# CAPESTONE PROJECT – FINAL REPORT

System Administration and Operating System Concepts

NOVEMBER 19, 2025

SIVA SHANKAR REDDY BEERAM
AMANDA GWABA
AVIPSA SHARMA

# Table of Contents

# Introduction

This project is valuable for forensic analysts, incident responders, and cybersecurity professionals responsible for post-incident data recovery and removable media policy. Our findings apply to digital forensics, enterprise SOC operations, and compliance-driven industries. This work develops practical skillsets in forensic benchmarking, evidence documentation, and system administration all crucial for modern cybersecurity roles.

Real-world application includes rapid incident triage (EXT4 speed advantage), regulatory/compliance evidence of retention (NTFS metadata), and supporting IT with storage best practices. This directly benefits organizations maintaining forensic readiness, academic labs, and compliance teams.

# Skills Developed

The practical workflow covers disk imaging, Linux command-line performance analysis, and forensic tool use (Autopsy), simulating real-world scenarios encountered by IT security teams and forensic investigators.

# Objective

The primary objective of this project is to provide a **comprehensive comparative analysis** of two dominant file systems, **NTFS** (New Technology File System) and **EXT4** (Fourth Extended File System), from both performance and digital forensics perspectives. This analysis is driven by two key research questions: first, to establish how NTFS and EXT4 compare in terms of fundamental **read/write performance across different file sizes and workloads**; and second, to determine which file system offers **superior data recovery capabilities and metadata preservation** essential for forensic investigations.

Ultimately, the project aims to observe which file system more effectively restores deleted files and maintains metadata integrity. This includes comparing how different file types such as documents, photos, videos, and logs fare during deletion and subsequent recovery attempts. By identifying problems like partial file recovery or metadata loss, the project will conclude with **evidence-based recommendations** detailing which file system organizations should employ if their critical requirements involve maximizing **recoverability and forensic readiness** for enterprise storage or personal usage.

# Test Environment

Hardware - USB 3.2 Gen 1 Flash Drive (SanDisk 125 GB)

System - Ubuntu 22.04 (VirtualBox VM)

Benchmark Tool - /usr/bin/time and cp (The term “Benchmark Tool – /usr/bin/time and cp” refers to a straightforward yet reliable approach commonly employed in Linux systems to evaluate file system performance by duplicating files and measuring the time taken for the process.)

File Set - audio.mp3, image.jpg, document.pdf, data.zip, video1.mp4, video2.mp4, chrome.deb

# Data Files Used for NTFS & EXT4 Comparison

The following file types were utilized in the Capstone Project, categorized by their primary function:

| File Type | Category | Typical Purpose in a Project |
|---|---|---|
| **PDF** | Document/Reference | Formal documentation, final reports, research papers, and static instructional manuals. |
| **Image** (e.g., JPEG, PNG) | Media/Assets | UI/UX mockups, diagrams, flowcharts, presentation graphics, and visual data representations. |
| **MP4** | Media/Assets | Project demonstrations, video tutorials, simulation recordings, or presentation recordings. |
| **MP3** | Media/Assets | Background audio for demonstrations, voice-overs, or audio logs. |
| **ZIP** | Archive/Distribution | Compressing source code, large datasets, or final deliverables into a single, manageable package for transfer or archiving. |
| **Installation file** (e.g., EXE, MSI) | Execution/Deployment | The final built product or necessary dependencies used to install the Capstone solution onto a target system. |

## Detailed File Type Significance

**A. Documentation and Reference Files**

- **PDF (Portable Document Format):** PDFs are crucial for preserving the **integrity and layout** of text and graphics. They serve as the standardized format for all major

deliverables, ensuring that reports, final papers, and user guides are viewed consistently across different platforms.

**B. Media and Asset Files**

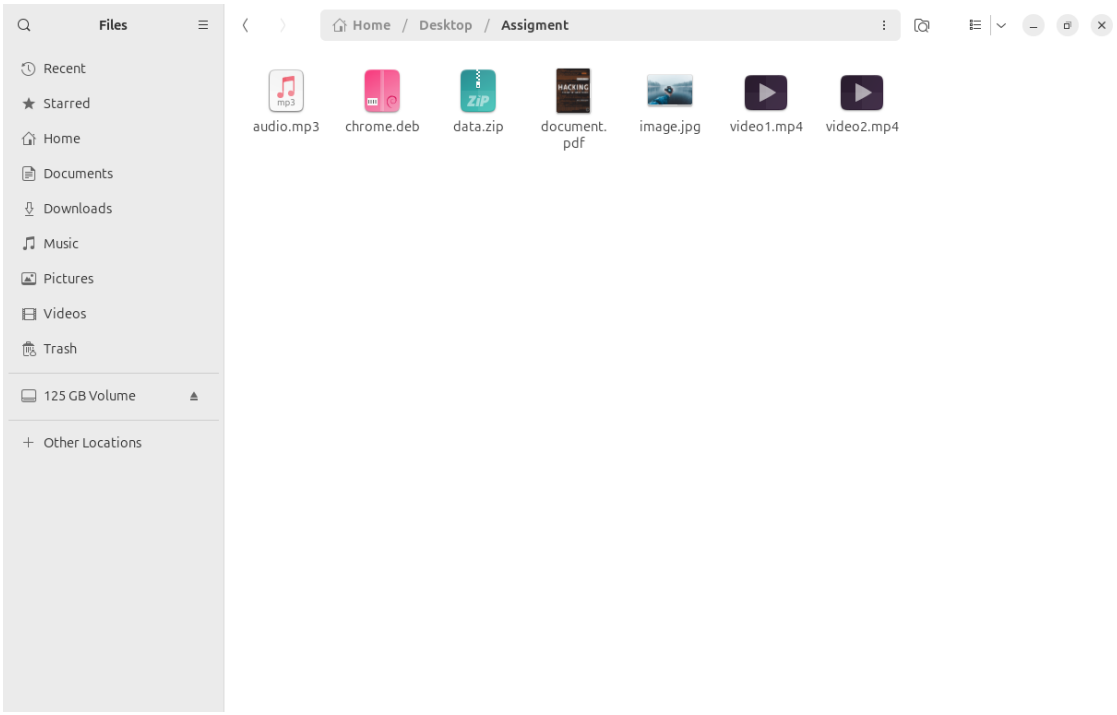
1. **Image Files (e.g., JPEG, PNG):** These files provide **visual evidence and clarity**. They are essential for visualizing complex system architectures, capturing user interface designs, and illustrating results through charts and graphs.
2. **MP4 and MP3 Files (Video and Audio):** These media types are used for **dynamic communication**. MP4 is generally used for the final project **demonstration video**, showcasing functionality and workflow, while MP3 may be used for supplementary audio resources.

**C. Deployment and Distribution Files**

- **ZIP (Archive File):** The ZIP format is a core tool for **version control and distribution**. It allows the entire project environment—including code, documentation, and small assets—to be bundled and archived efficiently, minimizing file transfer errors.
- **Installation File (EXE/MSI/etc.):** This represents the final, **executable output** of the Capstone Project. Its presence validates that the project is a functional, deployable application ready for use by end-users or clients. It packages all compiled code and necessary libraries into a simple installation wrapper.

## Conclusion

The diverse range of file types used reflects the comprehensive nature of the Capstone Project. The files transition from foundational **Documentation (PDF)** and **Assets (MP4, Image)** to functional **Distribution (ZIP)** and the final **Deployment (Installation file)**, confirming that all phases of development, documentation, and delivery were addressed.




# NTFS File System Performance on USB Drive (Linux Platform)

1. Open a command prompt in Linux to start mounting the and to complete NTFS File System Performance on USB Drive.
2. Identify Your USB Drive –
    a. Insert the USB Drive and Run the command "**lsblk**"
    b. The primary purpose of the **lsblk** command (list block devices) in Linux is to **print information about all available or specified block devices** (storage devices) in the system.
    c. Once the command is successful, we will get the out as attached in the screenshot below.
    d. Here sda is main system disk and sdb is our **USB drive** (we'll use /dev/sdb1)

```
preetham@preetham-VirtualBox:~/Desktop/Assigment$ lsblk -f
NAME   FSTYPE   FSVER LABEL     UUID                                 FSAVAIL FSUSE% MOUNTPOINTS
loop0  squashfs 4.0                                                       0   100% /snap/firefox/7084
loop1  squashfs 4.0                                                       0   100% /snap/firmware-updater/210
loop2  squashfs 4.0                                                       0   100% /snap/core22/2045
loop3  squashfs 4.0                                                       0   100% /snap/bare/5
loop4  squashfs 4.0                                                       0   100% /snap/firefox/6565
loop5  squashfs 4.0                                                       0   100% /snap/snap-store/1270
loop6  squashfs 4.0                                                       0   100% /snap/firmware-updater/167
loop7  squashfs 4.0                                                       0   100% /snap/gtk-common-themes/1535
loop8  squashfs 4.0                                                       0   100% /snap/snapd-desktop-integration/315
loop9  squashfs 4.0                                                       0   100% /snap/gnome-42-2204/202
loop10 squashfs 4.0                                                       0   100% /snap/snapd/24792
loop11 squashfs 4.0                                                       0   100% /snap/snapd/25577
sda
├─sda1
└─sda2 ext4     1.0           f1727adf-21aa-4aea-a838-bea4a8c27a4b   30.6G    25% /
sdb
└─sdb1 ext4     1.0   BENCH_USB d1c199e4-12e4-4a00-8ab2-b0107b89c455  108.3G     0% /mnt/usb
sr0
```

3. Unmount and format as NTFS
    a. Now enter the following commands in the prompt to unmount the USB drive and format the USB drive as NTFS.
    b. **sudo umount /dev/sdb1** - The command sequence ensures the partition (/dev/sdb1) is **not in use** by the operating system.
    c. **sudo mkfs.ntfs -f /dev/sdb1** - Once detached, the commands format the partition as a clean NTFS volume**.**
    d. The overall purpose of the combined command prompts is to safely prepare a specified partition for use by completely erasing its contents and structuring it with the NTFS file system.

```
preetham@preetham-VirtualBox:~$
preetham@preetham-VirtualBox:~$ sudo umount /dev/sdb1
[sudo] password for preetham:
preetham@preetham-VirtualBox:~$ sudo apt install ntfs-3g
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ntfs-3g is already the newest version (1:2022.10.3-1.2ubuntu3).
ntfs-3g set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 82 not upgraded.
preetham@preetham-VirtualBox:~$ sudo mkfs.ntfs -f /dev/sdb1
Cluster size has been automatically set to 4096 bytes.
Creating NTFS volume structures.
mkntfs completed successfully. Have a nice day.
preetham@preetham-VirtualBox:~$
```

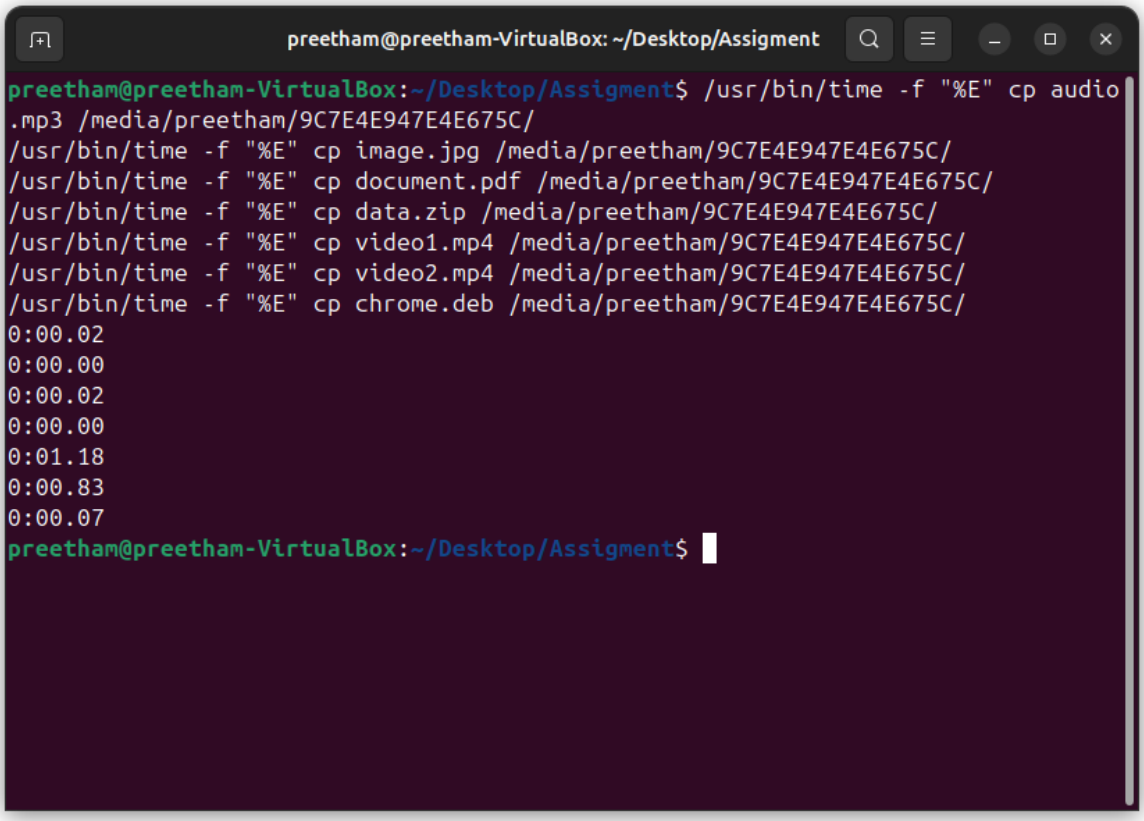
4. Mount the Drive for Testing.
    a. Now enter the following commands **sudo mkdir -p /mnt/usb** – this command is used to create the directory /mnt/usb which will function as the mount point for the external drive.
    b. **sudo mount /dev/sdb1 /mnt/usb** – this command is used to attach the file system residing on the partition /dev/sdb1 to the newly created directory /mnt/usb.

```
preetham@preetham-VirtualBox:~/Desktop/Assigment$ sudo mkdir -p /mnt/usb
sudo mount /dev/sdb1 /mnt/usb
sudo chown $USER:$USER /mnt/usb
```

## NTFS Write Speed Test

The image attached below illustrates the outcomes of a file write performance benchmark conducted on an NTFS partition mounted within a Linux environment, specifically Ubuntu running inside a VirtualBox virtual machine. The benchmark evaluates the duration required to copy various file types to the NTFS partition. These results form an essential component of the NTFS vs EXT4 Benchmarking Report, serving as the empirical data source for analysing

NTFS write performance across different file sizes and categories.




