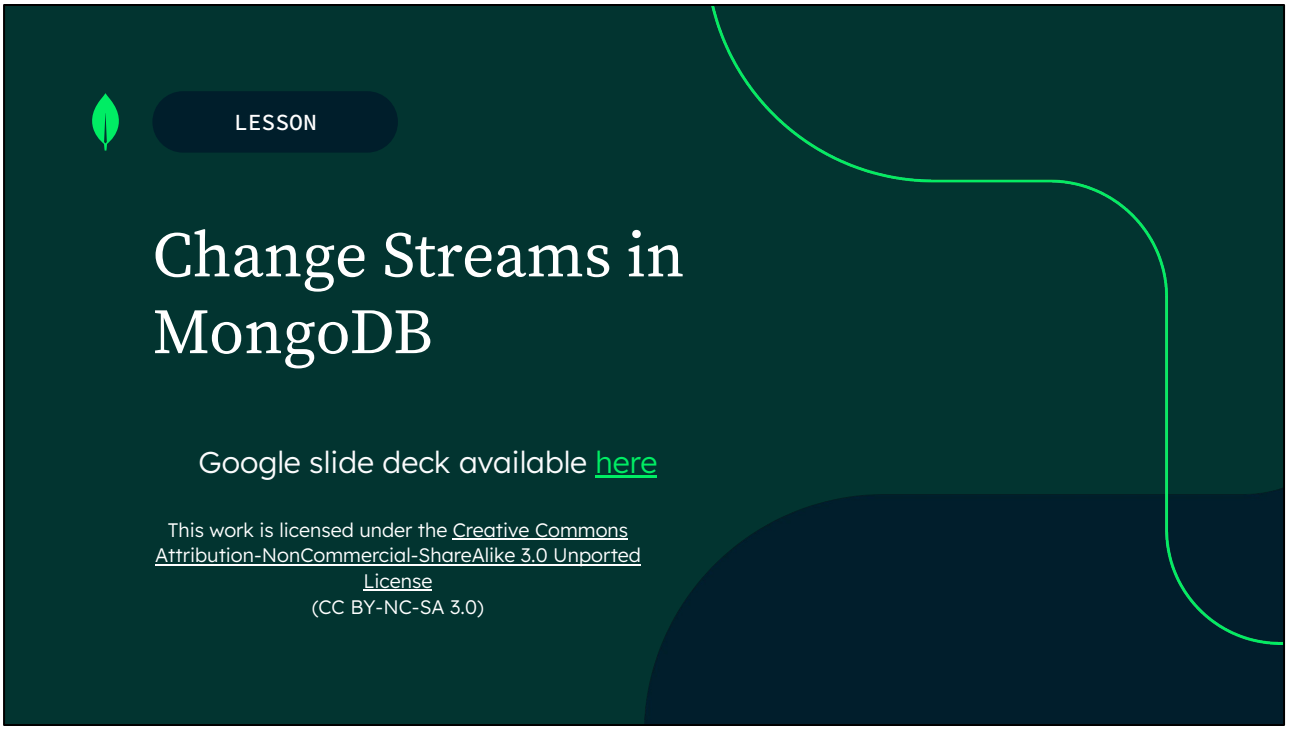

LESSON
Change Streams in MongoDB
Google slide deck available here
This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
(CC BY-NC-SA 3.0)

Overview

Learning Objectives

At the end of this lesson, learners will be able to:

- Define change streams and their purpose for application development.
- Define reactive programming and how it relates to change streams.
- Identify key advantages to using change streams.
- Identify the requirements for change streams to operate.
- Identify the three core components to the architecture of change streams in MongoDB.

Suggested Uses

- Lecture for one class period or spread across a couple of class periods
- Handouts / asynchronous learning
- Supplemental reading material - read on your own / not part of formal teaching
- Complement to University courses MongoDB for SQL Pros

This lesson is a part of the course Introduction to Modern Databases with MongoDB.

At a Glance

Length:
45 minutes

Level:
Advanced

Prerequisites:
MongoDB Architecture

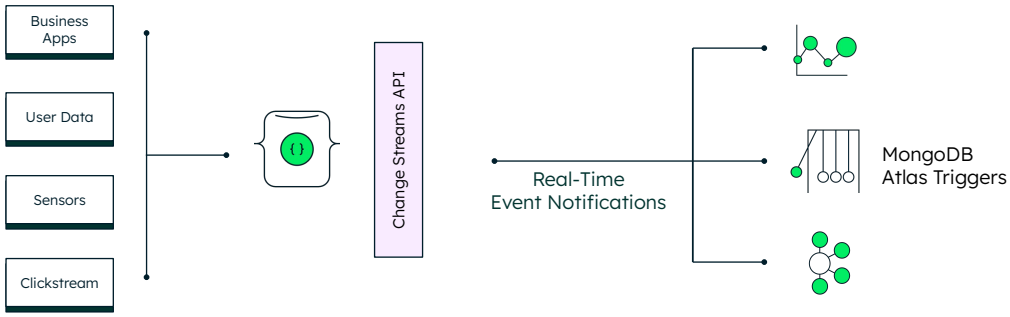
Share your feedback: We hope these curriculum materials will be a valuable resource for you and your learners. Let us know how the materials work for you, what we can improve on, and how MongoDB for Academia can support you via our brief feedback form.

