**Linux Administration:** User Access & System hardening

Objectives:

The overall objective of this project is to gain foundational Linux skills by configuring users, managing permissions, automating tasks, secure remote access and maintaining system services. All of this will be done on a terminal based Ubuntu Server Environment.

Purpose:

The purpose of this project is to demonstrate the ability to install and manage linux as a server. Then navigate and control the Linux file system. Manage users, group and permissions. Installing software and controlling services, Automating tasks with cron. Configure and secure SSH access then implement basic firewall rules.

Tools & Technologies:

Ubuntu Server 22.04 - Operating System to learn Linux

VirtualBox – Local Virtual Machine Virtualization

Terminal or Bash – Command Line Interface

Nano or Vim – File Editors

Ufw – Firewall configuration

Cron – Task Scheduling

Systemctl – Service Management

OpenSSH – Remote Access

Linux Commands: **System Setup & Update**

| **Command** | **Description** |
| --- | --- |
| sudo apt update | Updates package index |
| sudo apt upgrade -y | Upgrades installed packages |
| hostnamectl set-hostname newname | Changes the system hostname |
| ip a or ip addr | Shows system IP address |
| ping -c 4 google.com | Test network connectivity |

**User, Group and Permissions Management**

**User and Groups**

|  |  |
| --- | --- |
| adduser alice | Adds a new user (with password prompt) |
| passwd alice | Changes user’s password |
| deluser alice | Deletes a user |
| groupadd devs | Creates a new group |
| usermod -aG devs alice | Adds user to a group |
| groups alice | Shows groups a user belongs to |
| id alice | Shows UID, GID, and groups |

**Files and Permissions**

|  |  |
| --- | --- |
| mkdir /project | Creates directory |
| chown :devs /project | Changes group ownership |
| chmod 770 /project | Gives full access to owner/group only |
| chmod +t /project | Enables sticky bit (secure shared folder) |
| touch /project/file.txt | Creates a file |
| ls -l /project | Lists files with permissions |
| chmod 664 file.txt | Sets read/write permissions |
| stat file.txt | Displays detailed file info |

**Package & Service Management**

|  |  |
| --- | --- |
| apt install nginx -y | Installs nginx web server |
| systemctl start nginx | Starts nginx |
| systemctl stop nginx | Stops nginx |
| systemctl status nginx | Checks nginx status |
| systemctl enable nginx | Enables nginx at boot |
| apt remove nginx -y | Uninstalls nginx |

**SSH Configuration & Hardening**

|  |  |
| --- | --- |
| cp /etc/ssh/sshd\_config /etc/ssh/sshd\_config.bak | Backs up SSH config |
| nano /etc/ssh/sshd\_config | Edits SSH config file |
| systemctl restart ssh | Restarts SSH service |
| ufw allow 2222/tcp | Opens custom SSH port |
| ufw allow 80,443/tcp | Opens web ports |
| ufw enable | Activates firewall |
| ufw status | Checks firewall status |

**Scheduled Tasks (cron)**

|  |  |
| --- | --- |
| crontab -e | Edits current user’s crontab |
| crontab -l | Lists current cron jobs |
| echo "hello" >> test.txt | Appends output to file |
| date | Prints current date/time (useful in scripts) |

**Disk and Mounts**

|  |  |
| --- | --- |
| lsblk | Lists block devices and partitions |
| mount /dev/sr0 /mnt/test | Mounts disk to directory |
| umount /mnt/test | Unmounts the disk |
| mkdir /mnt/test | Creates mount point |
| nano /etc/fstab | Edits file for persistent mounts (use with caution!) |

**Bash Utilities**

|  |  |
| --- | --- |
| whoami | Shows current user |
| history | Displays command history |
| clear | Clears the terminal |
| sudo | Runs command with root privileges |
| reboot | Reboots system |
| shutdown now | Shuts down system |

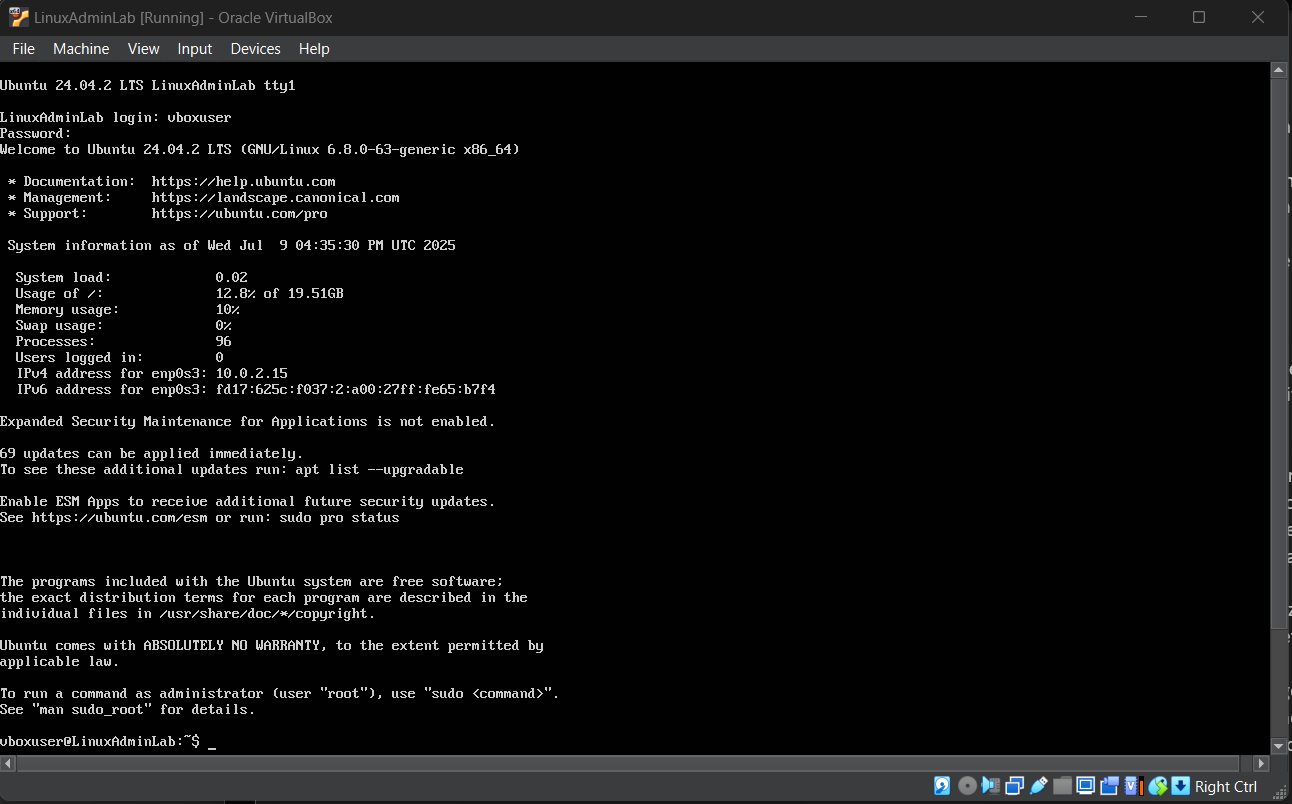
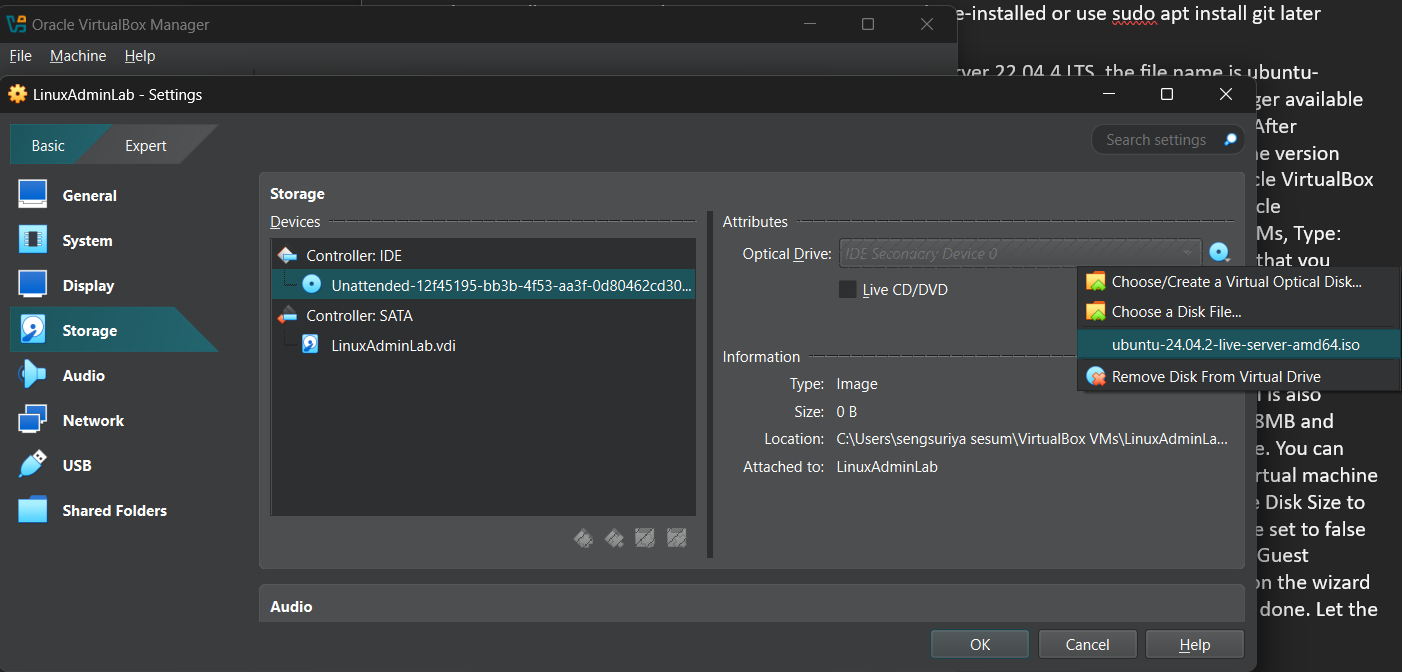
**Documentation (Git & GitHub)**

|  |  |
| --- | --- |
| git init | Initializes a new repo |
| git add . | Adds all files to staging |
| git commit -m "Initial commit" | Commits changes |
| git remote add origin <repo-url> | Connects to GitHub |
| git push -u origin main | Pushes project to GitHub |

**Installations:**

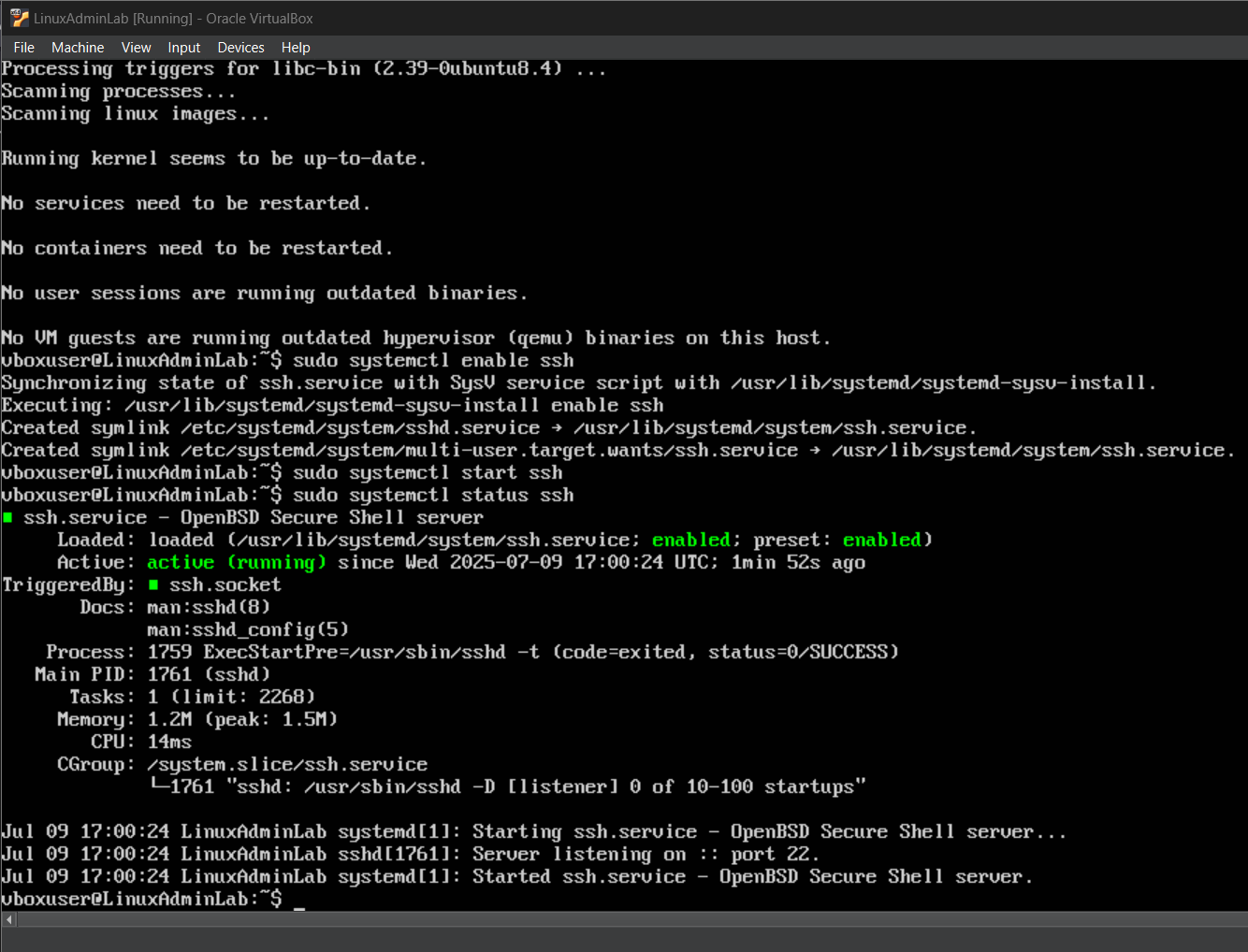
| Tool | Description | Download Link |
| --- | --- | --- |
| VirtualBox | Free hypervisor to run VMs | <https://www.virtualbox.org/wiki/Downloads> |
| Ubuntu Server ISO | The operating system (Linux) | <https://ubuntu.com/download/server> |
| (Optional) Git | For documentation version control | Pre-installed or use sudo apt install git later |

The first thing that we will need to do is install Ubuntu Server 22.04.4 LTS, the file name is ubuntu-22.04.4-live-server-amd64.iso The link is <https://ubuntu.com/download/server> If its no longer available then the latest version LTS or Long term support will be the one we can download instead. After downloading it, head over to <https://www.virtualbox.org/wiki/Downloads> and download the version according to your current OS version like windows for example then install it. Once the Oracle VirtualBox Manager is installed correctly, we can begin creating a Linux Virtual Machine. Open the Oracle VirtualBox, press new, Name: LinuxAdminLab, Folder: C:\Users\Account name\VirtualBox VMs, Type: Linux, Subtype: Ubuntu, Version: Ubuntu 64-bit and the selected ISO Image will be the one that you downloaded from the Ubuntu site. Detected OS type: Ubuntu (64-bit). This OS type can be installed unattendedly. The install will start after this wizard is closed. Press Next. Username: vboxuser and Password: onetwothree , Hostname: LinuxAdminLab, Domain Name: myguest.virtualbox.org and Guess Additions ISO: C:\Program Files\Oracle\VirtualBox\VBoxGuestAdditions.iso Press next, You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count. Enabling EFI is also possible. In this case we will just leave it to its default settings which is BaseMemory at 2048MB and processors 1 CPU. Press next, If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select an existing one. Alternatively, you can create a virtual machine without a virtual hard disk. In this case we will select create a Virtual Hard Disk Now, set the Disk Size to 20 GB, if available select type as VDI (VirtualBox Disk Image). The following items need to be set to false or if it is already done so, Skip unattended Install, Product Key, Install in Background, Install Guest Additions, EFI enabled set to false and pre-allocated full size is set to false too. Press finish on the wizard to create the new linux Virtual Machine, it should automatically launch the VM once that is done. Let the VM load on its own since it needs time to set up since there is a lot of installations need to be done automatically from the linux side of the VM. In the mean time, head to the Oracle VirtualBox Manager, under Machine go to settings. Head to storage under Controller: IDE, press the empty CD Icon, on the right press the disk icon and press choose a disk file, select the ubuntu-24.04.2-live-server-amd64.iso file then start the VM and the Ubuntu Server installation will launch.

