Lab 1 Boot

Task 1

Subtask 1.1 – See unit status and grouping

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

```
ubuntu@boot:~$ systemctl status --all
...
```

- Why are units organized in slices?

```
To manage resources that belong to a certain group. For example the slice
`user.slice` doesn't allow the user to use 100% of the hardware, because
it might block the OS from working.
```

- What do they represent?

```
 A slice unit is a concept for hierarchically managing resources of a
group of processes
```

- Compare the output of the previous command with the output of 'pstree'. Can you find the "bash" process in both? What is the difference?

```
pstree seems to show less process and also less information.
Yes, I can see bash in both outputs.
```

Subtask 1.2 – Targets

- Use the command to list all available target units

```
ubuntu@boot:~$ systemctl list-units --all -t target
  UNIT                                                    LOAD   ACTIVE
SUB    DESCRIPTION
  basic.target                                            loaded active
active Basic System
  blockdev@dev-disk-by\x2dlabel-cloudimg\x2drootfs.target loaded inactive
dead   Block Device Preparation for /dev/disk/by-label/cloudimg-rootfs
  blockdev@dev-disk-by\x2dlabel-UEFI.target               loaded inactive
dead   Block Device Preparation for /dev/disk/by-label/UEFI
```

```
  blockdev@dev-loop0.target                                  loaded inactive
dead   Block Device Preparation for /dev/loop0
  blockdev@dev-loop1.target                                  loaded inactive
dead   Block Device Preparation for /dev/loop1
  blockdev@dev-loop2.target                                  loaded inactive
dead   Block Device Preparation for /dev/loop2
  blockdev@dev-vda15.target                                  loaded inactive
dead   Block Device Preparation for /dev/vda15
  cloud-config.target                                        loaded active
active Cloud-config availability
  cloud-init.target                                          loaded active
active Cloud-init target
  cryptsetup.target                                          loaded active
active Local Encrypted Volumes
  emergency.target                                           loaded inactive
dead   Emergency Mode
  getty-pre.target                                           loaded inactive
dead   Login Prompts (Pre)
  getty.target                                               loaded active
active Login Prompts
  graphical.target                                           loaded active
active Graphical Interface
  local-fs-pre.target                                        loaded active
active Local File Systems (Pre)
  local-fs.target                                            loaded active
active Local File Systems
  multi-user.target                                          loaded active
active Multi-User System
  network-online.target                                      loaded active
active Network is Online
  network-pre.target                                         loaded active
active Network (Pre)
  network.target                                             loaded active
active Network
  nss-lookup.target                                          loaded active
active Host and Network Name Lookups
  nss-user-lookup.target                                     loaded active
active User and Group Name Lookups
  paths.target                                               loaded active
active Paths
  remote-fs-pre.target                                       loaded active
active Remote File Systems (Pre)
  remote-fs.target                                           loaded active
active Remote File Systems
  rescue.target                                              loaded inactive
dead   Rescue Mode
  shutdown.target                                            loaded inactive
dead   Shutdown
  sleep.target                                               loaded inactive
dead   Sleep
  slices.target                                              loaded active
active Slices
  sockets.target                                             loaded active
active Sockets
```

```
  swap.target                                                loaded active
active Swap
  sysinit.target                                             loaded active
active System Initialization
  time-set.target                                            loaded active
active System Time Set
  time-sync.target                                           loaded active
active System Time Synchronized
  timers.target                                              loaded active
active Timers
  umount.target                                              loaded inactive
dead   Unmount All Filesystems

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

36 loaded units listed.
To show all installed unit files use 'systemctl list-unit-files'.
```

- Find out which command to use to find the default target for the local operating system

```
ubuntu@boot:~$ systemctl get-default
graphical.target
```

- What is the default target in the VM?

```
graphical.target
```

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

```
ubuntu@boot:~$ systemctl cat graphical.target
# /lib/systemd/system/graphical.target
#  SPDX-License-Identifier: LGPL-2.1+
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published
by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
```

```
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-
manager.service
AllowIsolate=yes
```

- What service unit should be started with that target?

```
Requires=multi-user.target
```

- Is it required?

```
yes
```

- What's the difference between 'requires' and 'wants'?

```
Require --> The service needs all required units to be running or else it
can't run --> systemd won't start the service
Wants --> The service doesn't need the wanted services to be running, but
it would appreciate it
```

- There is a command that lets you see the "dependencies" of each unit.

```
ubuntu@boot:~$ systemctl list-dependencies
...
```

> why is multi-user.target not dependent on basic.target? Or how can you read the graph?

