- **Task 22: Configure iptables/UFW**
  → Example: *sudo ufw allow 80* (open port 80), *sudo ufw deny 21* (block FTP).
- **Task 23: Block Ports & Verify**
  → Block ports and verify using Nmap that they are closed.
- **Task 24: Enable SSL/TLS**
  → Generate a self-signed certificate and enable HTTPS for Apache or Nginx.

Task 22: Configure iptables/UFW
→ Example: *sudo ufw allow 80* (open port 80), *sudo ufw deny 21* (block FTP).

Option 1: Using UFW (on CentOS)

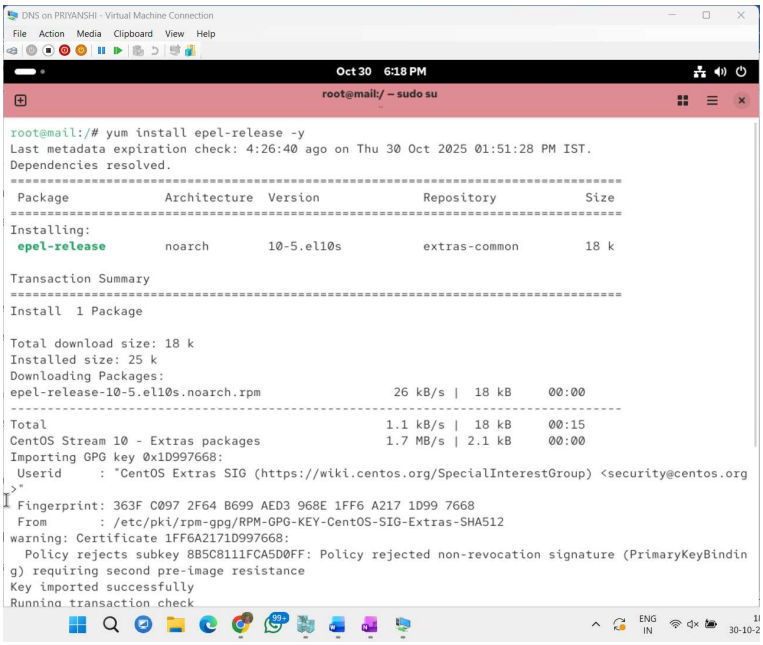
Step 1: Install and enable UFW

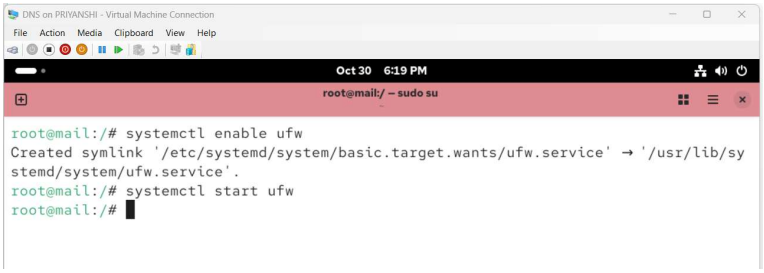
sudo yum install epel-release -y

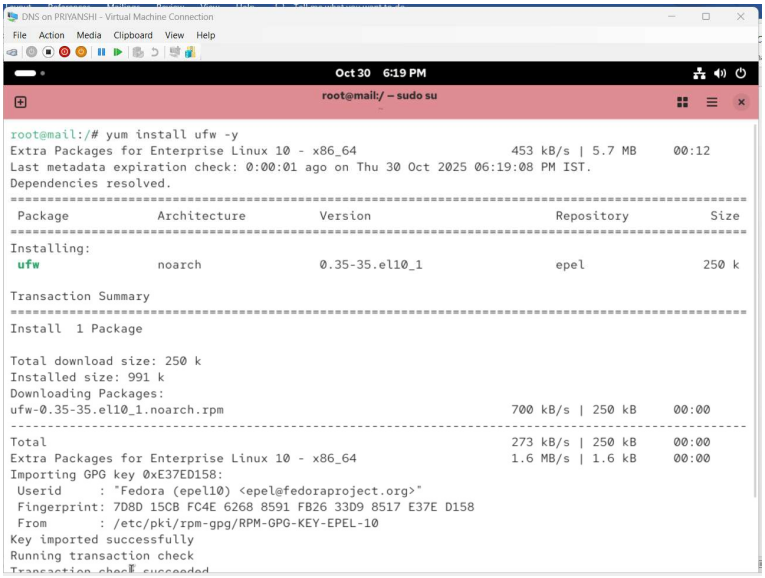
sudo yum install ufw -y

sudo systemctl enable ufw

sudo systemctl start ufw





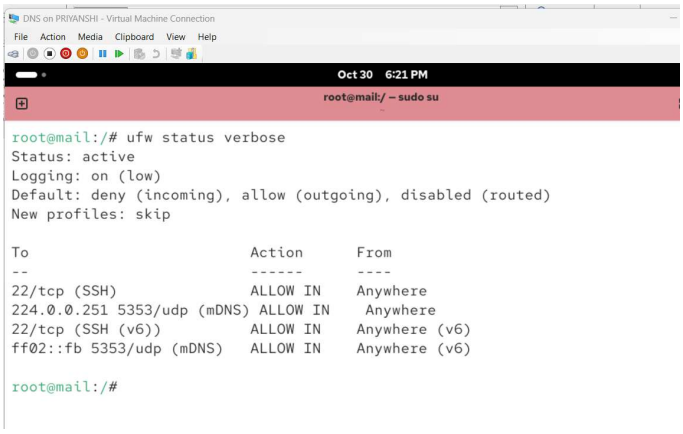


Step 2: Check status

sudo ufw status verbose

If it shows “inactive”, enable it:

sudo ufw enable



Step 3: Allow or deny ports

sudo ufw allow 80 # Allow HTTP (port 80)

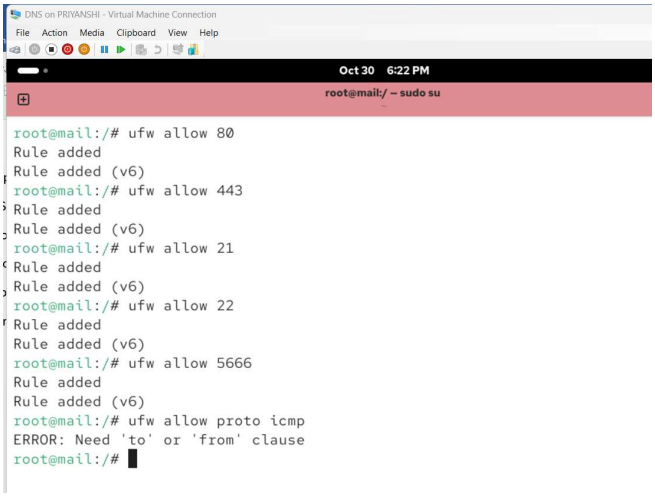
sudo ufw allow 443 # Allow HTTPS (port 443)

sudo ufw deny 21 # Block FTP (port 21)

sudo ufw allow 22 # Allow SSH (port 22)

