LESSON

Drivers, Connectors and the Ecosystem

Google Slide deck available here

Overview

Learning Objectives

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

- State the purpose of connecting a driver to a database.
- Connect a driver to MongoDB.
- Define the four main connectors available with MongoDB.
- Explain the functionality of the four connectors and why one would be used for application development.
- Explain how MongoDB Sink works in conjunction with the Kafka connector.
- Describe MongoDB Functions and how they can advance application development in Atlas.

Suggested Uses

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

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

At a Glance

Length:
90 minutes

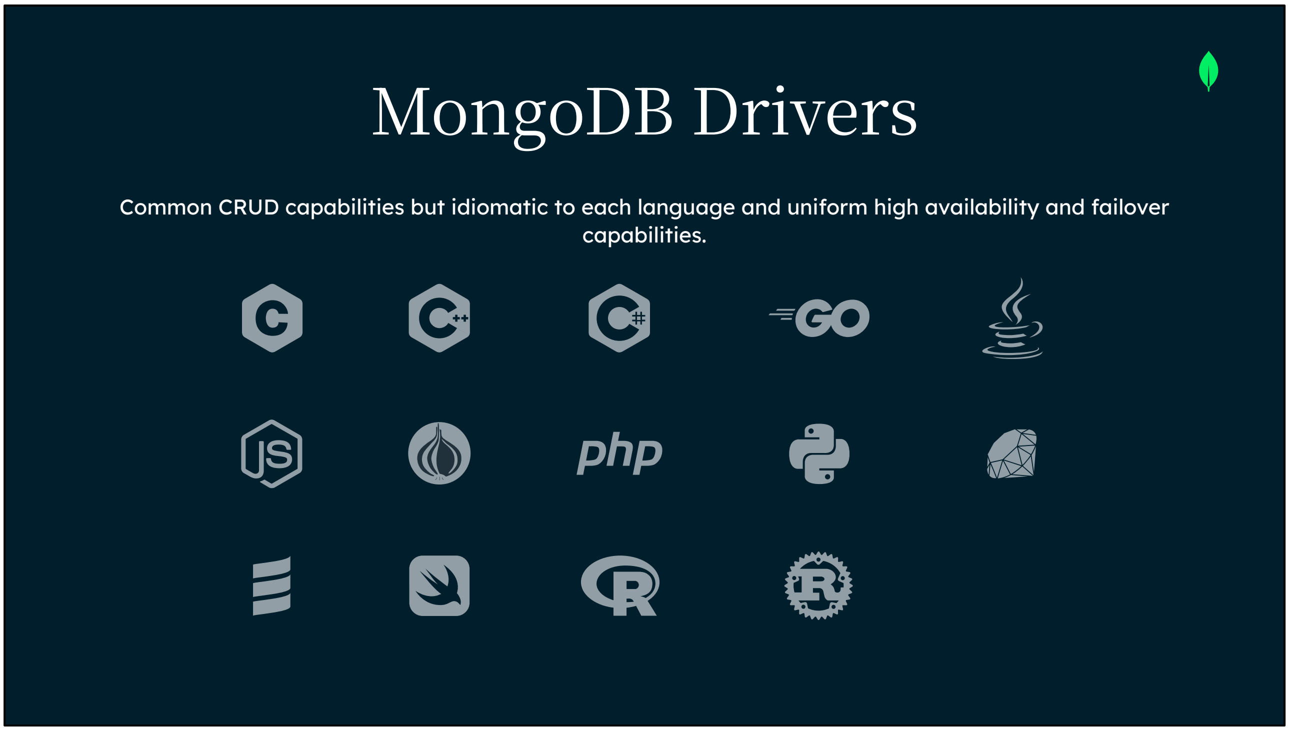
Level:
Advanced

Prerequisites:
Change Streams in MongoDB

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 Drivers

Common CRUD capabilities but idiomatic to each language and uniform high availability and failover capabilities.

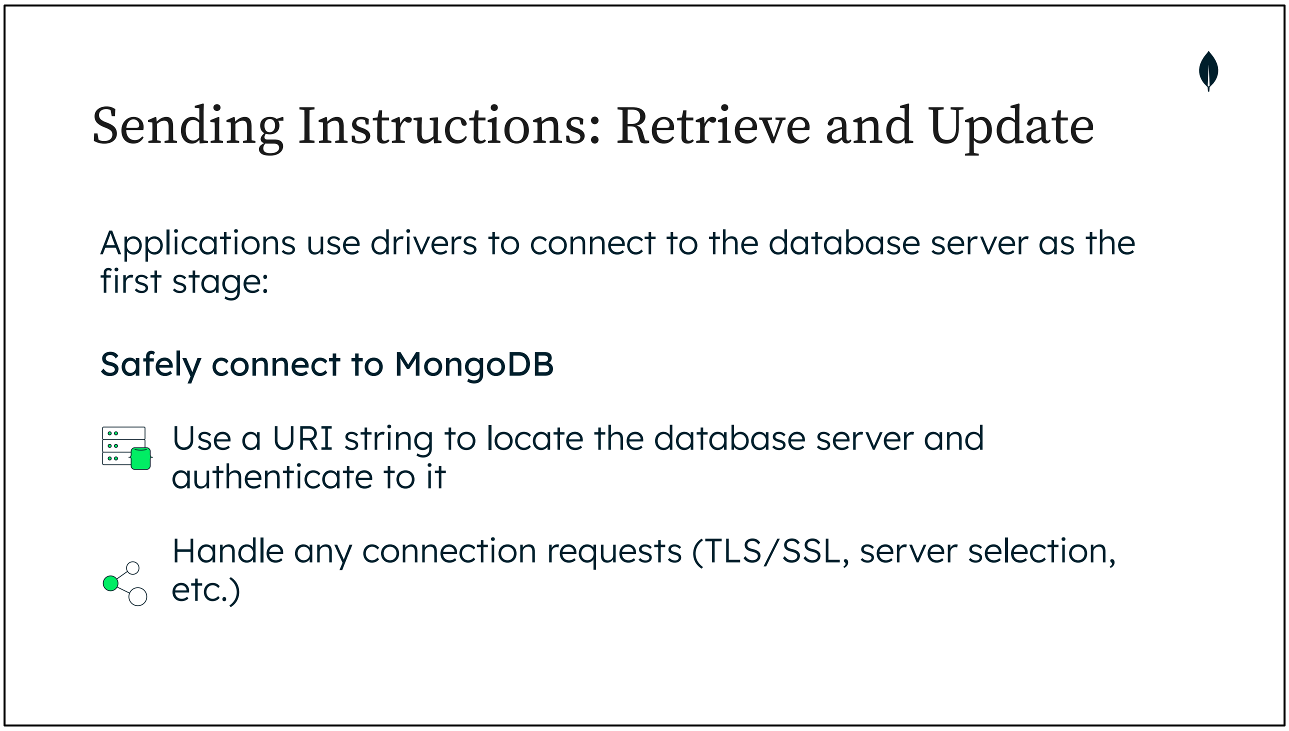
As all of the drivers implement the same specifications they all provide:

- Common CRUD capabilities but idiomatic to each language
- Uniform High Availability & Failover capabilities

MongoDB offers a wide range of drivers for various programming languages. They are all built to meet various driver specifications, these specifications are available on Github at https://github.com/mongodb/specifications and you can find more details at https://docs.mongodb.com/drivers/specs on the various drivers directly supported by MongoDB. There are more programming languages with community supported drivers.

Connecting to MongoDB

Let’s look now at some of the aspects involved in connecting a program to a MongoDB database, many of these issues and concerns are common across any kind of database.



Sending Instructions: Retrieve and Update

Applications use drivers to connect to the database server as the first stage:

Safely connect to MongoDB

- Use a URI string to locate the database server and authenticate to it
- Handle any connection requests (TLS/SSL, server selection, etc.)

Applications use drivers to connect to the database and via the drivers they issue data manipulation requests. Applications are the components that most people know about - the mobile apps, the websites, etc. Inside the code that runs the application, whatever language that application is written in, there's a library imported to connect to and issue requests to MongoDB. This is the driver which actually performs the connection and issues the data manipulation requests.

In order to safely connect to MongoDB, you will firstly need the uniform resource indicator (URI) string which the driver will use to firstly locate the database and then as part of connecting to it, it includes authenticate details to allow you pass the necessary security checks.

The driver also needs to handle other aspects of the connections including the encryption of the connection communications using Transport Layer Security / Secure Socket Layer (TLS/SSL). It also must choose which of the database instances to direct the requests to (typically this will be the Primary but this can be configured). The driver will also handle how to handle connections if there is a network error or indeed if there was an election within the MongoDB replica set.

Sending Instructions: Retrieve and Update

Once connected, drivers issue data manipulation requests to database and pass the results back to the applications:

Issue DML requests to MongoDB

- Format language-specific requests for the BSON protocol
- Satisfy read/write concerns, transactions, and sessions

After connecting and managing the connections to the database, the driver's other main purpose is to issue data manipulation (DML) requests.

Well, that's the real beauty of the driver. At the end of the day, MongoDB data is stored in BSON - this is not JSON, and it's definitely not human readable. It's a storage format that's been optimized for data types and disk usage. This means that the driver will translate any programming language specific requests into the appropriate BSON.

This idea is one of the key aspects behind the driver, why does a developer need to know the details of how data is stored on the server?

Building off this, if the developer doesn't need to know any of this stuff, why not just let them write code in their application's native language?

The driver must also satisfy other aspects such as read/write concerns, transactions, and sessions to ensuring requests sent to the database match these aspects as defined within the application.

This is an Important Point to reiterate:
Any of the complexities involved with connecting should be handled by the client, which is created by the driver. Any details around write concerns, transactions and server selection are abstracted away from the developer by the driver. If the developer needs a TLS/SSL connection, they just express this desire in Python, and the driver handles the rest.

Exceptions Managed by the Driver

The driver must manage errors or exceptions that occur whether in terms of a network interruption or some other error or issue.

Handling Exceptions in the Application Code

Operational Failures

Schema validation error,
transaction failure,
duplicate key error,
execution timeout

Network Failures

Server selection
timeout, network
timeout, write concern
timeout

Server Errors

Configuration error,
not primary error,
exceeded max
waiters

The driver does a lot of work - it provides an interface between the application and the MongoDB server. But there are some things the driver can't do, and in some cases it actually shouldn't do anything - these are raised in the form of exceptions.

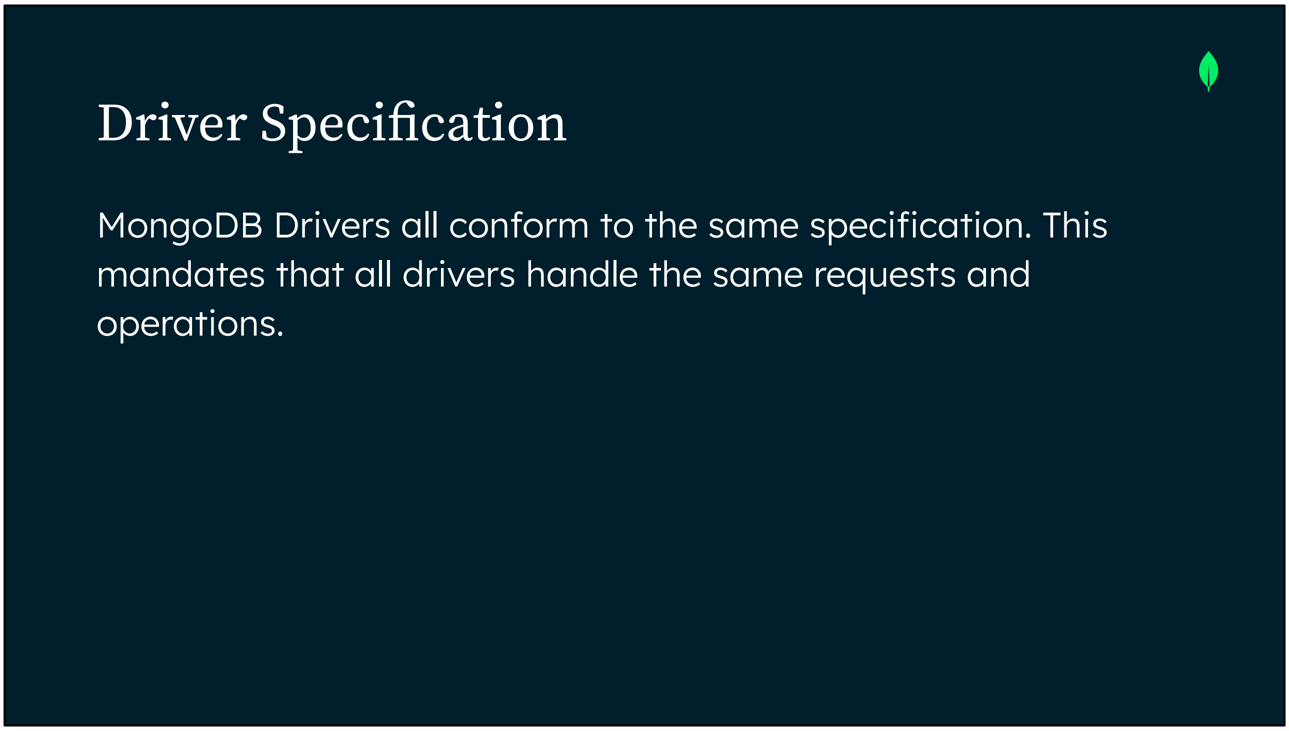
Operation failures can cover everything from schema validation errors, to transaction failures, to duplicate key errors, to execution timeouts as a few of the examples. These are related to some exception being raised by the database based on the request made by the application via the driver.

Network failures cover a range of problems from server selection timeout, to network timeout, to write concern timeouts as a few examples of the possible problems.

Server errors cover yet another set of problems from configuration errors, to issues with the node selection like it not being the primary when the request needs to be performed against the primary, to storage errors where WiredTiger may flag it has too many waiting requests.

As you can see, there's a few different types, because there's more than a few situations where the driver needs the application to decide what to do. For example, the driver can't decide what to do if the operation fails a Duplicate Key error - this exception is thrown when a unique field, such as the _id field, receives two of the same values. The driver doesn't know if you want to generate a new ObjectId(), or a new integer, or what. So it throws its hands up, and gives the application a chance to decide what to do.

The same concept applies to when your new document fails the collection's JSON schema, or when your requested Write Concern is not satisfied. The driver just punts it over to the application. It is up to the developer to decide what is the most appropriate course of action for these errors.

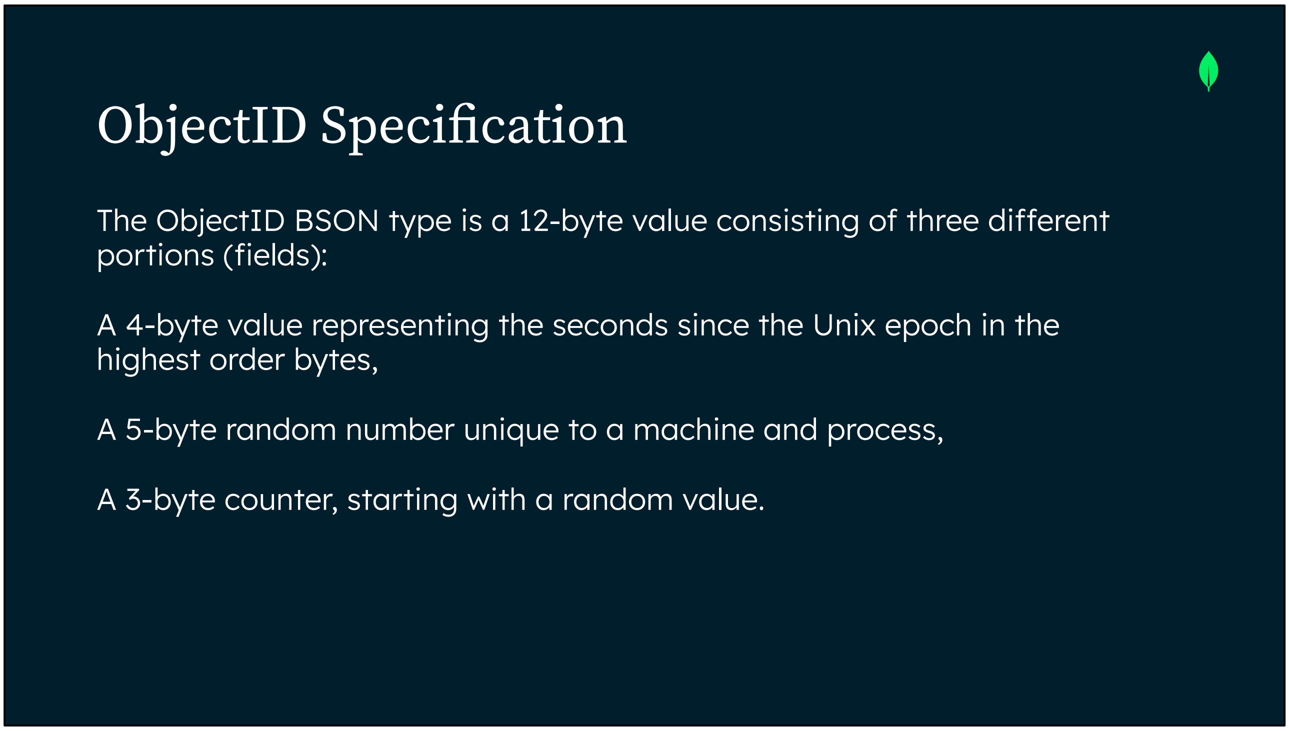


Driver Specification

MongoDB Drivers all conform to the same specification. This mandates that all drivers handle the same requests and operations.

