THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

Booting Up: A Fresh Look at the Modern Init.

Speaker: Antra Purohit - Microsoft
Hemant Bharadwaj - Microsoft

Who this talk is for ?

- AUDIENCE: new sysadmins,students, the curious
- NOT for: systemd internals contributors, kernel devs
- TAKEAWAY 1: The unit hierarchy — .service, .timer, .mount, .socket, .target
- TAKEAWAY 2: How targets replaced runlevels in the boot process
- TAKEAWAY 3: systemctl + journalctl muscle memory
- TAKEAWAY 4: Write your first unit file from scratch

What Systemd actually is ?

More than an init, systemd is a SUITE

systemd (PID 1) — service manager

systemd-journald — structured logging

systemd-logind — user sessions, /var/run/user/UID

systemd-networkd — declarative networking (optional)

systemd-resolved — DNS caching & resolution

systemd-timesyncd — lightweight NTP client

systemd-udevd — device hotplug & naming

systemd-tmpfiles, systemd-sysusers, systemd-boot, ...

The core mental model

systemd manages UNITS — and everything is a unit

.service — long-running daemons & one-shot tasks

.socket — IPC / network sockets that activate services

.timer — scheduled jobs (the cron replacement)

.mount — filesystem mount points

.automount — on-demand mount points

.target — grouping units (the runlevel replacement)

.path — file/directory watchers

.slice / .scope — cgroup resource grouping

.device — kernel-discovered hardware

.swap — swap files / partitions

Anatomy of a unit file

INI-style, three sections

```
[Unit]    – metadata, dependencies, ordering
[Service] – how to run it (only for .service files)
[Install] – how to enable it on boot
```

Search path (highest precedence first):

1. `/etc/systemd/system/` – admin overrides (YOU live here)
2. `/run/systemd/system/` – runtime, lost on reboot
3. `/usr/lib/systemd/system/` – vendor / package defaults

.service : long-running daemons

```
[Unit]
Description=My web app
After=network.target

[Service]
Type=simple        # simple | forking | oneshot | notify | dbus | idle
ExecStart=/usr/bin/python3 /opt/myapp/app.py
Restart=on-failure/always

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

Parallelism & dependency resolution

```
Wants=    – soft dependency (start it, but I don't care if it fails)
Requires= – hard dependency (if it fails, I fail too)
After=    – ORDERING only (start after, even if not required)
Before=   – inverse of After=
```

systemd builds a DEPENDENCY DAG and starts everything it can in parallel.

.target — the grouping unit

A .target is a synchronization point — a NAMED group of units

Examples shipped with every system:

```
  basic.target        — early boot, mounts, swap
  network.target      — network interfaces are up
  multi-user.target   — full system, no GUI
  graphical.target    — multi-user.target + display manager
Used as a milestone in After= / WantedBy= / RequiredBy=
```

Targets vs. runlevels

The mapping that demystifies it

RUNLEVEL 0 → poweroff.target (stops all units and power off machine)

RUNLEVEL 1 → rescue.target (minimal system, root FS mounted rw. Used for repairs and password resets)

RUNLEVEL 2 → multi-user.target (Full multi-user + networking, CLI only)

RUNLEVEL 3 → multi-user.target (network up, multiple user, no GUI)

RUNLEVEL 4 → multi-user.target (custom .target unit files)

RUNLEVEL 5 → graphical.target (multi-user + GUI)

RUNLEVEL 6 → reboot.target (sends SIGTERM and SIGKILL to all service, sync disk, reboot)

Switch at runtime: systemctl isolate multi-user.target

Set the default: systemctl set-default graphical.target

The verbs you'll use 95% of the time

```
systemctl start    foo.service     # start NOW
systemctl stop     foo.service     # stop NOW
systemctl restart  foo.service     # stop + start
systemctl reload   foo.service     # re-read config without restart
systemctl enable   foo.service     # start on every boot
systemctl disable  foo.service     # don't start on boot
systemctl status   foo.service     # is it running? what did it just say?
systemctl mask     foo.service     # force-disable, even via dependencies
```

systemctl status — deconstructed

```
antrapurohit@antrapurohit-azurelinux-vm:/datadrive$ systemctl status hello-world.service
● hello-world.service - Hello World demo daemon
     Loaded: loaded (/etc/systemd/system/hello-world.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2026-05-08 14:29:18 UTC; 5 days ago
   Main PID: 2018336 (bash)
      Tasks: 2 (limit: 9458)
     Memory: 896.0K
        CPU: 4min 50.007s
     CGroup: /system.slice/hello-world.service
             ├─2018336 bash /usr/local/bin/hello-world
             └─3203651 sleep 10
```

enable vs. start — the #1 confusion

```
start   → run it RIGHT NOW (does nothing on the next boot)
enable  → run it on EVERY BOOT (does nothing right now)

Want both? Use the shortcut:
systemctl enable --now foo.service
```

journalctl — debugging at the speed of grep

Logs with structure

Single binary, indexed log — NOT plain text

Every entry is tagged with metadata:

