Born2beroot

Virtual Machines (VMs)

A virtual machine (VM) is software that emulates a physical computer, allowing multiple VMs to run on a single physical machine by sharing resources like CPU, memory, and storage. VMs are efficient, flexible, and isolated, making them ideal for running different operating systems and applications.

Advantages:

1. **Resource Efficiency:** Multiple OSs share one machine's resources.
2. **Isolation:** Issues in one VM don't affect others.
3. **Flexibility:** Easy to create, modify, or delete.
4. **Testing & Development:** Safe environment for experiments.
5. **Disaster Recovery:** Quick backup and restoration.
6. **Scalability:** Adapt to changing demands.
7. **Cost Savings:** Reduces hardware needs.
8. **Portability:** Easily transferable between machines.

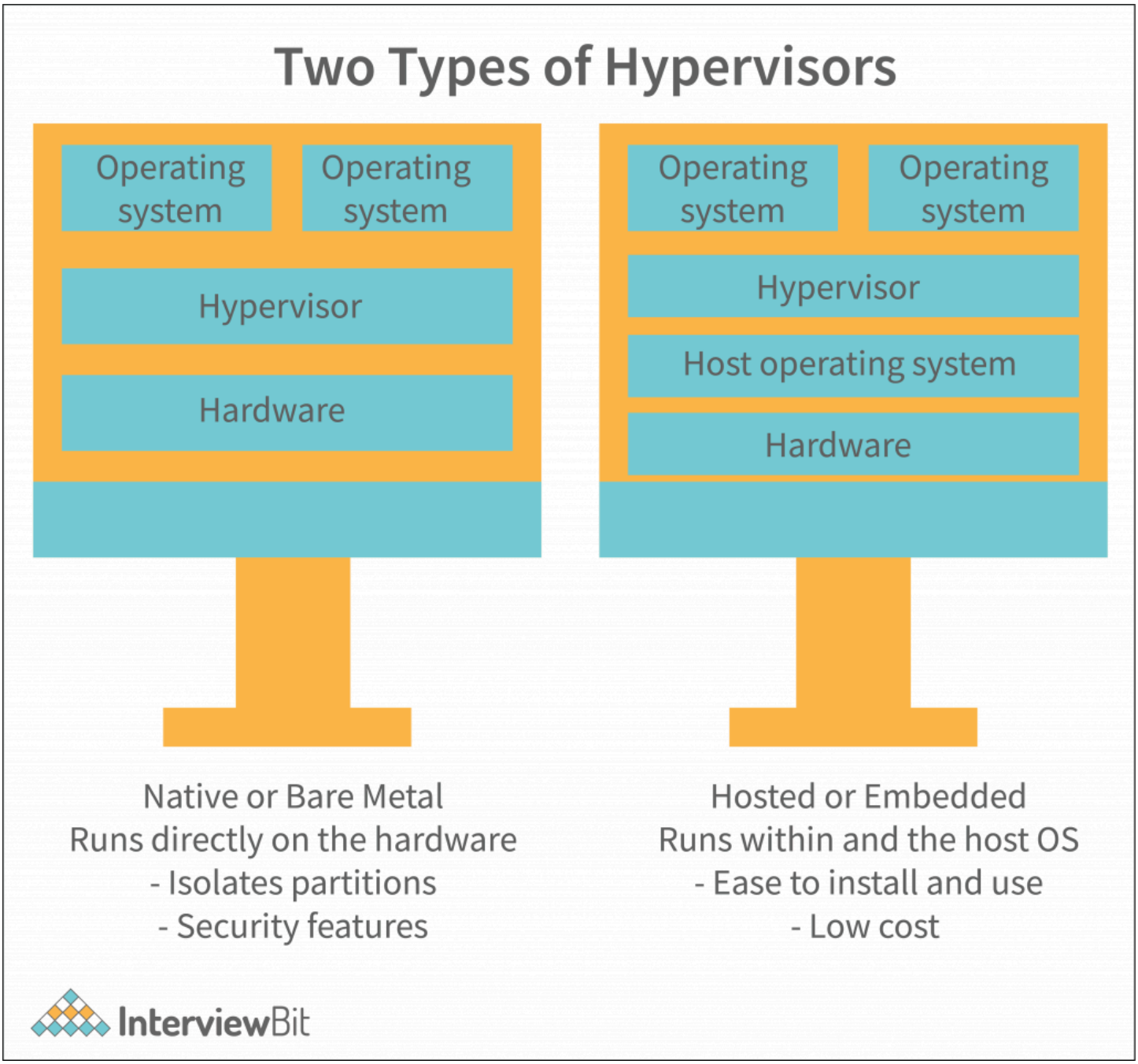
Types of Virtualization

1. Hardware Virtualization and Hypervisors

- **Hardware Virtualization** refers to creating virtual machines (VMs) where each VM behaves like a separate physical computer with its own CPU, memory, and storage.
- **Hypervisors**are the core technology that makes hardware virtualization possible. They are responsible for:
  - Dividing physical hardware into virtual resources.
  - Managing VMs and ensuring isolation between them.

  Types of Hypervisors in Hardware Virtualization:

  1. **Type 1 (Bare Metal)**:
     Runs directly on the hardware (no host OS). This is a pure implementation of hardware virtualization.
     Example: VMware ESXi, Microsoft Hyper-V, Citrix XenServer.
  2. **Type 2 (Hosted)**:
     Runs on top of a host OS. It virtualizes hardware by relying on the underlying OS to manage the physical resources.
     Example: VMware Workstation, Oracle VirtualBox.



2. Operating System Virtualization

- **OS Virtualization** is a different form of virtualization where the focus is not on hardware but on running multiple isolated environments (containers) on a **single OS kernel**.
- Containers share the host OS kernel, making this approach lightweight and efficient.