MongoDB for Academia: MongoDB for Academia offers resources for educators and students to support teaching and learning MongoDB. Check out our educator resources and join the Educator Community. Students can receive $50 in Atlas credits and free certification through the GitHub Student Developer Pack.

Last Update: March 2025

MongoDB Change Streams

Enabling developers to build reactive, real-time services



This diagram illustrates a variety of real time applications that use MongoDB and how change streams can be used to support these use cases.

Change streams were built to allow developers to easily create reactive, real-time services that stream data changes from MongoDB to applications which can then act upon those changes.

They support a reactive style of programming. In MongoDB, we use change streams in a number of other products, specifically charts, triggers, and kafka. We'll look at Kafka briefly at the end of this lesson and it is covered in more depth in the Connectors lesson.

What is Reactive Programming?

Reactive programming is concerned with data streams and change propagation

Events, **messages**, and **calls** can all be programmed with this paradigm

The approach means the code written is **asynchronous code**

Reactive system are an architectural style to build responsive distributed systems, they are built using reactive programming

Reactive programming is concerned with data streams and change propagation

Events, messages, and calls can all be programmed with this paradigm

The approach means the code written is asynchronous code

Reactive system are an architectural style to build responsive distributed systems, they are built using reactive programming

In MongoDB, we use change streams in a number of other products, specifically charts, triggers, and kafka. We'll look at Kafka briefly at the end of this lesson and it is covered in more depth in the Connectors lesson.

What are Change Streams?

Change Streams allow your application to **access real-time data changes**, through a **dedicated API.**

Change Streams **do not rely** on complex **oplog tailing commands**.

Provides **resumable** mechanisms.

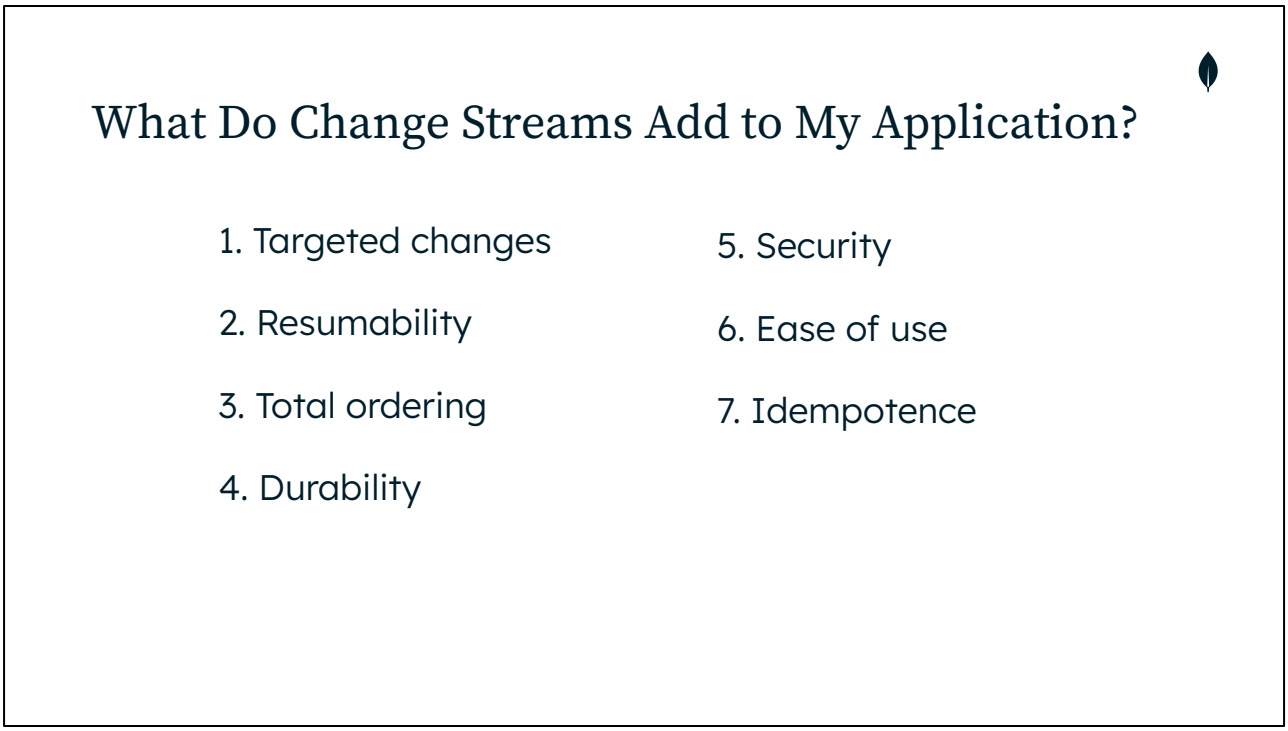
Can be combined with d**ata versioning** and **schema versioning** to provide **continuous delivery of data changes** to applications.

Changes Streams allow you to access real-time data changes through a dedicated API.

They were designed to replace ad-hoc custom scripting which used oplog tailing commands to provide a subset of similar functionality. Many customers and users developed these solutions, however they tended to be error prone and less secure. They did flag the need for change streams and helped see this function added to the core MongoDB database.

Change streams can be resumed and support a number of resumption mechanisms.