```
_PID, _UID, _GID, _COMM, _EXE, _SYSTEMD_UNIT, _BOOT_ID, PRIORITY
```

Captures everything: kernel (dmesg), early boot, all services' stdout/stderr, syslog

Replaces grepping across /var/log/messages, /var/log/syslog, /var/log/foo.log, ...

Output it however you want: short, json, json-pretty, cat, export

Filters that save your day

Memorize these

```
journalctl -u hello-world.service       # logs for ONE unit
journalctl -u hello-world -f            # follow (like tail -f)
journalctl -b                     # this boot only
journalctl -b -1                  # the PREVIOUS boot (for crash forensics)
journalctl -p err                 # priority err and above
(emerg|alert|crit|err)
journalctl --since '10 min ago'   # human-friendly time windows
journalctl --since yesterday --until '1 hour ago'
```

Combine them: `journalctl -u hello-world -b -p warning --since '2026-05-04'`

```
Manage size:
  journalctl --disk-usage              # how big is it now?
  journalctl --vacuum-size=500M        # trim to 500 MB
  journalctl --vacuum-time=2weeks      # drop entries older than
2 weeks
```

DIY: writing your first unit file

hello-world.service

Build a tiny daemon that:

- prints 'hello, world' every 10 seconds
- restarts itself if it crashes
- logs to the journal (so we can journalctl it)
- starts automatically on boot

Three artifacts:

1. /usr/local/bin/hello-world ← the script
2. /etc/systemd/system/hello-world.service ← the unit file
3. four systemctl commands ← the lifecycle

Step 1 — the script

```
#!/usr/bin/env bash
set -euo pipefail

while true; do
    echo "hello, world - $(date --iso-8601=seconds)"
    sleep 10
done
```

Step 2 — the unit file

```
[Unit]
Description=Hello World demo daemon
After=network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/hello-world
Restart=on-failure
RestartSec=5

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

Step 3 — the four-command lifecycle

```
# 1. Tell systemd to re-read unit files
$ sudo systemctl daemon-reload

# 2. Enable it on boot AND start it now
$ sudo systemctl enable --now hello-world.service
```

Step 3 — the four-command lifecycle

```
# 3. Confirm it's running
```

```
antrapurohit@antrapurohit-azurelinux-vm:~$  systemctl status hello-world.service
● hello-world.service - Hello World demo daemon
     Loaded: loaded (/etc/systemd/system/hello-world.service; enabled; vendor preset: enabled)
    Drop-In: /etc/systemd/system/hello-world.service.d
             └─override.conf
     Active: active (running) since Thu 2026-05-14 06:45:36 UTC; 2 days ago
   Main PID: 3205644 (bash)
      Tasks: 2 (limit: 9458)
     Memory: 704.0K
        CPU: 2min 25.212s
     CGroup: /system.slice/hello-world.service
             ├─3205644 bash /usr/local/bin/hello-world
             └─3798034 sleep 10

May 17 02:57:06 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T02:57:06+00:00
May 17 02:57:16 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T02:57:16+00:00
May 17 02:57:26 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T02:57:26+00:00
May 17 02:57:36 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T02:57:36+00:00
May 17 02:57:46 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T02:57:46+00:00
May 17 02:57:56 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T02:57:56+00:00
May 17 02:58:06 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T02:58:06+00:00
May 17 02:58:16 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T02:58:16+00:00
May 17 02:58:26 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T02:58:26+00:00
May 17 02:58:33 antrapurohit-azurelinux-vm systemd[1]: /etc/systemd/system/hello-world.service.d/override.conf:1: Assignment outside
```

```
# 4. Watch it work
```

```
antrapurohit@antrapurohit-azurelinux-vm:~$ journalctl -u hello-world.service -f
May 17 03:06:27 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T03:06:27+00:00
May 17 03:06:37 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T03:06:37+00:00
May 17 03:06:47 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T03:06:47+00:00
May 17 03:06:57 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T03:06:57+00:00
May 17 03:07:07 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T03:07:07+00:00
May 17 03:07:17 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T03:07:17+00:00
May 17 03:07:27 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T03:07:27+00:00
May 17 03:07:37 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T03:07:37+00:00
May 17 03:07:47 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T03:07:47+00:00
May 17 03:07:57 antrapurohit-azurelinux-vm hello-world[3205644]: hello, world - 2026-05-17T03:07:57+00:00
```

Step 4 — hardening it

```
[Unit]
Description=Hello World demo daemon
After=network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/hello-world
Restart=on-failure
RestartSec=5
# added hardening
PrivateTmp=yes
ProtectHome=true

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

