Phase 1 Setup Guide: Container Health Monitoring with n8n

📋 Table of Contents

1. Overview
2. Prerequisites
3. Step 1: Install Git
4. Step 2: Set Up GitHub Account
5. Step 3: Configure SSH Keys for GitHub
6. Step 4: Create GitHub Repository
7. Step 5: Initialize Local Git Repository
8. Step 6: Install n8n
9. Step 7: Create Telegram Bot
10. Step 8: Configure n8n Credentials
11. Step 9: Import and Configure Workflow
12. Step 10: Test the Workflow
13. Step 11: Disable Old Cron Job
14. Troubleshooting
15. Next Steps

Overview

This guide will help you migrate your container health check automation from a cron job to n8n. By the end, you'll have:

- ✅ n8n running on your server (ollivanders.home)
- ✅ Automated container health checks every 5 minutes
- ✅ Uptime Kuma integration for monitoring
- ✅ Telegram notifications for failures
- ✅ Version-controlled automation with Git

Time Required: 1-2 hours (first time)
Difficulty: Beginner-friendly with detailed instructions
Server: ollivanders.home (192.168.1.142)

Prerequisites

What You Need

- [ ] Access to ollivanders.home (192.168.1.142) via SSH or direct login
- [ ] Root or sudo privileges on the server

- [ ] Docker and Docker Compose already installed (you have this!)
- [ ] Active internet connection
- [ ] A Telegram account (for notifications)
- [ ] Basic command-line knowledge

Verify Docker Installation

```
# Check Docker version
docker --version
# Should show: Docker version 20.x or higher

# Check Docker Compose version
docker compose version
# Should show: Docker Compose version v2.x or higher
```

If these commands fail, Docker needs to be installed first.

Step 1: Install Git

Git is used for version control of your automation scripts and configurations.

1.1 Check if Git is Already Installed

```
git --version
```

If you see a version number (e.g., `git version 2.x.x`), skip to Step 2.

1.2 Install Git on RHEL

```
# Install Git using dnf (RHEL package manager)
sudo dnf install git -y

# Verify installation
git --version
```

1.3 Configure Git (First Time Setup)

```
# Set your name (replace with your actual name)
git config --global user.name "Your Name"

# Set your email (use the email you'll use for GitHub)
git config --global user.email "your.email@example.com"

# Verify configuration
git config --list
```

Step 2: Set Up GitHub Account

If you already have a GitHub account, skip to Step 3.

2.1 Create a GitHub Account

1. Go to https://github.com/signup (https://github.com/signup)
2. Enter your email address
3. Create a password
4. Choose a username
5. Complete verification
6. Check your email and verify your account

2.2 Why GitHub?

- **Version Control:** Track all changes to your automation
- **Backup:** Your configurations are safely stored in the cloud
- **Collaboration:** Easy to share or ask for help
- **Documentation:** Built-in wiki and README rendering

Step 3: Configure SSH Keys for GitHub

SSH keys allow you to securely push code to GitHub without entering your password every time.

3.1 Check for Existing SSH Keys

```
# Check if you already have SSH keys
ls -la ~/.ssh/id_*.pub
```

If you see files like `id_rsa.pub` or `id_ed25519.pub`, you already have keys. Skip to 3.3.

3.2 Generate New SSH Key

```
# Generate a new SSH key (replace with your GitHub email)
ssh-keygen -t ed25519 -C "your.email@example.com"

# When prompted:
# - Press Enter to accept default file location
# - Enter a passphrase (or press Enter for no passphrase)
# - Confirm passphrase

# Start the SSH agent
eval "$(ssh-agent -s)"

# Add your SSH key to the agent
ssh-add ~/.ssh/id_ed25519
```

3.3 Copy Your Public Key

```
# Display your public key
cat ~/.ssh/id_ed25519.pub

# The output will look like:
# ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGx... your.email@example.com
```

Copy the entire output (including `ssh-ed25519` at the beginning).

3.4 Add SSH Key to GitHub

1. Log in to GitHub
2. Click your profile picture → **Settings**
3. In the left sidebar, click **SSH and GPG keys**
4. Click **New SSH key**
5. Give it a title (e.g., “ollivanders.home”)
6. Paste your public key into the “Key” field
7. Click **Add SSH key**

3.5 Test SSH Connection

```
# Test connection to GitHub
ssh -T git@github.com

# You should see:
# Hi username! You've successfully authenticated, but GitHub does not provide shell
access.
```

Step 4: Create GitHub Repository

4.1 Create Repository on GitHub

1. Log in to GitHub
2. Click the **+** icon in the top right → **New repository**
3. Repository name: `homelab-automation`
4. Description: `Home lab automation with n8n - migrating from cron jobs`
5. Choose **Private** (recommended) or **Public**
6. **DO NOT** check “Initialize with README” (we already have one)
7. Click **Create repository**

4.2 Note Your Repository URL

GitHub will show you a page with setup instructions. You’ll need the SSH URL, which looks like:

```
git@github.com:yourusername/homelab-automation.git
```

Save this URL - you’ll need it in Step 5.

