The New Features of MongoDB 6.0

ApsaraDB January 12, 2023 2,507 0

This article reviews the history of MongoDB and explains the new features of MongoDB 6.0.

Quick View

MongoDB 6.0 was released in 2022. Alibaba Cloud ApsaraDB for MongoDB has completed the adaptation of version 6.0. You can try it on the official website console!

The main functional features of this version include:

- Enhanced Time Series Collection
- Enhanced Change Stream
- Queryable Encryption
- Enhanced Aggregation and Query Capabilities
- Cluster Synchronization

In summary, the new features of MongoDB 6.0 are designed to facilitate development and operations, eliminate data silos, and eliminate business complexity caused by the unnecessary use of additional third-party technologies.

New Features

Time Series Collections

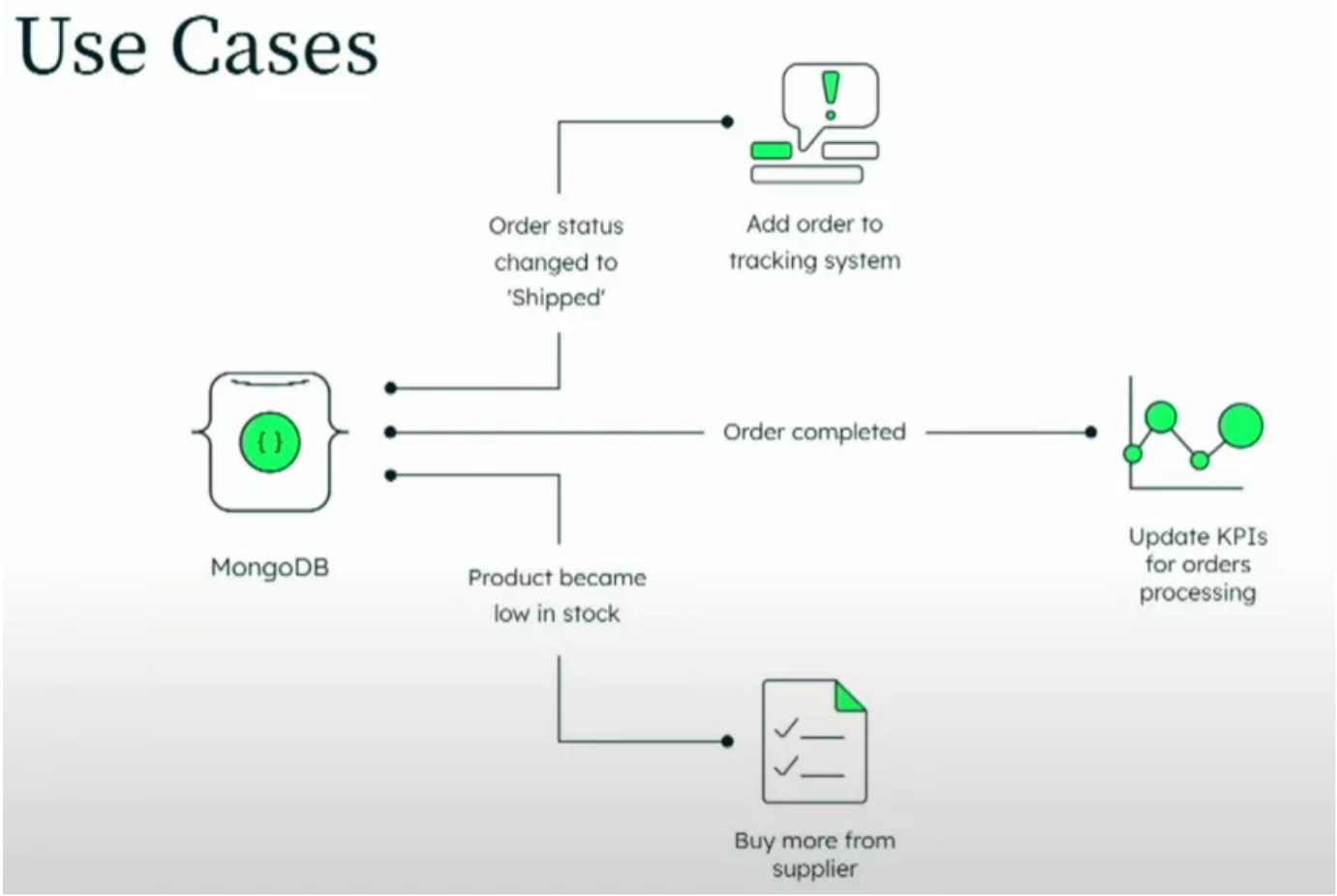
Time series data is a feature released in version 5.0. Since then, time series collection marks the determination of MongoDB (an OLTP database) to handle more AP scenarios. The official has maintained a high-speed update and improvement frequency for time series collection. For example, sharding was introduced in 5.1 to support the distribution of data better, columnar compression was introduced in 5.2 to improve storage usage, and densification and gap-filling were introduced in 5.3 to support timing analysis when some data points are missing.

