COHESITY

Version 1.2

July 2024

Data Sanitization in Cohesity Platform

Securely Deleting Data to Prevent Future Retrieval

ABSTRACT

Cohesity Platform provides a scalable, secure solution for protecting and managing enterprise data. This document describes the internal implementation of Cohesity Platform as it relates to the secure storage and deletion of data, as well as the procedures for cleaning up data spillage.

Table of Contents

Figures

Introduction

Cohesity Platform is a purpose-built solution for managing all types of data. It is designed for web-scale operation, providing users with high levels of data availability, integrity, and security.

This document describes the security aspects of Cohesity Platform as they relate to data storage, and particularly to data deletion. Security-conscious users require data storage solutions that do not allow data retrieval once it has been purged from the system. This document describes Cohesity Platform's storage architecture with a focus on providing that level of assurance to end users. The document also describes procedures for remediating data spillage.

For users interested in understanding how Cohesity's data management features impact data availability and durability specifically, see Cohesity Fault Tolerance — Data Integrity for Modern Web-Scale Environments.

Audience

This document is intended for security officers and IT system administrators working on US Government business as part of the Department of Defense. These enclaves have special security requirements, one of which is a high degree of assurance that data which is deleted from storage systems must be rendered completely irretrievable.

Solve Mass Data Fragmentation with Cohesity Platform

Cohesity Platform is a hyperconverged backup, archive, and restore platform designed for secondary data. Some of its salient characteristics are:

- Like other hyperconverged products, Cohesity Platform is deployed on industry-standard, x86-based server hardware, with each node containing compute, networking, and storage resources.
- The platform is software-defined, allowing customers to choose from a variety of different server hardware from different vendors.
- It is cluster-based, which means that a given deployment of Cohesity Platform always includes multiple server nodes, all of which work together to deliver a service. Cohesity clusters require a minimum of three nodes but can scale to an arbitrary number of nodes.
- The software follows an appliance model. Because each release of Cohesity software includes a host operating system for the platform, there is no need for administrators to install and maintain an operating system for Cohesity Platform to run on.

One key requirement for servers that run Cohesity Platform is the availability of different types of storage media. Cohesity nodes use flash media (SSD) to store metadata (and accelerate I/O operations) and spinning disk (HDD) to store protected data.

All storage I/O to the platform is via TCP/IP to a DNS round-robin hostname that cycles between virtual IP addresses owned by all cluster nodes. For any given I/O request coming from an external source, the cluster will decide which node will process the request.

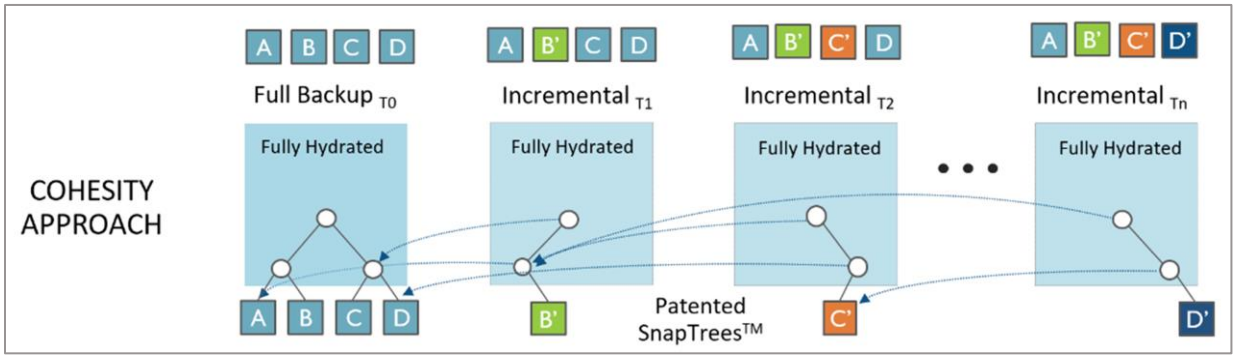
For data storage, Cohesity Platform uses a proprietary filesystem called SpanFS™ to manage data throughout the cluster. This filesystem uses the above-mentioned flash resources on each node to maintain all file metadata, as well as accelerate I/O operations on a selective basis. In addition, SpanFS is a snapshot-based filesystem, so each backup taken corresponds to a snapshot internal to the Cohesity cluster.

