fails:
- Check SSH connectivity to remote hosts
- Verify Docker Compose files exist
- Ensure containers are running before update
- Review error messages in the log file

Step 4: Configure Uptime Kuma Monitor ID

CRITICAL: You must update the workflow with YOUR Uptime Kuma monitor ID!

1. Get your monitor ID from Uptime Kuma (see Prerequisites section)
2. Edit the workflow JSON file:

```
cd /home/ubuntu/homelab-automation/workflows
nano docker-update-automation.json
```

1. Find the “Uptime Kuma Push” node (around line 125)
2. Replace `is26MWqum3` with YOUR monitor ID:

```
"url": "=http://192.168.1.142:3001/api/push/YOUR_MONITOR_ID_HERE",
```

1. Save and exit (Ctrl+X, Y, Enter)

Step 5: Import the n8n Workflow

1. Open n8n in your browser: `http://192.168.1.142:5678`
2. Click **Workflows → Add Workflow → Import from File**
3. Select: `/home/ubuntu/homelab-automation/workflows/docker-update-automation.json`
4. The workflow should import with all nodes configured

5. **Verify the workflow structure:**
   - Schedule trigger: Sundays at 2 AM
   - SSH execution node
   - Parse results node
   - **PARALLEL branches** from Parse Results:
     - Branch 1: Uptime Kuma Push (always runs)
     - Branch 2: Check Update Status → Telegram notifications

Step 6: Configure Workflow Settings

After importing, configure these settings:

A. SSH Credentials

1. Click on **“Execute Docker Update Script”** node
2. Under **Credentials**, select your SSH credentials for ollivanders.home
   - Should be the same credentials used in Phase 1 & 2
   - Type: SSH Password Authentication
   - Label: “SSH ollivanders.home”

B. Telegram Credentials

1. Click on **“Telegram - Success”** node
2. Verify Telegram Bot credentials are selected
3. Repeat for **“Telegram - Failure”** node

C. Telegram Chat ID

The workflow uses the n8n variable `$vars.TELEGRAM_CHAT_ID`:

1. Go to **Settings** → **Variables** in n8n
2. Ensure `TELEGRAM_CHAT_ID` is set to your chat ID
   - Should be set from Phase 1
   - Format: Usually a negative number for groups, positive for users

D. Schedule Configuration

Verify the schedule (should already be configured):
- **Cron Expression:** `0 2 * * 0`
- **Meaning:** Every Sunday at 2:00 AM
- **Timezone:** America/New_York (from n8n configuration)

Understanding the Workflow Architecture

Why Parallel Execution is Critical

MANDATORY PATTERN: The workflow uses parallel execution for Uptime Kuma notifications.

```
Parse Results
     ├→ Uptime Kuma Push (Branch 1: Always runs)
     └→ Check Update Status (Branch 2: Conditional Telegram)
            ├→ Telegram Success
            └→ Telegram Failure
```

Why This Matters:
- In sequential execution, Uptime Kuma could interfere with conditional logic
- Previous versions sent BOTH success AND failure Telegram messages (bug!)
- Parallel execution isolates Uptime Kuma from Telegram decision logic
- This pattern is consistent across all Phase 1-3 workflows

Reference: See `docs/WORKFLOW_PATTERNS.md` for detailed explanation.

How the Parse Node Works

The **“Parse Docker Update Output”** node extracts data from the script:

Script Output Format:

```
STATUS=SUCCESS
UPDATED=11
FAILED=0
DURATION=45s
WARNINGS=0
```

Parsed Data Structure:

```
{
  status: "SUCCESS" or "FAILURE",
  updatedCount: 11,
  failedCount: 0,
  duration: "45s",
  warningCount: 0,
  failedServices: "service1 service2" (only on failure),
  uptimeKumaStatus: "up" or "down",
  uptimeKumaMsg: "Docker Update OK - 11 services updated"
}
```

Defensive Parsing

The parse node includes defensive logic:
- Handles missing output gracefully
- Falls back to exit codes if parsing fails
- Ensures status is always “SUCCESS” or “FAILURE”
- Extracts last 30 lines of output for error reporting