All MongoDB Drivers must conform to the same specification and ensures that they handle requests and operations in the same way.

Let’s look at the specification for the ObjectID, the link for the URL is https://github.com/mongodb/specifications/blob/master/source/objectid.rst and we’ll look at some aspects of the specification in the next slides.



ObjectID Specification

The ObjectID BSON type is a 12-byte value consisting of three different portions (fields):

A 4-byte value representing the seconds since the Unix epoch in the highest order bytes,

A 5-byte random number unique to a machine and process,

A 3-byte counter, starting with a random value.

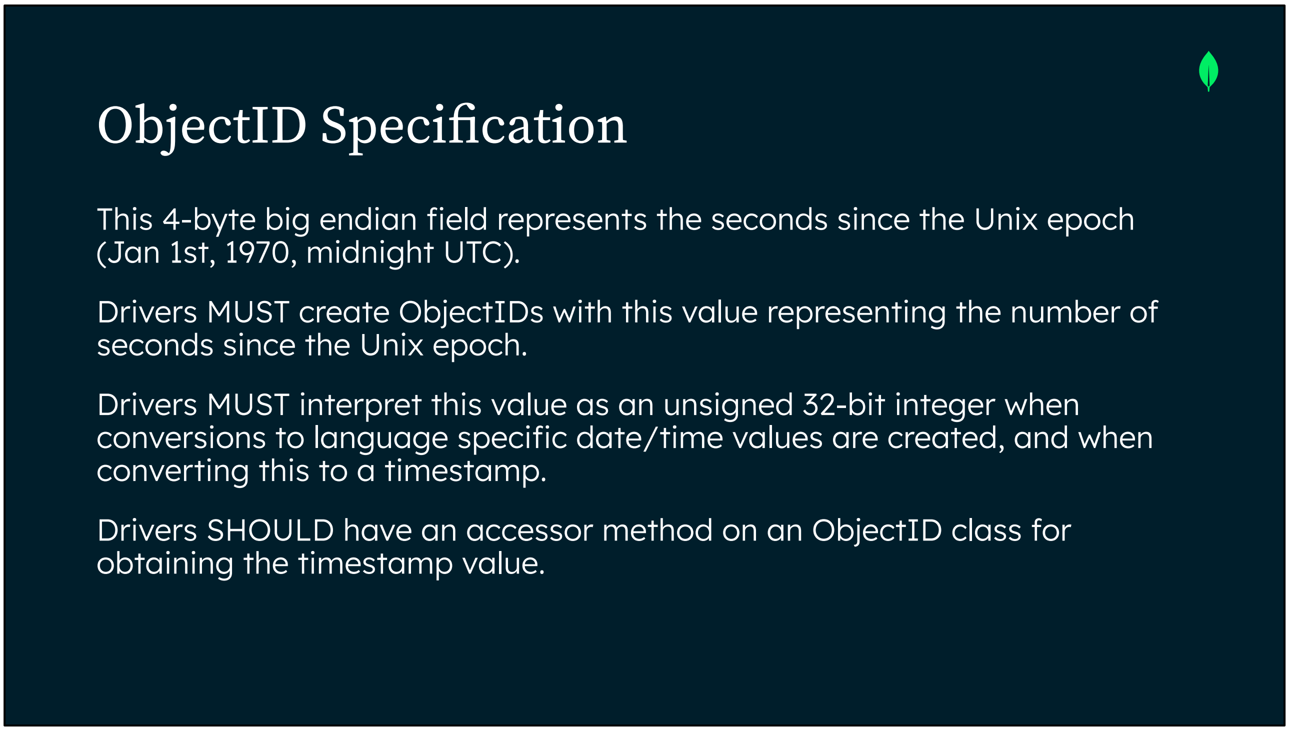
Firstly the ObjectID is a BSON type itself. It is a 12-byte value which consists of three different parts or fields.

The first part is a 4-byte value which represents the seconds since the Unix epoch in the highest order bytes.

The second field is a 5-byte random number which is unique the machine and the process (the mongod instance).

The third part is a 3-byte counter which starts with a random value.

Together, these three parts make up the complete ObjectID.



ObjectID Specification

This 4-byte big endian field represents the seconds since the Unix epoch (Jan 1st, 1970, midnight UTC).

Drivers MUST create ObjectIDs with this value representing the number of seconds since the Unix epoch.

Drivers MUST interpret this value as an unsigned 32-bit integer when conversions to language specific date/time values are created, and when converting this to a timestamp.

Drivers SHOULD have an accessor method on an ObjectID class for obtaining the timestamp value.

Looking at a little more detail in the specification, we find a number of pointers and aspects that if we were coding a driver we would need to support. Let’s look at the timestamp specification aspect of the ObjectID.

It indicates that the Unix epoch is Jan 1st, 1970, midnight UTC.

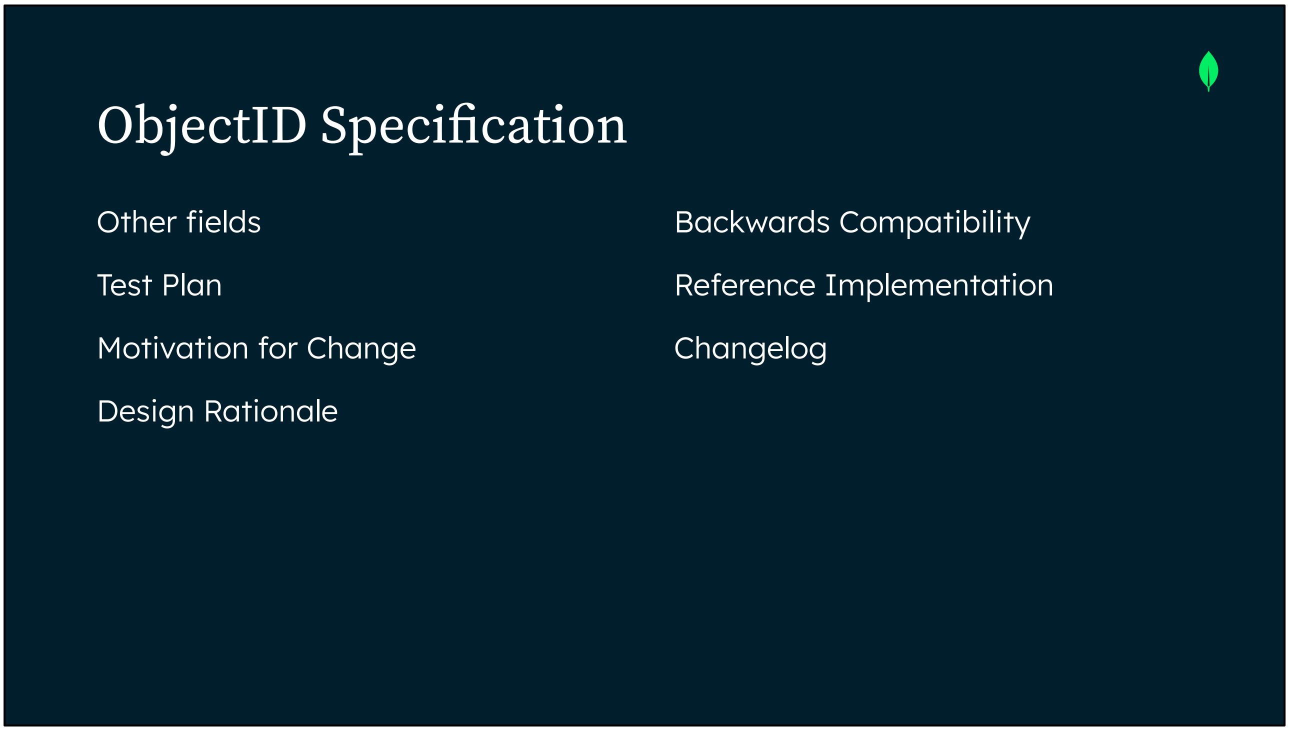
“Drivers MUST” means that this is a required part of the specification and must be implemented. The “Drivers SHOULD” means that this is an optional requirement.

In the case of the ObjectID Specification regarding the timestamp.

It requires that a driver must create ObjectIDs with the timestamp value only being since the Unix epoch.

It requires drivers to interpret this value only as an unsigned 32-bit integer when converting it to or from other variables/types.

It indicates that best practice for a driver is to allow an accessor method on the ObjectID class to allow the timestamp value to be programmatically obtained.



ObjectID Specification

Other fields

Test Plan

Motivation for Change

Design Rationale

Backwards Compatibility

Reference Implementation

Changelog

The ObjectID specification goes to similar depth in other aspects including:

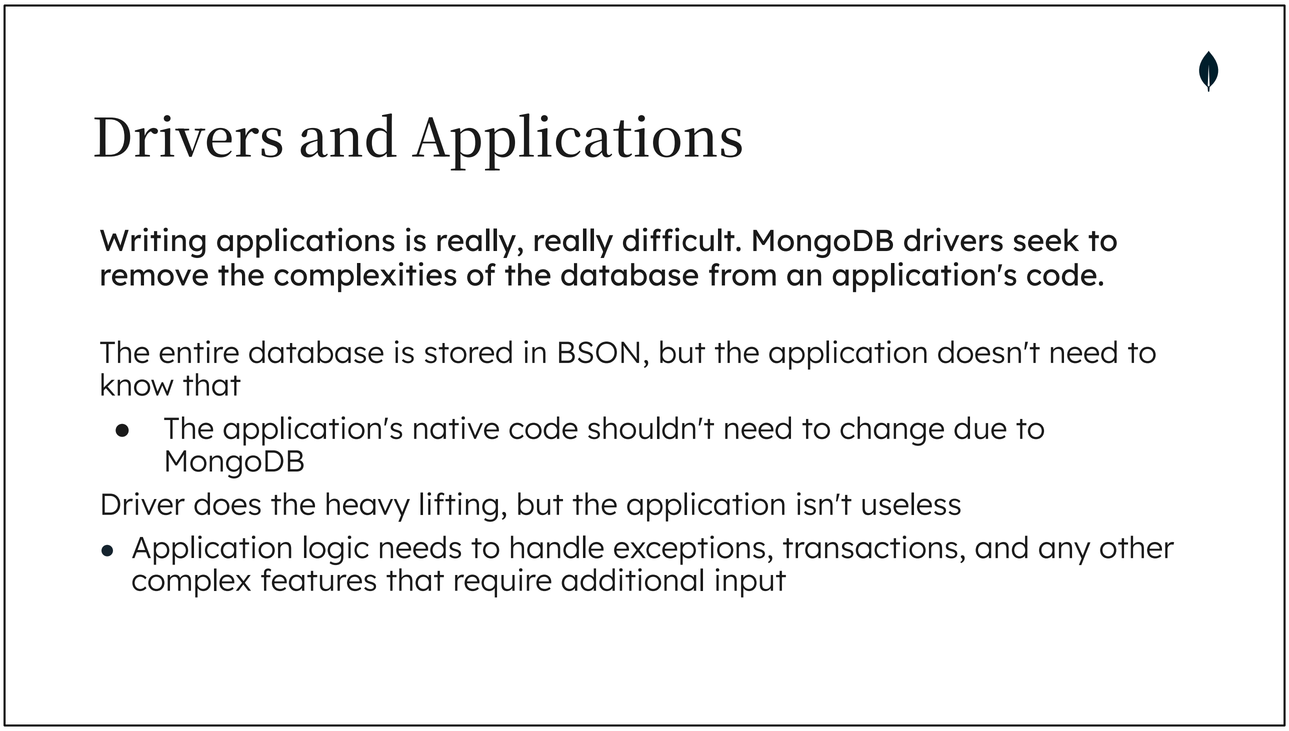
The other fields, specifically the random number and the counter are covered in similar depth.

It outlines how this (the ObjectID) should be tested and the plan around this.

It highlights the motivation for change (the ObjectID) as this occurred for this type. The ObjectID once was absolutely unique and now it is mostly but not guaranteed to be unique on a single machine.

The specification details the design rationale, the backwards compatibility as well as providing a reference implement. The specification also includes a changelog of the various changes to it.

This is only one of tens of different specifications that a driver must comply with to interact with MongoDB. The provision of these specifications ensures that it is possible to develop drivers that will interact with any MongoDB, indeed there are a number of open source drivers written by community members for various programming languages beyond those supported directly by MongoDB as a company.



Drivers and Applications

Writing applications is really, really difficult. MongoDB drivers seek to remove the complexities of the database from an application's code.

The entire database is stored in BSON, but the application doesn't need to know that

- The application's native code shouldn't need to change due to MongoDB

Driver does the heavy lifting, but the application isn't useless

- Application logic needs to handle exceptions, transactions, and any other complex features that require additional input

Writing an application is really, really difficult. The MongoDB drivers remove some of the database complexities and abstract/remove these from an application code. The goal is to help the developer write their code rather than having them code large chunks of essentially common functionality that other applications would similarly need/use.

The entire database is actually stored in BSON, the driver hides this and related storage details from the application and from the developer. An application doesn't need to know or deal with those storage details.

Changes in MongoDB, for instance when a new version is released, should not require changes in the application code for the most part. Obviously if a new feature is introduced this is different but in general, applications written once with a MongoDB driver shouldn't require any major changes unless to take advantage of new features within MongoDB.

So just to recap, the driver is meant to be an application's interface to the database. All the complexities around connecting to the server, and interacting with data that's natively in BSON are handled by the driver.

However, there are still responsibilities held by the code in the application.

All the transaction and session logic still needs to be handled on the application side, and any client-side encryption needs to be configured correctly.

Quiz

Quiz

Which of the following are true for any MongoDB Driver? More than one answer choice can be correct.

- A. Provides functionality to application to connect to the database
- B. Issues DML requests to the MongoDB database
- C. Handles the exceptions within the driver
- D. Must comply with Driver Specifications

Quiz

Which of the following are true for any MongoDB Driver? More than one answer choice can be correct.

