**Born2beroot**

**42 Common Core | Milestone 1**

**beatde-a@student.42lisboa.com**

# Virtual Machine (VM)

**Key concepts:**

* **Virtual machine** (also **guest machine):** software-based emulation of a physical computer hosted on a physical device (**host machine)**, with its own operating system (OS) and virtual hardware (CPU, RAM, storage device, network card, etc.).
* **Hypervisor:** software programme that creates and runs VMs. It controls the resources of the host machine, allocating the necessary resources to the VM and isolating it from the system hardware and other VMs.
* **VirtualBox (VB):** free and open-source virtualization software that serves a hypervisor. Type 1 and 2
* **Virtual Disk Image (.vdi)**: file used by VB to store the VM’s entire hard drive contents, including the operating system, applications, files, and setting. It functions as a virtual hard drive.
* **Disk’s signature:** 40-digit hexadecimal number extracted from .vdi using the Secure Hash Algorithm 1 (**sha1sum**) and stored in **signature.txt**.

**Why are VMs useful?**

* More convenient and cheaper than physical computers – inexpensive, less physical space required, less maintenance, simplified data backup.
* Software testing and development in multiple OS using only one physical computer.
* Safety and isolation for testing unstable or malicious programmes (cybersecurity).
* Flexibility for learning and legacy support.
* Simulation of real network environments.

**Useful commands:**

* Extract .vdi’s signature: **sha1sum <VM>.vdi > signature.txt**
* Check signature in the folder of .vdi: **sha1sum -c path/to/signature.txt**

# Debian

**Key concepts:**

* **Operating system (OS):** the core software that manages a computer's hardware and software resources and provides a platform for running applications. It functions as a middle layer between the user, the applications and the hardware.
* **Debian:** free and open-source Linux distribution, i.e. complete OS centred on Linux kernel with additional tools, programmes, package managers, etc. One of the oldest and most influential Linux distributions, known for its stability, reliability and vast software repository.
* **GRUB (Grand Unified Bootloader):** programme that manages the boot process on many Linux systems, including Debian.)

**Why Debian?**

* Beginner-friendly (recommended by 42): easier to install and customize, lightweight. Ideal for general users, learners, developers, and community-driven projects, whereas Rocky is more suited to enterprises.
* Completely free and open-source: 100% community project, with a vast and diverse user community, extensive documentation, and well-understood practices.
* Extremely stable.
* Secure by design: with frequent security patches and updates.
* Versatile: suitable for general-purpose use (for example, as a desktop, server, or embedded system).
* Vast software repository, including APT package manager, which is renowned for its simplicity, robustness, and ease of dependency management.

**Useful commands:**

* Check OS: **hostnamectl** OR **cat /etc/os-release**

# Disk Partitioning & LVM

**Key concepts:**

* **Storage device** (aka **disk**, **hard drive**)**:** hardware component in the computer where all data is permanently saved, even when the power is off. Examples: HDD (Hard Disk Drive) and SSD (Solid State Drive).
* **Disk partitioning:** process of dividing a hard drive (HDD or SSD) into separate, independent sections called **partitions**. Each partition acts like a separate disk, with its own file system and structure. This is useful to organize or isolate data on the same hardware. Yet, unlike VMs, partitions do not emulate hardware or provide OS isolation.
* **Master Boot Record (MBR) partitioning scheme**: older method for organizing and managing partitions on a hard drive. This is the default in VB. Replaced by the modern system GPT (GUID Partition Table).
* **Primary partition**: standard, standalone partition. One of them can be marked as “active” to boot an OS. You can have up to 4 primary partitions on a single MBR-style disk.
* **Extended partition**:special type of primary partition that acts as a container for multiple logical partitions. Only one extended partition is allowed.
* **Logical partitions**: used to bypass the 4-partition limit of MBR. You can have many logical partitions (=>128). They are part of the extended partition and not directly bootable.
* **Encrypted partition:** section of a storage device where all data is automatically stored in encrypted form and decrypted if you provide the correct password/key. It is useful to protect sensitive data from unauthorized access.
* **Logical Volume Manager (LVM):** flexible way to manage disk storage on Linux systems, allowing you to create, resize, and organize storage volumes (partitions) easily and dynamically. In particular, it can combine multiple physical disks or partitions into one large storage pool, create logical volumes, and resize volumes dynamically without repartitioning.
* **Physical Volume**: actual physical disk or partition added to LVM.
* **Volume Group**: pools of storage made by combining one or more physical volume.
* **Logical Volume**: virtual partition carved out by LVM. Unlike a partition, it is not created directly on a disk, is not restricted to one disk, and has no fixed size.
* **Mount Point**: directory in Linux file system where a storage device (partition, logical volume) is attached so you can access the data inside it.
* **Swap area**: portion of disk storage that acts as virtual memory when the system’s RAM is full – basically, extra backup memory beyond the physical RAM.