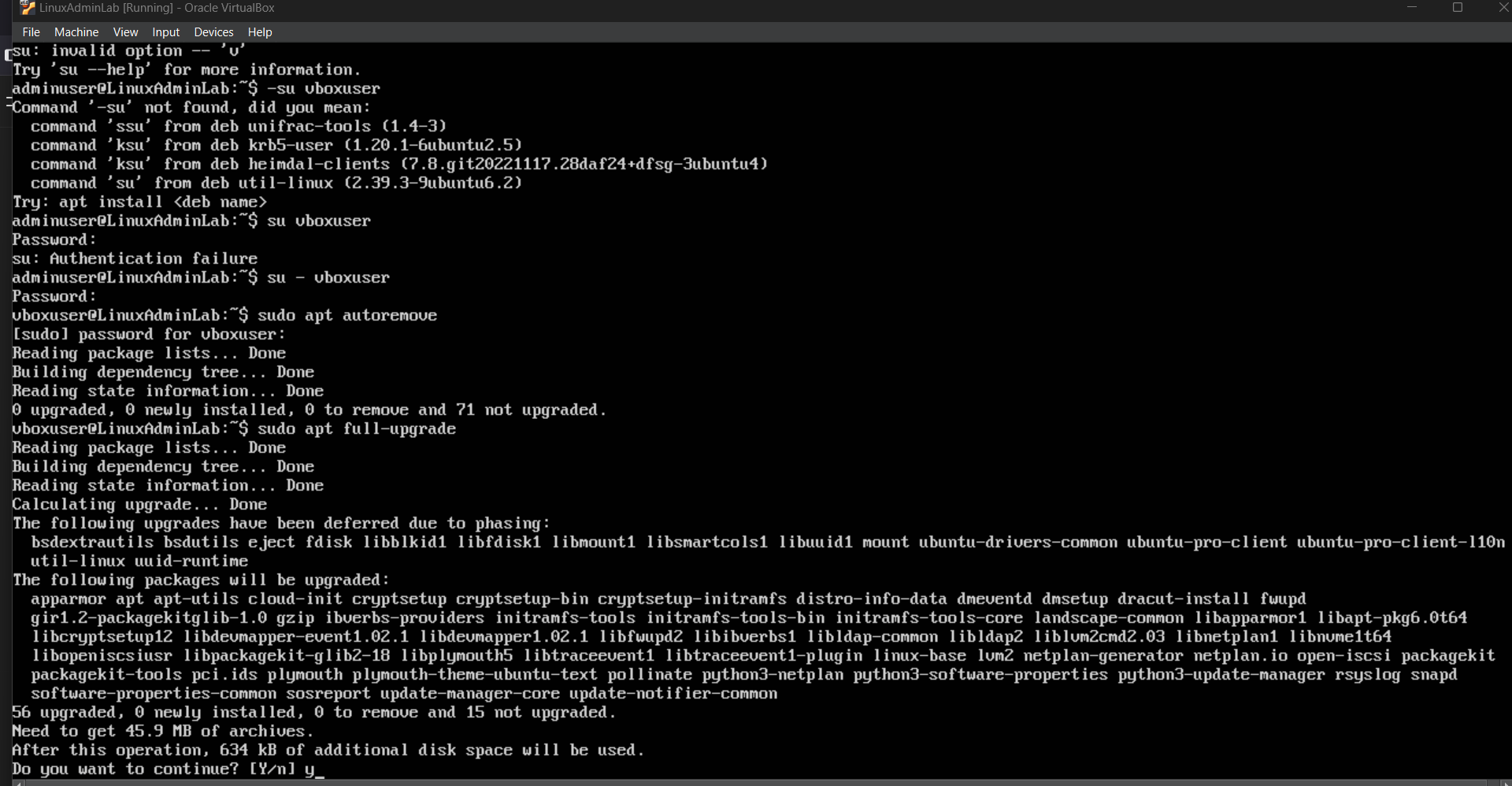


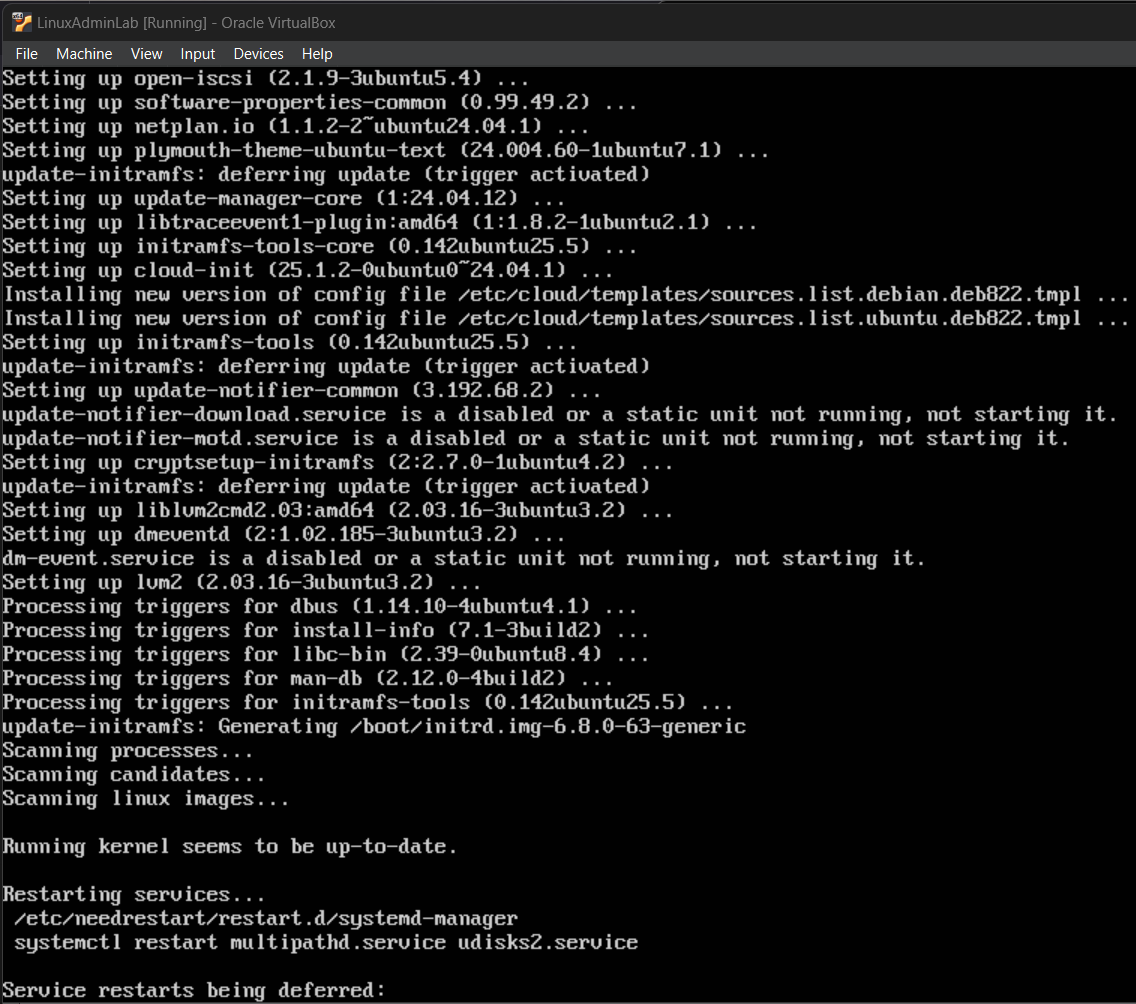
Once the installation and loading process is complete on the VM side, enter the login into you created and this is what the screen should look like. Since the Oracle VM manager automatically installed the linux and launch it. We need to verify a few things before we can continue. Such as Checking to see if SSH is installed and running, did we create a non-root Admin user, set up a hostname, is the VM using NAT and connected to the internet, is Ubuntu fully updated and is there enough disk size. Start by checking to see if the OpenSSH server was installed, the reason is because the project relies on SSH for remote management and hardening. Enter this command to check and see if it is installed and running: sudo systemctl status ssh. If it says **Unit ssh.service could not be found**, then enter each of the following commands: sudo apt update, sudo apt install openssh-server -y, sudo systemctl enable ssh, sudo systemctl start ssh

Once that is done, go ahead and enter sudo systemctl status ssh to check again and see if the SSH service exists by reentering the sudo systemctl status ssh in case the sudo systemctl start ssh command does not work. This is the output that you should receive if you have entered all of the previous commands in sequences: For better viewing go to view 🡪 Virtual Screen 1 🡪 Select Scale to 150%.

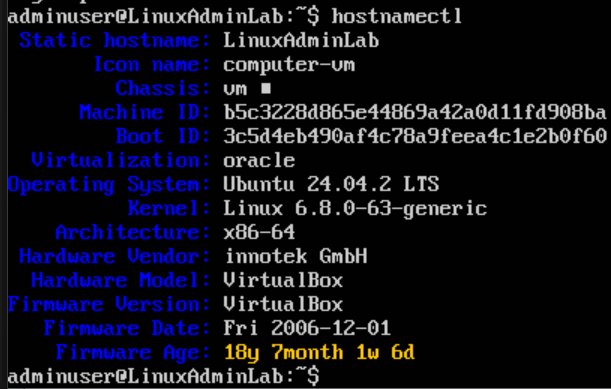


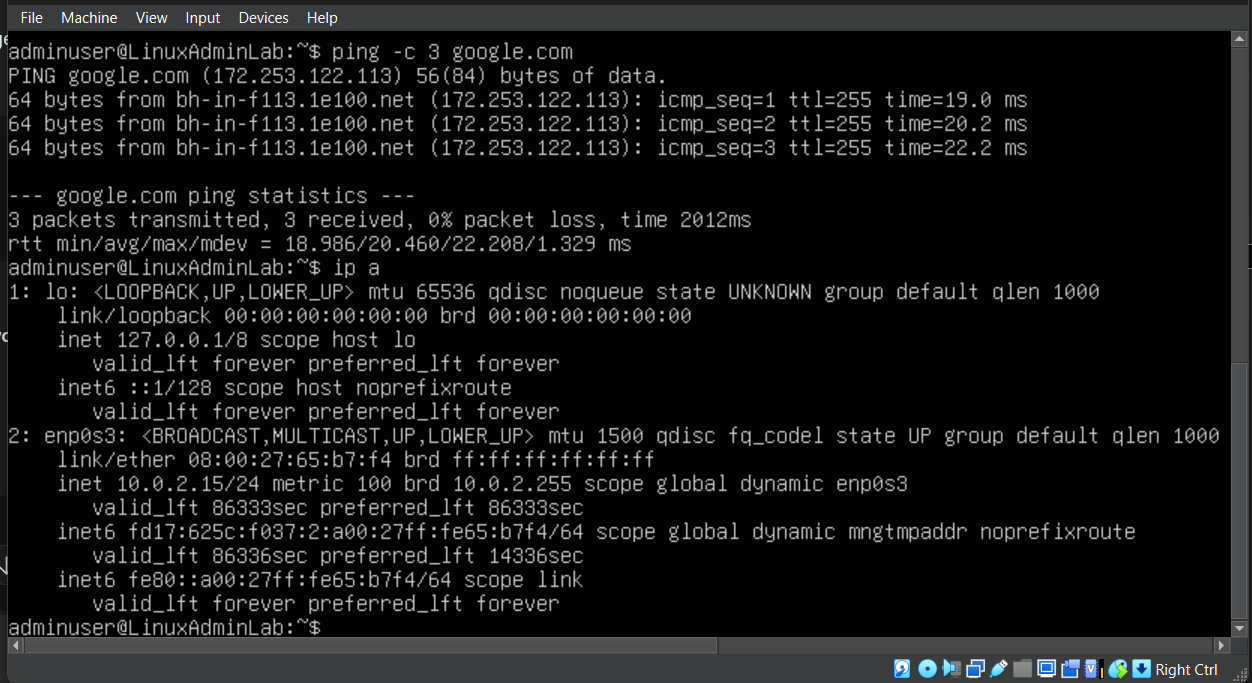
Since the output generates: loaded: loaded /usr/lib/systemd/system/ssh.service ; enabled; preset: enabled, active: active (running) since today, triggered by ssh.socket. This means that OpenSSH is fully installed and running, it’s set to start at boot, it is triggered by ssh.socket which is a system optimization and the VM is now SSH ready. Now that we are done with the SSH part we need to create a non root admin user because we should not run everything as root. Enter command: whoami. Since the root login was created during the installation of the linux VM we will create a separate admin user. Start by entering the following commands: sudo adduser adminuser and enter sudo usermod -aG sudo adminuser. Next switch by entering su – adminuser as the new user. If prompted to enter a new password then try fourfivesix. If prompted to enter full name just enter testing, room number: 1, work phone: 1111111111, home phone: 1111111111, Other: none, press y to continue. Once that is done the output should say add new user ‘adminuser’ to supplemental / extra groups ‘users’, adding user ‘adminuser’ to group ‘users’. Login for the new user: adminuser, password: fourfivesix. Keep in mind the “su – username” command is useful to switch user accounts in the linux VM. Keep in mind that if it takes too long for the VM to display anything after restart or relaunch then try going to settings under system and increase the processor unit to 3 CPU, save and restart the VM and launch it. After logging in, if there are any updates enter apt list –upgradable and enter sudo apt full-upgrade. Also enter sudo apt autoremove to clean up any old packages.





We just need to check to see If we have set up a hostname, enter hostnamectl and you should see this type of output if you set everything up properly when creating the Linux VM.



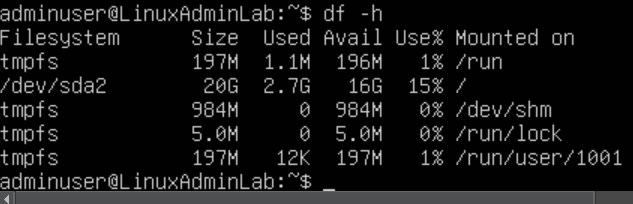


To check and see if the VM is connected to the internet and or using Network address translator just enter the following commands: ping -c 3 google.com, ip a

Then these are the results we should be seeing as shown in the image. Alternatively, you can check by going to VirtualBox 🡪 VM Settings 🡪 Network 🡪 Check Enable Network Adapter 🡪 Attached to NAT

In case the VM is not connecting to the internet just make sure the VM network settings is correct then reboot the entire VM before restarting the Linux OS, Alternatively you can enter sudo systemctl restart systemd-networkd to reboot the VM or restart the networking of the VM.