System administration

System administration is the management and maintenance of computer systems and networks. It involves tasks such as installing software, managing hardware, ensuring security, monitoring performance, and troubleshooting issues. A system administrator (or sysadmin) ensures that the systems are running smoothly, securely, and efficiently. They

also handle tasks like user management, backups, and system updates. Have you ever worked with system administration tasks?

Logical Volume Management

What is LVM?

LVM stands for **Logical Volume Management**. It’s a tool for managing your computer's disk storage in a smart and flexible way. Normally, when you divide your hard drive, you create fixed partitions. But LVM lets you manage storage like a pool of space that can be easily adjusted.

Think of it like this:

- Traditional partitioning is like dividing a pizza into fixed slices.
- LVM is like having dough that you can reshape and resize into new slices anytime.

This is very useful if your storage needs change over time.

How LVM Works

Here's a brief overview of LVM components and concepts:
1. Physical Volumes (PVs): These are the actual physical disks or partitions that are used by LVM. They are the lowest level of storage in LVM.
2. Volume Groups (VGs): A volume group is a collection of one or more physical volumes that are combined to form a single logical unit. VGs are used to allocate disk space to logical volumes.
3. Logical Volumes (LVs): These are the virtual partitions created within a volume group. They can be resized, moved, and snapshotted independently of the physical storage.
4. Physical Extents (PEs): The smallest unit of disk space that can be allocated to a logical volume. A physical volume is divided into physical extents, which are then used to create logical volumes.
5. Snapshots: LVM allows for the creation of snapshots of logical volumes. A snapshot is a point-in-time copy of a logical volume, which can be used for backup or testing purposes.
6. Striping: LVM supports striping, which is the process of splitting data across multiple physical volumes to improve performance. This can be done at the volume level or within a single logical volume.
7. Mirroring: LVM can also mirror data across multiple physical volumes to provide redundancy and improve data integrity.

Key Features of LVM

- **Resize Storage:** You can expand or shrink logical volumes without rebooting your system.

- **Snapshots:** Create a backup of your data at a specific moment, like a "save point" in a video game.
- **Combine Disks:** Combine multiple hard drives into one big storage space.
- **Performance:** Use **striping** to spread data across multiple disks for faster speed or **mirroring** to duplicate data for safety.

Example

a. Imagine you have two hard drives, one is 500GB, and another is 1TB.
b. You turn them into **Physical Volumes** (PVs).
c. Combine these PVs into one **Volume Group** (VG) of 1.5TB.
d. Inside this VG, you create **Logical Volumes**(LVs):
   - 300GB for your operating system.
   - 200GB for backups.
   - The rest for your personal files.

Now, if you need more space for backups, you can shrink the personal files LV and grow the backup LV.

Encryption:
Encryption is the process of converting readable data into an unreadable format using algorithms and keys to protect it. In Linux, encryption for Logical Volume Manager (LVM) is commonly done using **dm-crypt**, a kernel-level module that transparently encrypts block devices, including LVM logical volumes.

Purpose of Several Partitions:
Having multiple partitions instead of a single one improves organization, security, and performance. If one file system becomes corrupted, only that partition is affected. Other benefits include multi-boot capability, easier backups, and better file system flexibility.

Linux File System

A file system in Linux organizes and stores files on storage devices such as hard drives, SSDs, or network storage. It determines how data is stored, retrieved, and managed. The file system tracks files, directories, and disk space, allowing efficient and organized data access.

Types of File Systems:

a. **Ext (Extended File Systems):** Includes ext, ext2, ext3, and ext4.
   - **Ext**: First Linux system supporting up to 2TB of data.
   - **Ext2**: First Linux system supporting up to 2TB of data.
   - **Ext3**: Upgraded from Ext2, adds journaling but lacks file recovery or snapshot support.
   - **Ext4**: Faster and SSD-compatible, widely used as the default Linux file system.
b. **FAT32 and NTFS:** Windows file systems also compatible with Linux for removable storage.

c. **JFS (Journaled File System):** Developed by IBM for AIX Unix, suitable for systems with limited CPU power.
d. **Swap:** Used for memory paging or system hibernation; swap size is typically equal to RAM.

Linux Directory Structure:

Linux uses a hierarchical directory structure to store and access files efficiently. Key directories include:

- **/:** Root directory, the top-level folder.
- **/bin:** Essential binaries (commands) for system boot and operation.
- **/sbin:** System maintenance and administrative binaries.
- **/etc:** System-wide configuration files.
- **/home:** Personal files and directories for users.
- **/usr:** Multi-user utilities, applications, and libraries.
- **/var:** Variable files like logs and databases.
- **/tmp:** Temporary files for system and users.
- **/boot:** Boot files, including the Linux kernel.
- **/dev:** Device files for hardware interaction.
- **/proc:** Virtual filesystem for system and process info.
- **/mnt and /media:** Mount points for external file systems.

What is Mounting

Mounting in Linux is the process of making a file system available for use by attaching it to a specific directory in the existing file system hierarchy. This allows users and applications to access the files and directories on the mounted file system as if they were part of the main file system. The mount command is used to perform this operation, specifying the device and the mount point.

Why is mounting needed?

Imagine you have a USB drive.

- When you connect it to your computer, Linux does not show its files immediately.
- You must tell Linux:
  - **"Here is my USB drive (device)."**
  - **"I want to see it in this folder (mount point)."**

Example

1. You connect a USB drive to your computer.
   - The system recognizes it as something like /dev/sdb1 (this is the device path).
2. You create a folder (mount point) where you want to see the files:

```
sudo mkdir /mnt/usb
```

3. You "mount" the device into that folder:

```
sudo mount /dev/sdb1 /mnt/usb
```

Now the folder `/mnt/usb` will show all the files from your USB drive.

What is SUDO?

sudo stands for "superuser do" is a command in Linux that allows a permitted user to execute a command as the superuser or another user, as specified by the security policy. The sudo command is used to perform administrative tasks that require elevated privileges, such as installing software, modifying system files, or changing system settings.