SpanFS, SnapTree, and Data Storage in Cohesity Clusters

As mentioned, Cohesity uses a proprietary filesystem called SpanFS for data storage. The filesystem's snapshot implementation is called SnapTree™. All backups captured by Cohesity Platform correspond to a SnapTree snapshot within SpanFS.

SnapTree leverages a modified B+ tree data structure, which has a fixed depth but can go infinitely wide.

Figure 1: Cohesity SnapTree snapshots (Each snapshot is fully hydrated, and each is exclusively accessed through its own root node.)



In figure 1 above, the first full backup contains the structural elements of a SnapTree: one root node (at the top), intermediate nodes below the root nodes, and leaf nodes at the bottom. Every backup job is linked to a snapshot, so each backup job run is accessed through a separate root node in the SnapTree. Any request for a chunk of data from any snapshot can be fulfilled with three redirections: from root to intermediate, from intermediate to leaf, and then from leaf to the chunk of data.

Figure 1 shows the original full backup, and subsequent incrementals, each of which has one changed block of data. New blocks of data are linked back to the root node via new intermediate nodes, but connections to unchanged blocks are maintained by referencing the intermediate nodes from prior snapshots. Each snapshot is fully hydrated and fully consistent. This paradigm can extend for any number of snapshots.

Each time there are references to a single chunk, an incremented reference counter is added to that chunk. This counts the total number of outstanding references to this chunk that exist in the collection of snapshots. This count is important when it comes to data aging (retention) and garbage collection. As snapshots (backups) age out of retention requirements, the root node is removed from the tree — thus making a call to that point in time impossible — and reference counters on those chunks decrement accordingly (by one).

Backup Deletion and Garbage Collection

As mentioned above, when a backup is deleted, its root node in SnapTree is deleted. Without the root node, there is no way to assemble the data chunks on disk into a consistent view of the filesystem at that point in time.

Following deletion of the root node, any data chunk referenced by that backup will have their reference counters decremented by one. Once a data chunk's reference counter reaches zero, it becomes eligible for garbage collection and space reclamation. In a Cohesity cluster, this process is managed by MapReduce, and runs at a default interval of once every four hours. It is also possible to force garbage collection manually.

Data Management Features

One of the benefits of a cluster-based implementation for data storage is that resiliency models can allow for failure of individual components of the storage infrastructure while keeping data and services online. An administrator can choose different protection schemes for data that is stored in Cohesity Platform. Currently, replication-based schemes (2x and 3x replication) and erasure coding-based schemes (2:1, 4:2, and 5:2) are supported. Each of these protection schemes involves storage of data on more than one node.

Cohesity Platform also supports the use of data-reduction technologies such as inline compression and deduplication. Deduplication is global across the cluster, meaning that a common block of data that already exists in any workflow, on any node in the cluster will not be committed to disk a second time. When employed, the deduplication database is maintained in the flash tier and is distributed globally across the cluster.

Lastly, Cohesity Platform supports cluster-wide encryption, protecting all data — in flight and at rest — with the AES-256 CBC algorithm. Encryption keys are rotated on a periodic, user-selectable basis (unless an external key management system is being used), and cryptographic destruction of all data within a cluster is supported if the need arises.

IMPORTANT: The option for cluster-wide encryption is certified for FIPS 140-2 Level 1 and must be selected at the time of installation. For DoD environments, Cohesity strongly recommends the use of cluster-wide encryption.

Benefits of Cohesity Platform for Data Spillage Cleanup

In a data spillage, data of a higher classification is written to a storage medium of a lower classification. In DoD environments, information classification is treated very seriously, and usually means that systems of different classifications are maintained on separate, air-gapped networks. When a spillage occurs, Information Assurance departments require confirmation that the data that was improperly stored has been permanently, irretrievably deleted.