**Useful commands:**

* List information about block devices (i.e. devices used to store data in blocks, like hard drives, SSD, USB drives, partitions, LVM volumes, etc.): **lsblk**

**lsblk:**

* NAME: device name – sda for disk, sda1-4 for primary/extended partitions, sda5-… for logical partitions.
* MAJ:MIN: major and minor device numbers, with which Linux identifies and manages devices internally. Major – type of device. Minor – individual device.
* RM: removable device (1 yes, 0 no).
* SIZE: size of device or partition.
* RO: read-only device (1 yes, 0 no).
* TYPE: disk, part, lvm, rom, etc.
* MOUNTPOINT: where the device is mounted in the filesystem.

# Package management & APT

**Key concepts:**

* Both apt and aptitude are **package management tools** used on Debian-based systems to install, upgrade, search for and remove software packages.
* **Software package:** bundle of files or metadate that together provide a piece of software ready to be installed in a computer. It can include program files, libraries, configuration files, documentation, and metadata.
* **Dependencies**: other software packages or libraries that a programme needs in order to work properly. Without these, the programme may fail to run, crash, or behave incorrectly.
* **apt (Advance Package Tool):** default CLI tool used on Debian-base Linux systems for managing software packages. It is simple and user-friendly, automatically resolving and installing dependencies need by the packages. It uses **dpkg** under the hood, but adds dependency resolution and works with remote repositories.
* **aptitude:** non-default tool, which offers both CLI and TUI (text user interface – with interactive menus and layout). It offers a smarter and more flexible dependency resolution, suggesting multiple ways to resolve conflicts.
* **dpkg (Debian Package):** low-level package manager for Debian-based systems. It operates only with individual files at a local level, without handling dependencies automatically.

**Useful commands:**

* Download the latest list of available packages from the repositories: **sudo apt update**
* Upgrade all installed packages to their latest available versions: **sudo apt upgrade**
* Install a package: **sudo apt install <package\_name>**
* Remove a package (but keep its configuration files): **sudo apt remove <package\_name>**
* Show package information: **apt show <package\_name>**
* List all installed packages along with their versions: **apt list --installed** OR **dpkg –list**
  + to search for a specific package, add: **| grep <package\_name>** OR **| less** ( **/<keyword>** to search list, **q** to quit)

# Sudo

**Key concepts:**

* **superuser (su):** special user account on Unix/Linux systems, usually called **root,** that has full administrative privileges, like install or remove software, modify system files, change permissions, manage other users, shut down or reboot the machine, etc.
* **superuser do (sudo):** command that lets regular users temporarily run commands with superuser (root) privileges.
* **sudoers:** configuration file that controls who can use sudo and which commands they can run.
* **visudo:** safe command to edit sudoers, checking the syntax before saving and preventing mistakes that could lock you out of sudo access.
* **logfile:** specifies the path to the file that records which sudo commands were run.
* **iolog\_dir:** specifies the path to subdirectory that records the input and output of sudo commands.
* **requiretty:** forces sudo commands to be run only from a real terminal (**TTY – teletypewriter)**, ie interactively from a terminal but not from a non-interactive shell, script or automated process (like a cron job).
* **secure\_path:** defines which directories are searched for executables when using sudo, to prevent inhering the user’s environmental path and running programmes or scripts from unexpected or unsafe directories. Sudo can still run an executable from outside the path, but must specify its full or relative path.

**Why use sudo instead of logging in as root?**

* Safer: you don’t need to share or use root password directly.
* Auditable: all sudo commands can be logged.
* Limited access: you can restrict what commands regular users can run.
* Prevents accidental system-wide damage from casual mistakes.

**Useful commands:**

* Switch to root user: **(sudo)** **su**
* Give sudo privileges to a user: **adduser <username> sudo**
* Check sudo privileges: **sudo -l**
* Run a command as root: **sudo <command>**
* Edit the sudoers file safely: **sudo visudo**
* Check the log of sudo commands: **sudo cat /var/log/sudo/sudo.log**

# Users & groups

**Key concepts:**

* **Users:** people or system services who can log in or run processes. Each user has a username, UID, home directory, shell, and password.
* **Groups:** collections of users, used to manage permissions more easily. Each group has a group name, GID, and list of members.
* **/etc/passwd:** stores user account info (retrieved through **getent**).
* **Hostname:** name of computer (or host) on a network.
  + **static –** main, persistent (stored in /etc/hostname and /etc/hosts);
  + **transient –** temporary (set by DHCP or manually with hostname);
  + **pretty –** human-readable version, which can contain spaces.
* **hostnamectl**: used to query and change the system’s (static) hostname and related settings using system.

