COHESITY

Protect MongoDB with Cohesity

Cohesity Solution for Backup and Restore of MongoDB Databases with Best Practices

ABSTRACT

Get an overview of Cohesity Database Protection features, general workflows, options, and the best practices for MongoDB.

Table of Contents

Figures

MongoDB Protection using Cohesity

MongoDB administrators contend with the challenges of Increasing backup duration, rapid data growth, increased storage costs, and the lack of flexibility and storage management tools in MongoDB Studio.

In addition to these challenges, today's database protection must encompass more than getting a clean copy. It must include cybersecurity, storage efficiency, and minimizing the impact on production systems, automation, and scaling.

Cohesity protection for MongoDB provides a solution to these challenges. It reduces the complexity of database backups and restores with secure, streamlined workflows. You can protect and manage your workloads and execute available protection and recovery workflows with a single pane of glass with a few steps. Cohesity MongoDB protection provides flexible deployment options to make MongoDB backups and restores simple and secure.

This guide provides an overview of features and options and their related recommendations and best practices.

Use Cases

You can use this workflow for MongoDB protection:

- If you are looking for automated backup protection.
- If you are looking for protection for your MongoDB environment using an on-prem Cohesity cluster.
- If you want a simple, specific restore or a Point in Time restore.
- If you want a simple UI-based database restored to an alternate host.
- If you want to meet your backup SLA.
- If you want faster and more secure backups, restore performance using gRPC.
- If you want to move away from script creation and management.
- If you want automated storage configuration and management.
- If you want to reduce storage space and the cost of your backups.
- If you are looking for centralized monitoring and reporting.
- If you are looking for immutable, ransomware-proof backups.

Technical Considerations

Consider the following technical aspects before you make major decisions about your solution.

- No SSH or NFS mounts are needed, and gRPC is not used.
  We leverage MongoDB client that has a built-in data transport. The communication between MongoDB is direct.
- You can perform any point-in-time restores of a database.
- The first backup is always a FULL backup. After a Full backup, the incremental backup goes faster.
- Ensure third-party agents are removed or uninstalled.
- You can protect multiple databases using a single Protection Group.
- Verify that all collections have a unique value for _id. The field name _id is reserved for use as a document's primary key, and its value must be unique in the collection.
- The user specified in the MongoDB source registration page must have clusterAdmin and readWriteAnyDatabase roles.
- Refer to Plan and Prepare for MongoDB Protection Considerations for other considerations. Ensure that you are familiar with the different options available.

> **Warning**: A collection with more than one document with the same ID can cause data corruption as we cannot uniquely identify the documents. Even with data corruption, backup/restore will succeed without any errors.
>
> Refer to the following MongoDB documentation for more information:
>
> - Field Names
> - Unique Indexes

MongoDB Key Concepts

Replica Set

A Replica set consists of multiple copies of the same data, with one primary and multiple secondary nodes. Its processes maintain the same data set across multiple MongoDB servers. It is the simplest form of a cluster.

- The primary node receives all write operations (and reads)
- Secondary nodes replicate operations from the primary to maintain an identical data set
- Secondary nodes only receive Reads

MongoDB Cluster

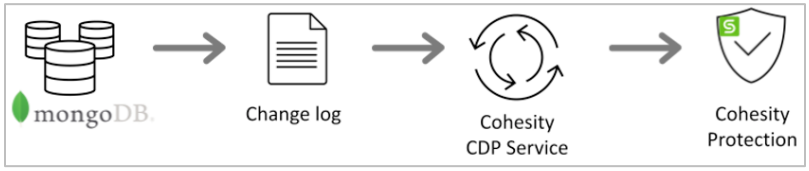
A MongoDB cluster is a collection of Replica sets spread across multiple servers. Its main goal is to improve performance, availability, and scalability.

Sharding

Sharding involves splitting data across multiple MongoDB clusters. Each cluster has primary and secondary nodes. The clusters are managed by S-nodes, which direct data read and write operations. For other aspects of sharding, please refer to MongoDB and Sharding.

Continuous Data Protection (CDP)

Cohesity's MongoDB CDP service continuously tracks log updates. The CDP service records changes to the Cohesity DataProtect platform and creates a log backup, allowing users to recover from any point on that timeline.