The data management features of Cohesity Platform make it straightforward to clean up data spillage events. Those features have critical implications for data recoverability. Specifically:

- **Cluster-based implementation**: For an adversary seeking to exfiltrate data from the system without proper access, there is no way to know ahead of time which nodes in the cluster contain a particular file from a particular backup.
- **Storage tiering with NVMe flash and spinning disk**: For a given file in a given backup, there is no way for an adversary to know ahead of time whether that file will be located on NVMe flash media (SDD) or on spinning disk (HDD).
- **SpanFS proprietary filesystem**: Without insider knowledge of SpanFS, the data volumes attached to Cohesity cluster nodes appear to outsiders as EXT4-formatted disks with no intelligible data.
- **Cluster-based resiliency models**: Not all system data is stored on all nodes. If erasure coding is being used, no single node within the system will contain a usable portion of data for a given object.

- **Data-reduction technologies**: If deduplication is employed, files will not necessarily be written in an intelligible format. SpanFS metadata structures would be essential to understanding any data committed to disk.
- **Encryption**: When cluster-wide encryption is enabled (at installation), all data that is written anywhere is encrypted using FIPS-approved algorithms. This means that all data, data segments, and metadata is encrypted on disk.

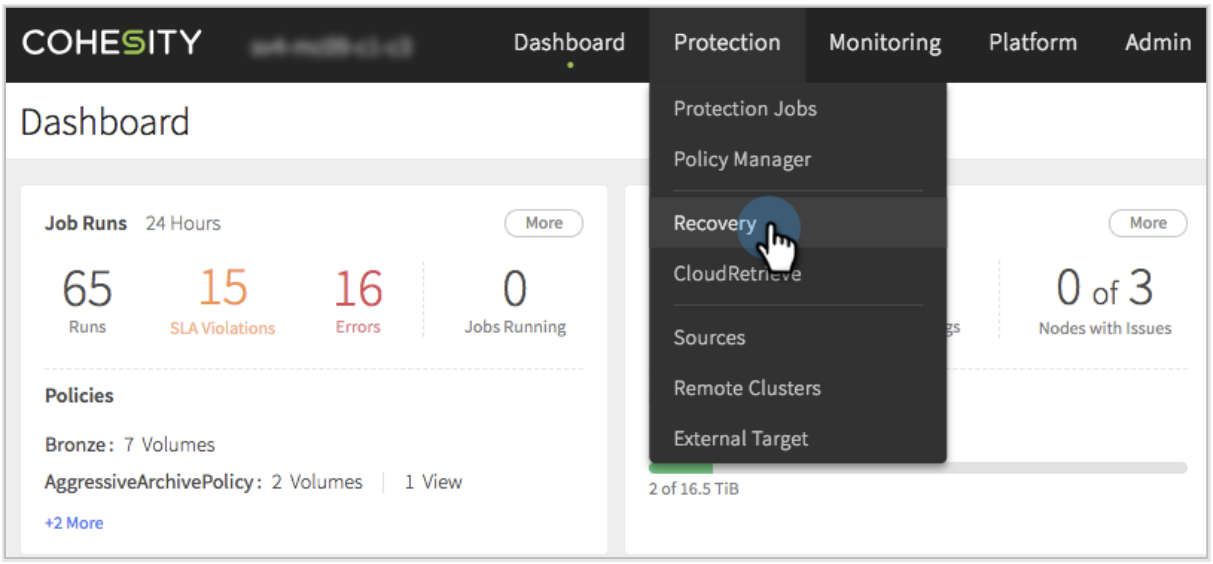
When these properties are used together, all user data written to the Cohesity cluster's physical disks is reduced to deduplicated, compressed, encrypted, and erasure-coded chunks of data. Those data chunks are dispersed throughout the system and are only rendered readable by virtue of the SnapTree snapshotting engine. As previously discussed, the only way to access data is through a SnapTree root node. When a backup is deleted, that SnapTree root node is immediately deleted, which has the immediate effect of rendering data from that point in time permanently irretrievable. Data that is no longer referenced by any other snapshots is marked for garbage collection, as described above.