Testing the Workflow

Test 1: Manual Execution (Success Scenario)

1. In n8n, open the “Docker Image Update Automation” workflow
2. Click **“Execute Workflow” → “Save and Execute”**
3. Watch the execution flow:
   - ✅ Schedule triggers (or manual execution)
   - ✅ SSH executes script
   - ✅ Parse node extracts data

- ✅ Uptime Kuma gets notified (parallel)
- ✅ Telegram success message sent (if status is SUCCESS)

4. **Check Telegram:**
   - Should receive ONE success message
   - Should NOT receive failure message

5. **Check Uptime Kuma:**
   - Docker Updates monitor should show “Up”
   - Status message should show update count

Test 2: Verify Parallel Execution

CRITICAL CHECK: Ensure branches are truly parallel!

1. In n8n workflow editor, click on **“Parse Docker Update Output”** node

2. Look at the connections:
   - Should have TWO output arrows
   - One to “Uptime Kuma Push”
   - One to “Check Update Status”

3. Verify node positions:
   - Uptime Kuma and Check Status should be side-by-side (same Y position)
   - NOT one above the other

Why This Matters: If nodes are sequential, you may get incorrect notifications!

Test 3: Failure Scenario (Optional)

To test failure handling (ONLY if you’re comfortable):

1. Temporarily stop a container:
   ```
   bash
     docker stop plex
   ```

2. Manually execute the workflow

3. The script should detect the stopped container and report failure

4. **Expected Results:**
   - Uptime Kuma: Status “Down”
   - Telegram: Failure message with details

5. Restore the container:
   ```
   bash
     docker start plex
   ```

Note: This test is optional. Don’t test in production if you’re uncomfortable with downtime!

Disabling the Old Cron Job

Once you’ve verified the n8n workflow works correctly, disable the old cron job:

Step 1: Backup Current Crontab

```
crontab -l > ~/crontab_backup_phase3_$(date +%Y%m%d).txt
```

Step 2: Edit Crontab

```
crontab -e
```

Step 3: Comment Out the Docker Update Job

Find this line (or similar):

```
0 2 * * 0 /bin/bash -lc '/mnt/server/tools/backup_and_restore/media_stack/dock-
er_daily_pull.sh >> /mnt/server/logs/docker-update-cron.log 2>&1; ec=$?; if [ $ec -
eq 0 ]; then curl -fsS -m 10 --retry 3 "http://192.168.1.142:3001/api/push/is26MWqum3?
status=up&msg=Docker+Update+OK"; else curl -fsS -m 10 --retry 3 "http://
192.168.1.142:3001/api/push/is26MWqum3?status=down&msg=Docker+Update+FAILED+exit+
$ec"; fi'
```

Comment it out:

```
# DISABLED: Migrated to n8n (Phase 3) - 2025-12-24
# 0 2 * * 0 /bin/bash -lc '/mnt/server/tools/backup_and_restore/media_stack/dock-
er_daily_pull.sh >> /mnt/server/logs/docker-update-cron.log 2>&1; ec=$?; if [ $ec -eq
0 ]; then curl -fsS -m 10 --retry 3 "http://192.168.1.142:3001/api/push/is26MWqum3?
status=up&msg=Docker+Update+OK"; else curl -fsS -m 10 --retry 3 "http://
192.168.1.142:3001/api/push/is26MWqum3?status=down&msg=Docker+Update+FAILED+exit+$ec";
fi'
```

Step 4: Verify Cron Changes

```
crontab -l | grep -i docker
```

Should show the commented line.

