Systemd.Timers

Why and how to use

Jan Baer, LeadDeveloper, CHECK24

17. August 2023

What can you do with Cron jobs

- Run jobs at a specific time for a specific user

What Cron jobs doesn't support

- Remember the state of the last executions
- Run jobs immediately after a missing execution (anachron might help)
- Show you when the next execution will happen
- Run jobs with a randomized delay

What features can you get from Systemd timers

- Run timers with a randomized delay
- Persist state of last execution
- Run job immediately after restart after missing execution
- Better control over environment variables
- Jobs can be set up to depend on other systemd units.
- Run a job once at a specific time

Some helpful Systemd timers commands

- `systemctl list-timers --all`
- `systemctl status backup-to-nas.timer/service`
- `systemctl edit --full backup-to-nas.timer`
- `journalctl -u backup-to-nas.timer/service`

How to create a Systemd timer

Define timer unit

```
[Unit]
Description=Timer backup-to-nas

[Timer]
OnCalendar=hourly
Persistent=True
AccuracySec=15sec
RandomizedDelaySec=15m
Unit=backup-to-nas.service

[Install]
WantedBy=timers.target
```

How to create a Systemd timer

Create the corresponding Systemd service unit

```
[Unit]
Description=Service for timer backup-to-nas

[Service]
Type=oneshot
Environment=JABASOFT_DS_SSH_PORT=26
ExecStart=/home/jan/bin/rsync-to-nas.sh
User=jan
```

Monotonic timers

Start at or after a specific system event

```
OnBootSec=15min
OnUnitActiveSec=1w
```

Multiple definitions can be used

Monotonic timers

Setting	Meaning
OnActiveSec	Defines a timer relative to the moment the timer unit itself is activated.
OnBootSec	Defines a timer relative to when the machine was booted up.
OnStartupSec	Defines a timer relative to when the service manager was first started.
OnUnitActiveSec	Defines a timer relative to when the unit the timer unit is activating was last activated
OnUnitInactiveSec	Defines a timer relative to when the unit the timer unit is activating was last deactivated.

Realtime timers

Starts at a specific time

```
OnCalendar=Mon,Tue *-*-01..04 12:00:00
OnCalendar=Mon..Fri 22:30
OnCalendar=Sat,Sun 20:00
OnCalendar=daily
```

Check you timer definition before running

```
systemd-analyze calendar daily

  Original form: daily
Normalized form: *-*-* 00:00:00
    Next elapse: Thu 2023-08-17 00:00:00 CEST
       (in UTC): Wed 2023-08-16 22:00:00 UTC
       From now: 2h 55min left
```

More info `man systemd.time`

Control the environment variables

Show Systemd environment for current user

```
systemctl -user show-environment
```

To import any variables missing in the Systemd environment, specify the following command at the end of your `~/.bashrc`:

systemctl –user import-environment VARIABLE1 VARIABLE2

Transient timer units

One can use systemd-run to create transient timer units.

```
systemd-run --on-calendar="13:00" /usr/local/bin/backup.sh
```

Debugging timers

- You can test the service, that should run with the timer also manually with

```
systemctl start myservice.service
```

- You can check the result with

```
journalctl -u myservice.service
```

Personal timers

Timers can be configured from every user, but you need to check, if the process /usr/lib/systemd/systemd –user is running for the current user. The timer definitions has to be located in ~/.config/systemd/user To activate such a timer, you need to use the –user Parameter

```
systemctl --user daemon-reload
systemctl --user enable --now backup.timer
systemctl --user reenable --now backup.timer
systemctl --user start backup.timer
```

Change existing timers

Customize an existing timer with

```
systemctl edit backup.timer
```

Will create the file

`/etc/systemd/system/backup-to-nas.service.d/override.conf`

Important settings

AccuracySec

Specify the accuracy the timer shall elapse with. Defaults to 1min. The timer is scheduled to elapse within a time window starting with the time specified in OnCalendar=, OnActiveSec=, OnBootSec=, OnStartupSec=, OnUnitActiveSec= or OnUnitInactiveSec= and ending the time configured with AccuracySec= later. Within this time window, the expiry time will be placed at a host-specific, randomized, but stable position that is synchronized between all local timer units. This is done in order to optimize power consumption to suppress unnecessary CPU wake-ups. To get best accuracy, set this option to 1us. Note that the timer is still subject to the timer slack configured via systemd-system.conf(5)'s TimerSlackNSec= setting.

Important settings

RandomizedDelaySec

Delay the timer by a randomly selected, evenly distributed amount of time between 0 and the specified time value. Defaults to 0, indicating that no randomized delay shall be applied. Each timer unit will determine this delay randomly before each iteration, and the delay will simply be added on top of the next determined elapsing time, unless modified with FixedRandomDelay=, see below.

This setting is useful to stretch dispatching of similarly configured timer events over a certain time interval, to prevent them from firing all at the same time, possibly resulting in resource congestion.

Links

- https://opensource.com/article/20/7/systemd-calendar-timespans
- https://documentation.suse.com/smart/systems-management/html/systemd-working-with-timers/index.html