- What is the command to list the dependencies of the default target?

```
ubuntu@boot:~$ systemctl list-dependencies graphical.target
graphical.target
● ├─accounts-daemon.service
● ├─apport.service
● ├─display-manager.service
● ├─e2scrub_reap.service
● ├─systemd-update-utmp-runlevel.service
● ├─udisks2.service
● └─multi-user.target
●   ├─apport.service
●   ├─atd.service
```

```
●   ├─console-setup.service
●   ├─cron.service
●   ├─dbus.service
●   ├─dmesg.service
●   ├─grub-common.service
●   ├─grub-initrd-fallback.service
●   ├─irqbalance.service
●   ├─lxd-agent-9p.service
●   ├─lxd-agent.service
●   ├─networkd-dispatcher.service
●   ├─ondemand.service
●   ├─open-vm-tools.service
●   ├─plymouth-quit-wait.service
●   ├─plymouth-quit.service
●   ├─pollinate.service
●   ├─rsync.service
●   ├─rsyslog.service
●   ├─secureboot-db.service
●   ├─snap-core20-1081.mount
●   ├─snap-lxd-21545.mount
●   ├─snap-snapd-13170.mount
●   ├─snap.lxd.activate.service
●   ├─snapd.apparmor.service
●   ├─snapd.autoimport.service
●   ├─snapd.core-fixup.service
●   ├─snapd.recovery-chooser-trigger.service
●   ├─snapd.seeded.service
●   ├─snapd.service
●   ├─ssh.service
●   ├─systemd-ask-password-wall.path
●   ├─systemd-logind.service
●   ├─systemd-networkd.service
●   ├─systemd-resolved.service
●   ├─systemd-update-utmp-runlevel.service
●   ├─systemd-user-sessions.service
●   ├─ua-reboot-cmds.service
●   ├─ufw.service
●   ├─basic.target
●   │ ├─-.mount
●   │ ├─tmp.mount
●   │ ├─paths.target
...
```

- What is the command to see the dependencies of the ‘sysinit.target’?

```
ubuntu@boot:~$ systemctl list-dependencies sysinit.target
sysinit.target
● ├─apparmor.service
● ├─blk-availability.service
● ├─dev-hugepages.mount
● ├─dev-mqueue.mount
● ├─finalrd.service
```

```
● ├─keyboard-setup.service
● ├─kmod-static-nodes.service
● ├─lvm2-lvmpolld.socket
● ├─lvm2-monitor.service
● ├─multipathd.service
● ├─open-iscsi.service
● ├─plymouth-read-write.service
● ├─plymouth-start.service
● ├─proc-sys-fs-binfmt_misc.automount
● ├─setvtrgb.service
● ├─sys-fs-fuse-connections.mount
● ├─sys-kernel-config.mount
● ├─sys-kernel-debug.mount
● ├─sys-kernel-tracing.mount
● ├─systemd-ask-password-console.path
● ├─systemd-binfmt.service
● ├─systemd-boot-system-token.service
● ├─systemd-hwdb-update.service
● ├─systemd-journal-flush.service
● ├─systemd-journald.service
● ├─systemd-machine-id-commit.service
● ├─systemd-modules-load.service
● ├─systemd-pstore.service
● ├─systemd-random-seed.service
● ├─systemd-sysctl.service
● ├─systemd-sysusers.service
● ├─systemd-timesyncd.service
● ├─systemd-tmpfiles-setup-dev.service
● ├─systemd-tmpfiles-setup.service
● ├─systemd-udev-trigger.service
● ├─systemd-udevd.service
● ├─systemd-update-utmp.service
● ├─cryptsetup.target
● ├─local-fs.target
● │ ├─-.mount
● │ ├─boot-efi.mount
● │ ├─systemd-fsck-root.service
● │ └─systemd-remount-fs.service
● └─swap.target
```

