Linux Commands and Scripting Report

By Hamza Alhalabi

Table of Contents

Objective

- Learning Linux and Bash scripting by building usfel tools that can be used in the real-world

- Building personal-customized backup script that compress files and directories and backup them in specified location
- Generating a system health report that helps in understanding what is going on with devices behind the scenes in a simpler way

- The main objective is to gain the knowledge and the experience by applying the scripting concepts and building real scripts to automate frequent tasks

Assignment Overview

- Our task is to write to scripts to automate some tasks
- The first script is a script that perform backup operations on files, with compression,error handling, and logging informationto a file
- The other script should generate a report about system health information like disk space, memory usage, and current services

Script Development

Backup Script

- **Purpose**: it performs a backup of user-specified directories and implements features like compression, logging of backup status to another separate file, and error handling to avoid crushes. It gives information about the size of backup files during the process
- **Features**:
  - Compression
  - Logging
  - Error handling
  - Backup to another location
  - User-friendly and interactive experience
  - Ability to be automated and scheduled
- At first I created the file, wrote the script, test it with real **source** and **target** directories to simulate the complete experience. When I finished and felt that I am ready to save my script, I added the command that triggers the script as an **alias** in my shell configurations with the name "backup-script" and became able to do files backup form any where in the system just by using this command

```
hamza-damas@Hamza-Dama: | hamza-damas@Hamza-Dama: | hamza-damas@Hamza-Dama: | hamza-damas@Hamza-Dama

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# alias for my backup script
alias backup-script='bash /home/hamza-damas/scripts/backup-script/backup-script.sh'
```

- I added a `show_help` function that display critical information on how to use the script, this function triggers when the user pass `--help` as the first argument

```
# Check if --help is provided as the first argument
if [[ "$1" == "--help" ]]; then
    show_help
    exit 0
fi
```

```
hamza-damas@Hamza-Damas:~/scripts/backup-script$ backup-script --help
Usage: backup-script [DIRECTORY1] [DIRECTORY2] ... [DESTINATION] [RETENTION_DAYS]
pass all directories and files that you want to backup, then pass the destination directory
at the end pass a number (number of days that backup will be deleted before it)
Example:  /home/user/docs /home/user/photos /backup/location 7

Options:
  --help                Show this help message and exit
  [DIRECTORY-N]         Specify all directories and files to backup
  [DESTINATION]         Specify the location of the backup
  [RETENTION_DAYS]      Specify the number of retention days
hamza-damas@Hamza-Damas:~/scripts/backup-script$
```

- When the user attempts to run the script without the valid format of arguments, this error is handled without any problems, using interactive prompts

```
# check if at least three arguments are provided (directories + destination + retention days)
if [ "$#" -lt 3 ]; then
    echo "Error: At least three arguments: one directory, a destination, and retention days are required"
    # show script usage instructions
    usage
fi
```

```
hamza-damas@Hamza-Damas:~/scripts/backup-script$ backup-script
Error: At least three arguments: one directory, a destination, and retention days are required
Usage: /home/hamza-damas/scripts/backup-script/backup-script.sh [DIRECTORY1] [DIRECTORY2] ... [DESTINATION] [RETENTION_DAYS]
Example: /home/hamza-damas/scripts/backup-script/backup-script.sh /home/user/docs /home/user/photos /backup/location 7
```

- When running the script directory with complete and existed paths for both source files and target destination, the files will be archived and compressed and sent to the destination location path in a new directory called `daily-backups` , in this directory there is a nested directoy inside it with the date of the day, inside this daily directory there is two files; the backup compressed file and a log file with the information related to the last backup operation

```
hamza-damas@Hamza-Damas:~$ backup-script /mnt/c/Users/HamzaWH/linux-practice/backup-script/sources/java-files/ /mnt/c/Users/
HamzaWH/linux-practice/backup-script/sources/photos/ /mnt/c/Users/HamzaWH/linux-practice/backup-script/targets/ 15
Backup Log - 20250124
Backup started at: Fri Jan 24 08:04:10 +03 2025
Directories backed up:
- /mnt/c/Users/HamzaWH/linux-practice/backup-script/sources/java-files/
- /mnt/c/Users/HamzaWH/linux-practice/backup-script/sources/photos/
tar: Removing leading `/' from member names
tar: Removing leading `/' from hard link targets
Backup successful! Archive created at '/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets//daily-backups/20250124/bac
kup-20250124.tar.gz'
Backup file size: 4.2M
Cleaning up old backups...
No backup files older than 15 days to delete
Backup completed at: Fri Jan 24 08:04:11 +03 2025
Backup process finished
hamza-damas@Hamza-Damas:~$
```

