Systemd

Stefan Hornburg (Racke)

Contents

Units

Rules

`%` and `$` need to escape as `%%` and `$$`:

```
ExecStart=/bin/bash -c '/home/{{ shopify_api_user }}/shopify-api/getorders.pl --filter created_at_mi
```

List units

```
$ systemctl list-units
```

Filter by type:

```
$ systemctl list-units --type=socket
```

Filter by pattern:

```
$ systemctl list-units sympa*
```

Failed units:

```
$ systemctl --failed
```

It is recommended to examine why these units failed, but in some case these are old or irrelevant errors.

You can also filter the lists of failed units:

```
$ systemctl list-units --failed 'check_mk*'
```

You can remove them for the list individually:

```
$ systemctl reset-failed interchange
```

Or all of them:

```
$ systemctl reset-failed
```

Display unit

As there might be more than file involved, the following command is handy:

```
$ systemctl cat sympa
```

Editing units

Unit files can be edited through *systemctl* commands.

```
$ systemctl edit mybackup.service
$systemctl edit --full mybackup.service
systemctl edit --full --force mybackup.service
```

Unit properties

This shows the properties of a currently running unit:

```
$ systemctl show ssh
```

You can query a specific property:

```
$ systemctl show --property=EnvironmentFiles ssh
EnvironmentFiles=/etc/default/ssh (ignore_errors=yes)
```

This relates to the following line in the unit file:

```
EnvironmentFile=-/etc/default/ssh
```

Drop-in units

List all drop-ins:

```
~# systemd-delta --type=extended
[EXTENDED]   /etc/systemd/system/check_mk.socket → /etc/systemd/system/check_mk.socket.d/10-ipacl.co
[EXTENDED]   /usr/lib/systemd/system/rc-local.service → /usr/lib/systemd/system/rc-local.service.d/d
[EXTENDED]   /usr/lib/systemd/system/systemd-resolved.service → /usr/lib/systemd/system/systemd-reso
[EXTENDED]   /usr/lib/systemd/system/systemd-timesyncd.service → /usr/lib/systemd/system/systemd-tim
```

See also: Using systemd drop-in units

Example: Reset StandardError

Unit directives

ExecStartPre

Commands to execute before the main process started.
This can be used to test the configuration:

```
ExecStartPre=/usr/sbin/nginx -t -q -g 'daemon on; master_process on;'
```

Adding a plus sign in front of the command indicates that the root user is executing this command:

```
ExecStartPre=+/bin/mkdir -p /run/sympa
ExecStartPre=+/bin/chown sympa:sympa /run/sympa
```

Group

User group used when running the service.

```
Group sympa
```

PermissionsStartOnly

This directive is deprecated. It was used to run directives like *ExecStartPre* as privileged (root) user when your service is running as a non privileged user.
Instead of using this directive you can prepend a plus sign to the command, which indicates that the root user is executing this command.

```
ExecStartPre=+/bin/mkdir -p /run/sympa
ExecStartPre=+/bin/chown sympa:sympa /run/sympa
```

Restart

Restart policy for the unit.
For debugging a failed unit it might make sense to turn off restarting:

```
Restart: no
```

RuntimeDirectory

This directive creates a directory in the default location (usually `/run`):

```
RuntimeDirectory sympa
```

If multiple related services are using the same directory it is important that only one service declares the runtime directory.
The other service units should declare *Requires* and *After* on the ”base” service.

User

User running the service.

```
User sympa
```

WorkingDirectory

This configures the working directory for the service, equivalent to `cd` in a shell.

```
WorkingDirectory=/home/sympa
```

Timers

Timers are the equivalents of cron jobs. They consist of a *service* unit which defines the command to be executed and a *timer* unit which defines the schedule.

Pros and Cons

Advantages of using timers instead of cron jobs:

- failed timers can be listed and monitored
- output is automatically logged
- timer is not executed if an instance is already running
- easier to locate compared to the various files and directories with cron jobs
- run job out of schedule without remembering the exact commandline
- memory and CPU limits
- precision is one second

Disadvantages:

- takes more time to set them up
- less control for users' jobs

Undecided:
Timers do not send automatically a notification on failed jobs.
With systemd timers ensure that your scripts provide a proper exit code on errors instead of relying on messages going to standard error output.

Activation

The timer needs to be enabled and started to be active:

```
$ systemctl enable certbot.timer
$ systemctl start certbot.timer
```

Display active timers:

```
$ systemctl list-timers
```

Display all timers:

```
$ systemctl list-timers --all
```

Schedule

You can always run the job manually independent with of the current schedule with:

```
$ systemctl start certbot.service
```

Every 20 minutes:

```
OnCalendar=*:0/20
```

Every even hour (0:00, 2:00, ...)

```
OnCalendar=0/2:00:00
```

Every odd hour (1:00, 3:00, ...)

```
OnCalendar=1/2:00:00
```

Once a day at midnight:

```
OnCalendar=daily
```

Once a day at 15:44:

```
OnCalendar=15:44
```

Monday noon:

```
OnCalendar=Mon 12:00
```

Start corresponding service immediately if last time was missed (e.g. system was down):

```
Persistent=true
```

Certbot example

Certbot service unit:

```
$ systemctl cat certbot.service
# /lib/systemd/system/certbot.service
[Unit]
Description=Certbot
Documentation=file:///usr/share/doc/python-certbot-doc/html/index.html
Documentation=https://letsencrypt.readthedocs.io/en/latest/
[Service]
Type=oneshot
ExecStart=/usr/bin/certbot -q renew
PrivateTmp=true
```