- What do the listed dependencies mean?

```
This recursively lists units following the Requires=, Requisite=,
ConsistsOf=, Wants=, BindsTo= dependencies.
```

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

```
ubuntu@boot:~$ systemctl list-dependencies --reverse
default.target
```

- Which target(s) depends on the ssh.service ?

```
ubuntu@boot:~$ systemctl list-dependencies ssh.service --reverse
ssh.service
● ├─cloud-init.service
● └─multi-user.target
●   └─graphical.target
```

Task 2 - Services

Subtask 2.1 – Inspect a service

- Can you tell from the status command the PID of the ssh process?

```
ubuntu@boot:~$ systemctl status sshd.service
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor
preset: enabled)
     Active: active (running) since Fri 2023-03-10 09:07:00 UTC; 1h 5min
ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 641 ExecStartPre=/usr/sbin/sshd -t (code=exited,
status=0/SUCCESS)
   Main PID: 681 (sshd)
      Tasks: 1 (limit: 9513)
     Memory: 6.8M
     CGroup: /system.slice/ssh.service
             └─681 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Mar 10 09:07:00 bsy-boot-lab systemd[1]: Starting OpenBSD Secure Shell
server...
Mar 10 09:07:00 bsy-boot-lab sshd[681]: Server listening on 0.0.0.0 port
22.
Mar 10 09:07:00 bsy-boot-lab sshd[681]: Server listening on :: port 22.
Mar 10 09:07:00 bsy-boot-lab systemd[1]: Started OpenBSD Secure Shell
server.
Mar 10 09:11:43 bsy-boot-lab sshd[865]: Connection reset by 160.85.148.96
port 55062 [preauth]
Mar 10 09:13:03 bsy-boot-lab sshd[874]: Accepted publickey for ubuntu from
160.85.148.96 port 55070 ssh2: RSA
SHA256:rh+4lkk3TFXrGrwPSYe5jK/aG0cxJrU54kgQ4aRrndU
Mar 10 09:13:03 bsy-boot-lab sshd[874]: pam_unix(sshd:session): session
opened for user ubuntu by (uid=0)
```

```
Yes --> Main PID: 681 (sshd)
```

- Can you tell on which port it is running from the logs?

```
Yes, on the port 22
```

- 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?

```
ubuntu@boot:~$ journalctl -u sshd.service
-- Logs begin at Fri 2023-03-10 07:55:14 UTC, end at Fri 2023-03-10
10:17:01 UTC. --
-- No entries --
```

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

```
ubuntu@bsy-boot-lab:~$ systemctl cat sshd.service
# /lib/systemd/system/ssh.service
[Unit]
Description=OpenBSD Secure Shell server
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify
RuntimeDirectory=sshd
RuntimeDirectoryMode=0755

[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Subtask 2.2 – Killing a service: Restart

- Use the “kill -SIGKILL “ 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?

```
ubuntu@bsy-boot-lab:~$ systemctl status cron.service
● cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor
preset: enabled)
     Active: active (running) since Tue 2023-03-14 17:06:30 UTC; 1h 3min
ago
       Docs: man:cron(8)
   Main PID: 699 (cron)
      Tasks: 1 (limit: 4682)
     Memory: 1.1M
     CGroup: /system.slice/cron.service
             └─699 /usr/sbin/cron -f