In order to find out if the Ubuntu was fully updated, you need to enter sudo apt update && sudo apt upgrade -y but do note that the VM will not let you perform specific high level administration commands unless you are using the root user account instead of a non-admin user account which is why you need to enter command su – “root user account name” in order to proceed. To Check and see if there is enough disk size just enter this command: df -h



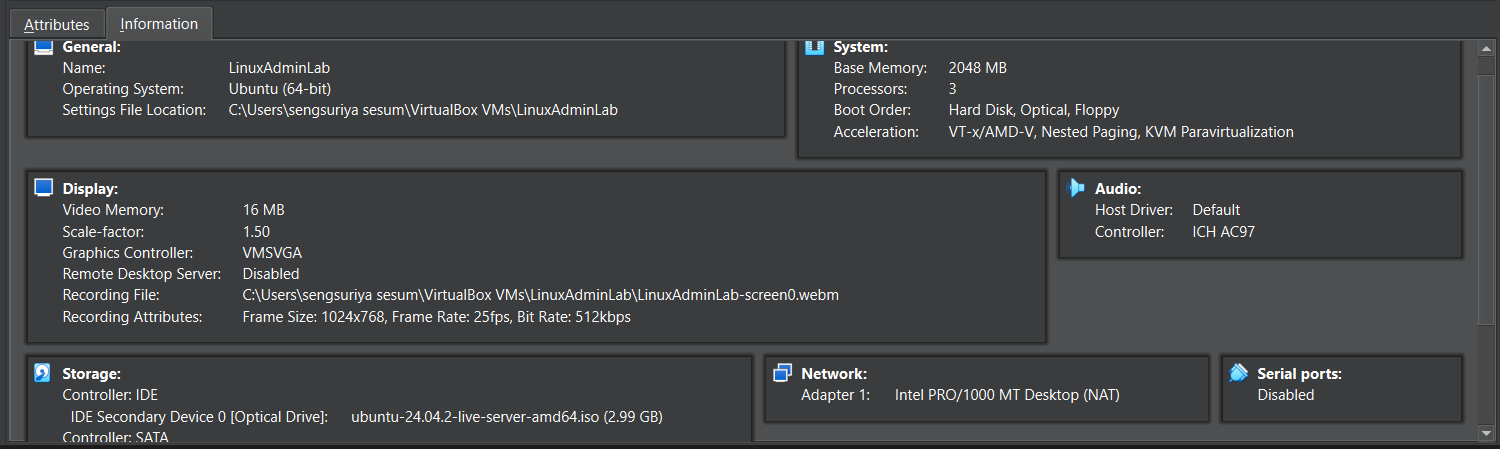
| **Item** | **Check It** | **Fix If Needed** |
| --- | --- | --- |
| OpenSSH installed | systemctl status ssh | Install it |
| Admin user created | whoami | adduser + usermod |
| Hostname set | hostnamectl | hostnamectl set-hostname |
| Internet working | ping google.com | Check NAT |
| System updated | apt update && upgrade | Run it now |
| Disk size 20–40GB | df -h | Reinstall if too small |
| SSH port forwarding | Optional | Set up in VirtualBox |

The last few things we need to do before we move on to the next phase from System Preparation is to install essential utilities through this command:

sudo apt install curl git nano tree net-tools -y && sudo apt install htop zip unzip -y

Once this is done it should let you know that the running kernel seems to be up to date.

In order to create a VM Snapshot or Backup Point start by turning off the VM then relaunch the Oracle VirtualBox Manager 🡪 Select VM LinuxAdminLab 🡪 Machine 🡪 Tools 🡪 Take Snapshot 🡪 Attributes Name Clean Phase 1 Base and Description as "System updated, SSH configured, admin user ready" 🡪 Press take to take snapshot. This will take a snapshot of all of the informational specifications about our VM in its current state. Shared Folder: none, USB Controller: OHCI, EHCI. Device Filters: 0 active



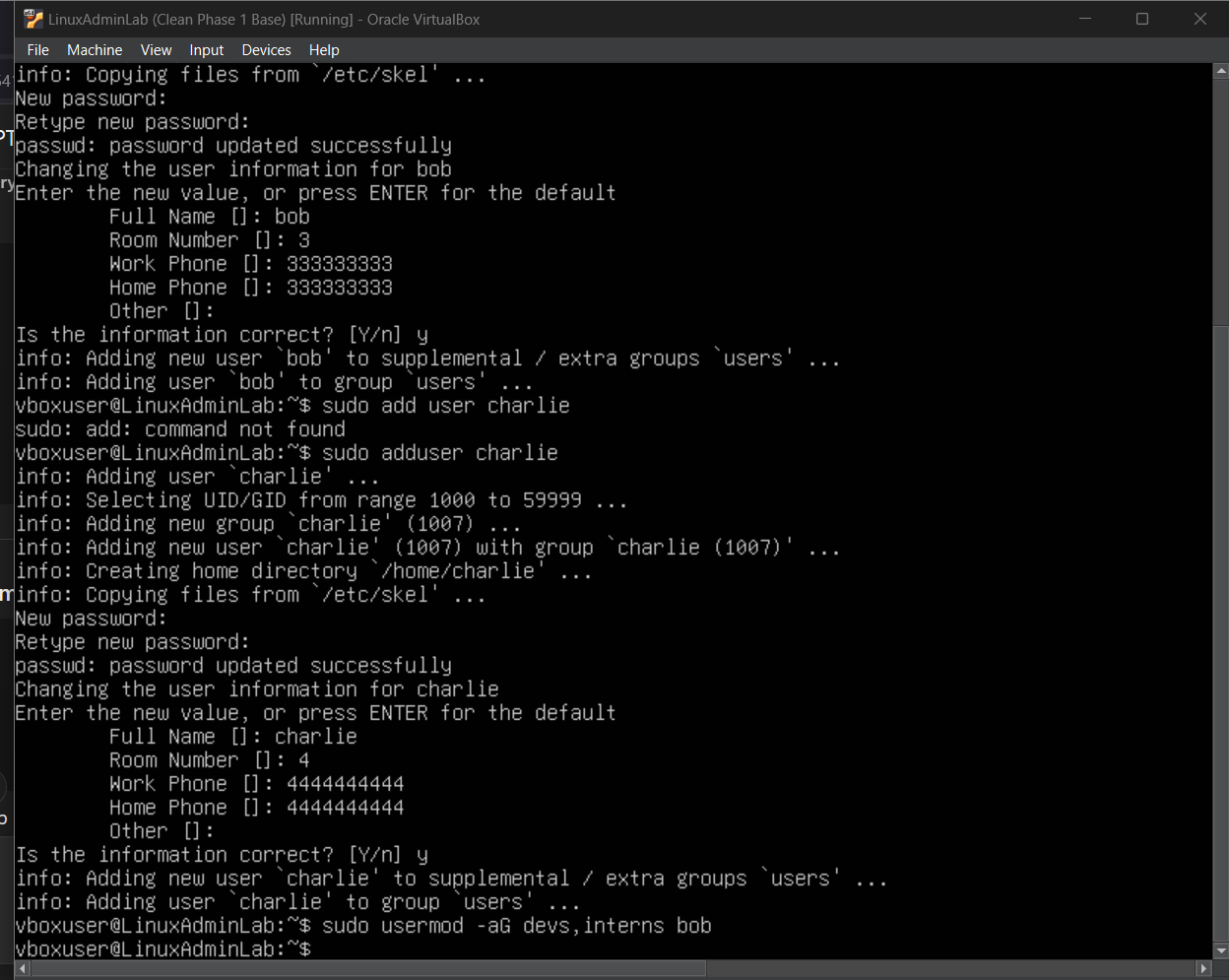
The Phase 1: System Preparation should contain the following tasks completed

| **Checklist** | **Status** |
| --- | --- |
| System updated and secured | ✅ |
| Admin user with sudo rights | ✅ |
| Networking & ping working | ✅ |
| OpenSSH server running | ✅ |
| SSH access tested (optional) | ✅ |
| Helpful tools installed | ✅ |
| VM snapshot taken | ✅ |
| UFW firewall prepared (optional) | ✅ |

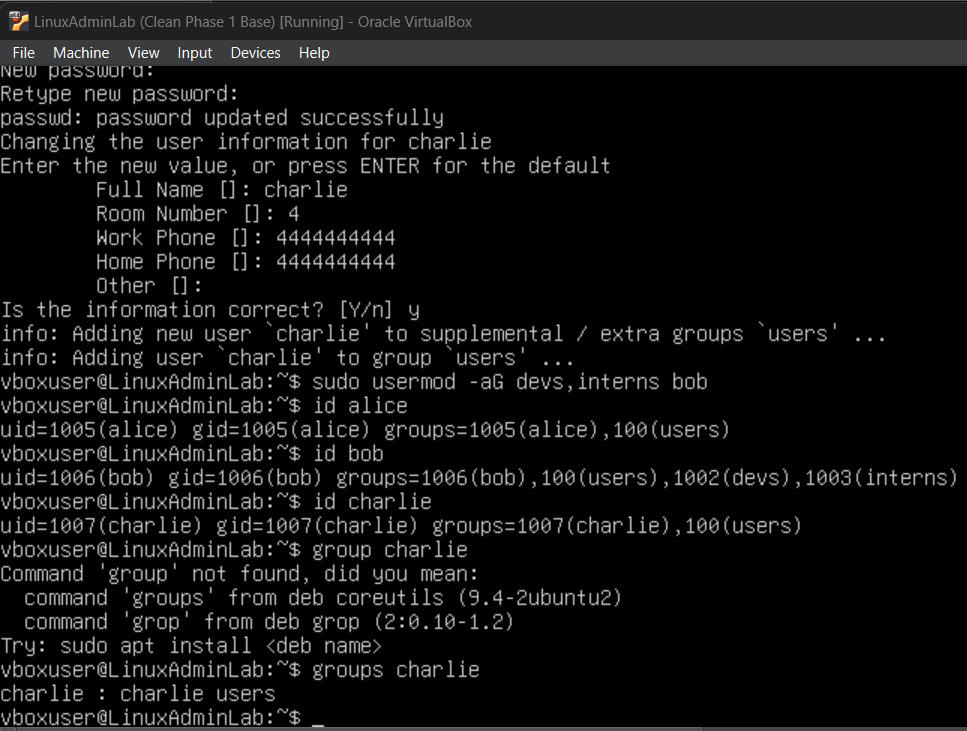
Starting with Phase 2: User, Group and Permission Management

| **Category** | **Examples** |
| --- | --- |
| Users & Groups | adduser, usermod, groupadd, passwd, deluser, delgroup |
| Permissions | chmod, chown, ls -l, umask |
| Shell Access | /bin/bash, /usr/sbin/nologin |
| File Ownership | Users vs. Groups, recursive permission edits |
| Account Restrictions | Expiry dates, password locks |
| Shared folders & home dirs | /home/, /etc/skel, ACLs |
| Security Practices | Principle of least privilege |
| Step | Task |
| 1 | Create multiple users and groups |
| 2 | Assign users to groups (primary & supplementary) |
| 3 | Set and test file/folder permissions |
| 4 | Set default permissions (umask) |
| 5 | Restrict shell access for specific users |
| 6 | Create a shared directory for one group |
| 7 | Set user expiration, password policies |
| 8 | Delete users & cleanup |

The objectives of phase 2 is to create users and groups, confirm users and group memberships, set file permissions for users, set default permissions, restrict shell access, create shared group directory, set user expiration and password aging then delete test users and groups. Switch to root user vbouser then begin the first step by running the following commands: sudo groupadd devs, sudo group add interns and sudo groupadd suspended, this is for testing purposes. Next enter sudo adduser alice then set password as 11111 and enter any random user information it asks. Do the same for sudo adduser bob and sudo adduser Charlie. Bob password will be 22222 or 33333 and Charlie password is 33333.



After that you should enter the command sudo usermod -aG devs, interns bob. When entering the ID or groups next to a name this is the result.



The next step is to set file permissions for the users.

Create test files: sudo -u alice touch /home/alice/testfile.txt

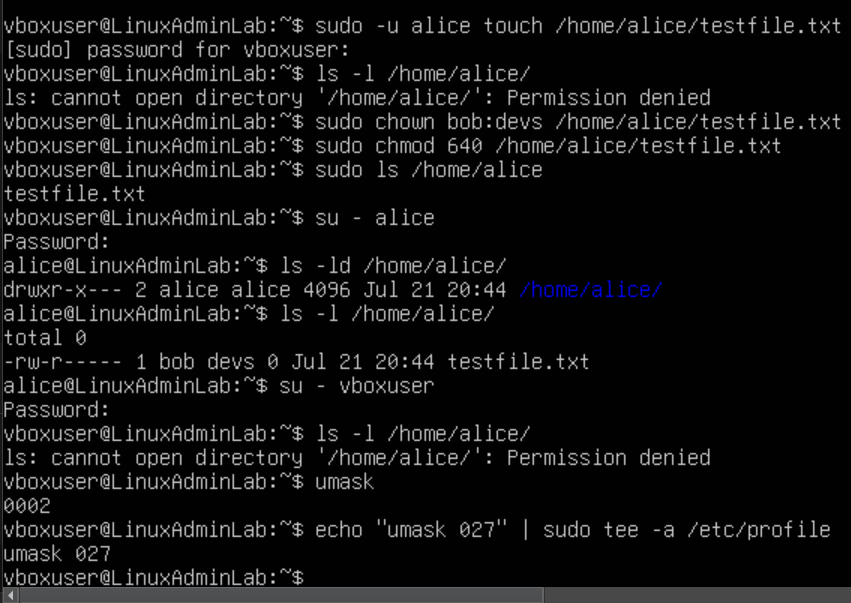
Check permissions: ls -l /home/alice/

Change ownership: sudo chown bob:devs /home/alice/testfile.txt

Modify permissions: sudo chmod 640 /home/alice/testfile.txt

640 = owner read/write, group read, others none

If you get ls: cannot open directory '/home/alice/': Permission denied is caused by linux’s security model which is normal because it protects each user’s file from being accessed by other users unless the user is alice or perhaps using sudo. For some odd reason even with root user permission, linux still prevents the root user from accessing another user’s file.



View current umask: umask, Set umask for user: echo “umask 027” | sudo tee -a /etc/profile

027 = owner full, group read/execute, others no access

Restrict shell access:

Remove login access from a suspended user: sudo usermod -s /usr/sbin/nologin Charlie

Lock a user’s password: sudo passwd -l Charlie

Create Shared Group Directory:

Create a shared folder for the devs group:

sudo mkdir -p /opt/devshare

sudo chown root:devs /opt/devshare

sudo chmod 2770 /opt/devshare

2 🡪 Sets the “sticky” group ID for files inherit group

Avoid entering these commands because it could cause an error with the VM to where the d bus service wont work anymore when the VM tries to launch which prevents you from logging in anymore even after restarting pc and restarting the VM. If that happens make sure to press restore current snapshot to restore the VM back to its original working state.

The safe setup for /opt/devshare directory:

Group name = devs

Users = alice, bob, Charlie

Directory Path = /opt/devshare

Create group: groupadd devs

Add users to the group: usermod -aG devs alice, usermod -aG devs bob

Verify by entering: Groups alice

Once that is all done enter the following commands using safe permissions:

Mkdir /opt/devshare

Chown root:devs /opt/devshare

Chmod 2770 /opt/devshare

Note: if linux says permission denied then entering sudo next to each command should work

The 2770 permission means the following:

2 🡪 Sets the **setgid** bit (shared group ownership on new files) such as devs

7 🡪 Full permissions for **owner** (read, write, execute)

7 🡪 Full permissions for group

0 🡪 No access for others

This makes /opt/devshare a secure shared directory between users in the same group.