Activating the n8n Workflow

Enable the Workflow

1. In n8n, open the “Docker Image Update Automation” workflow
2. Toggle the workflow to **Active** (top-right corner)
3. Verify the schedule is set correctly:
   - Next execution should show “Next Sunday at 2:00 AM”

Monitoring Activation

After activation:

1. **Uptime Kuma:**
   - Docker Updates monitor is now being updated by n8n
   - Old cron Uptime Kuma pushes are disabled

2. **First Automated Run:**
   - Will occur next Sunday at 2 AM
   - You'll receive Telegram notification
   - Check logs after first run: `/mnt/server/logs/docker_update_improved.log`

Monitoring and Validation

Daily Checks (First Week)

For the first week after activation, monitor:

1. **Uptime Kuma Dashboard:**
   - Docker Updates monitor status
   - Should update weekly after Sunday 2 AM runs

2. **Telegram Notifications:**
   - Should receive ONE message per week (success or failure)
   - Message should include update count and duration

3. **Log Files:**
   ```bash
   # Check latest update log
   ls -lth /mnt/server/logs/docker_update_improved.log* | head -5

# View most recent log
less /mnt/server/logs/docker_update_improved.log
```

1. **Container Health:**
   ```bash
   # Quick health check of all local containers
   docker ps –filter "name=plex|sonarr|radarr|overseerr|tautulli|heimdall|uptime-kuma|scrypted|frigate"

# Check specific container health
docker inspect plex –format='{{.State.Status}} - {{if .State.Health}}{{.State.Health.Status}}
{{else}}no healthcheck{{end}}'
```

Weekly Validation

After each Sunday update:

1. **Review Telegram Notification:**
   - Verify update count is correct (should be 11 if all succeed)

- Check for any warnings
- Note duration (normal: 2-5 minutes)

2. **Spot Check Containers:**
```bash
# Check a few containers are running
docker ps | grep -E “plex|sonarr|heimdall”

# Check remote containers
ssh dainja@192.168.4.99 “docker ps | grep sabnzbd”
ssh dainja@192.168.1.236 “docker ps | grep nginx-proxy-manager”
```

1. **Review Logs for Anomalies:**
```bash
# Check for any errors or warnings
grep -E "ERROR|WARN|FAIL" /mnt/server/logs/docker_update_improved.log
```

Monthly Audit

Once per month:

- Review all docker update logs
- Verify all services are receiving updates
- Check log file sizes and cleanup is working
- Confirm Uptime Kuma monitor is healthy

Troubleshooting

Issue: Script Fails to Execute via n8n

Symptoms:
- Workflow execution fails at “Execute Docker Update Script” node
- Error: “Command failed” or “SSH connection failed”

Solutions:

1. **Verify SSH Credentials:**
```bash
# Test SSH connection manually
ssh dainja@192.168.1.142 "echo 'SSH works'"
```

2. **Check Script Permissions:**
```bash
ls -la /home/ubuntu/homelab-automation/scripts/docker_update_improved.sh
# Should show: -rwxr-xr-x (executable)
```

3. **Verify Script Path:**
- n8n SSH node should have: `/home/ubuntu/homelab-automation/scripts/docker_update_improved.sh`
- Path must be absolute, not relative

Issue: Parse Node Fails to Extract Data

Symptoms:
- Parse node completes but data is missing
- Uptime Kuma or Telegram notifications have “Unknown” values

Solutions:

1. **Check Script Output Format:**
   ```bash
   # Run script manually and check output
   /home/ubuntu/homelab-automation/scripts/docker_update_improved.sh
   # Last few lines should show: STATUS=, UPDATED=, etc.
   ```

2. **Verify Defensive Parsing:**
   - Parse node should handle missing data gracefully
   - Should fall back to exit code if parsing fails

3. **Enable Debug Mode:**
   - In n8n, click “Execute Workflow” and watch the data flow
   - Check the output of “Parse Docker Update Output” node
   - Verify `status` field is “SUCCESS” or “FAILURE”

Issue: Both Success and Failure Telegram Messages Sent

Symptoms:
- Receiving TWO Telegram messages after each update
- One success message AND one failure message

ROOT CAUSE: Uptime Kuma is NOT running in parallel!