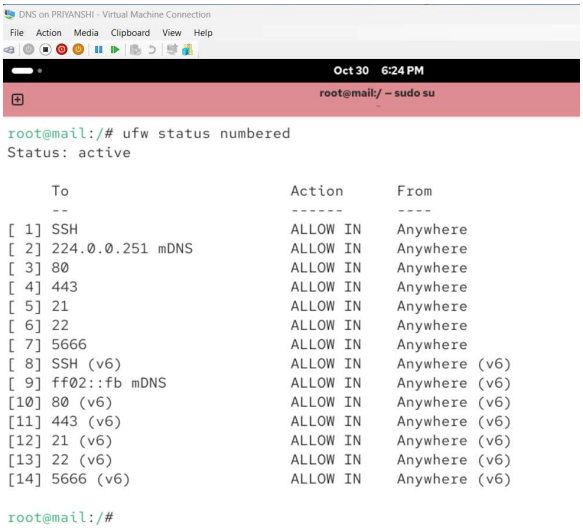
sudo ufw allow 5666 # Allow Nagios NRPE communication

sudo ufw allow proto icmp # Allow ping



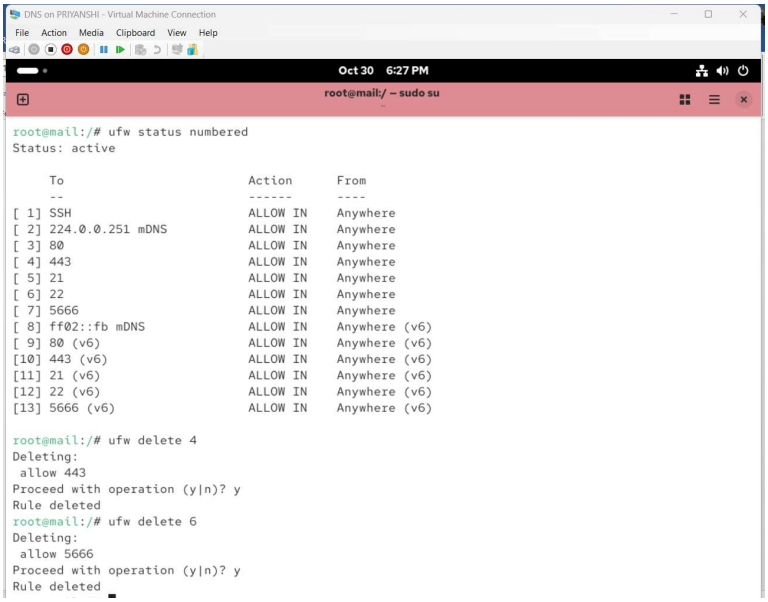
Step 4: Verify rules

sudo ufw status numbered



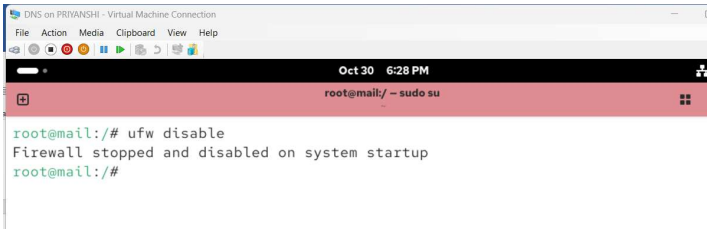
Step 5: Delete a rule if needed

sudo ufw delete <rule_number>



Step 6: Disable UFW if required

sudo ufw disable



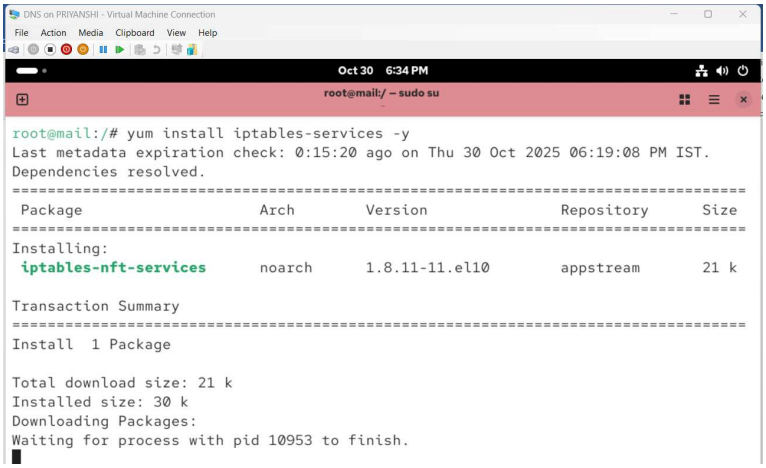
Option 2: Using iptables (CentOS)

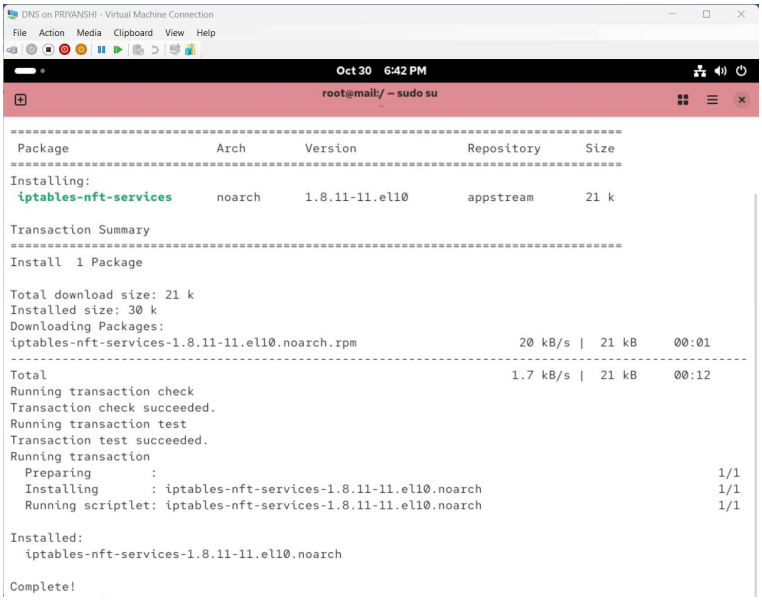
Step 1: Install and enable iptables

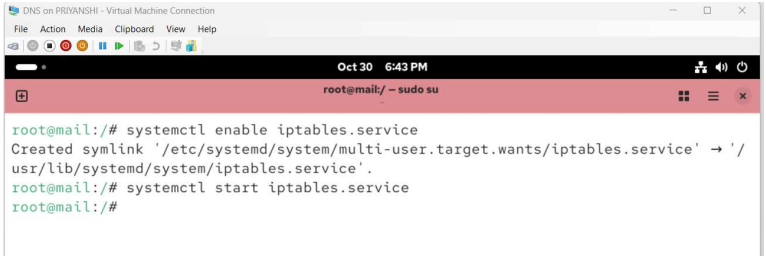
sudo yum install iptables-services -y

sudo systemctl enable iptables

sudo systemctl start iptables



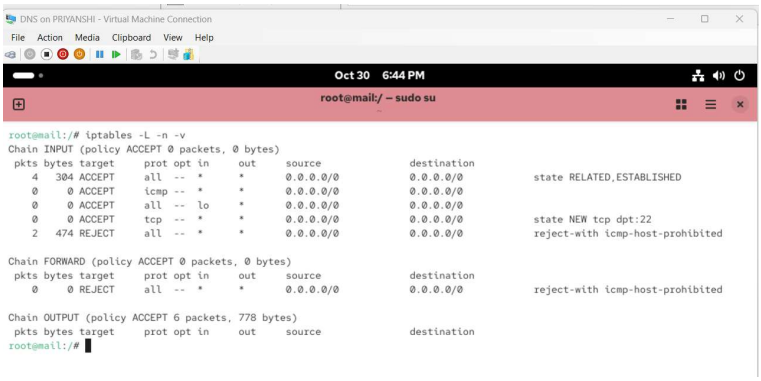




Step 2: Check current rules

sudo iptables -L -n -v

This shows all currently active firewall rules.