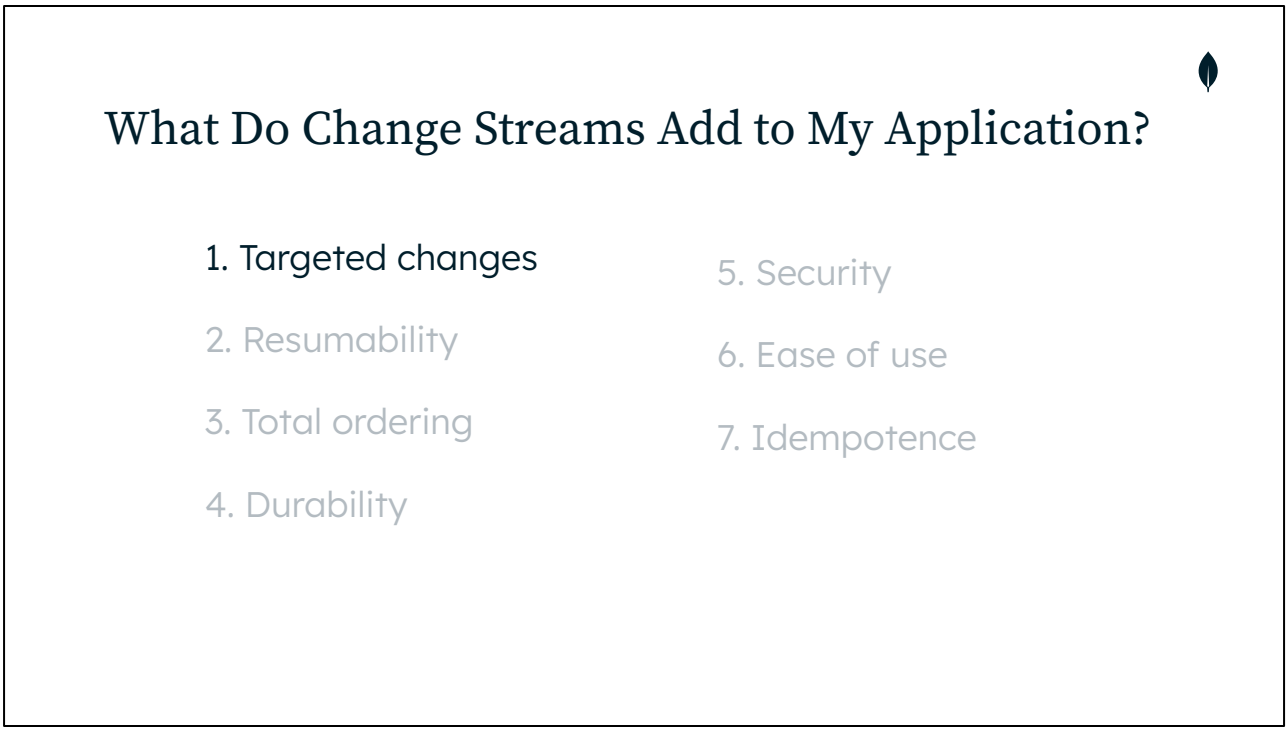
Change Streams use the concept of the document and pass the changes/events in a document. This allows them to be combined with the concepts of data versioning and schema versioning, we cover these in greater depth in our data modelling and schema design patterns lesson. This allows for the continuous delivery of data changes to applications.



Change streams allow you to manage targeted changes in the data.

Why should I care about change streams or more specifically what do change streams add to my application?

Firstly, Change streams allow for the tracking in real time of targeted changes so you can filter and pass only the relevant changes to any listening applications using change streams.