Step 5: Initialize Local Git Repository

Now we’ll turn your local folder into a Git repository and push it to GitHub.

5.1 Navigate to Your Project

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

5.2 Initialize Git Repository

```
# Initialize Git repository
git init

# Check status (should show untracked files)
git status
```

5.3 Create .gitignore File

```
# Create .gitignore to exclude sensitive files
cat > .gitignore << 'EOF'
# Environment files with secrets
n8n/.env

# n8n data directory (if mounted locally)
n8n/data/

# Log files
*.log

# OS files
.DS_Store
Thumbs.db

# Editor files
.vscode/
.idea/
*.swp
*.swo
EOF
```

5.4 Stage and Commit Files

```
# Add all files to staging
git add .

# Check what will be committed
git status

# Create your first commit
git commit -m "Initial commit: Phase 1 - Container health monitoring setup"
```

5.5 Connect to GitHub and Push

```
# Add GitHub as remote (replace with YOUR repository URL from Step 4.2)
git remote add origin git@github.com:yourusername/homelab-automation.git

# Verify remote
git remote -v

# Push to GitHub (main branch)
git branch -M main
git push -u origin main
```

5.6 Verify on GitHub

1. Go to your GitHub repository in a browser
2. You should see all your files (README.md, scripts/, n8n/, workflows/, docs/)
3. Notice that `n8n/.env` is NOT there (that's correct - it's in .gitignore)

🎉 **Congratulations!** Your code is now version-controlled and backed up to GitHub.

Step 6: Install n8n

Now let's install n8n using Docker Compose.

6.1 Navigate to n8n Directory

```
cd /home/ubuntu/homelab-automation/n8n
```

6.2 Create Environment File

```
# Copy the example environment file
cp .env.example .env

# Edit the .env file
nano .env
```

6.3 Configure Environment Variables

In the `nano` editor, update these values:

```
# Change this to a strong password!
POSTGRES_PASSWORD=YourStrongPasswordHere123!

# Change this to a strong password!
N8N_BASIC_AUTH_PASSWORD=YourN8nPasswordHere456!

# Optional: change the username
N8N_BASIC_AUTH_USER=admin
```

Important Security Notes:
- Use strong, unique passwords
- Never commit the `.env` file to Git (it's already in .gitignore)
- Keep these credentials safe - write them down or use a password manager

Save and exit nano:
- Press `Ctrl + X`
- Press `Y` to confirm
- Press `Enter` to save

6.4 Start n8n

```
# Start n8n and PostgreSQL in detached mode
docker compose up -d

# Check if containers are running
docker compose ps

# You should see two containers:
# - n8n (running on port 5678)
# - n8n-postgres (PostgreSQL database)
```

6.5 Check Logs

```
# View n8n logs
docker compose logs -f n8n

# Wait for this message:
# "n8n ready on 0.0.0.0:5678"

# Press Ctrl+C to exit logs
```

6.6 Access n8n Web Interface

1. Open a web browser
2. Go to: `http://192.168.1.142:5678`
3. You'll see a login prompt
4. Enter:
   - Username: `admin` (or what you set in .env)
   - Password: (the N8N_BASIC_AUTH_PASSWORD from .env)

🎉 **You should now see the n8n interface!**

Step 7: Create Telegram Bot

Telegram will be used for sending failure notifications.

7.1 Create a Bot with BotFather

1. Open Telegram (on your phone or desktop)
2. Search for **@BotFather**
3. Start a conversation and send: `/newbot`
4. Follow the prompts:
   - **Bot name:** `Homelab Monitor` (or any name you like)
   - **Username:** `ollivanders_homelab_bot` (must end in 'bot')
5. BotFather will give you a **token** like: `123456789:ABCdefGHIjklMNOpqrsTUVwxyz`
6. **Save this token** - you'll need it soon

7.2 Get Your Chat ID

You need to know your Telegram Chat ID to receive messages.

```
# Replace YOUR_BOT_TOKEN with the token from BotFather
# First, send a message to your bot in Telegram (any message like "Hello")

# Then run this command on your server:
curl -s "https://api.telegram.org/botYOUR_BOT_TOKEN/getUpdates" | grep -o '"chat":
{"id":[0-9]*' | grep -o '[0-9]*$'

# This will output your Chat ID (a number like 123456789)
```

Alternative method using browser:
1. Replace `YOUR_BOT_TOKEN` in this URL: `https://api.telegram.org/botYOUR_BOT_TOKEN/getUpdates`
2. Open it in a browser
3. Look for `"chat":{"id":123456789` - that number is your Chat ID

Save your Chat ID - you'll need it in Step 9.