Instaling sudo:
In order to install sudo, first we need to login as root.

```
su -
<enter the root password>
apt update
apt upgrade
apt install sudo
```

Now, we need to give our regular user the ability to use sudo, by adding the user to the sudo group (and checking that it was added successfully).

- `# sudo usermod -aG sudo <username>`
- `# getent group sudo`

- To check if we now have sudo privileges, we can exit the root session to go back to our own user, then run the following command:
  - `sudo whoami`
  - `<enter password>`
  - `root`

Once our password entered, we should get, as an answer, **root**. If that doesn't work, we might have to logout and log back in for the changes to take effect.
As a very last resort, if the user still has no sudo privileges after logging out and back in again, we will have to modify the **sudoers.tmp** file from the root session with the following commands:

- `su`
- `sudo visudo`

search for “User privileges” and write this line:

- `<username>  ALL=(ALL:ALL) ALL`

Configuring Sudoers Group

- For security reasons, the Born2beroot subject requires a few more sudo configurations. We must:
  - Limit authentication for sudo to 3 tries for a bad password.
  - Define a custom message in case of a bad password.
  - Log sudo commands in /var/log/sudo/.
  - And activate TTY to stop malicious software from giving itself root privileges via sudo.

In order to do this, we need to add the following lines to the sudoers.tmp file (as above, we must open it from the root session, with the sudo visudo command to be able to write changes to it):

or:

1. First type sudo nano /etc/sudoers to go the sudoers file
2. Now edit your sudoers file to look like the following by adding in all of the defaults in the image below -

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
Defaults        badpass_message="Password is wrong, please try again!"
Defaults        passwd_tries=3
Defaults        logfile="/var/log/sudo/sudo.log"
Defaults        log_input, log_output
Defaults        requiretty_