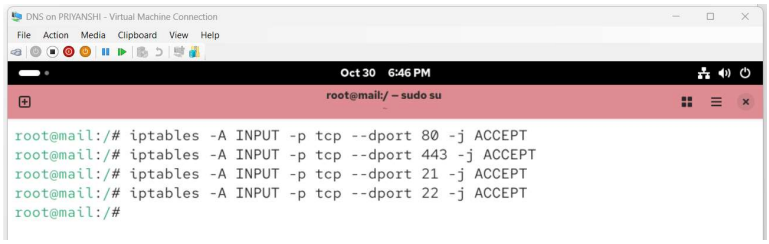
Step 3: Add firewall rules

sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT # Allow HTTP

sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT # Allow HTTPS

sudo iptables -A INPUT -p tcp --dport 21 -j DROP # Deny FTP

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT # Allow SSH

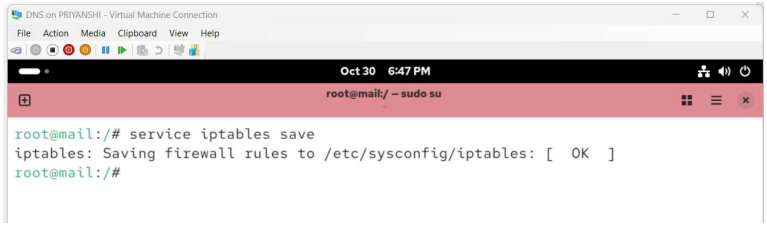


Step 4: Save iptables rules

sudo service iptables save

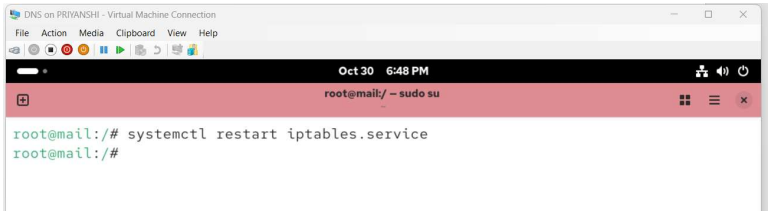
You should see:

Saving firewall rules to /etc/sysconfig/iptables: [OK]



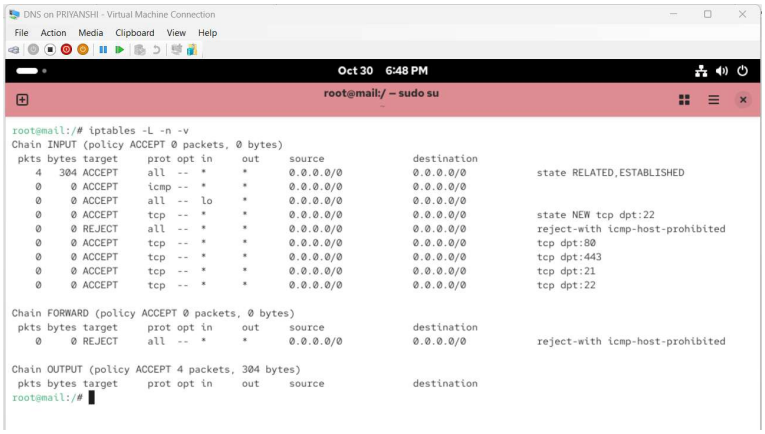
Step 5: Restart iptables service

sudo systemctl restart iptables



Step 6: Verify rules

sudo iptables -L -n -v

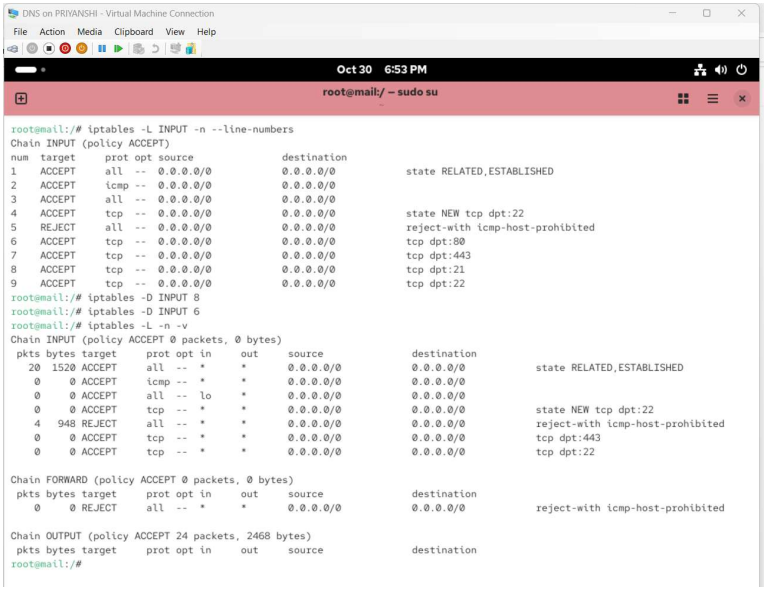


Step 7: Delete a rule (optional)

If you want to remove a specific rule:

sudo iptables -D INPUT 8

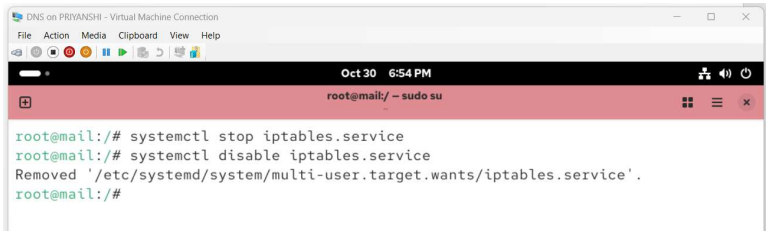
sudo iptables -D INPUT 6



Step 8: Disable iptables (optional)

sudo systemctl stop iptables

sudo systemctl disable iptables



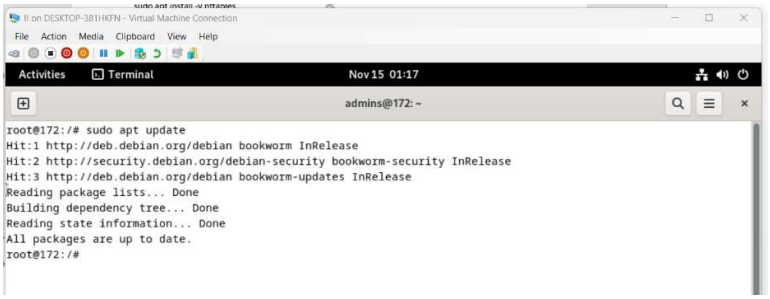
Task 23: Block Ports & Verify

Block ports and verify using Nmap that they are closed.

STEP 1 — Install nftables

Command:

sudo apt update
sudo apt install -y nftables



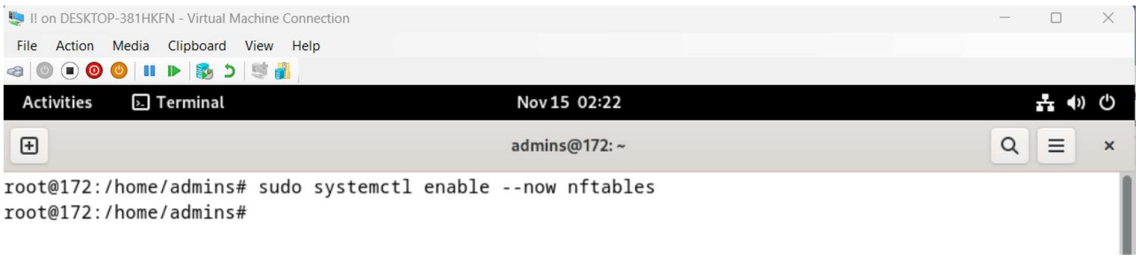
Explanation:

- apt update refreshes your Debian package list so it knows the latest available versions
- nftables is the modern Linux firewall system (it replaces older iptables).
- Installing it ensures the firewall commands (nft) work properly.

