A step-by-step guide on how to use the Raspberry Pi-based private cloud storage system, detailing how to upload, manage, and encrypt files

Raspberry Pi-Based Private Cloud Storage with Client-Side Encryption

Introduction

As data becomes central to personal and professional activities, privacy and security concerns arise when using third-party cloud providers like Google Cloud and OneDrive. This project leverages Raspberry Pi to create a private cloud storage system with client-side encryption, giving users full control and protection of their data. This solution is affordable and adaptable, providing small and medium enterprises with a secure and independent storage option without relying on third-party services.

Problem Statement

Relying on third-party cloud providers like Google Cloud and OneDrive raises concerns about data breaches, privacy, and lack of user control. With the increasing risk of unauthorized access and limited transparency in data management, there is a growing need for secure, user-controlled cloud storage. This project addresses these issues by developing a Raspberry Pi-based private cloud system with client-side encryption, ensuring full data protection and privacy.

Objective

1. To conduct a detailed study on Raspberry Pi's capabilities and its application in private cloud storage, enhancing the understanding and optimization of this technology for secure data storage solutions.
2. To create a fully operational private cloud storage system utilizing Raspberry Pi, providing a secure and personalised data storage solution.
3. To enable secure data transmission using HTTPS and create an automated CLI script for client-side encryption, streamlining the process of protecting files before cloud upload.
4. To create a detailed setup and usage documentation, along with the encryption script, on GitHub to help users establish their own private cloud storage systems.

Acknowledgement

I would like to express my deepest gratitude to Dr. Julia Juremi, my supervisor, for her invaluable guidance, feedback, and continuous support throughout this project. My thanks also go to Ts. Umapathy Eaganathan for his guidance on structuring the Integrated Report (IR). Special appreciation goes to the interviewees and survey participants for their insights and contributions. Finally, I extend heartfelt thanks to my family and friends, whose encouragement and support were instrumental in the success of this project.

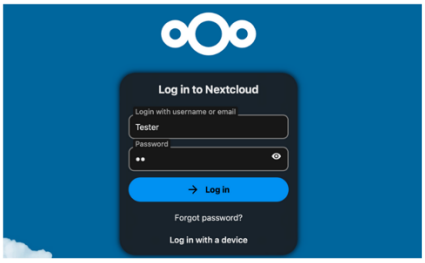
Conclusion

The "Raspberry Pi-Based Private Cloud Storage with Client-Side Encryption" project successfully provides a secure, self-managed storage solution. It ensures data privacy and protection, making it an affordable option for small and medium enterprises. While the project met its objectives, future improvements such as a user-friendly interface and enhanced security features can further improve its usability and scalability.

Authors and Supervisors

AUTHORS AND AFFILIATIONS
- Nur Aiman Iskandar Bin Nur Iskandar
- TP060914
- B.Sc. (Hons) Computer Science Specialism in Data Analytics
- Asia Pacific University of Technology and Innovation

SUPERVISORS
- Supervised by: Dr. Julia Juremi
- Second Marker: Ts. Umapathy Eaganathan