# Host alias specification
```

Packet Management in Debian

Package Management in Debian-Based Systems

In Linux systems like Debian, software installation relies on packages, which contain the necessary files to execute specific functionalities. Understanding package management is crucial, especially for your Born2beroot virtual machine.

APT (Advanced Package Tool):

APT is a command-line package management tool in Debian that simplifies the installation, upgrade, and removal of software, along with dependency management.

Key Commands in APT:

- **Install Packages:** apt install [package]
- **Update Package Lists:** apt update
- **Upgrade Installed Packages:** apt upgrade
- **Remove Packages:** apt remove [package]
- **Search for Packages:** apt search [package]
- **Clean Up Unnecessary Dependencies:** apt autoremove

Aptitude:

Aptitude is similar to APT but offers a text-based, menu-driven interface, making it user-friendly for those preferring graphical workflows. It includes advanced features like package filtering, previewing, and a robust dependency resolver.

Differences Between APT and Aptitude:

a. **Interface:** APT uses a command-line interface; Aptitude provides a text-based menu interface.
b. **Functionality:** Aptitude offers additional features, like filtering and previewing.
c. **User Preference:** Aptitude suits users who prefer interactive tools, while APT is ideal for command-line users.
d. **Resource Usage:** Aptitude uses slightly more resources due to its interactive interface.

Summary:
Both APT and Aptitude are efficient tools for managing packages in Debian-based systems. Choosing between them depends on user preference and workflow.

AppArmor

AppArmor is a Linux security module that restricts the capabilities of individual programs by enforcing security policies. It ensures that programs can only perform actions explicitly allowed by their profiles. AppArmor comes preinstalled on Debian.

AppArmor uses **profiles** to define security policies for each program. A profile is a set of rules specifying what actions a program can perform and on which files or directories. These profiles are stored in the /etc/apparmor.d/ directory.

AppArmor has **two profile modes**:

1. **Enforced mode**: Enforces the profile's rules and logs any violation attempts in /var/log/syslog.
2. **Complain mode**: Logs violation attempts without enforcing any rules.

The apparmor-utils package provides command-line tools for managing AppArmor. With these tools, you can change AppArmor's execution mode, check the status of profiles, or create new profiles.

Common AppArmor Commands:

1. **Check AppArmor's status**:
   - `aa-status`
2. **Switch a profile to complain mode**:
   - `sudo aa-complain /path/to/profile`

   a. Replace /path/to/profile with the actual path to the profile.
3. **Switch a profile to enforced mode**:
   - `sudo aa-enforce /path/to/profile`

UFW (Uncomplicated Firewall)

What is UFW?

UFW (Uncomplicated Firewall) is a user-friendly tool for managing *iptables* firewall rules in Linux. It simplifies the process of configuring a firewall, allowing users to set up and manage firewall rules easily without deep knowledge of *iptables*. UFW is designed to be straightforward, making it a popular choice for managing firewall settings on Linux systems.

IP Addresses & Ports

When discussing networks and traffic, understanding two key concepts is important: **IP addresses** and **ports**.

- **IP Address**: A unique identifier assigned to each device in a network, enabling communication with other devices. It can be:
  - IPv4 (most common)
  - IPv6 (newer version)
- **Port**: A communication endpoint that helps a computer manage multiple types of traffic at the same IP address. Ports are numbered from **0 to 65535**, with some common examples:
  - Port 80: HTTP
  - Port 443: HTTPS
  - Port 22: SSH

Together, an **IP address** and a **port number** form a *socket*, which identifies a specific process or service on a device. This allows multiple services to run simultaneously on the same machine.

nstalling & Configuring UFW

The firewall must be active when launching your virtual machine. Follow these steps:

1. **Update and install UFW:**
   - `sudo apt update && sudo apt upgrade && sudo apt install ufw`
2. **Enable UFW:**
   - `sudo ufw enable`
3. **Check UFW Status:**
   - `sudo systemctl status ufw`

a. You should see "active" in green.

4. **Allow only port 4242:**
   - `sudo ufw allow 4242`
5. **Manage Rules:**
   - To list all rules with indexes:
     - `sudo ufw status numbered`
   - To delete unnecessary rules:
     - `sudo ufw delete <rule_number>`

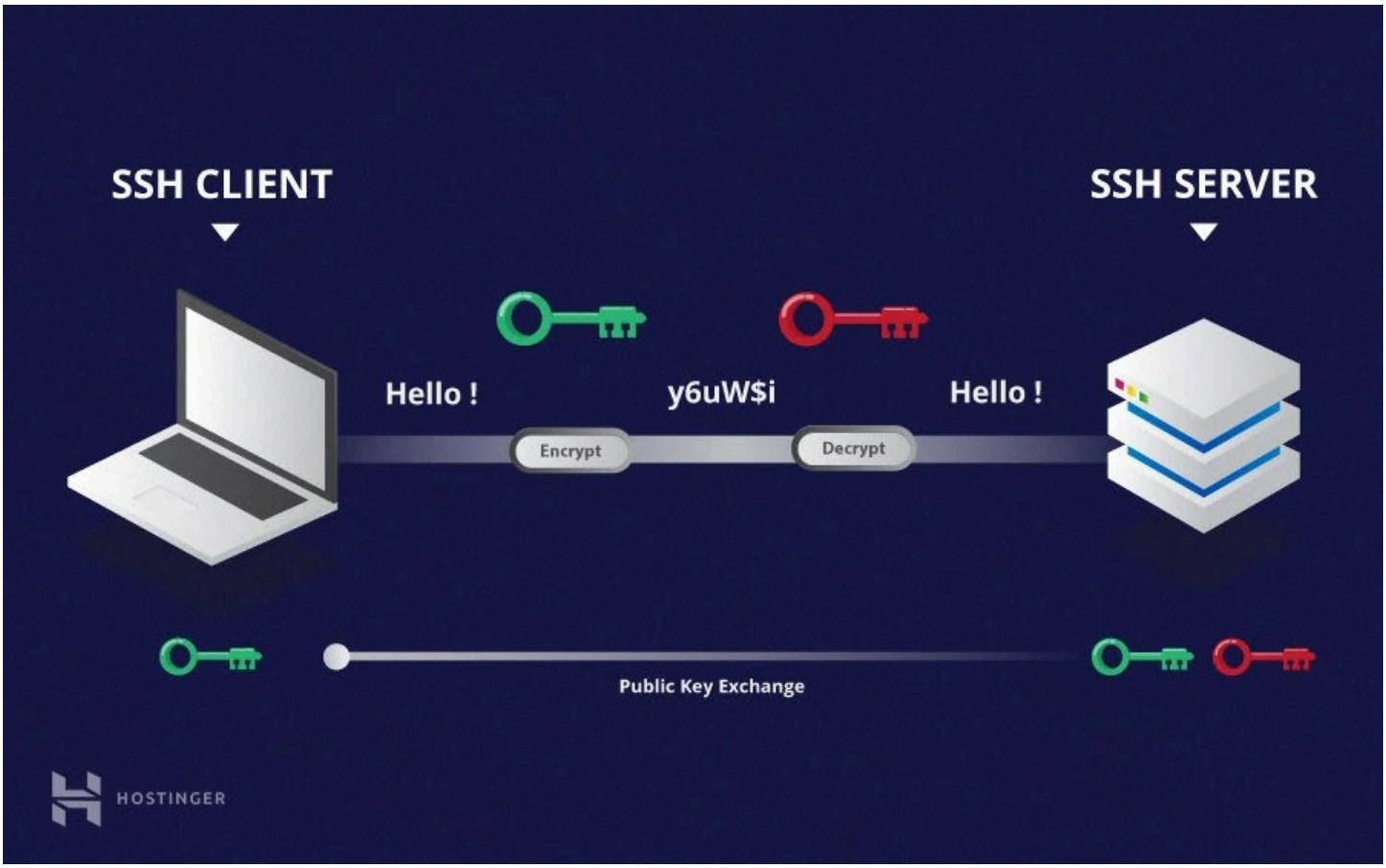
Summary

UFW simplifies firewall configuration on Linux systems. Understanding IP addresses, ports, and basic UFW commands allows you to manage network traffic effectively and securely.

SSH

- **What is SSH?**
  SSH (Secure Shell) is a cryptographic network protocol used for secure communication over an unsecured network. It provides a secure way to access and manage remote systems or devices over a network, such as the internet. SSH is commonly used for tasks like remote command execution, file transfer, and tunneling network connections.

- **How does SSH works?**
  SSH commonly uses asymmetric encryption as part of its authentication process. Asymmetric encryption, also known as public-key cryptography, works with a pair of cryptographic keys: a public key and a private key.
  The public key is used to encrypt data, while the private key is used to decrypt it. Messages encrypted with the public key can only be decrypted with the corresponding private key, ensuring secure communication without the need to share secret keys.



Installing & Configuring SSH
In Born2beroot, we are asked to install this protocol and route it trough the 4242 port. OpenSSH is the most popular and widespread tool, so let's install that one. Since we want to be able to connect to our Born2beroot machine from another machine, we need the openssh-server packet. In order to connect to another machine from the Born2beroot machine, we would need the openssh-client packet.

- `sudo apt update && sudo apt upgrade`
- `sudo apt install openssh-server`
- `sudo systemctl enable ssh.service`

To check SSH is up and running we can use the command line:

- `sudo systemctl status ssh.service`

We should see "active" in green.
Now we need to modify the ports that SSH is listening to, that can be done by editing the ssh configuration file:

- sudo nano /etc/ssh/sshd_config

The line we are looking for is towards the beginning of the file and reads "**#Port 22**". We want to **uncomment** that and change it to "**Port 4242**"

- ...
- Include /etc/ssh/sshd_config.d/*.conf

-
- Port 4242
- #AddressFamily any
- ...

Also the subject mentioned "For security reasons, it must not be possible to connect using SSH as root.", for that we can **uncomment** the rule **PermitRootLogin** and set it to **no**

- ...
- # Authentication:
-
- #LoginGraceTime 2m
- PermitRootLogin no
- #StrictModes yes
- ...