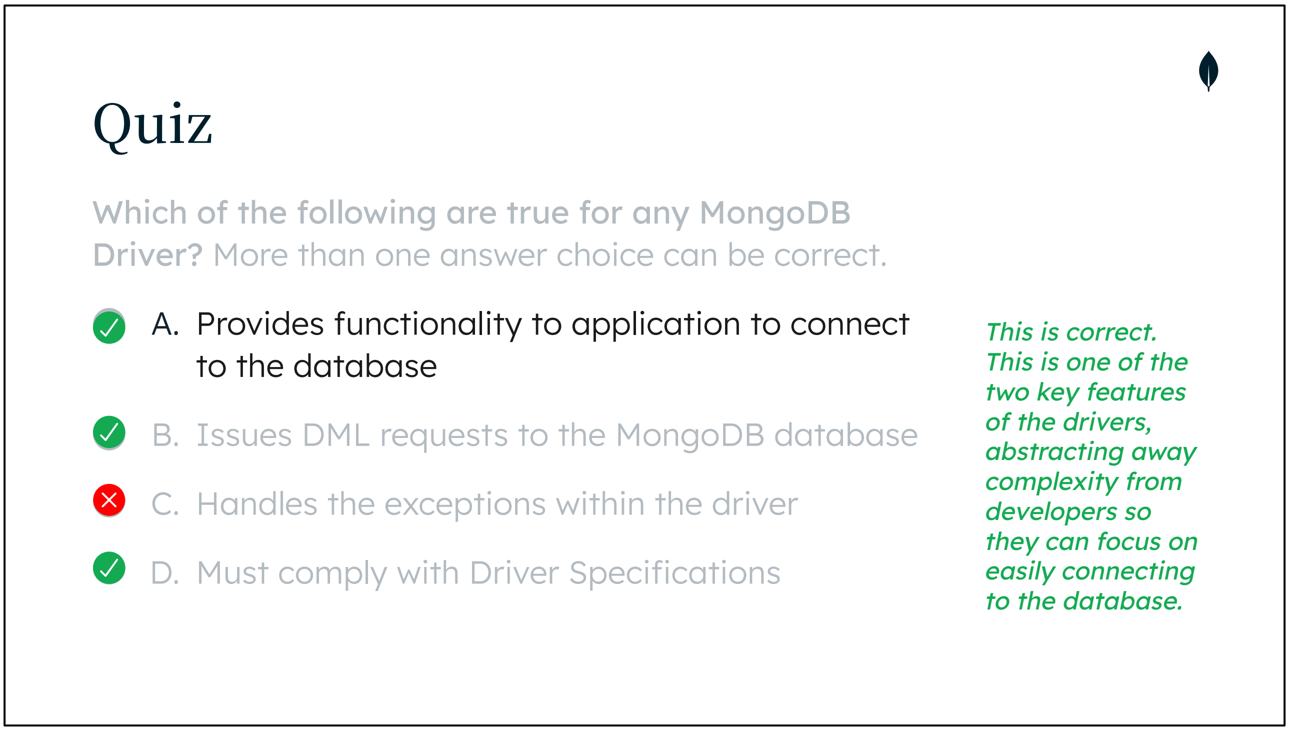
✓ A. Provides functionality to application to connect to the database

✓ B. Issues DML requests to the MongoDB database

✗ C. Handles the exceptions within the driver

✓ D. Must comply with Driver Specifications

CORRECT: Provides functionality to application to connect to the database - This is one of the two key features of the drivers, abstracting away complexity from developers so they can focus on easily connecting to the database.
CORRECT: Issues DML requests to the MongoDB database - This is the second main feature or function of a driver which is to create and manage any requests made from the application to the database.
INCORRECT: Handles the exceptions within the driver. - This is incorrect as the application must handle the exceptions passed from the driver.
CORRECT: Must comply with Driver Specifications. - This is correct as otherwise the functionality will not operate in the expected manner with the database and in all likelihood won't be accepted for processing by the database or could result in data corruption.



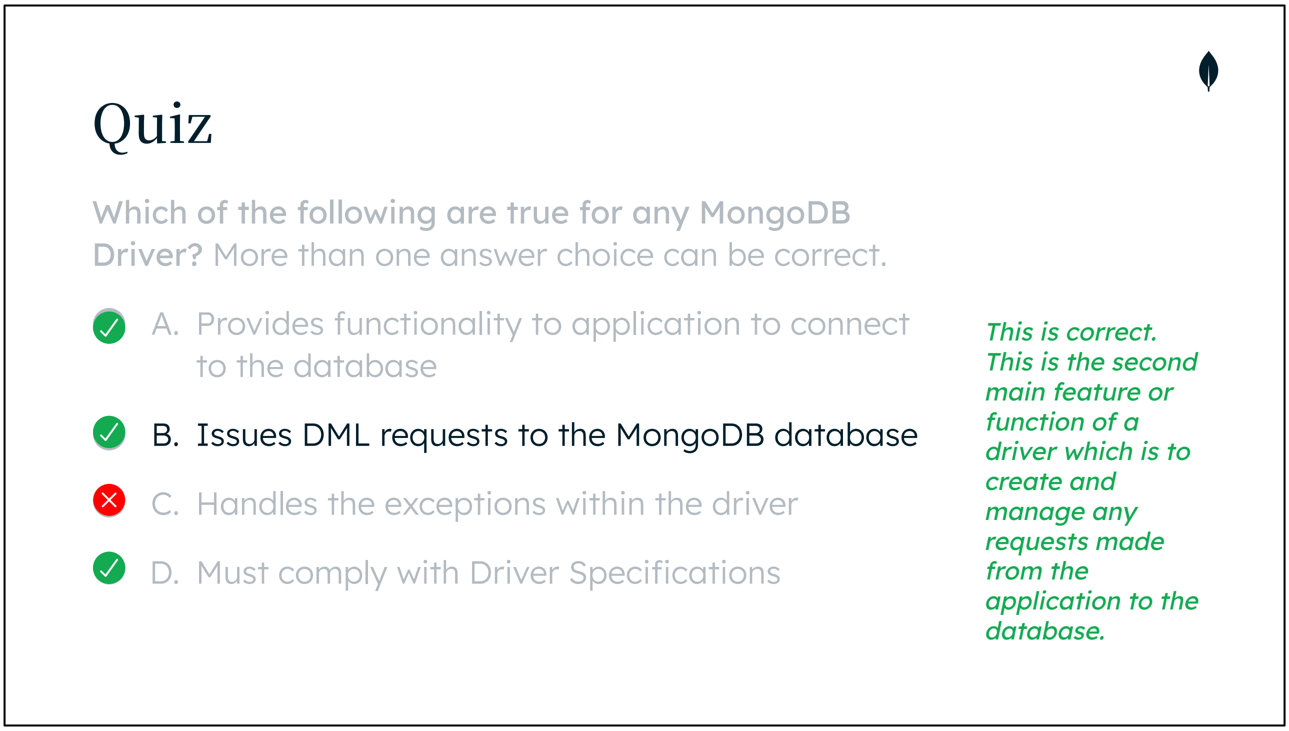
Quiz

Which of the following are true for any MongoDB Driver? More than one answer choice can be correct.

- ✓ A. Provides functionality to application to connect to the database
- ✓ B. Issues DML requests to the MongoDB database
- ✗ C. Handles the exceptions within the driver
- ✓ D. Must comply with Driver Specifications

This is correct. This is one of the two key features of the drivers, abstracting away complexity from developers so they can focus on easily connecting to the database.

CORRECT: Provides functionality to application to connect to the database - This is one of the two key features of the drivers, abstracting away complexity from developers so they can focus on easily connecting to the database.



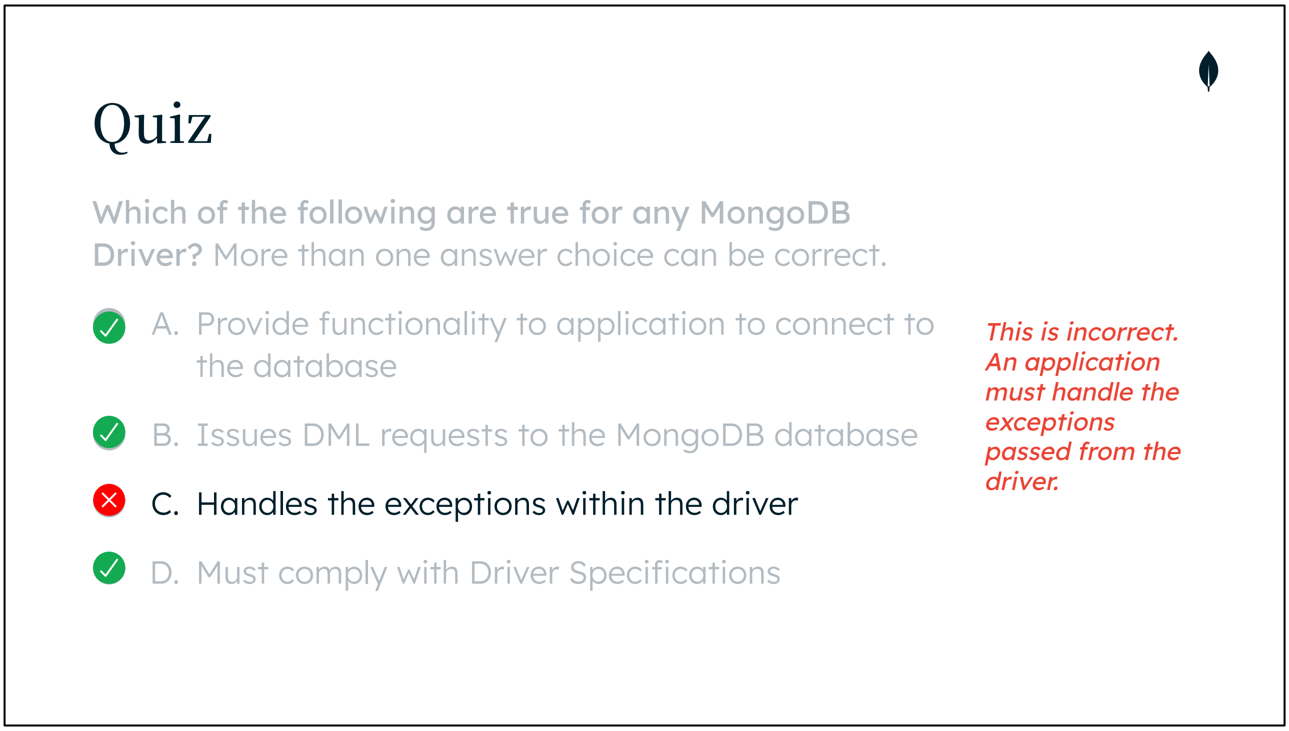
Quiz

Which of the following are true for any MongoDB Driver? More than one answer choice can be correct.

- ✓ A. Provides functionality to application to connect to the database
- ✓ B. Issues DML requests to the MongoDB database
- ✗ C. Handles the exceptions within the driver
- ✓ D. Must comply with Driver Specifications

This is correct. This is the second main feature or function of a driver which is to create and manage any requests made from the application to the database.

CORRECT: Issues DML requests to the MongoDB database - This is correct. This is the second main feature or function of a driver which is to create and manage any requests made from the application to the database.



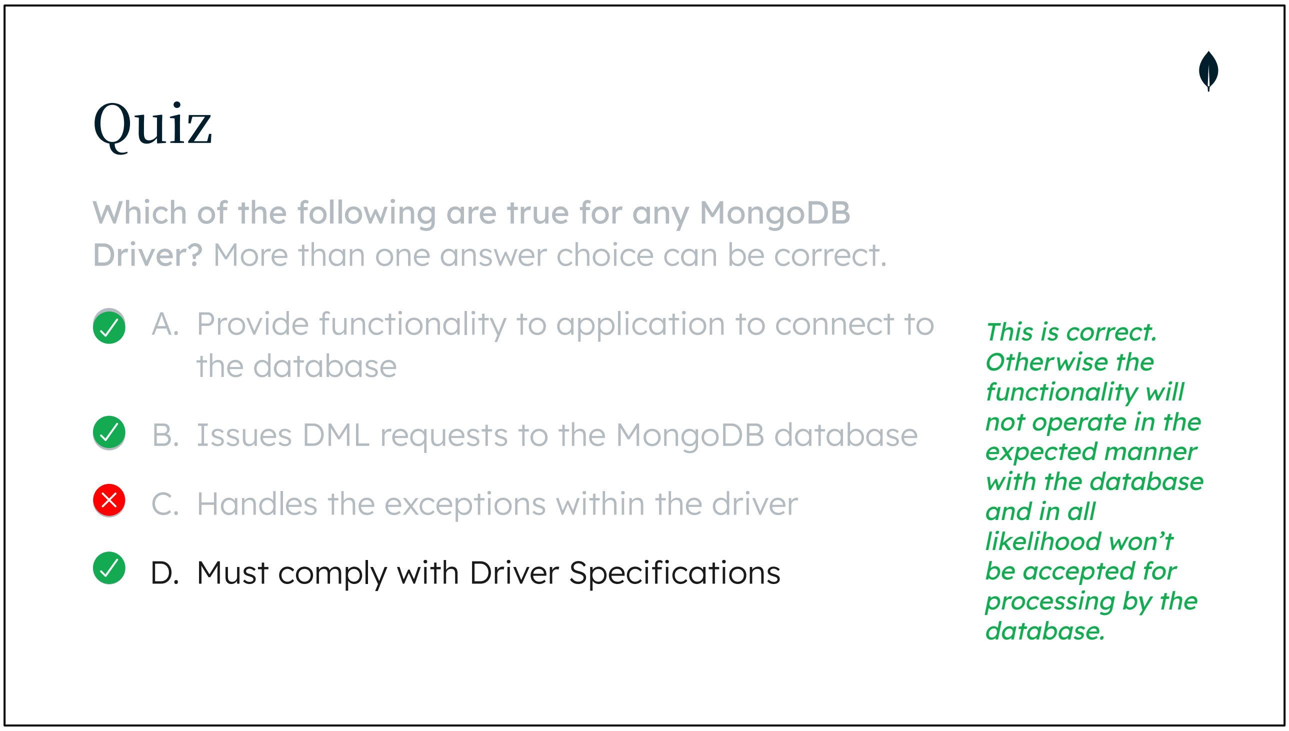
Quiz

Which of the following are true for any MongoDB Driver? More than one answer choice can be correct.

- ✓ A. Provide functionality to application to connect to the database
- ✓ B. Issues DML requests to the MongoDB database
- ✗ C. Handles the exceptions within the driver
- ✓ D. Must comply with Driver Specifications

This is incorrect. An application must handle the exceptions passed from the driver.

CORRECT: Issues DML requests to the MongoDB database - This is correct. This is the second main feature or function of a driver which is to create and manage any requests made from the application to the database.



Quiz

Which of the following are true for any MongoDB Driver? More than one answer choice can be correct.

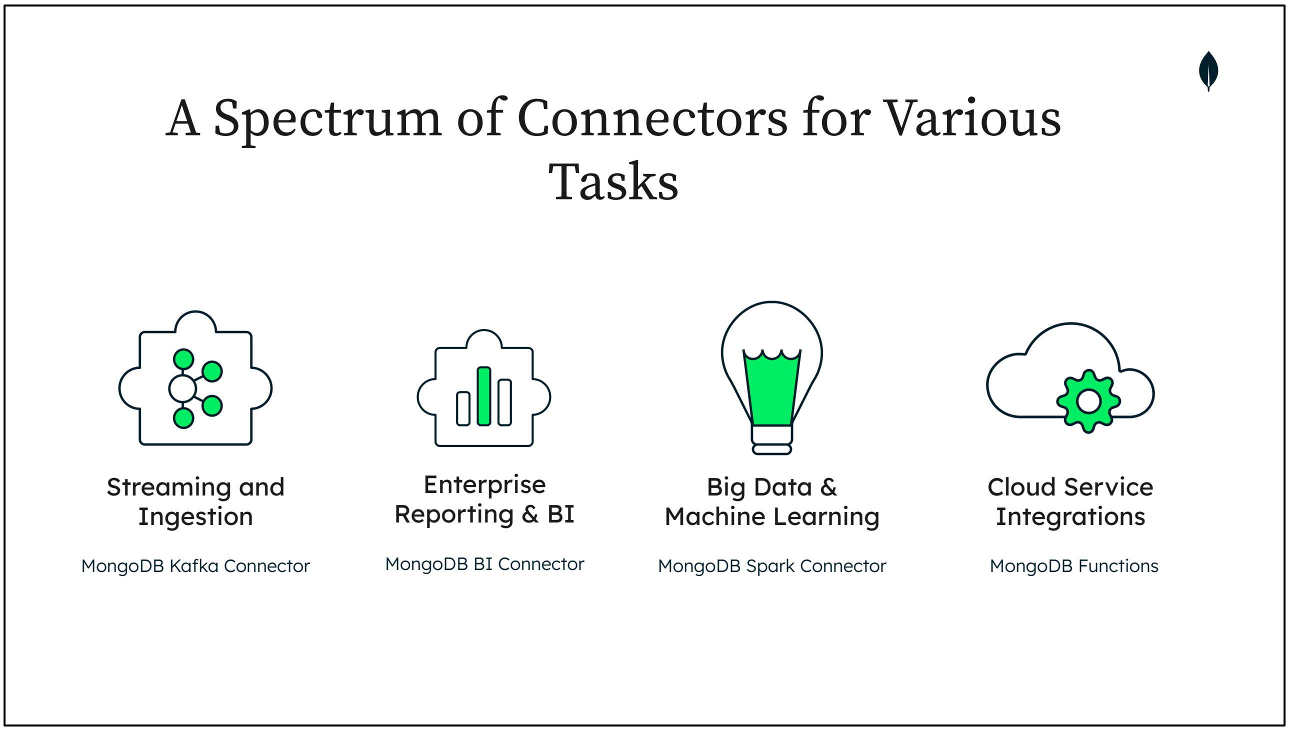
- ✓ A. Provide functionality to application to connect to the database
- ✓ B. Issues DML requests to the MongoDB database
- ✗ C. Handles the exceptions within the driver
- ✓ D. Must comply with Driver Specifications

This is correct. Otherwise the functionality will not operate in the expected manner with the database and in all likelihood won't be accepted for processing by the database.