Author : Nur Aiman Iskandar Bin Nur Iskandar

Email : aimaniskandar2018@gmail.com

Table of Contents

Step 1: Accessing the Cloud Platform

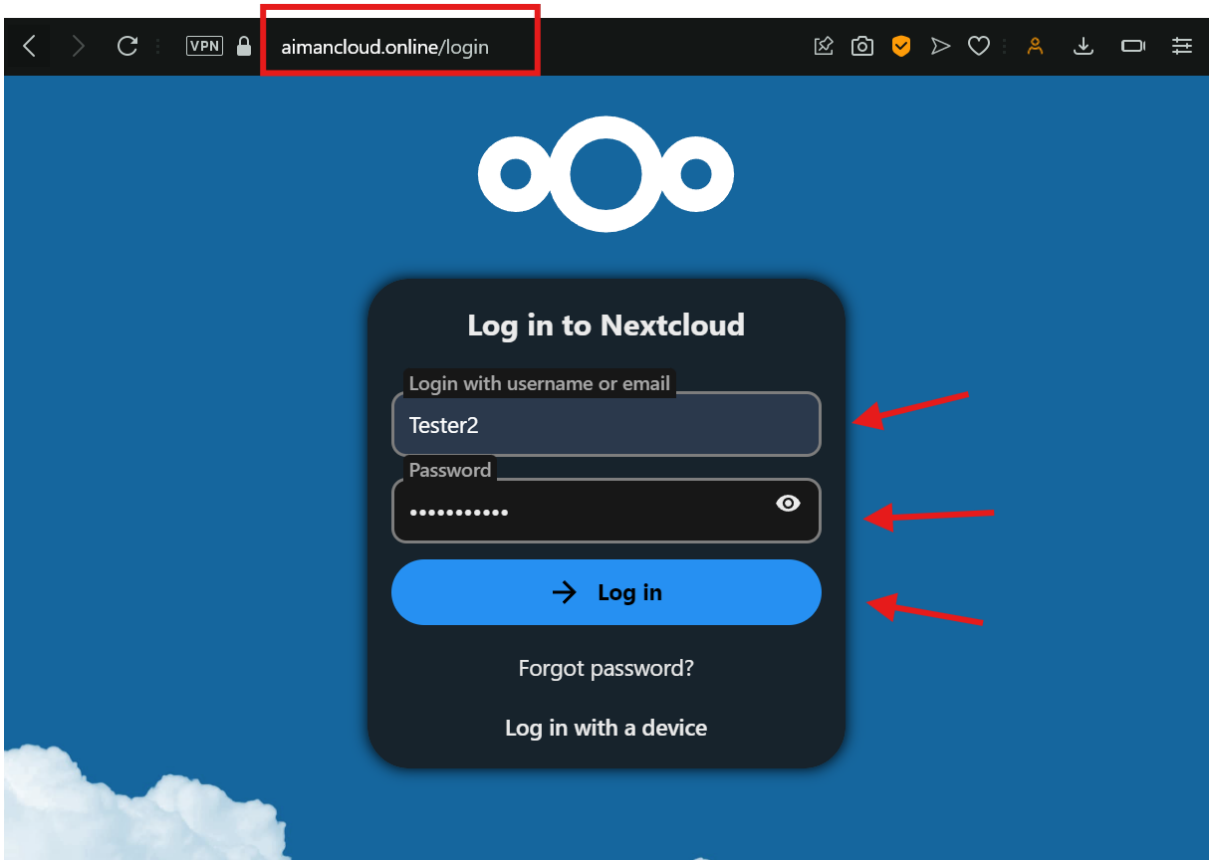


Figure 1: Accessing the Cloud Platform

Users start by navigating to **aimancloud.online**. At the login page, they enter their credentials, typically a username or email along with a password, and click the "Log in" button. This initial step authenticates the user and grants access to the Nextcloud platform where they can manage their files and data securely.

