The Ultimate PiHole (Part 2)

Automating Tasks Within PiHole

Now that we have our Pi-hole effectively blocking those pesky ads and keeping our traffic private with `unbound`, we can begin automating routine tasks to ensure optimal performance and reliability. By leveraging `cron` jobs, we can schedule various maintenance activities and monitoring tasks that will help keep our Pi-hole running smoothly.

Benefits of Automation

1. **Consistent Updates**: Automating updates for Pi-hole and the host system ensures you run the latest versions with security patches, reducing vulnerabilities.
2. **Health Monitoring**: Scheduled health checks can alert you via Telegram if the Pi-hole service goes down, allowing for quick remediation.
3. **Data Management**: Regular backups of Pi-hole configurations and logs protect against data loss and provide historical data for analysis.
4. **Resource Optimization**: Scheduling tasks during off-peak hours minimizes the impact on system performance.

Implementing Cron Jobs

1. **Creating Bash Scripts**: Write scripts to automate tasks like updating Pi-hole, checking service status, or backing up logs.
2. **Scheduling with Cron**: Use the `crontab` command to define how often each task runs—hourly, daily, weekly, or monthly—based on its importance.
3. **Notifications**: Integrate Telegram notifications into your scripts to stay informed about updates and service status.

Automating tasks within your Pi-hole environment enhances reliability and performance while keeping you informed about its health. With `cron` jobs in place, your Pi-hole setup operates smoothly, providing a cleaner and more private browsing experience.

Updating PiHole:

To automate Pi-hole updates using a Bash script, you can create a script that runs the update command and then schedule it with `cron`. Here's how to do it:

Step 1: Create the Bash Script

Create a new script file:

```
sudo vim /usr/local/bin/pihole_update.sh
```

Add the following content to the script:

```
#!/bin/bash
pihole update
```

- **Save and exit**

Step 2: Make the Script Executable

Run the following command to make the script executable:

```
sudo chmod +x /usr/local/bin/pihole_update.sh
```

Step 3: Schedule the Script with Cron

Edit the crontab:

```
sudo crontab -e
```

Add the following line to the crontab file:

```
 0 0 * * 0 /usr/local/bin/pihole_update.sh
```

- This will run the script every Sunday at 12:00 AM.

Save the changes and **exit**

Step 4: Verification

Check the crontab:

```
sudo crontab -l
```

Optional: Adding Telegram Bot For Notifications

Create a Telegram Bot

1. Download Telegram on your device of choice to receive these notifications
2. Start a chat with BotFather (Press Start)
3. Use the command `/newbot` to create a new bot.
4. Follow the prompts to name your bot and choose a username for it.
   After creation, you will receive a token for your bot.
   - Save this token.

Get Your Chat ID

1. Start a chat with your bot by searching for it in Telegram and sending it a message.

2. Get your chat ID by visiting the following URL in your web browser: (replace `YOUR_BOT_TOKEN` with the token you received from BotFather)

```
https://api.telegram.org/botYOUR_BOT_TOKEN/getUpdates
```

- Look for the `chat` object in the JSON response. Your chat ID will be labeled as `"id"` within the `chat` object.

Update Script

- Update the script to notify the Telegram Bot when the update has completed, like so:

```
#!/bin/bash

# Update Pi-hole
pihole update

# Send message to Telegram
TELEGRAM_BOT_TOKEN="YOUR_BOT_TOKEN"
CHAT_ID="YOUR_CHAT_ID"
MESSAGE="Pi-hole updated on $(date)"
curl -s -X POST
"https://api.telegram.org/bot$TELEGRAM_BOT_TOKEN/sendMessage" -d
"chat_id=$CHAT_ID&text=$MESSAGE"
```

Automation Continued

Updating Host:

Updating the host itself regularly is also a good idea.

- This can be integrated into the last script or a new one can be made if you'd like it to run at another time.

```
#!/bin/bash

# Update host
sudo apt update && sudo apt upgrade -y

# Send message to Telegram
TELEGRAM_BOT_TOKEN="YOUR_BOT_TOKEN"
CHAT_ID="YOUR_CHAT_ID"
MESSAGE="Host system updated on $(date)"
curl -s -X POST
"https://api.telegram.org/bot$TELEGRAM_BOT_TOKEN/sendMessage" -d
"chat_id=$CHAT_ID&text=$MESSAGE"
```

Updating Blocklists:

To keep your Pi-hole effective, regularly updating blocklists can be very beneficial.

- You can create a script like this to do it for you:

```
#!/bin/bash

# Update Pi-hole blocklists
pihole -g

# Send message to Telegram
TELEGRAM_BOT_TOKEN="YOUR_BOT_TOKEN"
CHAT_ID="YOUR_CHAT_ID"
MESSAGE="Pi-hole blocklists updated on $(date)"
curl -s -X POST
"https://api.telegram.org/bot$TELEGRAM_BOT_TOKEN/sendMessage" -d
"chat_id=$CHAT_ID&text=$MESSAGE"
```

Disk Usage Monitor:

Monitoring disk usage can help prevent issues related to low disk space.