CORRECT: Must comply with Driver Specifications. - This is correct as otherwise the functionality will not operate in the expected manner with the database and in all likelihood won't be accepted for processing by the database or could result in data corruption.

Connectors

Beyond the database itself, there may be requirements to have interoperability with other software systems. MongoDB has developed a number of software connectors to enable this interoperability.



A Spectrum of Connectors for Various Tasks

Streaming and Ingestion

MongoDB Kafka Connector

Enterprise Reporting & BI

MongoDB BI Connector

Big Data & Machine Learning

MongoDB Spark Connector

Cloud Service Integrations

MongoDB Functions

MongoDB provides a range of connectors to various other software or platforms.

These connectors allow for integrations with popular tools for moving, transforming, and analyzing data.

We'll look at just four of the available connectors in this lesson.

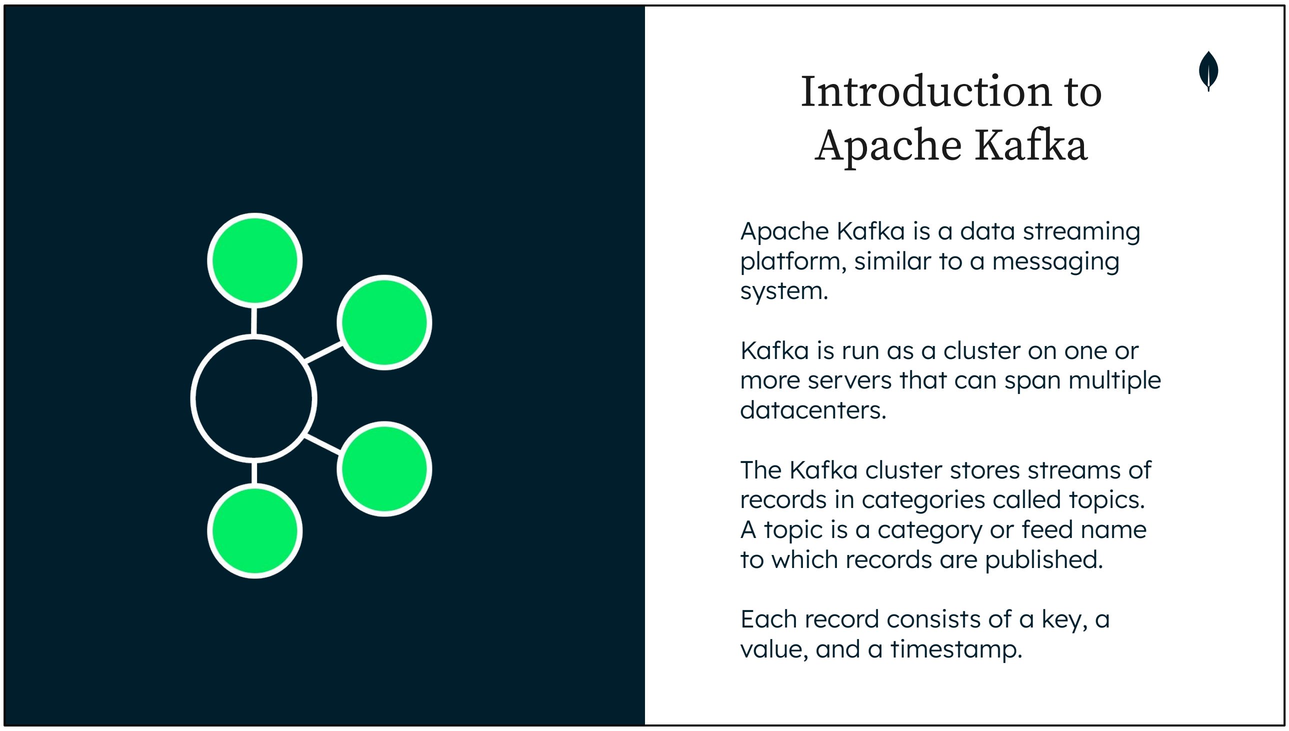
Firstly, we'll look at the Kafka tool which is used for streaming and ingesting data.

Secondly, we'll look at the MongoDB BI Connector which allows MongoDB to connect to a wide variety of Business Intelligence tools.

Thirdly, we'll look at the MongoDB Spark Connector which provides a means to connect MongoDB to Spark and the wider Hadoop ecosystem.

Fourthly and finally, we'll look at MongoDB Functions which allow for connections to the MongoDB serverless platform, which includes server-side processing for applications as well as triggers for real-time response with serverless functions that can transform data and integrate with third party and ecosystem tools.

Kafka Connector



Introduction to Apache Kafka

Apache Kafka is a data streaming platform, similar to a messaging system.

Kafka is run as a cluster on one or more servers that can span multiple datacenters.

The Kafka cluster stores streams of records in categories called topics. A topic is a category or feed name to which records are published.

Each record consists of a key, a value, and a timestamp.

Apache Kafka is a popular and widely used data streaming system, it has similarities to a messaging system. The project's web page is https://kafka.apache.org/

It is run on a cluster and can span multiple datacenters.

It processes streams of records in categories know as topics. Each topic is the category or feed name to which records are published.

Each record consists of a key, a value and a timestamp.

Why Connect to Kafka

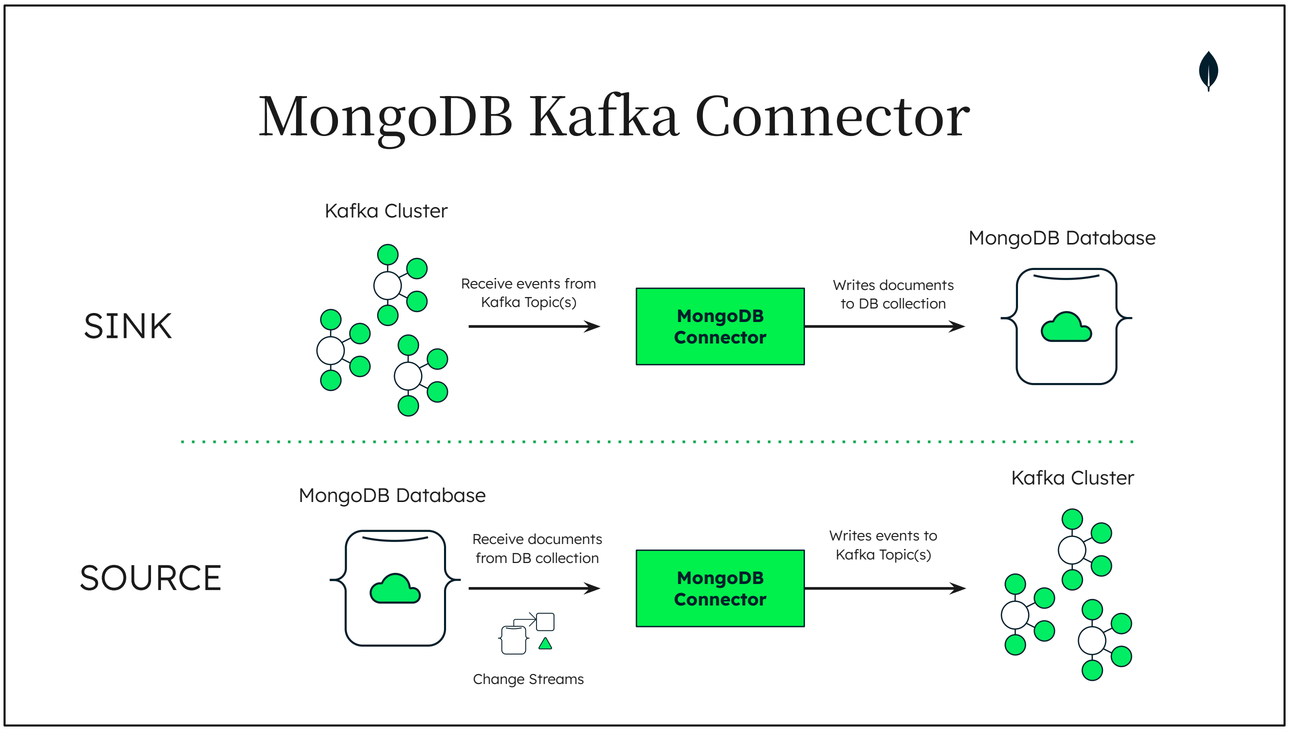
- Allows easy building of robust and reactive data pipelines that take advantage of stream processing between datastores, applications, and services in real-time.
- Eliminates the need for writing boilerplate integration code and ensures end-to-end data exchange between any source and sink.
- Developers can focus their time on efficient querying, data enrichment, and analytics.
- The connector enables MongoDB deployments to easily interface with other data technologies via Kafka topics and messages.

Kafka allows data pipelines to be built. Allows to easily build robust and reactive data pipelines that take advantage of stream processing between datastores, applications and services in real-time. These pipeline that allow for real-time streaming processing between datastores (such as MongoDB), applications, and other services.

The MongoDB Kafka Connector avoids the need for developers to write boilerplate integration code. It further ensures end-to-end data exchange between any source and any sink.

This allows developers to focus on more efficient querying, data enrichment, and analytics of the data being processed. This means that rather than writing integration code developers are instead focusing on important business problems that provide real value.

The connector is the key piece of software that supports the interoperability between MongoDB and Kafka and as well as other data technologies downstream from Kafka.



The MongoDB Kafka Connector was mentioned in our Change Streams lesson. We mentioned how it can be useful to highlight how we use change streams in a real product.

Let's look at Kafka itself and specifically at the MongoDB Kafka Connector. There are two main functionalities that it supports in conjunction with Kafka.

Firstly, it can be a SINK. A sink means that it will receive data from Kafka into the database.

Alternatively, it can be a SOURCE. A source means that MongoDB will send data to Kafka from the database.

The MongoDB Kafka Connector is integrated with the Confluent Kafka product through their Kafka Connect API. This allows Kafka users to easily integrate MongoDB into their workflows.

Kafka Sink

Apache Kafka uses a **sink** connector to consume records from a *topic* and save the data to a *datastore* such as MongoDB.

Load events from Kafka topics directly into MongoDB collections to support queries and analytics.

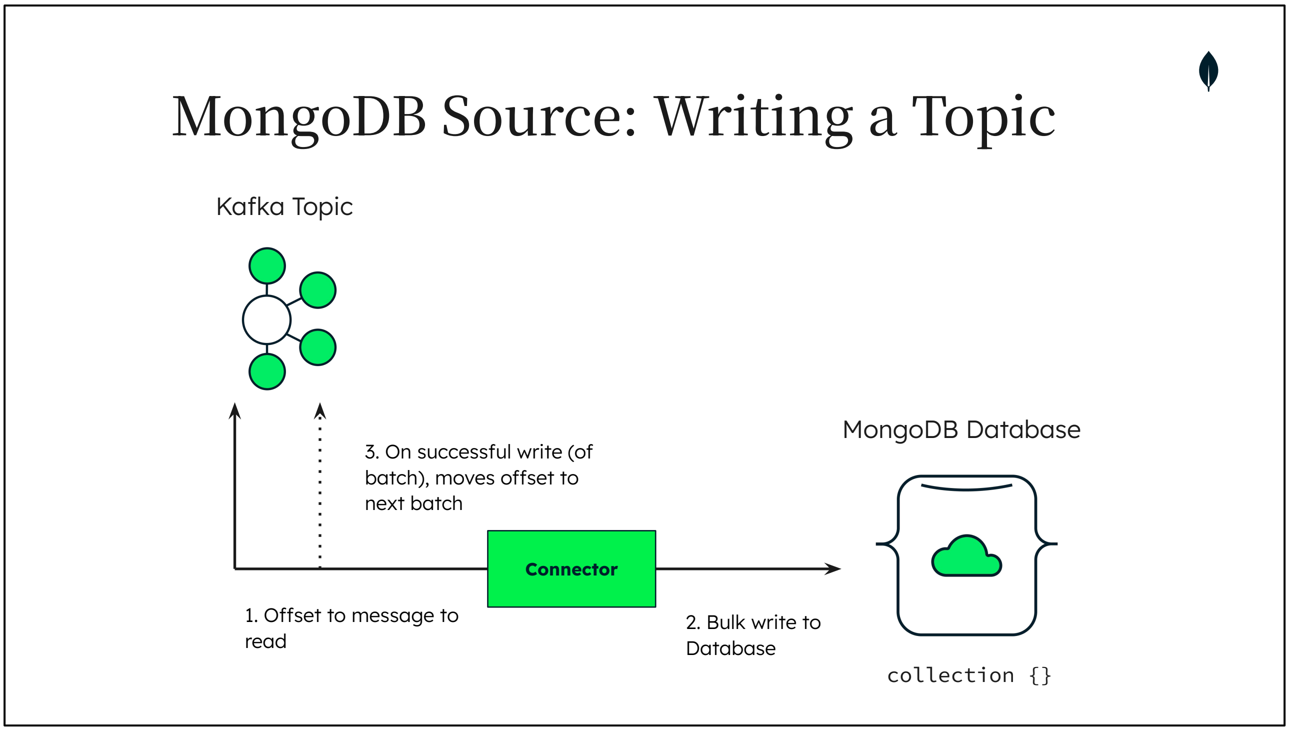
Kafka can be used as both a source and a sync in different parts of your application architecture:

- Data can be consolidated from sources
- Data can then be stored in MongoDB (sink) for analysis

Apache Kakda uses a sink connector to consume records from a topic and then save them to another data store such as MongoDB. The connector will convert the value in the Kafka Connector's Sink Record firstly to a MongoDB document and it will then perform either an insert or an upsert deping on the configuration that has been chosen.

This loading of events from Kafka helps support queries and analytics use cases. Specifically, by moving the event data into MongoDB it opens the full ecosystem of tooling that supports MongoDB to help with querying the data or visualising it.

Kafka can be used as both a sink and a source, for example some customers use Kafka to write data from multiple sources and consolidate it as Kafka sources. They then use MongoDB as a Kafka sink to store the consolidated data and analyse it.



MongoDB Source: Writing a Topic

Kafka Topic
3. On successful write (of batch), moves offset to next batch
Connector
MongoDB Database
1. Offset to message to read
2. Bulk write to Database
collection {}

Let’s look at how a MongoDB Sink works with Kafka to read messages from a topic.

Firstly it reads the messages from the topic using a pointer to a specific message in the topic.

Secondly, it then writes this message back into the MongoDB collection.

Thirdly, it moves the pointer to the next message in the topic in Kafka so it’s ready to read the next message in the topic.

Let’s recap that in the diagram.

It reads the message based on the offset in the topic.
It writes using the bulk write API to add this message to the MongoDB collection.
If it is successfully written to the MongoDB collection, the offset is moved to the next batch in Kafka to be read.

MongoDB Sink: Advanced Options

The number of messages written in a batch can be controlled by the maximum batch size variable

The number of retries and retry timeout can be specified to control how often on a error the sink should retry and how long to wait between these retries.

Field renaming, key and value deny lists as well as allow lists can also be configured as part of the post processing performed by the MongoDB Sink.