Step 2: Dashboard Overview

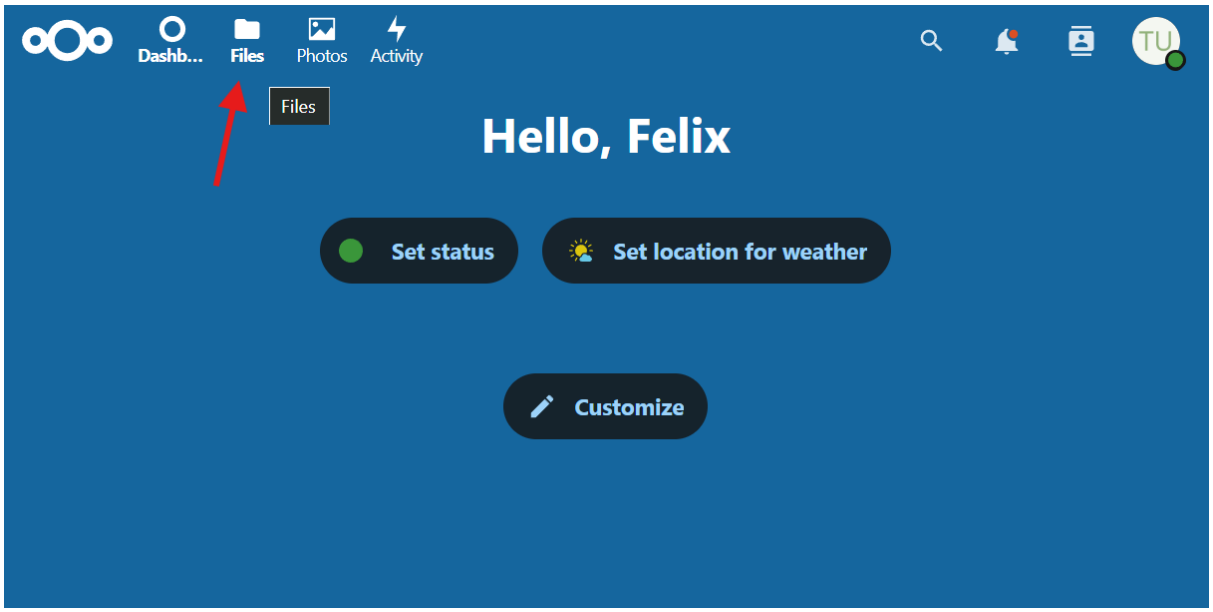


Figure 2: Dashboard Overview

Upon authentication, users access the dashboard. The primary interface of Nextcloud offers rapid access to various sections, including Files, Photos, and Activity. Users can easily access their profile settings, update their status, or modify configurations for localized elements such as weather.

Step 3: File Management

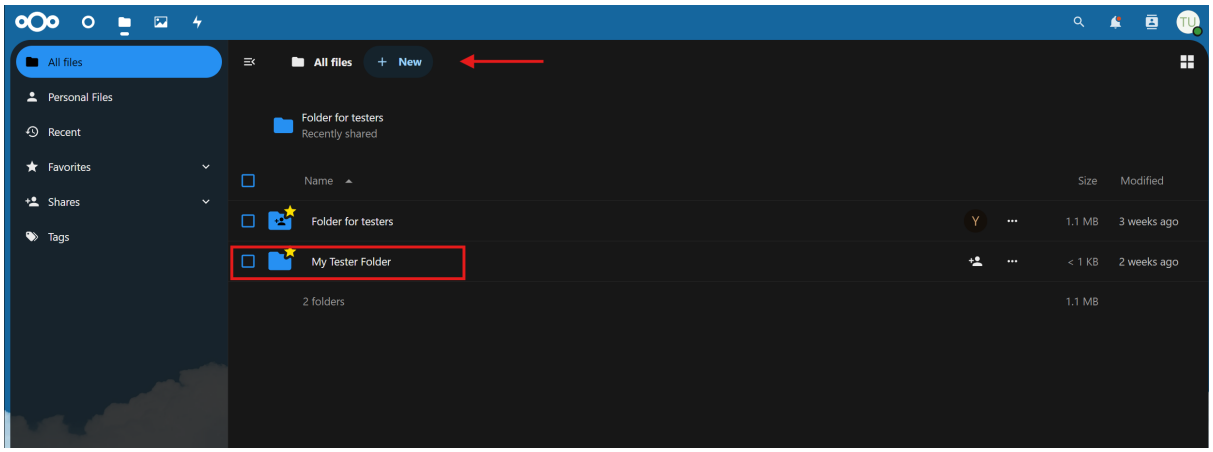


Figure 3: File Management

By selecting the "Files" option, users can view all saved data, create new folders, and access shared directories. The feature allows comprehensive file management, including the creation of directories designated for test environments or project data.

