Table of Contents

$ hostnamectl

Beginner

Control system hostname

hostnamectl manages the system hostname. It can set the static hostname (persistent), the pretty hostname (UTF-8 descriptive name), and the transient hostname (temporary).

hostnamectl is the systemd way to manage hostnames, replacing manual editing of /etc/hostname. Changes are permanent and tak...

Options & Flags

`status`	Show hostname and system info
`set-hostname`	Set the hostname
`--static`	Show/set static hostname
`--pretty`	Show/set pretty hostname
`--transient`	Show/set transient hostname

Practical Examples

Example: Show system info

```
$ hostnamectl
Static hostname: webserver01
Operating System: Ubuntu 22.04
Kernel: Linux 5.15.0
Architecture: x86-64
```

Shows hostname, OS, kernel, and architecture.

Example: Set hostname

```
$ sudo hostnamectl set-hostname production-web-01
```

Permanently changes the hostname.

Example: Set pretty hostname

```
$ sudo hostnamectl set-hostname "Production Web Server 01" --pretty
```

Sets a descriptive name (can include spaces and special characters).

Example: Update /etc/hosts

```
$ sudo hostnamectl set-hostname newname && sudo sed -i "s/oldname/newname/g" /etc/hosts
```

Changes hostname and updates /etc/hosts to match.

Tips & Best Practices

Pro Tip: Update /etc/hosts too: After changing hostname, update /etc/hosts to map the new name to 127.0.1.1. This prevents DNS resolution issues.

Note: Three hostname types: Static: persistent across reboots. Pretty: descriptive (UTF-8). Transient: temporary (from DHCP).

Warning: No spaces in static hostname: Static hostname must be a valid DNS label: lowercase, no spaces, hyphens allowed. Use --pretty for descriptive names.

$ journalctl

Beginner

Query and display systemd journal logs

journalctl queries and displays logs from the systemd journal. The journal captures all system logs including kernel messages, service output, and syslog messages in a structured, indexed format.

journalctl provides powerful filtering by service, time range, priority, and more. It replaces readi...

Options & Flags

`-u`	Show logs for specific service
`-f`	Follow (like tail -f)
`-n N`	Show last N lines
`--since`	Show logs since time
`--until`	Show logs until time
`-p`	Filter by priority
`-k`	Show kernel messages
`-b`	Show logs from current boot
`-o json`	Output in JSON format

Practical Examples

Example: Service logs

```
$ journalctl -u nginx --no-pager -n 50
```

Shows last 50 log lines for nginx.

Example: Follow logs

```
$ journalctl -u myapp -f
```

Follows log output in real time (like tail -f).

Example: Errors only

```
$ journalctl -p err -b
```

Shows only error-level messages from current boot.

Example: Time range

```
$ journalctl --since "2024-01-15 02:00" --until "2024-01-15 04:00"
```

Shows logs from a specific time window.

Example: Last hour

```
$ journalctl --since "1 hour ago"
```

Shows all logs from the last hour.

Tips & Best Practices

Pro Tip: --since is very flexible: "1 hour ago", "yesterday", "2024-01-15 03:00", "today". Very useful for "what happened at 3am?" questions.

Note: Priority levels: -p emerg, alert, crit, err, warning, notice, info, debug. -p err shows errors, critical, alert, and emergency.

Warning: Journal may not persist: By default, journal may not survive reboots. Create /var/log/journal/ to enable persistent storage.

$ loginctl

Intermediate

Control the systemd login manager

loginctl manages user login sessions through systemd-logind. It shows active sessions, enables/disables user lingering, and controls seat and user management.

loginctl is useful for managing user sessions on multi-user servers, checking who is logged in, terminating sessions, and enabling servic...

Options & Flags

`list-sessions`	Show active sessions
`list-users`	Show logged-in users
`show-session`	Show session details
`terminate-session`	End a session
`enable-linger`	Allow user services after logout
`disable-linger`	Disable user lingering

Practical Examples

Example: List sessions

```
$ loginctl list-sessions
SESSION  UID   USER   SEAT  TTY\n      1  1000  admin       pts/0
```

Shows all active user sessions.

Example: List users

```
$ loginctl list-users
UID  USER\n1000 admin
```

Shows all logged-in users.

Example: Terminate session

```
$ sudo loginctl terminate-session 3
```

Ends a specific user session.

Example: Enable linger

```
$ sudo loginctl enable-linger deploy
```

Allows the deploy user to run services even when not logged in.

Example: Session details

```
$ loginctl show-session 1
```

Shows detailed information about a session.

Tips & Best Practices

Pro Tip: Enable linger for user services: loginctl enable-linger user allows systemd user services to run after logout. Essential for background services running as non-root.

Note: Session management: loginctl manages sessions at the systemd level. It is more powerful than killing processes manually.

Warning: terminate-session kills everything: Terminating a session kills ALL processes in that session. Save work first.

$ service

Beginner

Run a System V init script (legacy service management)

service is a legacy command for managing System V init scripts. It starts, stops, restarts, and checks the status of services. On modern systemd systems, service acts as a compatibility wrapper for systemctl.

service is simpler than systemctl with a consistent syntax: service NAME ACTION. It sti...

Options & Flags

`start`	Start a service
`stop`	Stop a service
`restart`	Restart a service
`status`	Check service status
`--status-all`	List all services

Practical Examples

Example: Start service

```
$ sudo service nginx start
```

Starts the nginx service.

Example: Check status

```
$ service nginx status
```

Shows if the service is running.

Example: Restart

```
$ sudo service postgresql restart
```

Restarts PostgreSQL.