STEP 2 — Enable and start the nftables service

Command:

sudo systemctl enable --now nftables



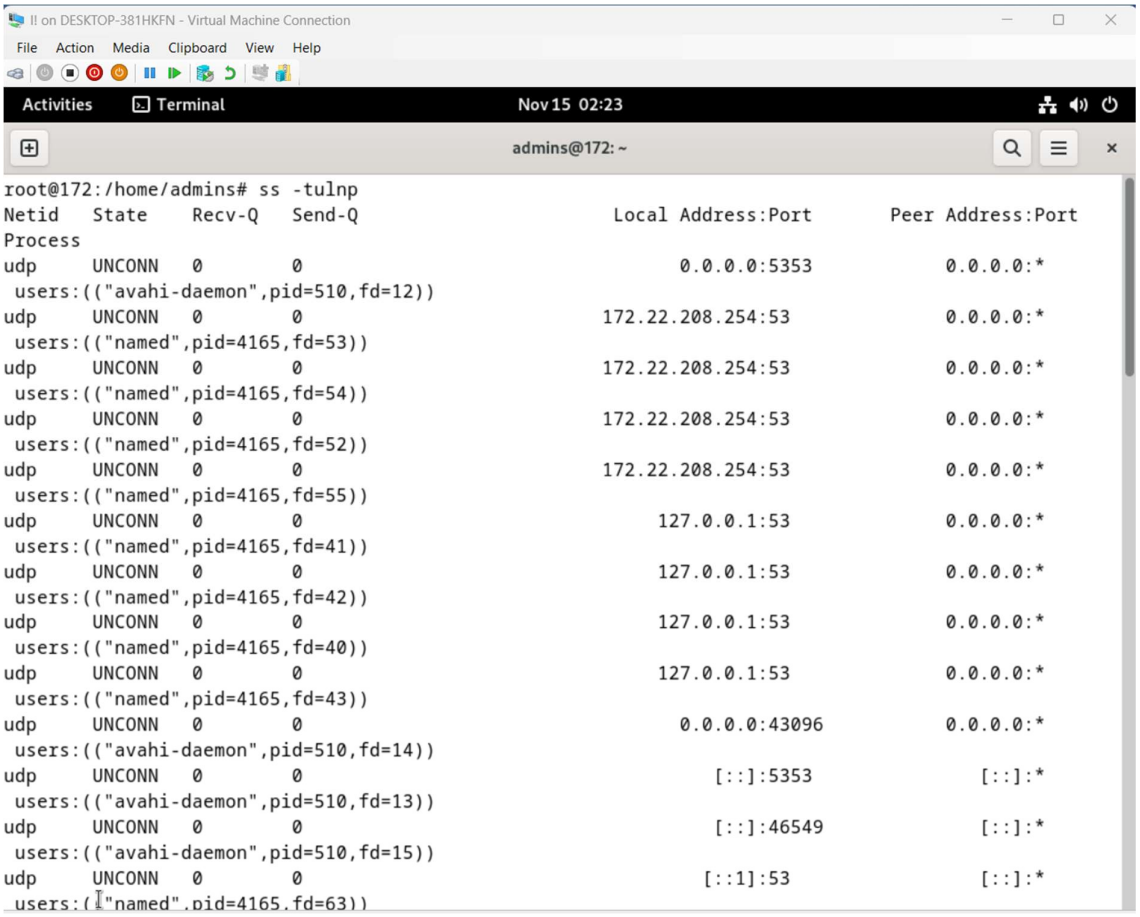
Explanation:

- systemctl enable → makes nftables start every time the system boots.
- --now → starts the nftables service immediately.
- Without enabling this, your firewall rules may not work after a reboot.
- This is required so that /etc/nftables.conf gets loaded automatically when the system starts.

STEP 3 — Check current open ports (optional but important)

Command:

ss -tulnp



Explanation:

- ss (socket statistics) shows what services are currently listening.
- -t → TCP
- -u → UDP
- -l → listening ports
- -n → show port numbers instead of service names
- -p → show the process name and PID

Example output:

LISTEN 0 128 0.0.0.0:22 *:* users:(("sshd",pid=700,fd=3))

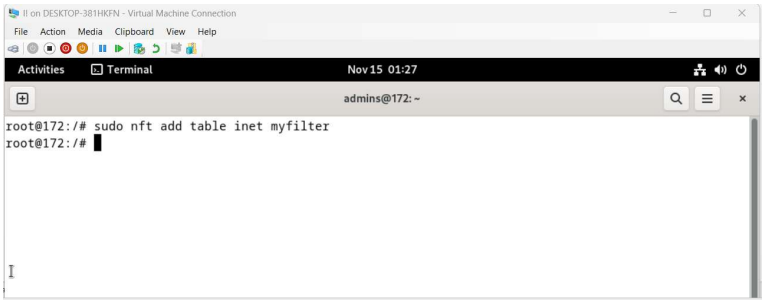
This shows SSH is listening on port 22.

Even if a service is listening, firewall can block it — firewall rules override listening services.

STEP 4 — Create a new nftables table

Command:

sudo nft add table inet myfilter



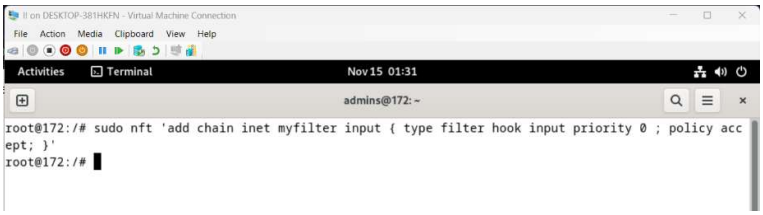
Explanation:

- You are creating a new firewall table named myfilter.
- inet means the table works for both IPv4 and IPv6.
- A table is like a folder where firewall chains and rules live.
- This keeps your firewall rules organized and clean.

STEP 5 — Create INPUT chain with policy

Command:

sudo nft 'add chain inet myfilter input { type filter hook input priority 0 ; policy accept; }'



Explanation:

- A chain is a list of rules that traffic passes through.
- input chain handles incoming traffic to your system (important).
- type filter → this chain is used for filtering packets.
- hook input → attaches this chain to the Linux kernel's input handling.
- priority 0 → normal priority for filtering.
- policy accept → default action is allow (we will explicitly block only required ports).

This structure ensures:

- System remains safe.
- You do not accidentally block everything.
- Rules work in the correct order.

STEP 6 — Add essential allow rules (SAFETY RULES)

These rules prevent system break or accidental lockout.

6.1 Allow ESTABLISHED and RELATED connections

Command:

sudo nft add rule inet myfilter input ct state established,related accept

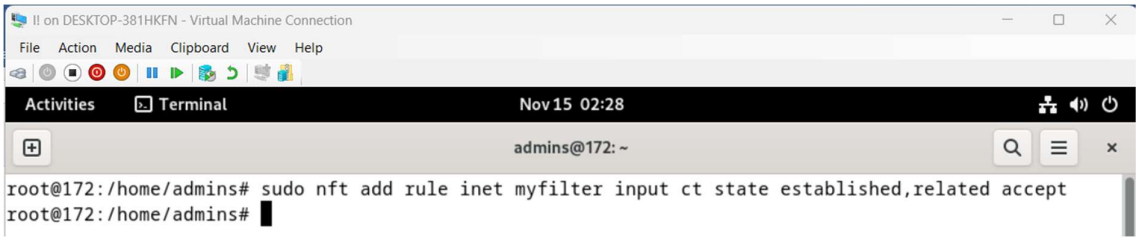
Explanation:

- This rule allows ongoing connections (SSH session, updates, downloads).
- ct state established → packets that belong to an existing connection.
- related → packets related to existing connections (like FTP data).
- This prevents your internet or SSH session from suddenly disconnecting.

