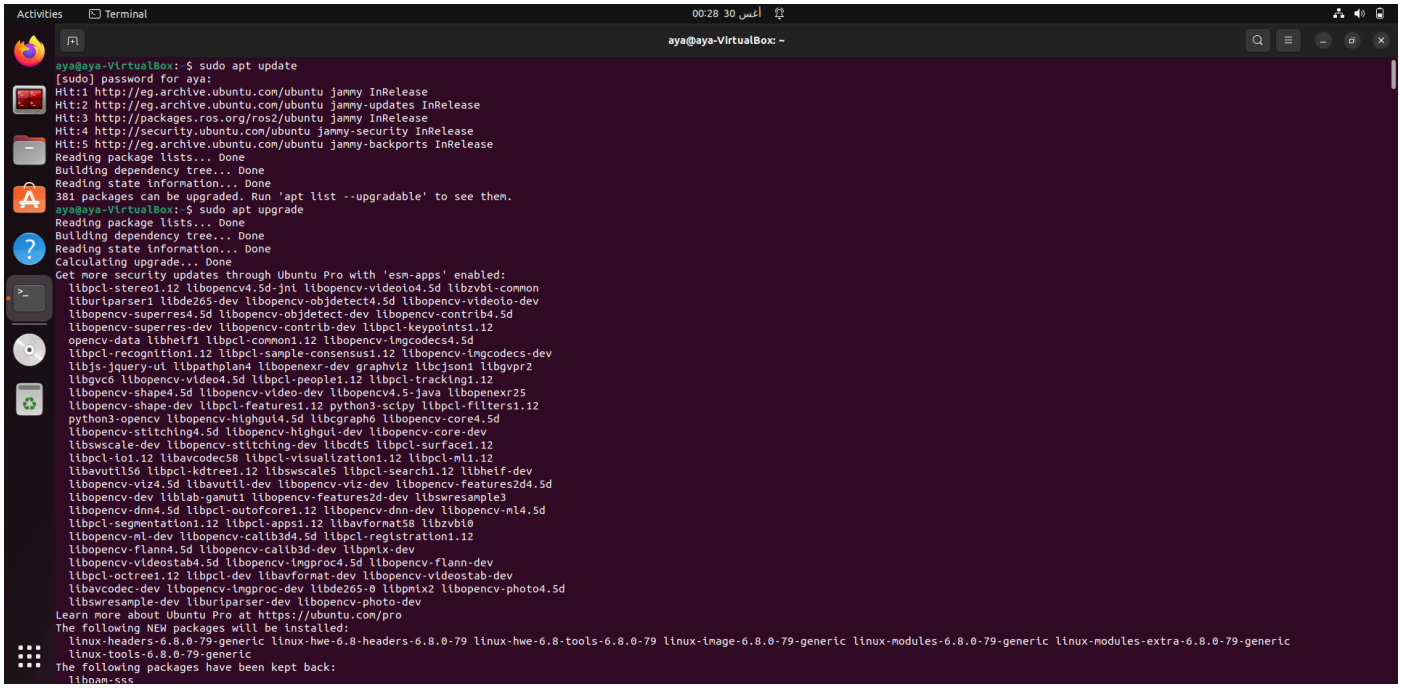