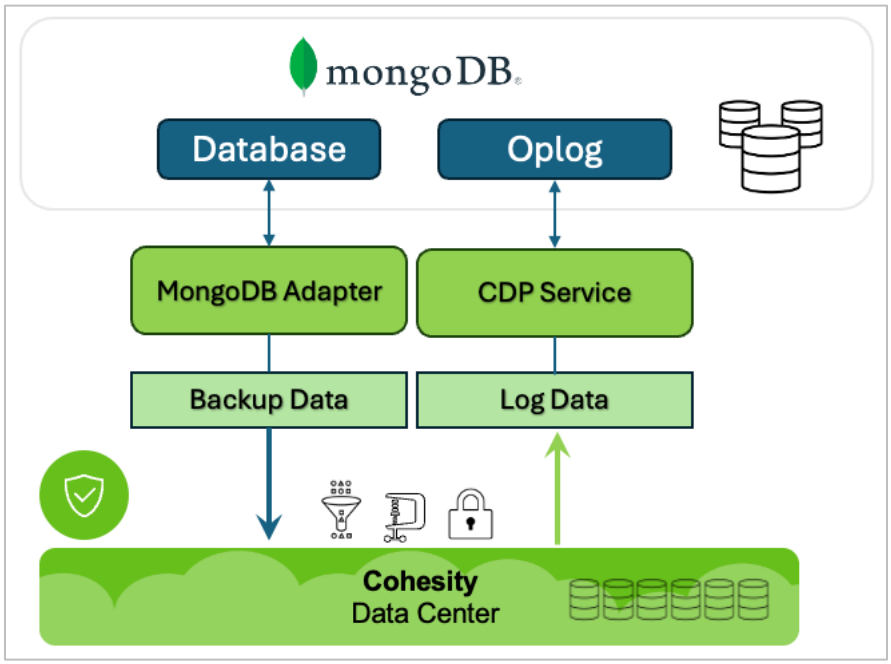
Ops Logs

CDP only works on MongoDB replica sets or sharded clusters. It doesn't work on MongoDB standalone servers because it lacks oplogs.

Important: A stand-alone MongoDB server must be converted to a sharded cluster to provide CDP services. See Convert a Stand-Alone MongoDB Server to a Replica Set.

If CDP is not enabled on the MongoDB cluster, only a backup point can be restored. A point-in-time restore cannot be performed.

Figure 1: Oplog Configuration



How Cohesity Works with MongoDB Databases

The Hadoop Service performs backups and recoveries. The CDP Services continuously captures changes to the database.

The MongoDB Adapter allows Cohesity to connect directly to MongoDB databases. Backups are then transferred to the Cohesity cluster. Cohesity offers the choice to perform either full or incremental backups. The initial backup is always full, and subsequent backups can be either full or incremental based on specific business needs.

Cohesity Features

Backup and Restore Features for MongoDB

You can find MongoDB Backup and Restore features at MongoDB Protection.

Feature	Linux
Full Backup	Yes
Incremental Backup	
Log Backup	
Restore Snapshot Same Host	
Restore Point In Time Same Host	
Restore Point in Time Alternate Host	
Policy or on-demand backups	
Support replica sets and sharded clusters.	
Granular backup and recovery - backup and recover individual collections	
Index definitions are backed up and restored.	
Option to back up from secondary MongoDB nodes	
Handle topology changes seamlessly (node addition, removal, IP changes, etc.)	
Support Kerberos, LDAP, and SCRAM-authenticated clusters.	
Support SSL-enabled clusters.	

Feature	Linux
Support Continuous Data Protection (CDP).	

Supported Versions

Platform	Supported Versions
MongoDB	7.0, 6.0, 5.0, 4.4

Deployment Steps

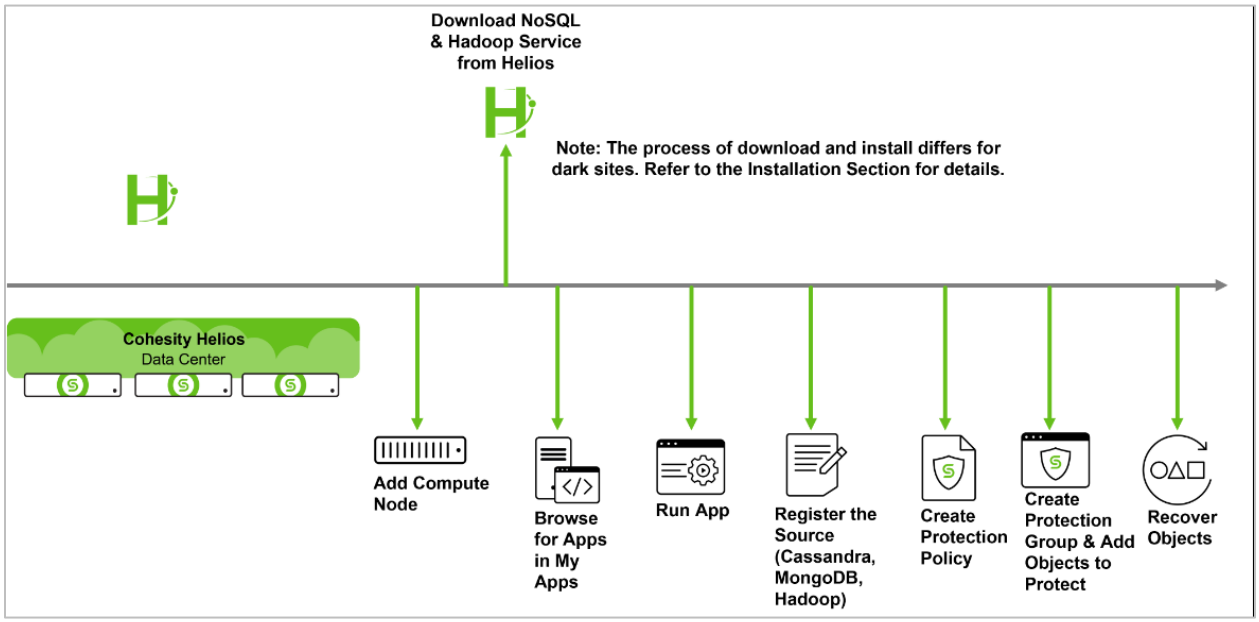
MongoDB does not require an adaptor. It is completely agentless.