In addition, the indexing of the time series collection is enhanced. From version 6.0, time series collection includes secondary and composite indexes of measurements to improve read performance. Time series data also begins to be combined more with spatial dimensions to form spatio-temporal data. Its combination begins to support geo-indexing. Developers can analyze scenarios involving examples and locations better by attaching geographic information to time series data. For example, tracking the temperature changes of cold-chain transport vehicles in summer and monitoring the fuel consumption of freight ships on specific routes.

There are some performance optimizations, mainly for query and sorting scenarios. For example, MongoDB can return the *last point* query in time series data more quickly instead of scanning the entire collection to obtain the desired data. The sorting operation can be completed efficiently through the clustered index and secondary index on the time and metadata fields.

Change Stream

As one of the core functions supporting CDC, Change Stream has made a major update since MongoDB 3.6. Change Stream allows businesses to easily obtain database changes in real-time and build various event-driven applications or systems based on it. They no longer need to rely on other data synchronization middleware.



Before introducing the new features, let's look at the update history of the Change Stream feature in previous versions:

Version	Feature Update
MongoDB 3.6	• Initial release with limited subscription dimensions and event types
	• Supports fault recovery and update to view the post-image
MongoDB 4.0	• Adds coarse-grained libraries and cluster dimensions
	• Supports drop, dropDatabase, and rename events
	• resumeToken Format Change (BinData->hex String)
MongoDB 4.2	• Supports more pipeline operators (such as $set/$unset)
	• Adds the startAfter option to start the listener according to time points
	• Error handling of modified id events
	• Removes the dependency on {readConcern: majority}
MongoDB 5.1	• Improves the execution efficiency of stages in some aggregation frameworks
	• Higher Resource Usage
MongoDB 5.3	• Supports filtering of orphaned document updates during chunk migration

In MongoDB version 6.0, the updates to Change Stream include:

- Supports pre-images and post-images of changes
- Supports more DDL, including create, createIndexes, modify, and shardCollection
- **Performance Improvement:** Pushdown and optimization of the stage in the pipeline on aggregation framework
- The change event adds the wallTime field. The timestamp can support multiple conversion and display operators ($toDate/$tsSeconds/tsIncrement) to facilitate business consumption.

Previously, the client could only obtain the changed state of the document through change events. Now, it can obtain the pre-changed state. This enables some downstream systems (used as proofreading or auditing) that need the changed state and pre-changed state to take full advantage of the native capabilities of the database without coupling themselves at the business level. More flexible functional forms will expand the future usage scenarios of changeStream. At the same time, supporting more DDL operations makes the application scenarios of Change Stream less restrictive. You will no longer encounter the situation where the resume is required after frequent interruptions. mongos was required to merge sharded clusters in previous scenarios, and it became the bottleneck of the changeStream with disappointing performance. This time, the performance improvement brought about by the optimization at the framework level and the stage pushdown in some scenarios is worth looking forward to.

All in all, this feature upgrade of the major version is a positive response to the wide range of users' demands (SERVER-36941 and SERVER-46414), indicating MongoDB's concern about the experience of database users.