Step 4: Navigating to Specific Folders

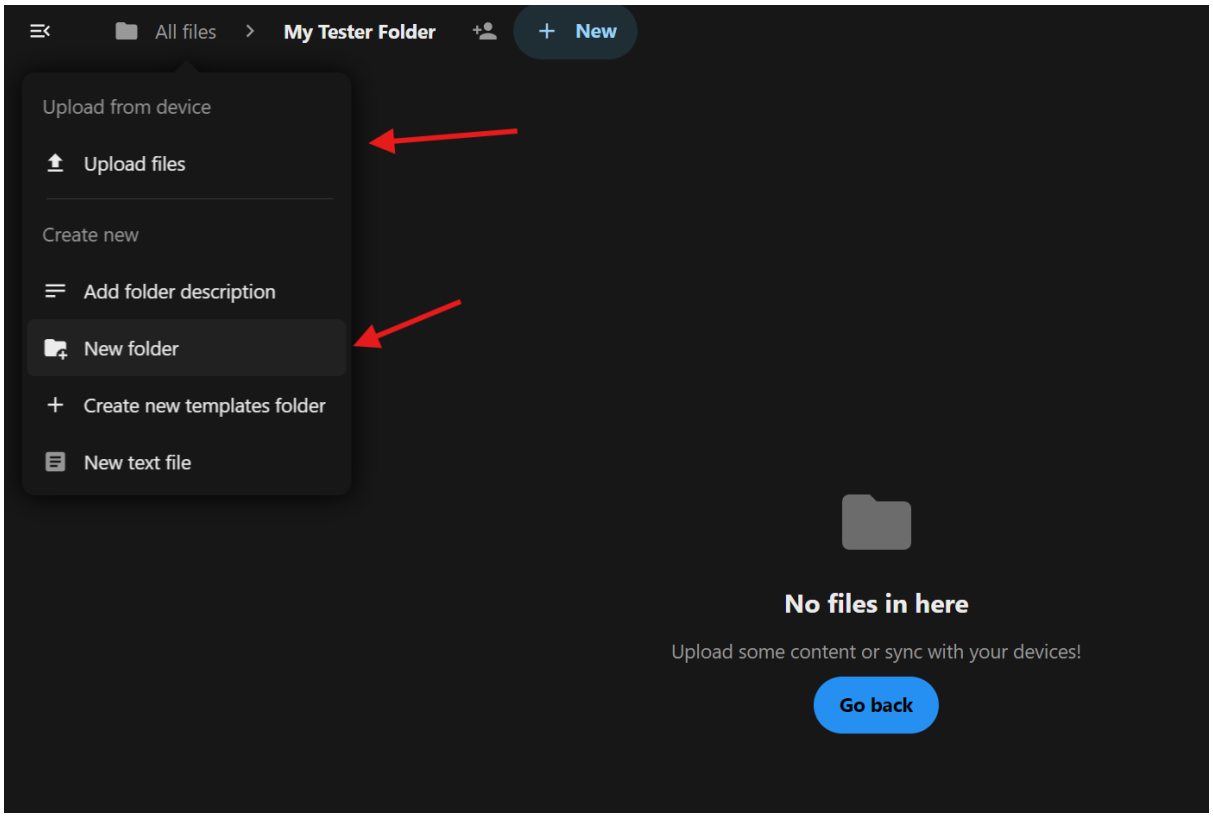


Figure 4:Navigating to Specific Folders

In the Files area, users can select individual folders, such as "My Tester Folder," to access or manage information relevant to their testing or project needs. Additionally, users can manually add files or arrange existing data according to their working needs.

Step 5: Local Environment Setup

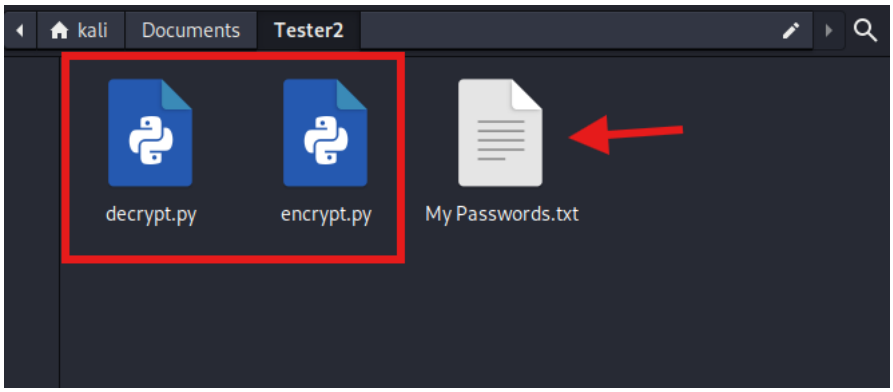


Figure 5: Local Environment Setup

Users then switch to their local machine, here exemplified by a Kali Linux system. Next, user will then need to download the encryption and decryption script from the GitHub repository to their local machine. Next, access their working directory, which contains the encryption and decryption scripts (encrypt.py and decrypt.py), along with the file MyPasswords.txt that contains sensitive data intended for encryption.