MongoDB requires compute node resources on the Cohesity cluster to run NoSQL & Hadoop services.

NOTE: There are two ways to provide the resources required for the NoSQL & Hadoop services: first, a compute node is recommended, or second, resources can be pooled from the existing data nodes.

There can be more than one CDP service to handle the workload. The number of CDP services is determined as part of the customer's sizing analysis.

Figure 2: MongoDB Setup



Once you have installed the compute node and the Hadoop service on the Cohesity cluster, you can register the database as a source.

Configuring Continuous Data Protection (CDP)

Activating Continuous Data Protection (CDP) is important to ensure optimal MongoDB backups and restores. This feature needs to be set up and configured correctly. Follow the steps below to get started.

- Ensure that the NoSQL and Hadoop Service app is installed and running.
- To enable the MongoDB CDP service, use the following command:

  ```
  $ imanis_cli.sh -c mongocdp enable --cpu <cpu-cores> --memory <memory-GB> --replica <num-replicas>
  ```

  - **cpu-cores:** Number of CPU cores allocated to each MongoCDP service instance
  - **memory-GB:** Amount of memory allocated to each MongoCDP service instance
  - **num-replicas:** Number of MongoCDP service instances. Depending on the load on MongoDB primary and the recovery point objective, one or more MongoDB CDP services can be provisioned. Using multiple smaller CDP services can help RPO in case nodes go down.
- **Sizing the Primary Oplog of MongoDB Primary**
  Ensure that MongoDB primary's oplog is sized to hold at least 30 minutes of data, even when a heavy load exists. This means that MongoDB primary would need a larger Oplog if it has a higher maximum ingestion rate.
- **Syncing Time Between MongoDB Cluster and Cohesity Cluster**
  Ensure that the time on MongoDB cluster and Cohesity cluster is in sync. This is a mandatory requirement as NoSQL and Hadoop Services access MongoDB Oplogs for incremental backups.

Cohesity Source Registration for MongoDB

You must first register your database as a source on Cohesity, as shown in the screenshots below.

Best Practices - Registering Sharded Clusters: With sharded clusters, you have Mongo

S-nodes. In this case, we don't connect to each shard directly; we go through the MongoDB S-nodes. So you're registering all the MongoDB S-nodes with the port the Mongo S service is on. So, it would be a comma-separated list of Mongo S-nodes when you register a sharded MongoDB cluster.

Whether primary or secondary, a change to any shards does not require modifications to the Cohesity registration.

Recommendation: Use a fully qualified domain name or IP address of the node and the port MongoDB is listening on.

Best Practices - Non-Sharded Replica Set: A replica set has one primary node and one or more secondary nodes. When registering a replica set, you must include a comma-separated list of the primary nodes and all secondary nodes belonging to that replica set cluster.

A change to the replica set means you must edit the comma-separated list to reflect the changes.

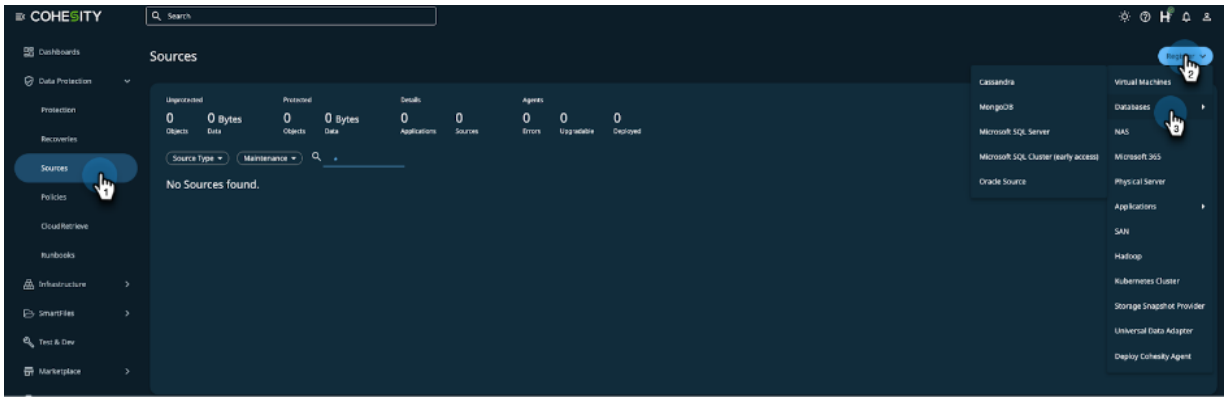
Recommendation: Use a fully qualified domain name or IP address of the node and the port MongoDB is listening on.

Register The MongoDB As A Source

To protect your databases with Cohesity, register them as a Cohesity source. Once they're registered in Cohesity, you can add them to a Protection Group and configure the settings for your databases.