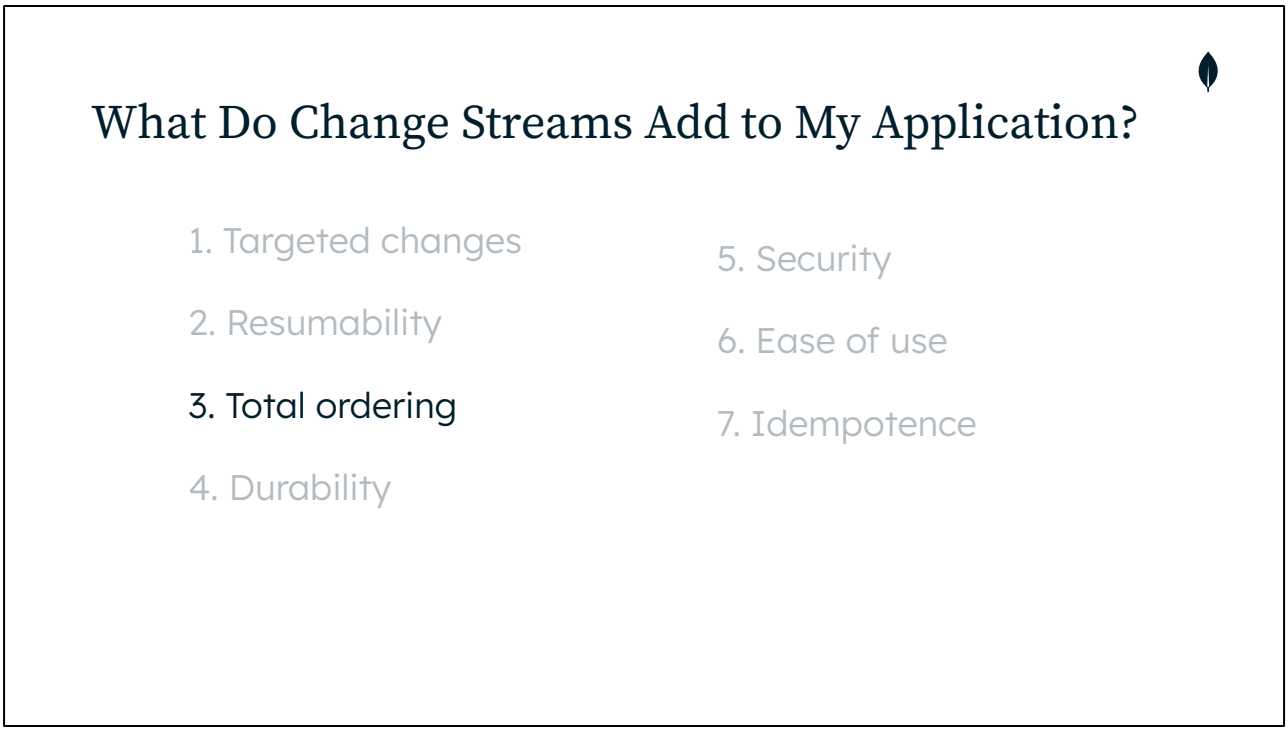


Firstly, Change streams allow for the tracking in real time of targeted changes so you can filter and pass only the relevant changes to any listening applications using change streams.

What Do Change Streams Add to My Application?

1. Targeted changes
2. Resumability
3. Total ordering
4. Durability
5. Security
6. Ease of use
7. Idempotence

They allow for resumability so your application does not need to be always connected and in the case of a temporary network blip, you can easily continue. Resumability has been expanded in recent versions to allow you to resume after events that delete or drop data. We'll learn more about the various types of events shortly.



Thirdly, you can get changes in an ordered fashion. The use of a cluster wide logical clock means that you can now be guaranteed as to the sequencing of events.

What Do Change Streams Add to My Application?

1. Targeted changes
2. Resumability
3. Total ordering
4. Durability
5. Security
6. Ease of use
7. Idempotence

Durability and specifically that only majority-committed changes are sent means that in many failure scenarios (e.g. the loss of a Primary and re-election of a new Primary) the data passed by the change stream is resilient and not impacted by these events.

What Do Change Streams Add to My Application?

1. Targeted changes
2. Resumability
3. Total ordering
4. Durability
5. Security
6. Ease of use
7. Idempotence

Security is a major concern in many applications and change streams supports using MongoDB's existing authentication and authorization mechanisms to ensure all the potential security aspects are covered.

What Do Change Streams Add to My Application?

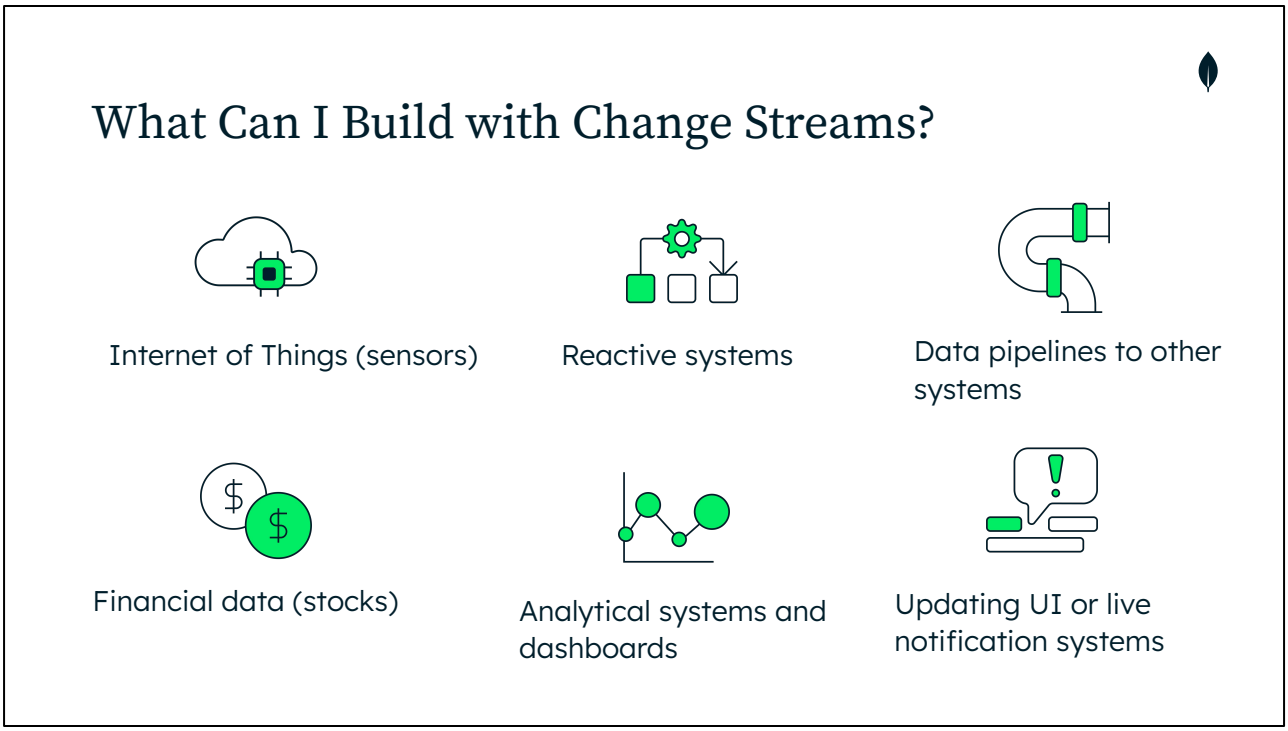
1. Targeted changes
2. Resumability
3. Total ordering
4. Durability
5. Security
6. Ease of use
7. Idempotence