```
  NAME                                          DESCRIPTION
✗ PrivateNetwork=                               Service has access to the host's network
✗ User=/DynamicUser=                            Service runs as root user
✗ CapabilityBoundingSet=~CAP_SET(UID|GID|PCAP)  Service may change UID/GID identities/capabilities
✗ CapabilityBoundingSet=~CAP_SYS_ADMIN          Service has administrator privileges
✗ CapabilityBoundingSet=~CAP_SYS_PTRACE         Service has ptrace() debugging abilities
✗ RestrictAddressFamilies=~AF_(INET|INET6)      Service may allocate Internet sockets
✗ RestrictNamespaces=~CLONE_NEWUSER             Service may create user namespaces
✗ RestrictAddressFamilies=~…                    Service may allocate exotic sockets
✗ CapabilityBoundingSet=~CAP_(CHOWN|FSETID|SETFCAP) Service may change file ownership/access mode/capabilities unrestricte
✗ CapabilityBoundingSet=~CAP_(DAC_*|FOWNER|IPC_OWNER) Service may override UNIX file/IPC permission checks
✗ CapabilityBoundingSet=~CAP_NET_ADMIN          Service has network configuration privileges
✗ CapabilityBoundingSet=~CAP_SYS_MODULE         Service may load kernel modules
✗ CapabilityBoundingSet=~CAP_SYS_RAWIO          Service has raw I/O access
✗ CapabilityBoundingSet=~CAP_SYS_TIME           Service processes may change the system clock
✗ DeviceAllow=                                  Service has no device ACL
✗ IPAddressDeny=                                Service does not define an IP address allow list
✓ KeyringMode=                                  Service doesn't share key material with other services
✗ NoNewPrivileges=                              Service processes may acquire new privileges
✓ NotifyAccess=                                 Service child processes cannot alter service state
✗ PrivateDevices=                               Service potentially has access to hardware devices
✓ PrivateMounts=                                Service cannot install system mounts
✓ PrivateTmp=                                   Service has no access to other software's temporary files
✗ PrivateUsers=                                 Service has access to other users
✗ ProtectClock=                                 Service may write to the hardware clock or system clock
✗ ProtectControlGroups=                         Service may modify the control group file system
✓ ProtectHome=                                  Service has no access to home directories
✗ ProtectKernelLogs=                            Service may read from or write to the kernel log ring buffer
✗ ProtectKernelModules=                         Service may load or read kernel modules
✗ ProtectKernelTunables=                        Service may alter kernel tunables
✗ ProtectProc=                                  Service has full access to process tree (/proc hidepid=)
✗ ProtectSystem=                                Service has full access to the OS file hierarchy
✗ RestrictAddressFamilies=~AF_PACKET            Service may allocate packet sockets
✗ RestrictSUIDSGID=                             Service may create SUID/SGID files
✗ SystemCallArchitectures=                      Service may execute system calls with all ABIs
✗ SystemCallFilter=~@clock                      Service does not filter system calls
lines 1-36
```

```
#!/usr/bin/env bash
set -euo pipefail
TMP_FILE="/tmp/my_10sec_log_tmp_directory.txt"

while true; do
    echo "hello, world - $(date --iso-8601=seconds)" >> "$TMP_FILE"
    sleep 10
done
```

```
antrapurohit@antrapurohit-azurelinux-vm:~$ sudo ls -lrth /tmp/systemd-private-46ddd55e90744fc1a86be3f69cff4e14-hello-world.service-dA
PxYu/tmp
total 4.0K
-rw-r--r-- 1 root root 129 May 17 04:20 my_10sec_log_tmp_directory.txt
```

```
#!/usr/bin/env bash
set -euo pipefail
OUTPUT_FILE="/home/antrapurohit/my_10sec_log.txt"

while true; do
    echo "hello, world - $(date --iso-8601=seconds)" >> "$OUTPUT_FILE"
    sleep 10
done
```

```
antrapurohit@antrapurohit-azurelinux-vm:~$ journalctl -u hello-world -f
May 17 03:45:52 antrapurohit-azurelinux-vm systemd[1]: Started Hello World demo daemon.
May 17 03:45:52 antrapurohit-azurelinux-vm hello-world[3804001]: /usr/local/bin/hello-world: line 6: /home/antrapurohit/my_10sec_log.
txt: No such file or directory
May 17 03:45:52 antrapurohit-azurelinux-vm systemd[1]: hello-world.service: Main process exited, code=exited, status=1/FAILURE
May 17 03:45:52 antrapurohit-azurelinux-vm systemd[1]: hello-world.service: Failed with result 'exit-code'.
May 17 03:45:57 antrapurohit-azurelinux-vm systemd[1]: hello-world.service: Scheduled restart job, restart counter is at 7.
May 17 03:45:57 antrapurohit-azurelinux-vm systemd[1]: Stopped Hello World demo daemon.
May 17 03:45:57 antrapurohit-azurelinux-vm systemd[1]: Started Hello World demo daemon.
May 17 03:45:57 antrapurohit-azurelinux-vm hello-world[3804011]: /usr/local/bin/hello-world: line 6: /home/antrapurohit/my_10sec_log.
txt: No such file or directory
May 17 03:45:57 antrapurohit-azurelinux-vm systemd[1]: hello-world.service: Main process exited, code=exited, status=1/FAILURE
May 17 03:45:57 antrapurohit-azurelinux-vm systemd[1]: hello-world.service: Failed with result 'exit-code'.
```

Thank you

Questions?