TITLE : Installing and configuring a Web server on Ubuntu(server)

This project focuses on setting up and securing a web server on Ubuntu to provide a reliable and secure hosting environment

Overview of the project :

1. Custom partition of Ubuntu server

2. Check system after install

3. Install ssh

4. Create an administrative user for managing remote the server

5. Configure and use a static ip

6.Generate ssh key for administrative user

7. Install ufw and configure

8. Configure backup plan (python script + cronjob)

9.Implement cronjob with python script for backup

10. Install and configure Apache2

11. Install and configure Noip2 and portforwarding for DNS

12. Try to acces the Web server

13.Lessons Learned and Conclusion

Documentations of project

1. Custom partition of Ubuntu server

I created custom partitions for the Ubuntu server to optimize resource allocation and ensure better system management.

The fallowing partitons were configured :

/(root) – hold OS and include all basic files for OS , /boot – boot loader files , /home – user space, /swap space – virtual memory .

I chose to allocate for :

- **/ (filesystem)** – 12 gb
- **Swap space** – 2x memory ram = 4 gb
- **/ boot** – 100 mb and
- **/home** – 5gb

Screenshot showing the allocated disk space for each partion:

```
├─sda2                    8:2    0  3.9G  0 part [SWAP]
├─sda3                    8:3    0 1000M  0 part /boot
├─sda4                    8:4    0  4.9G  0 part /home
└─sda5                    8:5    0 11.7G  0 part /
```

2.Check system after install

After the system it's installed , it's always recommanded to check the integrity.

One of the checks is to verify the boot process . To verify If the boot process has errors during the (dmesg command) can be use. (Dmesg command) display all the errors registerered during the boot process .

In my case I had one error on graphics interface vmwgfx . This error doesn't affect the performance. I chose to put that in blacklist .

Sudo vim /etc/modprobe.d/blacklist.conf

I added this line to explain the reasons for blacklisting this module here.

Blacklist vmwgix

Save and exit (wq!)

After doing changes in kernel configuration modules , it's recommended to reload modules and reboot the system : **sudo update-initramfs -u / sudo reboot** .

3. Install ssh

Installing this service is straightforward . I followed these steps :

Sudo apt install openssh-server

Sudo apt install openssh-client

-Next step is to verify if the service is running :

Systemctl status ssh

The output indicated that systemctl was not installed

Sudo apt install systemd-sysv – at this I have to deal with the error : '' Error , pkgProblemResolver ''

Steps for fixing this problem :

Sudo apt upgrade-fix-missing | sudo apt install -f

(usage of this is to repair dependencies or partial installed packages in our systems)

Need to reload with the installation of systemctl

Sudo apt install systemd-sysv

Now we can check the status of SSH . The first observation is that the service is disable from starting at system boot . This can be fixed with the command : sudo systemctl enable ssh and a quick verification with systemctl to be ensure the service is enable now .

Finally can test the connection with : **ssh user@ip.addresss**

4. Create an administrative user for managing remote the server

Scope : In this example for security reason I chose to have only one user with remote acces to the webserver also this user will be an administrative user with root access or 'superuser' .

I will show the commands that I used to acomplish this .

Sudo useradd -m -d /home/admi1 -c 'admin_level' -s /bin/bash –uid 1050 admin1

Sudo usermod -aG sudo admin1

It is recommended to perform the following checks after creating the user:

Sudo /etc/password and sudo /etc/groups

It's recommended to change the password and set rules for example : password expiration and restrictions on frequent password changes etc.

Also it's recommanded for the password to followed these rules :

The length to be minimum 8 characters / To contain upper characters and lower characters / It's must to have special characters .

5. Configure and use a static ip

It's always better to configure a static IP addresss for web server ,because this offers stable connection. In my case I want to avoid problems with connections that can arise from the DHCP service assigning a new lease ip addresss .

Steps that I followed :

I installed arp to identify the range of available IP addressses.

Command : **sudo arp-scan --localnet**

The output of the command showed me the available range of IP addressses from my ISP , because the server is configured in Bridge adapter . (Bridge Adapter – the virtual network it will be in same network as the local physical network)

I pinged the IP addresss that I wanted to use .