- As shown in images, the `daily-backups` directory contains backups for all the wanted days, we can determine the wanted days using `RETENTION_DAYS` argument which is the last argument

```
hamza-damas@Hamza-Damas:~$ cd /mnt/c/Users/HamzaWH/linux-practice/backup-script
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script$ ls
backup-second.sh  backup.sh  sources  targets
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script$ cd targets/
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets$ ls
daily-backups
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets$ cd daily-backups/
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets/daily-backups$ ls
20250122  20250123  20250124
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets/daily-backups$ cd 20250123
```

- Same contents that appear in the console will be appended to the log file, it mentions directories names, destination location, backup files size, and any relevent information

```
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets/daily-backups/20250123$ cat backup-2025012
3.log
Backup Log - 20250123
Backup started at: Thu Jan 23 23:59:04 +03 2025
Directories backed up:
- /mnt/c/Users/HamzaWH/linux-practice/backup-script/sources/java-files/
- /mnt/c/Users/HamzaWH/linux-practice/backup-script/sources/photos/
tar: Removing leading `/' from member names
tar: Removing leading `/' from hard link targets
Backup successful! Archive created at '/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets//daily-backups/20250123/bac
kup-20250123.tar.gz'
Backup file size: 4.2M
Cleaning up old backups...
No backup files older than 15 days to delete
Backup completed at: Thu Jan 23 23:59:05 +03 2025
Backup process finished
```

- The main used command in this script is `tar` command. I put the source directories in an archive to be in the format `.tar` and compressed it in the same command to result in `.tar.gz` format

```
# Create the tar archive
tar -czf "$DESTINATION/daily-backups/$BACKUP_DATE/$BACKUP_FILENAME" "${DIRECTORIES[@]}" 2>>"$LOG_FILE"

# Check if the tar command was successful
if [ $? -eq 0 ]; then
    # Get the size of the backup file
    BACKUP_SIZE=$(du -h "$DESTINATION/daily-backups/$BACKUP_DATE/$BACKUP_FILENAME" | cut -f1)
    echo "Backup successful! Archive created at '$DESTINATION/daily-backups/$BACKUP_DATE/$BACKUP_FILENAME'" >> "$LOG_FILE"
    echo "Backup file size: $BACKUP_SIZE" >> "$LOG_FILE"
else
    echo "Error: Failed to create the backup" >> "$LOG_FILE"
    exit 8
fi
```

- One of the interesting parts of my script was the idea of deleting backups that are older than certain date. The user passes that value (number of days), this helps with keeping only the most recent backups in the device not everything. This process was made using `find -delete` command

```
# Delete files older than the retention period
echo "Cleaning up old backups..." >> "$LOG_FILE"
DELETED_FILES=$(find "$DESTINATION/daily-backups/"* -mtime +$RETENTION_DAYS -print -delete)
if [ -n "$DELETED_FILES" ]; then
        echo "Deleted backup files older than $RETENTION_DAYS days:" >> "$LOG_FILE"
    echo "$DELETED_FILES" >> "$LOG_FILE"
else
        echo "No backup files older than $RETENTION_DAYS days to delete" >> "$LOG_FILE"
fi
```

- Finally I added my script with required arguments to Linux job scheduler to automate its execution daily, I added it so it will be running every day at `23:59` , this ensures that my files in this example will have a backup all the time

```
t$ crontab -e
```

```
hamza-damas@Hamza-Dama  hamza-damas@Hamza-Dama:  hamza-damas@Hamza-Dama:

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
59 23 * * * bash /home/hamza-damas/scripts/backup-script/backup-script.sh /mnt/c/Users/HamzaWH/linux-practice/backup-script/
sources/java-files/ /mnt/c/Users/HamzaWH/linux-practice/backup-script/sources/photos/ /mnt/c/Users/HamzaWH/linux-practice/ba
ckup-script/targets/ 15
~
```

- I faced many problems at first related files permissions (rwx) especially the write permission for the destination directory, I added an if statement validation

```
# Validate writting permissions
if [ ! -w "$DEST_DIR" ]; then
    echo "Error: No write permissions for the destination directory '$DEST_DIR'" | tee -a "$LOG_FILE" >&2
    exit 5
fi
```

