Systemd

Introduction

systemd is a software suite that provides an array of system components for Linux operating systems. Its main aim is to unify service configuration and behavior across Linux distributions; systemd's primary component is a "system and service manager"—an init system used to bootstrap user space and manage user processes.

Documentation

- systemd.unit
- systemd.service

Common Parameters

Unit

`Description`	A short description of the unit.
`Documentation`	A list of URIs referencing documentation.
`Before`, `After`	The order in which units are started.
`Requires`	If this unit gets activated, the units listed here will be activated as well. If one of the other units gets deactivated or fails, this unit will be deactivated.
`Wants`	Configures weaker dependencies than `Requires`. If any of the listed units does not start successfully, it has no impact on the unit activation. This is the recommended way to establish custom unit dependencies.
`Conflicts`	If a unit has a `Conflicts` setting on another unit, starting the former will stop the latter and vice versa.

Get a complete list of parameters by running `man systemd.unit`

Install

`Alias`	A space-separated list of additional names for the unit. Most `systemctl` commands, excluding `systemctl enable`, can use aliases instead of the actual unit name.
`RequiredBy`, `WantedBy`	The current service will be started when the listed services are started. See the description of `Wants` and `Requires` in the `[Unit]` section for details.
`Also`	Specifies a list of units to be enabled or disabled along with this unit when a user runs `systemctl enable` or `systemctl disable`.

Get a complete list of parameters by running `man systemd.unit`

Service

`Type`	Configures the process start-up type. One of: • `simple` (default) – starts the service immediately. It is expected that the main process of the service is defined in `ExecStart`. • `forking` – considers the service started up once the process forks and the parent has exited. • `oneshot` – similar to `simple`, but it is expected that the process has to exit before systemd starts follow-up units (useful for scripts that do a single job and then exit). You may want to set `RemainAfterExit=yes` as well so that systemd still considers the service as active after the process has exited. • `dbus` – similar to `simple`, but considers the service started up when the main process gains a D-Bus name. • `notify` – similar to `simple`, but considers the service started up only after it sends a special signal to systemd. • `idle` – similar to `simple`, but the actual execution of the service binary is delayed until all jobs are finished.
`ExecStart`	Commands with arguments to execute when the service is started. `Type=oneshot` enables specifying multiple custom commands that are then executed sequentially. `ExecStartPre` and `ExecStartPost` specify custom commands to be executed before and after `ExecStart`.

`ExecStop`	Commands to execute to stop the service started via `ExecStart`.
`ExecReload`	Commands to execute to trigger a configuration reload in the service.
`Restart`	With this option enabled, the service shall be restarted when the service process exits, is killed, or a timeout is reached with the exception of a normal stop by the `systemctl stop` command.
`RemainAfterExit`	If set to `True`, the service is considered active even when all its processes exited. Useful with `Type=oneshot`. Default value is `False`.

Get a complete list of parameters by running `man systemd.service`

Example

```
[Unit]
Description=The NGINX HTTP and reverse proxy server
After=syslog.target network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t
ExecStart=/usr/sbin/nginx
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s QUIT $MAINPID
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

```
[Unit]
Description=The Apache HTTP Server
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/httpd
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
```

```
ExecReload=/usr/sbin/httpd $OPTIONS -k graceful
ExecStop=/bin/kill -WINCH ${MAINPID}
KillSignal=SIGCONT
PrivateTmp=true

[Install]
WantedBy=multi-user.target
[Unit]
Description=Redis persistent key-value database
After=network.target
```

```
[Service]
ExecStart=/usr/bin/redis-server /etc/redis.conf --daemonize no
ExecStop=/usr/bin/redis-shutdown
User=redis
Group=redis

[Install]
WantedBy=multi-user.target
```

source: shellhacks.com

Revision #3
Created 2021-03-01 02:26:49 UTC by dustin@swigg.net
Updated 2021-11-08 13:21:28 UTC by dustin@swigg.net