Managing Services in Linux

- **Asst. Prof. Ashwini Mathur**
- **Reference:** Service management with systemd Lab manual Michal Sekletár msekleta@redhat.com 29 October 2017
- **Linux Administration**
- **Practical Based Learning Along with self Assessment.**

Student

USN :
Name :

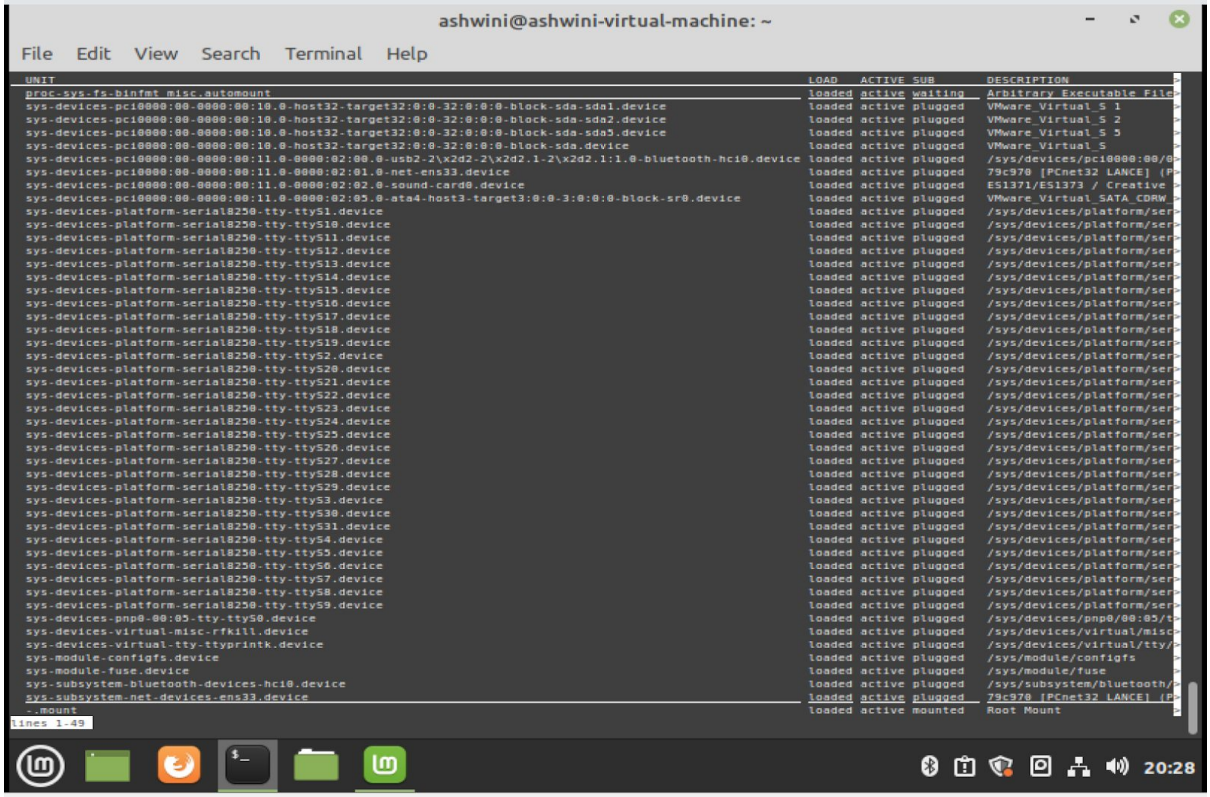
Initial Setup for Practice

There are several options on how to get your hands on the system that is running systemd.