Queryable Encryption (Preview)

Data security is becoming important. Existing transmission encryption or at-rest encryption methods cannot achieve encryption during use, which makes complex encryption and convenient queries seem to be contrary to each other. Queryable encryption in MongoDB 6.0 allows users to encrypt sensitive data from the client, store it as random encrypted data on the database server, and directly perform rich queries on the encrypted data.

As shown in the following example, only the client can see the plaintext of sensitive information. When the query reaches the server, the encryption key obtained from KMS is included. Then, the server queries and returns it in plaintext. Finally, the client uses the driver to decrypt the key and presents it in plaintext.

Its features and advantages include:

- Encrypting sensitive data from the client, and only the client has the encryption key.
- Data is encrypted throughout its lifecycle (transmission, storage, use, audit, or backup).
- The client can perform rich queries on encrypted data (equivalent match, range, prefix, suffix, substring, and other query types).
- **Strong Data Privacy Protection Capabilities:** Only authorized users with access to client applications and encryption keys can see plaintext data.
- **Lightweight Application Development:** Developers involved in sensitive data do not need to consider too many security and compliance matters since databases directly provide comprehensive encryption solutions.
- Reducing security concerns about migrating sensitive data to the cloud

It is worth mentioning that the queryable encryption function is still in the preview version. We do not recommend using it directly in the production environment. Community Version does not support automatic encryption with Client-Slide Field Level Encryption. You can use display encryption (you need to use the driver's encryption library lib).

Aggregation Feature

The aggregation feature of MongoDB allows users to process multiple documents and return calculation results. After combining multiple operators into an aggregation pipeline, users can build a data processing pipeline complex enough to extract data and analyze it. MongoDB 6.0 continues to develop the aggregation function and brings some new functions and optimizations, including:

- Improved $lookup support for JOINS

- Improved $graphlookup support for graph traversal
- $lookup and $graphlookup support sharded clusters
- $lookup performance improvement (up to 100 times in some scenarios)

Query

It adds support for some query operators (such as $maxN, $topN, $minN, $bottomN, $lastN, and $sortArrary). The main purpose is to use operators to sink computing from the business to the database, making the business layer lightweight. These $xxxN can be regarded as a supplement to the original $min/$max/$last operators. You can use different operators based on the need to return one or more results in real business scenarios. Let's take $maxN as an example. It can be used in arrays or integrated into the pipeline of aggregation statements.

```
# Used in the array
db.scores.insertMany([
   { "playerId" : 1, "score" : [ 1, 2, 3 ] },
   { "playerId" : 2, "score" : [ 12, 90, 7, 89, 8 ] },
   { "playerId" : 3, "score" : [ null ] },
   { "playerId" : 4, "score" : [ ] }
   { "playerId" : 5, "score" : [ 1293, "2", 3489, 9 ]}
])
# Add a new field to each document and keep the maximum 2 scores.
db.scores.aggregate([
   { $addFields: { maxScores: { $maxN: { n: 2, input: "$score" } } } }
   ])

# # Used in aggregation pipelines
db.gamescores.insertMany([
   { playerId: "PlayerA", gameId: "G1", score: 31 },
   { playerId: "PlayerB", gameId: "G1", score: 33 },
   { playerId: "PlayerC", gameId: "G1", score: 99 },
   { playerId: "PlayerD", gameId: "G1", score: 1 },
   { playerId: "PlayerA", gameId: "G2", score: 10 },
   { playerId: "PlayerB", gameId: "G2", score: 14 },
   { playerId: "PlayerC", gameId: "G2", score: 66 },
   { playerId: "PlayerD", gameId: "G2", score: 80 }
])
# Find 3 high scores for a single game.
db.gamescores.aggregate( [
   {$match : { gameId : "G1" }},
   {
      $group:
         {
            _id: "$gameId",
            maxThreeScores:
               {$maxN:{input: ["$score","$playerId"],n:3}}
               }
```

```
        }
] )
```

$maxN and $topN seem to be duplicated, but they have different scenarios. TopN returns the sorted results, but maxN does not depend on specific sorting rules.