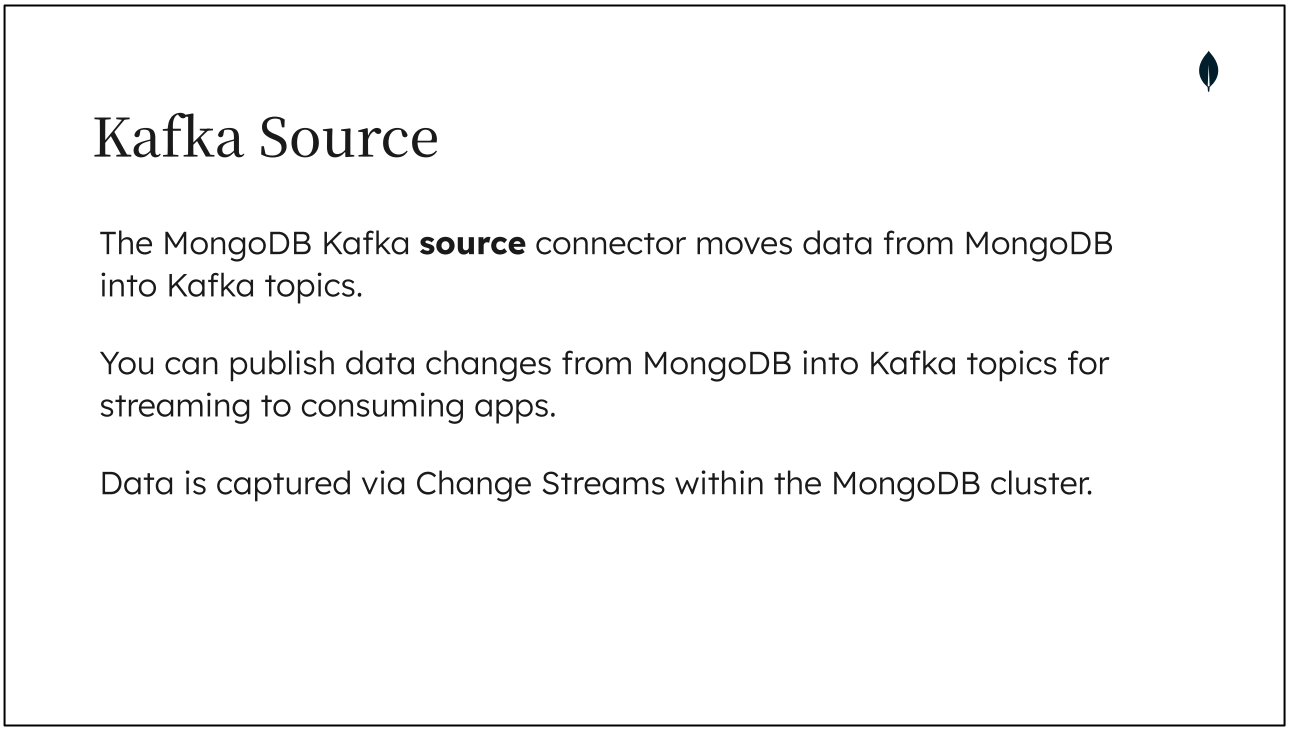
There are a number of advanced options that can be configured with the Kafka Connector when it is operating as a MongoDB Sink.

Firstly, the number of message written in a single batch is configurable by the maximum batch size variable.

Secondly, the number retries and retry timeout can be configured to control how often on a error the sink should retry and how long to wait between those retries.

Finally, the field name, key and value lists as well as allow lists can be configured for the post processing for security.

https://docs.mongodb.com/kafka-connector/current/kafka-sink-properties/



Kafka Source

The MongoDB Kafka **source** connector moves data from MongoDB into Kafka topics.

You can publish data changes from MongoDB into Kafka topics for streaming to consuming apps.

Data is captured via Change Streams within the MongoDB cluster.

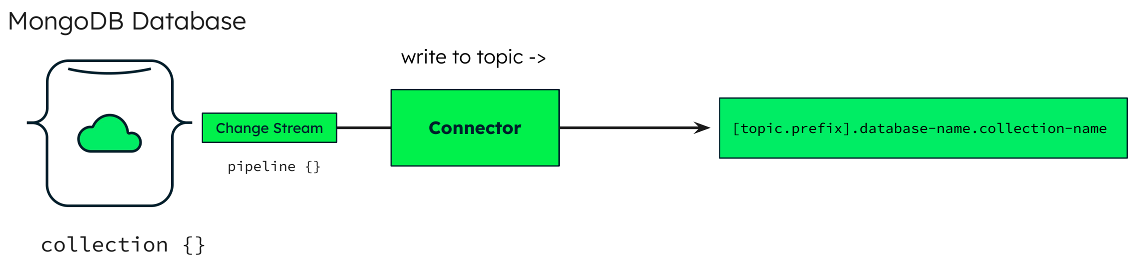
The second type of functionality offered by the MongoDB Kafka connector is to operate as a Kafka source. This is where data is sent from MongoDB to a Kafka topic.

The source connector functionality allows you to publish changes from MongoDB into Kafka and it is setup to support streaming of these changes to Kafka.

It streams the changes and data from MongoDB using Change Streams.

https://docs.mongodb.com/kafka-connector/current/kafka-sink-properties/

MongoDB Source: Writing a Topic



- Writes to topic based on database and collection name
- Optionally specify pipeline to manage change stream events to watch
- Optionally set a topic.prefix in the connector configuration

The MongoDB Connector when used as a source allows for changes in MongoDB based on their database and collection name to be written to Kafka. This is achieved using the Change Stream functionality in MongoDB.

You can optionally specify the pipeline that controls the change stream events being watched.

You can also define a topic prefix in the connector configuration to control the destination for the data in Kafka.

MongoDB Source: Advanced Options

It is possible to subscribe to changes from entire cluster, single database, or single collection with configuring MongoDB as a source to Kafka.

The MongoDB Source connector can be configured to:

Include the full document with the message

To copy existing data from source collections

It is possible to configure that the changes from an entire cluster, a single database or indeed just a single collection can be configured by the MongoDB Connector to be sent to Kafka.

The MongoDB Source connector can be configured to include the full document with the message and it can also copy existing data from source collections in MongoDB to Kakfa.

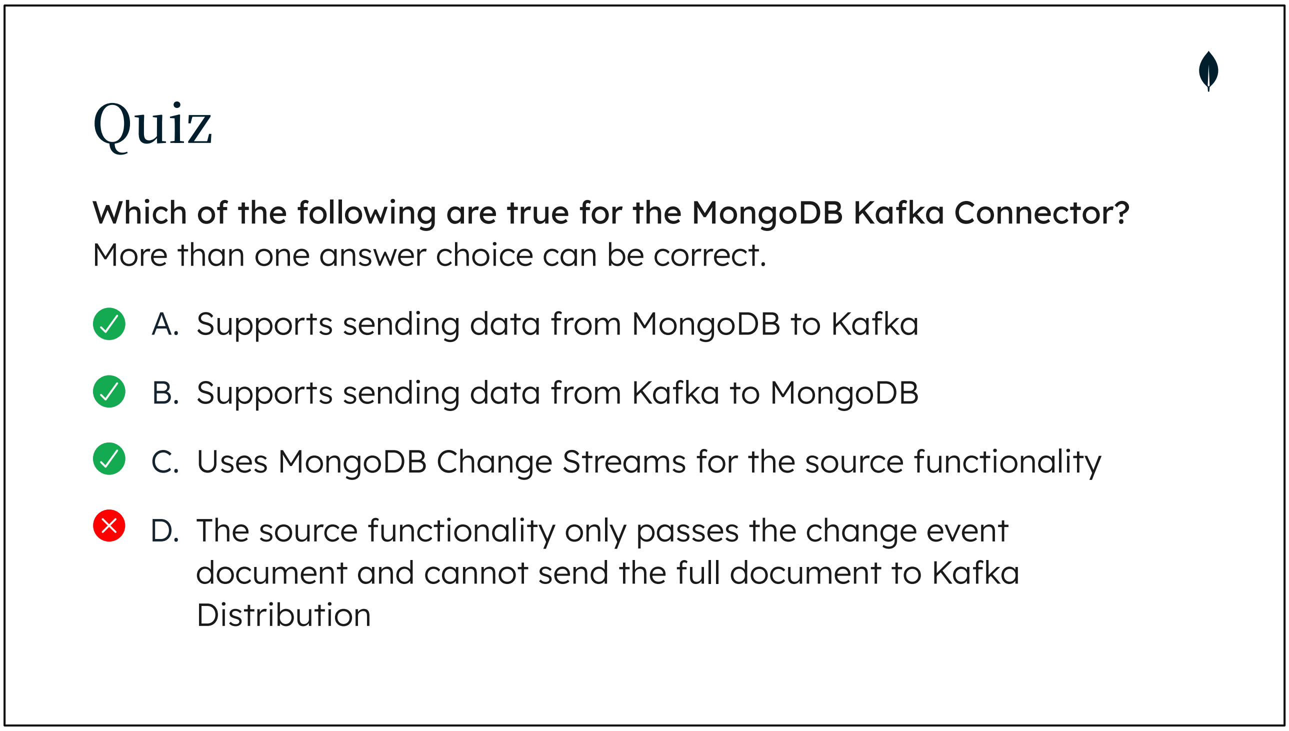
There are more details on this web page for these options
https://docs.mongodb.com/kafka-connector/current/kafka-source/

Quiz

Quiz

Which of the following are true for the MongoDB Kafka Connector?
More than one answer choice can be correct.

- A. Supports sending data from MongoDB to Kafka
- B. Supports sending data from Kafka to MongoDB
- C. Uses MongoDB Change Streams for the source functionality
- D. The source functionality only passes the change event document and cannot send the full document to Kafka Distribution



Quiz

Which of the following are true for the MongoDB Kafka Connector?
More than one answer choice can be correct.

✓ A. Supports sending data from MongoDB to Kafka

✓ B. Supports sending data from Kafka to MongoDB

✓ C. Uses MongoDB Change Streams for the source functionality

✗ D. The source functionality only passes the change event document and cannot send the full document to Kafka Distribution

CORRECT: Supports sending data from MongoDB to Kafka. This is correct. There is bi-directional communication from and to MongoDB with Kafka. MongoDB has implement source connector functionality to provide this functionality.
CORRECT: Supports sending data from Kafka to MongoDB. This is correct.There is bi-directional communication from and to MongoDB with Kafka. MongoDB has implement sink connector functionality to provide this functionality.
CORRECT: Uses MongoDB Changes Streams for the source functionality. This is correct. Changes Streams allow for changes in real time to be sent to Kafka when changes occur.
INCORRECT: The source functionality only passes the change event document and cannot send the full document to Kafka. This is incorrect. The complete document can optionally be configured to be sent to Kafka.

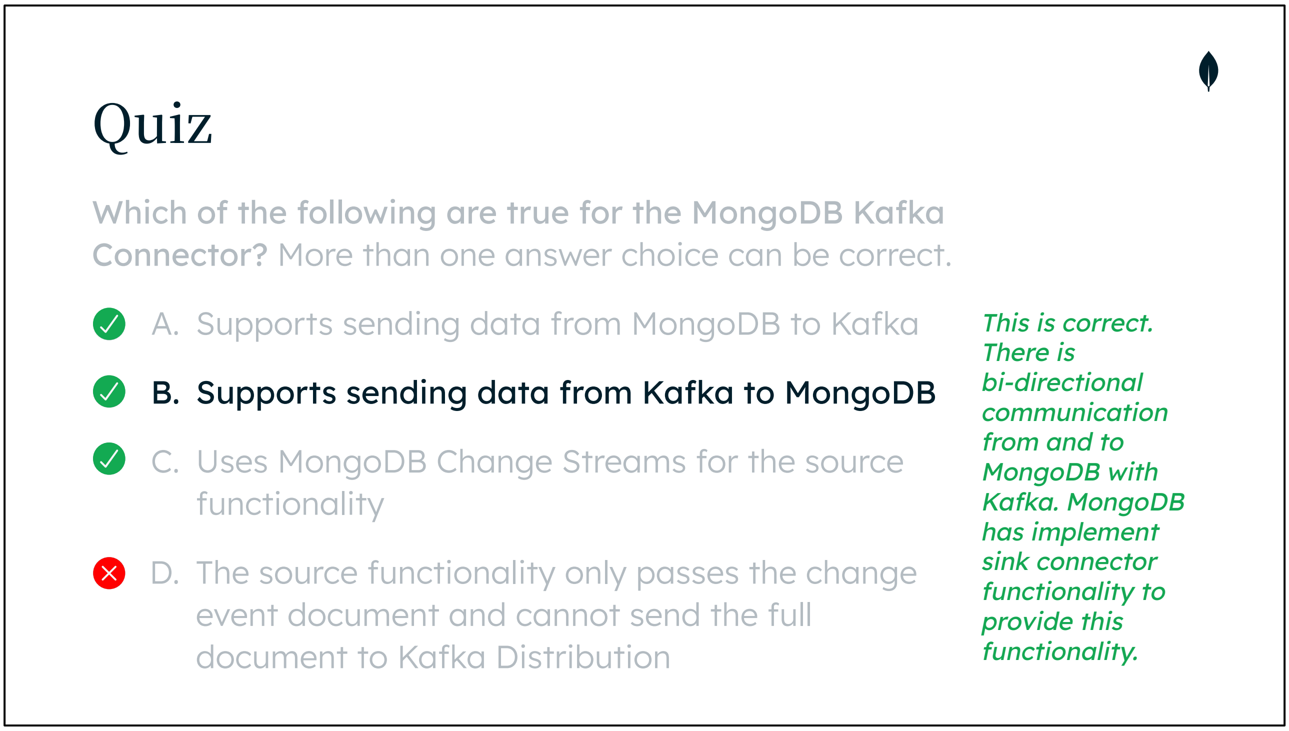
Quiz

Which of the following are true for the MongoDB Kafka Connector? More than one answer choice can be correct.

- ✓ A. **Supports sending data from MongoDB to Kafka**
- ✓ B. Supports sending data from Kafka to MongoDB
- ✓ C. Uses MongoDB Change Streams for the source functionality
- ✗ D. The source functionality only passes the change event document and cannot send the full document to Kafka Distribution

This is correct. There is bi-directional communication from and to MongoDB with Kafka. MongoDB has implement source connector functionality to provide this functionality.

CORRECT: Supports sending data from MongoDB to Kafka. This is correct. There is bi-directional communication from and to MongoDB with Kafka. MongoDB has implement source connector functionality to provide this functionality.



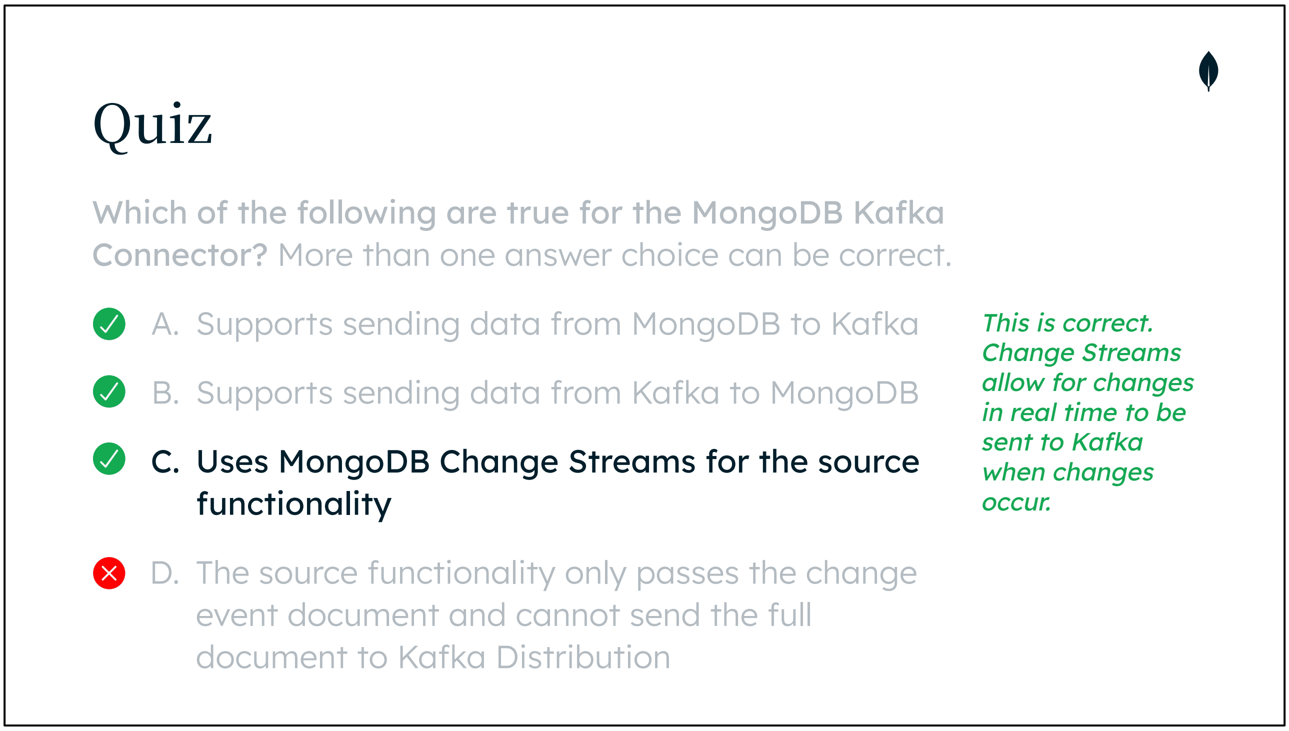
Quiz

Which of the following are true for the MongoDB Kafka Connector? More than one answer choice can be correct.