To register your database as a Source in Cohesity:

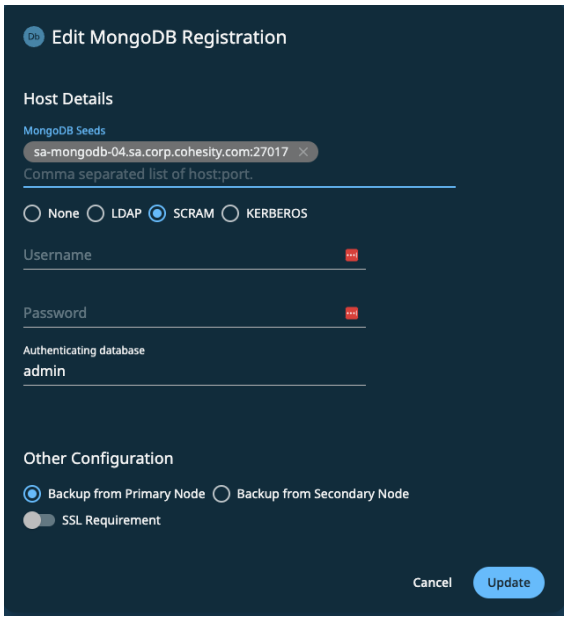
1. Navigate to **Sources** > **Register** > **Databases**.



The **MongoDB Registration** form will guide you through the registration steps.

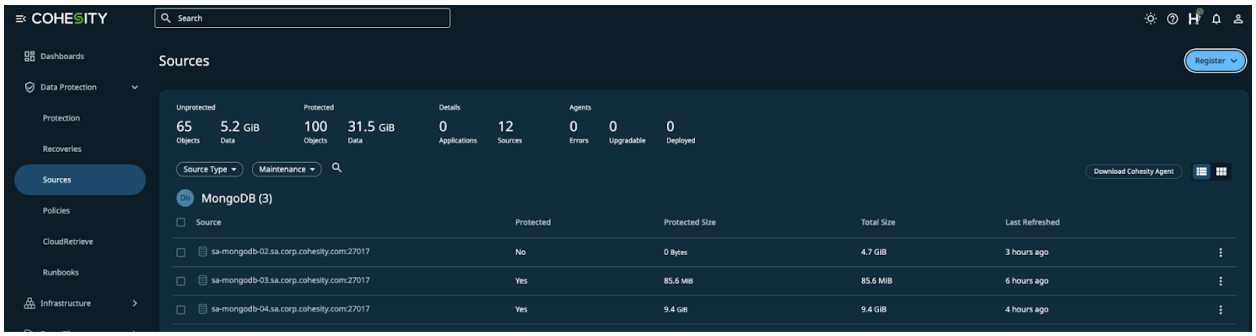
2. In the **Edit MongoDB Registration** form, enter the MongoDB cluster's primary and secondary replica members. You can use either the IP addresses or FQDN.

> **Remember:** use the S-nodes if it is a sharded cluster.



3. A completed form will automatically expand with more options. Complete your source registration by clicking **Register.**

Successful Registration



More information about registering your MongoDB host can be found at Register and Manage the MongoDB Source.

To protect your newly registered Source, you'll create a Cohesity Protection Group for it in the next chapter.

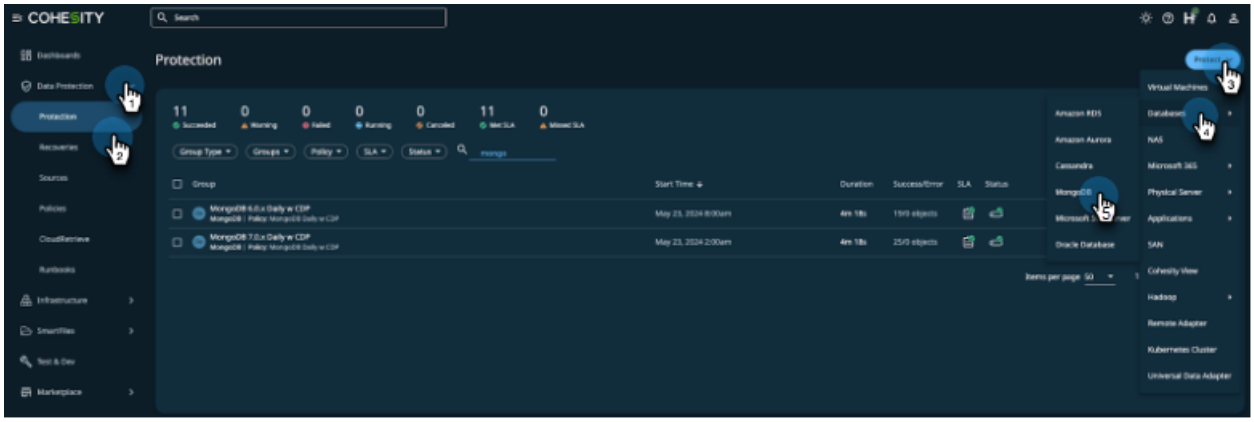
MongoDB Database Protection

After registering the MongoDB host as a source on Cohesity, you can start configuring your backups.