$sortArray enhances the sorting of several sub-objects in an array. You can specify fields, nested fields, or compound fields in any sub-document to sort the array as needed. This operator can simplify some compound aggregated queries with array sorting.

```
db.engineers.insertOne(
   {"team":
         [
           {"name": "pat","age": 30,"address": { "street": "12 Baker St", "city":
           {"name": "dallas","age": 36,"address": { "street": "12 Cowper St", "cit
           {"name": "charlie","age": 42,"address": { "street": "12 French St", "c
           ]
    }
)

# Multiple query methods
db.engineers.aggregate( [
  { $project:{_id: 0,result:{$sortArray: { input: "$team", sortBy: { name: 1 } }}}
  ] )
  db.engineers.aggregate( [
    { $project:{_id: 0,result:{$sortArray: { input: "$team", sortBy: { age: -1, na
] )
db.engineers.aggregate( [
   { $project:{_id: 0,result:{$sortArray: { input: "$team", sortBy: { "address.cit
] )
```

Please refer to the official document of the relevant query operators for more details.

Elasticity

Initial sync is used in the data synchronization phase when a node is just added to a replica set. This phase has been criticized for poor performance. In version 6.0, it supports the initialization synchronization mode based on file copy, which improves the synchronization performance by about four times. **Currently, only the enterprise version supports this feature.**

In terms of horizontal scaling, MongoDB 6.0 made significant improvements in sharding. The original default chunk size of 64MB is adjusted to 128 MB. Larger data blocks mean less data migration frequency and lower network and routing layer overhead. This adjustment is made since many customers deploying large amounts of data on sharded clusters will encounter the bottleneck of chunk number and performance problems. MongoDB 6.0 supports configureCollectionBalancing commands to avoid arbitrary global parameters. You can set different chunk sizes for different sharded tables. For example, the chunk size is adjusted to

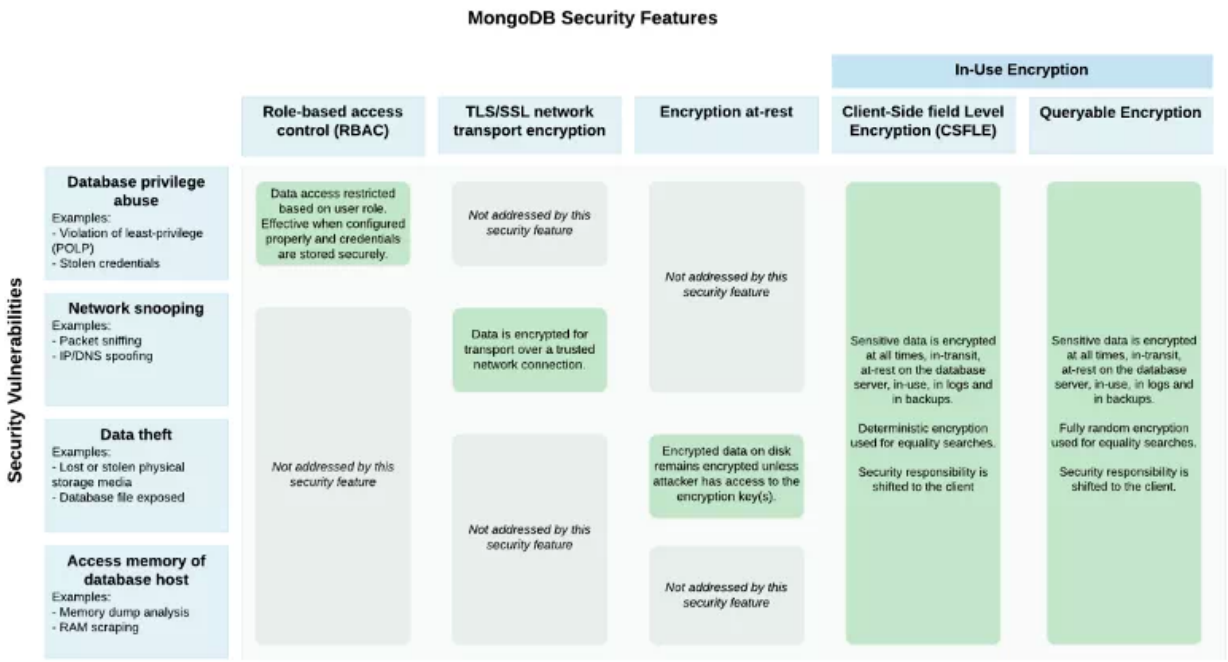
256MB for a sharded table with a large data size. The chunk size is adjusted to 64 or 32 MB for a table with a small data size but an evenly distributed shard.