Ease of use refers specifically to well defined API calls that sit within the existing MongoDB Drivers so are already both language idiomatic and familiar to developers. It also refers to the fact that in the past the only approach to achieve this functionality was to write custom error prone code to tail the oplog. This API is a far superior replacement to this earlier hand crafted scripting approach.

What Do Change Streams Add to My Application?

1. Targeted changes
2. Resumability
3. Total ordering
4. Durability
5. Security
6. Ease of use
7. Idempotence

Idempotence as the events can be resumed and the ordering is guaranteed such that even when a change stream is resumed the changes will still reach the same consistent state. This is a similar result to oplog entries which are similarly idempotent.



Internet of Things (sensors) are one of the wider variety of applications and use cases that can be supported with change streams.

A common use for change streams is passing changes in collections which are storing IoT sensor data. The sensor write their updates to the database and the change streams pass these changes to the listening applications.

Similarly, financial data such as stock data is passed to a MongoDB collection and then the changes / updates are sent using change streams to listening applications.

Reactive systems are applications that await and then act upon the receipt of data. This is different to applications that request or poll for data. Change streams support the reactive programming style.

A variety of analytical and dashboard systems can be built to utilise change streams allowing changes to the database to be reflected in real time to these systems.

We mentioned Kafka and we'll look to it later in this lesson but it's a good example of how data in MongoDB can be passed using change streams as a data pipeline to other systems (such as Kafka).

Applications that are continuous updating their user interface or provide live notification can use change streams to provide the mechanism to transmit the change from the database to the UI/notification system.

Change Streams Requirements

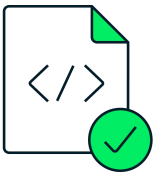
Replica Set or Sharded Cluster

A **collection, a database, or an entire replica set** which will be monitored

Support up to **100 concurrent streams**

Notification only occurs **after the data is persisted to the majority of the data-bearing** members

In sharded clusters, change stream must be opened against as '**mongos**' to see all the changes



At the core of change streams, the technology is built on the MongoDB replication mechanism so this means that change streams are only available to replica sets or sharded clusters.

A collection, a database, or an entire replica set can be monitored.

There is a recommended limit of approximately 100 concurrent streams. Beyond this and the performance may vary so this should be factored into their system design.

A key aspect of change streams is that the data will only be sent as a notification after it has already been persisted to the majority of data bearing members in the deployment.

In terms of sharded cluster, you must open a change stream against a mongos routing process to ensure all the changes across the deployment are captured. If you only open it against a primary in a sharded cluster, only that shard's changes are visible and it is not recommended to do this for sharded clusters.

For reference, change streams can be opened against all non-system collections across all databases except for *admin*, *local*, and *config*.

Quiz

Quiz

Which of the following is true for Change Streams in MongoDB?
More than one answer choice can be correct.

- A. Require a replica set or sharded cluster
- B. Recommended to use with 30 or less continuous Change Streams
- C. Can monitor a collection only
- D. Require majority committed data

Quiz

Which of the following is true for Change Streams in MongoDB?
More than one answer choice can be correct.

- ✓ A. Require a replica set or sharded cluster
- ✗ B. Recommended to use with 30 or less continuous Change Streams
- ✗ C. Can monitor a collection only
- ✓ D. Require majority committed data

CORRECT: Requires Replica Set or Sharded Cluster - Change streams require the oplog as such either a replica set or a sharded cluster is necessary.
INCORRECT: Recommend to use with 30 or less continuous Change Streams - The recommendation is for approximately 100 continuous change streams
INCORRECT: Can monitor a collection only - Change Streams can monitor a collection, a database, or an entire deployment.
CORRECT: Requires majority committed data - This is correct, only data which has been committed to the majority of data bearing members will be sent in a change stream as an event.

Quiz

Which of the following is true for Change Streams in MongoDB? More than one answer choice can be correct.

- ✓ A. Require a replica set or sharded cluster
- ✗ B. Recommended to use with 30 or less continuous Change Streams
- ✗ C. Can monitor a collection only
- ✓ D. Require majority committed data

This is correct. Change streams require the oplog as such either a replica set or a sharded cluster is necessary.

CORRECT: Require a Replica Set or Sharded Cluster - Change streams require the oplog as such either a replica set or a sharded cluster is necessary.

Quiz

Which of the following is true for Change Streams in MongoDB? More than one answer choice can be correct.

- ✓ A. Require a replica set or sharded cluster
- ✗ B. Recommended to use with 30 or less continuous Change Streams
- ✗ C. Can monitor a collection only
- ✓ D. Require majority committed data

This incorrect. The recommendation is for approximately 100 continuous change streams.