Example: List all services

```
$ service --status-all
 [ + ]  nginx\n [ + ]  postgresql\n [ - ]  apache2
```

Shows status of all services. [+] running, [-] stopped.

Example: Reload config

```
$ sudo service nginx reload
```

Reloads configuration without full restart.

Tips & Best Practices

Note: Wrapper for systemctl: On systemd systems, service nginx start actually calls systemctl start nginx. Both work.

Pro Tip: Simpler syntax: service name action vs systemctl action name. Some people find service syntax easier to remember.

Warning: Use systemctl for new work: systemctl provides more features: enable/disable, mask, journal access. Use service for compatibility only.

$ systemctl

Beginner

Control the systemd system and service manager

systemctl is the primary command for managing systemd services, the init system used by most modern Linux distributions. It starts, stops, restarts, enables, and checks the status of system services.

systemd manages the entire system lifecycle: services, timers, mounts, targets, and more. system...

Options & Flags

`start`	Start a service
`stop`	Stop a service
`restart`	Restart a service
`reload`	Reload config without restart
`status`	Show service status
`enable`	Enable on boot
`disable`	Disable on boot
`is-active`	Check if running
`list-units`	List active units

Practical Examples

Example: Check service status

```
$ systemctl status nginx
? nginx.service - A high performance web server
   Active: active (running) since Mon 2024-01-15
   Main PID: 1234 (nginx)
```

Shows if nginx is running, enabled, and recent log entries.

Example: Start and enable

```
$ sudo systemctl enable --now nginx
```

Starts nginx AND enables it to start on boot in one command.

Example: Restart service

```
$ sudo systemctl restart php-fpm
```

Stops and starts the PHP-FPM service.

Example: List all services

```
$ systemctl list-units --type=service --state=running
```

Shows all currently running services.

Example: Check if enabled

```
$ systemctl is-enabled nginx
enabled
```

Shows if the service starts on boot.

Tips & Best Practices

Pro Tip: enable --now: systemctl enable --now service starts it immediately AND enables on boot. Saves a separate start command.

Warning: daemon-reload after changes: After modifying a .service file, run systemctl daemon-reload before restart. Otherwise systemd uses the old config.

Note: mask vs disable: disable prevents auto-start. mask prevents ALL starting (even manual). Use mask for dangerous services.

$ systemd-analyze

Intermediate

Analyze system boot-up performance

systemd-analyze provides tools for analyzing and debugging systemd startup performance. It shows boot time, identifies slow services, and visualizes the boot process.

systemd-analyze is invaluable for optimizing boot time - it reveals which services take the longest and which could be started in...

Options & Flags

`(no args)`	Show total boot time
`blame`	Show time per service (slowest first)
`critical-chain`	Show critical boot path
`plot`	Generate SVG boot chart
`security`	Audit service security

Practical Examples

Example: Boot time

```
$ systemd-analyze
Startup finished in 2.5s (kernel) + 8.3s (userspace) = 10.8s
```

Shows total boot time breakdown.

Example: Slowest services

```
$ systemd-analyze blame | head -10
5.2s  NetworkManager-wait-online.service\n3.1s  snapd.service\n1.5s  postgresql.service
```

Shows the 10 slowest services during boot.

Example: Critical chain

```
$ systemd-analyze critical-chain
```

Shows the chain of services that determined boot time.

Example: Boot chart

```
$ systemd-analyze plot > boot.svg
```

Creates a visual timeline of the boot process.

Example: Security audit

```
$ systemd-analyze security nginx
```

Audits the security settings of a service unit.

Tips & Best Practices

Pro Tip: blame for slow boots: systemd-analyze blame shows which services slow down boot. Disable unnecessary ones: systemctl disable slow-service.

Note: critical-chain shows dependencies: The critical chain shows which service is actually blocking boot completion - often different from the slowest individual service.

Warning: NetworkManager-wait-online: This service often tops the blame list. If not needed, mask it: systemctl mask NetworkManager-wait-online.service.

$ timedatectl

Beginner

Control system time and date settings

timedatectl controls system time and date settings. It manages the system clock, timezone, NTP synchronization, and RTC (hardware clock) settings.

timedatectl is the systemd way to manage time. It replaces manual date commands and ntpdate for time management. It can enable/disable automatic time...

Options & Flags

`status`	Show current time settings
`set-timezone`	Set the timezone
`list-timezones`	List available timezones
`set-ntp`	Enable/disable NTP sync
`set-time`	Set time manually

Practical Examples

Example: Show time status

```
$ timedatectl
Local time: Mon 2024-01-15 14:30:00 UTC
Time zone: UTC (UTC, +0000)
NTP synchronized: yes
```

Shows current time, timezone, and NTP status.

Example: Set timezone

```
$ sudo timedatectl set-timezone Europe/Budapest
```

Changes the system timezone.

Example: List timezones

```
$ timedatectl list-timezones | grep Europe
Europe/Amsterdam\nEurope/Berlin\nEurope/Budapest\nEurope/London
```

Shows available European timezones.

Example: Enable NTP

```
$ sudo timedatectl set-ntp true
```

Enables automatic time synchronization.

Example: Set time manually

```
$ sudo timedatectl set-ntp false && sudo timedatectl set-time "2024-01-15 14:30:00"
```

Disables NTP and sets time manually (disable NTP first).

Tips & Best Practices

Pro Tip: Always use NTP: Enable NTP sync: timedatectl set-ntp true. Manual time setting drifts. NTP keeps the clock accurate.

Note: Timezone vs UTC: Servers typically use UTC. timedatectl set-timezone UTC. Workstations use local time.

Warning: Disable NTP before manual time: You must disable NTP (set-ntp false) before setting time manually. NTP will override manual changes.