- ✓ A. Supports sending data from MongoDB to Kafka
- ✓ **B. Supports sending data from Kafka to MongoDB**
- ✓ C. Uses MongoDB Change Streams for the source functionality
- ✗ D. The source functionality only passes the change event document and cannot send the full document to Kafka Distribution

This is correct. There is bi-directional communication from and to MongoDB with Kafka. MongoDB has implement sink connector functionality to provide this functionality.

CORRECT: Supports sending data from Kafka to MongoDB. This is correct.There is bi-directional communication from and to MongoDB with Kafka. MongoDB has implement sink connector functionality to provide this functionality.



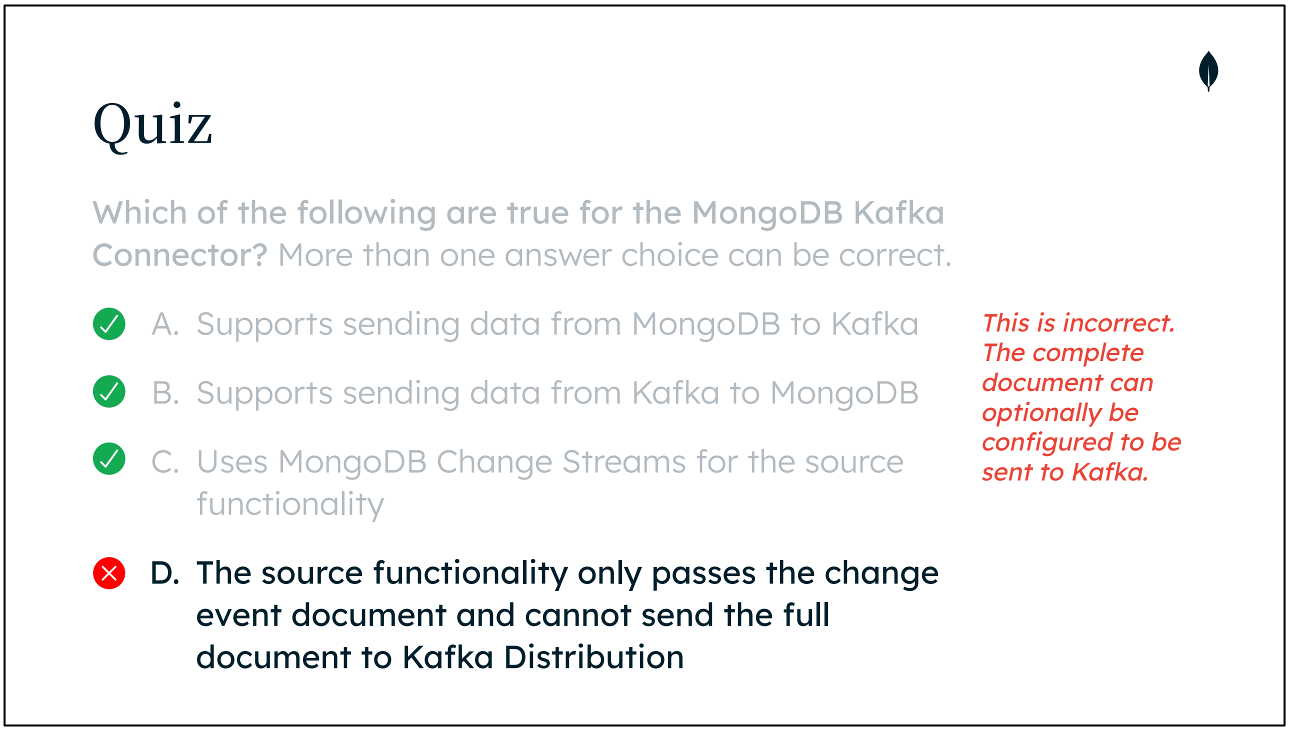
Quiz

Which of the following are true for the MongoDB Kafka Connector? More than one answer choice can be correct.

- ✓ A. Supports sending data from MongoDB to Kafka
- ✓ B. Supports sending data from Kafka to MongoDB
- ✓ **C. Uses MongoDB Change Streams for the source functionality**
- ✗ D. The source functionality only passes the change event document and cannot send the full document to Kafka Distribution

This is correct. Change Streams allow for changes in real time to be sent to Kafka when changes occur.

CORRECT: Uses MongoDB Change Streams for the source functionality. This is correct. Change Streams allow for changes in real time to be sent to Kafka when changes occur.



Quiz

Which of the following are true for the MongoDB Kafka Connector? More than one answer choice can be correct.

- ✓ A. Supports sending data from MongoDB to Kafka
- ✓ B. Supports sending data from Kafka to MongoDB
- ✓ C. Uses MongoDB Change Streams for the source functionality
- ✗ **D. The source functionality only passes the change event document and cannot send the full document to Kafka Distribution**

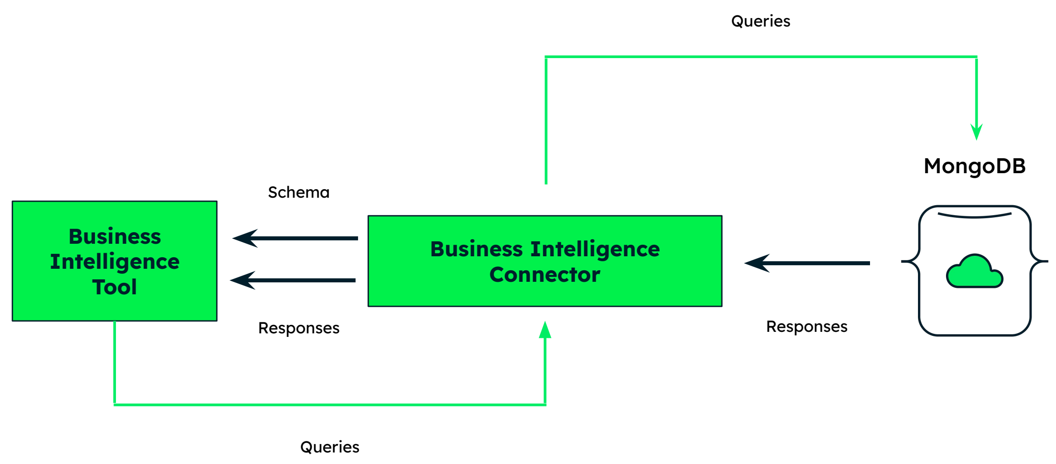
This is incorrect. The complete document can optionally be configured to be sent to Kafka.

INCORRECT: The source functionality only passes the change event document and cannot send the full document to Kafka. This is incorrect. The complete document can optionally be configured to be sent to Kafka.

MongoDB Connector for BI

Let’s start by discussing the MongoDB Connector for BI.

What Does the BI Connector Do?



The business intelligence connector sits between your BI tool and your MongoDB deployment. It's purpose is to translate from MongoDB to BI and vice-versa to allow data in your database to be analysed or visualised easily in whatever BI tool you use. The connector translates from SQL to MQL and vice versa.

The Business Intelligence tools provide a means for visualizing, for analyzing, and for reporting against a MongoDB database using SQL queries.

The BI Connector translates these queries from SQL to MQL.

This opens a wide variety of such as Tableau, MS Excel, Power BI and allows them to operate smoothly with MongoDB data.

The connector translates from SQL to MQL and vice versa.
The Business Intelligence tools provide a means for visualizing, for analyzing, and for reporting against a MongoDB database using SQL queries.

The BI Connector translates these queries from SQL to MQL.

This opens a wide variety of such as Tableau, MS Excel, Power BI and allows them to operate smoothly with MongoDB data.

Components

The connector for BI has 2 main components:

- Schema definition utility: ***mongodrdl***
- Connector daemon: ***mongosqldl***

There is also a related ODBC driver that provides connectivity between a SQL client and the MongoDB Connector for BI.

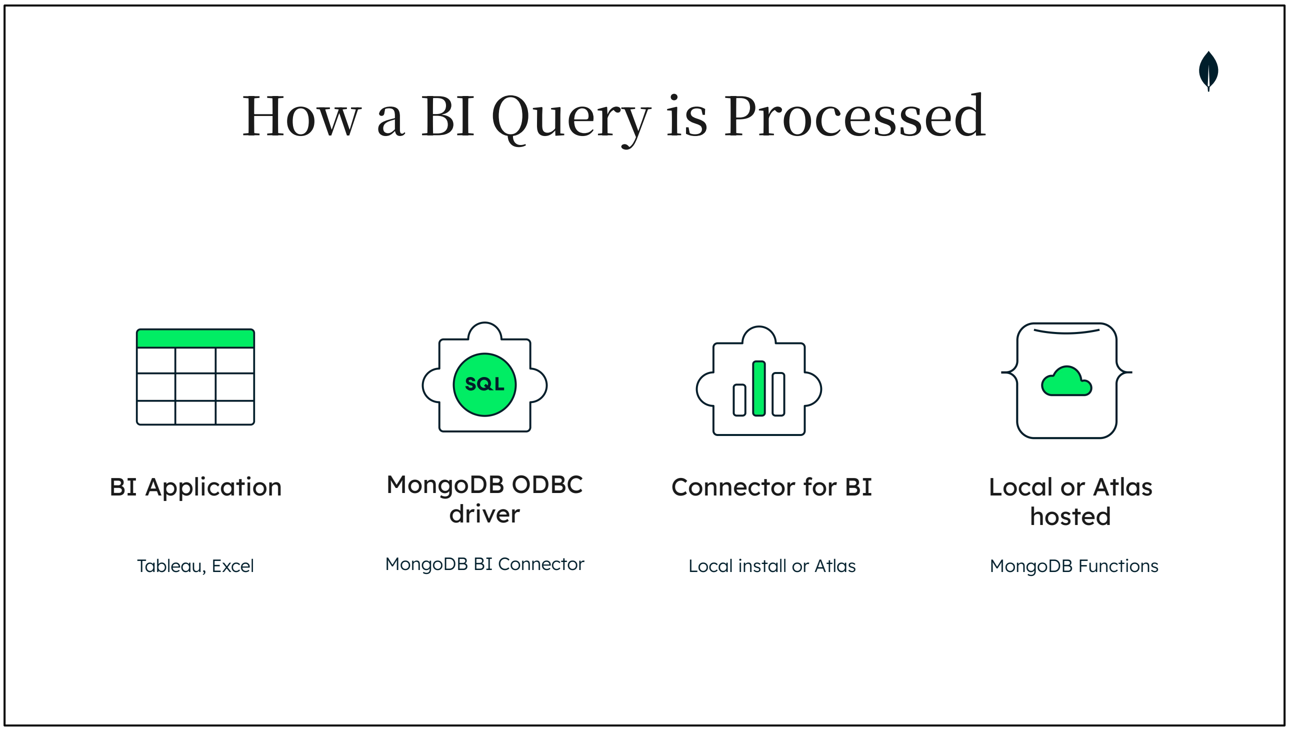
The connector is made up of two major components.

The first is a schema definition utility, *mongodrdl.* This utility program generates *drdl (*Document-Relation Definition Language) files from the databases and collections in a MongoDB deployment. It maps your documents to a simple relational schema.

The second is a daemon process, *mongosqld.* It runs as a server *daemon* and responds to incoming SQL queries. The BI Tool via the connector translates the SQL query and sends the MQL query to connected MongoDB deployment, using the schema in the .drdl file. It translates results from MongoDB back to a SQL format for the BI Tool.

In addition, there is an ODBC driver that is used to provide connectivity between a SQL client and the MongoDB BI Connector.

Additional, there is a component *mongotranslate*. This is used to translate SQL to MongoDB Aggregation Operations.



How a BI Query is Processed

BI Application

Tableau, Excel

MongoDB ODBC driver

MongoDB BI Connector

Connector for BI

Local install or Atlas

Local or Atlas hosted

MongoDB Functions

Let’s look at how a query from a BI Application is processed.

Firstly, we execute the query within our BI Application (e.g. Tableau, Excel, etc.).

This is converted into SQL by the BI Application and then processed by the MongoDB ODBC driver which manages the connectivity between BI applications and the MongoDB Connector for BI.

The Connector for BI then translates this SQL from the ODBC driver into MQL. It then executes the resulting MQL and processes the database results back into SQL. This SQL is passed back through the ODBC driver back to the BI Application.

BI Connector for Data Analysis

Use corporate BI tools to examine both MongoDB data and SQL data.

Analyze data generated by MongoDB's flexible data model alongside traditional data stored in relational databases.

Allows us to combine traditional SQL statements with MongoDB query expressions for data analysis.

Allows for data in MongoDB to be analyzed without the need for ETL processes.

The BI connector allows companies to use one set of analytical tooling for both their MongoDB databases and data in SQL/relational databases.

It provides a means to analyse both type of data despite the different in their schemas.

The ability to write traditional SQL and apply those queries against MongoDB is particularly useful for data analysts as they don't then also need to learn how to query MongoDB.

A big advantage to this approach is the data within the MongoDB database does not need to be copied or moved to another system for analytics - there is no need for any additional ETL processes.

Quiz

Quiz

Which of the following are true for the MongoDB Connector for Business Intelligence? More than one answer choice can be correct.

- A. Can translate from MQL to SQL and vice-versa
- B. Requires some additional software (e.g ODBC driver) and has an overhead cost in terms of memory and of I/O
- C. Encrypts the data in transit
- D. Requires no additional ETL processes

Quiz

Which of the following are true for the MongoDB Connector for Business Intelligence? More than one answer choice can be correct.