Step 6: Running the Encryption Script

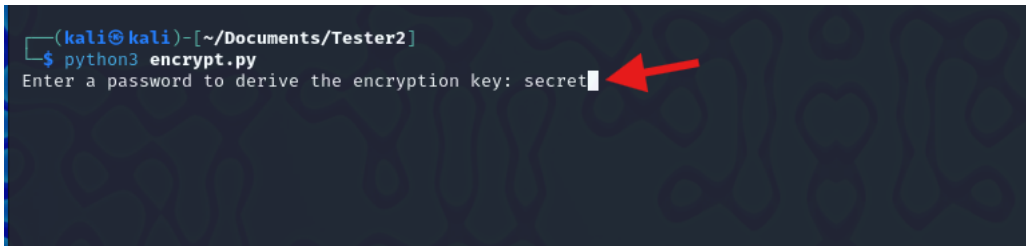


Figure 6: Running the Encryption Script

To start the encryption process, the encrypt.py script is executed, prompting the user to enter a password to derive the AES-256 encryption key. In this example, the encryption key “secret” was chosen. This step highlights the importance of generating keys securely, which is critical for the process of encryption.

Step 7: Encryption Process

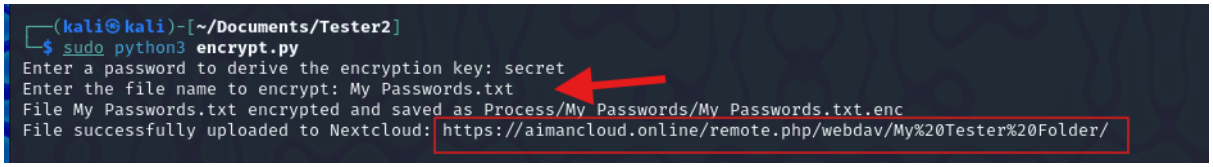


Figure 7: Encryption Process

Once the encryption key is entered, the user specifies the file name (MyPasswords.txt) for the encryption process. The script thoroughly handles the file, ensuring its contents are securely encrypted and uploading the encrypted file to the Nextcloud folder specified by the user.

Step 8: File and Folder Structure 1

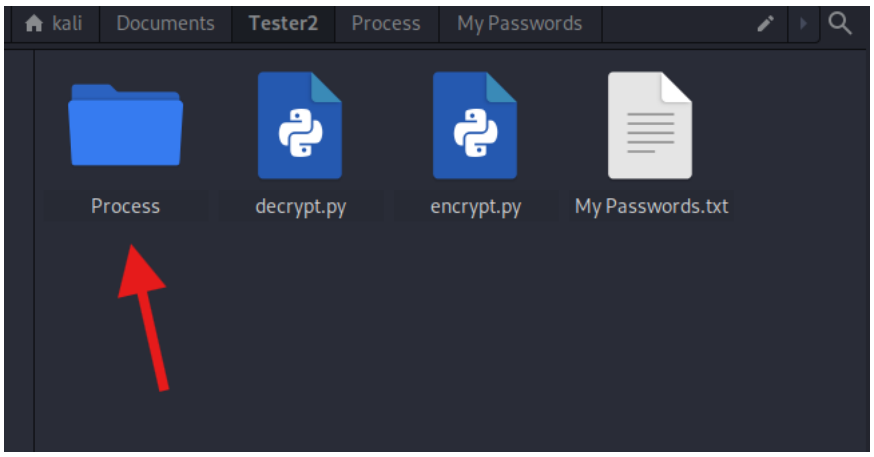


Figure 8: File and Folder Structure 1.1

In addition, the script simultaneously creates a new Process folder if it does not exist, and then save the encrypted version as MyPasswords.txt.enc within it. All encrypted files and their cryptographic components are specifically stored in the Process folder.

Step 9: File and Folder Structure 2

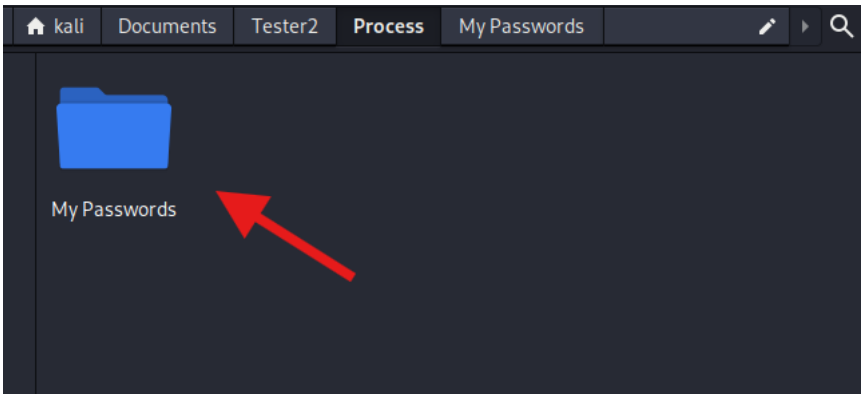


Figure 9: File and Folder Structure 1.2

To effectively organize relevant data, a subfolder is created inside this folder and named after the encrypted file (for example, My Passwords).