**Useful commands:**

* List all user accounts: **getent passwd**
  + to search for a specific user, add: **<username>**
* List all groups: **getent group**
  + to search for a specific group, add: **<groupname>**
* List all the groups a user belongs to: **groups <username>** OR **id <username>**
* Add user: **sudo adduser <username>**
* Delete user: **sudo deluser <username>**
* Add group: **sudo addgroup <groupname>**
* Delete group: **sudo delgroup <groupname>**
* Add user to group: **sudo adduser <username> <groupname>** OR **sudo usermod -aG <groupname> <username>**
* Remove user from group: **sudo nano /etc/group**
* Check password settings: **sudo chage -l <username>**
* Check hostname: **hostname** OR **hostnamectl** (more info)
* Change hostname:
  + **hostnamectl set-hostname <new\_hostname>** OR **sudo nano /etc/hostname**
  + **sudo nano /etc/hosts**

# AppArmor

**Key concepts:**

* **AppArmor (Application Armor):** Linux security module that enforces mandatory access control (MAC) on programmes – it limits what programmes can do (for example, which files, networks, or capabilities they can access) with security profiles. It helps to contain damage if a programme is compromised or buggy.
* **Mandatory Access Control (MAC):** security module in which the operating system strictly controls access to resources, and users or programmes cannot override or change those rules, even if they are administrators or root.

**Useful commands:**

* Check status of AppArmor: **systemctl status apparmor**
* Show AppArmor status and loaded profiles: **sudo aa-status**

# UFW (Uncomplicated Firewall)

**Key concepts:**

* **Firewall:** security system (software or hardware) that controls incoming and outgoing network traffic based on a set of rules. It protects the system or network from unauthorized access and malicious traffic.
* **UFW (Uncomplicated Firewall):** user-friendly interface or frontend for managing the built-in firewall on Linux (iptables or nftables). It is not a firewall itself, but a tool that simplifies the use of the system’s firewall.

**Useful commands:**

* Install UFW: **sudo apt install ufw**
* Show current UFW rules: **sudo ufw status**
* Show current UFW rules in an ordered list: **sudo ufw status numbered**
* Enable UFW: **sudo ufw enable**
* Allow ports: **sudo ufw allow <port>**
* Deny ports: **sudo ufw deny <port>**
* Delete a rule: **sudo ufw delete <rule\_number>**

# Secure Shell (SSH)

**Key concepts:**

* **Secure Shell (SSH):** network protocol that allows a secure connection from one computer (client) to another (usually a server) over a network. SSH is secure because it encrypts everything between computers (such as login credentials and all commands and data transferred).
* **OpenSSH:** most widely used open-source implementation of the SSH protocol.
* **sshd\_config** and **ssh\_config:** configuration files for OpenSSH: sshd\_config controls how the SSH server behaves, i.e., how it accepts and handles incoming connections, while ssh\_config controls how the local SSH client behaves when connecting to a remote server.
* **Network Address Translation (NAT):** default network mode in VB that allows the VM to access the internet but keeps it hidden from the rest of the network. It doesn’t allow any direct connections into the VM from the host, unless through port forwarding.
* **Port:** access point through which data enters or leaves a computer over a network.
* **Port forwarding**: gives the host access to services inside the VM by forwarding a port on the host machine to a port on the VM.

**Useful commands:**

* Install OpenSSH: **sudo apt install openssh-server**
* Check status of ssh: **sudo systemctl status ssh**
* Edit the sshd server configuration file: **sudo nano /etc/ssh/sshd\_config**
* Edit ssh client file: **sudo nano /etc/ssh/ssh\_config**
* Restart ssh service: **sudo systemctl restart ssh**
* Connect via ssh: **ssh <guest\_username>@localhost -p 4242**

# Password Policy

**Key concepts:**

* **libpam-pwquality:** PAM (Pluggable Authentication Module) that enforces strong password policies on Linux systems. It is called from the PAM config file /etc/pam.d/common-password and reads its rules from /etc/security/pwquality.conf**.**

**What are the advantages and disadvantages of a strong password policy?**

* **Advantages**:
  + reduces risk of credential theft;
  + protects sensitive systems and data;
  + ensures compliance with standards;
  + encourages good user habits.
* **Disadvantages**:
  + causes frustration;
  + makes it easier to forget passwords;
  + promotes workarounds and bad habits like writing passwords downs, reusing passwords with minor changes and using predictable patterns;
  + promotes a false sense of security.