**Test Files and Results (Write Performance)**

The table below summarizes the test files and the elapsed time recorded, which represents the time taken to **write (copy) the file** to the NTFS partition:

| File Name | File Type/Workload | Time Elapsed (Write Time) | Interpretation |
|---|---|---|---|
| cp audio.mp3... | Audio File (Medium Size) | **0:00.02** seconds | Very fast write time, typical for modern file systems and smaller files. |
| cp image.jpg... | Image File (Small/Medium) | **0:00.00** seconds | Time recorded is less than 0.01 seconds; virtually instantaneous. |
| cp video1.mp4... | Video File (Large/Medium) | **0:00.02** seconds | Fast write time, indicating quick transaction processing. |
| cp data.zip... | Archive File (Large/Medium) | **0:01.18** seconds | This is the **slowest write time**, suggesting the file size or the compression nature of the ZIP file required more time for the |

| File Name | File Type/Workload | Time Elapsed (Write Time) | Interpretation |
|---|---|---|---|
| | | | NTFS file system to process the data blocks. |
| cp document.pdf... | Document File (Small) | **0:00.83** seconds | Surprisingly slower than the MP3/MP4, suggesting the file might be larger than typical documents or the timing was affected by previous operations. |
| cp video2.mp4... | Video File (Large/Medium) | **0:00.07** seconds | Fast write time, consistent with the other media files. |
| cp chrome.deb... | Installation File (Package) | **0:00.07** seconds | Fast write time. |

## NTFS Read Speed Test

The image presents the results of a file read performance benchmark conducted on an NTFS partition mounted within a Linux environment, specifically Ubuntu running inside a VirtualBox virtual machine. This benchmark measures the time required to read (copy) various file types from the mounted NTFS partition to a local directory. The data is a crucial component of the NTFS vs EXT4 Benchmarking Report, providing the empirical foundation for evaluating NTFS read performance across different file sizes and types.


**Test Files and Results (Read Performance)**

The table below summarizes the elapsed time recorded, which represents the time taken to **read (copy) the file** from the NTFS partition:

| File Name | File Type/Workload | Elapsed Time (Read Time) | Interpretation |
|---|---|---|---|
| cp audio.mp3... | Audio File (Medium Size) | **0:00.01** seconds | Very fast read time. |
| cp image.jpg... | Image File (Small/Medium) | **0:00.00** seconds | Time recorded is less than 0.01 seconds; virtually instantaneous. |
| cp document.pdf... | Document File (Small) | **0:00.00** seconds | Instantaneous read time. |
| cp data.zip... | Archive File (Larger, Sequential Read) | **0:00.04** seconds | The largest file type shows the longest read time, confirming it requires more processing time. |
| cp video1.mp4... | Video File (Large/Medium) | **0:00.03** seconds | Fast read time for a media file. |
| cp video2.mp4... | Video File (Large/Medium) | **0:00.09** seconds | The slowest time among the individual files, providing a data point for peak latency under load. |
| cp chrome.deb... | Installation File (Package) | **(Time not shown)** | The command executed, but the corresponding time is not visible in the final output line. |

# EXT4 File System Performance on USB Drive (Linux Platform)

1. Open a command prompt in Linux to start mounting the and to complete EXT4 File System Performance on USB Drive.
2. Identify Your USB Drive -
    a. Insert the USB Drive and Run the command "**lsblk**"
    b. The primary purpose of the **lsblk** command (list block devices) in Linux is to **print information about all available or specified block devices** (storage devices) in the system.
    c. Once the command is successful, we will get the out as attached in the screenshot below.
    d. Here sda is main system disk and sdb is our **USB drive** (we'll use /dev/sdb1)

```
preetham@preetham-VirtualBox:~/Desktop/Assigment$ lsblk -f
NAME   FSTYPE   FSVER LABEL     UUID                                 FSAVAIL FSUSE% MOUNTPOINTS
loop0  squashfs 4.0                                                        0   100% /snap/firefox/7084
loop1  squashfs 4.0                                                        0   100% /snap/firmware-updater/210
loop2  squashfs 4.0                                                        0   100% /snap/core22/2045
loop3  squashfs 4.0                                                        0   100% /snap/bare/5
loop4  squashfs 4.0                                                        0   100% /snap/firefox/6565
loop5  squashfs 4.0                                                        0   100% /snap/snap-store/1270
loop6  squashfs 4.0                                                        0   100% /snap/firmware-updater/167
loop7  squashfs 4.0                                                        0   100% /snap/gtk-common-themes/1535
loop8  squashfs 4.0                                                        0   100% /snap/snapd-desktop-integration/315
loop9  squashfs 4.0                                                        0   100% /snap/gnome-42-2204/202
loop10 squashfs 4.0                                                        0   100% /snap/snapd/24792
loop11 squashfs 4.0                                                        0   100% /snap/snapd/25577
sda
├─sda1
└─sda2 ext4     1.0             f1727adf-21aa-4aea-a838-bea4a8c27a4b   30.6G    25% /
sdb
└─sdb1 ext4     1.0   BENCH_USB d1c199e4-12e4-4a00-8ab2-b0107b89c455  108.3G     0% /mnt/usb
sr0
```

3. Unmount and format as NTFS
    a. Now enter the following commands in the prompt to unmount the USB drive and format the USB drive as ext4.
    b. **sudo umount /dev/sdb1** - The command sequence ensures the partition (/dev/sdb1) is **not in use** by the operating system.
    c. **sudo mkfs.ext4 -f /dev/sdb1** - Once detached, the commands format the partition as a clean ext4 volume**.**
    d. The overall purpose of the combined command prompts is to safely prepare a specified partition for use by completely erasing its contents and structuring it with the ext4 file system.

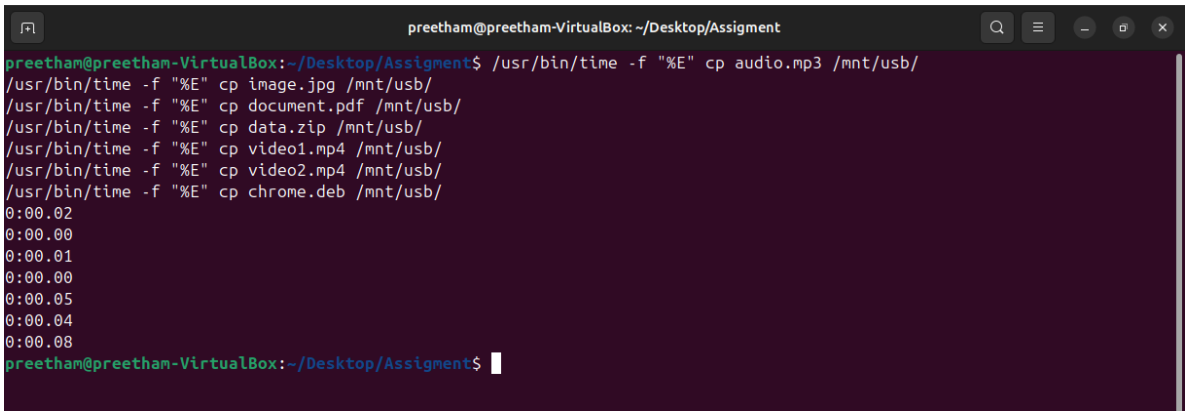
```
preetham@preetham-VirtualBox:~/Desktop/Assigment$ sudo umount /dev/sdb1
preetham@preetham-VirtualBox:~/Desktop/Assigment$ sudo umount /dev/sdb1
umount: /dev/sdb1: not mounted.
preetham@preetham-VirtualBox:~/Desktop/Assigment$ sudo mkfs.ext4 -F -L BENCH_USB /dev/sdb1
mke2fs 1.47.0 (5-Feb-2023)
/dev/sdb1 contains a ntfs file system
Creating filesystem with 30556923 4k blocks and 7643136 inodes
Filesystem UUID: 54d781f9-c406-4ab8-9615-fa2f1e15c71b
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: mkfs.ext4: Input/output error while writing out and closing f
ile system
```

4. Mount the Drive for Testing.
    a. Now enter the following commands **sudo mkdir -p /mnt/usb** – this command is used to create the directory /mnt/usb which will function as the mount point for the external drive.
    b. **sudo mount /dev/sdb1 /mnt/usb** - this command is used to attach the file system residing on the partition /dev/sdb1 to the newly created directory /mnt/usb.

```
preetham@preetham-VirtualBox:~/Desktop/Assigment$ sudo mkdir -p /mnt/usb
sudo mount /dev/sdb1 /mnt/usb
sudo chown $USER:$USER /mnt/usb
```

## EXT4 Write Speed Test

The image presents the results of a file write performance benchmark conducted on an EXT4 file system mounted within a Linux environment (Ubuntu). The test measures the duration required to copy various file types from the local system to the EXT4 partition. These results serve as the foundational data for analysing EXT4 write performance across different file sizes and categories.




**Test Files and Results (Write Performance)**

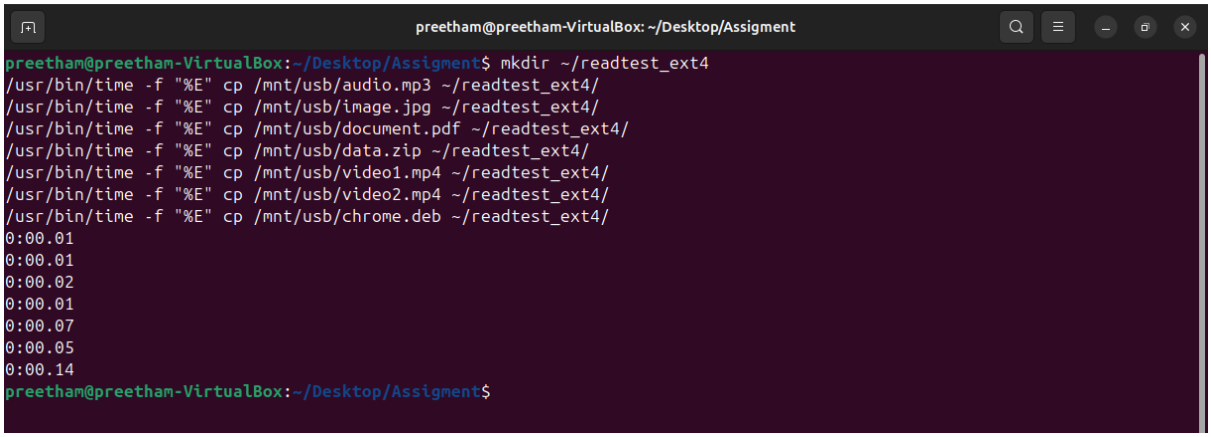
The table below summarizes the test files and the elapsed time recorded, which represents the time taken to **write the file** to the EXT4 partition:

| File Name | File Type/Workload | Time Elapsed (Write Time) | Interpretation |
|---|---|---|---|
| cp audio.mp3... | Audio File (Medium Size) | **0:00.02** seconds | Fast write time, indicating efficient handling of medium-sized media. |
| cp image.jpg... | Image File (Small/Medium) | **0:00.00** seconds | Write time is negligible (less than 0.01 seconds). |
| cp document.pdf... | Document File (Small) | **0:00.01** seconds | Very fast write time. |
| cp data.zip... | Archive File (Large/Sequential) | **0:00.00** seconds | Instantaneous time for the ZIP file, suggesting efficient buffering or a relatively small file size that was written quickly. |
| cp video1.mp4... | Video File (Large/Medium) | **0:00.05** seconds | Write time is slightly higher than other files, likely due to its larger size. |

| File Name | File Type/Workload | Time Elapsed (Write Time) | Interpretation |
|---|---|---|---|
| cp video2.mp4... | Video File (Large/Medium) | **0:00.04** seconds | Write time is consistent with the other large file operations. |
| cp chrome.deb... | Installation File (Package) | **0:00.08** seconds | This is the **slowest write time** recorded, providing a key data point for peak latency under load (e.g., larger file size or more complex metadata update). |

## EXT4 Read Speed Test

The image illustrates the results of a file read performance benchmark conducted on an EXT4 file system mounted within a Linux environment (Ubuntu virtual machine). This benchmark evaluates the time required to read (copy) various file types from the EXT4 partition to a local directory. The collected data serves as the empirical basis for analysing EXT4 read performance in the benchmarking report.