Clean Up Data Spillage

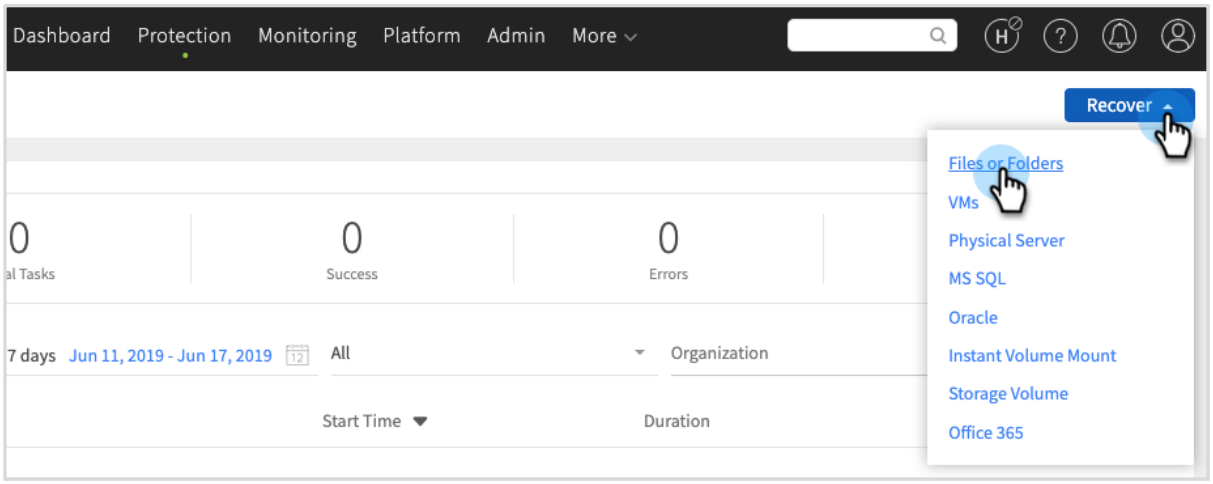
Determining whether a data spillage has occurred usually begins with the realization that a file containing classified information has been stored improperly. Once the name of the file is known, Cohesity Platform's file indexing engine assists the administrator in determining which servers contained the file.

To find out which servers contain the contraband file and then remove that file, start with the Recovery feature:

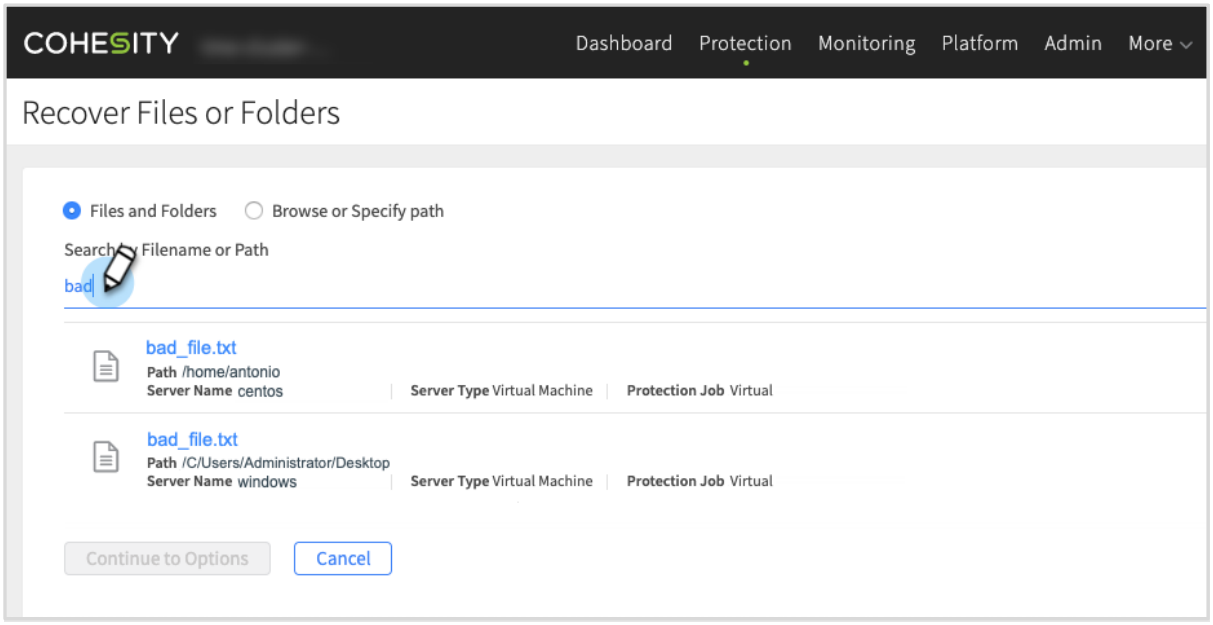
1. Log in to Cohesity Platform and select **Protection > Recovery**.