Step 8: Configure n8n Credentials

n8n needs credentials to connect to your server and Telegram.

8.1 Create SSH Credential (for localhost)

1. In n8n, click your **profile icon** (top right) → **Settings** → **Credentials**
2. Click **Add Credential**
3. Search for "SSH" and select **SSH**
4. Fill in:
   - **Name:** `SSH localhost`
   - **Host:** `localhost`
   - **Port:** `22`
   - **Username:** Your server username (likely `ubuntu` or your current user)
   - **Authentication Method:** `Password` or `Private Key`

If using Password:
- **Password:** Your server password

If using Private Key (more secure):
- **Private Key:** Paste your private SSH key
`bash`
```
    # Display your private key
    cat ~/.ssh/id_rsa
    # Copy the entire output including BEGIN and END lines
```

1. Click **Test** to verify connection
2. Click **Save**

8.2 Create Telegram Credential

1. In n8n, go back to **Settings** → **Credentials**
2. Click **Add Credential**
3. Search for "Telegram" and select **Telegram API**

4. Fill in:
   - **Name:** `Telegram Bot`
   - **Access Token:** (paste the bot token from Step 7.1)
5. Click **Test** (optional)
6. Click **Save**

Step 9: Import and Configure Workflow

Now let’s import the pre-built container health check workflow.

9.1 Import Workflow

1. In n8n, click **Workflows** in the left sidebar
2. Click the **+** button to create a new workflow
3. Click the **⋮** (three dots) menu in the top right
4. Select **Import from File**
5. Navigate to: `/home/ubuntu/homelab-automation/workflows/container-health-check.json`
6. Click **Open**

The workflow should now appear with all nodes connected.

9.2 Configure SSH Node

1. Click the **SSH: Check Containers** node
2. In the **Credentials** section, select: `SSH localhost` (the credential you created)
3. Verify the command looks correct (it should check Docker containers)

9.3 Configure Telegram Node

1. Click the **Telegram Alert** node
2. Update:
   - **Credentials:** Select `Telegram Bot`
   - **Chat ID:** Enter your Chat ID from Step 7.2 (replace `YOUR_TELEGRAM_CHAT_ID`)
3. Review the message template (customize if desired)

9.4 Verify Uptime Kuma URL

1. Click the **Uptime Kuma Push** node
2. Verify the URL is: `http://192.168.1.142:3001/api/push/hM6oDQYkfH`
3. If your Uptime Kuma push monitor uses a different ID, update it here

9.5 Save Workflow

1. Click the 💾 **Save** button in the top right
2. The workflow is now saved but **NOT active yet**

Step 10: Test the Workflow

Before activating the workflow, let’s test it manually.

10.1 Manual Test Execution

1. In the workflow editor, click **Execute Workflow** (top right)
2. Watch the nodes light up as they execute
3. Check the output of each node by clicking on it
4. Verify:
   - SSH command executed successfully
   - Container status was parsed correctly
   - Uptime Kuma received the push
   - If containers are healthy, Telegram should NOT send (that's correct!)

10.2 Test Failure Notification

To test the Telegram notification:

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

2. In n8n, click **Execute Workflow** again

3. This time, you should receive a Telegram notification!

4. Check your Telegram for the alert message

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

10.3 Check Uptime Kuma

1. Open Uptime Kuma: `http://192.168.1.142:3001`
2. Find your push monitor
3. Verify it's receiving updates

Step 11: Disable Old Cron Job

Once you've verified the workflow works, disable the old cron job.

11.1 View Current Cron Jobs

```
# View your current crontab
crontab -l
```

You should see a line like:

```
*/5 * * * * /bin/bash -lc '/mnt/server/tools/backup_and_restore/media_stack/
check_media_containers.sh >> /mnt/server/logs/container_check.log 2>&1'
```

11.2 Edit Crontab

```
# Edit crontab
crontab -e
```

11.3 Disable the Cron Job

Option 1: Comment it out (recommended for now)

Add a `#` at the beginning of the line:

```
# */5 * * * * /bin/bash -lc '/mnt/server/tools/backup_and_restore/media_stack/
check_media_containers.sh >> /mnt/server/logs/container_check.log 2>&1'
```

Option 2: Delete the line entirely

Just delete the entire line.