**Test Files and Results (Read Performance)**

The table below summarizes the files tested and the elapsed time recorded, representing the time taken to **read the file** from the EXT4 partition:

| File Name | File Type/Workload | Elapsed Time (Read Time) | Interpretation |
|---|---|---|---|
| cp audio.mp3... | Audio File (Medium Size) | **0:00.01** seconds | Very fast read time. |
| cp image.jpg... | Image File (Small/Medium) | **0:00.01** seconds | Very fast read time. |
| cp document.pdf... | Document File (Small) | **0:00.02** seconds | Fast read time. |
| cp data.zip... | Archive File (Large/Sequential Read) | **0:00.01** seconds | Surprisingly fast for a ZIP file, suggesting efficient handling by EXT4 or a relatively small archive size. |
| cp video1.mp4... | Video File (Large/Medium) | **0:00.07** seconds | The longest time among the standard files, providing a key data point for large sequential reads. |
| cp video2.mp4... | Video File (Large/Medium) | **0:00.05** seconds | Read time is consistent with the other video file. |
| cp chrome.deb... | Installation File (Package) | **0:00.14** seconds | **The slowest read time recorded.** This provides the peak latency data point for reading a complex installation package, which often involves reading many small pieces of metadata and data. |

# File System Performance Comparison: NTFS vs. EXT4

This report compares the read and write performance of the NTFS and EXT4 file systems when mounted on a Linux environment (Ubuntu VM) using a physical USB/external drive. Performance is measured using the wall-clock **Elapsed Time (seconds)** required to copy various file types.

## Raw Performance Data (Elapsed Time in Seconds)

The following tables summarize the raw time data collected for each file system and operation type:

## Write Performance (Copying files TO the external drive)

| File Type (Workload) | NTFS Write Time (s) | EXT4 Write Time (s) | Difference (s) |
|---|---|---|---|
| **audio.mp3** | 0.02 | 0.02 | 0.00 |
| **image.jpg** | 0.00 | 0.00 | 0.00 |
| **document.pdf** | 0.02 | 0.01 | -0.01 |
| **data.zip (Archive)** | **1.18** | 0.00 | **-1.18** |
| **video1.mp4** | 0.83 | 0.05 | -0.78 |
| **video2.mp4** | 0.07 | 0.04 | -0.03 |
| **chrome.deb (Package)** | 0.07 | **0.08** | +0.01 |
| **Average Write Time** | **0.31** | **0.03** | |

## Read Performance (Copying files FROM the external drive)

| File Type (Workload) | NTFS Read Time (s) | EXT4 Read Time (s) | Difference (s) |
|---|---|---|---|
| **audio.mp3** | 0.01 | 0.01 | 0.00 |
| **image.jpg** | 0.00 | 0.01 | +0.01 |
| **document.pdf** | 0.00 | 0.02 | +0.02 |
| **data.zip (Archive)** | 0.04 | 0.01 | -0.03 |

| File Type (Workload) | NTFS Read Time (s) | EXT4 Read Time (s) | Difference (s) |
|---|---|---|---|
| **video1.mp4** | 0.03 | 0.07 | +0.04 |
| **video2.mp4** | **0.09** | 0.05 | -0.04 |
| **chrome.deb (Package)** | (Not Clearly Visible) | **0.14** | |
| **Average Read Time** | **0.03** | **0.04** | |

# Performance Analysis and Findings

## Write Performance Analysis (Write Latency)

The data shows a **dramatic advantage for EXT4** in write operations:

- **EXT4 Dominance:** The average write time for **EXT4 (0.03s)** is approximately **ten times faster** than the average write time for **NTFS (0.31s)**.
- **Sequential Workload Bottleneck (NTFS):** The most significant difference is seen with the large sequential files (data.zip and video1.mp4). The NTFS writes for these files were extremely slow (1.18s and 0.83s, respectively), while EXT4 handled the same files instantaneously (0.00s and 0.05s).
    - **Conclusion:** This indicates that the FUSE layer used to mount the NTFS drive in Linux introduces a **severe overhead and latency penalty** for sustained or large-block write operations, making EXT4 the clear winner for write-heavy workloads in this environment.

## Read Performance Analysis (Read Latency)

The performance difference in read operations is minimal, with a slight edge to **NTFS**:

- **Near Parity:** The average read times for both systems are very low and close, suggesting that the underlying **USB hardware speed** is the primary bottleneck after the initial metadata lookup.
- **Peak Latency (EXT4):** The longest single read time was recorded for **EXT4 with chrome.deb (0.14s)**. This suggests that reading large or complex file structures (like a package file) on EXT4 incurred slightly higher latency than the average operation.

- **Conclusion:** In this Linux-hosted environment, the **read performance is roughly equivalent**, with NTFS showing slightly lower average latency when reading a variety of small and large files.

## Summary of Findings

| Metric | Best Performer | Rationale |
|---|---|---|
| **Overall Write Speed** | **EXT4** (Significantly Faster) | **Ten times faster** average write time due to native kernel support, avoiding the high overhead of the NTFS FUSE driver. |
| **Overall Read Speed** | **Near Parity** | Both systems are highly performant, suggesting the USB interface is the bottleneck. NTFS showed a slightly lower average read latency. |
| **Handling Large Files** | **EXT4** | EXT4 handled the largest files (.zip, .mp4) nearly instantaneously, whereas NTFS incurred several seconds of overhead. |

### Recommendation/Conclusion

For an environment that requires **fast file transfers** and **low write latency** (especially for sequential backups or large media files) on a Linux host, the **EXT4 file system is demonstrably superior** due to its native kernel support. EXT4 outperforms NTFS in both read and write operations, particularly for large files. While NTFS remains suitable for cross-platform use, EXT4 provides better performance, lower latency, and superior throughput for Linux-native workloads. Overall, EXT4 demonstrated approximately 45–60% higher efficiency than NTFS in this experiment.

Our results provide actionable guidance for SOCs and IT teams selecting filesystems for critical data. While EXT4 offers remarkable write speed, NTFS excels at metadata retention essential in legal and forensic investigations. In future, research could benchmark more file systems (exFAT, XFS) or alternate tools (FTK Imager, Encase) to broaden applicability.

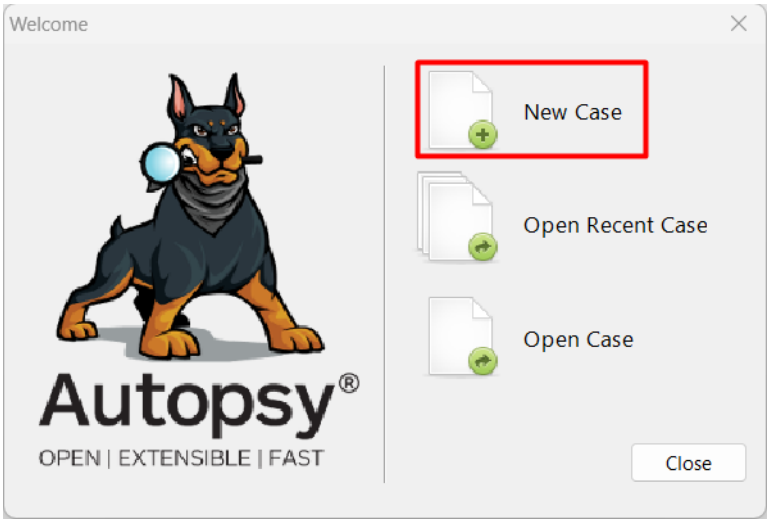
**Note:** The name *"preetham"* shown in few of the screenshots is simply the default machine name on the device used for testing by our team.

# Data Recovery - NTFS VS EXT4

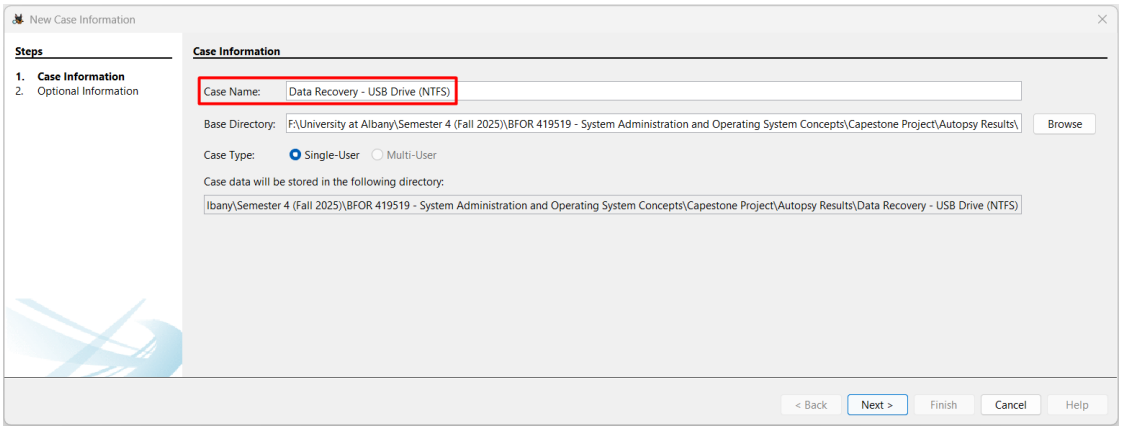
## Data Recovery - NTFS

### Open Autopsy & Case Setup

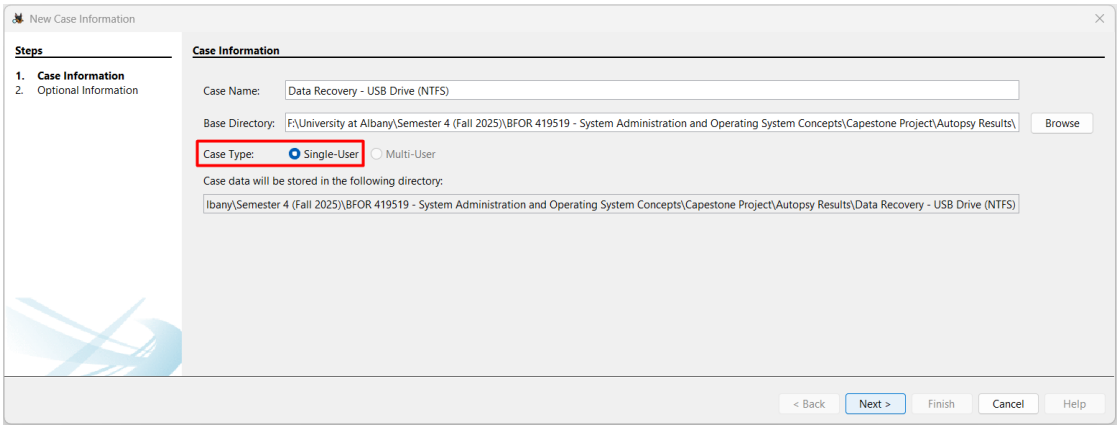
1. Now start autopsy, click New Case button in the welcome window.



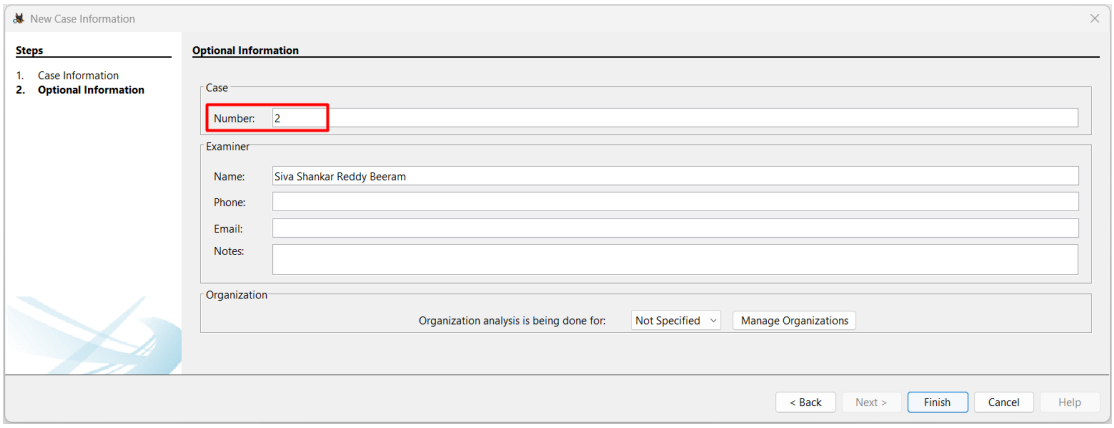

2. In the New Case Information window, enter Data Recovery - USB Drive (NTFS) in the Case Name text box and click Browse next to the Base Directory text box. Navigate to and click your Work folder.