INCORRECT: Recommended to use with 30 or less continuous Change Streams - This is incorrect. The recommendation is for approximately 100 continuous change streams

Quiz

Which of the following is true for Change Streams in MongoDB? More than one answer choice can be correct.

- ✓ A. Require a replica set or sharded cluster
- ✗ B. Recommended to use with 30 or less continuous Change Streams
- ✗ C. Can monitor a collection only
- ✓ D. Require majority committed data

This incorrect. Change Streams can monitor a collection, a database, or an entire deployment.

INCORRECT: Can monitor a collection only - This is incorrect. Change Streams can monitor a collection, a database, or an entire deployment.

Quiz

Which of the following is true for Change Streams in MongoDB? More than one answer choice can be correct.

- ✓ A. Require a replica set or sharded cluster
- ✗ B. Recommended to use with 30 or less continuous Change Streams
- ✗ C. Can monitor a collection only
- ✓ D. Require majority committed data

This is correct. Only data which has been committed to the majority of data bearing members will be sent in a change stream as an event.

CORRECT: Require majority committed data - This is correct. Only data which has been committed to the majority of data bearing members will be sent in a change stream as an event.


Architecture &
Concepts

Change Streams and the Oplog

Change Streams **query the oplog** and sometimes query the collection being watched.

The **namespace** identifies the resource being watched.

The **clusterTime** identifies the time from the specific oplog entry globally across the deployment.

The **txnNumber** (transaction number) and the lsid (logical session id) are used if the operation is part of a transaction.

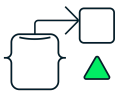
Change Streams use the oplog mostly but they can also query the collection being watched.

The namespace is what explicitly identifies the resource being watched by the change stream.

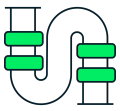
The clusterTime uses the global cluster time to specifically identify the time for an event and it's taken from the oplog entry.

The transaction number (txnNumber) and the logical session identifier (lsid) are used additionally if the operation involves a transaction to further identify it.

Change Stream Architecture and Concepts



Change Events



Aggregation Framework

Change Stream Cursor

There are three core components within the Change Stream architecture.

Firstly, there are change events. These are the various event types and related information that are sent from Change Streams to the listening applications.

Secondly, the Aggregation Framework is a key element to the processing and filtering aspects of Change Streams. This allows for changes to be passed through an aggregation pipeline and further modified before being set to the listening applications if required.

Thirdly and finally, the Change Stream Cursor is the cursor which holds the connection from the database to the listening application.

We'll cover each of these in a little more depth in the coming slides.

For more details see:
https://www.mongodb.com/blog/post/an-introduction-to-change-streams

Change Events

insert	drop
delete	rename
replace	dropDatabase
update	invalidate

Invalidate Events

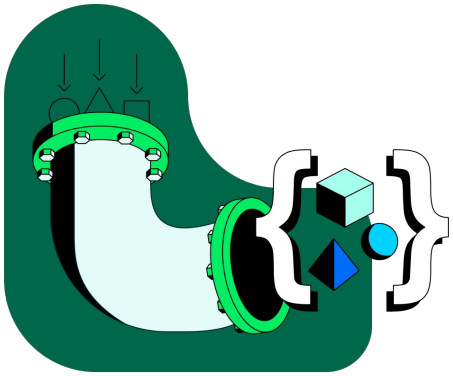
CRUD Events

There are eight distinct change events that are tracked by change streams.

You can track insertion events, delete events, replace events and update events. These first four change events are typically categorised as CRUD events.

You can also track drop events, rename events, dropDatabase events, and invalidate events.
These are typically categorised as Invalidate Events. Specifically the drop, the rename, and the dropDatabase event types are always followed by an invalidate e

Aggregation Framework



Enables change streams to allow applications to filter for specific changes or transformation notifications.

The Aggregation Framework powers the filtering and data change capabilities within Change Streams. This is a well known feature within MongoDB which helps developers more easily develop code that performs filters or changes as it doesn't require any new or additional learning.

Let's look at how to use Change Streams with Python to monitor inserts to a collection.

How to Monitor Inserts on a Collection

```
try:
    with db.collection.watch(
            [{'$match': {'operationType': 'insert'}}]) as stream:
        for insert_change in stream:
            print(insert_change)
except pymongo.errors.PyMongoError:
    # The ChangeStream encountered an unrecoverable error or the
    # resume attempt failed to recreate the cursor.
    logging.error('...')
```

In this example, we've left out the connection and other aspects of typical scaffolding for your Python application but the code illustrates the main functionality we are interested in, that is configuring monitoring on a collection to watch for insertion events.

Let's look at this more deeply in the next slide.

For more details see https://pymongo.readthedocs.io/en/stable/api/pymongo/collection.html

How to Monitor Inserts on a Collection

```
try:
    with db.collection.watch(
            [{'$match': {'operationType': 'insert'}}]) as stream:
        for insert_change in stream:
            print(insert_change)
except pymongo.errors.PyMongoError:
    # The ChangeStream encountered an unrecoverable error or the
    # resume attempt failed to recreate the cursor.
    logging.error('...')
```

We are just going to focus on configuring our change stream cursor to only match insert events.

We use the ‘watch’ command on a collection to configure the change stream.