2. Click **Recover** and select **Files or Folders**.



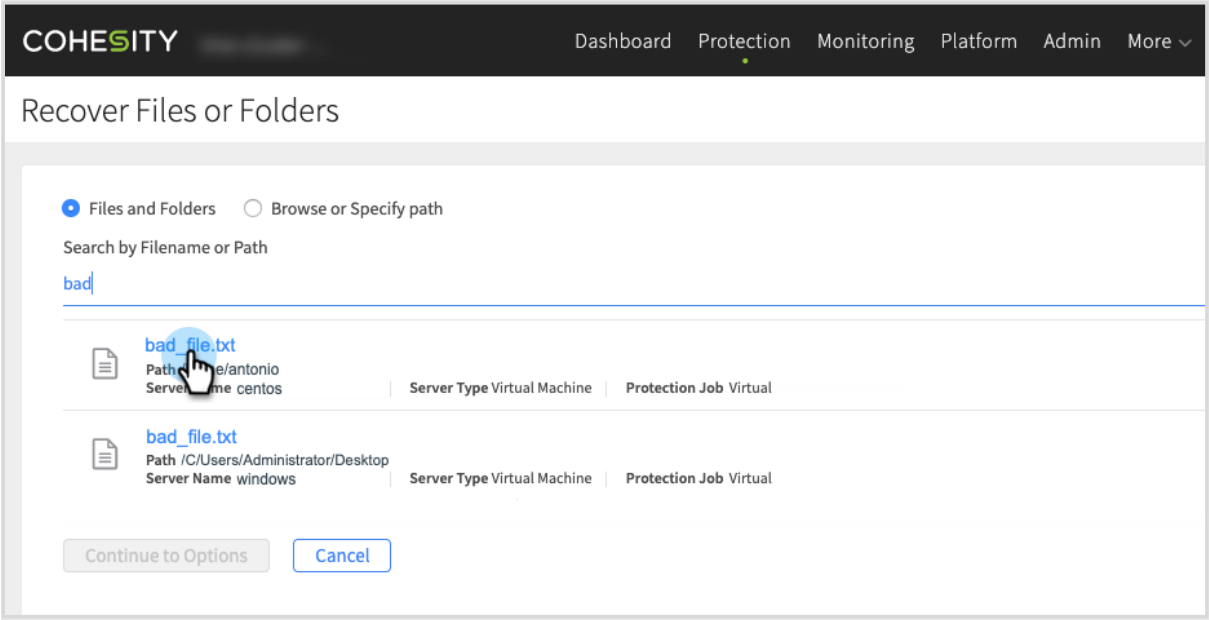
3. To retrieve a list of the files and folders in question, enter part or all of the **Filename** or **Path**. In our example, we searched for the corrupted file, bad_file.txt with the substring 'bad.' The search output shows each occurrence of the file on each server separately. In our example, we discover that the file bad_file.txt appears on two servers, centos and windows.