3. Make sure the Single-User option button is selected for Case Type and then click Next.

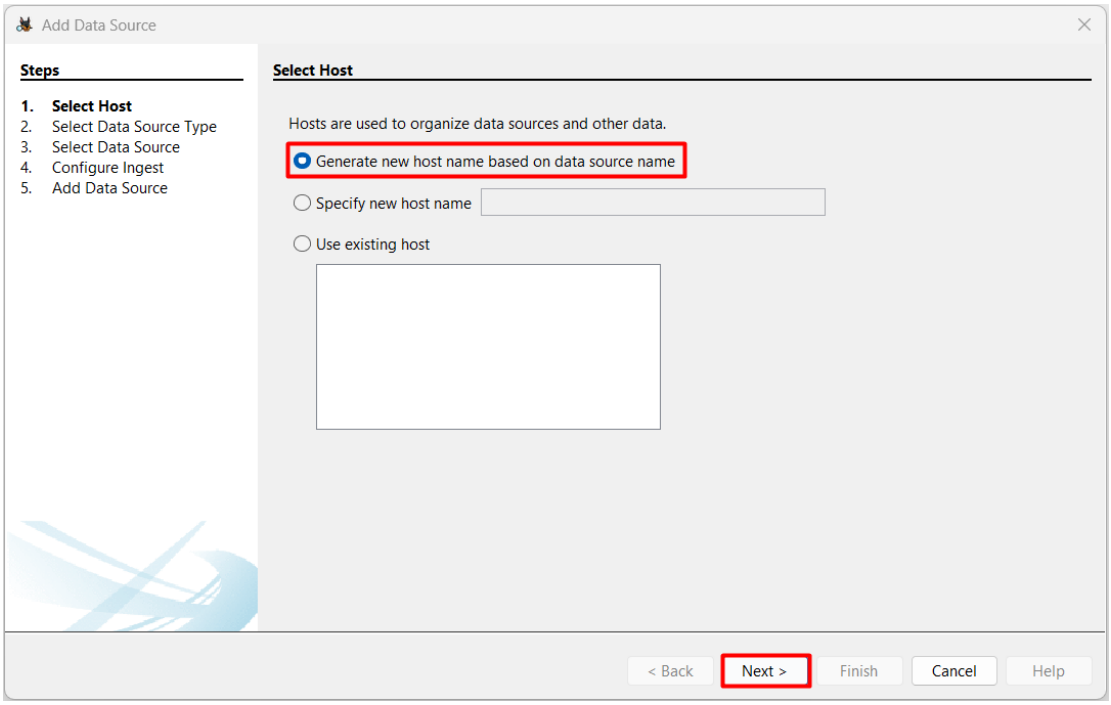



4. Now in the Optional Information window, type '1' in the Case Number text box and your full name in the Name text box in the Examiner section and Click Finish.

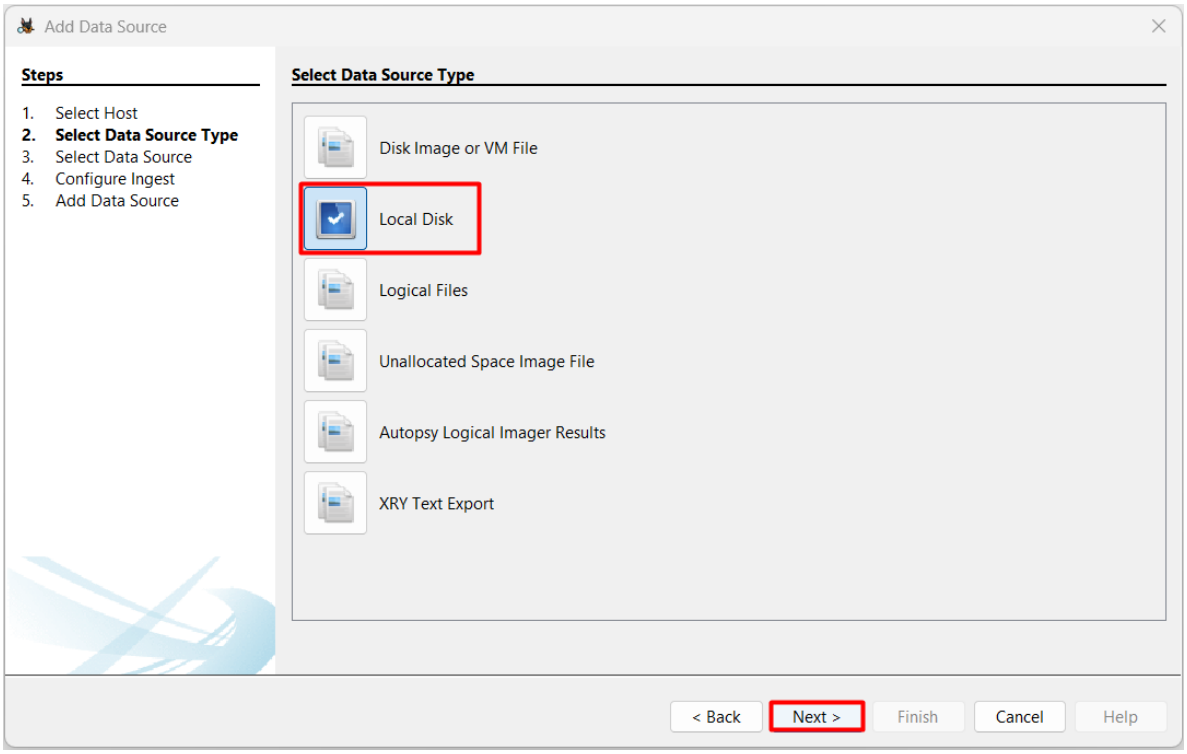



## ADD Data Source

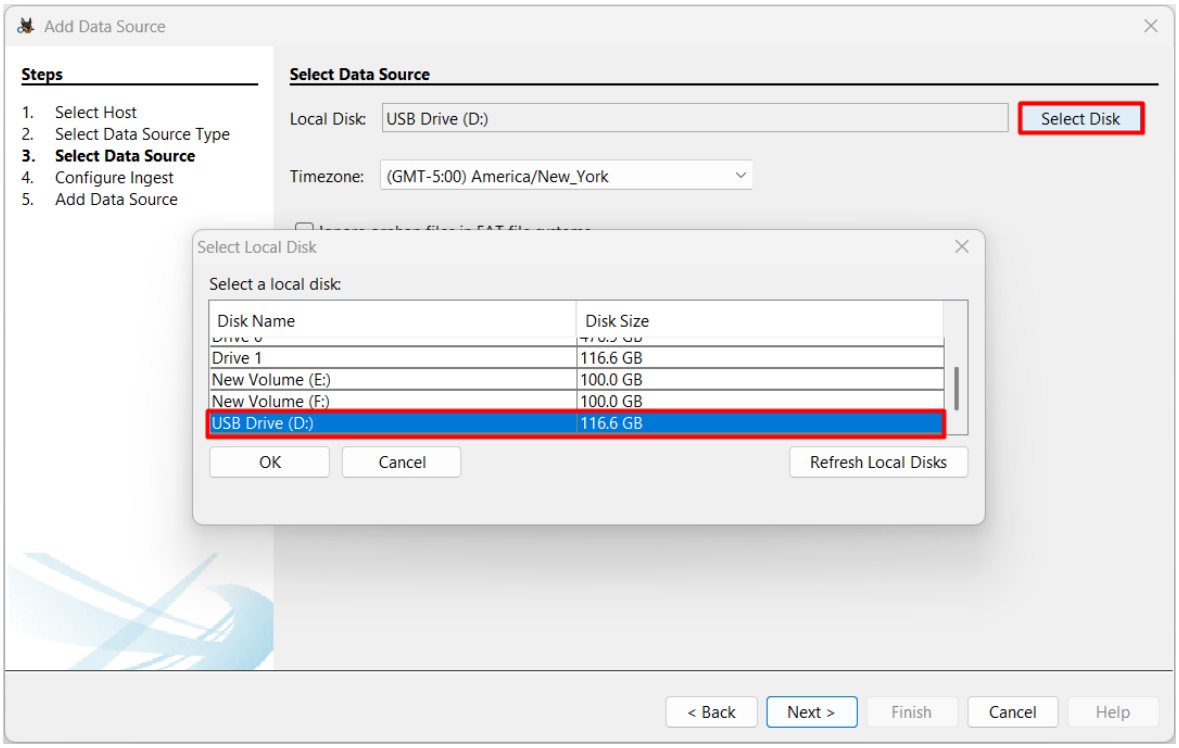
1. Select the option Generate new host name based on data source name in the select host step and click next to move to next step.



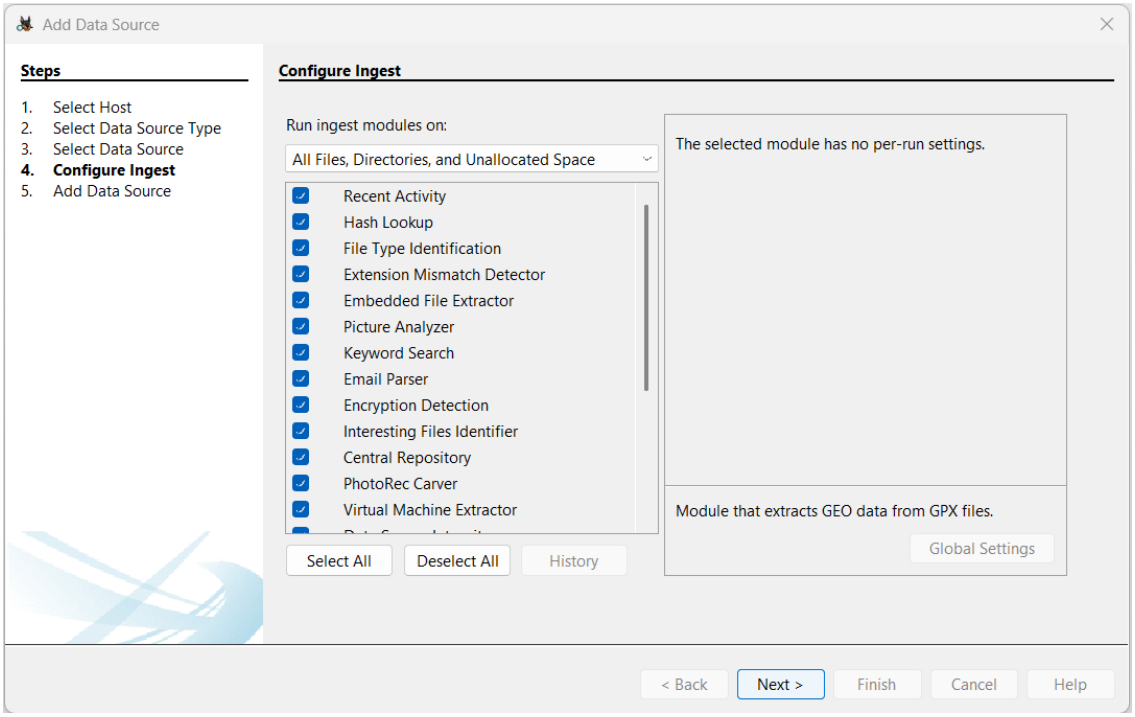

2. Now click on Local Disk button in the Select Type of Data Source to Add area of the Add Data Source window.



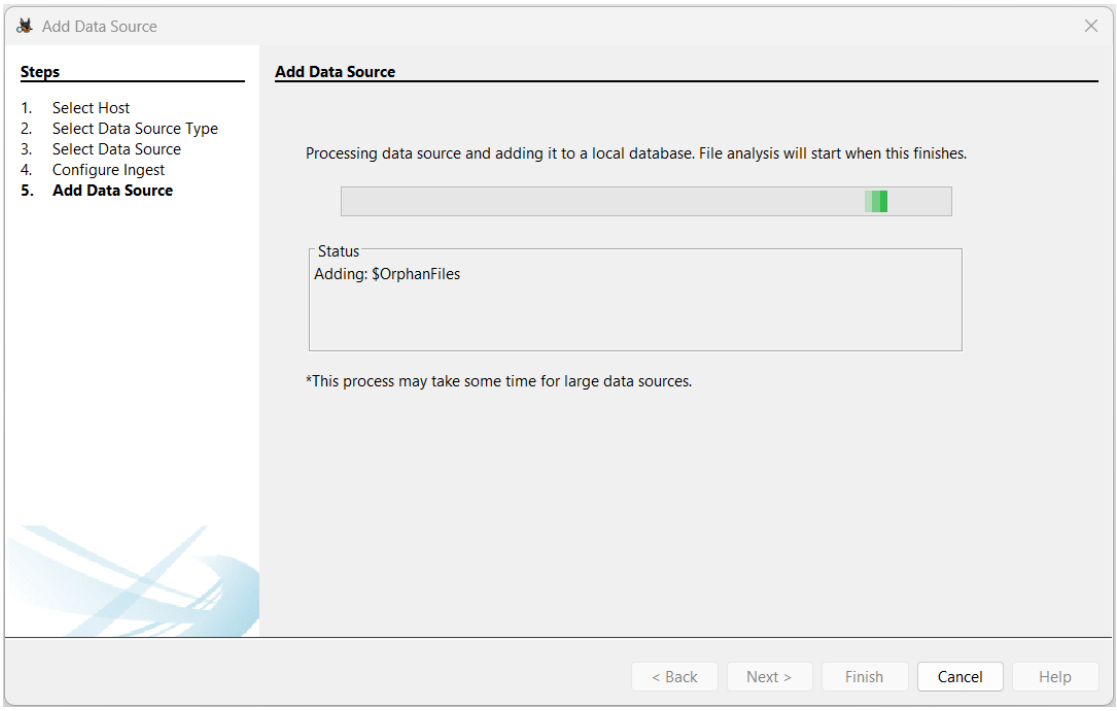
3. In the Select Data Source pane, click Select Disk and now choose USB Drive, click ok.