The expected result:

- Host unreachable

And

- I performed an extra check with the : netstat -v / netscan -sc [ipv4-chosen]

The command provided :

Detailed information on available sections for : ip / icmp / tcp / udp with detailed information

-this information it's useful to determine whether the chosen IP addresss is in use.

However this don't ensure the fact the IP addresss will not be in use later. Since this is a home lab and I will proceed despite potential issues .

Implement the static IP – Steps :

Navigate to : /etc/netplan

Create a new configuration file : **touch 99-network_config.yaml**

Edit the file with vim :

* Important: when configuring static networking , ensure proper indentation , as this may cause errors when running the configuration file .

Config example :

```
root@webserver:/etc# cat netplan/01-Net_config.yaml
    network:
      version: 2
      renderer: networkd
      ethernets:
        enp0s3:
          dhcp4: false
          addresses: [redacted]
          routes:
            - to: default
              via: [redacted]
          nameservers:
              addresses: [8.8.8.8,8.8.4.4]
```

Why I am using Networkd ?

Networkd is part of Sytemd-networkd. It represents a service that handles availabe networks in systemd system . This service is used to manage the interface configuration of network and allows for settings such as : IP addressses , gateway , DNS-servers , VLANs.

Advantages of Networkd on Ubuntu : It is preferred standard for Ubuntu server , because it doesn't require supplementary dependencies and does not need to use a graphic interface .

To enhance security , following permissions are recommended :

- For the Netplan directory— **741** (permissions in octal mode)
- For the network_config.yaml configuration file — **600** (permissions in octal mode)

Before applying the new configuration it's always better to check with the command : **netplan try**

If there are no errors , you can proceed with the command : **netplan apply**

6.Generate ssh key for administrative user

In my case I generated the SSH key for the administrative user. Only this user can connect remotely to the server .

Command :

Sudo ssh-keygen -t rsa

- When asked where to save the key I chose my administrator home directory.
- For additional security I added a passphrase required when connecting via SSH.

Adherenced the key with the administrator user : **ssh-copy-id [username]@[Ipaddresss]**

Next step : Administrator user need to have the public key on the device that will use to connect to the server .

Enhance security for SSH:

This step involves modifying the file **/etc/ssh/sshd_config**

I use the following options :

PermitRootLogin no

Password Authentication no

AllowUsers [user] (this represent the administrator usef of the server)

Pubkey Authentication yes

AuthorizedKeysFile .ssh/authorized_keys

-after these change was made ssh service require a restart – **systemctl restart ssh**

It is important to do one more check on permissions . The permissions are recommended to be as follows :

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/id_rsa
chmod 644 ~/.ssh/id_rsa.pub
```

Last to see in real time logs can use : **sudo tail -f /vat/log/auth.log** .

7. Install ufw and configure

In many distributions, UFW comes install by default . To check I can use the command : **ufw –help**

In my case UFW is already install .

First configuration I will do is to add a new rule for port 22 which is used for ssh conection .

Command : **sudo ufw allow 22**

Enable this new rule so it runs every time when the system starts.

Command : **sudo ufw enable**

Check the status of ufw and the rules with :

Command : **sudo ufw status**

8. Configure backup plan (Python script)

I will add a new hardisk to my virtual machine . The storage size of hardisk is 5 GB and it will be used for backups

After adding the disk I check the device blocks to see if it was added :

Command : **lsblk -a** -> The output reveals the new device block - **/dev/sdb**

Next steps are to format the new disk and create a file system that can be mounted at the end of the configuration .

Steps :

1. Create new partion with a size of 3GB.

Fdisk /dev/sdb (options chosen : primary partitions , number 1 , block_size=+3072MB)

2. Create physical drive

Pvcreate /dev/sdb1 . Command for check – pvs

After this step it is important to update the kernel with command : sudo partprobe /dev/sdb

3. Create volume group (vg) with the name – backup

Vgcreate backup /dev/sdb1

4. Create a logical volume (lvm) that has 2.8GB with the name : webServer_bck

Lvcreate –name webServer_bck –size 2.8 backup . Command for check : lvs

5. Create a new file system (ext4) that can be mounted

mkfs -t ext4 /dev/backup/webServer_bck

6. Create a new directory in '/'to mount the new file system. The new directory name is : backup_Webserver.

7. Mount the new file system in backup_Webserver

Sudo mount /dev/backup/webServer_bck /backup_Webserver

Check : lsblk

To check free space of the directory : **df -h /backup_Webserver**

9. Ensure the file system is mounted after reboot the server . This step involves permanent mount of file system .

Sudo vim /etc/fstab