```
Include /etc/ssh/sshd_config.d/*.conf

Port 4242
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

Then, we have to restart the ssh service for the change to take effect.

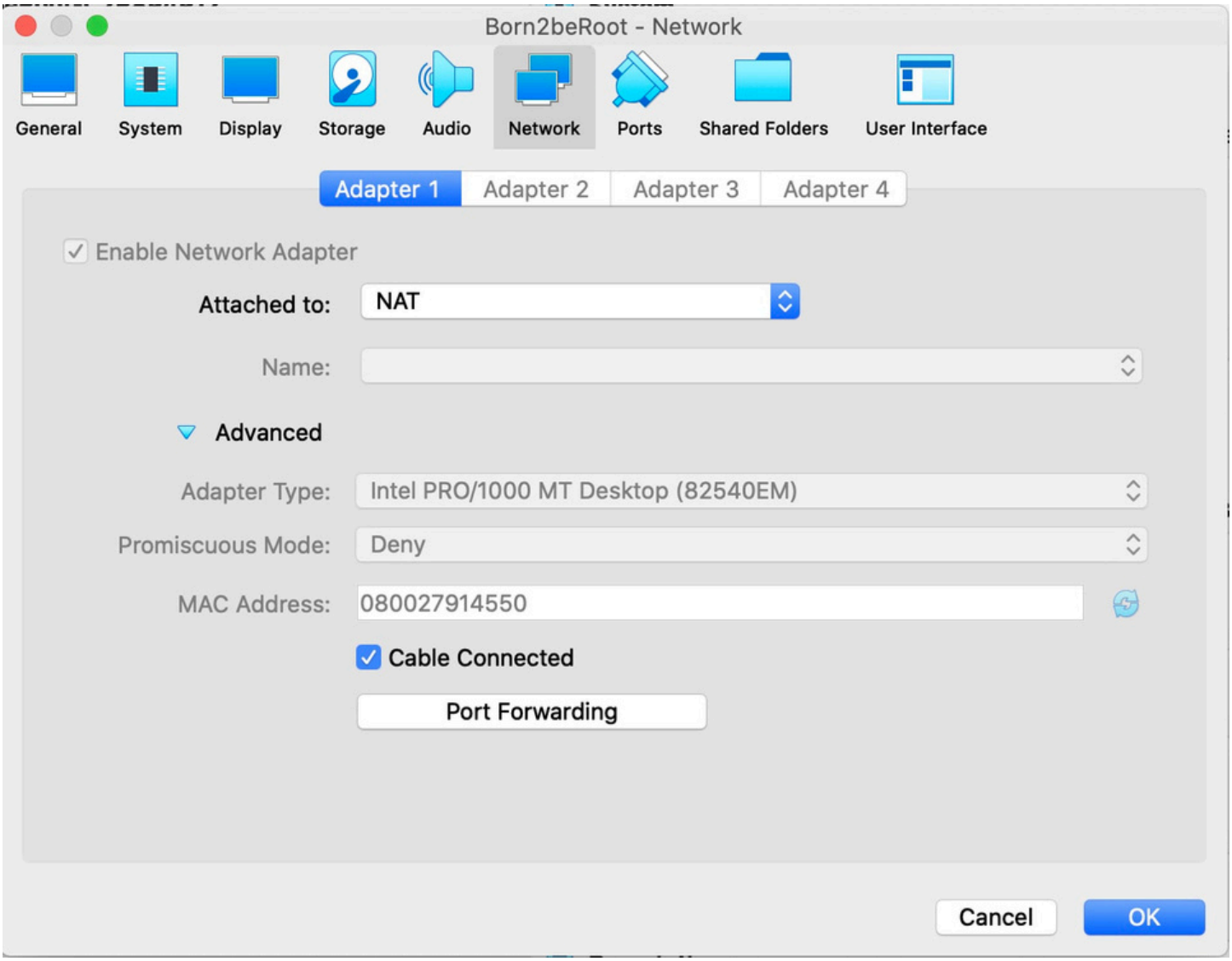
- sudo systemctl restart ssh

If you want to generaye your public & private keys you can use the command ssh-keygen. Let’s not forget to tell our firewall to authorize the 4242 port connection! We might also have to delete a new rule about Port 22 which was added automatically with OpenSSH’s installation.