6.2 Allow loopback interface

Command:

sudo nft add rule inet myfilter input iif lo accept



Explanation:

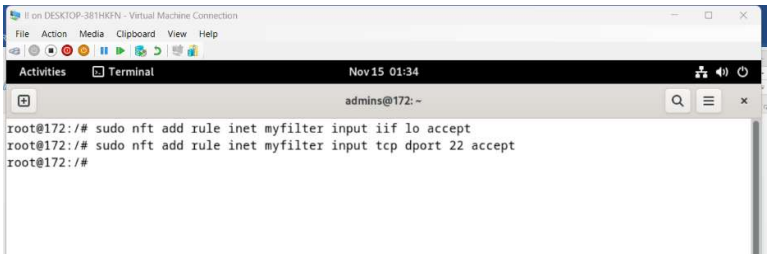
- The loopback interface (lo) is your system's internal network (127.0.0.1).
- Many applications (e.g., MySQL, Apache, systemd) talk to themselves using this interface.
- Blocking this would break internal system communication.

So we explicitly allow it to keep system stable.

6.3 Allow SSH (VERY IMPORTANT)

Command:

sudo nft add rule inet myfilter input tcp dport 22 accept



Explanation:

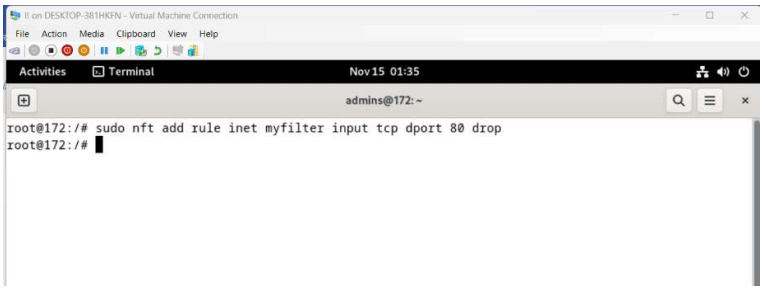
- If you are connected via SSH (22), blocking port 22 would lock you out permanently.
- This rule ensures SSH remains open even after you start blocking other ports.
- Always allow SSH first in any firewall configuration.

STEP 7 — BLOCK the ports you want

7.1 Block port 80 (HTTP)

Command:

sudo nft add rule inet myfilter input tcp dport 80 drop



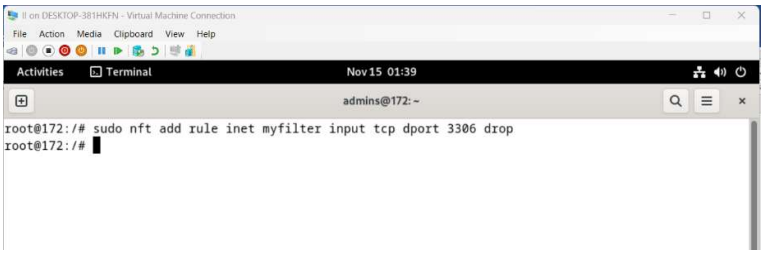
Explanation:

- tcp dport 80 → matches traffic coming to port 80.
- drop → silently discard packets (no reply to sender).
- When dropped, Nmap will show these ports as filtered.
- This is useful in secure environments—attackers don't know what's behind the firewall.

7.2 Block port 3306 (MySQL)

Command:

sudo nft add rule inet myfilter input tcp dport 3306 drop



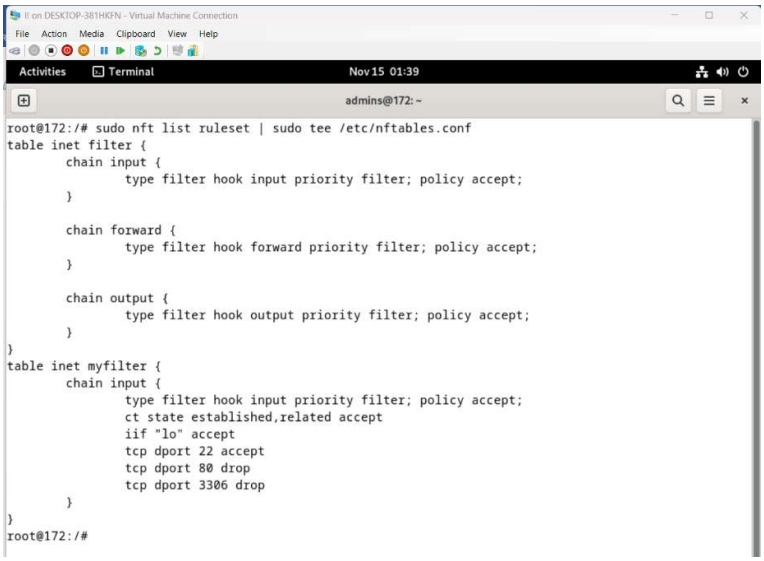
Explanation:

- Same as above, but targeting MySQL port.
- Recommended because exposing DB ports publicly is a major security risk.
- Dropping the port ensures remote systems cannot scan or access it.

STEP 8 — Save your rules permanently

Command:

sudo nft list ruleset | sudo tee /etc/nftables.conf



Explanation:

- nft list ruleset prints the entire active firewall configuration.
- tee /etc/nftables.conf writes it to the main config file.
- /etc/nftables.conf is loaded on every boot automatically.

STEP 9 — Verify your rules

Command:

This lets you confirm that:

- Table created
- Input chain exists
- SSH allowed

- Loopback allowed
- Established connections allowed
- Port 80 & 3306 blocked

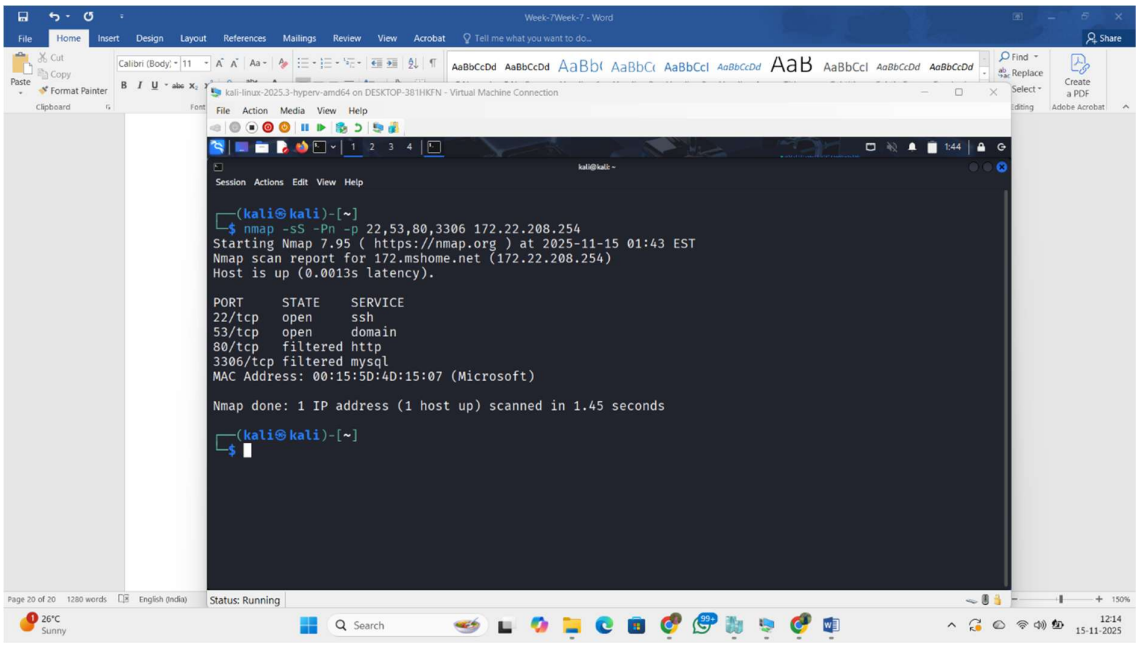
If everything looks correct → firewall is configured properly.