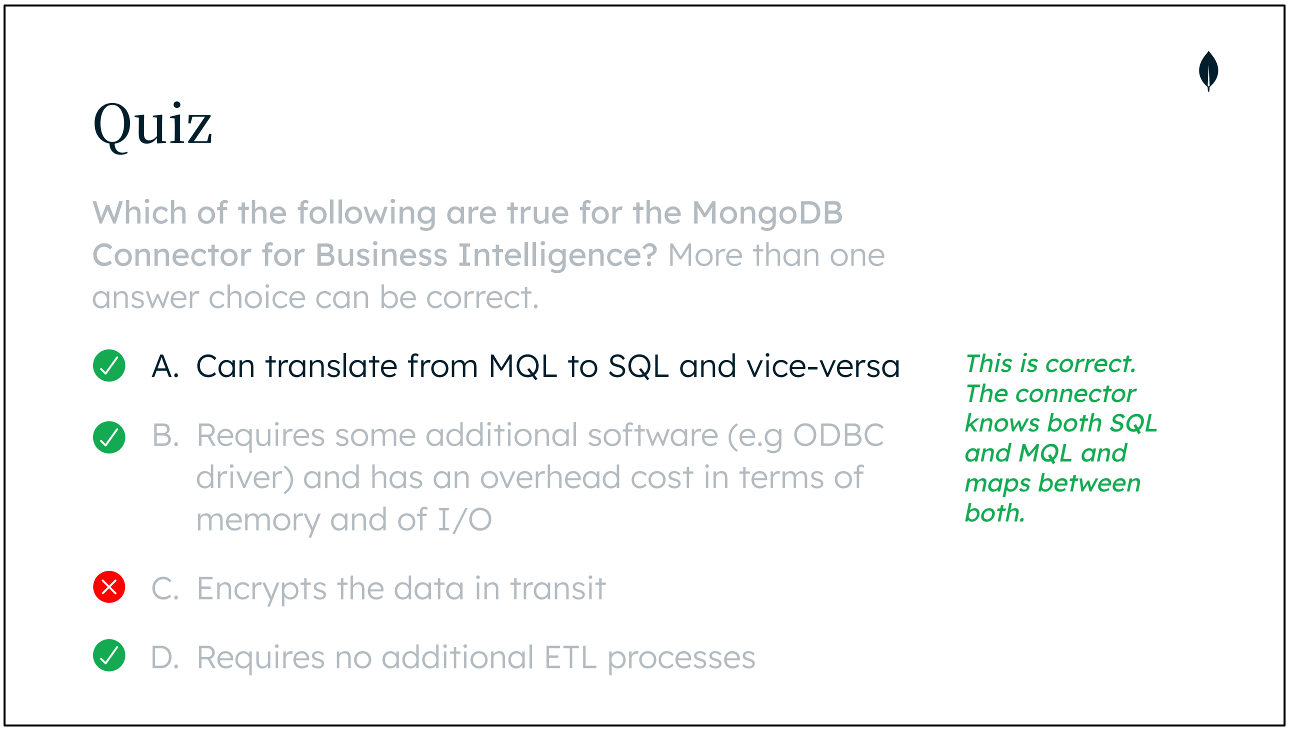
✓ A. Can translate from MQL to SQL and vice-versa

✓ B. Requires some additional software (e.g ODBC driver) and has an overhead cost in terms of memory and of I/O

✗ C. Encrypts the data in transit

✓ D. Requires no additional ETL processes

CORRECT: Can translate from MQL to SQL and vice-versa: This is correct, the connector knows both SQL and MQL and maps between both.
CORRECT: Requires some additional software (e.g ODBC driver) and has an overhead cost in terms of memory and of I/O. This is correct, there are several components required including the ODBC driver, a schema definition utility which maps documents in MongoDB to a relational schema, a daemon process, as well as an optional SQL to aggregation mapping utility.
INCORRECT: Encrypts the data in transit by default. This is incorrect as the BI connector does not enable TLS by default but this can be configured to provide security for data in transit.
CORRECT: Requires no additional ETL processes. This is correct as the queries from the BI application are translated as are the results from running the query on the MongoDB database. Only the query and the result are sent so an extraction loading process is not required to move the data to another datastore for analysis.



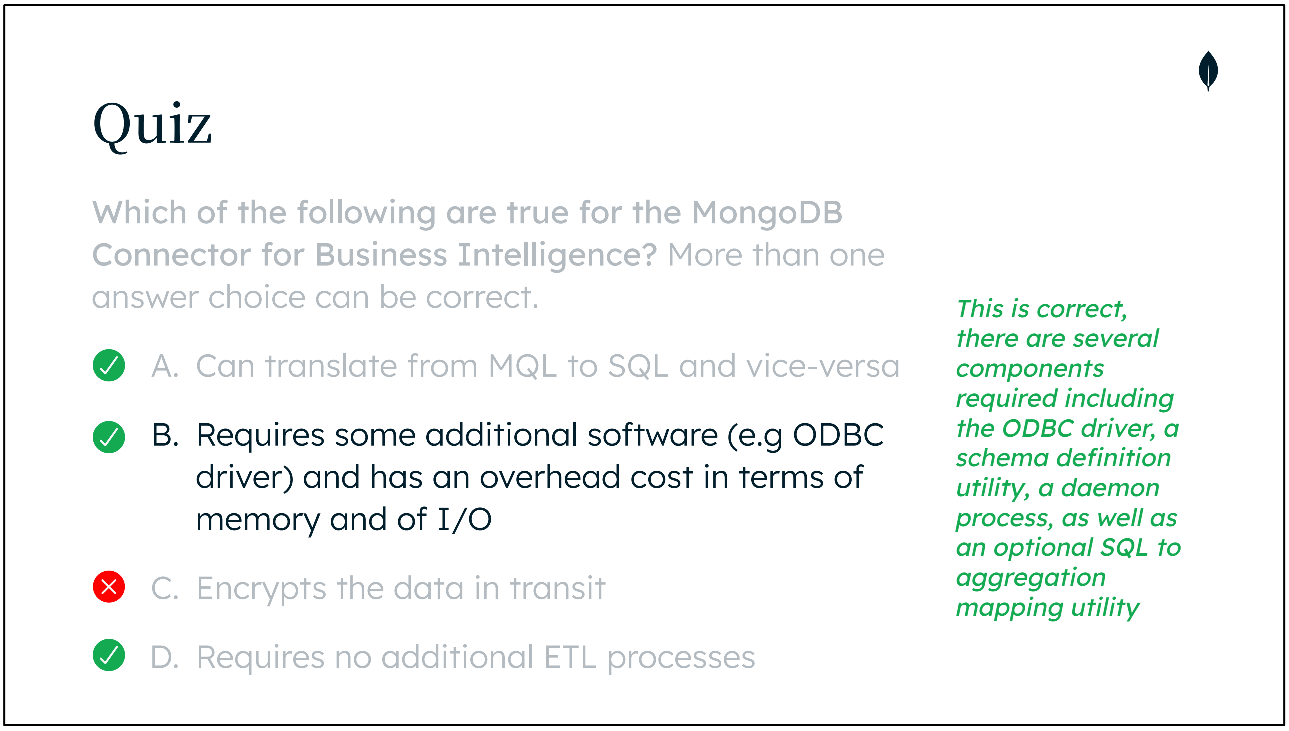
Quiz

Which of the following are true for the MongoDB Connector for Business Intelligence? More than one answer choice can be correct.

- ✓ A. Can translate from MQL to SQL and vice-versa
- ✓ B. Requires some additional software (e.g ODBC driver) and has an overhead cost in terms of memory and of I/O
- ✗ C. Encrypts the data in transit
- ✓ D. Requires no additional ETL processes

This is correct. The connector knows both SQL and MQL and maps between both.

CORRECT: Can translate from MQL to SQL and vice-versa: This is correct, the connector knows both SQL and MQL and maps between both.



Quiz

Which of the following are true for the MongoDB Connector for Business Intelligence? More than one answer choice can be correct.

✓ A. Can translate from MQL to SQL and vice-versa

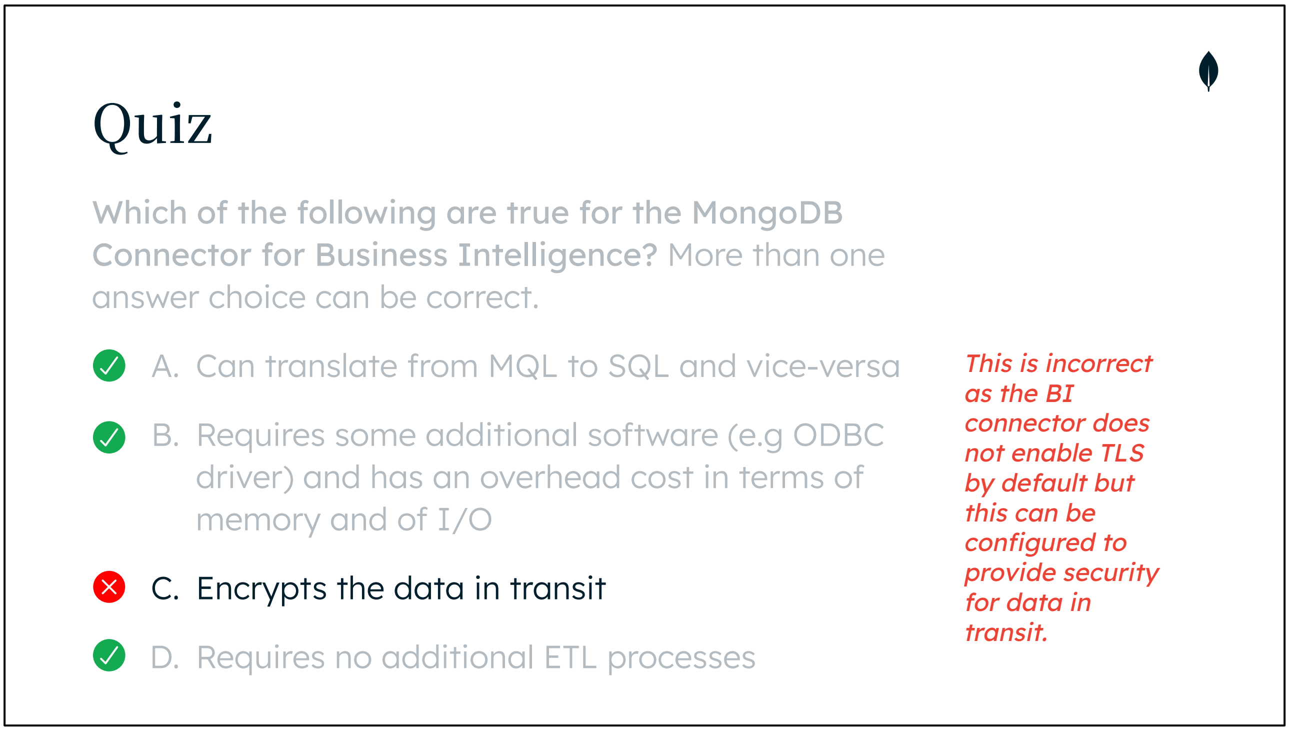
✓ B. Requires some additional software (e.g ODBC driver) and has an overhead cost in terms of memory and of I/O

✗ C. Encrypts the data in transit

✓ D. Requires no additional ETL processes

This is correct, there are several components required including the ODBC driver, a schema definition utility, a daemon process, as well as an optional SQL to aggregation mapping utility

CORRECT: Requires some additional software (e.g ODBC driver) and has an overhead cost in terms of memory and of I/O. This is correct, there are several components required including the ODBC driver, a schema definition utility which maps documents in MongoDB to a relational schema, a daemon process, as well as an optional SQL to aggregation mapping utility.



Quiz

Which of the following are true for the MongoDB Connector for Business Intelligence? More than one answer choice can be correct.

- ✓ A. Can translate from MQL to SQL and vice-versa
- ✓ B. Requires some additional software (e.g ODBC driver) and has an overhead cost in terms of memory and of I/O
- ✗ C. Encrypts the data in transit
- ✓ D. Requires no additional ETL processes

This is incorrect as the BI connector does not enable TLS by default but this can be configured to provide security for data in transit.

INCORRECT: Encrypts the data in transit by default. This is incorrect as the BI connector does not enable TLS by default but this can be configured to provide security for data in transit.

Quiz

Which of the following are true for the MongoDB Connector for Business Intelligence? More than one answer choice can be correct.

- ✓ A. Can translate from MQL to SQL and vice-versa
- ✓ B. Requires some additional software (e.g ODBC driver) and has an overhead cost in terms of memory and of I/O
- ✗ C. Encrypts the data in transit
- ✓ D. Requires no additional ETL processes

This is correct. Only the query and the result are sent so an extraction loading process is not required to move the data to another datastore for analysis.

CORRECT: Requires no additional ETL processes. This is correct as the queries from the BI application are translated as the results from running the query on the MongoDB database. Only the query and the result are sent so an extraction loading process is not required to move the data to another datastore for analysis.

Spark Connector

Let’s now explore the Spark connector for MongoDB.

Introduction to Spark

- Spark is an in-memory data processing engine.
- Based on Hadoop mapreduce.
- Spark has multiple libraries for machine learning, streaming, graph and other data processing algorithms.
- Spark supports over 100 different operators and algorithms for processing data.
- Can be scaled out across multiple compute nodes for distributed computing.

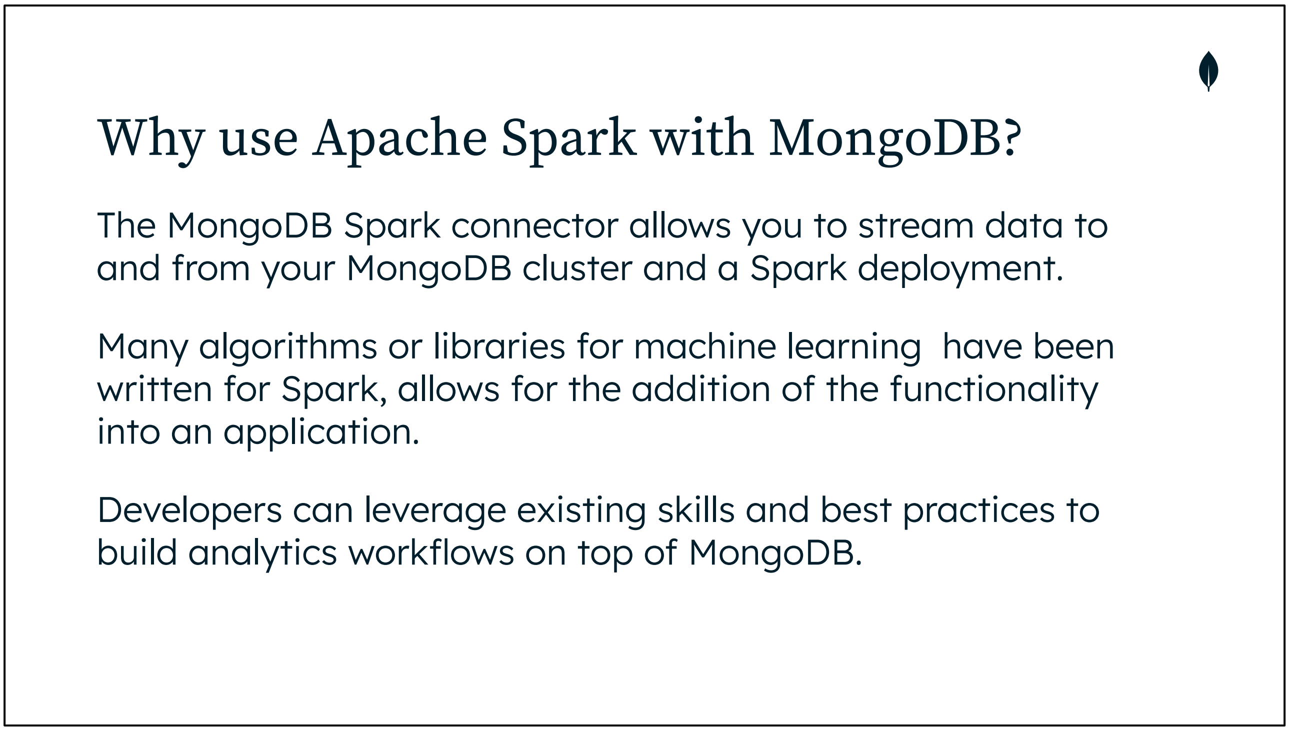
Spark is an in-memory data processing engine. It performs its calculations in memory as well as holding the data to be processed in-memory. This offers a number of performance improvements as information need not be saved or retrieved from disk, avoid costly (in terms of performance) I/O.

Spark is an evolution of the Hadoop mapreduce concept where calculations are broken down and split across many different nodes, each of which performs a subset of the calculation.

Spark is used extensively in the data analysis and machine learning communities. It provides multiple different libraries for machine learning, streaming, graph processing and other data processing tasks.

Spark has over 100 different operators and algorithms to process data.

It is designed to easily scale out over multiple compute nodes providing a simple mechanism for distributing the computation.



Why use Apache Spark with MongoDB?

The MongoDB Spark connector allows you to stream data to and from your MongoDB cluster and a Spark deployment.

Many algorithms or libraries for machine learning have been written for Spark, allows for the addition of the functionality into an application.

Developers can leverage existing skills and best practices to build analytics workflows on top of MongoDB.

The MongoDB Spark Connector allows for the streaming to and also from a MongoDB Cluster to a Spark deployment. This allows for applications to utilise the functionality in the Spark ecosystem within applications that are using MongoDB to store their data.

Spark is a very popular framework and the connector allows developers to leverage their knowledge of this framework to easily build analytics workflows atop of MongoDB.