We can see how we use the standard Aggregation Framework syntax and pass this as the pipeline parameter to the watch function to create and open the change stream cursor.

It’s possible to watch every type of event by simply passing no parameters to the watch() function.

For more details see https://pymongo.readthedocs.io/en/stable/api/pymongo/collection.html

Valid Aggregation Stages for Change Streams

- $addFields
- $match
- $project
- $replaceRoot
- $replaceWith
- $redact
- $set
- $unset

They're eight valid aggregation stages that can be used with change streams as of MongoDB 4.4.

$addFields allows us to add fields to the document for the event type we are watching.

$match as we saw on the last slide allows us to limit our change stream to certain types of change events but it also allow us to limit it to certain values as we would typically with the Aggregation Framework.

$project allows for certain fields to be selected and the other fields will not be included in the output document.

$replaceRoot allows for the replacement of the entire document.

$replaceWith is an alias of $replaceRoot.

$redact can be used to restrict the contents of the documents based on information stored in the documents themselves

$set is an alias for $addFields

$unset provides the functional to remove fields from the document.

Change Stream Cursor

Remains open until:

- It is **explicitly closed**, or
- An **invalidation event** occurs.

However, a new cursor can be open with a **Resume Token** to restart after a specific event.

The third component to the change stream architecture is the concept of the change stream cursor.

It's similar to a normal database cursor with the exception it does not timeout rather it will remain open until it is either explicitly closed or an invalidation event occurs.

In either case, you can resume the change stream using a resume token which we will discuss later in next slide.

See: https://docs.mongodb.com/manual/changeStreams/#resume-a-change-stream

Resuming a Change Stream

resumeAfter

startAfter

Types of Resume Token

You can resume a change stream using a resume token, there are two kinds of resume tokens.

The first, resumeAfter, which uses the _id value of the change stream event document to indicate when it should restart from. The oplog must also be long enough to support this timeframe. The resumeAfter token cannot be used after an invalidateEvent.

The second type of resume token is a startAfter token. This can be used after an invalidateEvent. It resumes notifications after an invalidate event by creating a new change stream. It also uses the _id value of the change stream event document to indicate when it should restart from. The oplog must also be long enough to support this timeframe.

See: https://docs.mongodb.com/manual/changeStreams/#resume-a-change-stream

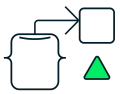
Change Stream Gotchas

MongoDB **16MB BSON document limit**

Change steams need to be updated if collections/databases are dropped

Enough data bearing members must be available

High levels of activity can cause issues keeping up with changes, **filter the information** (particularly in large sharded clusters)



In terms of using Change Streams there are a number of issues or gotchas to be aware of.

The MongoDB 16MB BSON document limit still applies so change documents should be kept relatively small.

If collections or databases are dropped then you may need to update your change stream or code your application to handle this gracefully.

For change streams to work, you will need sufficient data bearing members to be available. In terms of majority committed data, this means that your deployments should always have sufficient (and more if possible) data bearing members available to support the write concern of majority committed data. In cases with arbiters but insufficient data bearing members, the change stream will not get any change data as the precondition of majority committed cannot be satisfied.

Finally, for large active cluster with high levels of activity, particularly for large sharded clusters, you should use filtering to limit the information rather than taking all the information as the change stream may not be able to maintain the pace required to transmit this information to your application before it befores swamped.

See https://docs.mongodb.com/manual/administration/change-streams-production-recommendations/

Use Cases

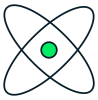
Use Cases

Access Control

Authentication and authorization can be applied to a Change Stream

Event Notifications

Notification only occur with majority-committed changes



Collation

Can use an explicit collation or defaults to a binary comparison

Change streams have three typical use cases:

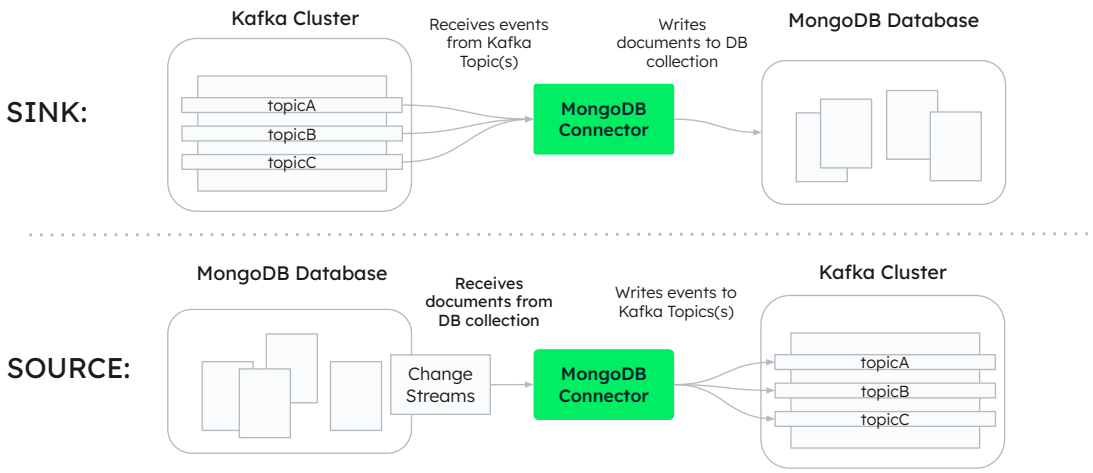
Firstly, where you want additionally access controls on the data stream. In earlier oplog tailing approaches, the security aspects of authentication and authorization were not present. A pertinent example was a security issue with the MeteorJS framework which used the earlier approach of oplog tailing scripts and leaked data because it didn't have the same security features as change streams (https://forums.meteor.com/t/mongodb-oplog-tailing-and-security/32976).