Certbot timer unit:

```
$ systemctl cat certbot.timer
# /lib/systemd/system/certbot.timer
[Unit]
Description=Run certbot twice daily

[Timer]
OnCalendar=*-*-* 00,12:00:00
RandomizedDelaySec=43200
Persistent=true

[Install]
WantedBy=timers.target
```

Certbot timer status:

```
$ systemctl status certbot.timer
● certbot.timer - Run certbot twice daily
   Loaded: loaded (/lib/systemd/system/certbot.timer; enabled; vendor preset: enabled)
   Active: active (waiting) since Wed 2019-04-03 08:00:27 CEST; 2 months 13 days ago
  Trigger: Mon 2019-06-17 10:36:39 CEST; 18h left
```

Testing

Testing *OnCalendar* settings:

```
$ systemd-analyze calendar $(systemctl cat myservice.timer | sed -n 's/^OnCalendar=//p')
  Original form: *:15:44
Normalized form: *-*-* *:15:44
    Next elapse: Sun 2020-01-05 18:15:44 CET
       (in UTC): Sun 2020-01-05 17:15:44 UTC
       From now: 45min left
```

Resources

archlinux Wiki `https://wiki.archlinux.org/index.php/Systemd/Timers`

Hostname

Set hostname:

```
$ hostnamectl set-hostname buster-test-box
```

This is not persistent on Ubuntu. In order to change this, you need to edit `/etc/cloud/cloud.cfg`:

```
preserve_hostname: true
```

Targets

Show dependencies for a target:

```
$ systemctl list-dependencies network-online.target
network-online.target
● ├─networking.service
● └─NetworkManager-wait-online.service
```

DNS resolver

Show status of DNS resolver:

```
$ sudo resolvectl status
```

Time

Synchronization (NTP)

Configuration file: `/etc/systemd/timesyncd.conf`

Specific NTP servers (e.g. Active Domain Controllers) can be added in the NTP= line, separated by whitespace. Fallback NTP servers are configured in the FallbackNTP= line, for example

```
[Time]
NTP=
FallbackNTP=0.debian.pool.ntp.org 1.debian.pool.ntp.org 2.debian.pool.ntp.org 3.debian.pool.ntp.org
```

Restart `systemd-timesyncd` service after changing the configuration:

```
$ sudo systemctl restart  systemd-timesyncd
```

Timezone

Show time and date details:

```
$ timedatectl
...
```

Change timezone:

```
$ timedatectl set-timezone Europe/Berlin
```

IP access control

This feature was introduced with systemd 235, which is available in recent Linux distributions like Debian Buster, Ubuntu Bionic and CentOS 8.

The following example comes from a check_mk monitoring agent which is listening on port 6556. To restrict access to the monitoring server we add the following drop-in (e.g. `etc/systemd/system/check_mk.socket.d/10-ipacl.conf`):

```
[Socket]
IPAddressDeny=any
IPAddressAllow=203.0.113.114
```

Resources

IP Accounting and Access Lists with systemd

Logging (journalctl)

Messages since last reboot

```
$ sudo journalctl -b
```

Show kernel messages

This equivalent to `dmesg -T`.

```
$ sudo journalctl -k
```

Line wrapping

Long log lines are not wrapped as with *less*, which makes it hard to see the complete message on the console.

Example to fix that:

```
$ systemctl status fail2ban.service --no-pager --full
```

You can change this behaviour with the *SYSTEMD_PAGER* environment variable:

```
$ export SYSTEMD_PAGER="less -r"
```

Vacuum logs

Retain only the past four days:

```
$ sudo journalctl --vacuum-time=4d
```

Retain only the past 750 MB:

```
$ sudo journalctl --vacuum-size=750M
```

Templates

Example: `/lib/systemd/system/postgresql@.service` on Debian for PostgreSQL cluster.

OOM

```
# prevent OOM killer from choosing the postmaster (individual backends will
# reset the score to 0)
OOMScoreAdjust=-900
```

Troubleshooting

```
● mariadb.service - MariaDB 10.3.22 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
     Active: failed (Result: exit-code) since Wed 2020-03-25 11:32:09 CET; 1h 1min ago
       Docs: man:mysqld(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 638935 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysqld (code=e

Mar 25 11:32:09 belukha systemd[1]: Starting MariaDB 10.3.22 database server...
Mar 25 11:32:09 belukha systemd[638935]: mariadb.service: Failed to determine user credentials: No s
Mar 25 11:32:09 belukha systemd[638935]: mariadb.service: Failed at step USER spawning /usr/bin/inst
Mar 25 11:32:09 belukha systemd[1]: mariadb.service: Control process exited, code=exited, status=217
Mar 25 11:32:09 belukha systemd[1]: mariadb.service: Failed with result 'exit-code'.
Mar 25 11:32:09 belukha systemd[1]: Failed to start MariaDB 10.3.22 database server.
```

Fixed by:

```
systemctl daemon-reexec
systemctl start mariadb
```

Old init scripts

Run them without systemd stepping on its toes:

```
export _SYSTEMCTL_SKIP_REDIRECT=1
/etc/init.d/sympa start
```

Linuxia Wiki

Stefan Hornburg (Racke)
Systemd

wiki.linuxia.de