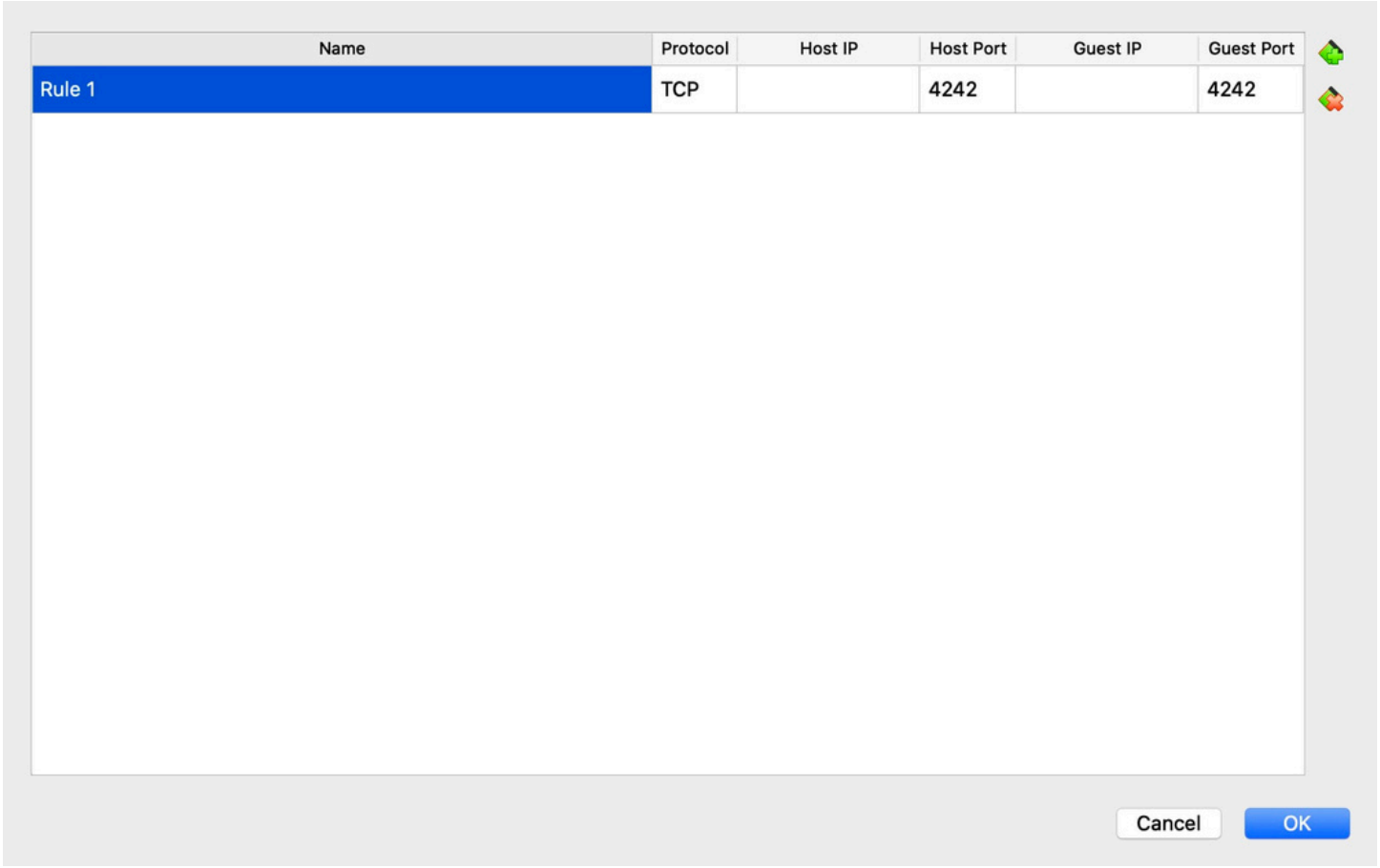
If you want to generaye your public & private keys you can use the command **ssh-keygen**.
Let's not forget to tell our firewall to authorize the 4242 port connection! We might also have to delete a new rule about Port 22 which was added automatically with OpenSSH's installation.
cmd for delete a Port

- `sudo ufw delete <rule_number>`

Port Forwarding in VirtualBox:
Before we can connect to the virtual machine from another computer via SSH, we have to make a little adjustment in VirtualBox. Indeed, the connection will be refused until we forward the host port to the VM port.
In VirtualBox, select the Born2beroot machine and go into configuration settings.



Then, go to network >> Adapter 1 >> Advanced >> Port Forwarding. Then we will redirect the host port 4242 to the guest port 4242 like this:



Finally, on your virtual server, we are going to restart the SSH server once again and check its status:

- `sudo systemctl restart ssh`
- `sudo systemctl status ssh`

Logging into the Born2beroot Server via SSH:
Now that we have configured everything, we can check the SHH connection by attempting to log into the Born2beroot virtual machine from the host machine terminal. Of course, the virtual machine must be turned on to be able to connect to it.
From the host machine's terminal, we can connect via SSH with this command:

- `ssh <username_server>@<server_IP_address> -p <ssh_port>`

The **username** will of course be that of the virtual machine user and the port will be **4242**. But what is the IP address of our Born2beroot server? Since the virtual machine shares the host's IP address, we can simply use the localhost IP address. Localhost is an internal shortcut that every computer uses to refer to their own IP address. The localhost IP is **127.0.0.1**.
So we can transcribe the previous command in one of the two following ways:

- ssh <user>@localhost -p 4242
- or
- ssh <user>@127.0.0.1 -p 4242
- <enter user passwor>

Once we enter the user password, we can control the virtual machine from the outside! Let's note that the command prompt has changed and now shows the virtual machine's hostname.

In order to put an end to the SSH connection, all we need to do is:

- exit

Password Policy

The subject consist of having a strong password policy, we have to implement the following reguirement:

- Your password has to expire every 30 days.
- The minimum number of days allowed before the modification of a password will be set to 2.
- The user has to receive a warning message 7 days before their password expires.
- Your password must be at least 10 characters long. It must contain an uppercase letter, a lowercase letter, and a number. Also, it must not contain more than 3 consecutive identical characters.
- The password must not include the name of the user.
- The following rule does not apply to the root password: The password must have at least 7 characters that are not part of the former password.
  In order to configure the first **3 rules**, we have to edit the **/etc/login.defs** file:
  - sudo nano /etc/login.defs

  And find the "**Password aging controls"** section to change the values such that

  - PASS_MAX_DAYS 30
  - PASS_MIN_DAYS 2
  - PASS_WARN_AGE 7

However, these changes will not automatically apply to preexisting users. For both root and our first user, we need to use the chage command to enforce these rules. The **-l** flag is available to display which rules apply to a certain user.

- sudo chage -M 30 <username/root>
- sudo chage -m 2 <username/root>
- sudo chage -W 7 <username/root>
- sudo chage -l <username/root>

For the rest of the rules, we will have to install the password quality verification library.