STEP 10 — Verify using Nmap

Run from another machine (NOT from same Debian system):

Command:

nmap -sS -Pn -p 22,80,3306 <your_server_ip>



Explanation:

- -sS → SYN scan (stealth scan — fastest & standard)
- -Pn → skip ping (firewall may block ping)
- -p → specify ports to check

Expected Output:

22/tcp open
80/tcp filtered
3306/tcp filtered

Meaning:

- open → allowed
- filtered → blocked by firewall

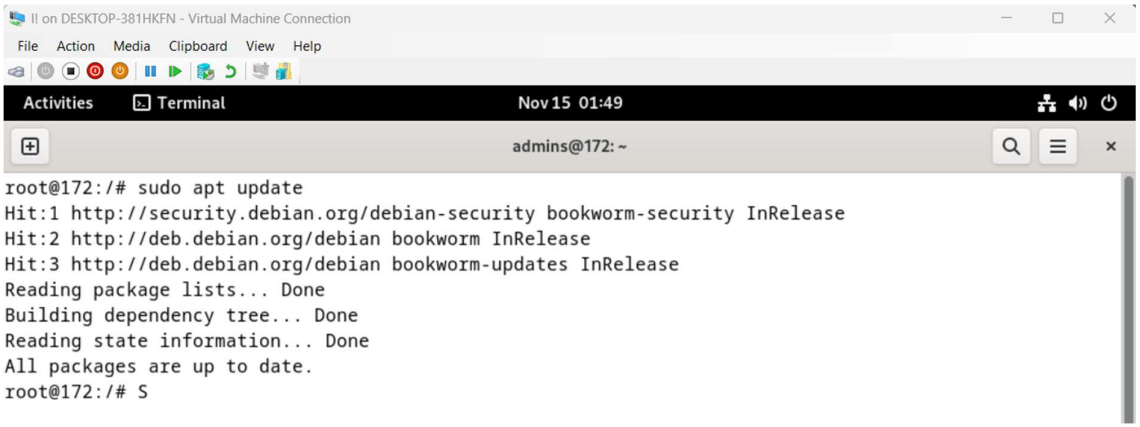
Task 24: Enable SSL/TLS

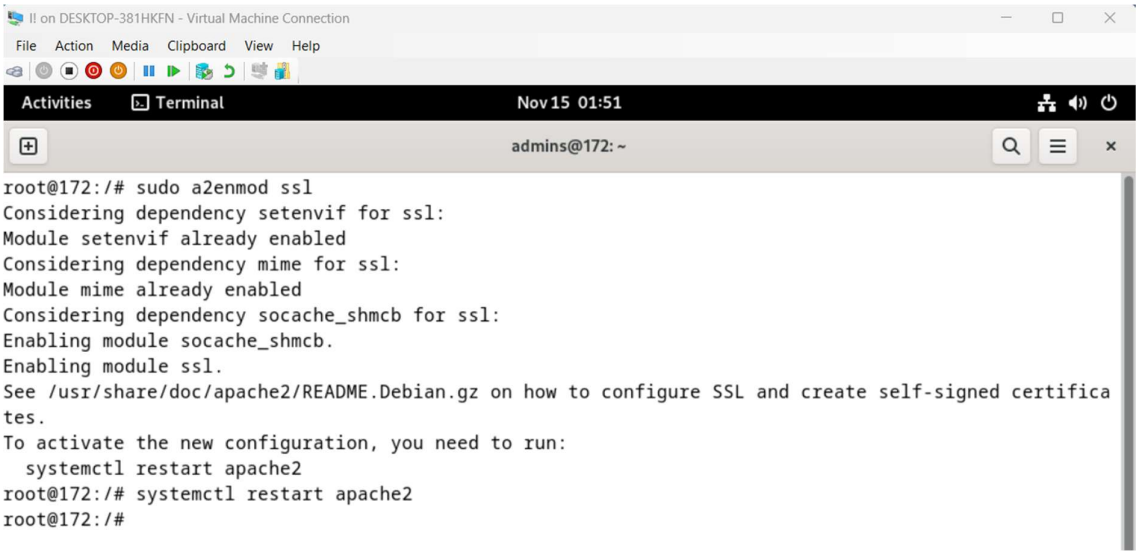
→ Generate a self-signed certificate and enable HTTPS for Apache or Nginx.

You will:
Install SSL module
Create a self-signed certificate
Create secure virtual host
Enable HTTPS
Test HTTPS

Step 1: Install SSL module in Apache

sudo apt update
sudo apt install openssl
sudo a2enmod ssl



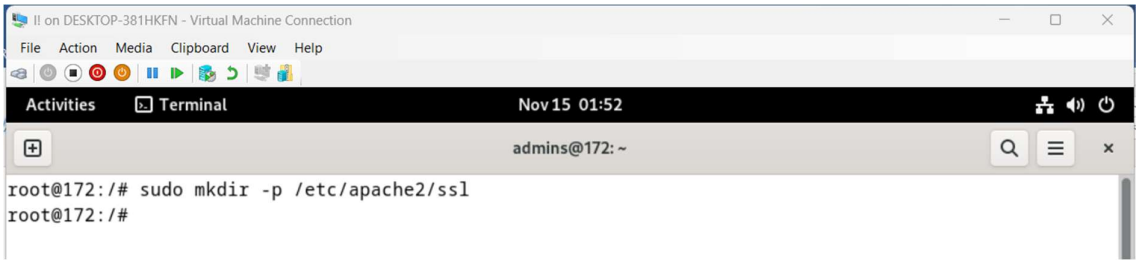


Explanation:

- openssl → Tool used to generate certificates
- a2enmod ssl → Enables SSL module in Apache so Apache can serve HTTPS traffic.