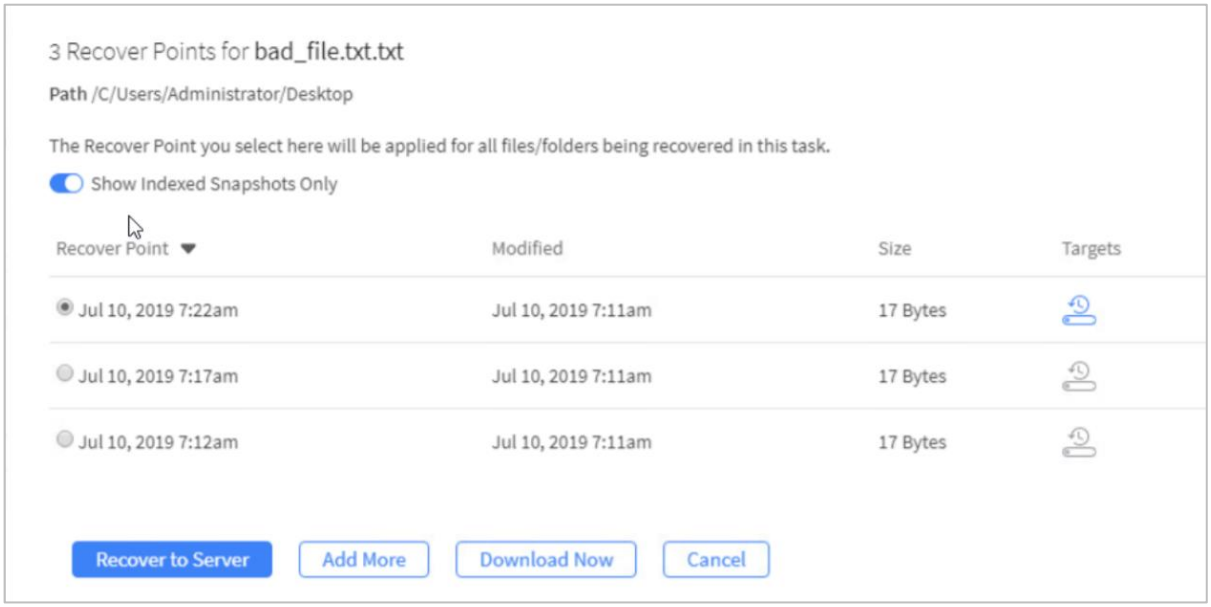
The first step the administrator should take is to delete these files from the hosts in question, and take all necessary steps to sanitize the servers, in the steps that follow.

> **IMPORTANT**: Write down the Protection Job name listed for each bad file; you'll need it in a few steps, to remove each run of the Protection Job runs ('snapshots') that contains the file.

4. To view all recovery points (that is, Protection Job runs) that contain the contraband file on that server, click the Filename in the search results.