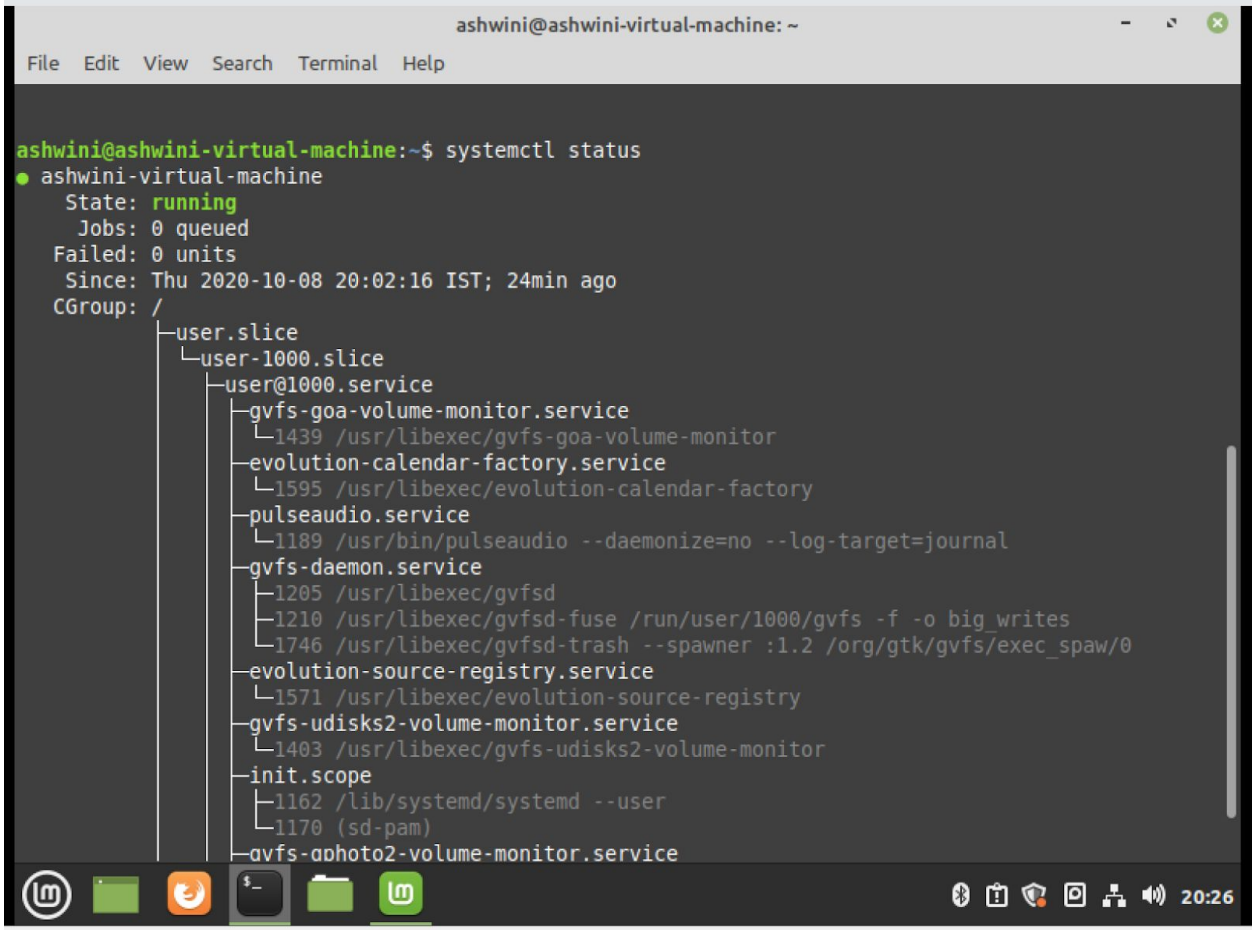
1. **Your own laptop If you are running recent versions of any popular Linux distribution** then you most likely already have systemd on your system.