Recap on what to avoid:

* **Never** chmod or chown entire system directories like /usr/bin, /opt, or / unless you’re 100% sure
* Avoid running chmod or chown without verifying the correct syntax, especially as root
* Always **snapshot** your VM before experimenting with system-level permissions

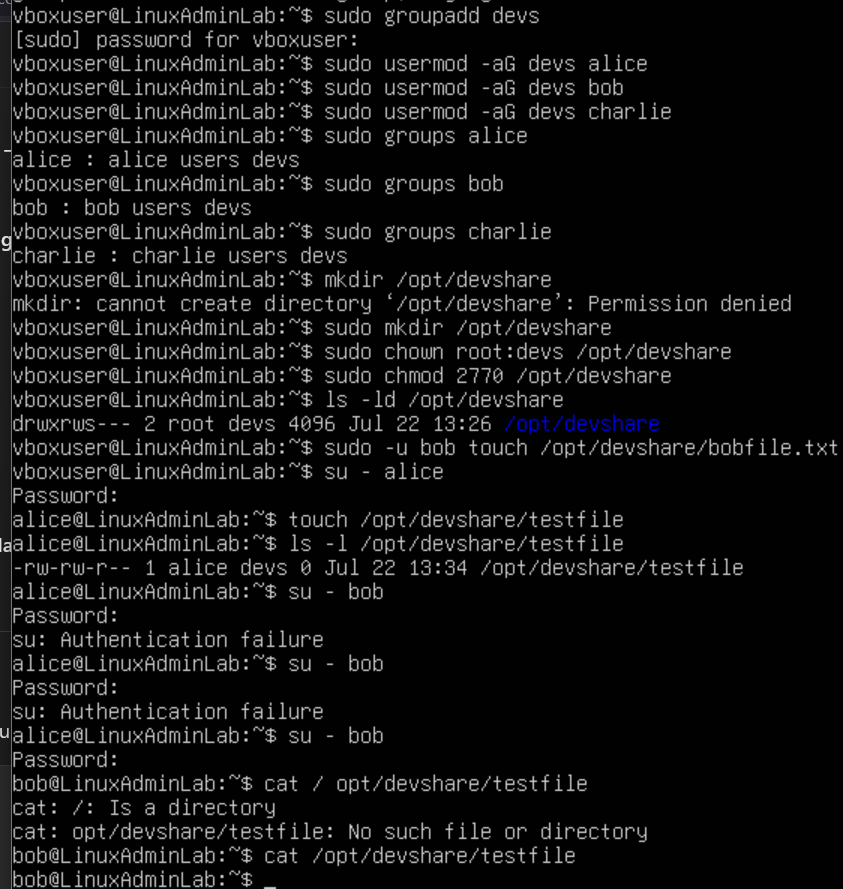
Only group members can write/read

Test file creation: sudo -u bob touch /opt/devshare/bobfile.txt

Switch to alice: su – alice, create a test file: touch /opt/devshare/testfile

check file ownership: ls -l /opt/devshare/testfile, switch to bob: su – bob

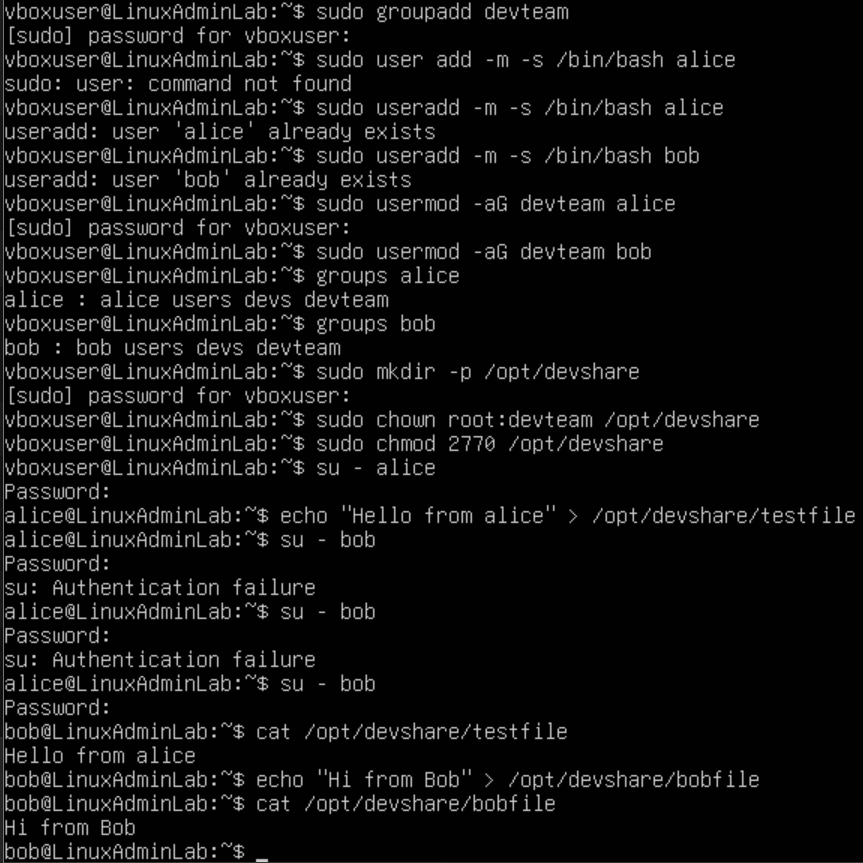
access file: cat /opt/devshare/testfile



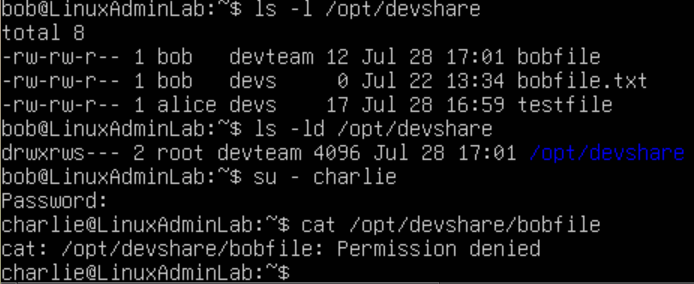
**Phase 3**: Shared Group Directory Configuration

The main objective is to set up a secure shared directory where members of a specific group can create, view, edit and share files with each other while others are restricted. Tools: Ubuntu, terminal, root or Sudo privileges. The steps that will be involved in this phase is creating the group, create users and assign them to a group, create shared directory, set group ownership to devteam, set permissions for collaborative access, switch to user (or open multiple terminals or SSH sessions) and verify permissions.

To begin start by creating a group that will have shared access to the directory by entering the following command: Sudo groupadd devteam otherwise the message will prompt groupadd: group ‘devteam’ already exists. Once that is done, begin creating two users and assign them to group. As we can see we already created alice and bob as users and have passwords for them. Verify what group each user is in by entering: groups (user). Example: Alice : Alice users devs devteam. Next to create a shared directory enter: sudo mkdir -p /opt/devshare. In order to set group ownership to devteam enter this command: sudo chown root:devteam /opt/devshare This causes the group devteam to own the directory (even though the root user owns the user). In order to set permissions for collaborative access, enter this command: sudo chmod 2770 /opt/devshare For context: 2 = **Setgid** bit — makes all new files inherit the group devteam. 7 = full access for owner (root), 7 = full access for group devteam and 0 = no access for others. Once all of this is done, enter each of the following commands: su – alice, Password: 11111, enter echo “Hello from Alice” > /opt/devshare/testfile, enter su – bob, cat /opt/devshare/testfile and enter echo “Hi from bob” > /opt/devshare/bobfile.

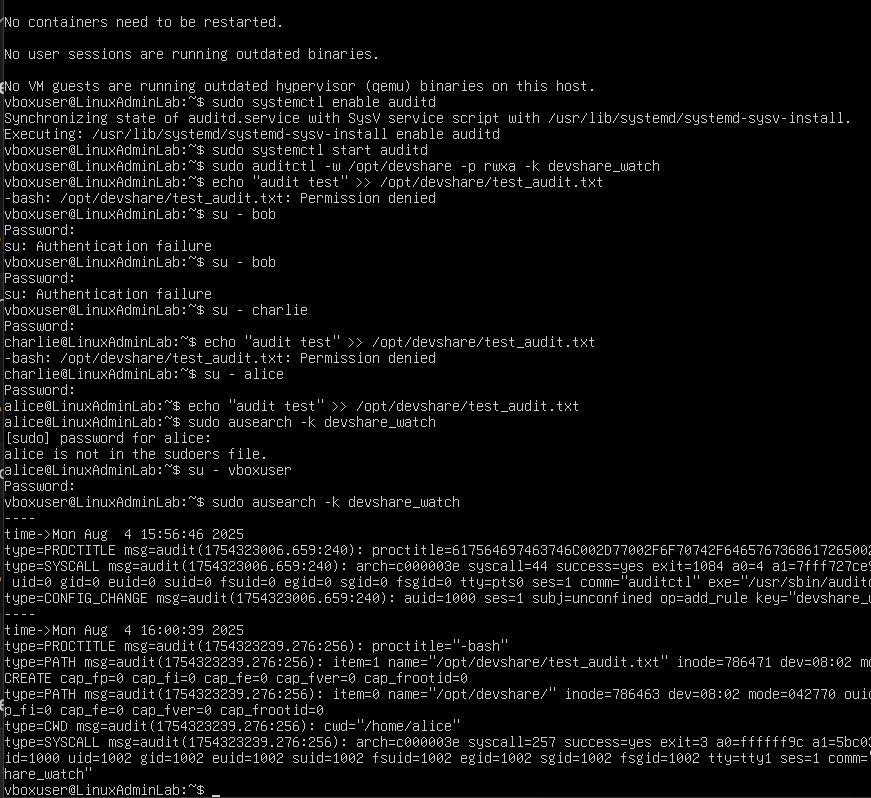


The next thing we need to do is check ownership and permissions by using this command: ls -l /opt/devshare then to check the directory itself enter this: ls -ld /opt/devshare, these are the outputs we should have here: but keep in mind a non-group user should be denied access if they try to access it.

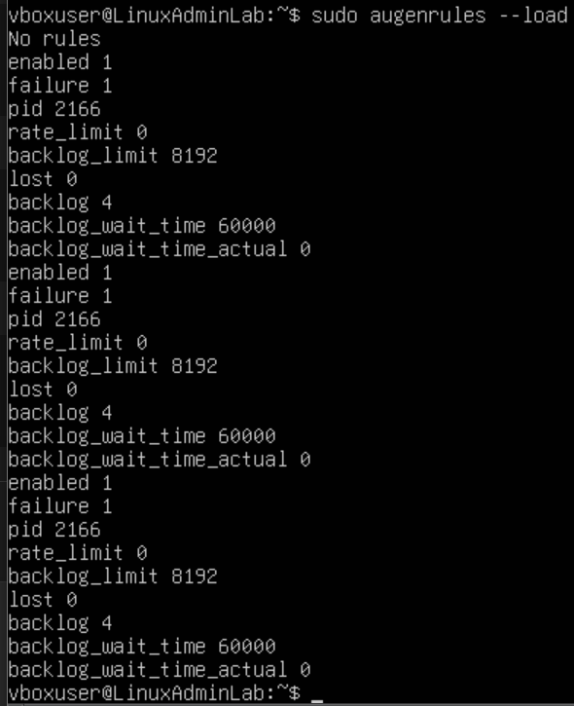


**Phase 4:** Fine-Grained file permissions & Basic Audit Logging via Ubuntu

Fine grained control over shared files in /opt/devshare, means that only authorized users can access or modify files, file changes and access are trackable via basic audit logs using auditd. This ensures proper security and accountability within shared workspaces. Auditd is audit daemon for file access logging and root access is required. The steps needed to be completed is setting up user umask to avoid world readable files, apply default ACLs for devteam group inheritance, test access restriction for non-group users, install and start auditd, create audit rule to monitor devshare access, test the logging, and make audit rules persistent after reboot. The first thing we need to do is edit the global bash profile to tighten default permissions for new files, by entering sudo nano /etc/profile then add this line at the bottom: umask 007, press ctrl + x to exit, save it and reboot the VM. Once this is done, enter the following commands: sudo setfacl -d -m g::rwx /opt/devshare, sudo setfacl -d -m o::--- /opt/devshare. Doing this will ensure all new files will inherit rwx for group devteam and it can not be accessible to others. If you receive an error: sudo: setfacl: command not found thn this means the acl package is not installed. This can be resolved by entering sudo apt update, sudo apt install acl -y. Then enter those sudo setfacl commands as previously mentioned. These commands will set default group permissions to full rwx and “others” permissions to none. Once that is done we will switch to a non group user and test it out. Such as Charlie (Password: 33333) then enter cat /opt/devshare/testfile and it should say permission denied which means ACLs and group permissions are working. Next install and start auditd by entering the following commands: sudo apt update, sudo apt install auditd audispd-plugins -y, sudo systemctl enable auditd, sudo systemctl start auditd. Once that is done enter sudo auditctl -w /opt/devshare -p rwxa -k devshare\_watch, -w = watch directory, -p rwxa = log read, write, execute, attribute changes, -k = add a key for easy filtering. Once that command is entered, switch to alice (password: 11111) then run the following command: echo “audit test” >> /opt/devshare/test\_audit.txt and enter this command to check the audit logs: sudo au search -k devshare\_watch



A formatted command would be sudo aureport -f | grep devshare. In order to make audit rules persist we will edit by entering sudo nano /etc/audit/rules.d/devshare.rules, then enter this -w /opt/devshare -p rwxa -k devshare\_watch, exit and save it then reload by entering sudo augenrules –load. Then this is the result we should be expecting:



**Phase 5: Automating Permissions & Scheduled Audits**

The objective is to implement auditing and mmonitoring using linux tools so an admin can track system events, user actions and changes to important files or directories. This is essential for system security and compliance.

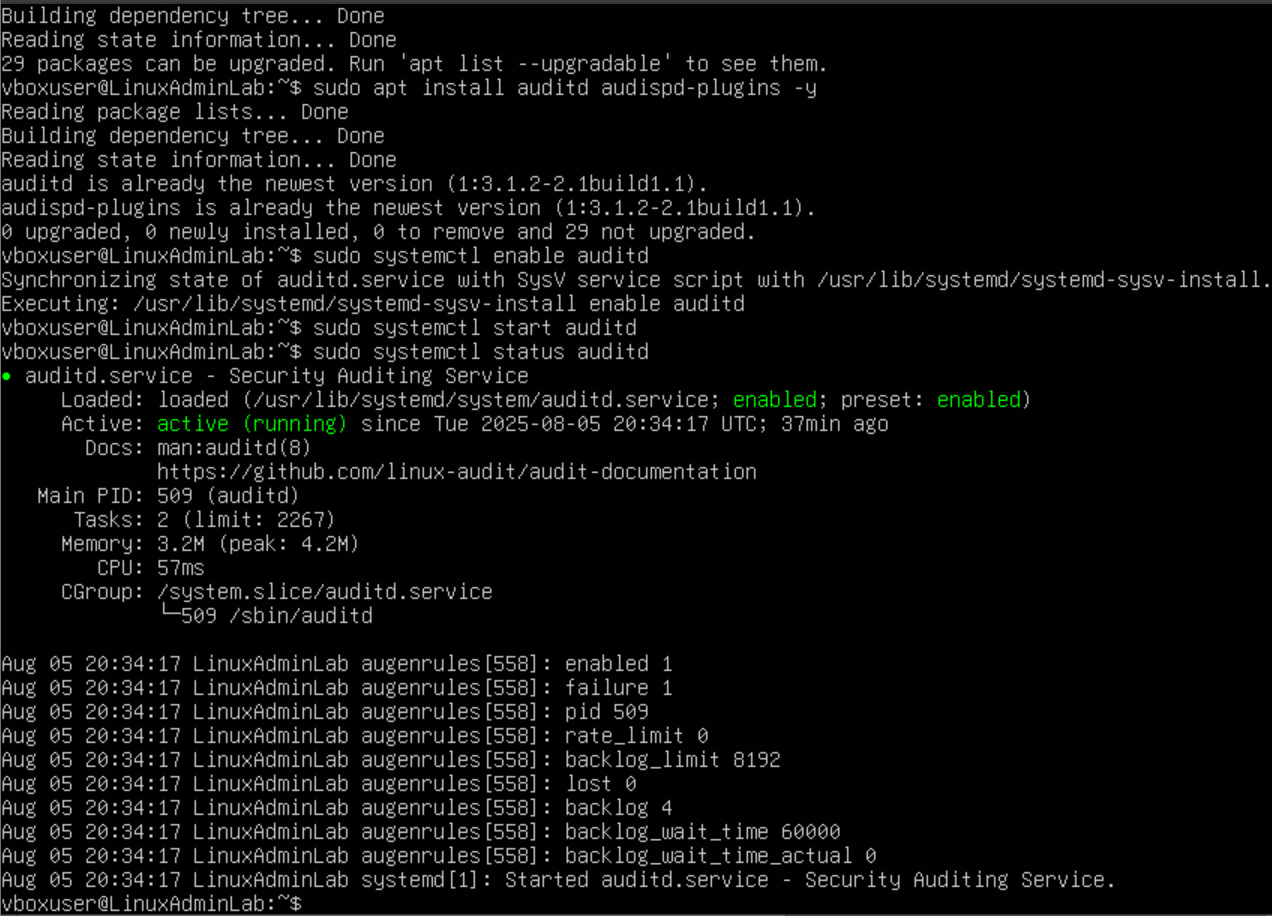
auditd – the Linux Auditing Daemo

ausearch, aureport – query tools for audit logs

ufw – uncomplicated firewall (optional for logging blocked access)

rsyslog or journald – for system logs

The steps that will be taken are the following: install audit system, enable and start auditd, configure basic audit rules, test audit logging and add real time log forwarding. The first thing we will do is install the audit system by entering the following commands: sudo apt update, sudo apt install auditd audispd-plugins -y. auditd = Audit daemon that records system activities, audispd-plugins = Extends logging functionalities. Then after the installation we will enable and start auditd by entering the following commands: sudo systemctl enable auditd, sudo systemctl start auditd. Then to confirm its running enter: sudo systemctl status auditd. Then this is the result we should be seeing after entering those commands:



Next we will configure basic audit rules by edit or create rules through this command: sudo nano /etc/audit/rules.d/audit.rules

Then ensure that this is written into the file:

# Monitor changes to /etc/passwd and /etc/shadow

-w /etc/passwd -p wa -k passwd\_changes

-w /etc/shadow -p wa -k shadow\_changes

# Monitor access to the shared directory

-w /opt/devshare -p war -k devshare\_monitor

# Monitor sudo command usage

-w /usr/bin/sudo -p x -k sudo\_activity

-w = watch path, -p = permissions to watch (read, write, execute, attribute change)

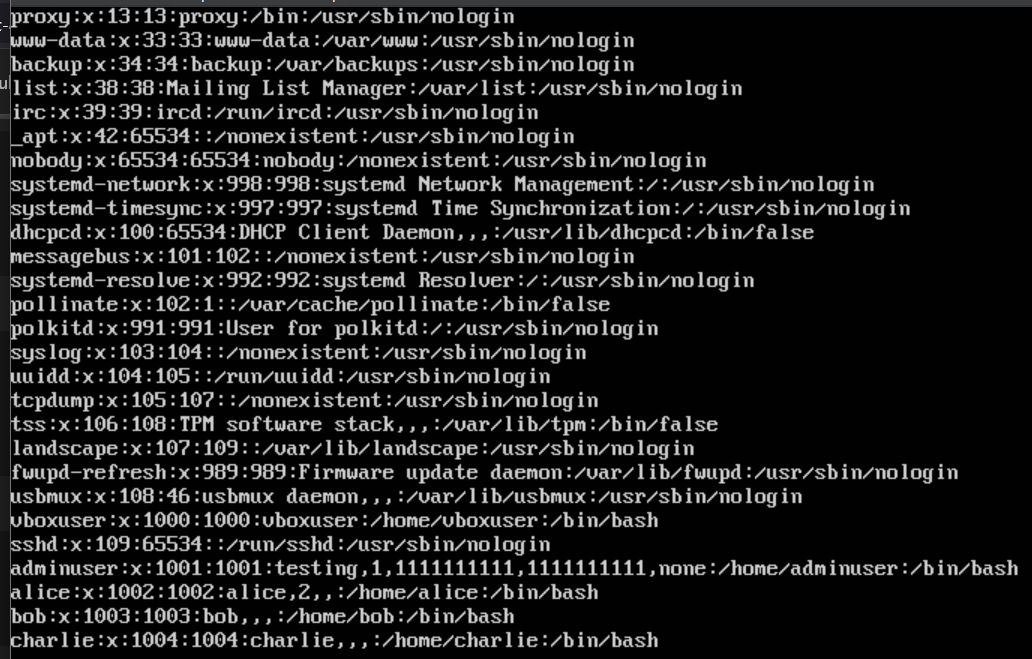
-k = give it a name (tag for filtering logs), ctrl x to exit then y to save it then press enter to toggle back to the Command line interface terminal. Reload the rules by entering sudo augenrules –load

If you are prompted an error message: writing etc/audit/rules.d/audit.rules: no such file or directory exist. Then it means the directory path /etc/audit/rules.d/ does not exist yet, the system cannot save the rules file. We need to enter this command to create the missing directory: sudo mkdir -p /etc/audit/rules.d this command will create the rules.d folder under /etc/audit . after that, redo the commands as previously mentioned to create the rules file. When saving try Ctrl + o to write, enter to confirm and ctrl + x to exit. Ignore any message saying rule already exist then enter sudo auditctl -l which should show the print out of each rules listed. Such as the following rules shown here:

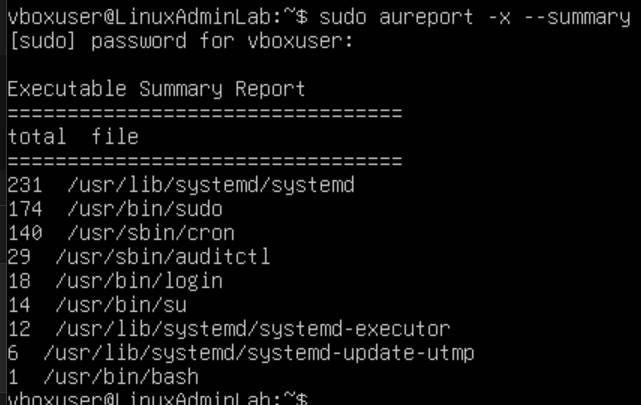
-w /etc/passwd -p wa -k passwd\_changes, -w /etc/shadow -p wa -k shadow\_changes,

-w /opt/devshare -p rwa -k devshare\_monitor, -w /usr/bin/sudo -p x -k sudo\_activity,

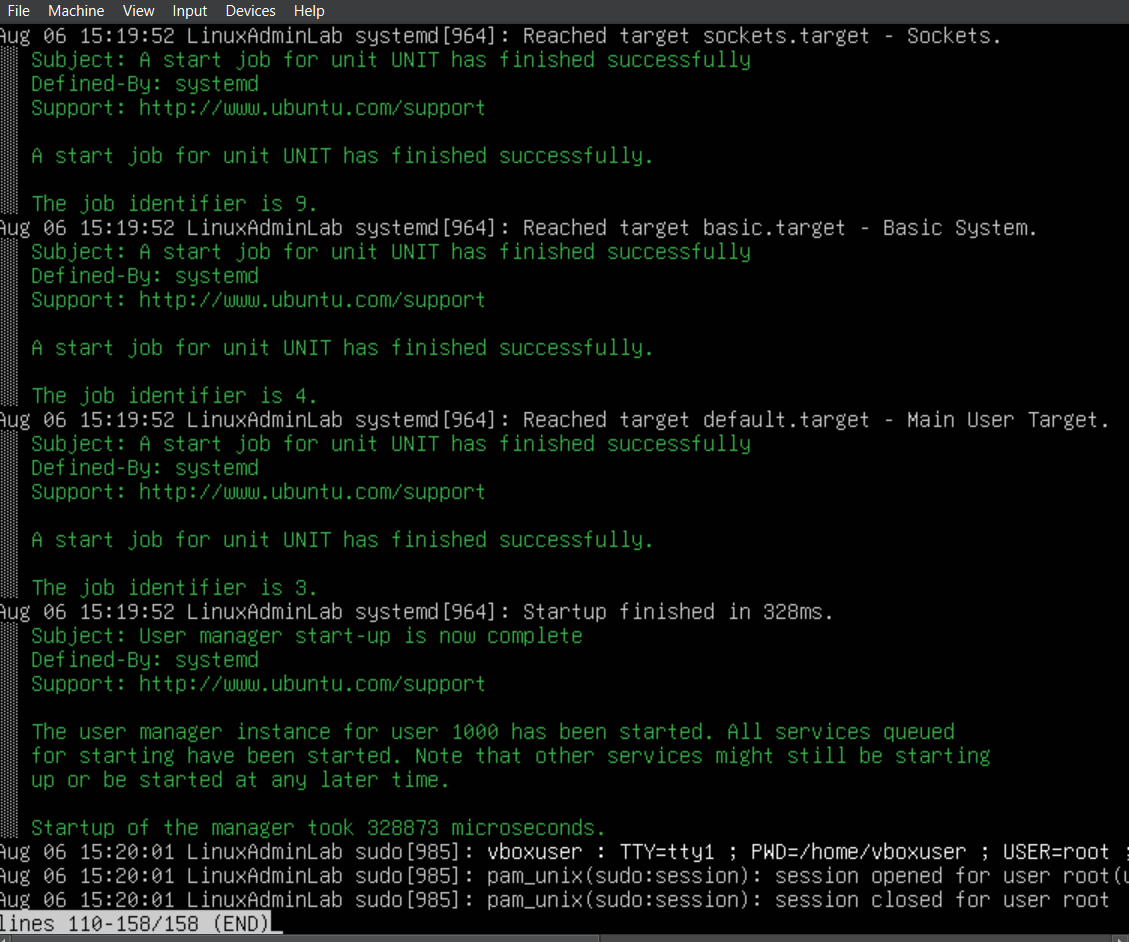
-w /opt/devshare -p rwxa -k devshare\_watch, now that this is done we will begin testing audit logs. Start by entering the following commands: sudo nano /etc/passwd or cat /opt/devshare/testfile.



Then enter sudo aureport -x –summary to generate a summary report and this is the result we should get.



Next enter Journalctl -xe to configure journalctl for centralized logging or external log forwarding

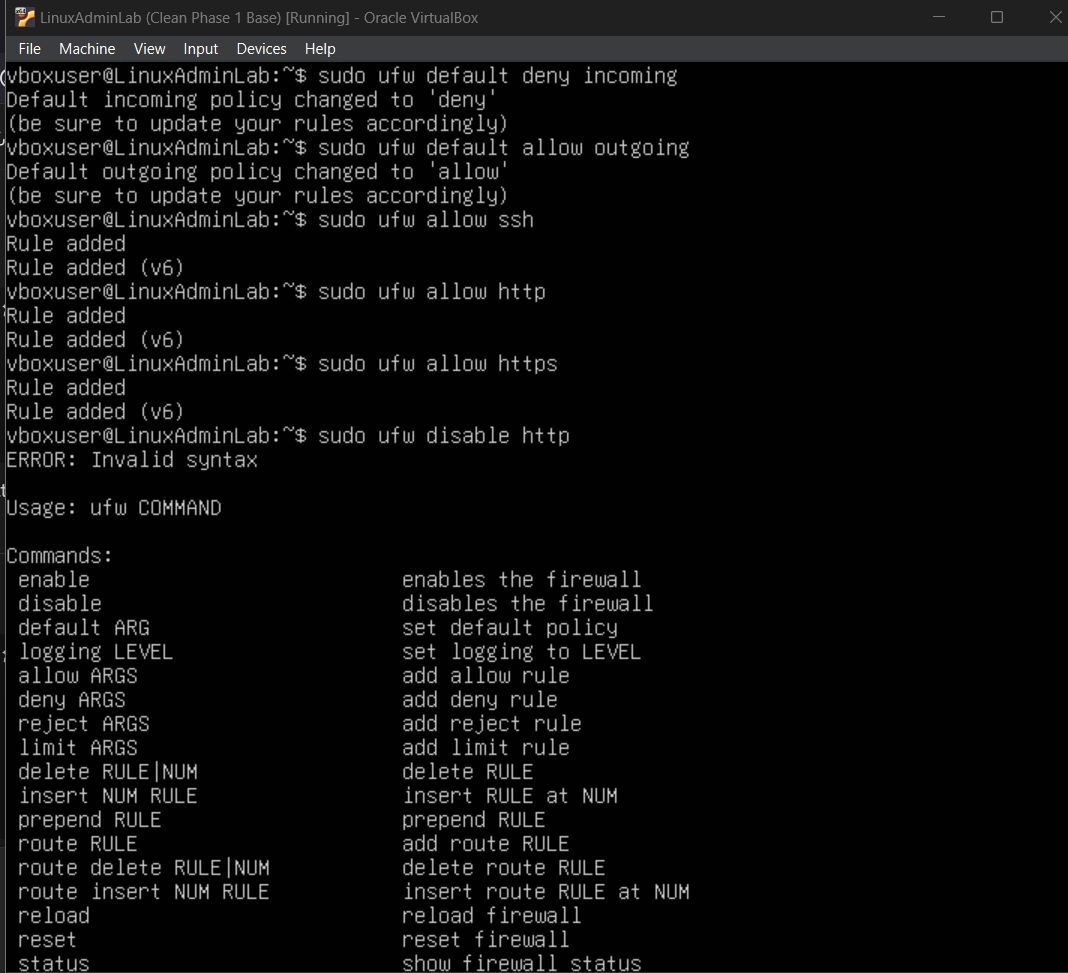


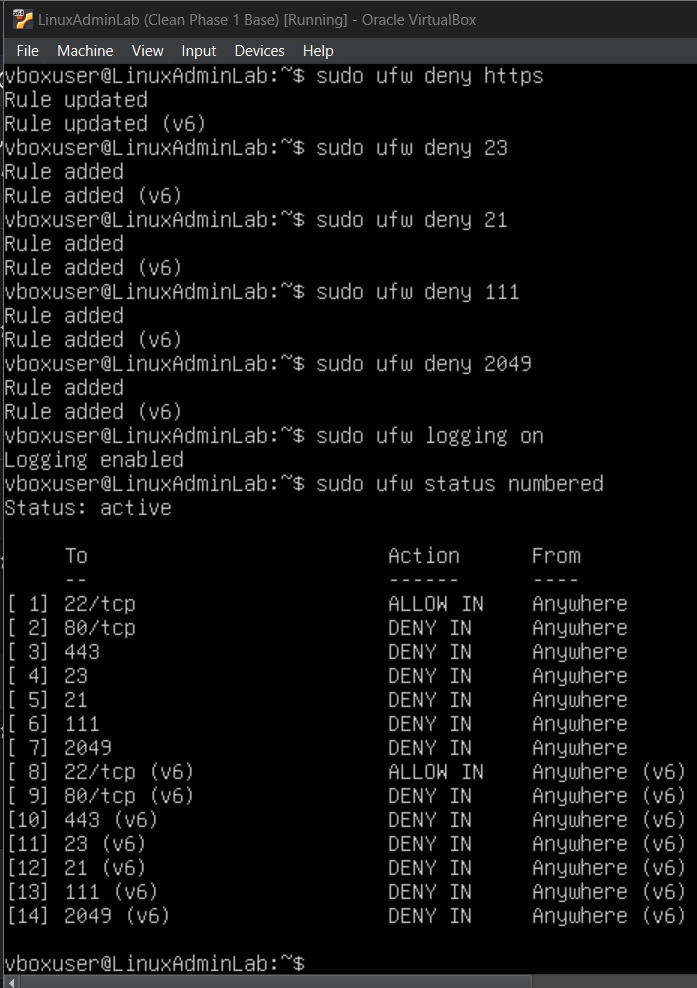
Auditd should actively log to /var/log/audit/audit.log

Search audit logs by keywords using ausearch and key files and directories are being monitored

Phase 6: **Firewall & Network Hardening**

Secure the Ubuntu system by configuring firewall rules, blocking unneeded ports, enabling only necessary traffic and applying basic network hardening settings to lower risk of attack surfaces. We will do the following enable and configure Uncomplicated firewall, Kernel level network hardening sysctl and disable unused network services. Start by checking the Uncomplicated firewall status by entering this command: sudo ufw status verbose, if it says inactive then enable it by entering sudo ufw enable. Next we will configure the default policies to deny all incoming traffic and allow outgoing traffic by entering the following commands: sudo ufw default deny incoming, sudo ufw default allow outgoing. Next we will configure the settings to only allow essential services such as ssh port 22, http port 80 and https port 443. We will be entering the following commands: sudo ufw allow ssh, sudo ufw allow http and sudo ufw allow https. If there are no web servers needed for this project then skip http port 80 and https port 443. Next we will deny services that we don’t need such as port 23 telnet, port 21 ftp, port 111 RPCbind and port 2049 NFS by entering the following commands: sudo ufw deny 23, sudo ufw deny 21, sudo ufw deny 111, sudo ufw deny 2049. Next enable logging by entering sudo ufw logging on and test the fire wall by entering sudo ufw status numbered.