Step 10: .enc file and salt.bin creation

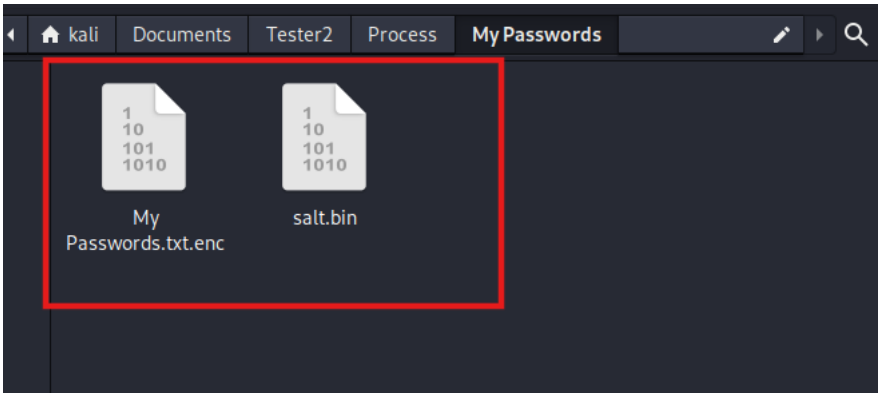


Figure 10: .enc file and salt.bin creation

Within the Process folder, the user can find the MyPasswords.txt.enc file, which holds the encrypted version of the original text. The .enc extension ensure that file is securely encrypted and unreadable without a proper decryption process.

Step 11: Viewing the Encrypted File

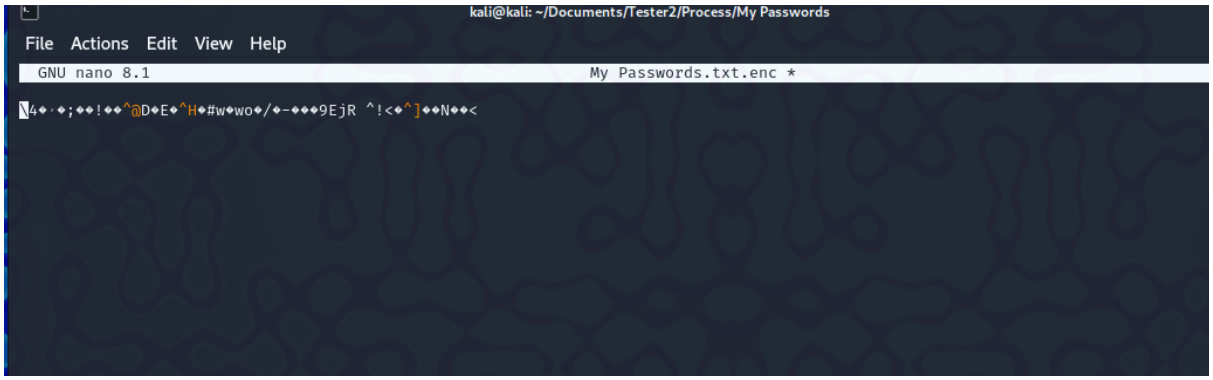


Figure 11: Viewing the Encrypted File

When a user attempts to access the encrypted file My Passwords.txt.enc with a text editor such as GNU Nano, the content displays as scrambled and indecipherable characters.