Run the following command to verify systemd ,

systemctl



systemctl status



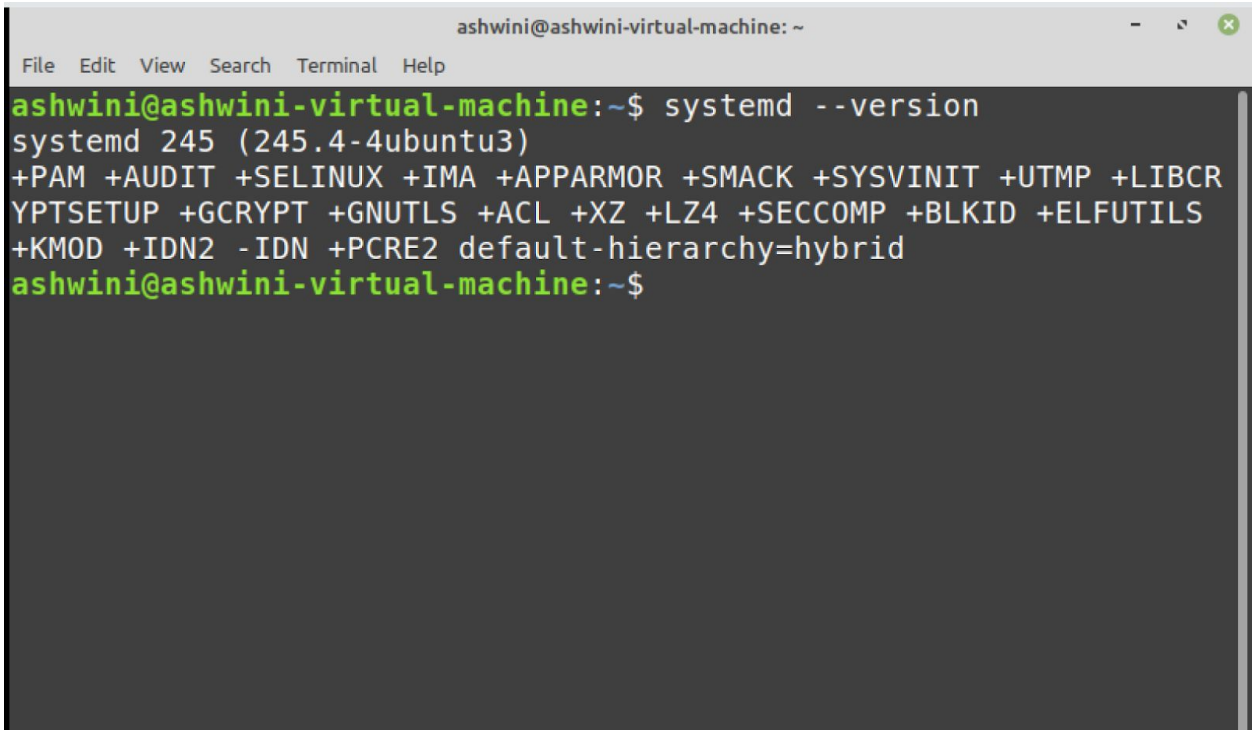
If you get some output back then you have systemd running on your system.

Now, Let's Examine Systemd in the our distro Environment

Aim of this lab is to get familiar with the way how systemd is packaged in your distribution of choice and to figure out what version of systemd you are running.

1. systemd is very actively developed, so more often that not, it is useful to know what package version3 you are running,

systemctl --version

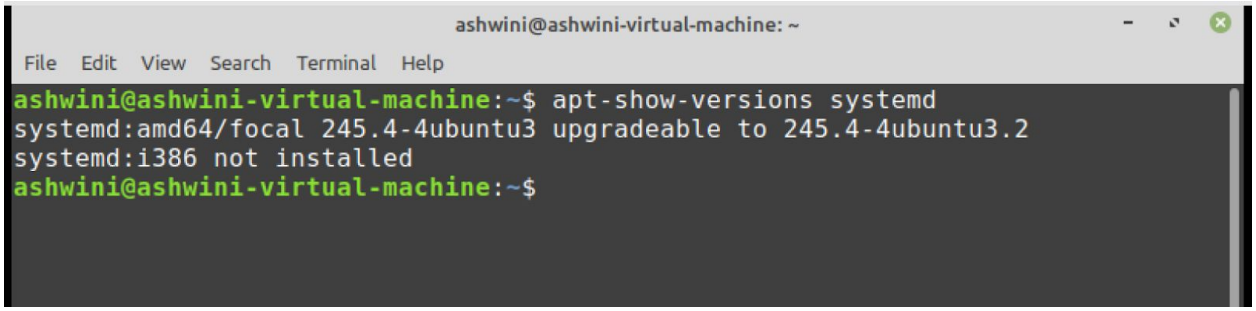


Previous command showed us systemds major version. However, distros tend to patch software they ship,