The next step is to perform kernel level network hardening via sysctl, to backup current settings enter this command: sudo cp /etc/sysctl.conf /etc/sysctl.conf.bak then to edit it enter sudo nano /etc/sysctl.conf then enter this at the bottom:

# Disable IP forwarding (unless needed)

net.ipv4.ip\_forward = 0

net.ipv6.conf.all.forwarding = 0

# Ignore ICMP broadcast (mitigates smurf attacks)

net.ipv4.icmp\_echo\_ignore\_broadcasts = 1

# Ignore bogus ICMP responses

net.ipv4.icmp\_ignore\_bogus\_error\_responses = 1

# Disable source packet routing

net.ipv4.conf.all.accept\_source\_route = 0

net.ipv6.conf.all.accept\_source\_route = 0

# Disable ICMP redirects

net.ipv4.conf.all.accept\_redirects = 0

net.ipv6.conf.all.accept\_redirects = 0

# Enable Reverse Path Filtering (anti-spoofing)

net.ipv4.conf.all.rp\_filter = 1

# Log suspicious packets

net.ipv4.conf.all.log\_martians = 1

# Disable TCP timestamps (reduce info disclosure)

net.ipv4.tcp\_timestamps = 0

# Enable SYN cookies (mitigate SYN flood)

net.ipv4.tcp\_syncookies = 1

Next enter sudo sysctl -p then it should list all of the rules you just implemented in the sysctl.conf

Phase 7: **Monitoring Service Manager Setup**

The purpose is to set up monitoring service that lets you be able to see the details on status, load, CPU, memory, etc. The steps being done will be to verify audit rules load, monitor audit logs in real time, add inotifywait for instant file changes alerts and create a custom alert script. To verify audit rules are loaded start by entering sudo auditctl -l then to check live audit events enter this: sudo ausearch -m PATH -ts recent . Next add inotifywait for instant file change alerts by entering install inotify-tools, then switch to user alice (password 11111) then enter inotifywait -m /opt/devshare this will print a message in real time when files are created, motified, deleted or accessed. To install auditd email alerting, start by entering sudo apt install mailutils -y then select internet site then press enter twice. Next enter sudo nano /etc/audit/auditd.conf then at the bottom enter action\_mail\_acct [=your\_email@example.com](mailto:=your_email@example.com) then restart service by entering sudo systemctl restart auditd. In terms of using a monitoring tool, we have have monit watch directories and send alerts via email. Start by entering sudo apt install monit, sudo systemctl enable monit then enter sudo nano /etc/monit/monitrc to open the config file. Then at the bottom enter the following:

set mailserver smtp.gmail.com port 587

username "your\_email@gmail.com"

password "your\_app\_password"

using tlsv1

set alert [your\_email@gmail.com](mailto:your_email@gmail.com)

check directory devshare with path /opt/devshare

if changed timestamp then alert # This will trigger an alert if anything in /opt/devshare changes

exit and save it then enter these commands: sudo monit reload, sudo moit status

if you receive this message “the monit http interface is not enabled, please add the 'set httpd' statement and use the 'allow' option to allow monit to connect” then go back to editing the monit config file by sudo nano/etc/monit/monitrc then enter the following at the bottom:

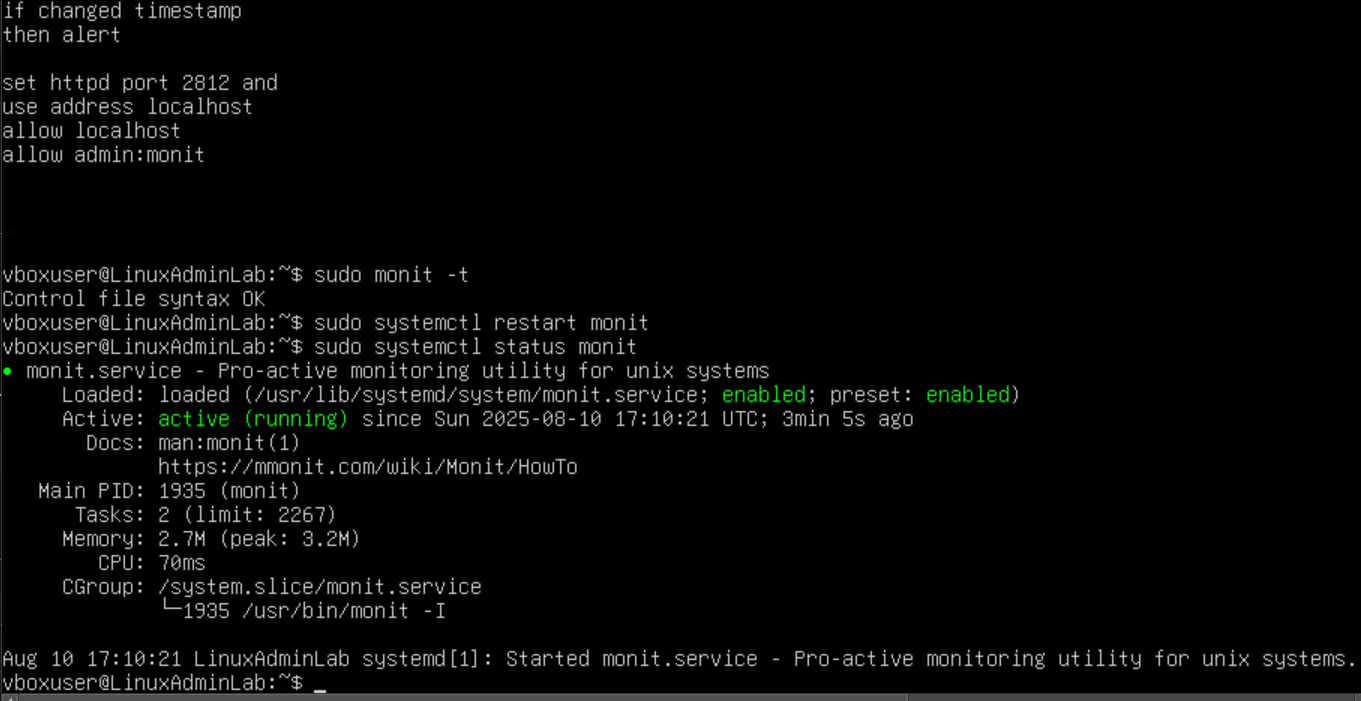
set httpd port 2812 and

use address localhost

allow localhost

allow admin:monit

Enter sudo monit -t then you should see Control file syntax OK which means there are no syntax errors. Next enter sudo systemctl restart moni then enter sudo systemctl status monit to make sure the status says active which should give us something like this as a result based on everything we did so far:



If you entered <http://localhost:2812> on a browser and it says unable to connect then we will need to run this command to see what ports monit is bound to: sudo ss -tlnp | grep monit

Output: listen 0 1024 127.0.0.1:2812 0.0.0.0:\* users:(("monit",pid=1935,fd=6))

Listen 0 1024 [::1]2812 [::]:\* users:(("monit",pid=1935,fd=7))

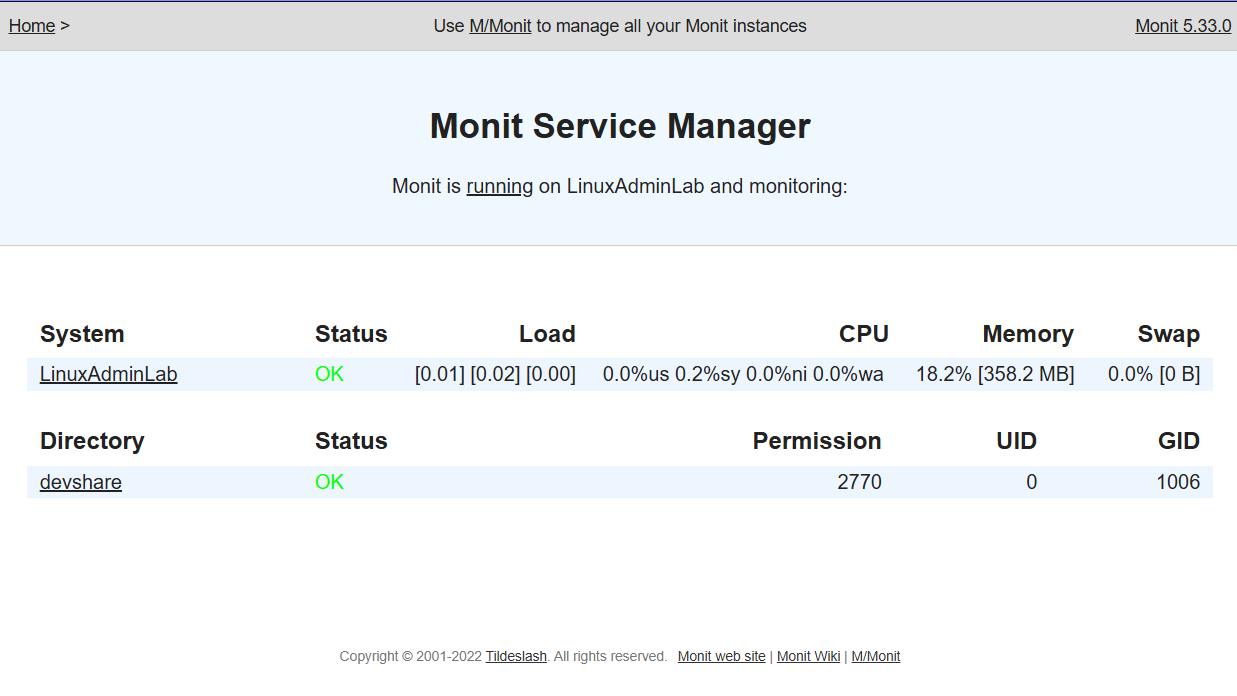
If you see 127.0.0.1:2812, Monit listens **only locally**. You must access it on the **same machine** via <http://localhost:2812>. You cant access it remotely unless you change monit’s config to listen to all interfaces or a specific IP. To enable remote access we will need to do this enter sudo nano /etc/monit/monitrc then add the following script at the end:

set httpd port 2812 and use address 0.0.0.0 allow admin:monit then exit and save. Enter sudo monit reload, check the ip of the host machine by entering hostname-I then if you enter curl -v <http://10.0.2.15:2812> you will be prompted with this result:



This error message recv failure: connection reset by peer indicates that monit web interface accepted the TCP connection but then immediately closed it before sending any response. The caused of this could be due to Monit’s HTTP interface isn’t configured properly in /etc/monit/monitrc, firewall/network blocking between machine and server or access control in monit is rejecting because of either not being on the allowed list or did not authenticate properly. But if we take a look at the config file of monit then we can see that monitrc has both set httpd port 2812 and use address localhost allow localhost

allow admin:monit and set httpd port 2812 and use address 0.0.0.0 allow admin:monit. This results in two conflicting HTTP interface definitions which prevents monit from binding properly, there should only be one set httpd block. For our case we will remove those http interface definition and just stick to using accessible only by locally which requires ssh tunneling to be able to reach it remotely. We will just be ensuring that this http interface definition is the one we will be using: set httpd port 2812, use address 0.0.0.0 allow admin:monit Given that the server runs inside an oracle virtual box on a local machine, the cause of the connection timeout from the PC browser is because the VM’s network settings does not allow the host to reach the VM’s IP or ports. Before we can enter http://<vm-ip>:2812 in our browser we will need to check and configure the VM’s Network adapter. Head to the VirtualBox Manager, go to the network’s settings in the selected VM’s settings then change the setting from NAT to bridged adapter since the NAT settings causes the VM’s network to be behind the virtual router which prevents the host from being directly able to reach its IP/ports. To get the vm ip just enter hostname -I as the command then enter that ip in the http://<vm-ip>:2812 which should finally prompt you to enter the login which is username: admin and password: monit then this is the page that you should finally be prompted to see:



Note: to make a file readable writeable by any user enter sudo chmod /filepath/file

If a directory is locked down even if the file is accessible then enter sudo chmod 777 /filepath/filedirectory

Phase 8: **System Auditing and Security Monitoring**

The purpose of this phase is to enhance system’s security posture by setting up auditing and monitoring tools that track user activities, changes and maybe security breaches. Configure auditd to monitor critical system files and user activities, enabling detection of unauthorized changes or suspicious behavior. Start by entering sudo apt update, sudo apt install auditd audispd-plugins -y then enter these commands: sudo systemctl start auditd, sudo systemctl enable auditd, sudo systemctl status auditd . Once that is done we will create rules to monitor critical files and activities. Audit rules will be implemented in /etc/audit/rules.d/ as the rules file. Start by entering sudo nano /etc/audit/rules.d/custom.rules then enter the following rules into the file then exit and save it.