Mar 14 17:06:30 bsy-boot-lab cron[699]: (CRON) INFO (pidfile fd = 3)
Mar 14 17:06:30 bsy-boot-lab cron[699]: (CRON) INFO (Running @reboot jobs)
Mar 14 17:06:30 bsy-boot-lab systemd[1]: Started Regular background
program processing daemon.
Mar 14 17:17:01 bsy-boot-lab CRON[1470]: pam_unix(cron:session): session
opened for user root by (uid=0)
Mar 14 17:17:01 bsy-boot-lab CRON[1470]: pam_unix(cron:session): session
closed for user root
Mar 14 17:33:01 bsy-boot-lab CRON[1528]: pam_unix(cron:session): session
opened for user root by (uid=0)
Mar 14 17:33:01 bsy-boot-lab CRON[1529]: (root) CMD (   test -x
/etc/cron.daily/popularity-contest && /et>
Mar 14 17:33:01 bsy-boot-lab CRON[1528]: pam_unix(cron:session): session
closed for user root
```

```
ubuntu@bsy-boot-lab:~$ systemctl status cron.service
● cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor
preset: enabled)
     Active: active (running) since Tue 2023-03-14 18:11:32 UTC; 6s ago
       Docs: man:cron(8)
   Main PID: 13519 (cron)
      Tasks: 1 (limit: 4682)
     Memory: 448.0K
     CGroup: /system.slice/cron.service
             └─13519 /usr/sbin/cron -f

Mar 14 18:11:32 bsy-boot-lab systemd[1]: cron.service: Scheduled restart
job, restart counter is at 1.
Mar 14 18:11:32 bsy-boot-lab systemd[1]: Stopped Regular background
program processing daemon.
Mar 14 18:11:32 bsy-boot-lab systemd[1]: Started Regular background
program processing daemon.
Mar 14 18:11:32 bsy-boot-lab cron[13519]: (CRON) INFO (pidfile fd = 3)
```

```
Mar 14 18:11:32 bsy-boot-lab cron[13519]: (CRON) INFO (Skipping @reboot
jobs -- not system startup)
```

```
The service was stopped / killed, but the service is auto-restarting by
itself. Check the Main PID (it changed)
```

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

```
ubuntu@bsy-boot-lab:~$ sudo kill 13519
ubuntu@bsy-boot-lab:~$ systemctl status cron.service
● cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor
preset: enabled)
     Active: inactive (dead) since Tue 2023-03-14 18:17:12 UTC; 3s ago
       Docs: man:cron(8)
    Process: 13519 ExecStart=/usr/sbin/cron -f $EXTRA_OPTS (code=killed,
signal=TERM)
   Main PID: 13519 (code=killed, signal=TERM)

Mar 14 18:11:32 bsy-boot-lab systemd[1]: cron.service: Scheduled restart
job, restart counter is at 1.
Mar 14 18:11:32 bsy-boot-lab systemd[1]: Stopped Regular background
program processing daemon.
Mar 14 18:11:32 bsy-boot-lab systemd[1]: Started Regular background
program processing daemon.
Mar 14 18:11:32 bsy-boot-lab cron[13519]: (CRON) INFO (pidfile fd = 3)
Mar 14 18:11:32 bsy-boot-lab cron[13519]: (CRON) INFO (Skipping @reboot
jobs -- not system startup)
Mar 14 18:17:01 bsy-boot-lab CRON[13527]: pam_unix(cron:session): session
opened for user root by (uid=0)
Mar 14 18:17:01 bsy-boot-lab CRON[13527]: pam_unix(cron:session): session
closed for user root
Mar 14 18:17:12 bsy-boot-lab systemd[1]: cron.service: Succeeded.
```

```
The service is now inactive (dead) and will not be auto-starting
```

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

```
ubuntu@boot:~$ systemctl cat cron.service
# /lib/systemd/system/cron.service
[Unit]
Description=Regular background program processing daemon
Documentation=man:cron(8)
After=remote-fs.target nss-user-lookup.target
```

```
[Service]
EnvironmentFile=-/etc/default/cron
ExecStart=/usr/sbin/cron -f $EXTRA_OPTS
IgnoreSIGPIPE=false
KillMode=process
Restart=on-failure

[Install]
WantedBy=multi-user.target

Restarting(Restart=) condition is on-failure.

