Automated SSD/HDD Monitoring and Backup System with Python

Prepared By:
Mahmoud Alsalloum

Table of Contents

Contents

Technology Stack

Ubuntu-Client - Used as the testing platform to ensure the script functions correctly without risking production systems.

Python - The primary programming language used to develop the monitoring and backup script

SMART Monitoring Tools (smartmontools) – Used to retrieve SSD/HDD health data.

Rsync – Utilized for performing efficient differential backups.

Systemd – Configured to manage the script as a background service and ensure it starts at boot

Ventoy HDD – Used as the backup target, with backups stored in a dedicated folder to avoid interfering with existing ISO files.

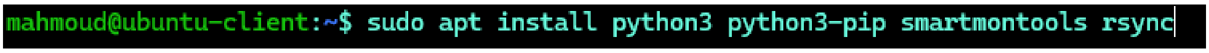
Project Overview

Welcome to my project!

Thank you for exploring my portfolio and taking the time to review my work. Disk health monitoring and data backup are critical components of system reliability and data protection. Failures in storage devices can lead to data loss, downtime, and significant recovery efforts. In this project, I developed a Python script that continuously monitors the health of SSDs using SMART data. The script runs in the background, checking the status of the drives every hour. If it detects potential issues, such as bad sectors or signs of failures, it sends a notification and triggers an automated backup process. The backup system is designed to perform an initial full backup and subsequent differential backups to save time and storage space. The backup is stored on a Ventoy HDD in a dedicated folder, ensuring physical separation from the main system and avoiding interference with existing ISO files. The script is configured to start automatically at system boot, ensuring uninterrupted monitoring and protection.

Implementation Steps

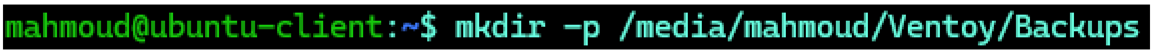
I began by installing the necessary tools and dependencies on my Ubuntu-Client. This included Python, smartmontools for SMART data retrieval, and rsync for backups. See Figure 1.



```
mahmoud@ubuntu-client:~$ sudo apt install python3 python3-pip smartmontools rsync
```

Figure 1: The installation of tools and dependencies

To use the Ventoy HDD as the backup target, I created a dedicated folder named Backups inside existing Ventoy directory. This ensures that backups do not interfere with the ISO files stored on the HDD. See Figure 2.



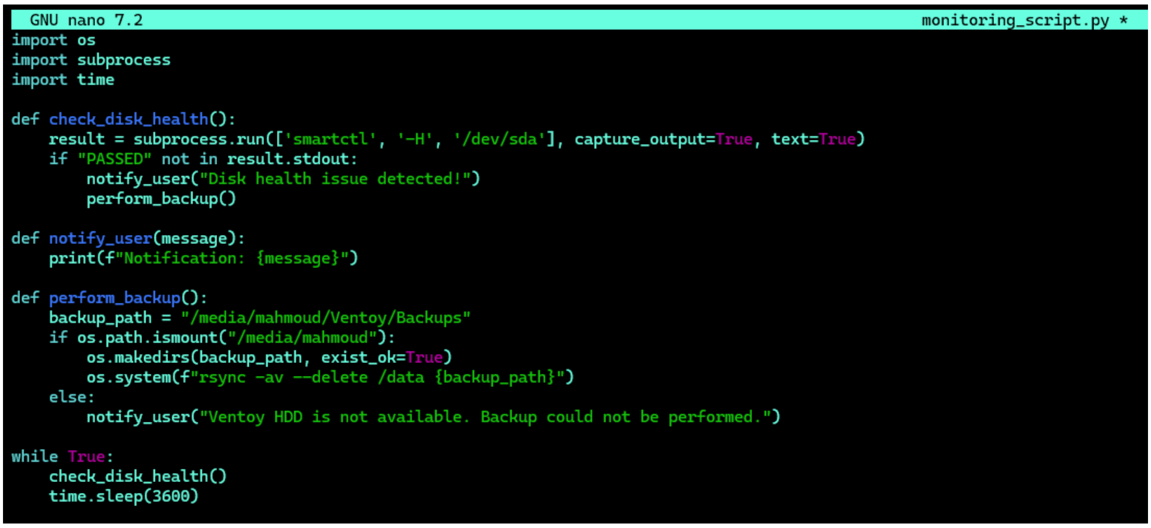
```
mahmoud@ubuntu-client:~$ mkdir -p /media/mahmoud/Ventoy/Backups
```

Figure 2: Create Backup-Folder in HDD

The HDD is automatically mounted under /media/mahmoud when inserted. The script checks for this mount point before performing backups. I developed a Python script that performs the following tasks:

- Retrieves SMART data for the SSD/HDD using smartctl.
- Analyzes the data for potential issues, such as bad sectors or signs of imminent failure.
- Sends a notification if a problem is detected.
- Triggers a backup process using rsync.