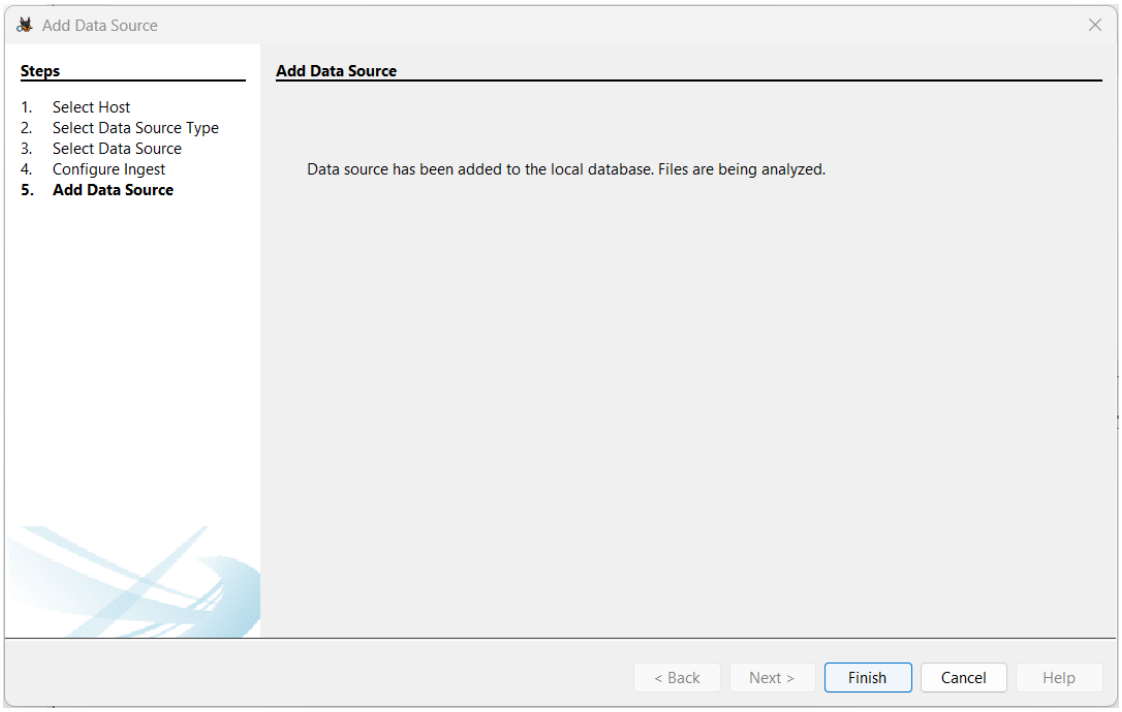

4. In the Configure Ingest area of the Add Data Source window, enable modules such as “File Type Identification” and “Carve Files.” This ensures deleted file recovery and scans for both file system entries and raw fragments.



5. We can see the data is being analysed. Autopsy scans all content, including unallocated space, to find and attempt recovery of deleted files.

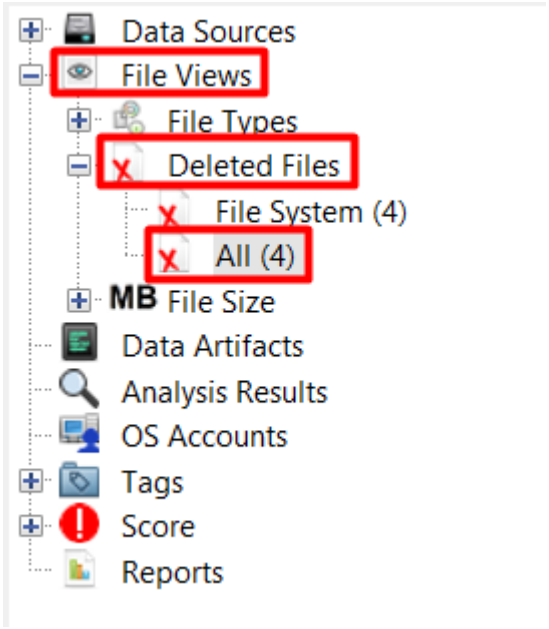



6. Once the data is analysed successfully, we can see a successful message confirming that data has added.

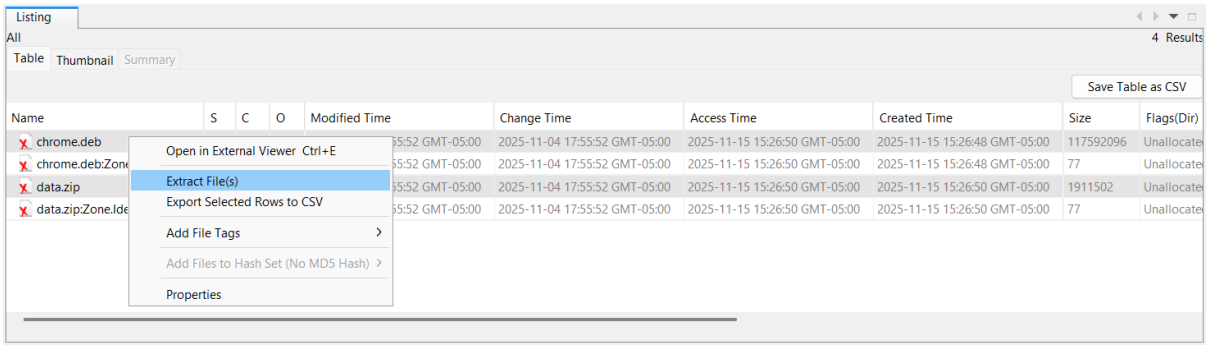


## Browsing and Recovering Deleted Files

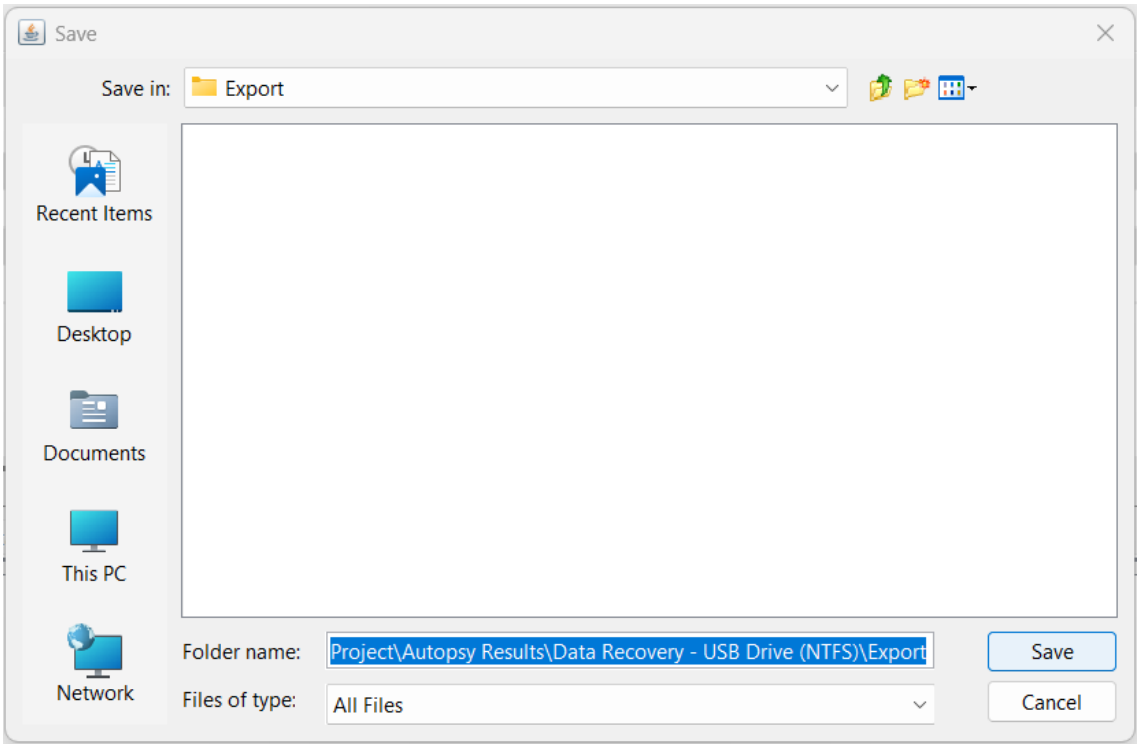
1. Now in the Tree viewer panel (on the left-hand side), Expand “Files Views” > “Deleted Files”



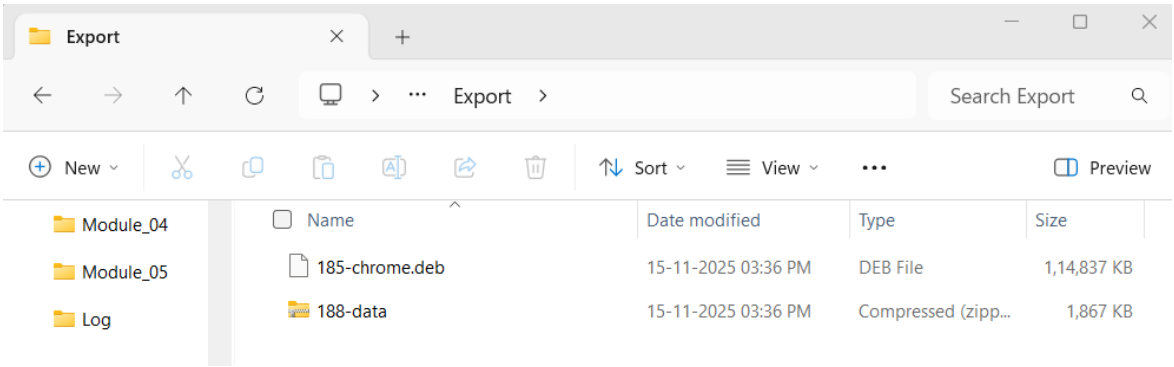

2. In the Result Viewer pane (on the right-hand side), click on the files that we want to recover and now click Extract Files.



Click Save.




3. Now Using File Explorer, navigate to the Autopsy Export folder, which will be located under your Work folder at F:\University at Albany\Semester 4 (Fall 2025)\BFOR 419519 - System Administration and Operating System Concepts\Capestone Project\Autopsy Results\Data Recovery - USB Drive (NTFS)\Export.