**Useful commands:**

* Edit password settings: **sudo nano /etc/login.defs**
* Change max days until password expiration: **chage -M 30 <username>**
* Change min days before modifying a password: **chage -m 2 <username>**
* Change number of days before a password expires for warning: **chage -W 7 <username>**
* Check password settings (expiry): **chage -l <username>**
* Change password complexity rules: **sudo nano /etc/security/pwquality.conf**
* Change password: **passwd <username>**

# Cron

**Key concepts:**

* **Cron:** time-based job scheduler in Unix-like OS, used to automate tasks (**cron jobs**) that run periodically at fixed times, dates, or intervals – like backups, updates, or system maintenance scripts.
* Cron runs a daemon (**crond**) in the background, which checks cron configuration files (**crontabs**) every minutes. If a job matches the current time, it runs the command.
* **Daemon:** background process that runs continuously on a Unix-like system to perform specific tasks or provide services.

**Useful commands:**

* Edit the crontab for the root user: **sudo crontab -u root -e**

**Crontab syntax:**

\*Minute \*hour \*day of the month \*month \*day of the week

\*/<interval>

# Monitoring script

**System architecture:**

* **uname (Unix name) - a:** displays all available system information about the machine and kernel in one line.

**Number of physical and virtual processors:**

* **CPU (Central Processing Unit**, aka **processor):** the brain of the computer – it performs all the instructions and calculations that make programmes and the OS run.
  + **physical:** real hardware chip installed on the motherboard;
  + **virtual:** software abstraction of a CPU, found in VMs, which can share physical hardware.
* **grep:** searches text pattern insides files or output.
* **/proc/cpuinfo:** stores detailed info about each CPU.
* **sort:** sorts the matched line.
* **uniq:** filters duplicate adjacent lines.

**Current available RAM on your server and its utilization rate as a percentage:**

* **RAM (Random Access Memory):** computer’s short-term memory – it temporarily stores data and instructions that the CPU needs right now or in a very near future. Unlike a hard drive (which stores data long-term), RAM loses all data when the computer is turned off.
* **free --mega:** displays system memory usage in megabytes.
* **awk:** text-processing language used to extract, transform, and analyse text, especially in columns.

**Current available storage on your server and its utilization rate as a percentage:**

* **df (disk free):** displays info about disk space usage on mounted filesystems.
* **-h (human-readable):** sizes are displayed in readable format like MB and GB instead of raw bytes.
* **--total:** adds a summary line at the end that combines all filesystems.

**Current utilization rate of your processors as a percentage:**

* **top:** shows real-time dynamic view of running processes and system resource usage (it updates every few seconds).
* **-b:** runs top in batch mode, outputting data once and then exits.
* **-n1:** specifies the number of iterations top should run.

**Date and time of the last reboot:**

* **who -b:** displays the last system boot time.

**Whether LVM is active or not:**

* **grep -q:** runs grep in quiet or silent mode, without producing output. Exit status 0 if match is found (true) and 1 if not found (false).

**Number of active connections:**

* **Internet socket:** software endpoint used for sending and receiving data across a network – virtual plug that connects a computer to another.
* **/proc/net/sockstat:** stores statistics about network sockets.

**Number of users using the server:**

* **who:** shows info about users currently logged into the system (who is logged in, when, and from where).

**IPv4 address of your server and its MAC (Media Access Control) address:**

* **IP address:** logical address used to identify a device on a network – can change depending on network configuration. **IPv4**: most common format, 32-bit, 4 numbers.
* **MAC address:** physical address burned into the computer’s network card (ethernet or wi-fi) – permanent and unique to hardware.
* **hostname -I:** displays the IP addresses assigned to the current host (computer).
* **ip link:** displays information about network interfaces.

**Number of commands executed with the sudo program:**

* **/var/log/sudo/sudo.log:** stores the log of all commands used with sudo (as defined in sudoers).

**wall (write all):** sends a message to all logged-in users’ terminals on a Linux system.

# General

**Key concepts:**

* **systemd**: initialization system in most modern Linux distributions (including Debian 12) used to manage system services (daemons), resources and the boot process.
* **systemctl:** CLI used to interact with systemd.

**Useful commands:**

* Start a service: **sudo systemctl start <service\_name>**
* Stop a service: **sudo systemctl stop <service\_name>**
* Restart a service: **sudo systemctl restart <service\_name>**
* Check service status: **sudo systemctl status <service\_name>**
* Enable service at boot: **sudo systemctl enable <service\_name>**
* Disable service at boot: **sudo systemctl disable <service\_name>**
* Show all services: **systemctl list-units --type=service**