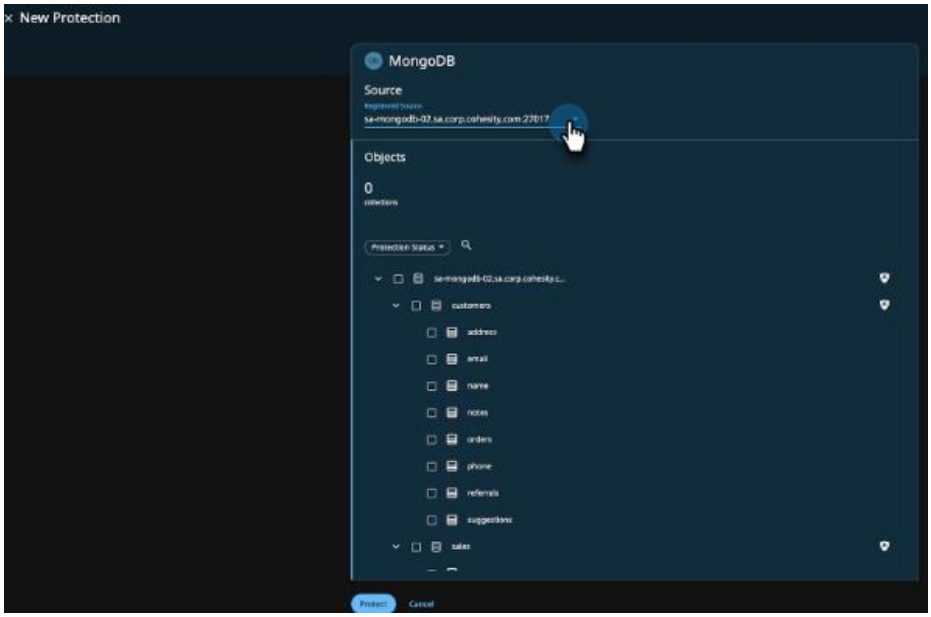
Create a Protection Group

To create a Protection Group:

1. Log in to Cohesity and navigate to **Data Protection > Protection**. Then click **Protect** and select **Databases > MongoDB**.



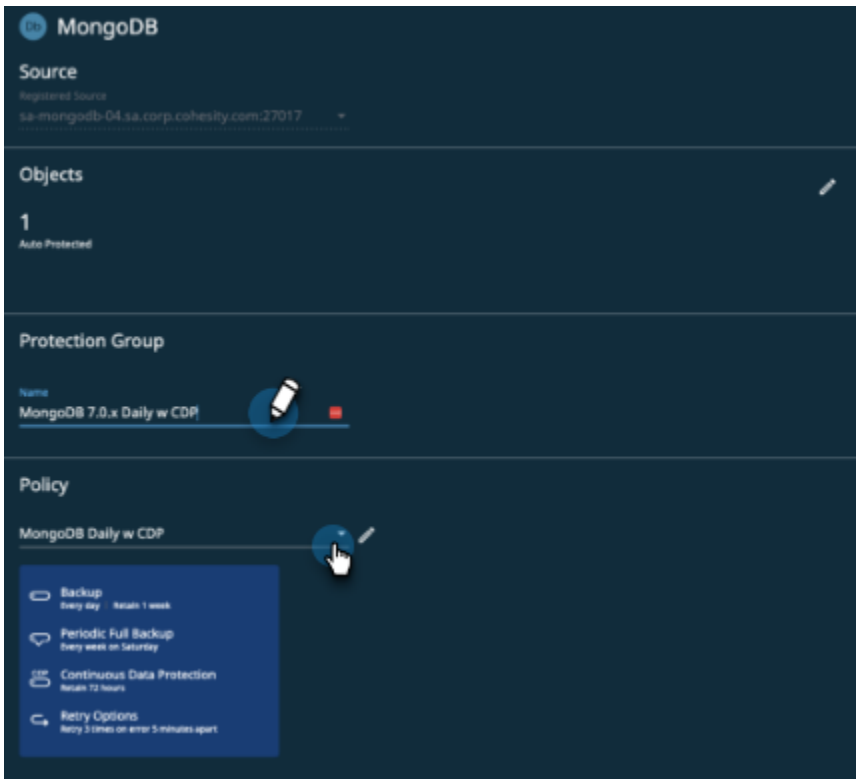
2. In the **New Protection** form, under **Source**, select the Source you registered earlier.
3. Select the collections you want to protect.



Auto-Protect - you can selectively choose collections.

> **Best Practice—Auto-Protect:** Auto-protect can be assigned at the cluster, database, or collection levels. It protects the objects at the assigned level and all the objects underneath. Auto-protect detects an object's addition and includes it the next time a backup is performed.
>
> Auto-protect is used for new objects that are introduced either on the cluster level or the database level.
>
> If Auto-protect is not assigned at the cluster or database level, you must individually choose which objects to protect by simply putting a checkmark next to that object.