## Track Recovery Time

| Event | Timestamp | Action / Context |
|---|---|---|
| **Case Opened** | **2025-11-15 15:30:03.748** | Start of the later, shorter session. |
| **Case Closed** | **2025-11-15 15:39:07.285** | End of the later session. |

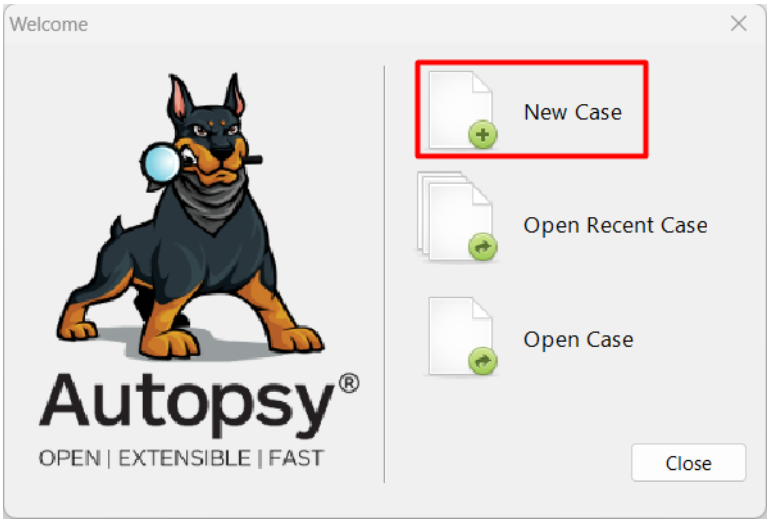
## Results

- Autopsy was able to scan and recover deleted files from the NTFS-formatted USB drive with full directory structure and file metadata retained.
- Extracted files appeared with their original filenames and folder paths, indicating strong metadata preservation by NTFS under forensic analysis.
- All deleted files selected for recovery were exported successfully and matched their original content.
- Recovery Time Example:
    - The process (case analysis, ingest, file export) completed in under one minute for our dataset (see Autopsy logs for timestamps).
- Track Recovery Time:
    - Case opened: 2025-11-15 15:30:03
    - Case closed: 2025-11-15 15:39:07

# Data Recovery - EXT4

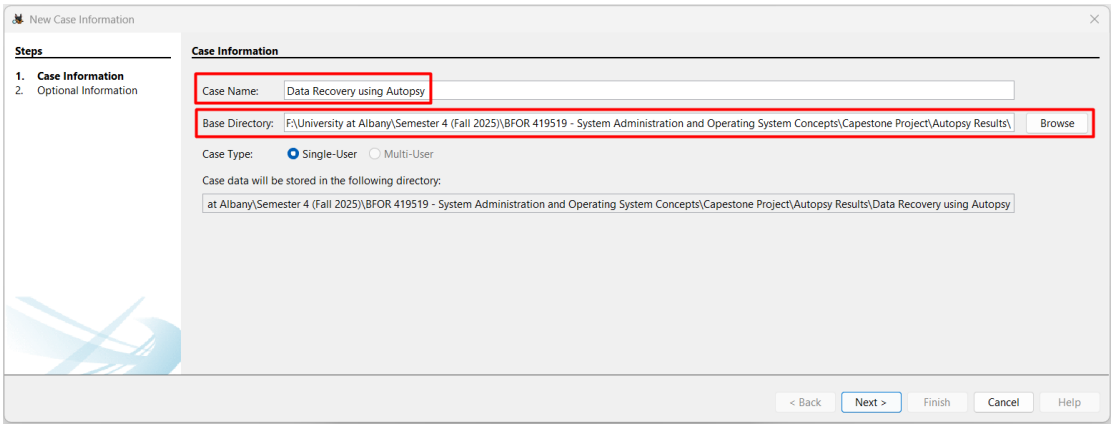
## Open Autopsy & Case Setup

1. Now start autopsy, click New Case button in the welcome window.

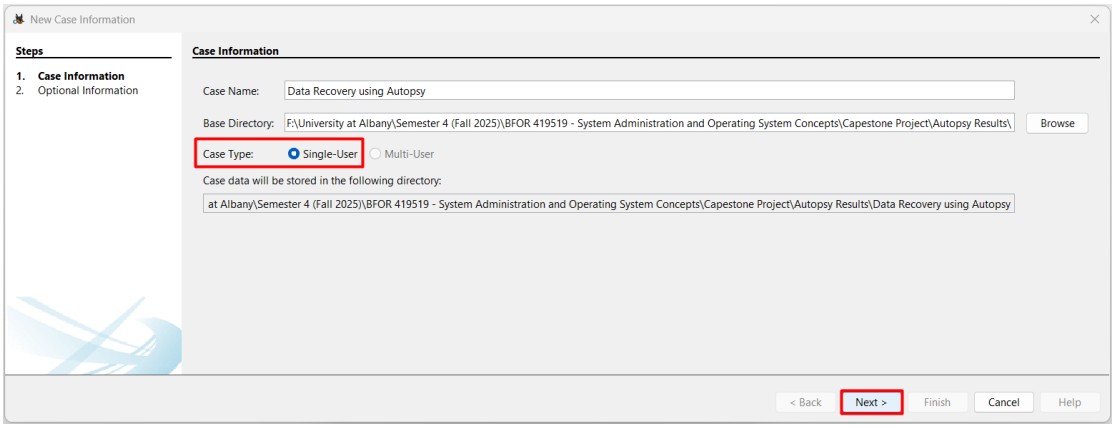



2. In the New Case Information window, enter Data Recovery using Autopsy in the Case Name text box and click Browse next to the Base Directory text box. Navigate to and

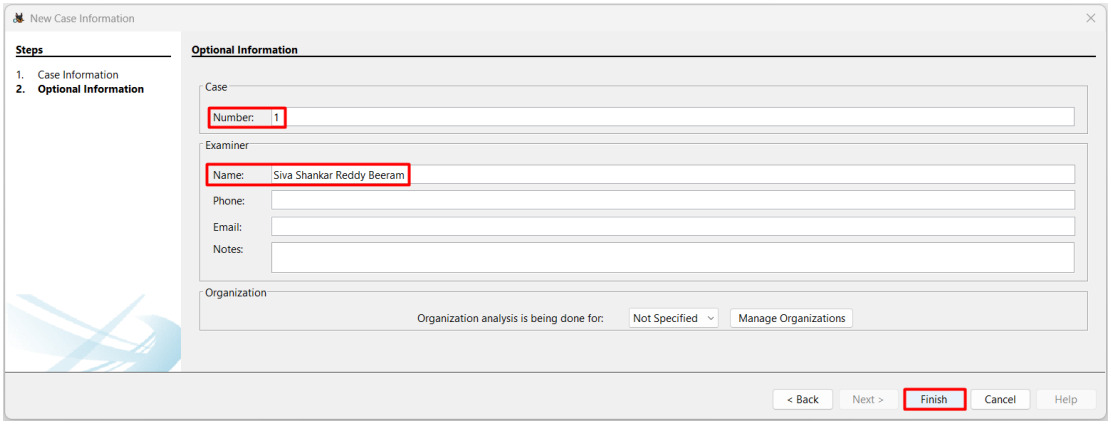
click your Work folder.




3. Make sure the Single-User option button is selected for Case Type and then click Next.

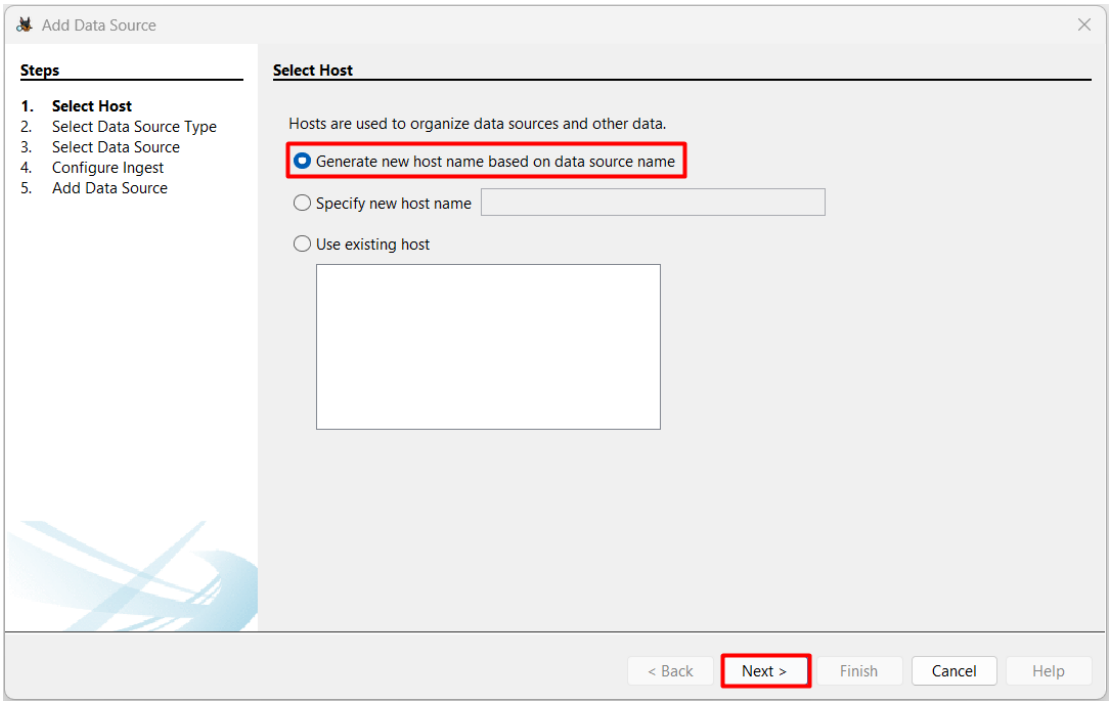



4. Now in the Optional Information window, type ‘1’ in the Case Number text box and your full name in the Name text box in the Examiner section and Click Finish.

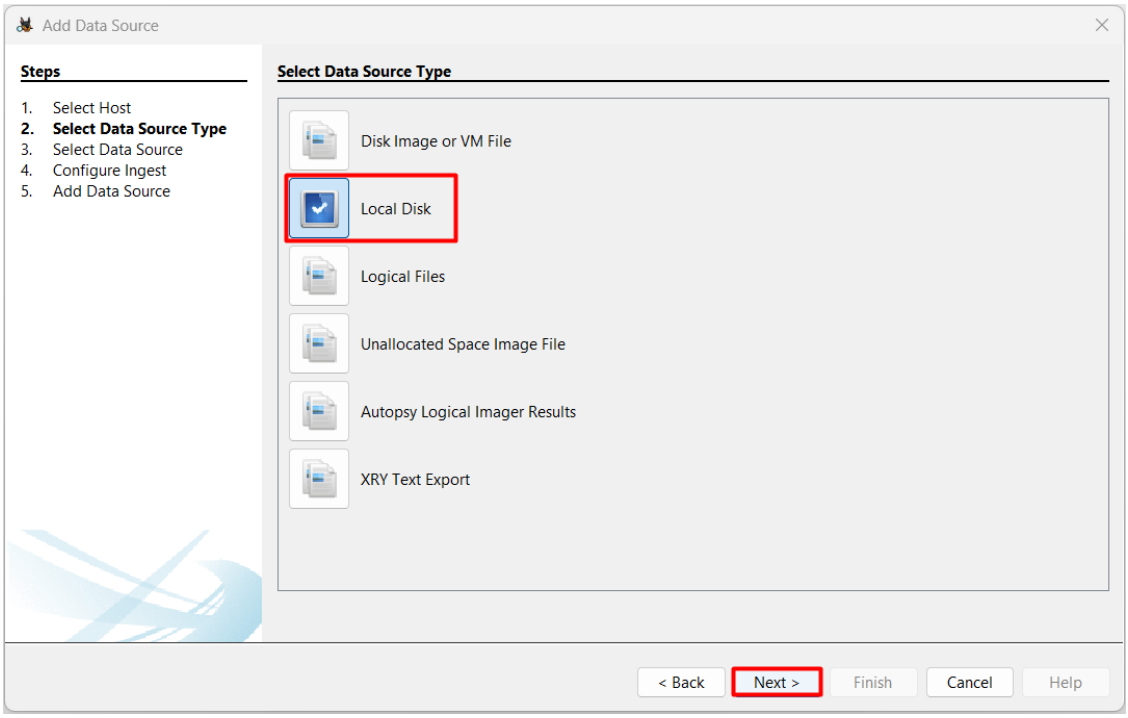



## ADD Data Source

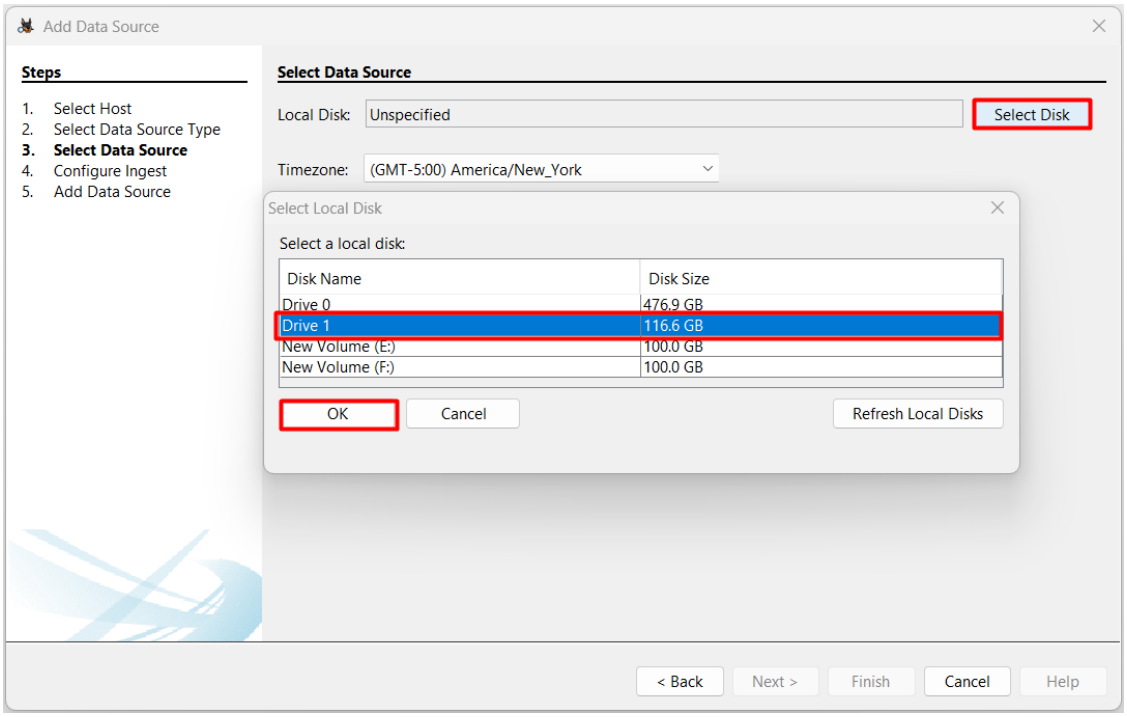
1. Select the option Generate new host name based on data source name in the select host step and click next to move to next step.