Solution:

1. **Verify Workflow Structure:**
   - Open workflow in n8n editor
   - Click on “Parse Docker Update Output” node
   - Check connections: should have TWO outputs
   - Both “Uptime Kuma Push” and “Check Update Status” should connect directly to Parse node

2. **Recreate Connections if Needed:**
   - Delete the connection between Parse and Check Status
   - Delete the connection between Parse and Uptime Kuma
   - Recreate BOTH connections from Parse node output
   - Ensure both target nodes are at the same vertical position (parallel)

3. **Reference:**
   - See `docs/WORKFLOW_PATTERNS.md` for detailed explanation
   - See `docs/USER_MODIFICATIONS_SUMMARY.md` for bug context

Issue: Remote Container Updates Fail

Symptoms:
- Local containers update successfully
- Remote containers (sabnzbd or nginx-proxy-manager) fail

Solutions:

1. **Verify SSH Access to Remote Hosts:**
   ```bash
   # Test SSH to sabnzbd host
   ssh dainja@192.168.4.99 “docker ps”

# Test SSH to nginx-proxy-manager host
ssh dainja@192.168.1.236 “docker ps”
```

1. **Check Docker Compose Paths:**
   ```bash
   # Verify compose files exist
   ssh dainja@192.168.4.99 "ls -la /home/dainja/sabnzbd/docker-compose.yml"
   ssh dainja@192.168.1.236 "ls -la /opt/npm/docker-compose.yml"
   ```

2. **Verify SSH Keys:**
   - Ensure SSH keys are configured for passwordless authentication
   - Test: `ssh dainja@192.168.4.99 "echo 'test'"` should not prompt for password

3. **Check Network Connectivity:**
   ```bash
   # Ping remote hosts
   ping -c 3 192.168.4.99
   ping -c 3 192.168.1.236
   ```

Issue: Post-Update Health Check Fails

Symptoms:
- Container updates successfully
- Post-update health check reports container unhealthy
- Rollback is triggered

Solutions:

1. **Check Container Startup Time:**
   - Some containers take longer than 10 seconds to start
   - Edit script to increase wait time:
   ```bash
     # Find line: sleep 10
     # Change to: sleep 20 (or 30)
   ```

2. **Verify Healthcheck Configuration:**
   ```bash
   # Check if container has healthcheck defined
   docker inspect plex --format='{{if .State.Health}}{{.State.Health.Status}}{{else}}no healthcheck{{end}}'
   ```

3. **Review Container Logs:**
   ```bash
   # Check why container is unhealthy
   docker logs plex --tail 50
   ```

4. **Disable Healthcheck Validation (Last Resort):**
   - Edit script to only check running status, not health status
   - This is NOT recommended unless you understand the risks

Issue: Rollback Fails After Update Failure

Symptoms:
- Post-update health check fails
- Rollback attempt also fails
- Container is left in broken state

Solutions:

1. **Manual Container Restart:**
   ```bash
   # Navigate to compose directory
   cd /mnt/server/plex

# Restart the service
docker compose restart plex
```

1. **Revert to Previous Image (Advanced):**
   ```bash
   # Pull specific version (if you know it)
   docker pull linuxserver/plex:1.32.8

# Recreate container with specific version
docker compose up -d plex
```

1. **Check for Configuration Issues:**
   - Verify docker-compose.yml is valid
   - Check volume mounts are correct
   - Ensure environment variables are set

2. **Emergency Recovery:**
   ```bash
   # Stop and remove container
   docker compose down plex

# Recreate from scratch
docker compose up -d plex
```

Issue: Uptime Kuma Not Receiving Updates

Symptoms:
- Workflow executes successfully
- Telegram notifications work
- Uptime Kuma monitor shows “Pending” or outdated status

Solutions:

1. **Verify Monitor ID:**
   - Check workflow: Uptime Kuma Push URL

- Confirm monitor ID matches your Uptime Kuma setup
- Format: `http://192.168.1.142:3001/api/push/YOUR_MONITOR_ID`

2. **Test Uptime Kuma Endpoint:**
```bash
# Test push manually
curl "http://192.168.1.142:3001/api/push/YOUR_MONITOR_ID?status=up&msg=test"
```

3. **Check Uptime Kuma Logs:**
```bash
# View Uptime Kuma container logs
docker logs uptime-kuma --tail 50
```

4. **Verify Network Connectivity:**
- Ensure ollivanders.home can reach Uptime Kuma
- Test: `curl http://192.168.1.142:3001`