- I added validations to ensure that all arguments paths are correct, real directories locations, and valid to manipulate

```
# Validate that source directories are readable
for dir in "${DIRECTORIES[@]}"; do
    if [ ! -r "$dir" ]; then
        echo "Error: Directory '$dir' is not readable" | tee -a "$LOG_FILE" >&2
        exit 3
    fi
done

# Validate the destination
DEST_DIR=$(dirname "$DESTINATION")
if [ ! -d "$DEST_DIR" ]; then
    echo "Error: Destination directory '$DEST_DIR' does not exist" | tee -a "$LOG_FILE" >&2
    exit 4
fi
```

- Another validation for the disk space, to be certain that there is enough space for backups

```
# Validate that there is available disk space
REQUIRED_SPACE=$(du -s "${DIRECTORIES[@]}" | awk '{total+=$1} END {print total}')
AVAILABLE_SPACE=$(df "$DESTINATION" | awk 'NR==2 {print $4}')
if (( REQUIRED_SPACE > AVAILABLE_SPACE )); then
    echo "Error: Insufficient disk space at destination" | tee -a "$LOG_FILE" >&2
    exit 7
fi
```

- During my searching in the internet, I found a tool called **rsync**, I tried to use it but an error occured and I couldn't solve it, so I discarded it. But I read that it can be used to delete backup files in the backup directory if they were deleted from their main directory, this synchronization is a great advantage, but I did not needed it in my backup case, and I could not use it from the beginning

System Health Check Script

- **Purpose**: check the health of the system, this includes disk space, memory usage, running services, and system updates. The result will be a report with recommendations for user to do.
- **Features**:
  - Disk space check
  - Memory usage report
  - Running services status
  - Recent updates check
  - Simplified report contents
  - Easy-to-read
  - Can be sent for the user easily
- After writing the script, I added the bath of the file to `.bashrc` file so I can use the script as a command anytime from anywhere

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# alias for my backup script
alias backup-script='bash /home/hamza-damas/scripts/backup-script/backup-script.sh'

# alias for my system health report
alias sys-health-report='bash /home/hamza-damas/scripts/system-health/sys-health-report.sh'
```

- The same as the previous script, I added interactive prompts so the user can easily understand how to use the script, and add services that are critical to the system and should be running all the time

```
# Display the help message
function show_help() {
    echo "Usage: sys-health-report [critical_service1] [critical_service2] ... [critical_serviceN] "
    echo "Description: Generates a system health report and checks critical services"
    echo "If no arguments are provided, default critical services will be checked"
    echo
    echo "Options:"
    echo "  --help                    Show this help message and exit"
    echo "  [critical_serviceN]       Specify critical services names like [name.service]"
}

# Redirect all output to the console and log file
exec > >(tee "$log_file") 2>&1

# Check if --help is provided as the first argument
if [[ "$1" == "--help" ]]; then
    show_help
    exit 0
fi
```

```
hamza-damas@Hamza-Damas:~$ sys-health-report --help
hamza-damas@Hamza-Damas:~$ Usage: sys-health-report [critical_service1] [critical_service2] ... [critical_serviceN]
Description: Generates a system health report and checks critical services
If no arguments are provided, default critical services will be checked

Options:
  --help                    Show this help message and exit
  [critical_serviceN]       Specify critical services names like [name.service]

hamza-damas@Hamza-Damas:~$
```

- All the contents of the report will be on a log file in the same directory as the script, this gives me the ability to send the file and manipulate it with high flexibility

```
hamza-damas@Hamza-Damas:~$ sys-health-report
================================
        System Health Report
        Fri Jan 24 09:11:32 +03 2025
================================
Disk Space Check:

Total disk space: 1007.0 GiB
Used space: 2.0 GiB
Available space: 954.0 GiB
Usage: 1% full
Disk usage is within healthy limits.

================
Memory Usage Check:
```

- For disk space check, I used mainly `df` command to check for the available and used space

```
# Disk usage details for the root partition
disk_info=$(df -h / | awk 'NR==2')
total_space=$(echo "$disk_info" | awk '{printf "%.1f GiB", $2+0}')
used_space=$(echo "$disk_info" | awk '{printf "%.1f GiB", $3+0}')
available_space=$(echo "$disk_info" | awk '{printf "%.1f GiB", $4+0}')
disk_usage=$(echo "$disk_info" | awk '{print $5}' | sed 's/%//')
```

- I used `free` command to get the information of the memory usage, with processing the data and using only important values

```
total_mem=$(free -m | awk '/Mem:/ {print $2}')
used_mem=$(free -m | awk '/Mem:/ {print $3}')
mem_usage_percent=$((used_mem * 100 / total_mem))
echo "Memory usage: ${mem_usage_percent}% ($(printf "%.2f" "$
```