New Protection Group Form



4. In this form, enter a unique Protection Group **Name.**

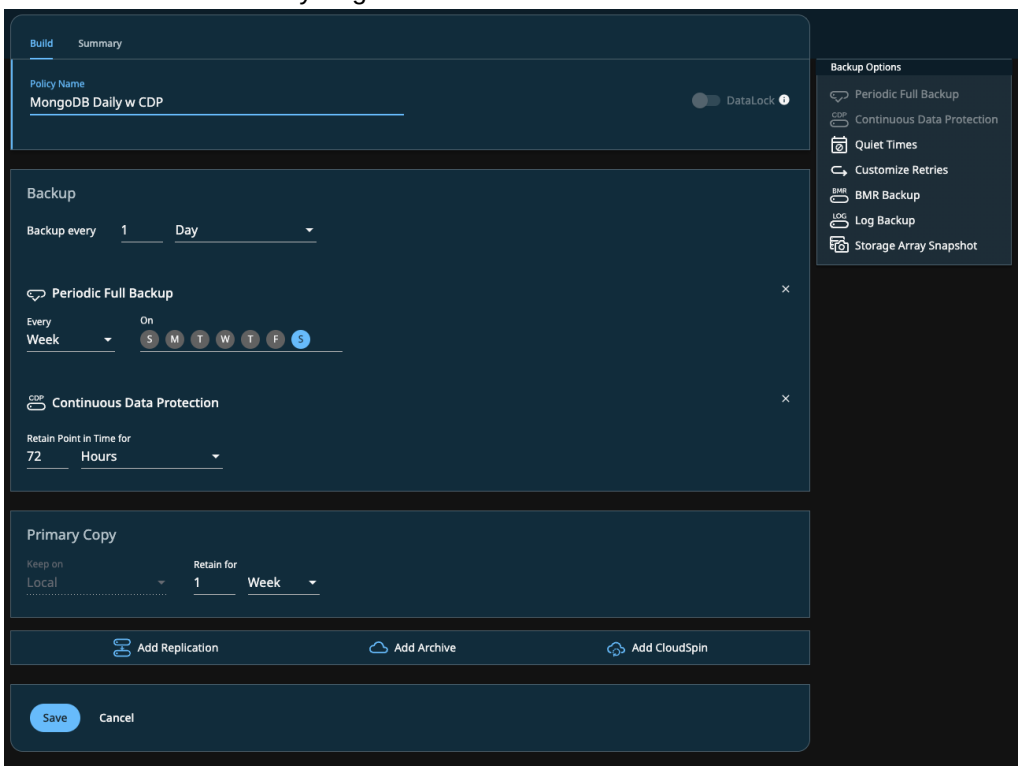
> **Important:** We can back up key spaces. However, special considerations must be made when restoring a Key Space to an alternate cluster.

5. Continue by selecting a **Policy.**
   You can use the default standard policies or create your own. Policies save time because they reduce the effort required to enter settings repetitively.

   A policy is a reusable set of settings that define how and when objects are protected, replicated, or archived. You select which policy to use when configuring a Protection Group.

> **Important:** The Policy defaults to a MongoDB Incremental backup. You must add a Periodic Full Backup. In this example, a FULL backup is scheduled every Saturday, followed by Incremental backups on M, W, and F. Without the Full backup, you cannot restore the database. The Policy should not contain a log backup since we capture the transaction logs with CDP.

Create a Protection Policy Page



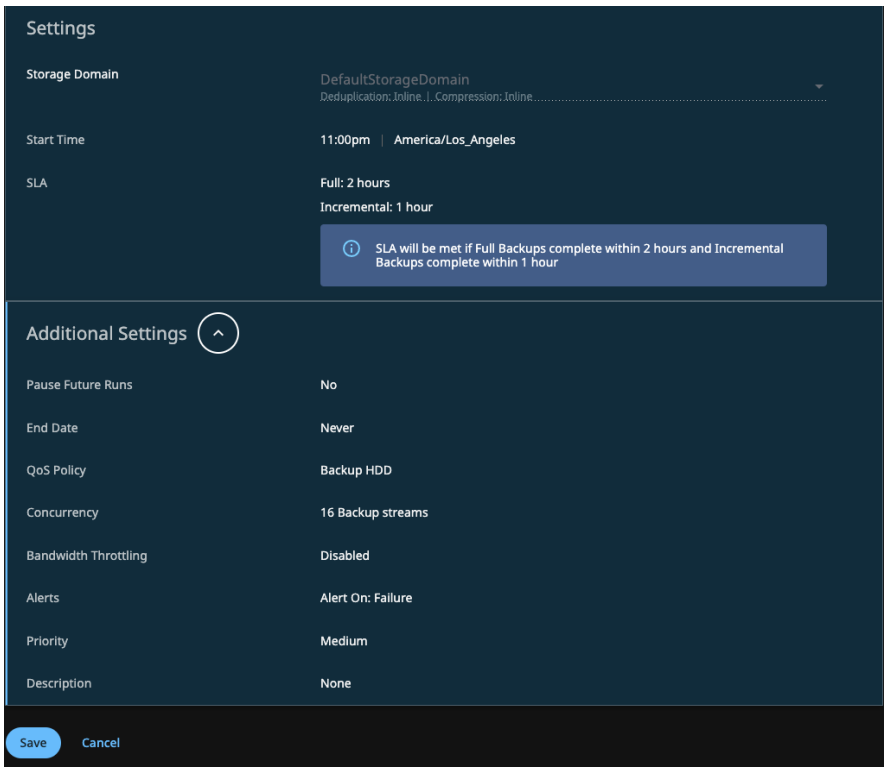
Continuous Data Protection: specify the retain point in time configuration.