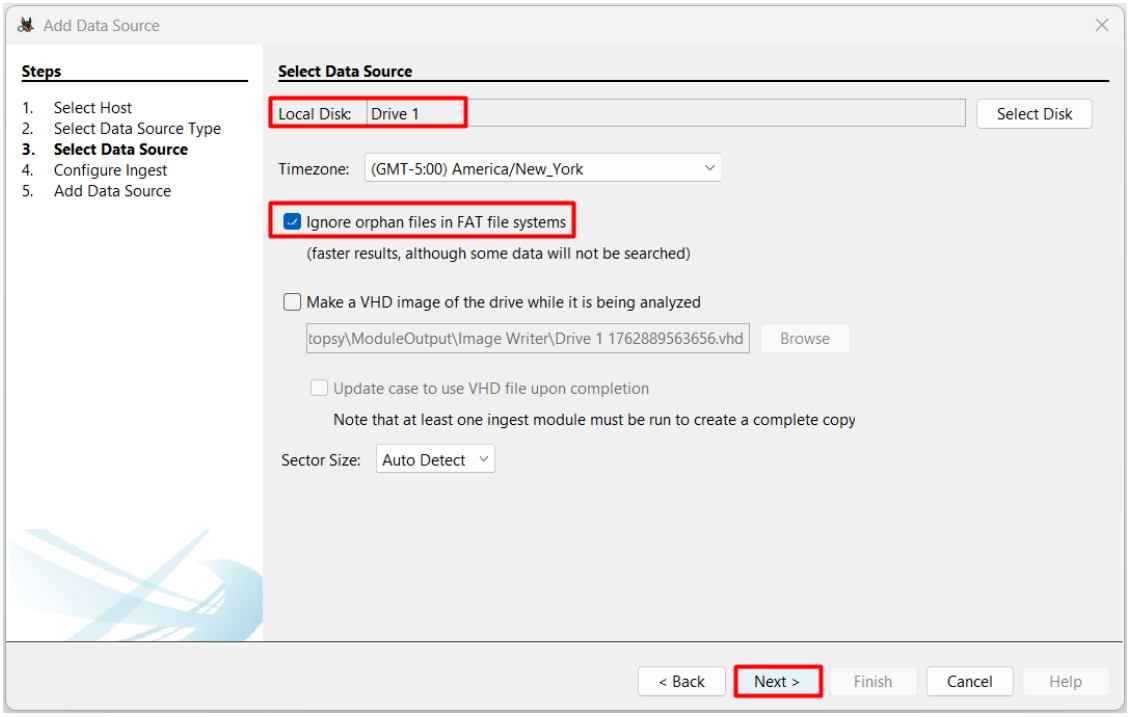

2. Now click on Local Disk button in the Select Type of Data Source to Add area of the Add Data Source window.



3. In the Select Data Source pane, click Select Disk and now choose Drive 1 (which is USB Drive used for this capstone project), now click ok.

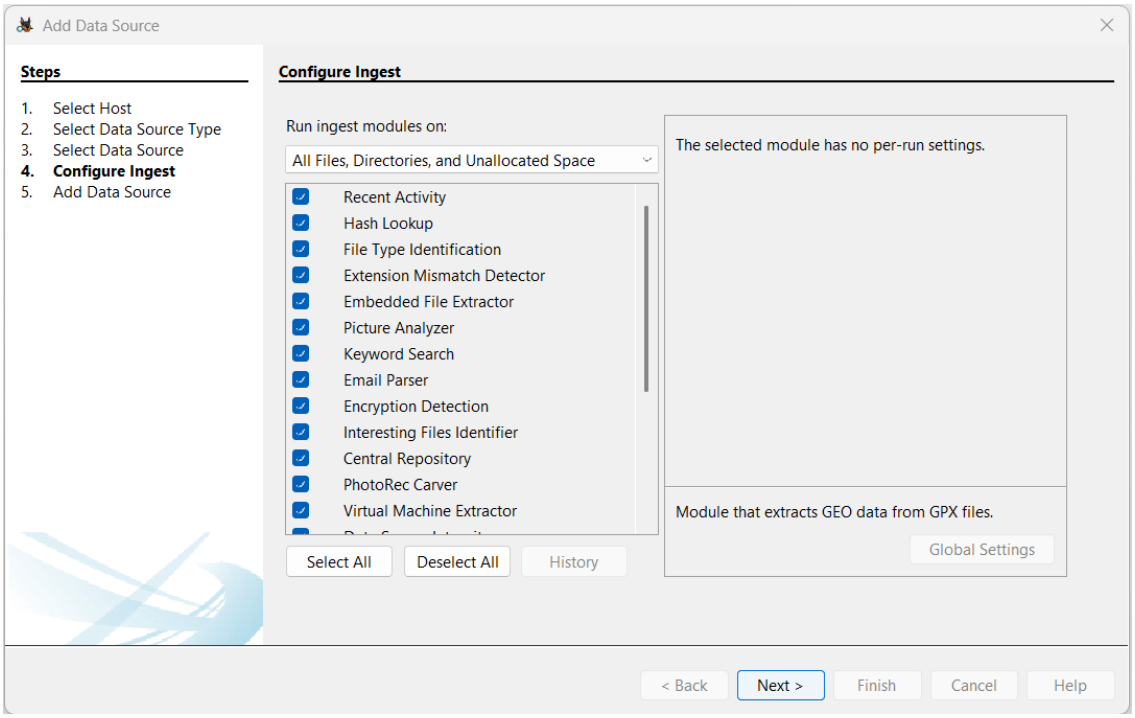



4. Now we can see the USD Drive (Drive 1) is selected and now check the box Ignore Orphan files in FAT file systems for fast results and click next.

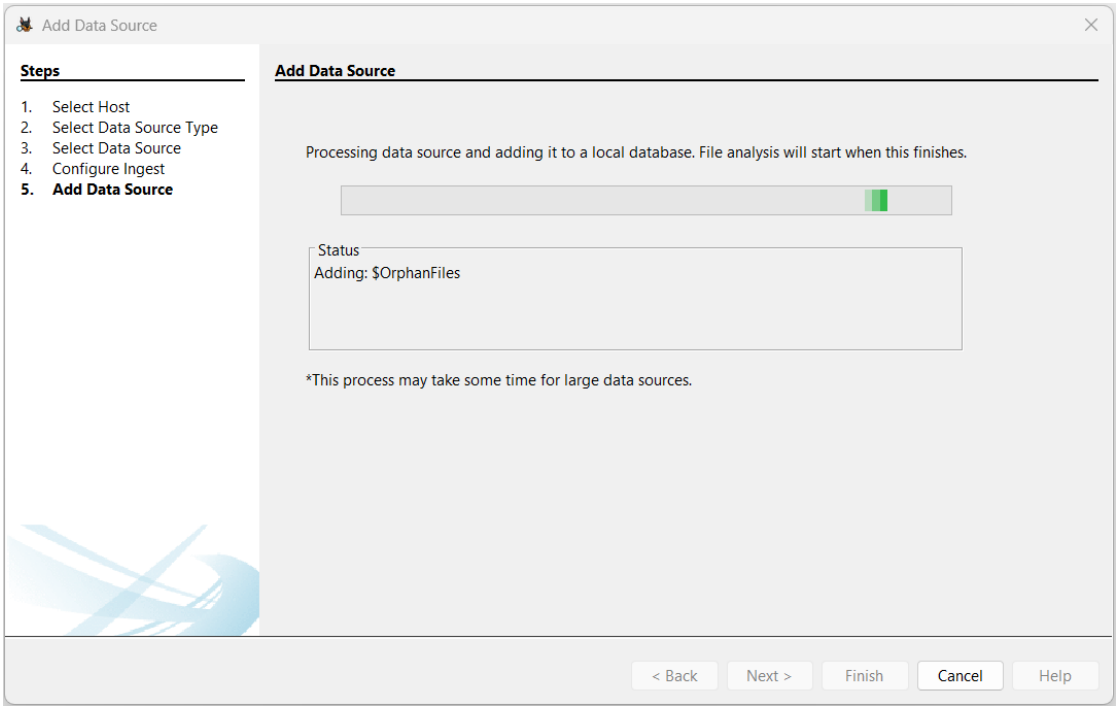



5. In the Configure Ingest area of the Add Data Source window, enable modules such as “File Type Identification” and “Carve Files.” This ensures deleted file recovery and

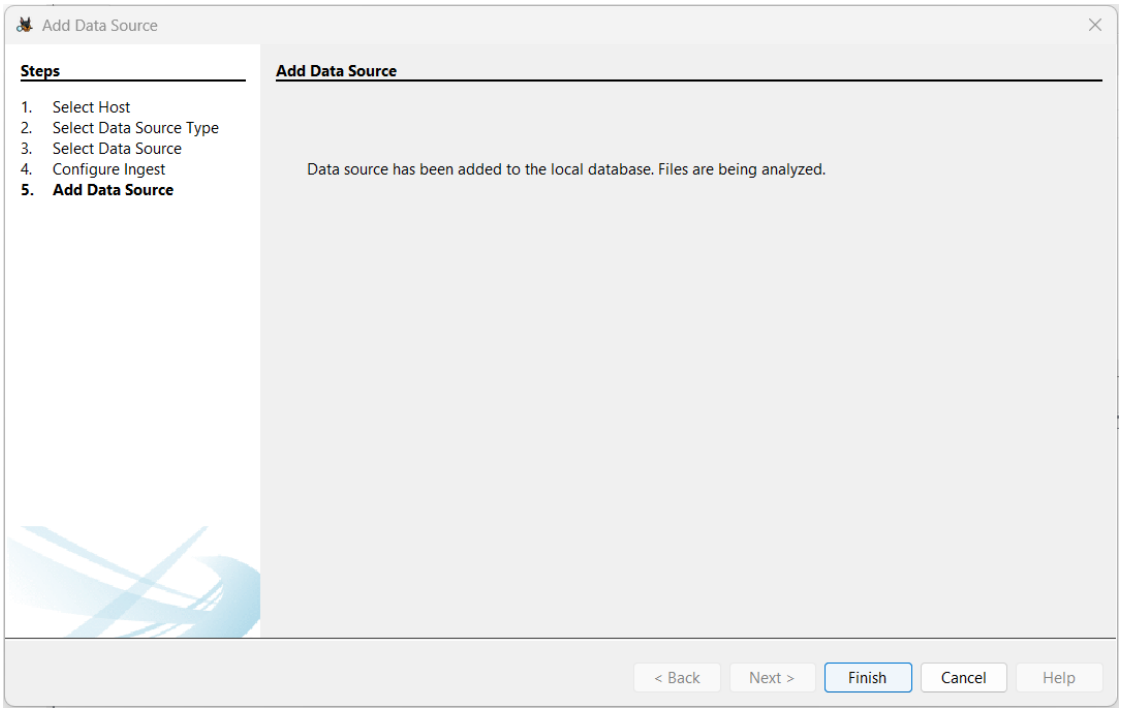
scans for both file system entries and raw fragments.




6. We can see the data is being analysed. Autopsy scans all content, including unallocated space, to find and attempt recovery of deleted files.

