systemd Services and logs

START HERE

`systemctl` **`status`** *`<unit>`*
State and recent log

`systemctl` **`--failed`**
What is broken?

`sudo systemctl start` *`<unit>`*
Start now

`journalctl -u` *`<unit>`* **`-f`**
Follow a unit's log

`sudo systemctl restart` *`<unit>`*
Stop, then start

`journalctl -b -p err`
Errors since boot

`sudo systemctl enable --now` *`<unit>`*
Start now and at boot

`sudo systemctl daemon-reload`
Re-read unit files

Unit file /etc/systemd/system → `daemon-reload` → **Loaded** known to systemd → `start` → **Active** running → `enable` → **Boot target** starts at boot

CONCEPTS

A unit is anything systemd manages: a service, timer, socket, target or mount. The .service suffix is optional on the command line.

enable only controls start at boot; start only controls now. Use enable --now for both.

Edit units in /etc/systemd/system (or via systemctl edit); /usr/lib/systemd/system belongs to the package and gets overwritten.

CONTROL SERVICES

`systemctl` **`start`** *`<unit>`*
Start a unit now
System units need root; add sudo.

`systemctl` **`stop`** *`<unit>`*
Stop a unit

`systemctl` **`restart`** *`<unit>`*
Stop and start again

`systemctl` **`reload`** *`<unit>`*
Re-read config without restart
Only works if the service defines a reload action.

`systemctl` **`reload-or-restart`** *`<unit>`*
Reload, else restart

`systemctl` **`try-restart`** *`<unit>`*
Restart only if running

`systemctl` **`kill -s`** *`<signal>`* *`<unit>`*
Signal a unit's processes

`systemctl` **`reset-failed`** *`<unit>`*
Clear the failed state

`systemctl` **`--user`** *`<command>`*
Manage your own user units

INSPECT

`systemctl` **`status`** *`<unit>`*
State, PID and last log lines

`systemctl` **`is-active`** *`<unit>`*
Print active state, exit code reflects it

`systemctl` **`is-enabled`** *`<unit>`*
Will it start at boot?

`systemctl` **`list-units --type=service`**
Loaded services

`systemctl` **`--failed`**
List failed units

INSPECT (cont.)

`systemctl` **`list-unit-files`**
All installed unit files with state

`systemctl` **`cat`** *`<unit>`*
Show unit file plus drop-ins

`systemctl` **`show -p`** *`<property>`* *`<unit>`*
Print one property

`systemctl` **`list-dependencies`** *`<unit>`*
Tree of required units

BOOT AND UNIT FILES

`systemctl` **`enable`** *`<unit>`*
Start at boot

`systemctl` **`disable`** *`<unit>`*
Do not start at boot

`systemctl` **`enable --now`** *`<unit>`*
Enable and start in one go

`systemctl` **`disable --now`** *`<unit>`*
Disable and stop in one go

`systemctl` **`mask`** *`<unit>`*
Block a unit from starting at all
Links the unit to /dev/null; undo with unmask.

`systemctl` **`unmask`** *`<unit>`*
Undo mask

`systemctl` **`daemon-reload`**
Re-read changed unit files

`systemctl` **`edit`** *`<unit>`*
Override settings in a drop-in

`systemctl` **`edit --full`** *`<unit>`*
Edit a full copy of the unit

`systemctl` **`reboot`**
Restart the machine

`systemctl` **`poweroff`**
Shut the machine down

LOGS (JOURNALCTL)

`journalctl -u` *`<unit>`*
Logs of one unit

`journalctl -u` *`<unit>`* **`-f`**
Follow new lines

`journalctl -n 100`
Last 100 entries

`journalctl -b`
Current boot only

`journalctl -b -1`
Previous boot

`journalctl -p err`
Errors and worse

LOGS (JOURNALCTL) (cont.)

`journalctl --since "`*`<date>`***`" --until "`***`<date>`***`"`**
Time window
Accepts values like yesterday, "1 hour ago" or "2026-10-08 12:00".

`journalctl -g` *`<regex>`*
Filter messages by pattern

`journalctl -e`
Open pager at the end

`journalctl -r`
Newest first

`journalctl -k`
Kernel messages

UNIT FILE BASICS

Minimal service

```
# /etc/systemd/system/myapp.service
[Unit]
Description=My app
After=network-online.target
Wants=network-online.target

[Service]
ExecStart=/usr/local/bin/myapp --serve
Restart=on-failure
User=myapp

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

Common directives

Directive	Meaning
`ExecStart=`	Command to run (absolute path)
`Restart=`	on-failure, always, no
`After= / Before=`	Ordering only
`Wants= / Requires=`	Pull in other units (weak / hard)
`WantedBy=`	Target that enables it
`Environment=`	Set VAR=value
`WorkingDirectory=`	Start directory
`Type=`	simple, exec, forking, oneshot, notify

Drop-in override

```
# systemctl edit myapp creates:
# /etc/systemd/system/myapp.service.d/override.conf
[Service]
Environment=LOG_LEVEL=debug
# clear a list setting before replacing it
ExecStart=
ExecStart=/usr/local/bin/myapp --debug
```

`systemd-analyze verify` *`<file>`*
Lint a unit file

TIMERS AND ONE-OFFS

`systemctl` **`list-timers`**
Timers and next/last run

`systemctl` **`list-timers --all`**
Include inactive timers

`systemd-analyze calendar "`*`<expr>`***`"`**
Check an OnCalendar expression

`systemd-run --user --on-active=`*`<time>`* *`<cmd>`**
Run a command later, once

`systemd-run --unit=`*`<name>`* *`<cmd>`**
Run a command as a transient service

OnCalendar examples

Expression	Fires
`daily`	Every midnight
`Mon *-*-* 09:00`	Mondays 09:00
`*-*-* *:00/15`	Every 15 min
`weekly`	Monday 00:00

Enable the .timer, not the .service. A timer starts the same-named service unless Unit= says otherwise.

BOOT ANALYSIS

`systemd-analyze`
Total boot time

`systemd-analyze blame`
Slowest units at boot

RECIPES

Install and start a new service

```
sudoedit /etc/systemd/system/myapp.service
systemd-analyze verify /etc/systemd/system/myapp.service
sudo systemctl daemon-reload
sudo systemctl enable --now myapp
systemctl status myapp
```

Run a script every night

```
# backup.service: [Service] Type=oneshot, ExecStart=/usr/local/bin/backup
# backup.timer:
[Timer]
OnCalendar=*-*-* 02:30
Persistent=true
[Install]
WantedBy=timers.target
# then:
sudo systemctl enable --now backup.timer
```

Why did it fail?

```
systemctl --failed
systemctl status <unit>
journalctl -u <unit> -b -e
journalctl -u <unit> --since "10 min ago" -p warning
```