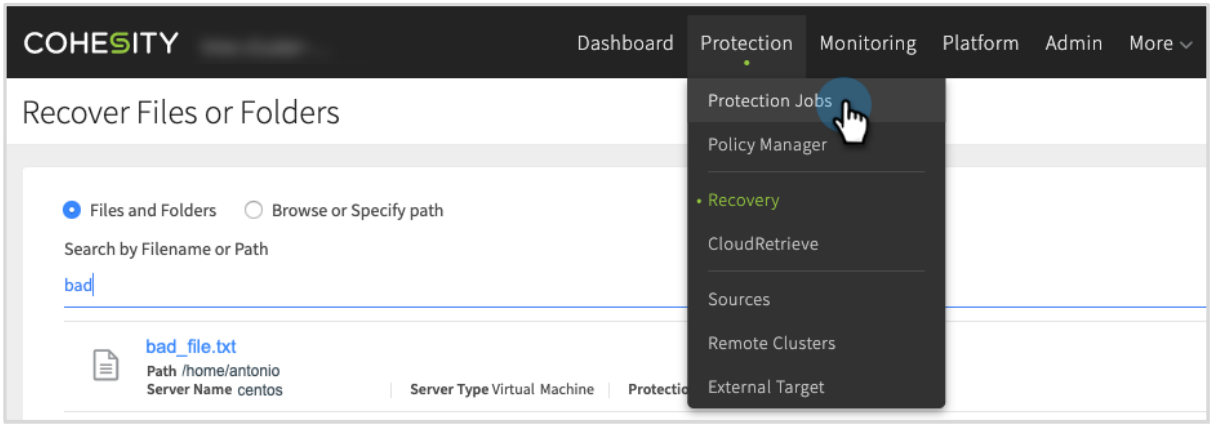


5. A dialog opens to show all the **Recover Points** that contain the contraband file.

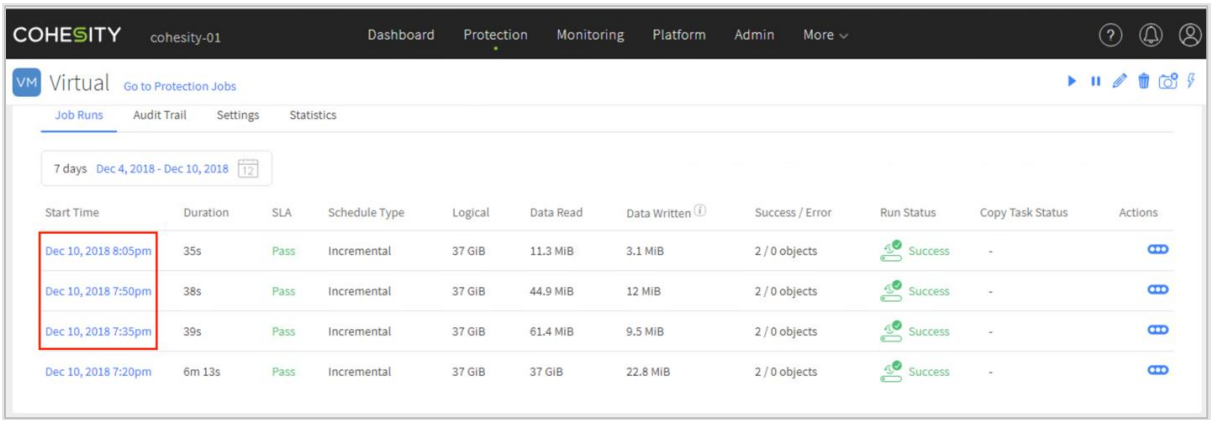


Each recovery point that contains the contraband file is a backup that needs to be deleted in order for the file to be purged from the system. Write down each recovery point that contains the file.

6. Next, select **Protection > Protection Jobs** to navigate to the Protection Job (listed in the above search) for the bad file.

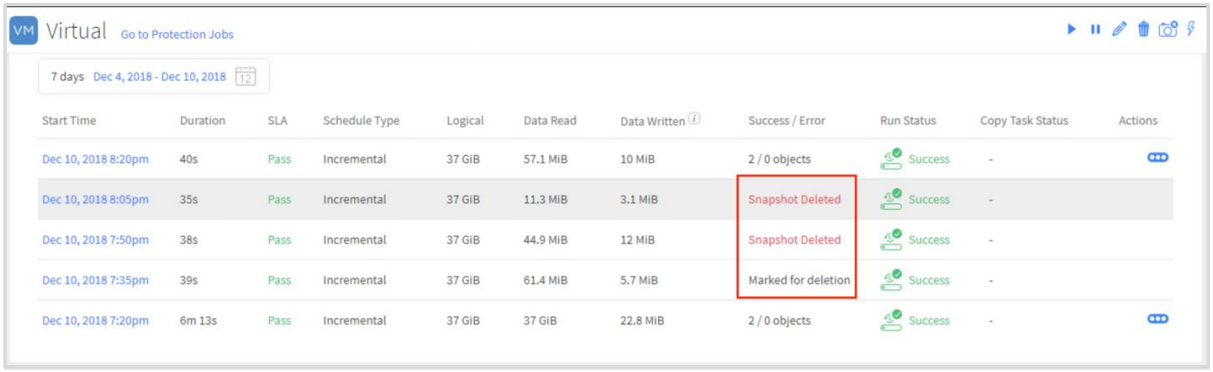


7. On the **Protection Jobs** page, scroll past the summary statistics to the list of Protection Jobs and click the Protection Job listed in the search results above.
8. The Protection Job details page opens to show all the runs of that Protection Job. In our example, the contraband file occurs in a Protection Job called **Virtual**. In the details for Virtual, we find the backups (Protection Job runs) that contain the file we need to remove.



9. Click the link for each job run for the details of that job run. From there, click **Delete Run Snapshot**. When prompted, confirm the deletion.

10. The change for that run of the Protection Job is reflected in the **Success/Error** column of the Protection Job details page. In our example, we removed three of the runs.