```
# /backup_Webserever was on /dev/sdb1/backup/webServer_bck , purpose backup for /etc

/dev/backup/webServer_bck /backup_Webserver ext4 defaults 0 0
```

Check for erorrs : **mount -a**

Test :

- create a new text file in /backup_Webserver that contains some text
- reboot server
- after the reboot : verify again the mount point , the file that was created and display the text from the file .

10. Change permissions , attributes , create new group for administration

- **Sudo groupadd -g 1051 -p admin1 admin_webserver**
- **Add in admin_webserver users : admin1 and root**
- **Sudo usermod -aG admin_webserver root**
- **Sudo usermod -ag admin_webserver admin**

  Check : cat /etc/group | grep -I admin

11. Change permissions recursively for /backup_Webserver

Sudo chown -R admin1:admin_webserver /backup_Webserver

12. Add 'sticky bit' . This prevents deletion of directory and files , these are allowed only of the owner of the directory

Sudo chmod 1751 /backup_Webserver

13. To copy files from webserver to backup I will use rsync .

14. It's time to create the environment for python to can create scripts for the backup .

- **Sudo apt update / sudo apt upgrade -y**
- **Install python – sudo apt install python3 python3-pip-y**
- **Install virtual environment for python – sudo apt install python3.12-venc**
- **Create new virtual environment – sudo python3 -m venv backup_env**
- **Activate virtual environment – sudo bash backup_env/bin/activate**

15. I was concept a script that will simulate the backup. To copy files from /etc to the backup directory .

```python
# Aceasta va fi un script care va folosi commanda rsync pentru a creea un backup pentru /etc in /backup_Webserver

import os
import subprocess

def backup_etc():
    # Definire cale sursă și cale destinație
    source = "/etc"
    destination = "/backup_Webserver"

    # Comanda rsync pentru backup cu simulare (dry-run)
    rsync_command = ["rsync", "-av", "--update", "--dry-run", source, destination]
    print(f"Comanda rsync: {rsync_command}")  # Afișăm comanda pentru verificare

    try:
        # Execută comanda rsync în modul de testare
        subprocess.run(rsync_command, check=True)
        print("Simulare backup completată cu succes! Nicio modificare nu a fost efectuată.")
    except subprocess.CalledProcessError as e:
        print(f"Eroare la rularea comenzii rsync: {e}")

if __name__ == "__main__":
    backup_etc()
```

Result of simulating script :

```
sent 76,511 bytes  received 5,590 bytes  164,202.00 bytes/sec
total size is 2,367,261  speedup is 28.83 (DRY RUN)
Simulare backup completată cu succes! Nicio modificare nu a fost
admin1@webserver:/backup_Webserver/python_scripts$
```

9.Implement cronjob with python script for backup

To work with cronjob, it is important to know 2 commands which are :

- **Crontab -e** (used to edit or to add a new cronjob)
- **Crontab -l** (used to visualize existing cronjobs)

The cronjob that I scheduled for my Python script looks like this :