thus we also need to figure out specific package version,

Install the package [sudo apt-get install apt-show-versions]

sudo apt-show-versions systemd



Components of systemd

We will discover the main components of the systemd framework. We will examine the current system state, on disk configuration and we explain the hierarchical nature of systemd's configuration.

systemd provides many **command line client tools**.

Among them, the main tool that is used to interact with systemd is **systemctl**.

For communication with systemd it uses DBus IPC mechanism.

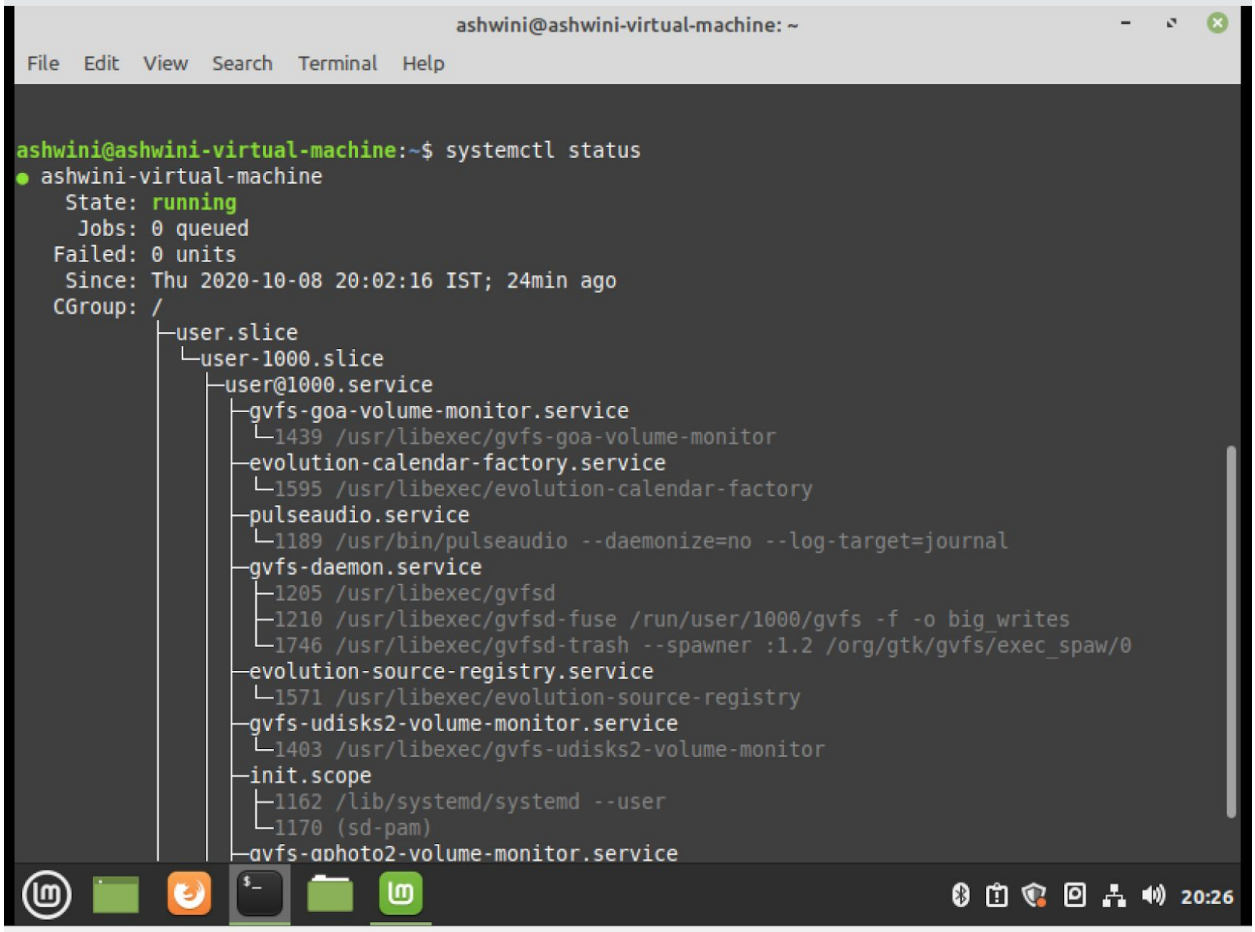
Additional Information:
Note that systemctl can be also used in environments where DBus server isn't running. Fact that it uses DBus IPC doesn't imply running dbus-daemon. systemd is very tightly integrated with logging service – journald. We believe that proper logging support is inherently necessary to provide useful service management capabilities to system administrators.