At the same time, the command allows active defragmentation of the collection. Compared with the compact command, it provides better defragmentation services and reduces disk usage.

```
db.adminCommand(
  {
    configureCollectionBalancing: "<collection>",
    chunkSize: <num>,
    defragmentCollection: <bool>
   }
)
```

Security Enhancement

The following figure shows the security feature matrix of the entire MongoDB product. In addition to the new feature queryable encryption, MongoDB 6.0 brings other security enhancements.



Client-side field-level encryption (CSFLE) has been widely used in the management of sensitive data since its release in 2019, especially in the scenario of migrating data to the public cloud. CSFLE for MongoDB6.0 will support any Key Management Interoperability Protocol (KMIP)-compliant key management provider. As a leading industry standard, KMIP optimizes the storage, operation, and processing of encrypted objects (such as encryption keys and certificates), making the entire process standardized. This indicates that in addition to keyfile-based local key management, MongoDB supports integrating itself with third-party key management devices through KMIP to provide users with security guarantees.

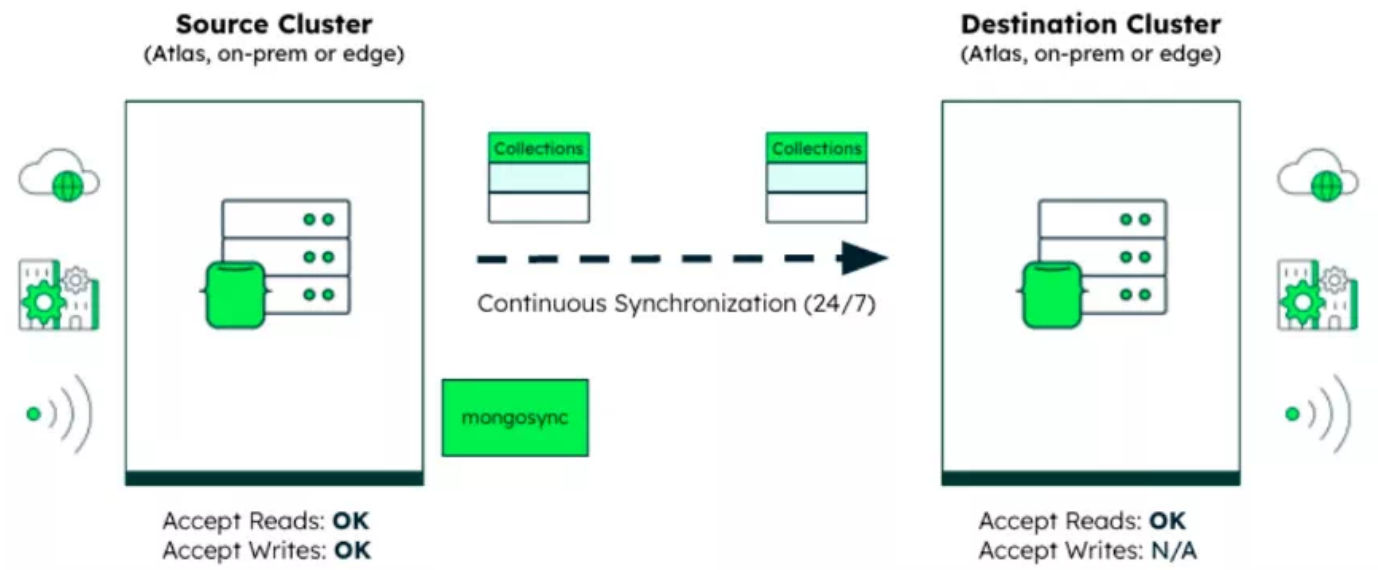
In addition, MongoDB 6.0 allows for the encryption of audit logs. The audit log may contain important information, which can be compressed, encrypted, and written to disk. It remains encrypted during the propagation of the audit log.