Step 12: Downloading the Encrypted File from Nextcloud

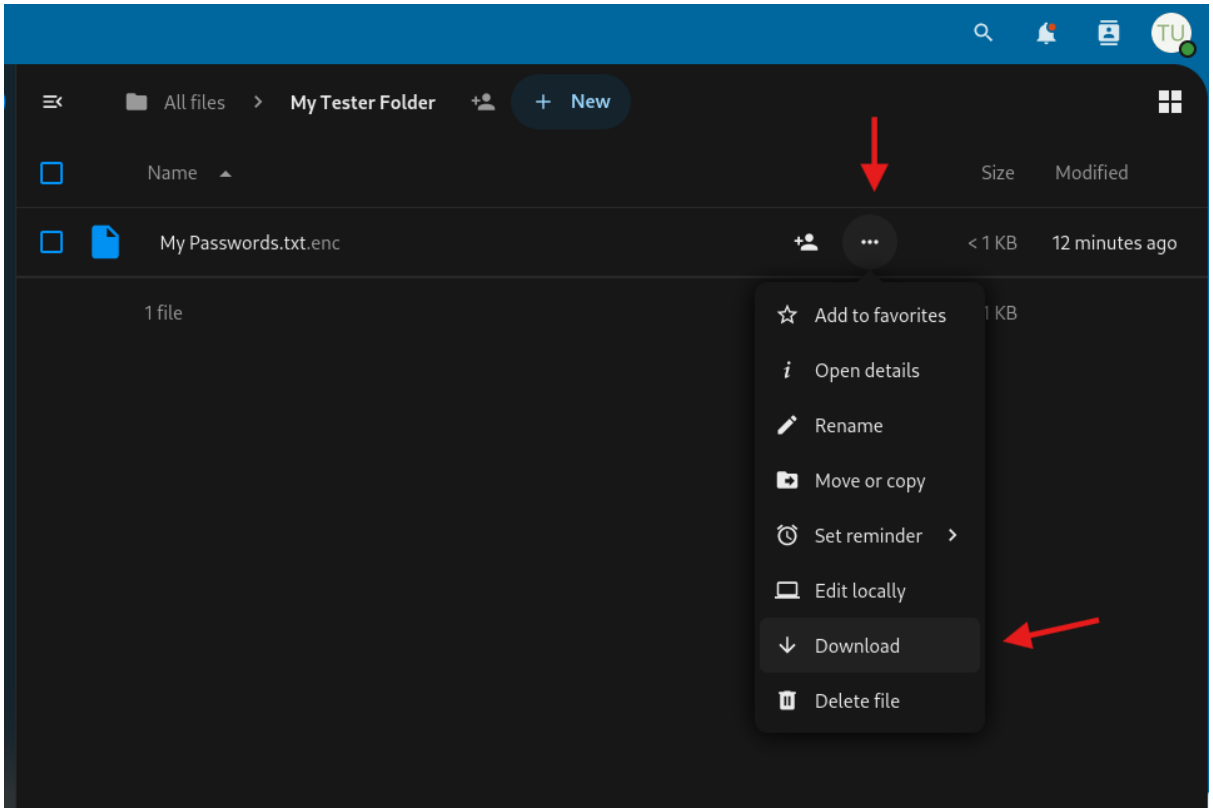


Figure 12:Downloading the Encrypted File from Nextcloud

The user navigates back to the Nextcloud interface and accesses My Tester Folder, where the encrypted file is stored. To get the My Passwords.txt.enc file ready for the decryption procedure, the user can then download it to their local kali machine.

Step 13: Location of the Downloaded Encrypted File

Figure 13: Location of the Downloaded Encrypted File

Once it has been downloaded, the encrypted file will be stored in the local Downloads directory on the Kali Linux machine that the user is using. The file will be referenced during the decryption process.

Step 14: Executing the Decryption Script

Figure 14: Executing the Decryption Script

To begin decryption, the user executes the decrypt.py script. Throughout this procedure, the script requests the user to provide the path to the encrypted file, as shown in figure above.