From the manpage:

If set to on-failure, the service will be restarted when the process exits
with a non-zero exit code, is terminated by a signal (including on core
dump, but excluding the aforementioned four signals ( SIGHUP, SIGINT,
SIGTERM, or SIGPIPE)), when an operation (such as service reload) times
out, and when the configured watchdog timeout is triggered.
```

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.

```
ubuntu@boot:~$ sudo cp /usr/lib/systemd/system/cron.service
/etc/systemd/system/
ubuntu@boot:~$ sudo vim /etc/systemd/system/cron.service
ubuntu@boot:~$ sudo systemctl daemon-reload
ubuntu@boot:~$ sudo systemctl start cron.service
ubuntu@bsy-boot-lab:~$ sudo systemctl status cron.service
● cron.service - Regular background program processing daemon
     Loaded: loaded (/etc/systemd/system/cron.service; enabled; vendor
preset: enabled)
     Active: active (running) since Tue 2023-03-14 18:29:22 UTC; 17s ago
       Docs: man:cron(8)
   Main PID: 13654 (cron)
      Tasks: 1 (limit: 4682)
     Memory: 448.0K
     CGroup: /system.slice/cron.service
             └─13654 /usr/sbin/cron -f

Mar 14 18:29:22 bsy-boot-lab systemd[1]: Started Regular background
program processing daemon.
Mar 14 18:29:22 bsy-boot-lab cron[13654]: (CRON) INFO (pidfile fd = 3)
Mar 14 18:29:22 bsy-boot-lab cron[13654]: (CRON) INFO (Skipping @reboot
jobs -- not system startup)
ubuntu@bsy-boot-lab:~$ sudo kill 13654
```

```
ubuntu@bsy-boot-lab:~$ sudo systemctl status cron.service
● cron.service - Regular background program processing daemon
     Loaded: loaded (/etc/systemd/system/cron.service; enabled; vendor
preset: enabled)
     Active: active (running) since Tue 2023-03-14 18:30:12 UTC; 5s ago
       Docs: man:cron(8)
   Main PID: 13681 (cron)
      Tasks: 1 (limit: 4682)
     Memory: 444.0K
     CGroup: /system.slice/cron.service
             └─13681 /usr/sbin/cron -f

Mar 14 18:30:12 bsy-boot-lab systemd[1]: cron.service: Scheduled restart
job, restart counter is at 1.
Mar 14 18:30:12 bsy-boot-lab systemd[1]: Stopped Regular background
program processing daemon.
Mar 14 18:30:12 bsy-boot-lab systemd[1]: Started Regular background
program processing daemon.
Mar 14 18:30:12 bsy-boot-lab cron[13681]: (CRON) INFO (pidfile fd = 3)
Mar 14 18:30:12 bsy-boot-lab cron[13681]: (CRON) INFO (Skipping @reboot
jobs -- not system startup)
```

```
This shows that it worked, because the service is still running even after
sending SIGTERM.
```

Subtask 2.4 – KillMode

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

```
ubuntu@boot:~$ systemctl cat sshd.service
# /lib/systemd/system/ssh.service
[Unit]
Description=OpenBSD Secure Shell server
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify
RuntimeDirectory=sshd
```

```
RuntimeDirectoryMode=0755

[Install]
WantedBy=multi-user.target
Alias=sshd.service

KillMode=process

According to the manage:

Note that it is not recommended to set KillMode= to process or even none,
as this allows processes to escape the service manager's lifecycle and
resource management, and to remain running even while their service is
considered stopped and is assumed to not consume any resources.

