Lab BSY BOOT

Introduction & Prerequisites

This laboratory is to learn how to:

- Find out systemd configuration and unit files with their locations
- Interact with systemctl
- Write service unit files

The following resources and tools are required for this laboratory session:

- Any modern web browser,
- Any modern SSH client application
- OpenStack Horizon dashboard: https://ned.cloudlab.zhaw.ch
- OpenStack account details: please contact the lab assistant in case you have not yet received your access credentials.
-

Important access credentials:

- Username to login with SSH into VMs in ned.cloudlab.zhaw.ch OpenStack cloud from your laptops
    - `ubuntu`

Help Materials

https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files

Time

The entire session will take 90 minutes.

Assessment

No assessment foreseen

Task 1 – Setup & Basic Tasks

Login into the VM.

Note: by default systemd related commands are “**paged**”. You can scroll up and down the output of commands with arrow keys, and you can exit by pressing ‘q’.
You can also disable paging by appending “--no-pager” to the command.

Useful commands for this lab are:

- systemctl
- journalctl

Subtask 1.1 – See unit status and grouping

Use the command to see the **status** of **all units** in the system

Why are units organized in **slices**? What do they represent? Look at online documentation to find out more about slices.

Compare the output of the previous command with the output of ‘pstree’. Can you find the “bash” process in both? What is the difference?

Subtask 1.2 – Targets

Use the command to list all **available target units**

Find out which command to use to find the **default target** for the local operating system. What is the default target in the VM?

Which command can you use to see the **unit file** of the default target and its **location**?

Looking at that unit file, can you tell which **service unit** should be started together with that target? Is it required? What’s the difference between **‘requires’** and **‘wants’**?

There is a command that lets you see the “**dependencies**” of each unit.
What is the command to list the dependencies of the default target?
What is the command to see the dependencies of the ‘sysinit.target’?
What do the listed dependencies mean?

Use the man command to see which systemctl command allows to see **which units depend on** a specific unit.

Which targets depend on the ssh.service? (i.e., which targets either want or require ssh.service?)

Task 2 - Services

Subtask 2.1 – Inspect a service

Let's check the **status** of the sshd.service with systemctl
Can you tell from the status command the **PID** of the ssh process? Can you tell on which port it is running from the logs?

In order to see the **full logs** of the service we can use a specific command. How do you see the logs of the ssh.service unit?

What command can you use to see the unit-file of the ssh.service? Can you also see its location on the file system?

Bonus Question (Optional)
Take a look at the unit file, can you see how the process is started? Can you see how the kill command (sending a signal) is used to reload the service configuration (without killing the process)? Notice the usage of the variable $MAINPID

Subtask 2.2 – Killing a service: Restart

The proper way to stop a service unit is with the "systemctl stop" command. But before we do that, let's try to kill a service process manually.

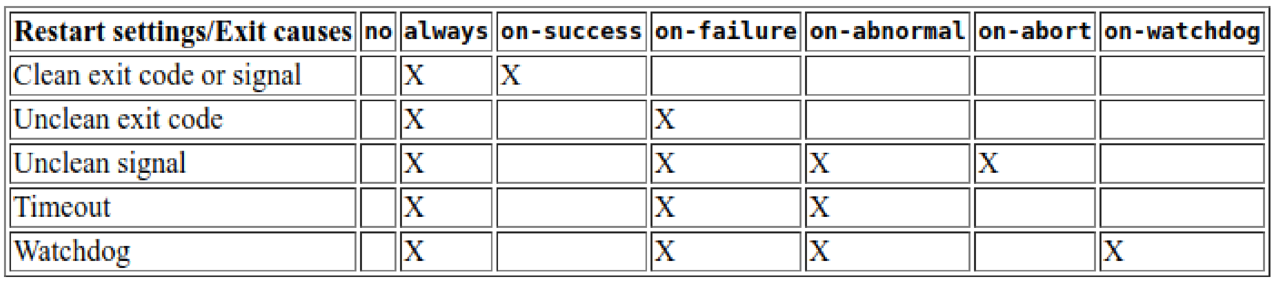
Use the "kill -SIGKILL <PID>" command (e.g., "sudo kill -9 676") to kill the "cron.service".
If you check the status of the cron service before and after killing the cron process, what do you see?

Let's now try to use a different signal (TERM) to kill cron (without '-9'): sudo kill <PID>
What happens in this case?

Look at the cron.service unit file with the systemctl cat command. What are the restarting conditions?

With "on-failure" restart conditions a service process is restarted only at the conditions in table below. Sending the TERM signal the cron process exits cleanly and is not restarted.

Table 1. Exit causes and the effect of the `Restart=` settings on them



Restart settings/Exit causes	`no`	`always`	`on-success`	`on-failure`	`on-abnormal`	`on-abort`	`on-watchdog`
Clean exit code or signal		X	X				
Unclean exit code		X		X			
Unclean signal		X		X	X	X	
Timeout		X		X	X		
Watchdog		X		X	X		X

Subtask 2.3 – Making the service restart always

Let's make a copy of the original cron.service file in **/etc/systemd/system** (configuration in /etc overrides the one in /lib)
Use a text editor to change the "**Restart**" condition to "**always**"
Force systemd to reload the units with the **daemon-reload** subcommand
Verify that also when using the TERM signal to kill the cron process it is restarted

Subtask 2.4 – KillMode

Take a look at the ssh.service unit file, in particular at the attribute KillMode.
Have a look at the documentation of the KillMode config:
https://www.freedesktop.org/software/systemd/man/systemd.kill.html

What would happen if you were to stop the ssh.service with systemctl stop?
Would your ssh session (and bash terminal) be killed?

NOTE: Don't try killing the ssh service or you will lose access to your VM!

Task 3 - Create a Service Unit

Subtask 3.1 – Unit File

Use as a reference the simple service unit file provided here:
https://github.com/aviborg/systemdHelloWorld/blob/master/sHW.service

Create a service that writes the file /home/ubuntu/service_started each time it gets started.

Hints:
- In which directory should you write the service unit file? (See for instance where other unit files are saved using the appropriate systemctl command)
- What command can you use to create a new file in the service unit?

Use the appropriate command to start your service.
Check its status.
Verify it performed the intended action.

Subtask 3.2 – Set dependencies with Install section

Read the documentation about the "Install Section" at the following link:
https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files

How would you extend your service unit file to make it automatically start with the multi-user.target?

In order to make your service autostart you will also need to enable it with the appropriate command.

Does this start the service? What command enables and immediately starts the service?

Task 4 - Systemd User

Subtask 4.1 – Unit File Installation

You are given the following unit file for the “http_share.service”:

```
-------------------------

[Unit]
Description=HTTP Sharing Server

[Service]
Type=simple
WorkingDirectory=/home/ubuntu/shared
ExecStart=/usr/bin/python3 -m http.server 8080
Restart=always

[Install]
WantedBy=multi-user.target

------------------------
```

Install it, enable it, start it as a user service

Make sure your VM has security group ALL_IN configured, can you use your browser to access the service?

Subtask 4.2 – Follow up service

Can you write a user service that starts right after the service above and saves the html page returned by http_share.service? Use the wget command (man wget)

Subtask 4.3 – Start Without User Login

How do you inform systemd that you want the services for the user ubuntu to start at boot instead of when she logs in?

Cleanup

IMPORTANT: At the end of the lab session:

- **Delete** all OpenStack VMs, volumes, security group rules that were created by your team.
- **Release** all floating IPs back to the central pool for others to use.
  - Go to Network -> Floating IPs to release IPs back to the pool