Step 15: Specifying the Salt File Location

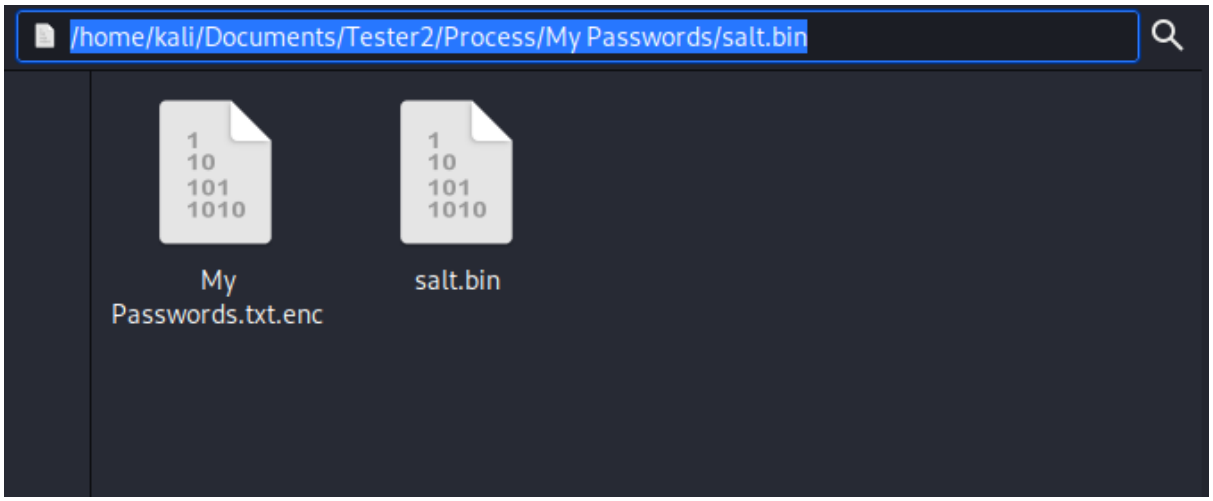


Figure 15: Specifying the Salt File Location

The script requires the path of the salt.bin file linked to the encrypted file. The user accesses the Process folder, which contains both the salt file and the encrypted file, to specify this path.

Step 16: Providing the Decryption Key

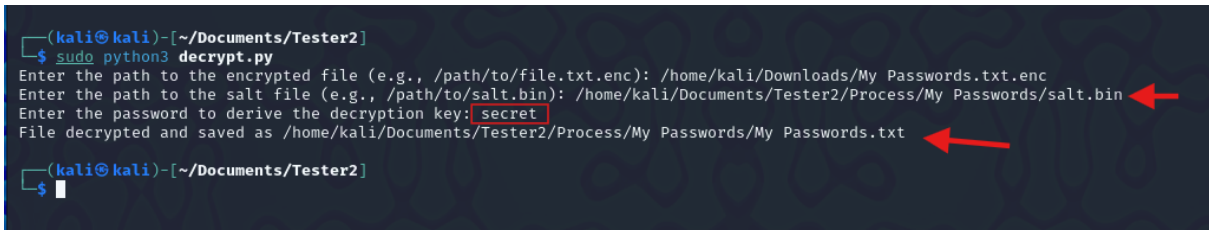


Figure 16: Providing the Decryption Key

Along with the file paths, the user is prompted to input the password used to derive the decryption key. This key is crucial as it interacts with the salt to generate the correct key for decrypting the file, ensuring that only someone with the password can access the original data.

Step 17: Decryption and Accessing the Decrypted File

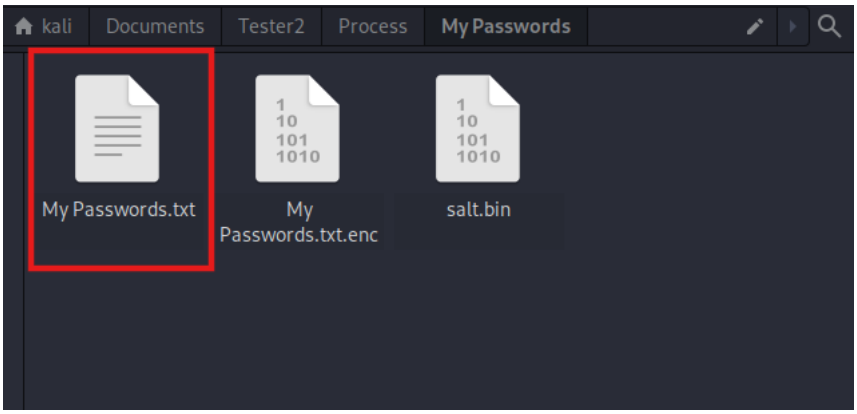


Figure 17: Decryption and Accessing the Decrypted File

After the correct paths and password are provided, the decryption script processes the encrypted data and outputs the decrypted file in the same directory as the salt file. This file, My Passwords.txt is formatted to become readable and accessible for any usage.