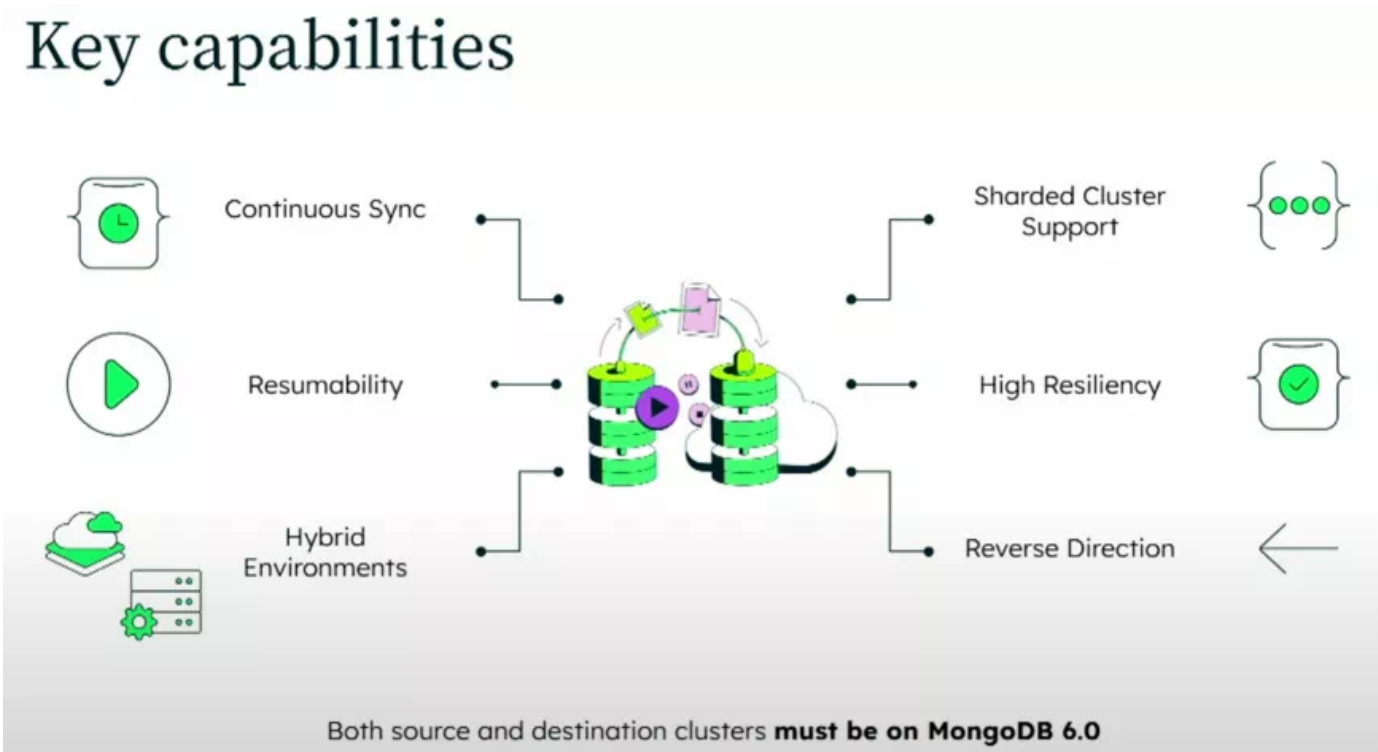
This encryption feature only applies to enterprise versions that use the WT engine. Atlas is enabled by default.