```
Memory Usage Check:

Memory usage: 18% (0.71 GiB used of 3.76 GiB total)
Memory usage is within healthy limits.
```

- I added recommendations if the disk space bacame more than **80%**, or the memory usage exceeds **90%**

```
# Check if disk usage exceeds 80%
if [ "$disk_usage" -gt 80 ]; then
    echo "Warning! Disk space usage is higher than 80%"
    echo "Recommendation: Clean up the disk space. Remove unnecessary files or expand yo
else
    echo "Disk usage is within healthy limits."
fi
echo ""
echo "=================="
```

```
# Check if memory usage exceeds 90%
if [ "$mem_usage_percent" -gt 90 ]; then
    echo "Warning! Memory usage is higher than 90%"
    echo "Recommendation: Consider closing unused applications or increas
else
    echo "Memory usage is within healthy limits."
fi
echo ""
echo "=================="
```

- In running services check, I used `systemctl` command to know what I need about services

```
# List all active services
readarray -t running_services < <(systemctl list-units --type=service --state=running --no-pager --no-legend

if [ -z "$running_services" ]; then
    echo "No services are currently running"
    echo "Recommendation: Start necessary services if required for your system's functionality."
else
    echo "The following services are currently running:"
    printf "  - %s\n" "${running_services[@]}"
    echo ""
```

```
==================
Running Services Check:

No arguments provided. Using default critical services: cron.service

The following services are currently running:
  - console-getty.service
  - cron.service
  - dbus.service
  - getty@tty1.service
  - polkit.service
```

- User can enter services names that need to run all the time, the script ensures that all of them are running, and if one of them was stopped, it will be running again

```
# Critical Services information
echo "Critical Services:"
echo ""
for service in "${critical_services[@]}"; do
    if ! echo "$running_services" | grep -qw "$service"; then
        echo "Warning: Critical service $service is not running."
        echo "$service is stopped. Attempting to restart..."
        sudo systemctl restart "$service"
        if [ $? -eq 0 ]; then
            echo "$service restarted successfully."
        else
            echo "Failed to restart $service. Please check manually."
        fi
    else
        echo "$service is running."
    fi
done
```

```
Critical Services:

Warning: Critical service cron.service is not running.
cron.service is stopped. Attempting to restart...
cron.service restarted successfully.

System Load:
 09:28:17 up  3:27,  1 user,  load average: 0.00, 0.02, 0.00

=================
```

- For system updates finally, I used the command `apt` , to list and view recent updates that are available for the user

```
# Check for available updates
updates=$(sudo apt list --upgradable 2>/dev/null | grep -v "Listing" | wc -l)

if [[ $updates -gt 0 ]]; then
    echo "There are $updates package(s) available for update."
    echo ""
    echo "Here is a categorized list of updates (Kernel, Security, and Other):"
    echo ""

    # Fetch list of upgradable packages
    upgradable_packages=$(sudo apt list --upgradable 2>/dev/null | grep -v "Listing")
```

- At the end of the report, the availabe updates will be listed based on their category, and some recommendations will be displayed for the user

```
  - vim-tiny/noble-updates,noble-security 2:9.1.0016-1ubuntu7.6 amd64 [upgradab
  - vim/noble-updates,noble-security 2:9.1.0016-1ubuntu7.6 amd64 [upgradable fr
  - xxd/noble-updates,noble-security 2:9.1.0016-1ubuntu7.6 amd64 [upgradable fr

Recommendations:
1. Run 'sudo apt upgrade' to apply the updates.
2. Run 'sudo apt dist-upgrade' for a full upgrade (if necessary).
3. Run 'sudo apt autoremove' to clean up unnecessary packages after upgrading.