- Here’s a simple script that will alert you when it passes a custom threshold:

```
#!/bin/bash

# Check disk usage
THRESHOLD=80  # Set the threshold percentage
USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

if [ "$USAGE" -gt "$THRESHOLD" ]; then
    # Send alert to Telegram
    TELEGRAM_BOT_TOKEN="YOUR_BOT_TOKEN"
    CHAT_ID="YOUR_CHAT_ID"
    MESSAGE="Disk usage is at ${USAGE}%isk usage is at ${USAGE}%.
Please check your system!"
    curl -s -X POST
"https://api.telegram.org/bot$TELEGRAM_BOT_TOKEN/sendMessage" -d
"chat_id=$CHAT_ID&text=$MESSAGE"
fi
```

Pi-Hole Backup:

Creating regular backups of your Pi-hole configuration is essential for recovery in case of data loss.

- Here’s a backup script that I run on the first of every month:

```
#!/bin/bash

# Backup Pi-hole
BACKUP_DIR="/path/to/backup"  # Change this to your backup directory
TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
FILENAME="pihole_backup_$TIMESTAMP.tar.gz"

# Create backup
pihole -a -t > "$BACKUP_DIR/$FILENAME"

# Send message to Telegram
TELEGRAM_BOT_TOKEN="YOUR_BOT_TOKEN"
```

```
CHAT_ID="YOUR_CHAT_ID"
MESSAGE="Pi-hole backup created: $FILENAME on $(date)"
curl -s -X POST
"https://api.telegram.org/bot$TELEGRAM_BOT_TOKEN/sendMessage" -d
"chat_id=$CHAT_ID&text=$MESSAGE"
```

In addition to backing up your Pi-hole configuration, you may also want to back up the logs to retain historical data for analysis or troubleshooting.

- Here's an example of how to back up Pi-hole logs:

```
#!/bin/bash

# Backup Pi-hole logs
LOG_BACKUP_DIR="/path/to/log_backup"
TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
LOG_FILENAME="pihole_logs_$TIMESTAMP.tar.gz"

# Create a backup of Pi-hole logs
tar -czf "$LOG_BACKUP_DIR/$LOG_FILENAME" /var/log/pihole.log

# Send message to Telegram
TELEGRAM_BOT_TOKEN="YOUR_BOT_TOKEN"
CHAT_ID="YOUR_CHAT_ID"
MESSAGE="Pi-hole logs backup created: $LOG_FILENAME on $(date)"
curl -s -X POST
"https://api.telegram.org/bot$TELEGRAM_BOT_TOKEN/sendMessage" -d
"chat_id=$CHAT_I
```

Service Monitor:

Monitoring the status of critical services ensures that your Pi-hole remains operational.

- Here's a simple service monitor script that alerts you using the Telegram Bot:

```
#!/bin/bash

# Check if Pi-hole is running
if ! pgrep -x "pihole-FTL" > /dev/null; then
    # Send alert to Telegram
    TELEGRAM_BOT_TOKEN="YOUR_BOT_TOKEN"
    CHAT_ID="YOUR_CHAT_ID"
    MESSAGE="Pi-hole service is down on $(date). Please check!"
    curl -s -X POST
"https://api.telegram.org/bot$TELEGRAM_BOT_TOKEN/sendMessage" -d
"chat_id=$CHAT_ID&text=$MESSAGE"
fi
```

Understanding Crontab Numbering

The following scripts can now be added to the crontab entries. Some of these, like the backup script, you might want to run less frequently than the blocklist updating script. Crontab entries consist of five fields that specify when a command should run. The fields are as follows:

1. **Minute (0-59)**: The minute of the hour when the command will run.
2. **Hour (0-23)**: The hour of the day when the command will run (24-hour format).
3. **Day of the Month (1-31)**: The day of the month when the command will run.
4. **Month (1-12)**: The month when the command will run.
5. **Day of the Week (0-7)**: The day of the week when the command will run (0 and 7 both represent Sunday).

Example of Crontab Entry Breakdown

```
0 2 1 * 0 /path/to/script.sh
```

- `0`: At minute 0
- `2`: At hour 2 (2 AM)
- `1`: On the 1st day of the month
- `*`: Every month

- `0`: On Sunday

This entry means the script will run at 2:00 AM on the first day of every month, but only if that day is also a Sunday. If you want it to run every month regardless of the day of the week, you would replace the `0` with `*`.

- This flexibility allows you to schedule tasks precisely according to your needs. Here is how I have mine setup for reference:

```
# m h  dom mon dow   command
0 0 * * 0 /usr/local/bin/pihole_update.sh
0 1 * * 0 /usr/local/bin/radahn_update.sh
0 2 * * 0 /usr/local/bin/disk_mon.sh
0 7 * * * /usr/local/bin/blocklist_update.sh
0 6 * * * /usr/local/bin/pihole_service_mon.sh
0 1 1 * * /usr/local/bin/pihole_backup.sh
```

Automating tasks within your Pi-hole environment significantly enhances its reliability and performance by ensuring essential maintenance activities—such as updates, health monitoring, backups, and log management—are executed consistently through cron jobs. This proactive approach, combined with Telegram notifications, keeps you informed about the system's status, enabling quick responses to any issues that arise. Ultimately, these automated processes not only save you time but also maintain the effectiveness of your ad-blocking solution, ensuring a seamless and secure browsing experience while adapting to the evolving demands of your network.