Cluster-to-Cluster Sync

Both homogeneous synchronization (mongo-to-mongo) and heterogeneous synchronization (others-to-mongo and mongo-to-others) of data are part of MongoDB ecology. The official has not planned *data synchronization* well in the tool matrix of mongoimport/mongoexport/mongodump/mongorestore. Therefore, the data synchronization of MongoDB prospers. This includes an early connector, MongoShake, DTS services from cloud service providers, and commercial products from some startups.

The official synchronization tool, mongosync, is available with the release of MongoDB 6.0. It can provide continuous and one-way data synchronization between two MongoDB clusters across any environment, whether it is a hybrid environment, Atlas, local, or edge environment. Users can control and monitor the entire synchronization process in real-time and start, stop, resume, or reverse synchronization as needed.





Data synchronization within a cluster is completed by replica sets and sharding. Cluster-to-Cluster Sync hopes to solve the synchronization problem between clusters, mainly covering the following scenarios:

- Migration from self-built to cloud services or vice versa
- Create a separate privatization database
- Support for DevOps policies (such as blue-green deployment)
- Build a dedicated analysis environment
- Meet localization requirements for auditing and compliance
- Center-to-edge synchronization

The usage method is the same as other data synchronization tools:

1) Prepare the source and target clusters

2) Deploy mongosync

3) Start synchronization as needed

4) After full synchronization is completed, incremental data will be synchronized

5) Use the common endpoint to terminate the synchronization relationship

Unfortunately, the Cluster-to-Cluster Sync is only compatible with the source and target clusters that run in MongoDB 6.0 or later. This means other lower versions that are widely used still rely on previous third-party synchronization tools to complete the data migration/synchronization scenarios mentioned above.

We will not list them due to limited space. Please see the release notes for more information about MongoDB version 6.0 changes:
https://www.mongodb.com/docs/v6.0/release-notes/6.0/

MongoDB | Database | Encryption | Change Data Capture | Time Series Collections | Change Stream | Cluster-to-Cluster | mongo-to-mongo

Read previous post:

AnalyticDB for MySQL Data Lakehouse Edition: Build a Cloud-Native Comprehensive Data Analysis Platform from Lake to Warehouse

Read next post:

Best Practices for PolarDB: Akulaku Cloud-Native Database Architecture Evolution

ApsaraDB

303 posts | 32 follow...

Follow

You may also like

An Excellent Three-Year Cooperation between MongoDB and Alibaba Cloud

ApsaraDB - December 22, 2022

Redis Core Team Welcomes Zhao Zhao from Alibaba Cloud to Run the Redis Project

ApsaraDB - October 12, 2020

[Infographic] Highlights | Database New Feature in November

ApsaraDB - December 19, 2022

On-demand Schemaless Slicing in PostgreSQL with TimescaleDB

digoal - May 16, 2019

The Evolution of Online Analytics from Alibaba Economy Ecosystem to the Cloud

ApsaraDB - February 20, 2021

See the Past and Future of Redis from Redis 7.0

ApsaraDB - June 13, 2022

Comments

Write your comment...

Post

More Posts by ApsaraDB

See All >

- Unveil New Possibilities at the Alibaba Cloud Data Management Summit in Indonesia
- Hands-On Lab | Migrate User-Created PostgreSQL to ApsaraDB RDS for PostgreSQL through Cloud Migration
- Best Practice: Create a Chatbot with LLM and AnalyticDB for PostgreSQL on Alibaba Cloud
- Hands-On Lab | Migrate User-Created MySQL to PolarDB for MySQL with DTS and Explore DTS Full Data Verification
- Hands-On Lab | Migrate User-Created MySQL to ApsaraDB RDS for MySQL with DTS
- Paper Interpretation | PolarDB Cost-Based Query Transformation
- The Intelligent Filter Condition Pushdown Principle of AnalyticDB for MySQL
- Hands-On Lab | Quick Backup and Restore ApsaraDB for MongoDB
- Hands-On Lab | Get Started with ApsaraDB for MongoDB
- What Are the Region Code and Availability Zone (AZ) Code of Alibaba Cloud ApsaraDB


TRUSTe
Verified Privacy
Powered by TrustArc

TRUSTe
Certified Privacy
Powered by TrustArc