This means that my SSH session would not be killed by systemd and will
outlive the systemd unit lifecycle
```

Task 3 - Create a Service Unit

Subtask 3.1 – Unit File

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

> 1. methood

```
sudo nano /usr/local/bin/create_file.sh
```

```
#!/bin/bash
touch /home/ubuntu/service_started
```

```
sudo nano /etc/systemd/system/file_creation.service
```

```
[Unit]
Description=Service to create /home/ubuntu/service_started file

[Service]
Type=simple
ExecStart=/usr/local/bin/create_file.sh
WatchdogSec=20
Restart=on-failure
TimeoutStopSec=30

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

```
sudo systemctl daemon-reload
sudo systemctl start file_creation.service
sudo systemctl status file_creation.service
```

```
ls -l /home/ubuntu/service_started
```

> 2. method

```
ubuntu@bsy-boot-lab:~$ cd /etc/systemd/user/
ubuntu@bsy-boot-lab:/etc/systemd/user$ sudo nano create_file.service
ubuntu@bsy-boot-lab:/etc/systemd/user$ systemctl start --user
create_file.service
ubuntu@bsy-boot-lab:/etc/systemd/user$ systemctl status --user
create_file.service
● create_file.service - if start touch children...I mean file
     Loaded: loaded (/etc/xdg/systemd/user/create_file.service; disabled;
vendor preset: enable>
     Active: inactive (dead)

Mar 14 19:26:38 bsy-boot-lab systemd[1286]: Started if start touch
children...I mean file.
Mar 14 19:26:38 bsy-boot-lab systemd[1286]: create_file.service:
Succeeded.

ubuntu@bsy-boot-lab:/etc/systemd/user$ ll /home/ubuntu/service_started
-rw-rw-r-- 1 ubuntu ubuntu 0 Mar 14 19:26 /home/ubuntu/service_started
```

Subtask 3.2 – Set dependencies with Install section

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

```
already done previously
```

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

```
ubuntu@bsy-boot-lab:/etc/systemd/user$ systemctl enable --user
create_file.service
Created symlink /home/ubuntu/.config/systemd/user/multi-
user.target.wants/create_file.service →
/etc/xdg/systemd/user/create_file.service.
```

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

```
ubuntu@bsy-boot-lab:/etc/systemd/user$ systemctl enable --now --user
create_file.service
```

Task 4 - Systemd User

Subtask 4.1 – Unit File Installation

```
ubuntu@bsy-boot-lab:/etc/systemd/user$ sudo nano http_share.service
ubuntu@bsy-boot-lab:/etc/systemd/user$ systemctl enable --user
http_share.service
Created symlink /home/ubuntu/.config/systemd/user/multi-
user.target.wants/http_share.service →
/etc/xdg/systemd/user/http_share.service.
ubuntu@bsy-boot-lab:/etc/systemd/user$ systemctl start --user
http_share.service
```

Subtask 4.2 – Follow up service

```
ubuntu@bsy-boot-lab:/etc/systemd/user$ sudo nano
/usr/local/bin/download_html.sh

#!/bin/bash
VM_PUBLIC_IP="160.85.37.84"
wget -O /home/ubuntu/downloaded_page.html http://$VM_PUBLIC_IP:8080

chmod +x /usr/local/bin/download_html.sh
```

```
sudo nano download_html.service

[Unit]
Description=Download HTML page from HTTP Sharing Server
After=http_share.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/download_html.sh

[Install]
WantedBy=default.target
```

```
subuntu@bsy-boot-lab:/etc/systemd/user$ systemctl --user daemon-reload

ubuntu@bsy-boot-lab:/etc/systemd/user$ systemctl --user enable --now
download_html.service
```

Subtask 4.3 – Start Without User Login

```
ubuntu@bsy-boot-lab:/etc/systemd/user$ sudo nano
/etc/systemd/system/start_ubuntu_user_services.service

[Unit]
Description=Start ubuntu user services at boot
After=network.target

[Service]
Type=simple
User=ubuntu
Group=ubuntu
ExecStart=/usr/bin/systemctl --user --no-block start default.target
ExecStop=/usr/bin/systemctl --user --no-block stop default.target
Restart=on-failure

[Install]
WantedBy=multi-user.target

ubuntu@bsy-boot-lab:/etc/systemd/user$ sudo systemctl enable
start_ubuntu_user_services.service
Created symlink /etc/systemd/system/multi-
user.target.wants/start_ubuntu_user_services.service →
/etc/systemd/system/start_ubuntu_user_services.service.
```