The second use case here but the prime use for change streams is typically event notification and this only occurs with majority-committed changed.

The final use case for change streams is that it can more explicitly (for a general set of spoken languages) compare with the collation feature. This expands beyond the typical binary comparison more commonly available in this type of data streaming functionality.

MongoDB, Kafka + Change Streams

MongoDB Kafka Connector



The MongoDB Kafka Connector is covered in more depth in the Drivers, Connectors, and Ecosystem lesson. We will focus in this slide on how it uses change streams.

It's useful to highlight how we use change streams in a real product.

Change streams allow a MongoDB Database and the document changes to be written to Kafka in real-time as they occur. In Kafka terminology the MongoDB Database acts as a source to the Kafka cluster.

Equally, MongoDB's Kafka Connector can take data from Kafka to a MongoDB Database where the database then becomes a 'sink'. It doesn't use change streams.

Additionally, it is worth nothing that Atlas uses Change Streams for the Database Trigger functionality it provides.

Quiz

Quiz

Which of the following are use cases for Change Streams in MongoDB? More than one answer choice can be correct.

- A. Access control
- B. Event notification
- C. Schema translation
- D. Collation

Quiz

Which of the following are use cases for Change Streams in MongoDB? More than one answer choice can be correct.

- ✓ A. Access control
- ✓ B. Event notification
- ✗ C. Schema translation
- ✓ D. Collation

CORRECT: Access control, Change Streams provide the MongoDB authentication and authorization access control mechanisms
CORRECT: Event notification, Change Streams provide the ability for event notifications of data changes in MongoDB in real-time.
INCORRECT: Schema translation, Change Streams can provide schema translation via the Aggregation Framework but it is not translated in the sense of an identified / desired output and input schemas. It is possible with additional code to perform schema translation but as it's not a use case or a simple parameter/option to a function we'll consider it incorrect.
CORRECT: Collation, Change Streams do provide for comparisons in a number of spoken languages so that it can offer a richer comparison than a binary comparison in these cases.

Quiz

Which of the following are use cases for Change Streams in MongoDB? More than one answer choice can be correct.

- ✓ A. Access control
- ✓ B. Event notification
- ✗ C. Schema translation
- ✓ D. Collation

This is correct. Change Streams provide the MongoDB authentication and authorization access control mechanisms.

CORRECT: Access control, Change Streams provide the MongoDB authentication and authorization access control mechanisms

Quiz

Which of the following are use cases for Change Streams in MongoDB? More than one answer choice can be correct.

- ✓ A. Access control
- ✓ B. Event notification
- ✗ C. Schema translation
- ✓ D. Collation

This is correct. Change Streams provide the ability for event notifications of data changes in MongoDB in real-time.

CORRECT: Event notification. - This is correct. Change Streams provide the ability for event notifications of data changes in MongoDB in real-time.

Quiz

Which of the following are use cases for Change Streams in MongoDB? More than one answer choice can be correct.

- ✓ A. Access control
- ✓ B. Event notification
- ✗ C. Schema translation
- ✓ D. Collation

This incorrect. Change Streams can provide schema translation via the Aggregation Framework but it is not translated in the sense of an identified / desired output and input schemas.

INCORRECT: Schema translation. - This is incorrect. Change Streams can provide schema translation via the Aggregation Framework but it is not translated in the sense of an identified / desired output and input schemas.

Note: It is possible with additional code to perform schema translation but as it's not a use case or a simple parameter/option to a function.

Quiz

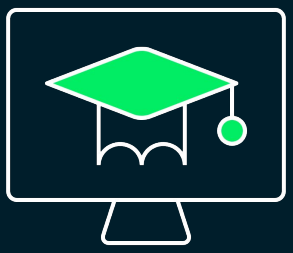
Which of the following are use cases for Change Streams in MongoDB? More than one answer choice can be correct.

- ✓ A. Access control
- ✓ B. Event notification
- ✗ C. Schema translation
- ✓ D. Collation

This is correct. Change Streams do provide for comparisons in a number of spoken languages so that it can offer a richer comparison than a binary comparison in these cases.

CORRECT: Collation. - This is correct. Change Streams do provide for comparisons in a number of spoken languages so that it can offer a richer comparison than a binary comparison in these cases.

Continue Learning!



MongoDB University has free self-paced courses and labs ranging from beginner to advanced levels.

GitHub Student Developer Pack

Sign up for the MongoDB Student Pack to receive $50 in Atlas credits and free certification!

This concludes the material for this lesson. However, there are many more ways to learn about MongoDB and non-relational databases, and they are all free! Check out MongoDB's University page to find free courses that go into more depth about everything MongoDB and non-relational. For students and educators alike, MongoDB for Academia is here to offer support in many forms. Check out our educator resources and join the Educator Community. Students can receive $50 in Atlas credits and free certification through the GitHub Student Developer Pack.