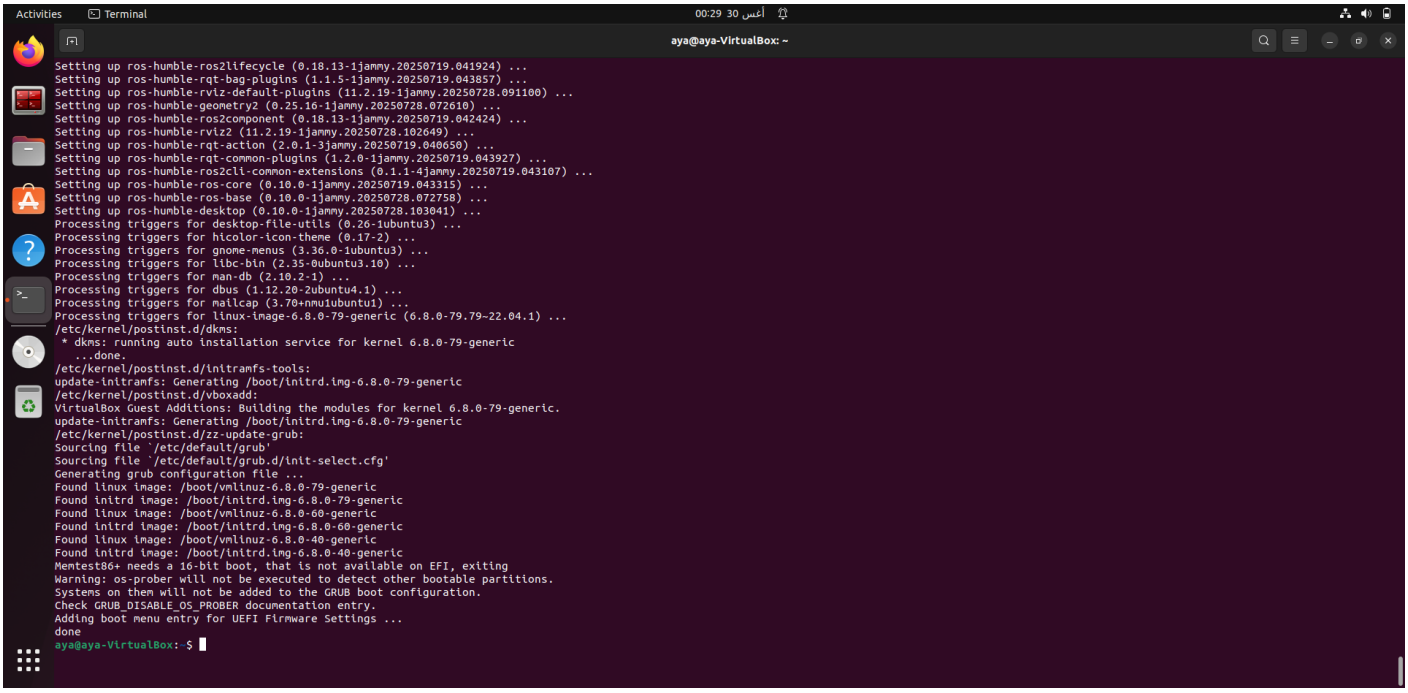
