Crontab Explainer

Crontab is a powerful utility tool included in the base package of almost all Linux distros. It enables you to automate your workflow by executing commands and scripts as scheduled tasks, aka cron jobs.

In essence, crontab is a command-line interface to the cron daemon (or `crond`), which runs in the background and executes tasks per the schedule specified in the crontab file.

> When adding scripts to cron jobs, especially those that spawn processes or perform lengthy operations, it is crucial to exercise caution. If a script fails to exit properly or if multiple instances of the same script run simultaneously, it could quickly lead to unwanted behaviors. Your automation workflow can be interrupted, or your process limit can be reached, which results in your service becoming inaccessible.
>
> To prevent such issues, ensure your script(s) includes proper exit conditions and, if needed, uses a locking mechanism. The **file lock** method is a simple yet effective solution. This approach ensures that only one instance of the script runs at any given time, preventing conflicts and potential interruption of your service. An example of where the lock file method has been used, can be found here.

The Crontab Format

Before setting up cron jobs, it is essential to understand the syntax of the crontab file. Each line in the crontab file represents a cron job consisting of a schedule and a command. A quick and straightforward way to configure cron schedules is to use crontab.guru.

- The basic structure of a cron job in a crontab file is:

```
* * * * * command_to_execute
```

- This format includes five fields, followed by the command to be executed:
  1. **Minute** (0-59): The minute of the hour the command should run.
  2. **Hour** (0-23): The hour of the day the command should run.
  3. **Day of the month** (1-31): The day of the month the command should run.
  4. **Month** (1-12): The month the command should run.
  5. **Day of the week** (0-7): The day of the week the command should run. Both 0 and 7 represent Sunday.

For example, to run a command at 4:30 AM every day, the cron job would look like this:

```
30 4 * * * command_to_execute
```

- The schedule is on the system timezone, Central European Time (CET) or UTC+1.
- When entering the command to execute, always use the full path to your script.

Special Characters

Crontab also supports special characters that help in defining more complex schedules:

- `*` **(Asterisk)**: Represents all possible values for a field (e.g., every minute, every hour).
- `,` **(Comma)**: Separates multiple values (e.g., 1,2,3 for the first three minutes).
- `-` **(Hyphen)**: Specifies a range of values (e.g., 1-5 for the first five days of the month).
- `/` **(Slash)**: Specifies step values (e.g., */2 in the minute field means every two minutes).

Configure Cron Jobs

To access the crontab, where you can create or edit cron jobs, connect to your service via SSH. Once connected, you can access the crontab by executing `crontab -e`.

- If this is your first time accessing the crontab, you will be asked to select an editor.

```
ultra@docs:~$ crontab -e
no crontab for ultra - using an empty one


Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/mcedit
  4. /usr/bin/vim.tiny
  5. /bin/ed

Choose 1-5 [1]:
```

- For simplicity, select `nano` by pressing `1` and confirm with `ENTER`.
- Next, you will be presented with the crontab, and you can start creating cron jobs.
  - As previously mentioned, each line in the crontab represents a cron job.
- Below are a couple of cron job examples.

```
1 12,00 * * * /home/username/scripts/myscript.sh >> /home/username/scripts/logs/myscript.log
2>&1
0 1 15,30 * * app-prowlarr upgrade >> /home/username/scripts/logs/prowlarr-upgrade.log 2>&1
0 2 15,30 * * app-jackett restart > /dev/null 2>&1
```

- Here's an explanation of the above cron job examples.
  - The top cron job executes a script called `myscript.sh` and logs the output to a log file called `myscript.log`.
  - The middle cron job executes an upgrade of Prowlarr, and logs the command output to a log file called `prowlarr-upgrade.log`.
  - The bottom cron job executes a restart of Jackett, but instead of piping the command output to a log file, it will be piped to `/dev/null` (the void of nothingness).
- Once you have configured your cron jobs, press `CTRL+X`, then `Y` and `ENTER` to save and exit the crontab.

Revision #7
Created 21 August 2024 14:13:18 by varg
Updated 23 August 2024 07:23:00 by varg