# Monitor /etc/passwd for write and attribute changes

-w /etc/passwd -p wa -k passwd\_changes

# Monitor /etc/shadow for write and attribute changes

-w /etc/shadow -p wa -k shadow\_changes

# Monitor /usr/bin/sudo for execution attempts

-w /usr/bin/sudo -p x -k sudo\_activity

# Monitor /var/log/auth.log for write activity

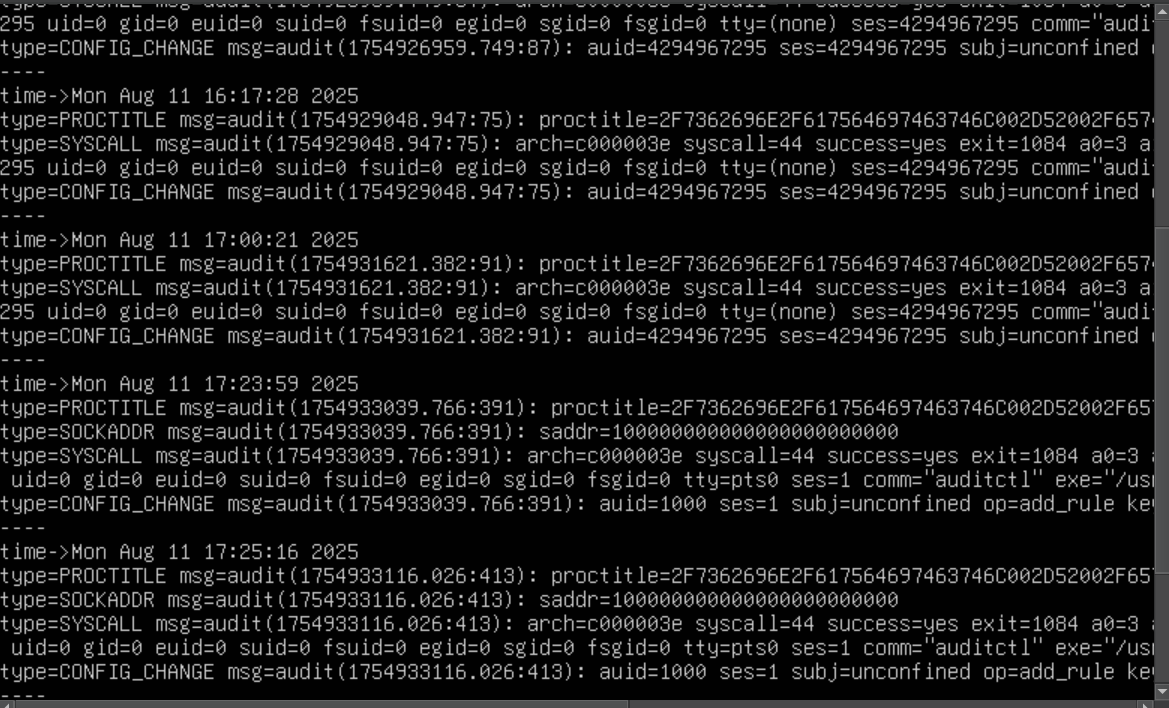
-w /var/log/auth.log -p wa -k authlog\_changes

# Monitor /opt/devshare directory for all access

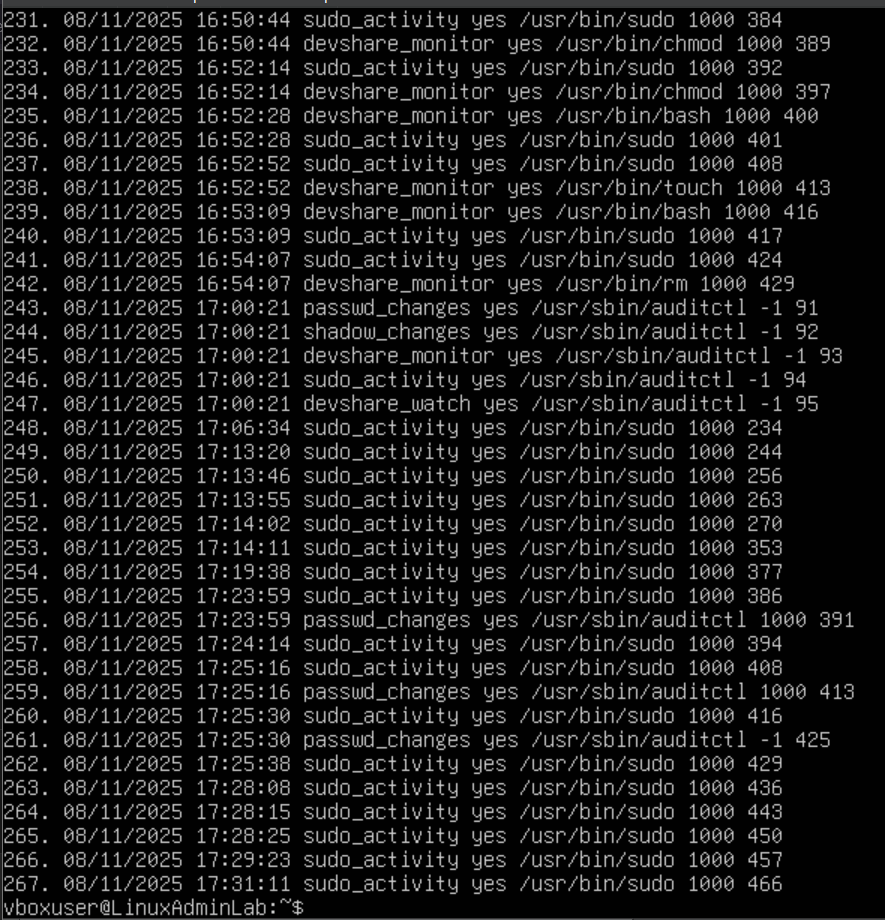
-w /opt/devshare -p rwa -k devshare\_monitor

Enter Sudo systemctl restart auditd then enter sudo auditctl -l to check the loaded rules. Once that is done we will search the audit logs, summary reports of events and reports of user logins.

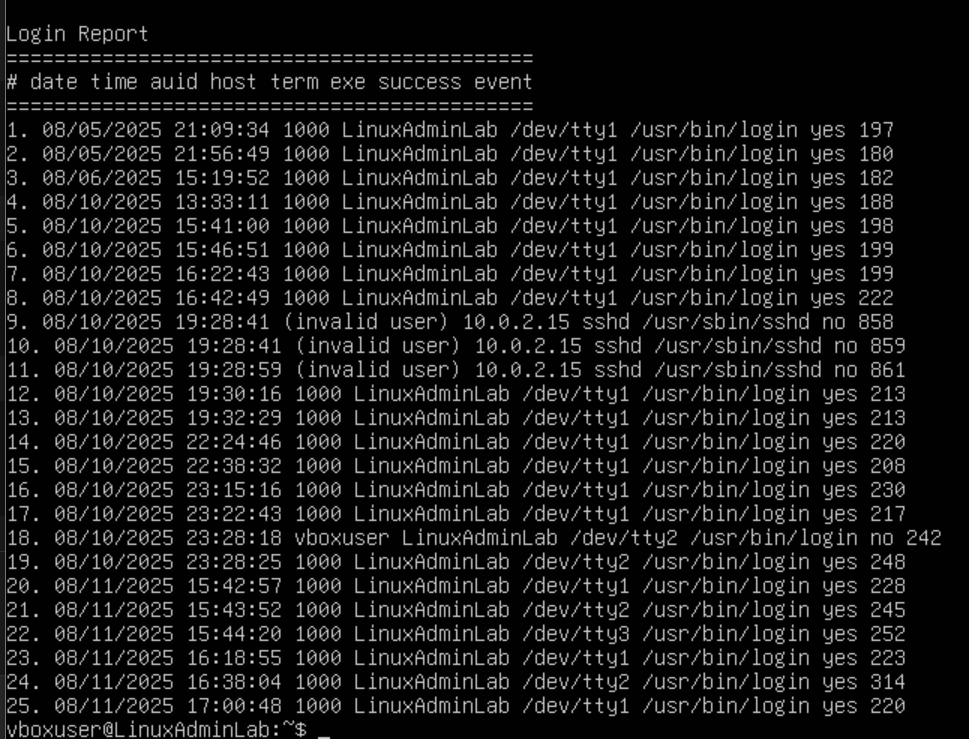
Sudo ausearch -k passwd\_changes



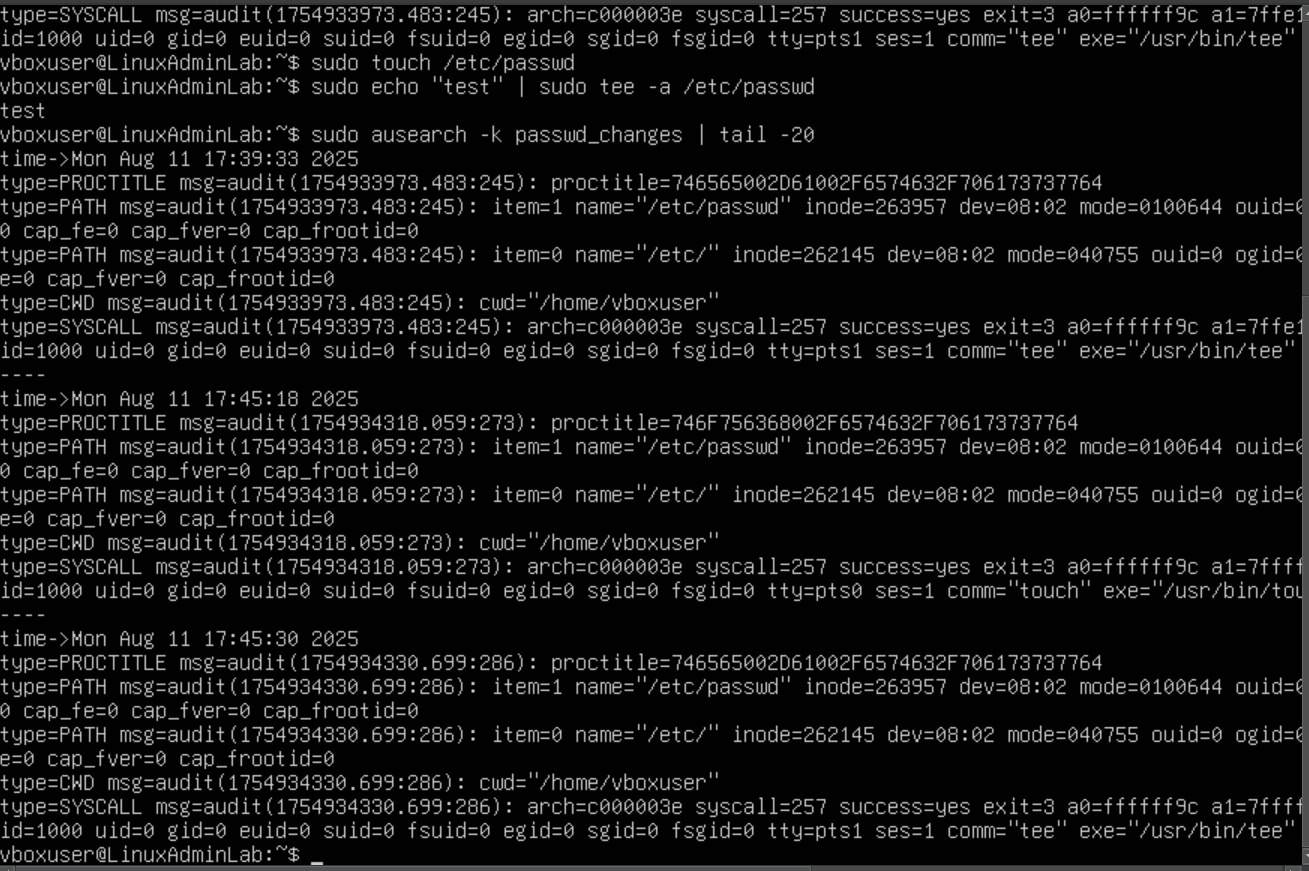
Next enter sudo aureport -k



After that enter sudo aureport -l



Next we will make changes to a watched file to cause an audit event: by entering these commands: sudo touch /etc/passwd, sudo echo “test” | sudo tee -a /etc/passwd, then check the log by entering sudo ausearch -k passwd\_changes | tail -20 then this is the result we should see



Next we can try configure the log rotation for audit logs since audit logs are stored in /var/log/audit/audit.log start by entering sudo nano /etc/audit/auditd.conf then enter max\_log\_file = 10 max\_log\_file\_action = ROTATE

Phase 9: **Patch Management & Software Updates**

The purpose is to secure and stabilize the Linux system by automating updates and patching vulnerabilities such as outdated packages. Since we are running Oracle Virtual Box with Ubuntu Linux, to update the current system enter: sudo apt update && sudo apt upgrade -y && sudo apt autoremove -y. Enable automated updates by entering sudo dpkg-reconfigure --priority=low unattended-updates and enter yes to enable automatic updates. To configure what gets updated enter sudo nano /etc/apt/apt.conf.d/50unattended-upgrades , make sure these lines are uncommented: "${distro\_id}:${distro\_codename}-security"; "${distro\_id}:${distro\_codename}-updates";

At the bottom enter this: Unattended-Upgrade::Remove-Unused-Dependencies "true";

Unattended-Upgrade::Automatic-Reboot "true"; Unattended-Upgrade::Automatic-Reboot-Time "02:00";

This will clean up old packages and reboot at 2AM if update requires it. Next set the update frequency by entering sudo nano /etc/apt/apt.conf.d/20auto-upgrades then set the file to the following script: APT::Periodic::Update-Package-Lists "1"; # Check daily

APT::Periodic::Download-Upgradeable-Packages "1"; # Download daily

APT::Periodic::Unattended-Upgrade "1"; # Install daily

To test the setup run sudo unattended-upgrade –dry-run –debug then enter sudo cat /var/log/unattended-upgrades/unattended-upgrades.log

Phase 10: **Full System Backup & Recovery**

This phase is about ensuring that you can restore it fast if the system gets corrupted, hacked or accidentally broken. Start off with a shutdown state of the VM, in the VirtualBox’s menu select take a snapshot, name it clean install – Aug 2025 and press OK. Click on the created snapshot and press restore, this is useful if you want to roll back due to testing new patches, risky configuration changes or malware/security related testing.

Phase 11: **Log Management & Centralized Monitoring**

The purpose of this phase is to let you collect, store and review logs for intrusion attempts, user actions and system events without the need for email configurations and no script configurations due to the amount of complicated unresolvable debugging that takes place if you were to attempt to configure both of them, instead we will primarily use Linux native tools. On Ubuntu, system logs are managed within system-journal then stored in /var/log/

Important log files:

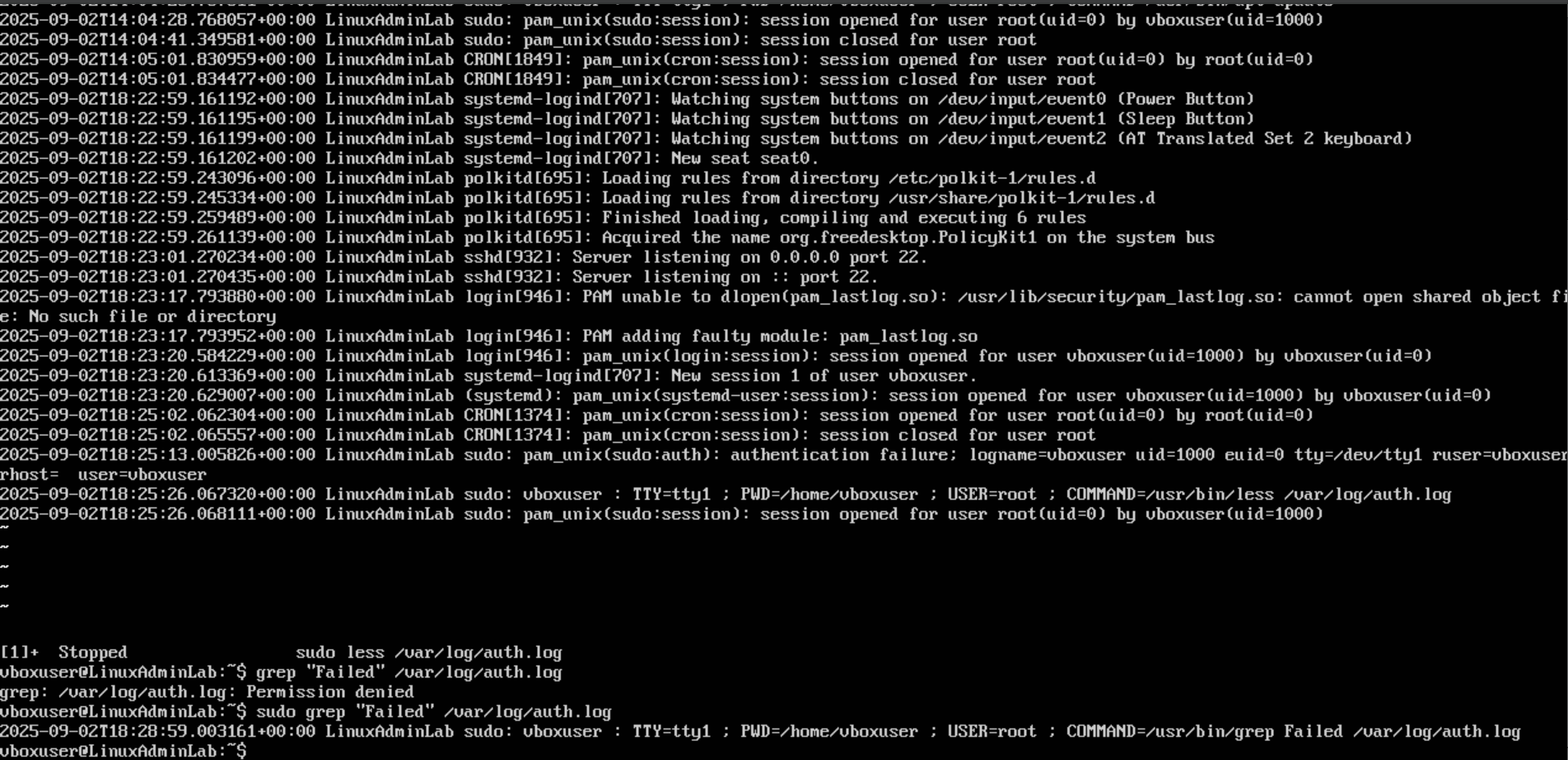
* /var/log/auth.log → logins, sudo, SSH activity
* /var/log/syslog → system events, services
* /var/log/kern.log → kernel messages
* /var/log/audit/audit.log (if auditd is enabled)