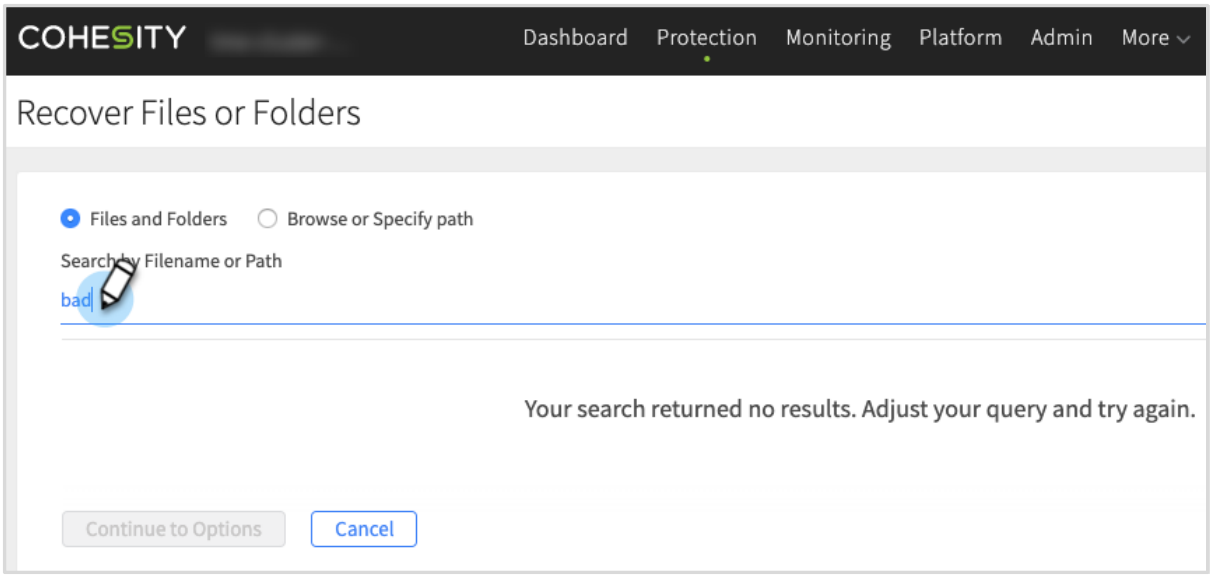


Start Time	Duration	SLA	Schedule Type	Logical	Data Read	Data Written	Success / Error	Run Status	Copy Task Status	Actions
Dec 10, 2018 8:20pm	40s	Pass	Incremental	37 GiB	57.1 MiB	10 MiB	2 / 0 objects	Success	-	
Dec 10, 2018 8:05pm	35s	Pass	Incremental	37 GiB	11.3 MiB	3.1 MiB	Snapshot Deleted	Success	-	
Dec 10, 2018 7:50pm	38s	Pass	Incremental	37 GiB	44.9 MiB	12 MiB	Snapshot Deleted	Success	-	
Dec 10, 2018 7:35pm	39s	Pass	Incremental	37 GiB	61.4 MiB	5.7 MiB	Marked for deletion	Success	-	
Dec 10, 2018 7:20pm	6m 13s	Pass	Incremental	37 GiB	37 GiB	22.8 MiB	2 / 0 objects	Success	-	

> **NOTE**: Snapshot (backup run) deletion is an asynchronous process. If you have deleted a run that is still queued for deletion, it will show "**Marked for deletion**." Once it is completely removed, it shows "**Snapshot Deleted**."

11. Navigate back to the **Protection > Recovery > Recover Files or Folders** search page. Once all snapshots containing the contraband file have been scheduled for deletion, you will see the following screen:

12. Once all snapshots have been deleted from the system (that is, no longer showing the "**Marked for deletion**" status), the file recovery search will present a "no results" page.



You did it! This confirms that the system has been purged of all instances of the contraband file.

Your Feedback

Was this document helpful? Send us your feedback!

About the Authors

Tony Ibanez is a Senior Systems Engineer for Cohesity Federal. In his role, Tony focuses on data resiliency, data center architectures, and storage systems.

Other essential contributors include:

- Bart Abicht, Sr. Technology Writer and Editor

Document Version History

VERSION	DATE	DOCUMENT HISTORY
1.2	July 2024	Republishing
1.1	Jan 2022	Rebranding updates
1.0	Jun 2019	First release

ABOUT COHESITY

Cohesity is a leader in AI-powered data security and management. Aided by an extensive ecosystem of partners, Cohesity makes it easier to protect, manage, and get value from data – across the data center, edge, and cloud. Cohesity helps organizations defend against cybersecurity threats with comprehensive data security and management capabilities, including immutable backup snapshots, AI-based threat detection, monitoring for malicious behavior, and rapid recovery at scale. Cohesity solutions are delivered as a service, self-managed, or provided by a Cohesity-powered partner. Cohesity is headquartered in San Jose, CA, and is trusted by the world's largest enterprises, including six of the Fortune 10 and 44 of the Fortune 100.

Visit our website and blog, follow us on Twitter and LinkedIn and like us on Facebook.