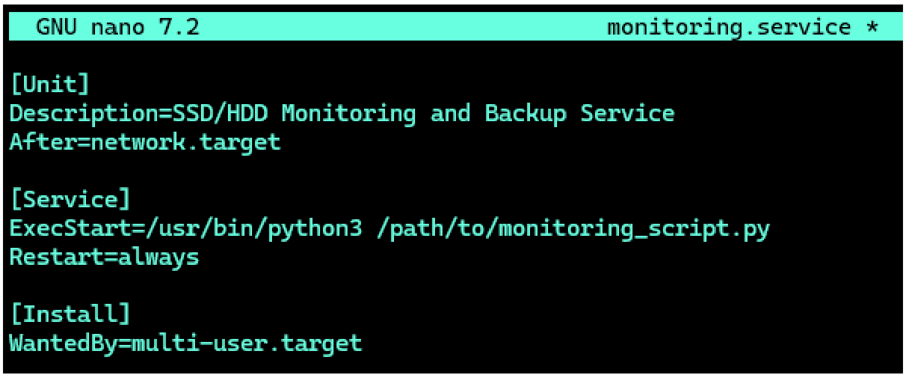
The script is designed to run in the background and checks the disk status every hour. See Figure 3 for a snippet of the script.



```
  GNU nano 7.2                                                            monitoring_script.py *
import os
import subprocess
import time

def check_disk_health():
    result = subprocess.run(['smartctl', '-H', '/dev/sda'], capture_output=True, text=True)
    if "PASSED" not in result.stdout:
        notify_user("Disk health issue detected!")
        perform_backup()

def notify_user(message):
    print(f"Notification: {message}")

def perform_backup():
    backup_path = "/media/mahmoud/Ventoy/Backups"
    if os.path.ismount("/media/mahmoud"):
        os.makedirs(backup_path, exist_ok=True)
        os.system(f"rsync -av --delete /data {backup_path}")
    else:
        notify_user("Ventoy HDD is not available. Backup could not be performed.")

while True:
    check_disk_health()
    time.sleep(3600)
```

Figure 3: Python script

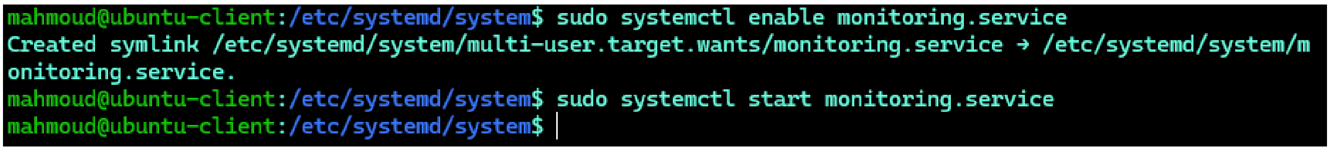
To ensure efficient backups, I configured rsync to perform an initial full backup and subsequent differential backups. The backup directory is stored in the Ventoy HDD under /media/mahmoud/Ventoy/Backups. To ensure the script starts automatically at system boot, I created a systemd service file. See Figure 4.



```
  GNU nano 7.2                                   monitoring.service *

[Unit]
Description=SSD/HDD Monitoring and Backup Service
After=network.target

[Service]
ExecStart=/usr/bin/python3 /path/to/monitoring_script.py
Restart=always

[Install]
WantedBy=multi-user.target
```

Figure 4: Service file for python script

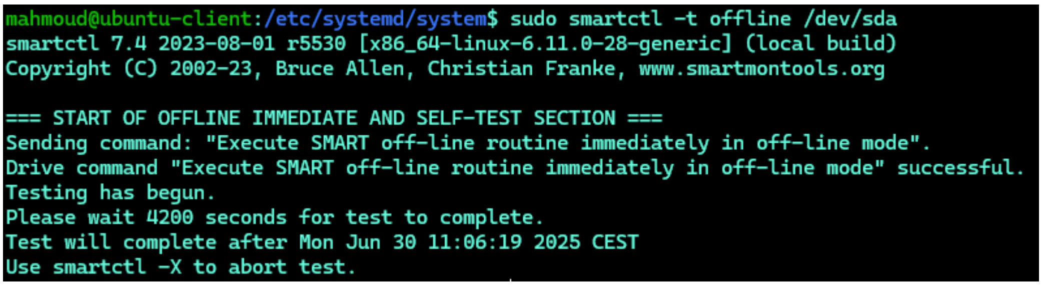
I enabled and started the service using the following command: See Figure 5.



```
mahmoud@ubuntu-client:/etc/systemd/system$ sudo systemctl enable monitoring.service
Created symlink /etc/systemd/system/multi-user.target.wants/monitoring.service → /etc/systemd/system/m
onitoring.service.
mahmoud@ubuntu-client:/etc/systemd/system$ sudo systemctl start monitoring.service
mahmoud@ubuntu-client:/etc/systemd/system$ 
```

Figure 5: Enable and start service

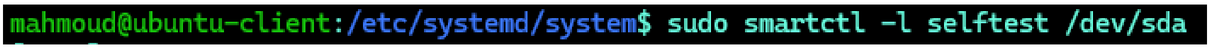
After setting up the script and service, I tested the system by simulating disk issues and verifying that the script correctly detected the problem, sent a notification, and performed a backup. See Figure 6.



```
mahmoud@ubuntu-client:/etc/systemd/system$ sudo smartctl -t offline /dev/sda
smartctl 7.4 2023-08-01 r5530 [x86_64-linux-6.11.0-28-generic] (local build)
Copyright (C) 2002-23, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF OFFLINE IMMEDIATE AND SELF-TEST SECTION ===
Sending command: "Execute SMART off-line routine immediately in off-line mode".
Drive command "Execute SMART off-line routine immediately in off-line mode" successful.
Testing has begun.
Please wait 4200 seconds for test to complete.
Test will complete after Mon Jun 30 11:06:19 2025 CEST
Use smartctl -X to abort test.
```

Figure 6: System test

The SMART offline test takes approximately 70 minutes (4200 seconds) to complete. After the test finishes, the results can be reviewed using the following command: See Figure 7.



```
mahmoud@ubuntu-client:/etc/systemd/system$ sudo smartctl -l selftest /dev/sda
```

Figure 7: Selftest for system