================================
Health check completed.
hamza-damas@Hamza-Damas:~$
```

- In the future, I want to improve the script and add a feature to automate this check, and send an email to the user with the content of the report, highlighting the most important recommendations

Performance Analysis

- I was making sure to test the script part by part as I am writing it, and keep the modularity while building it, this methodology helps with testing as it reduce numbers of errors and contradictions in the script.
- In general, I did not feel that there is any "inefficiency" in my scripts, so I am not aware of any.

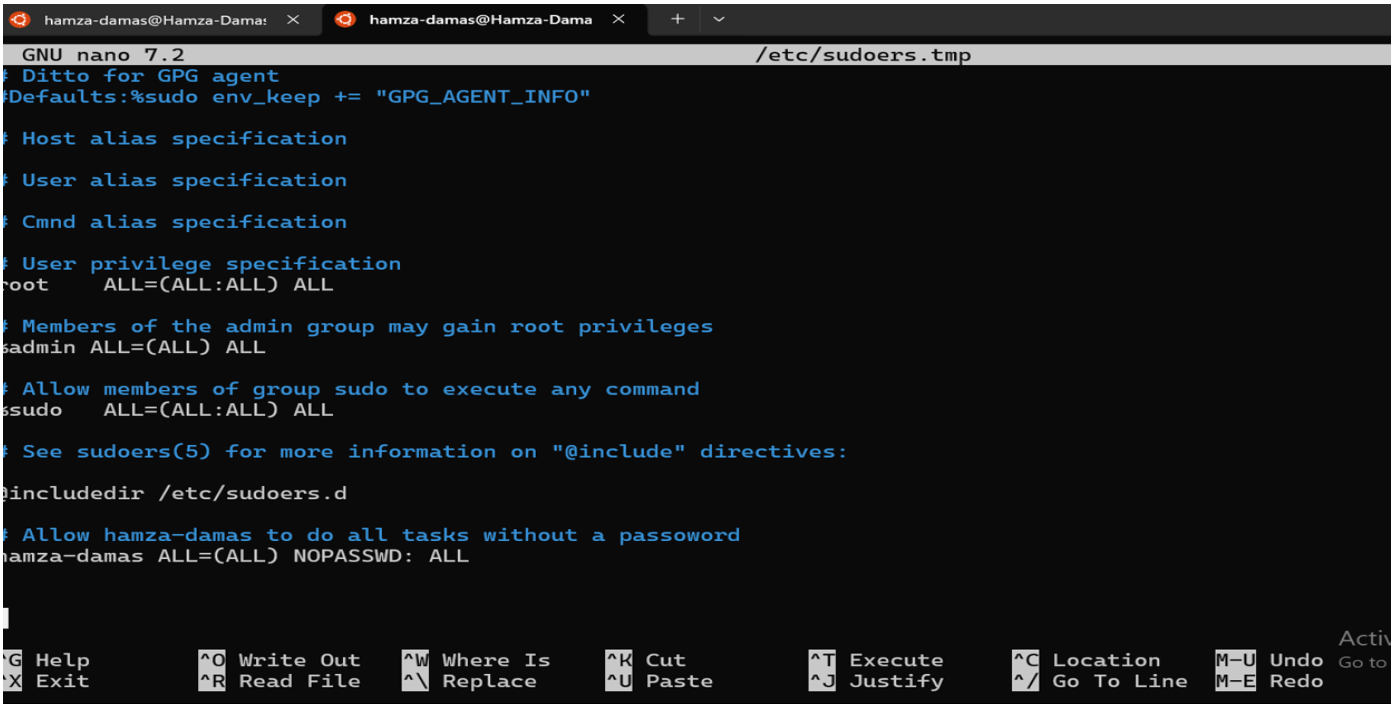
Optimizations and Impact

- One of the changes that I made to improve the script was when I noticed that the backup directory creates new directory for every minute, means that if two different backups happend in two different minutes within the same hour, there would be two directories with the same backup files. After noticing that I modified the script so only one directory is made in the day, and it contains only one backup, even more than one using of the script happend in the same day, it will keep the laest files

```
hamza-damas@Hamza-Damas:~$ cd /mnt/c/Users/HamzaWH/linux-practice/backup-script
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script$ ls
backup-second.sh  backup.sh  sources  targets
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script$ cd targets/
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets$ ls
daily-backups
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets$ cd daily-backups/
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets/daily-backups$ ls
20250122  20250123  20250124
hamza-damas@Hamza-Damas:/mnt/c/Users/HamzaWH/linux-practice/backup-script/targets/daily-backups$ cd 20250123
```

- I added also a mechanism to delete backups that are older than particular number of days, this days is an input from the user. This keeps the directories clean and contans only relevent data

- In system health report, I tried to update automatically all the needed updates, but I found that overwhelming, so I just wrote it to send a recommendation to the user
- I faced some problems with using `sudo` command, as I need to re-input my password everytime and this is not possible with automation scripts, so I edited the configurations to not typing any passwords, this may affect the security, but it was okay in my case



Conclusion

My experience with Linux and this scripting language was great, this was my first time trying new thing other than Windows, and it was so much easier.
I started think seriously in migrating to Linux for its simplicity, automation abilities, and open-source community.
I hope that this learning journey improves my skills and help me in my carrer in the future.