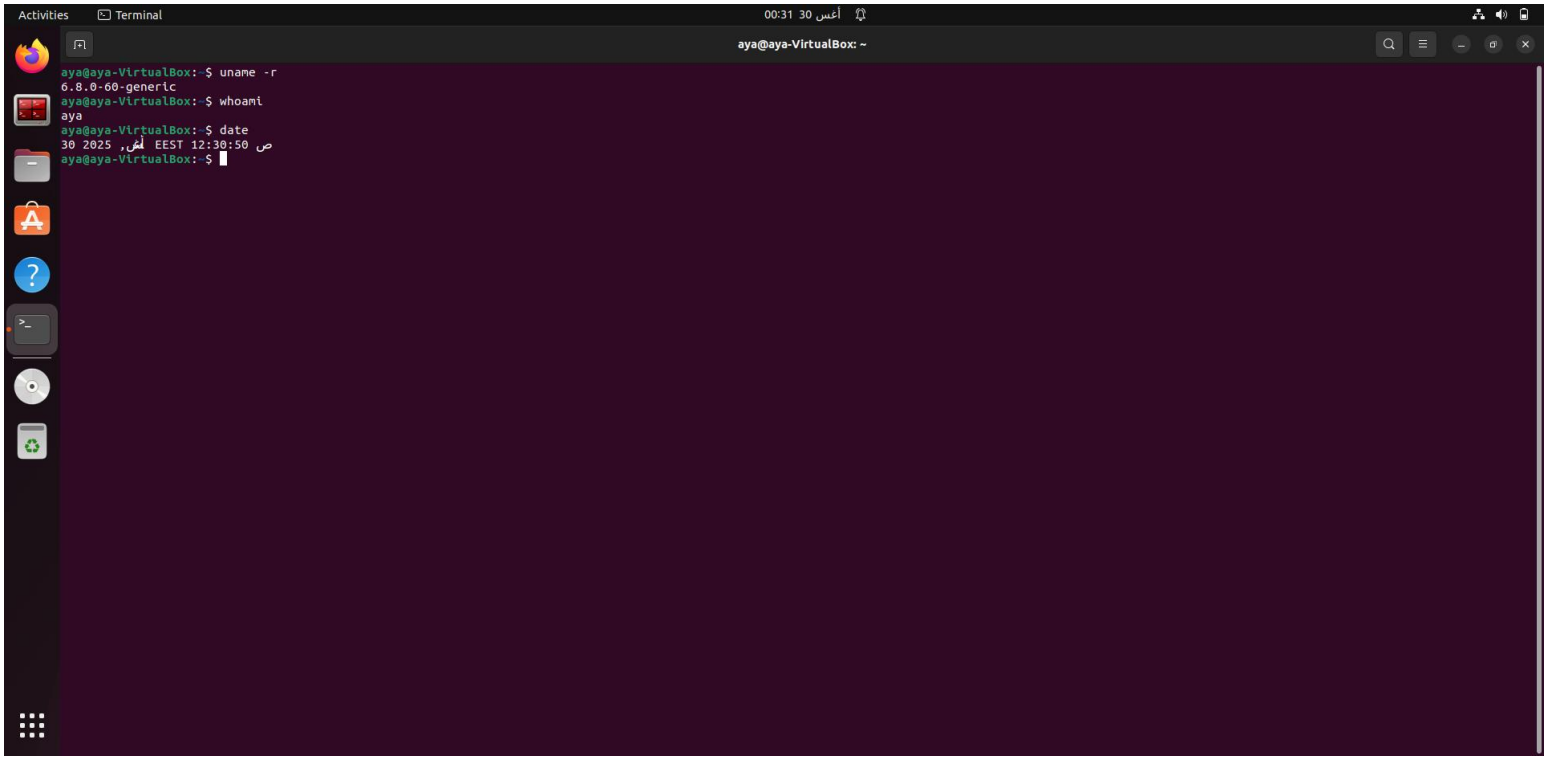
Command Part