Unlike PID 1, journald doesn't offer DBus API, but it offers command line clients.

Journalctl is used to **display system logs** and it allows for very powerful filtering to quickly display desired parts of the system log. Next very important component issts. It is a system service that tracks sessions, users and implements multi-seat support.

Introspecting system state

- To get the very basic idea of current system state (as far as service manager is concerned) you can run,
  systemctl status



Output is typically quite long because we also get a detailed view of the process tree, but one of the most important pieces of information in the output is overall system state.

In pretty much all cases you want to see, State: running. Other, unfortunately very frequent states are degraded. It means that system is operational but some systemd unit is in the failed state

Listing of all services and their states can be obtained by running,

systemctl -type=service list-units command.

[???#paste the Screenshot of practice command here#???]

Next information that we are maybe interested in is the status of specific service. For example we can query systemd for status of dbus.service.

systemctl status dbus.service

[???#paste the Screenshot of practice command here#???]

Similarly to overall system status, service status also provides us with useful information about the service.

Displayed information is much richer and useful than typical initscript output 4 Running as PID 100.

Because systemd is tightly integrated with journald we also get up to ten log lines that were recently logged by the service.

A lot of useful information about systemd boot can be learned from output of the systemd-analyze tool.

systemd-analyze
systemd-analyze blame
[???#paste the Screenshot of practice command here#???]
But what runlevel (target) we actually booted into? Let's identify the default target,

systemctl get-default

[???#paste the Screenshot of practice command here#???]

Quick look at **systemd-journald** is an integral part of systemd framework. It is a **central log aggregator**, that stores a lot of meta-data alongside **log messages**.

Hence it allows for very powerful filtering.

Now we just look at some often used capabilities provided by journald.

1. We can display log messages for any system service since the machine was booted

journalctl -b -u dbus.service

[???#paste the Screenshot of practice command here#???]

2. Second very useful feature is **displaying log messages** that are related to certain device (e.g. failing disk)

journalctl -b /dev/sda

[???#paste the Screenshot of practice command here#???]

User session tracking

Since user session tracking is again a very important aspect of the system administration , systemd project introduced a new system service systemd-logind designed for this purpose.

There is couple of reasons why logind exists, but main ones are,

- user tracking
- user session tracking
- session device management
- suspend and hibernation handling

1. In order to quickly see what users are currently logged on the system you can run,

loginctl list-users

[???#paste the Screenshot of practice command here#???]

2. We can also list all sessions open by logged on users,

loginctl list-sessions

[???#paste the Screenshot of practice command here#???]

3. Next we pick one session and we obtain more detailed status of that session,

loginctl session-status

[???#paste the Screenshot of practice command here#???]