Retention for Backups

DBAs maintain a combination of backups to *restore* a database at any point in time. A good combination of backups consists of FULL, Incremental, and CDP backups.

For more information about Policy features, see Create or Edit a Standard Policy in the online Help.

Continue defining the **New Protection Group** by selecting the remaining settings.



6. **Storage Domain:** For maximum space savings and security, choose a Storage Domain with compression, deduplication, and encryption enabled. For details, see Create or Edit Storage Domains in the online Help.
7. **Start Time**: Take the default.
8. Concurrency:

> **Best Practice—Concurrency**: The recommendation is to stick with the defaults 16 for backups and 8 for recoveries. Performance will depend on the number of data nodes and compute nodes in the Cohesity cluster and also on the resources of the source cluster.
>
> Take time to measure performance before changing the default settings for concurrency.

After you have completed the settings, if you need to change any additional settings on the New Universal Data Adapter Protection Group page, scroll down and click Edit on the right.

Your new Protection Group is active and running and appears on the **Protection** page.

Field Notes

Q: What happens to the protection job if there is an outage in the MongoDB cluster?

A: If a secondary node is chosen for the backup and goes down, the backup will fail because, in Cohesity, there is no automatic failover to a different secondary node. Change the source registration to a different secondary node to backup to correct this.

If a primary node is chosen for the backup and it goes down, Cohesity detects a new primary node (a secondary node that was promoted to primary) and protects the new primary node without fail. There is no need to modify the source registration.

If it is a sharded cluster and the S-nodes were registered, then any change to the cluster's topography does not require a change in the registration.

MongoDB Database Restore

Cohesity allows you to restore individual databases or collections to their original or alternate locations.

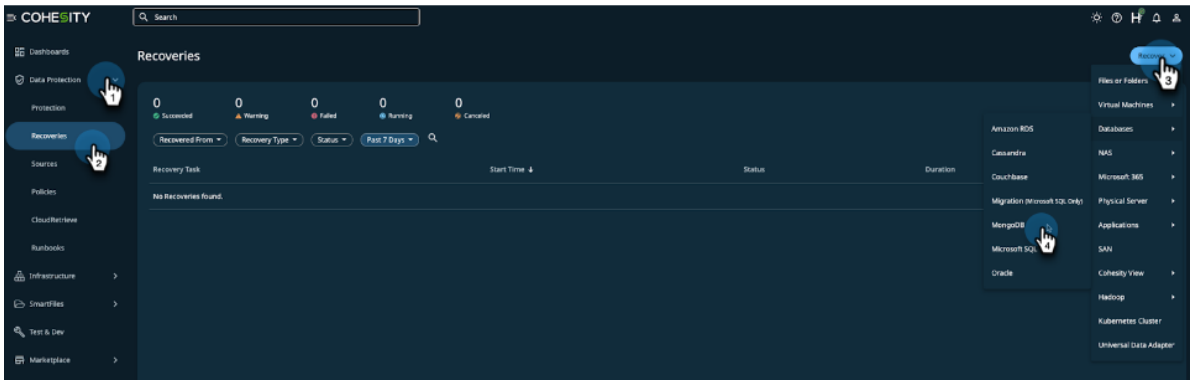
Restores are not assured between different versions of MongoDB. The admin must ensure sufficient storage space on the target cluster to restore. MongoDB recommends periodic full backups, and we follow this recommendation.

Cohesity creates a database if one does not exist when restoring a collection.

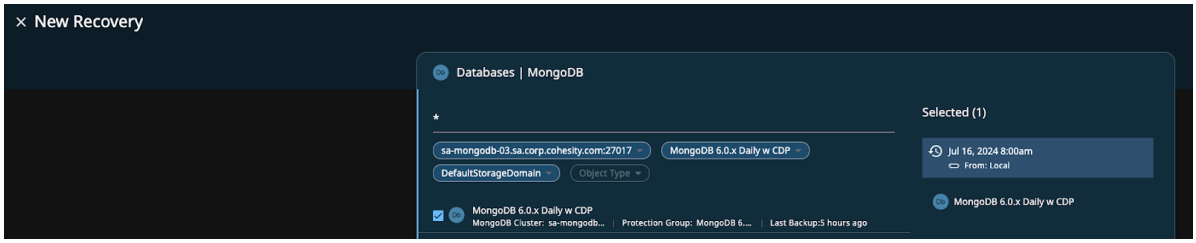
Restore Specific Backup

To restore the database:

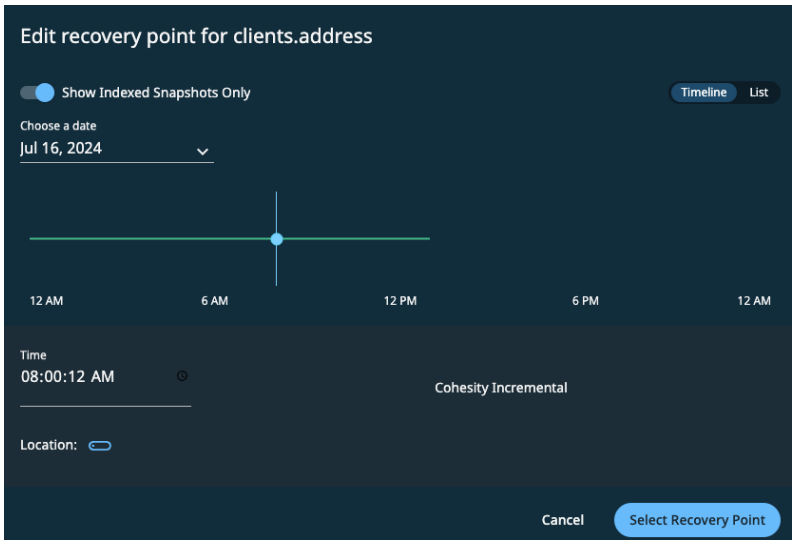
1. Log in to Cohesity, navigate to **Data Protection > Recoveries,** and click **Recover > MongoDB**.



2. Search for the backup. You can start with the wildcard "*****" to get a general listing.



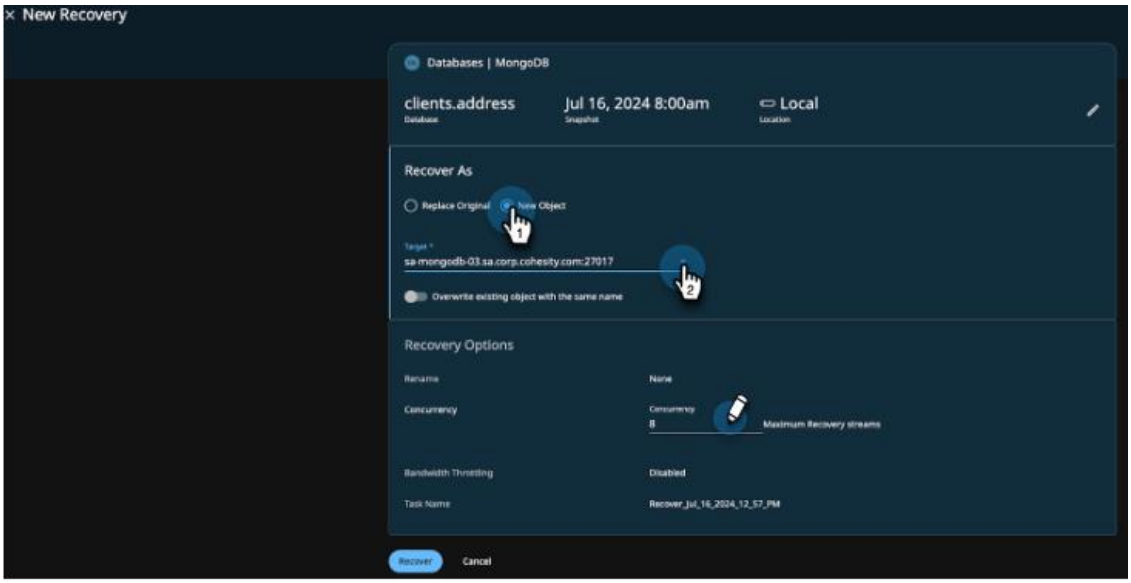
3. Using the **Edit Recovery Point** form, choose a recovery point and click **Select Recovery Point**. FULL and Incremental backups are shown as blue dots.



Select a Recovery Point: Use the slider to choose a valid date.

- The timeline shows 24 hours with valid backups.
- Each blue dot on the timeline represents a FULL or Incremental backup point.
- Blue dots can sometimes be clumped together if the backups are taken frequently.
- The green line represents valid log ranges (if CDP is enabled).
- Gaps in the green line represent breaks in the log chain.
- When positioned in an invalid range, the slider will snap back to the latest valid time.

4. Select the recovery point and complete the recovery options.



- **Recover As**: Select **New Object** to restore to an alternate server.
- **Target:** The name of the target server
- **Rename:** New name of the object
- **Concurrency:** A concurrency of 8 (default) is recommended.

5. Click **Recover** to start the recovery job.

Your Feedback

Was this document helpful? Send us your feedback!

About the Authors

Scott Lorenz is a Staff Solutions Engineer at Cohesity. Scott focuses on business-critical databases, applications, cloud storage, and enterprise data protection in his role. Scott has over 26 years of experience as an enterprise DBA.

Other essential contributors included:

- Edric Bulalacao is a Senior Solutions Architect.

Document Version History

VERSION	DATE	DOCUMENT HISTORY
1.0	Sep 2024	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.