- sudo apt install libpam-pwquality

Then, we must edit the **/etc/security/pwquality.conf** configuration file. Here, we will need to remove the comment sign **(#)** and change the values of the various options. We will end up with something like this:

```
# Number of characters in the new password that must not be present in the
# old password.
difok = 7
# The minimum acceptable size for the new password (plus one if
# credits are not disabled which is the default)
minlen = 10
# The maximum credit for having digits in the new password. If less than 0
# it is the minimun number of digits in the new password.
dcredit = -1
# The maximum credit for having uppercase characters in the new password.
# If less than 0 it is the minimun number of uppercase characters in the new
# password.
ucredit = -1
# ...
# The maximum number of allowed consecutive same characters in the new password.
# The check is disabled if the value is 0.
maxrepeat = 3
# ...
# Whether to check it it contains the user name in some form.
# The check is disabled if the value is 0.
usercheck = 1
# ...
# Prompt user at most N times before returning with error. The default is 1.
retry = 3
# Enforces pwquality checks on the root user password.
# Enabled if the option is present.
enforce_for_root
# ...
```

User And Group Management

User management

For the **evaluation**, we must be able to show a **list of all users**, **add** or **delete** user accounts, change their **usernames**, add or remove them from groups, etc. The following commands are necessary to do all of that:
useradd <username> : creates a new user.
usermod <username> : changes the user's parameters: **-l** for the username, **-c** for the full name, **-g** for groups by group ID.
userdel -r <username> : deletes a user and all associated files.
id -u <username> : displays user ID.
users : shows a list of all currently logged in users.
cat /etc/passwd | cut -d ":" -f 1 : displays a list of all users on the machine.
cat /etc/passwd | awk -F ":" '{print $1}' : same as above.

Group Management

In the same way, we will have to manage user groups. Our default personal user must be in the **sudo** and **user42 groups**. The following commands need to be mastered for the evaluation:
groupadd : creates a new group.
gpasswd -a : adds a user to a group.
gpasswd -d : removes a user from a group.
groupdel : deletes a group.
groups : displays the groups of a user.
id -g : shows a user's main group ID.
getent group : displays a list of all users in a group.

Monitoring Script

The script:
The last thing we have to do for mandatory part is a bash script named **monitoring.sh**, it must display the following information every 10 minutes broadcasted on evey terminal connected to our virtual machine:

- The architecture of your operating system and its kernel version:

```
# uname --all
```

- The number of physical processors.

```
# lscpu | grep 'Socket(s)' | awk '{print $2}'
```

- The number of virtual processors.

```
# lscpu | grep 'Core(s)' | awk '{print $4}'
```

- The current available RAM on your server and its utilization rate as a percentage.

```
# free --mega | grep 'Mem' | awk '{print $3"/"$2"MB"}'
# free --mega | grep 'Mem' | awk '{printf "%.2f%%\n", $3 / $2 * 100}'
```

- The current available memory on your server and its utilization rate as a percentage.

```
# df -h --total | grep 'total' | awk '{print $3 "/" $2}'
# df -h --total | grep 'total' | awk '{printf "%.2f%%\n", $3 / $2 * 100}'
```

- The current utilization rate of your processors as a percentage.

```
# mpstat | grep 'all' | awk '{print 100 - $13"%"}'
```

- The date and time of the last reboot.

```
# who -b | awk '{print $3 " " $4}'
```

- Whether LVM is active or not.

```
# if [ $(lsblk | grep 'lvm' | wc -l) == 0 ]; then echo "No"; else echo "Yes"; fi
```

- The number of active connections.

```
# netstat -t | grep 'tcp' | wc -l
```

- The number of users using the server.

```
# who | awk '{print $1}' | sort -u | wc -l
```

- The IPv4 address of your server and its MAC (Media Access Control) address.

```
# echo "IP " && hostname -I && ip link | grep 'link/ether' | awk '{printf "(%s)", $2}'
```

- The number of commands executed with the sudo program.

```
# grep 'COMMAND' /var/log/sudo/sudo.log | wc -l && echo "cmd"
```

NOTE ⚠️ : Some of the above command won't work without root permission, we need to login as root and create the monitoring.sh file on thier session. We must also grant the file execution righ:

```
# chmod 755 monitoring.sh
```

The command `mpstat` is defined on `sysstat` package, also as the command `netstat` on `net-tools`.

```
# apt install sysstat
# apt install net-tools
```

- The Wall Command:
  The wall command in Linux is used to broadcast a message to all users currently logged into the system. It sends a message to all terminals, including those that are idle or inactive. This can be useful for system administrators to communicate important information or announcements to all users at once.
  Example:

```
# echo "System will be rebooted in 10 minutes. Please save your work." | wall
or
# wall "System will be rebooted in 10 minutes. Please save your work."
```

- The Cron Service:
  Cron is a program that enables the execution of scripts or software in an automatic way, at a certain date and time or at a specified interval. It is installed by default in Debian (we can check this with the `apt list cron` command). To be certain it will run at system startup, we should enable it:

```
# systemctl enable cron
```

  Cron uses crontab files to schedule jobs. Each user can have one, or many. As the root user, we will now create one with the following command:

```
# crontab -e
```

  The syntax of a cron file might seem obscure, but it's not too hard to wrap your head around:

```
* * * * * <command to execute>
```

here is what each star represents:

```
.----------------------- minutes (0-59)
| .--------------------- hours (0-23)
| | .------------------- day of the month (1-31)
| | | .----------------- month of the year (1-12)
| | | | .--------------- day of the week (0-6, 0 = sunday)
| | | | |
* * * * * <command to execute>
```

By replacing the stars with numerical values, we can define when our command must be executed.
So is this the way we should write "every 10 minutes"?

```
        10 * * * * bash /root/monitoring.sh
```

Almost, but no. This instruction means "execute this at the tenth minute of each hour, every day of every month". So our monitoring script won't execute every 10 minutes, but only at midnight 10, 1:10, 2:10, 3:10, 4:10 and so on.
So how are we supposed to say "every 10 minutes", then? Well, we can simply "divide" the minute star by 10.

```
*/10 * * * * bash /root/monitoring.sh
```

If the wall command wasn't incorporated directly into the monitoring.sh script, we can pipe it into this cron rule, like so:

```
*/10 * * * * bash /root/monitoring.sh | wall
```

Evaluation FAQ

- **How a virtual machine works?**

  A virtual machine (VM) is a software-based version of a computer that uses resources like CPU, memory, and storage from a physical host computer or remote server. It behaves like an actual computer, often running a different operating system. VMs are isolated from the host system, preventing interference between the software inside the VM and the host computer's operating system.