```
0 1 * * 0 /usr/bin/python3 /backup_Webserver/python_scripts/py_script.py >>
/backup_Webserver/python_scripts/backup_log 2>&1
```

Explanation for what this cronjob does :

- This cronjob will start Sunday at exactly 1 AM.
- At the endl, added >> **/backup_log 2>&1** .It's cope is to store the error logs.

10. Install and configure Apache2

Install Apache2 : Command : **sudo apt-get install apache2** .

Check the status : **sudo systemctl status apache**.

After verifying the status I saw the need to run one more command . The command is :

sudo systemctl enable apache (this ensure the service will be active when the server starts)

Settings are found at : **/etc/apache2** .

The Index for the webpage can be found at : **/var/www/http/index.html** .

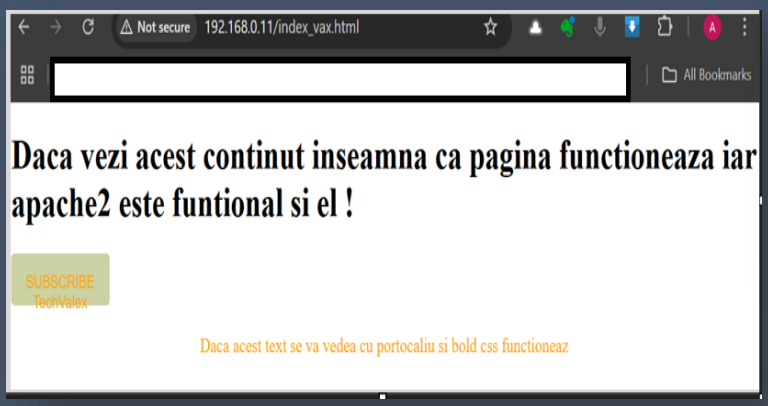
In my system I use UFW for firewall. I need to add a new rule for the HTTP port (for this lab I don't use the more secure version,HTTPS) . HTTP uses port 80.

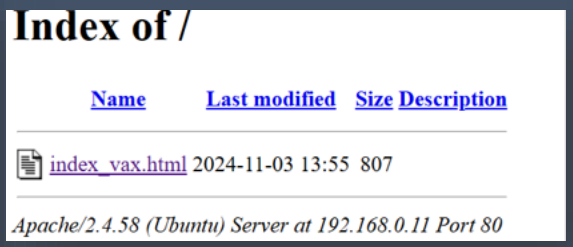
Set new rule at UFW :

- **Sudo ufw allow 80**
- **Sudo ufw enable**
- **Sudo ufw status**

Next step is to create a custom index and test it . (I just changed the index name and create some basic HTML and CSS elements , nothing special , becouse the scope of this project is the configuration and set up of a functional web server).

Test in browser the index with the custom page :





11. Install and configure Noip2 and portforwarding for DNS

I use for NO-IP for DNS . This isn't the best choice , but for my home lab and for this basic project, it works just fine , because I can show the DNS implementations. However in a future project I will choose a more suitable option that fits better in the enterprise domain .

Steps for setting up Noip2 :

- Create an account at Noip
- Manually install No-ip (this doesn't have a repository)

Commands for these :

```
1. cd /usr/local/src/
2. wget http://www.noip.com/client/linux/noip-duc-linux.tar.gz
3. tar xf noip-duc-linux.tar.gz
4. cd noip-2.1.9-1/
5. make install
```

- Configure apache for DNS

    - vim /etc/apache2/apache2.conf
    - Add this line : ServerName vaxtexting.serverhttp.com

- Create a new file for configuration in /etc/apache2/sites-available

The new file must contain this configuration :

```
<VirtualHost *:80>
    ServerAdmin admin@vaxtesting.servehttp.com
    ServerName vaxtesting.servehttp.com
    DocumentRoot /var/www/html

    <Directory /var/www/html>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/vaxtesting_error.log
    CustomLog ${APACHE_LOG_DIR}/vaxtesting_access.log combined
</VirtualHost>
```

- Testing the configuration — sudo apache2ctl -t

- Implement the configuration — sudo a2ensite vaxtesting.conf

- Restart Apache2 services — sudo reload apache2

- Set static ip for noip2 — sudo noip2 -I 192.168.0.11

- Reconfiguration noip2 — sudo /usr/local/bin/noip2 -C

- Port forwarding — this need to be done at the home local router . In the port-forwarding I use ip address of the web server and the port

Testing :

Important info : For this configuration to remain persistent after a system reboot it's important to create a service in systemd for that.

13.Lessons Learned and Conclusion

This project represented my first step into a complete practical implementation of a web server. It wasn't easy, but every challenge taught me something valuable. I encountered errors, made mistakes, and spent time troubleshooting, which helped me improve my understanding of system administration, network configuration, and scripting automation.

In conclusion, this project gave me confidence to tackle real-world scenarios. I realized that mistakes are part of learning, and solving them step by step makes me grow. I am proud of what I accomplished here, and this will be a foundation for my next steps in IT.