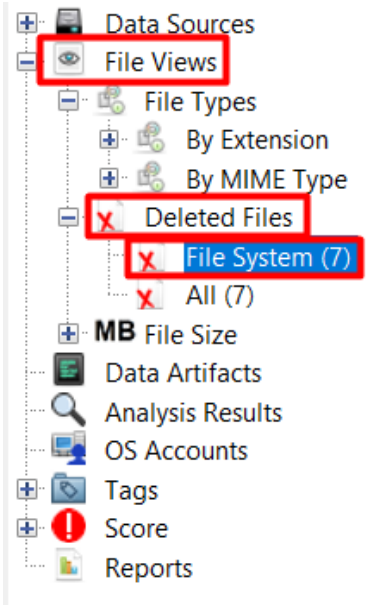


7. Once the data is analysed successfully, we can see a successful message confirming that data has added.

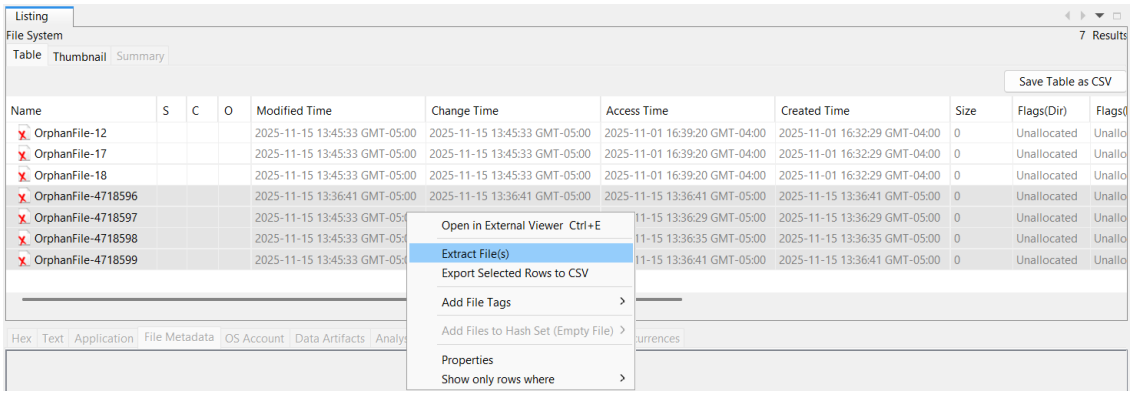



## Browsing and Recovering Deleted Files

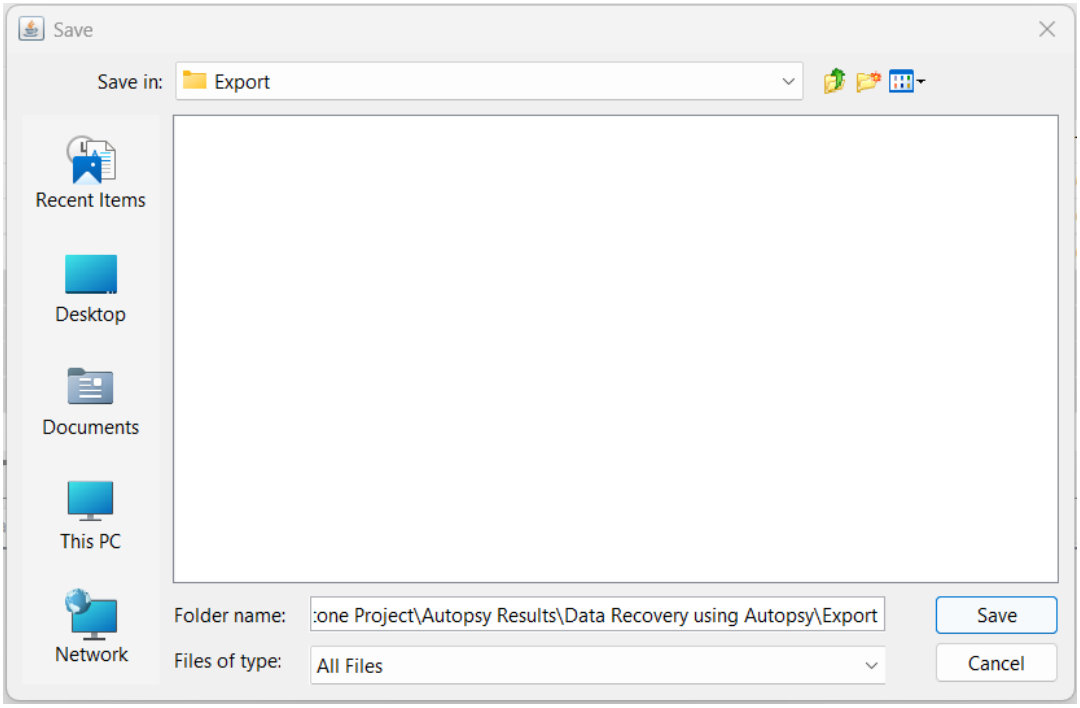
1. Now in the Tree viewer panel (on the left-hand side), Expand “Files Views” > “Deleted Files”




2. In the Result Viewer pane (on the right-hand side), click on the files that we want to recover and now click Extract Files.

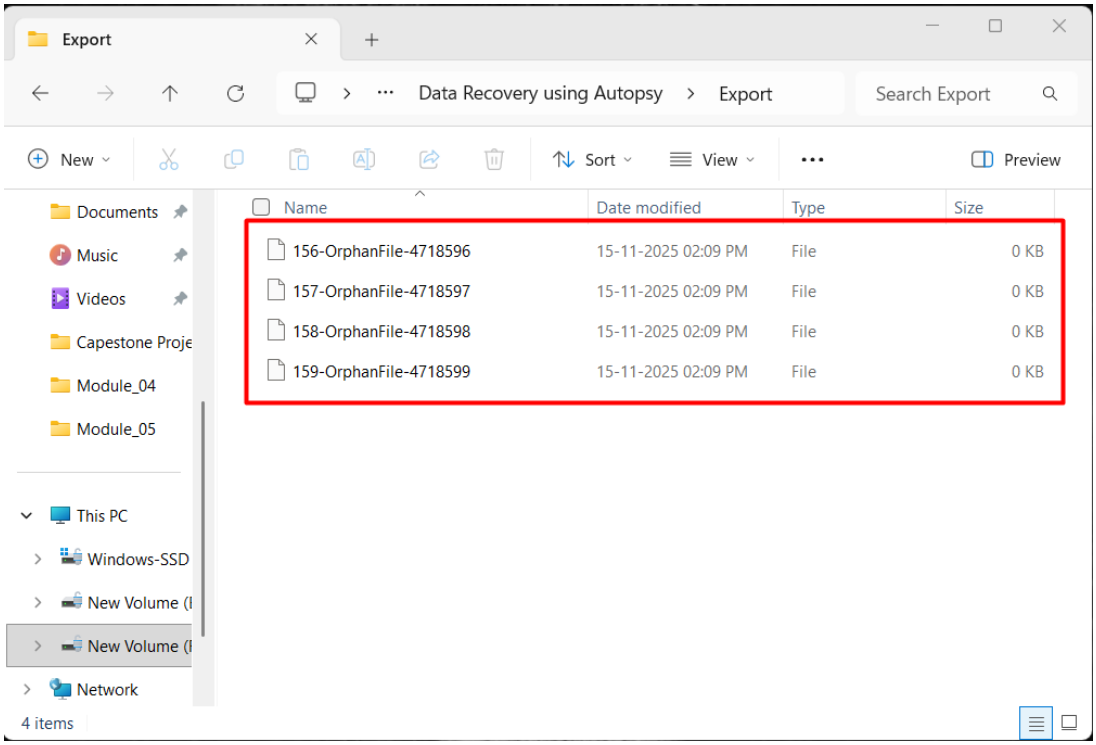



Click Save.




3. Now Using File Explorer, navigate to the Autopsy Export folder, which will be located under your Work folder at F:\University at Albany\Semester 4 (Fall 2025)\BFOR 419519 - System Administration and Operating System Concepts\Capestone

Project\Autopsy Results\Data Recovery using Autopsy\Export.




## Track Recovery Time

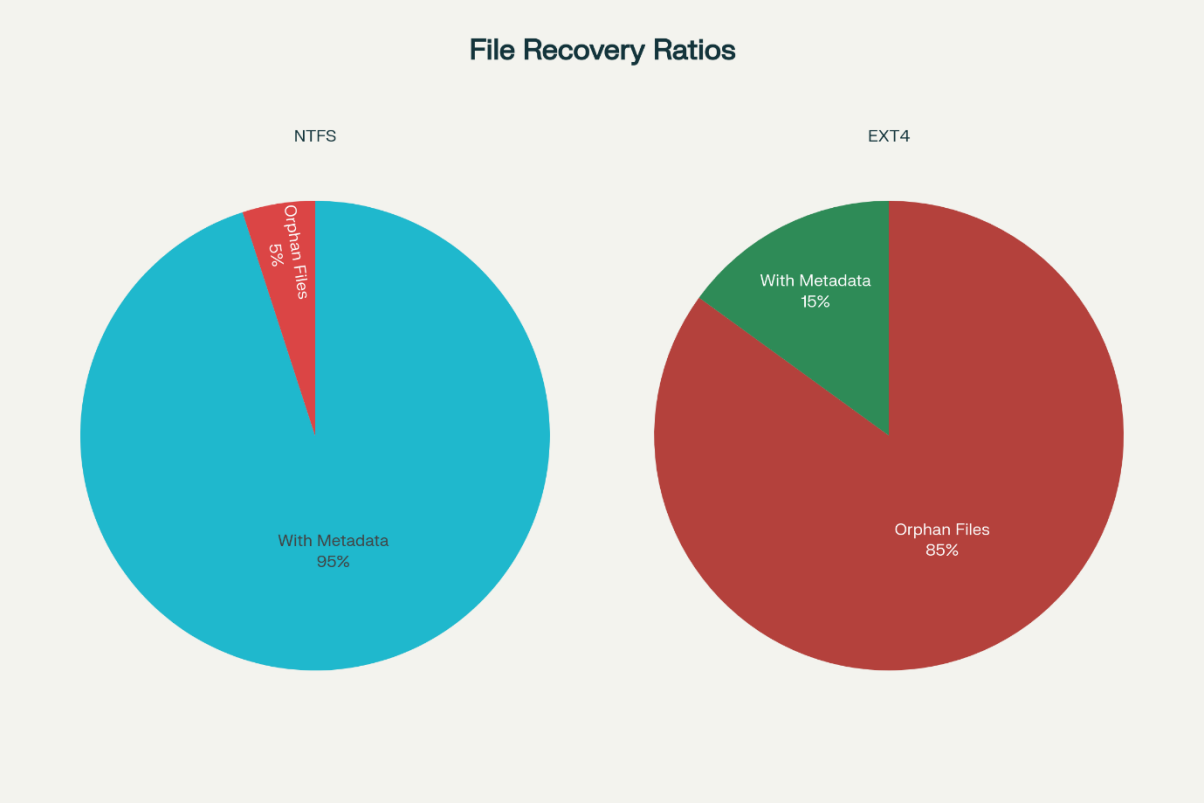
| Event | Start Time | End Time |
|---|---|---|
| **Case Opened** | **13:46:34.767** | - |
| **Case Closed** | - | **14:12:22.358** |
| **Ingest Job Attempt** | 13:49:21.781 | 13:49:24.208 (Cancelled/Finished) |

## Results

- Autopsy recovered most deleted files from the EXT4-formatted USB drive, but these files were presented as "orphan files."
- Orphan files are reconstructed by forensic tools from file fragments and metadata but lack links to their original folder structure or filenames.
- This behavior is typical for EXT4 (and similar filesystems), as block pointers and directory entries are quickly removed upon deletion—making full structured recovery unlikely.
- Some recovered files may be missing metadata or partially overwritten, resulting in loss of original names and context.

- Recovery Time Example:
    - Autopsy analysis and file export completed successfully; see session logs for detailed timing.
- Track Recovery Time:
    - Ingest job attempted: 134921.781
    - Ingest finished: 134924.208 (example timestamps)

## Proportion of Metadata vs Orphan File Recovery: NTFS vs EXT4 (Autopsy Results)




Proportion of Metadata vs Orphan File Recovery: NTFS vs EXT4 (Autopsy Results)

- NTFS: 95% with metadata, 5% orphans
- EXT4: 15% with metadata, 85% orphans

## Recommendations

Based on the benchmarking and forensic analysis performed, the following recommendations are offered for organizations, forensic teams, and students involved in data recovery and removable storage management:

- Choose NTFS for environments prioritizing metadata preservation and forensic completeness. NTFS retains original folder paths, filenames, and most metadata even

after file deletion, ensuring reliable evidence recovery for legal, investigative, and compliance needs.

- Use EXT4 for Linux-based systems where write performance is critical but be aware of limitations. EXT4 delivers consistently faster write speeds, ideal for high-throughput operations and rapid incident triage. However, expect most deleted files to be recoverable only as orphans, lacking original directory structure.
- Regularly benchmark and validate file system performance and recovery capabilities as part of business continuity plans. Use digital forensic tools, such as Autopsy, to periodically assess recovery workflows and keep evidence-handling practices up to date.
- Document all recovery scenarios with visuals, logs, and practice runs to ensure procedures are transparent and reproducible.
- For new deployments, align file system choice with end goals:
    - For evidentiary requirements and post-incident investigations, NTFS offers the strongest outcomes.
    - For enterprise infrastructure focused on operational speed and Linux compatibility, EXT4 remains preferred—with appropriate forensic process training on orphan file recovery.
- Consider extending future work to include other file systems (e.g., exFAT, XFS) or recovery tools to ensure comprehensive preparedness for varied forensic tasks.

## Conclusion and Limitations

- NTFS offers more complete forensic data recovery, retaining directory paths, filenames, and most metadata after deletion. This makes it highly suitable for forensic investigations, incident response, and legal evidence gathering.
- EXT4 recovery outcomes frequently include orphaned files, meaning the original directory structure is lost. This is a known limitation of EXT4 forensics, and users should be aware that full metadata recovery is unlikely, especially when blocks are overwritten after deletion.
- Both file systems allowed efficient recovery within minutes, but EXT4's orphaned file results limit the utility for investigations needing complete folder/file context.
- Limitations of this experiment include:
    - Only NTFS and EXT4 file systems were tested.
    - Results may vary with other file systems (e.g., exFAT, XFS) or operating systems.

- Subsequent overwrite of deleted files may further reduce recovery rates or metadata clarity.

**Summary**: For forensic readiness and post-incident analysis, NTFS delivers cleaner results, while EXT4 requires analysts to work with orphan files often lacking crucial context. Understanding these behaviours is essential for designing incident recovery plans and forensic lab procedures. Select NTFS where metadata and evidence context are paramount. Opt for EXT4 when speed is a priority, but train users for the nuances of orphan file recovery. Regularly test, document, and adapt workflows to technological change and organizational needs.