Question [Self Assessment]

1. What version of systemd are you running?

2. What exact version of the systemd package is in your distribution, when it was built, by whom?

3. What is your system state?

4. Are there any processes on your system which are not tracked by the systemd unit?

5. Do you have any failed services? If yes, can you tell from service log messages what is wrong? (Example) Demonstration Snap

Units and unit files

In this lab understanding of basic **systemd concept**s, units and unit files.

- After completing this lab you should be able to precisely articulate difference between them and know what types of units you can encounter while working with systemd and how to configure them via unit files.

Units First lets examine what are all units that systemd is currently aware of.

systemctl --all list-units

[???#paste the Screenshot of practice command here#???]

Notice their load state, active state and sub-state respectively. Usually we are interested only is a subset of all units, for example, in all running services on the system.

systemctl --type=service list-units

[???#paste the Screenshot of practice command here#???]

Even more interesting is to only list units that are in the failed state.

systemctl --type=service --state=failed list-units

[???#paste the Screenshot of practice command here#???]

If your system is in running state then you should see no failed units, however if system is running in degraded state then you will have some failed units.

In unit listing we can see there is a lot more unit types than just service. However, the workflow to obtain unit details is always the same. We list-units with appropriate **--type argument** and then we get unit status.

For each unit, systemd tracks a lot more information than is included in output of
systemctl status.
[???#paste the Screenshot of practice command here#???]

To get really detailed information you can call,
systemctl show dbus.service
[???#paste the Screenshot of practice command here#???]

To easily access unit file for given unit you can type,
systemctl cat dbus.service
[???#paste the Screenshot of practice command here#???]

Extending unit files

Now we know how to list-units and observe their state.

Go ahead and create following test unit file, using your favorite text editor,

---

 # /usr/lib/systemd/system/lisa17.service
[Service]
Type=simple
ExecStart=/bin/bash -c "echo hello LISA attendees; sleep 3600;"

[Install] WantedBy=multi-user.target

By saving your editor buffer to disk you create unit file.

However, systemd doesn't immediately create corresponding unit.

We can convince ourselves about this,
systemctl --all list-units | grep lisa17
[???#paste the Screenshot of practice command here#???]

As you can see systemd really didn't create unit just yet. Any time we changed on disk configuration, either because we have added new unit files or edited old ones we need to explicitly tell systemd to load the new config,

systemctl daemon-reload

After you have created the unit file and reloaded systemd we can start the unit and examine its status.

systemctl start lisa17.service

systemctl status lisa17.service

[???#paste the Screenshot of practice command here#???]

Notice that status of the service contains path to corresponding unit file and state of the unit file.

Since only thing we did was start of the service we still see that unit file state is disabled. Let's make sure our test service is enabled on next boot and run status once again,

systemctl enable lisa17.service

systemctl status lisa17.service

[???#paste the Screenshot of practice command here#???]

Before we proceed, try to answer for your self following questions,

- What information was displayed by enable command?
- Can you interpret meaning of the message?

- What has changed in status output?

In next part of the lab we will experiment with extending the configuration for our test unit with configuration drop-ins. First, we need to create directory where we place our drop-in config files,

mkdir -p /etc/systemd/system/lisa17.service.d

Now we can put our drop-ins in place and reload systemd.

Note that some configuration will have effect immediately after systemd reloads while other requires you to restart the service.

cat > /etc/systemd/system/lisa17.service.d/dependency.conf << EOF [Unit] After=network-online.target EOF cat > /etc/systemd/system/lisa17.service.d/tmp.conf << EOF [Service] PrivateTmp=yes EOF To make sure our configuration takes effect we issue daemon reload6 .

Furthermore we query service status. As you will see, status also show drop-in configuration that we've just applied.

systemctl daemon-reload
systemctl restart lisa17.service
systemctl status lisa17.service
[???#paste the Screenshot of practice command here#???]

Let's look at unit details and verify that drop-in configuration is applied,

systemctl show lisa17.service
[???#paste the Screenshot of practice command here#???]

****

Now you successfully learn the fundamentals of systemd concepts along with implementation.

****