11.4 Save and Verify

1. Save and exit the editor (in nano: `Ctrl+X`, `Y`, `Enter`)
2. Verify:
   `bash`
   `   crontab -l`
3. The line should now be commented out or gone

Step 12: Activate n8n Workflow

Final step - activate the workflow!

12.1 Activate Workflow

1. In n8n, open your **Container Health Check** workflow
2. Toggle the **Inactive** switch to **Active** (top right)
3. The workflow is now running and will execute every 5 minutes

12.2 Monitor First Runs

1. Click **Executions** in the left sidebar
2. Watch for new executions every 5 minutes
3. Click on an execution to see details
4. Verify everything is working correctly

Troubleshooting

Issue: Can’t Access n8n Web Interface

Check if n8n is running:

```
cd /home/ubuntu/homelab-automation/n8n
docker compose ps
```

Check logs:

```
docker compose logs -f n8n
```

Restart n8n:

```
docker compose restart n8n
```

Issue: SSH Connection Fails

Test SSH locally:

```
ssh localhost
```

If this fails, you may need to:
1. Install OpenSSH server: `sudo dnf install openssh-server -y`
2. Start SSH service: `sudo systemctl start sshd`
3. Enable SSH on boot: `sudo systemctl enable sshd`

Issue: Telegram Notifications Not Received

Verify bot token:

```
curl "https://api.telegram.org/botYOUR_TOKEN/getMe"
```

Verify you've messaged the bot first (bots can't initiate conversations)

Check Chat ID is correct

Issue: Uptime Kuma Not Receiving Updates

Test the push URL manually:

```
curl -G --data-urlencode "status=up" --data-urlencode "msg=Test" \
  "http://192.168.1.142:3001/api/push/hM6oDQYkfH"
```

Check if the push monitor exists in Uptime Kuma

Issue: Docker Command Not Found in SSH

This means the PATH isn't set correctly for SSH sessions.

Solution: The workflow already uses the full path: `/usr/bin/docker`

If needed, find Docker location:

```
which docker
```

Issue: Workflow Executes but No Output

Check SSH command output:
1. Click the SSH node
2. Look at the "Output" panel
3. Check for error messages

Manually test the script:

```
bash /home/ubuntu/homelab-automation/scripts/check_media_containers.sh
```

Getting Help

1. **Check n8n logs:** `docker compose logs -f n8n`
2. **Check execution history** in n8n UI
3. **n8n Community Forum:** https://community.n8n.io/ (https://community.n8n.io/)
4. **GitHub Issues:** Create an issue in your repository with details

---

Next Steps

Phase 1 Complete! 🎉

You now have:
- ✅ n8n running and monitoring your containers
- ✅ Automated health checks every 5 minutes
- ✅ Uptime Kuma integration
- ✅ Telegram notifications
- ✅ Version-controlled automation

What's Next?

Phase 2: Backup Automation
- Migrate `backup_media_stack.sh` to n8n
- Add backup verification
- Enhance notifications with backup status

Phase 3: Docker Updates
- Migrate `docker_weekly_pull.sh` to n8n
- Add version tracking
- Create changelog notifications

Phase 4: Advanced Features
- Resource monitoring (CPU, disk, memory)
- Certificate expiration tracking
- Automatic remediation workflows
- Custom dashboards

Regular Maintenance

1. **Weekly:** Check n8n execution history
2. **Monthly:** Review and update credentials

3. **As needed:** Update Docker images:
   ```
   bash
     cd /home/ubuntu/homelab-automation/n8n
     docker compose pull
     docker compose up -d
   ```

Keep Learning

- **n8n Documentation:** https://docs.n8n.io/ (https://docs.n8n.io/)
- **n8n Templates:** https://n8n.io/workflows/ (https://n8n.io/workflows/)
- **Docker Compose Docs:** https://docs.docker.com/compose/ (https://docs.docker.com/compose/)

Summary Checklist

- [ ] Git installed and configured
- [ ] GitHub account created
- [ ] SSH keys set up for GitHub
- [ ] Repository created and code pushed
- [ ] n8n installed and accessible
- [ ] Telegram bot created
- [ ] n8n credentials configured (SSH + Telegram)
- [ ] Workflow imported and configured
- [ ] Workflow tested (manual execution)
- [ ] Failure notification tested
- [ ] Old cron job disabled
- [ ] Workflow activated
- [ ] First scheduled execution verified

Congratulations on completing Phase 1! 🚀

Your home lab automation is now more maintainable, observable, and scalable. Happy automating!

Document Version: 1.0
Last Updated: December 2024
Author: Home Lab Administrator
Feedback: Please create an issue on GitHub with suggestions or corrections