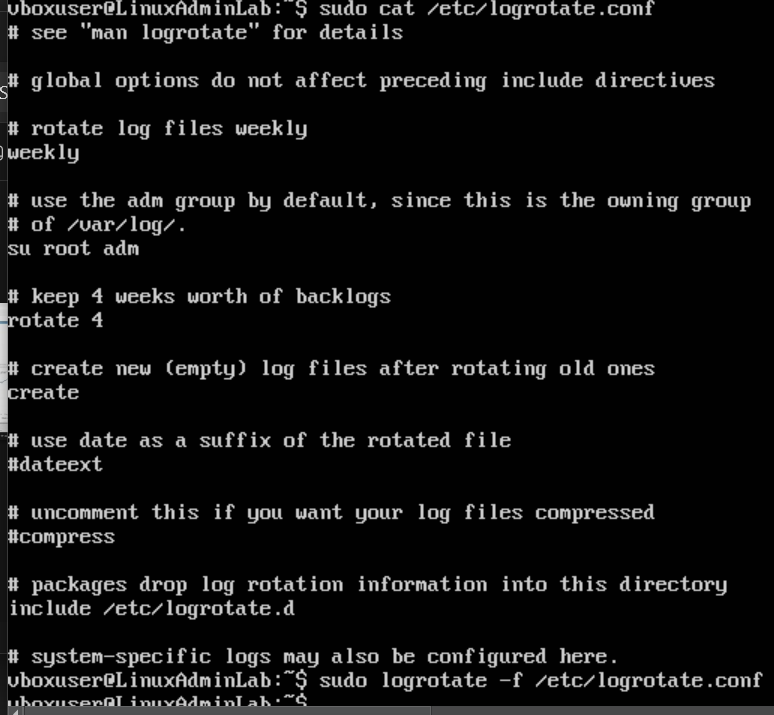
Here are the list of commands that Journalctl is the system log viewer:

Journalctl – See all logs

* Journalctl -b – logs since boot
* Journalctl -f – follow logs in real time (like tail -f)
* Journalctl -u ssh – see logs for SSH service
* Journalctl --since “1 hour ago” – see logs for last 1 hour

To Check failed logins and sudo attempts enter: sudo less /var/log/auth.log then press Ctrl z then enter sudo grep “Failed” /var/log/auth.log Then these are the results that should be seen:



When logs get too large, its best to use ubuntu’s logrotate, to check the config just enter sudo cat /etc/logrotate.conf generally logs rotate weekly and compress old ones. You can trigger the logrotate manually by enter sudo logrotate -f /etc/logrotate.conf 

The conclusions of this phase is knowing where logs are stored such as /var/log and Journalctl, allows you to search, filter and watch system events then log rotation helps keep your VM from running out of space so that way outdated log information gets cleared if possible.

Phase 13: **Final Hardening & Compliance Audit**

This phase is filling in any last gaps and ensuring the system is secure, stable and compliant. If you are on Ubuntu Linux in Virtual box, we will avoid using any external scripts nor emails due to the complicated nature of debugging without a perfect and proper solution towards ensuring its possible to get the scripts and email connection to work.

To list all the users enter cut -d: -f1 /etc/passwd then to enforce password complexity enter sudo nano /etc/login.defs then set the following configurations: PASS\_MAX\_DAYS 120 PASS\_MIN\_DAYS 7

PASS\_WARN\_AGE 14

You can force a password change on any existing user such as this sudo chage -M 90 -m 7 -W 14 charlie

Remove or Disable Unnecessary Services by entering the follow command:

sudo systemctl list-unit-files --type=service --state=enabled then disable anything not needed by entering sudo systemctl disable “any service”

To check the fire wall rules enter this: sudo ufw status verbose then the default result should be something like this: Default: deny in (incoming), allow in (outgoing) then allow what you sudo ufw allow ssh

To perform kernel & sysctl Hardening enter sudo nano /etc/sysctl.conf

Add or ensure

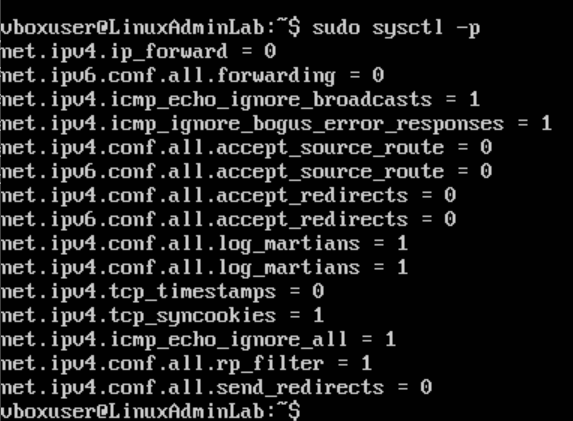
net.ipv4.icmp\_echo\_ignore\_all=1

net.ipv4.conf.all.rp\_filter=1

net.ipv4.conf.all.accept\_redirects=0

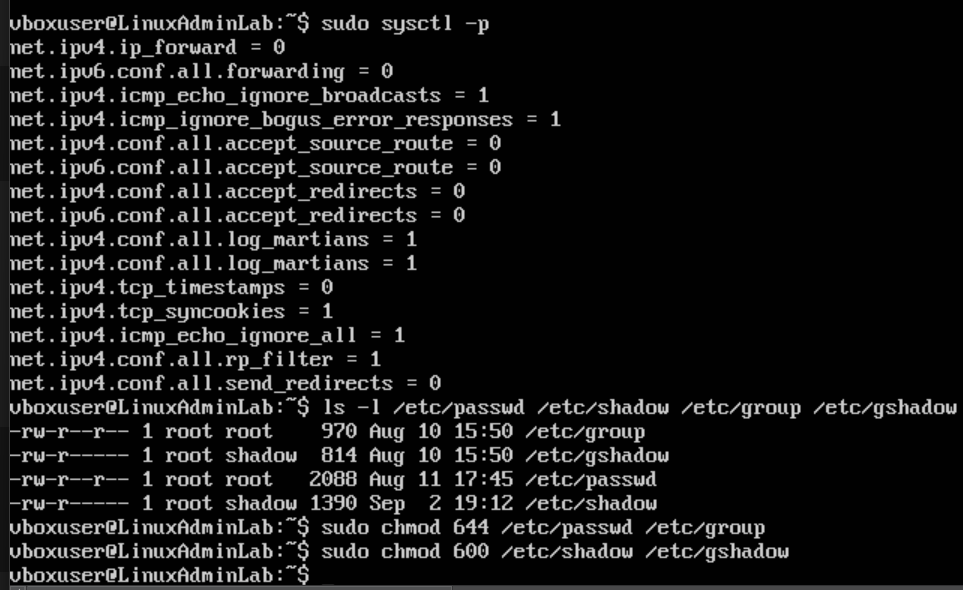
net.ipv4.conf.all.send\_redirects=0

apply change by entering sudo sysctl -p then the output should be like this:

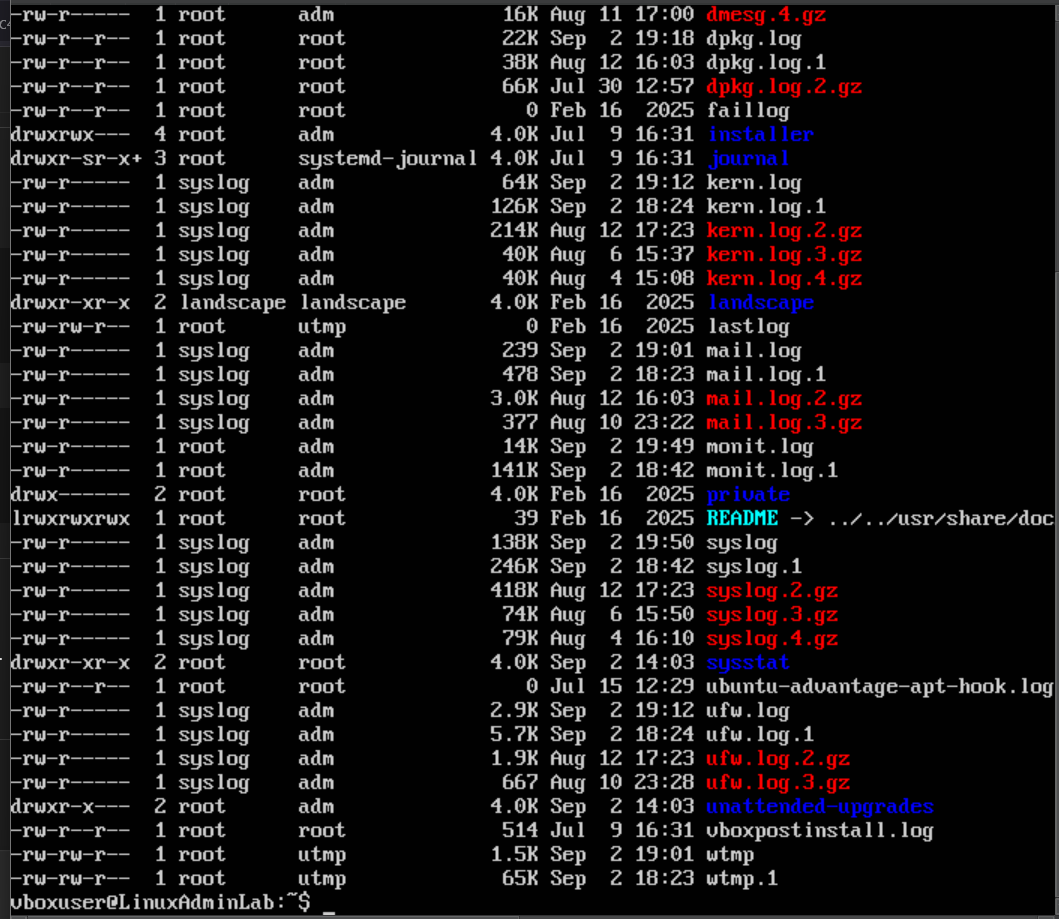


To check the permissions on sensitive files enter the following commands:

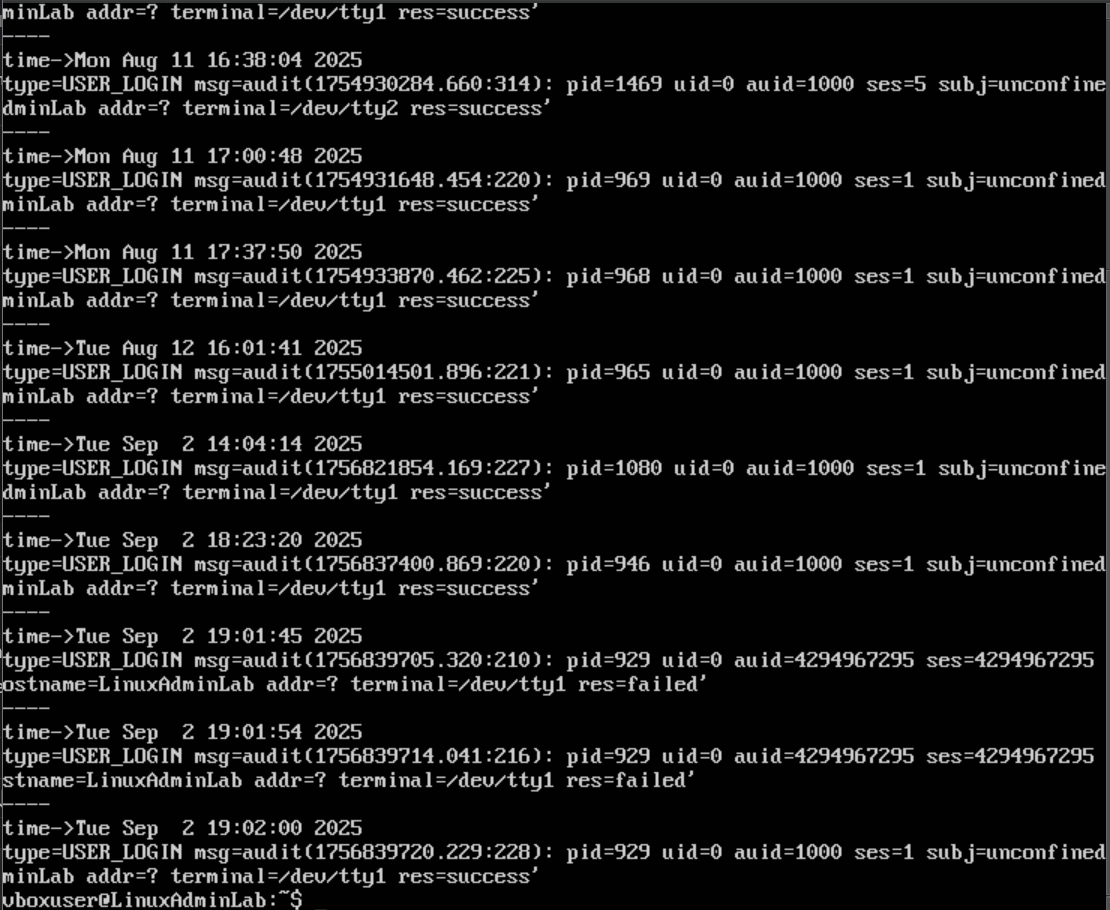
ls -l /etc/passwd /etc/shadow /etc/group /etc/gshadow



To verify that the logs are being written enter ls -lh /var/log/ then this should be prompted:



If auditd is installed then enter: sudo ausearch -m USER\_LOGIN



Close the VM then under the oracle Virtual box select the VM and press take snapshot since the system is now harden.

Summary:

**Linux System Hardening & Administration Project — Ubuntu (Virtualized Environment)**  
Completed a 32 page document on the successful deployment and securing an Ubuntu Linux system inside Oracle VirtualBox, demonstrating full-cycle administration and security hardening. Performed system preparation and user management by creating and organizing users, groups, and shared directories with correct permissions and fine-grained ACL controls. Configured audit logging to track critical file and authentication changes, automated recurring permission checks and audit scheduling, and applied firewall and network hardening with ufw and kernel sysctl rules. Implemented service monitoring with systemd, system auditing with auditd, and established patch management using automated security updates. Completed full system backup and recovery procedures with the Oracle VirtualBox’s Snapshot feature, centralized log monitoring via journalctl and logrotate, and validated final compliance through password policies, service reviews, kernel security configurations, and sensitive file permission checks. This project reinforced hands-on experience with Linux security principles, system monitoring, and operational resilience in a virtualized environment.