Rollback Procedures

Rolling Back the Entire Phase 3

If you need to revert to the old cron-based system:

1. **Disable n8n Workflow:**
- Open workflow in n8n
- Toggle to **Inactive**

2. **Re-enable Cron Job:**
```bash
crontab -e
# Uncomment the docker update line
```

3. **Verify Old Script Still Works:**
```bash
/mnt/server/tools/backup_and_restore/media_stack/docker_daily_pull.sh
```

Rolling Back a Single Failed Update

If a specific container fails to update:

1. **Check Current Image:**
```bash
docker inspect plex --format='{{.Config.Image}}'
```

2. **Pull Previous Version:**
```bash
# Find previous version from Docker Hub
# Example: linuxserver/plex:1.32.8
docker pull linuxserver/plex:1.32.8
```

3. **Update docker-compose.yml:**
```yaml
services:
```

```
    plex:
      image: linuxserver/plex:1.32.8  # Pin to specific version
```

4. **Recreate Container:**
   ```
   bash
   cd /mnt/server/plex
   docker compose up -d plex
   ```

5. **Skip in Future Updates:**
   - Temporarily remove from LOCAL_PROJECTS in script
   - Or pin image version in docker-compose.yml

Performance Considerations

Update Duration

Expected Duration:
- Local containers (9): 2-3 minutes
- Remote containers (2): 1-2 minutes
- **Total:** 3-5 minutes for all 11 services

Factors Affecting Duration:
- Image sizes (Plex is large, ~1GB)
- Network speed (especially for remote hosts)
- Container startup time
- Number of healthchecks

Resource Usage

During Updates:
- CPU: Moderate (pulling/extracting images)
- RAM: Minimal (containers share same RAM footprint)
- Disk: Temporary increase (old + new images)
- Network: High (downloading images)

Optimization Tips:
- Run updates during low-usage periods (2 AM is ideal)
- Ensure sufficient disk space (/var/lib/docker)
- Monitor network bandwidth if other services are running

Next Steps

Phase 4 Preview: Future Automation Ideas

Now that Phases 1-3 are complete, consider these future automations:

1. **Disk Space Monitoring:**
   - Alert when disk usage exceeds threshold
   - Automated cleanup of old Docker images

2. **Service Availability Monitoring:**
   - Periodic checks of service endpoints
   - Alert when services are unreachable

3. **Certificate Renewal Monitoring:**
   - Track SSL certificate expiration
   - Alert before certificates expire

4. **Media Library Statistics:**
   - Weekly reports on library size
   - New content notifications

5. **Resource Usage Reporting:**
   - CPU/RAM usage trends
   - Container resource limits

Documentation

Keep These Docs Updated:
- Update `README.md` with current status
- Document any customizations in `USER_MODIFICATIONS_SUMMARY.md`
- Track issues and solutions in your own notes

Maintenance Schedule

Weekly:
- Review Telegram notifications
- Spot check container health

Monthly:
- Review all docker update logs
- Verify all services updated successfully
- Check log file sizes

Quarterly:
- Audit all n8n workflows
- Review Uptime Kuma monitors
- Test failure scenarios

Summary

You've successfully completed Phase 3! Your Docker image updates are now:

✅ **Automated:** Weekly updates via n8n, no manual intervention
✅ **Safe:** Health checks before and after updates
✅ **Resilient:** Automatic rollback on failures
✅ **Transparent:** Detailed Telegram notifications
✅ **Monitored:** Uptime Kuma tracking with proper parallel execution

Your Complete Automation Stack:
- **Phase 1:** Container health monitoring (every 5 minutes)
- **Phase 2:** Automated backups (every 3 days)
- **Phase 3:** Docker image updates (weekly)

All systems are now fully migrated to n8n with intelligent orchestration!

For questions or issues, refer to:
- `docs/WORKFLOW_PATTERNS.md` - Workflow design patterns
- `docs/USER_MODIFICATIONS_SUMMARY.md` - Historical context
- GitHub repository issues section

Document Version: 1.0
Last Updated: 2025-12-24
Phase 3 Status: ✅ Complete