- **Why choice Debian not Rocky?**

Here are some reasons why you might choose Debian over Rocky Linux:

a. **Stability**: Debian is well-known for its stability and reliability. It's an excellent choice if you need a system that "just works" for a long time without frequent updates.
b. **Package Management**: Debian uses the **APT package manager**, which is considered user-friendly and has a large repository of software.
c. **Community Support**: Debian has a huge community, so finding support, documentation, and tutorials is relatively easy.
d. **Customizability**: Debian offers more control during installation, allowing you to tailor the system to your needs.
e. **Wide Hardware Support**: Debian supports many architectures, making it suitable for a broader range of devices, from servers to IoT devices.

- **The basic differences between Rocky and Debian?**

Origin:

  - Rocky Linux is a free, community-driven alternative to Red Hat Enterprise Linux (RHEL).
  - Debian is a versatile, community-driven distribution, known for its stability and wide support across architectures.
- **Package Management**:
  - Rocky Linux uses **RPM** and **YUM/DNF**.
  - Debian uses **APT** and **DEB**.
- **Target Audience**:
  - Rocky Linux is aimed at **enterprise and server environments**.
  - Debian is suitable for **developers, servers, and desktop users**.
- **Release Cycle**:
  - Rocky Linux has long-term support (LTS), similar to RHEL (about 10 years).
  - Debian has a stable release cycle, with updates for about 5 years.
- **Desktop Environment**:
  - Rocky Linux is **server-focused** and doesn't come with a default desktop.
  - Debian offers various desktop environments during installation.
- **Security and Support**:
  - Rocky Linux offers **enterprise-grade security** and support.
  - Debian is known for its **community-driven security** and stability.

- **The purpose of virtual machines?**

  Virtual machines (VMs) allow you to run multiple operating systems on a single physical computer. They are used to create isolated environments for different tasks, such as testing, running software that requires different OS versions, or separating applications for security reasons.

- **What is the difference between aptitude and APT (Advanced Packaging Tool)?**
  a. **Interface:** APT uses a command-line interface; Aptitude provides a text-based menu interface.
  b. **Functionality:** Aptitude offers additional features, like filtering and previewing.
  c. **User Preference:** Aptitude suits users who prefer interactive tools, while APT is ideal for command-line users.
  d. **Resource Usage:** Aptitude uses slightly more resources due to its interactive interface.

Password Rules

For the password rules, we use the password quality checking library and there are two files the common-password file which sets the rules like upper and lower case characters, duplicate characters etc and the login.defs file which stores the password expiration rules (30 days etc). Sudo nano /etc/login.defs Sudo nano /etc/pam.d/common-password

What is LVM

Logical Volume Manager – allows us to easily manipulate the partitions or logical volume on a storage device.

UFW (Uncomplicated Firewall)

UFW is a interface to modify the firewall of the device without compromising security. You use it to configure which ports to allow connections to and which ports to close. This is useful in conjunction with SSH, can set a specific port for it to work with.

What is SSH?

SSH or Secure Shell is an authentication mechanism between a client and a host. It uses encryption techniques so that all communication between clients and hosts is done in encrypted form. User on Mac or Linux can use SSH the terminal to work on their server via SSH.

What is Cron?

Cron or cron job is a command line utility to schedule commands or scripts to happen at specific intervals or a specific time each day. Useful if you want to set your server to restart at a specific time each day.

- **what APPArmor is?**

  Linux security system that provides Mandatory Access Control (MAC) security. Allows the system admin to restrict the actions that processes can perform. It is included by default with Debian. Run aa-status to check if it is running.

- **important Cmmand Lines**
  - cd /usr/local/bin : to show monitoring.sh
  - sudo visodu: sudores file we use it to add user to sudo group and run the minotoring script and addd some password rolus
  - sudo nano /etc/ssh/sshd_config to add port 4242 and make the root can't connect using ssh
  - cd /var/log/sudo : sudo logs file
  - sudo nano /etc/login.defs: add some password policy
  - sudo nano /etc/security/pwquality.conf or sudo nano /etc/pam.d/common-password to add more password policy
  - sudo crontab -u root -e to edit the cron job
  - change script to */1 * * * * sleep 30s && script path : to run it every 30 seconds, delete the line to stop the job from running.
- Evaluation Commands for UFW, Group, Host, lsblk and SSH
  - sudo ufw status
  - sudo systemctl status ssh
  - getent group sudo
  - getent group user42
  - sudo adduser <new_username>
  - sudo groupadd <group_name>
  - sudo usermod -aG <groupname> <username>
  - sudo chage -l username check password expire rules
  - hostname
  - hostnamectl set-hostname <new_hostname> to change the current hostname
  - sudo reboot: Restart your Virtual Machine.
  - sudo nano /etc/hosts : change current hostname to new hostname
  - lsblk: to display the partitions
  - dpkg -l | grep sudo :to show that sudo is installed
  - sudo ufw status numbered : list roles
  - sudo ufw allow <port-id> allow port
  - sudo ufw delete <rule number>
  - ssh your_user_id@127.0.0.1 -p 4242
- **user managment:**
  - useradd <username> : creates a new user.
    usermod <username> : changes the user's parameters: -l for the username, -c for the full name, -g for groups by group ID.
    userdel -r <username> : deletes a user and all associated files.
    id -u <username> : displays user ID.
    users : shows a list of all currently logged in users.
    cat /etc/passwd | cut -d ":" -f 1 : displays a list of all users on the machine.
    cat /etc/passwd | awk -F ":" '{print $1}' : same as above.
- **Group Management:**
  groupadd : creates a new group.
  gpasswd -a : adds a user to a group.
  gpasswd -d : removes a user from a group.
  groupdel : deletes a group.
  groups : displays the groups of a user.
  id -g : shows a user's main group ID.
  getent group : displays a list of all users in a group.

Created by : Said Zemmouri