Step 2: Create a Directory for SSL Certificates

sudo mkdir -p /etc/apache2/ssl

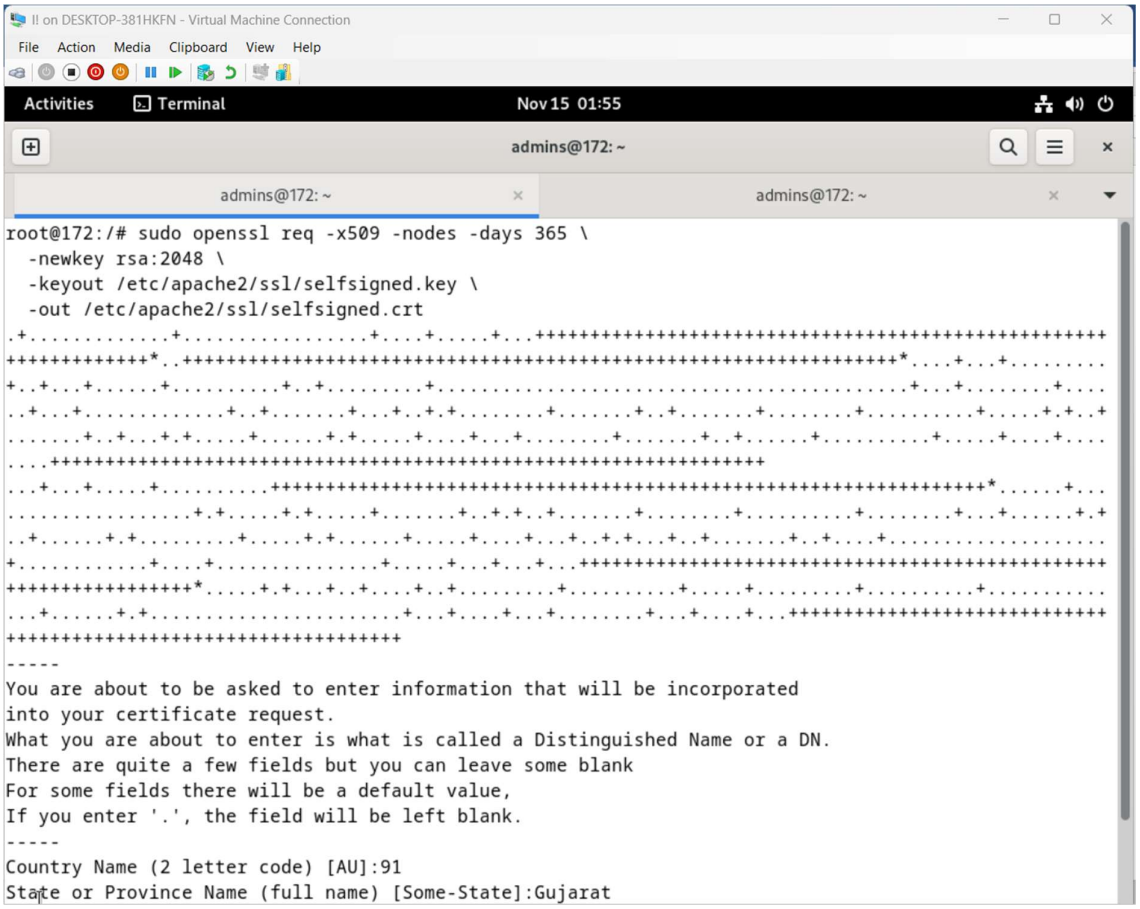


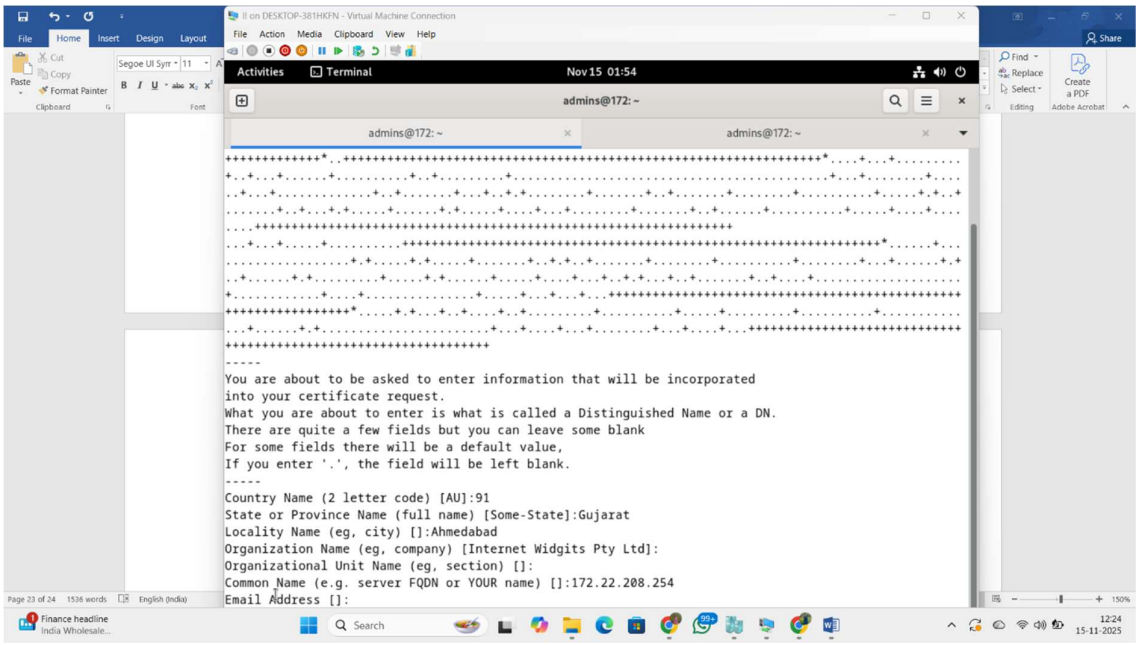
Explanation:

This is the folder where your key and certificate will be stored.

Step 3: Generate Self-Signed SSL Certificate

```
sudo openssl req -x509 -nodes -days 365 \
  -newkey rsa:2048 \
  -keyout /etc/apache2/ssl/selfsigned.key \
  -out /etc/apache2/ssl/selfsigned.crt
```





Explanation (Very Simple):

- -x509 → Create certificate
- -nodes → No password protection
- -days 365 → Valid for 1 year
- -newkey rsa:2048 → Create private key
- selfsigned.key → Private key
- selfsigned.crt → Public certificate

You will be asked some details:

Just press ENTER for all except Common Name:

Common Name:
Enter your server IP, for example:

172.22.208.254

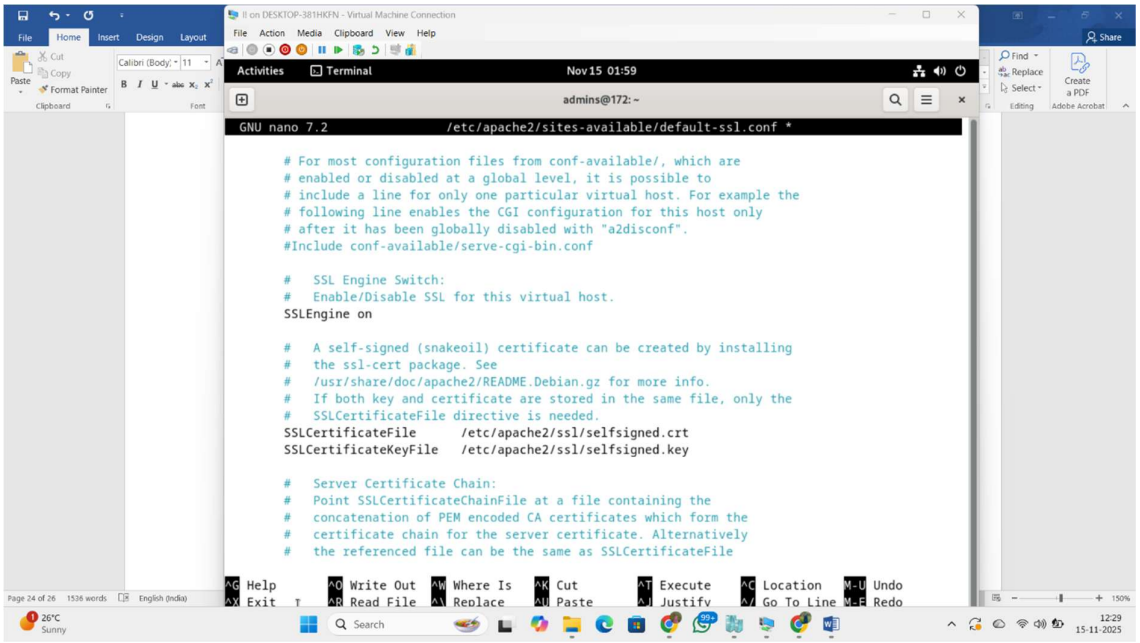
Step 4: Create HTTPS Virtual Host

Open SSL configuration file:

sudo nano /etc/apache2/sites-available/default-ssl.conf

Modify these lines:

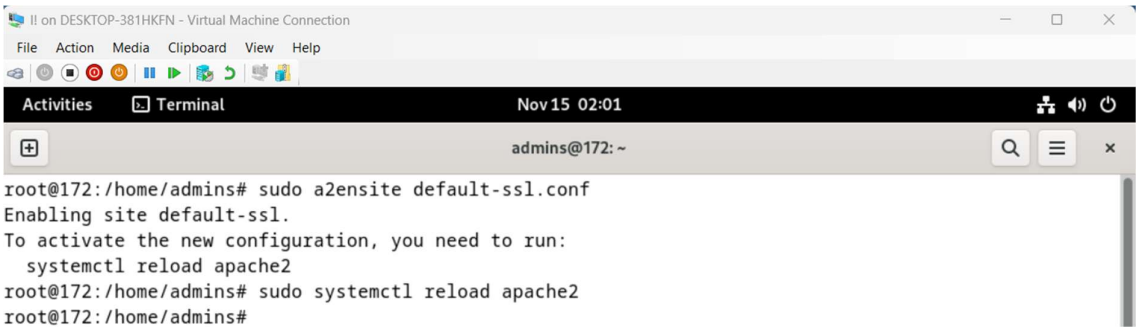
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/selfsigned.crt
SSLCertificateKeyFile /etc/apache2/ssl/selfsigned.key



Step 5: Enable the SSL Site

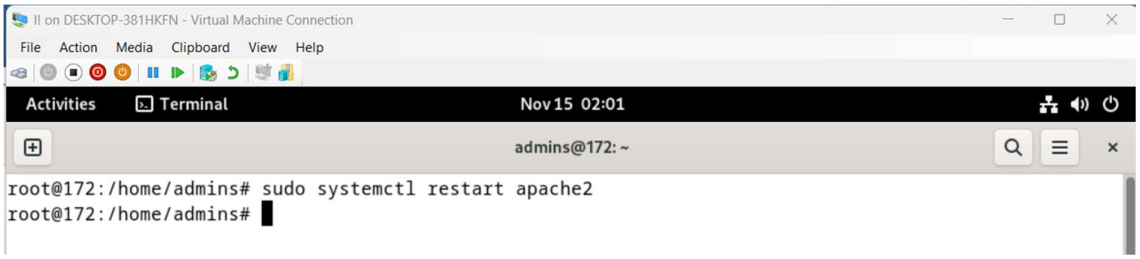
sudo a2ensite default-ssl.conf

sudo systemctl reload apache2



Step 6: Restart Apache

sudo systemctl restart apache2



Step 7: Test HTTPS

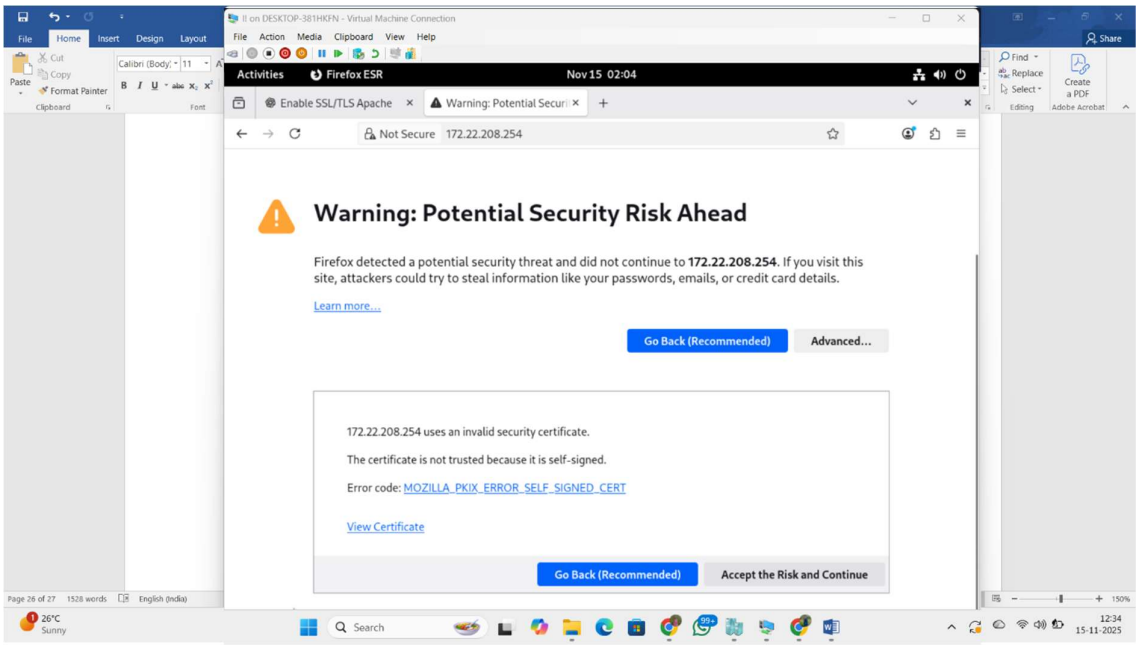
Open your browser and type:

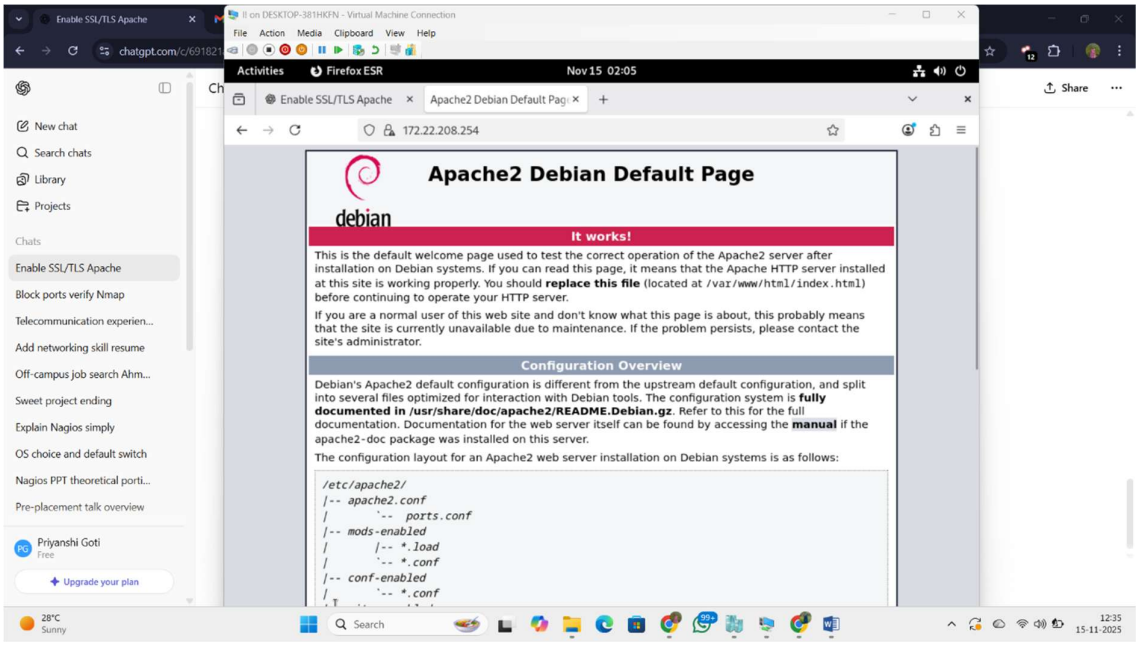
https://192.168.10.11

You will get a browser warning:

⚠ “Your connection is not private”

This is normal for self-signed certificates.
Click → Advanced → Proceed





RESULT: HTTPS successfully enabled!

Now your Apache site supports:

- Encrypted traffic
- Secure Nagios/Apache monitoring and Modern TLS security layer