Refresh package lists and upgrade the system.:

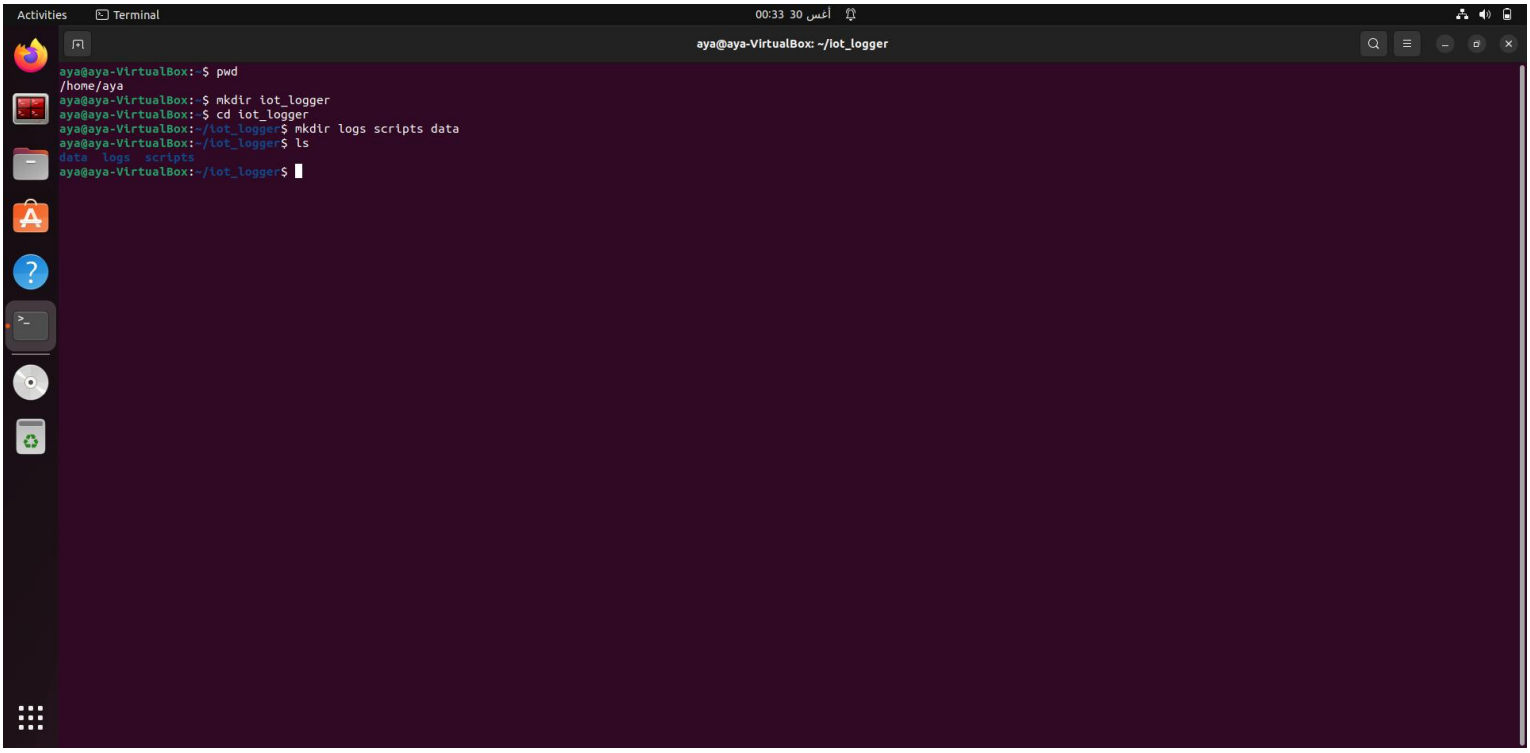




Verify system details: kernel version, user, time:



Create /home//iot_logger with subdirectories: logs, scripts, data:



Open-Ended Questions

Draw or describe the Linux architecture layers (hardware → kernel → shell → user space). Where do system calls fit?

- Linux is built in layers. At the bottom is the hardware (CPU, memory, disk, devices), which provides raw resources. Above it sits the kernel, the core of Linux, which manages those resources and makes sure programs can share them safely. Programs in user space (like the shell, browsers, or editors) cannot touch hardware directly, so they use system calls to ask the kernel for services, such as reading files, creating processes, or sending data over the network. The shell is just one user program that lets us type commands, which are then executed through these system calls.

Explain the purpose of these directories: /, /bin, /sbin, /usr, /etc, /var.

- In Linux, the root directory / is the starting point of the entire file system, and every file or directory branches from it, like the C:\ in Windows. The /bin directory contains essential user commands such as ls and cp, while /sbin holds important system administration commands like shutdown and fdisk that are mostly used by the root user. The /usr directory stores user system resources, including installed software, libraries, and documentation, with subfolders like /usr/bin for applications. System-wide configuration files are located in /etc, for example /etc/passwd for user accounts. The /var contains variable data that changes during system operation, such as logs in /var/log, mail, and job spools.

Why does Linux treat everything as a file? Explain the difference between a program and a process

- Linux treats everything as a file to provide a simple and consistent way of accessing resources. Whether it is a text document, a device like a hard disk, or even information about running processes, all are represented as files that can be opened, read, or written using the same system calls. This design keeps the system flexible and easy to manage. A **program**, on the other hand, is a file containing instructions stored on disk, while a **process** is a running instance of that program in memory with its own resources and process ID.