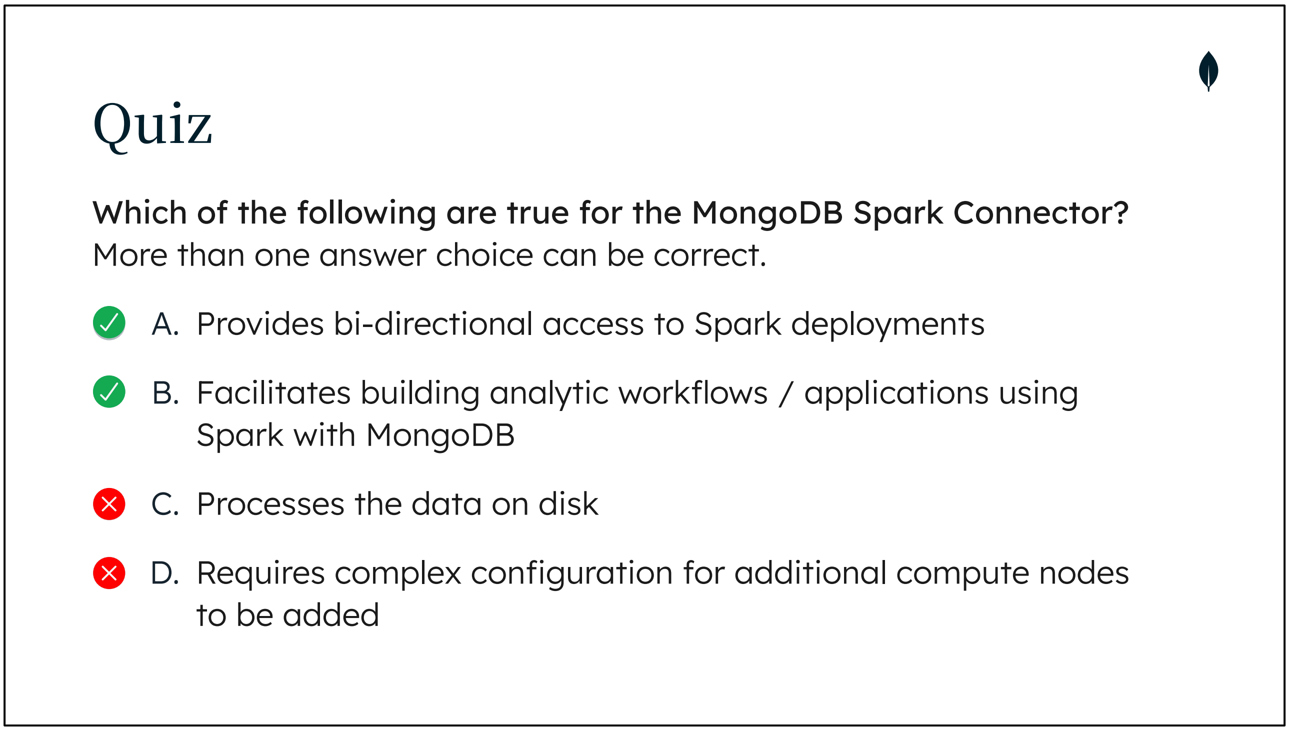
The Connector allows MongoDB to utilise the vast number of algorithms in Spark for data analysis and for machine learning.

Quiz

Quiz

Which of the following are true for the MongoDB Spark Connector?
More than one answer choice can be correct.

- A. Provides bi-directional access to Spark deployments
- B. Facilitates building analytic workflows / applications using Spark with MongoDB
- C. Processes the data on disk
- D. Requires complex configuration for additional compute nodes to be added



Quiz

Which of the following are true for the MongoDB Spark Connector?
More than one answer choice can be correct.

- ✓ A. Provides bi-directional access to Spark deployments
- ✓ B. Facilitates building analytic workflows / applications using Spark with MongoDB
- ✗ C. Processes the data on disk
- ✗ D. Requires complex configuration for additional compute nodes to be added

CORRECT: Provides bi-directional access to Spark deployments. This is correct, the MongoDB Spark connector allows for data to be sent to and from a MongoDB deployment to a Spark deployment.
CORRECT: Facilitates building analytic workflows / applications using Spark with MongoDB. This is correct, Spark offers a number of machine learning and data processing libraries and the MongoDB Spark Connector makes it easy to use these to add analytics capabilities to a MongoDB application.
INCORRECT: Processes the data on disk. This is incorrect, Spark use memory to perform its calculations. The results can be stored to disk but it's high performance when compared to systems like Hadoop is its focus on in-memory data processing.
INCORRECT: Requires complex configuration for additional compute nodes to be added. This is incorrect. Spark does not require complex configuration to easily add or remove compute nodes to a deployment.

Quiz

Which of the following are true for the MongoDB Spark Connector? More than one answer choice can be correct.

✓ A. Provides bi-directional access to Spark deployments

✓ B. Facilitates building analytic workflows / applications using Spark with MongoDB

✗ C. Processes the data on disk

✗ D. Requires complex configuration for additional compute nodes to be added

This is correct. The MongoDB Spark connector allows for data to be sent to and from a MongoDB deployment to a Spark deployment

CORRECT: Provides bi-directional access to Spark deployments. This is correct, the MongoDB Spark connector allows for data to be sent to and from a MongoDB deployment to a Spark deployment.

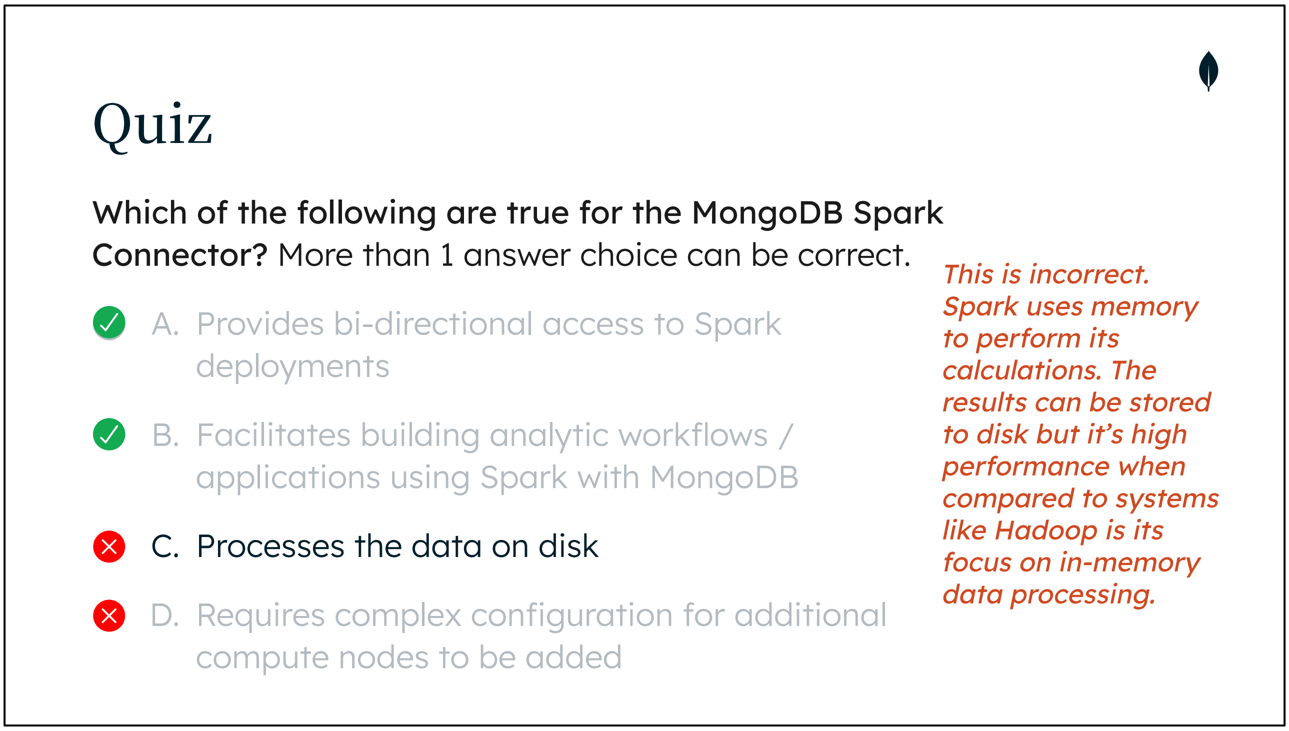
Quiz

Which of the following are true for the MongoDB Spark Connector? More than 1 answer choice can be correct.

- ✓ A. Provides bi-directional access to Spark deployments
- ✓ B. Facilitates building analytic workflows / applications using Spark with MongoDB
- ✗ C. Processes the data on disk
- ✗ D. Requires complex configuration for additional compute nodes to be added

This is correct. Spark offers a number of machine learning and data processing libraries and the MongoDB Spark Connector makes it easy to use these to add analytics capabilities to a MongoDB application.

CORRECT: Facilitates building analytic workflows / applications using Spark with MongoDB. This is correct. Spark offers a number of machine learning and data processing libraries and the MongoDB Spark Connector makes it easy to use these to add analytics capabilities to a MongoDB application.



Quiz

Which of the following are true for the MongoDB Spark Connector? More than 1 answer choice can be correct.

✓ A. Provides bi-directional access to Spark deployments

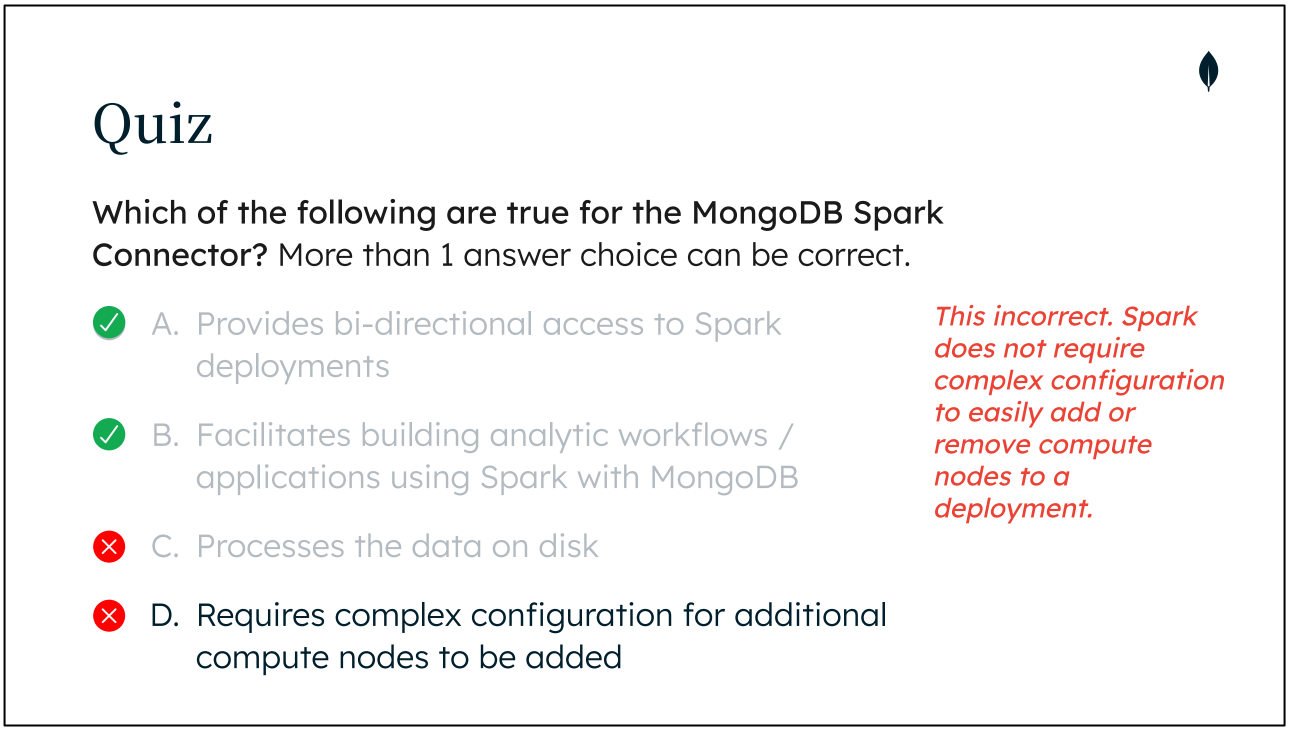
✓ B. Facilitates building analytic workflows / applications using Spark with MongoDB

✗ C. Processes the data on disk

✗ D. Requires complex configuration for additional compute nodes to be added

This is incorrect. Spark uses memory to perform its calculations. The results can be stored to disk but it's high performance when compared to systems like Hadoop is its focus on in-memory data processing.

INCORRECT: Processes the data on disk. This is incorrect. Spark uses memory to perform its calculations. The results can be stored to disk but it's high performance when compared to systems like Hadoop is its focus on in-memory data processing.



Quiz

Which of the following are true for the MongoDB Spark Connector? More than 1 answer choice can be correct.

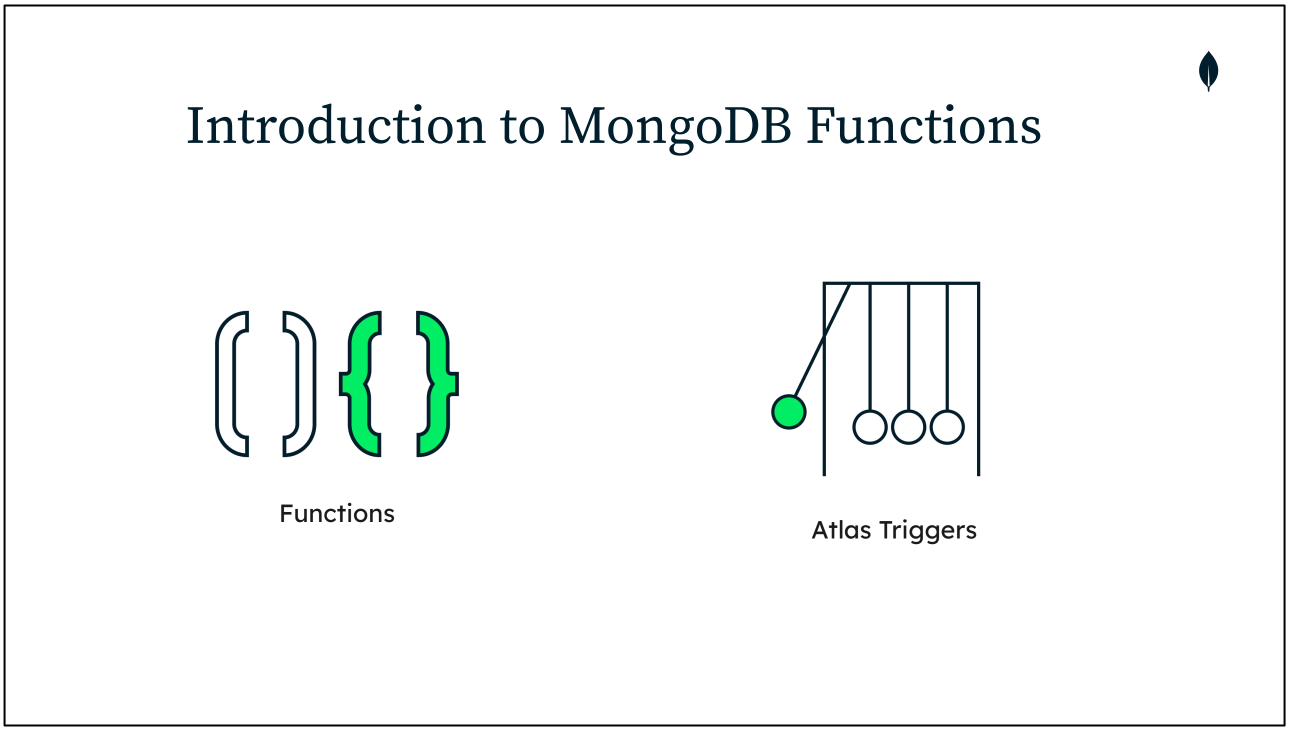
- ✓ A. Provides bi-directional access to Spark deployments
- ✓ B. Facilitates building analytic workflows / applications using Spark with MongoDB
- ✗ C. Processes the data on disk
- ✗ D. Requires complex configuration for additional compute nodes to be added

This incorrect. Spark does not require complex configuration to easily add or remove compute nodes to a deployment.

INCORRECT: Requires complex configuration for additional compute nodes to be added. This is incorrect. Spark does not require complex configuration to easily add or remove compute nodes to a deployment.

MongoDB Functions

MongoDB Functions are another form of connector that provides additional functionality within the MongoDB Atlas platform.



Introduction to MongoDB Functions

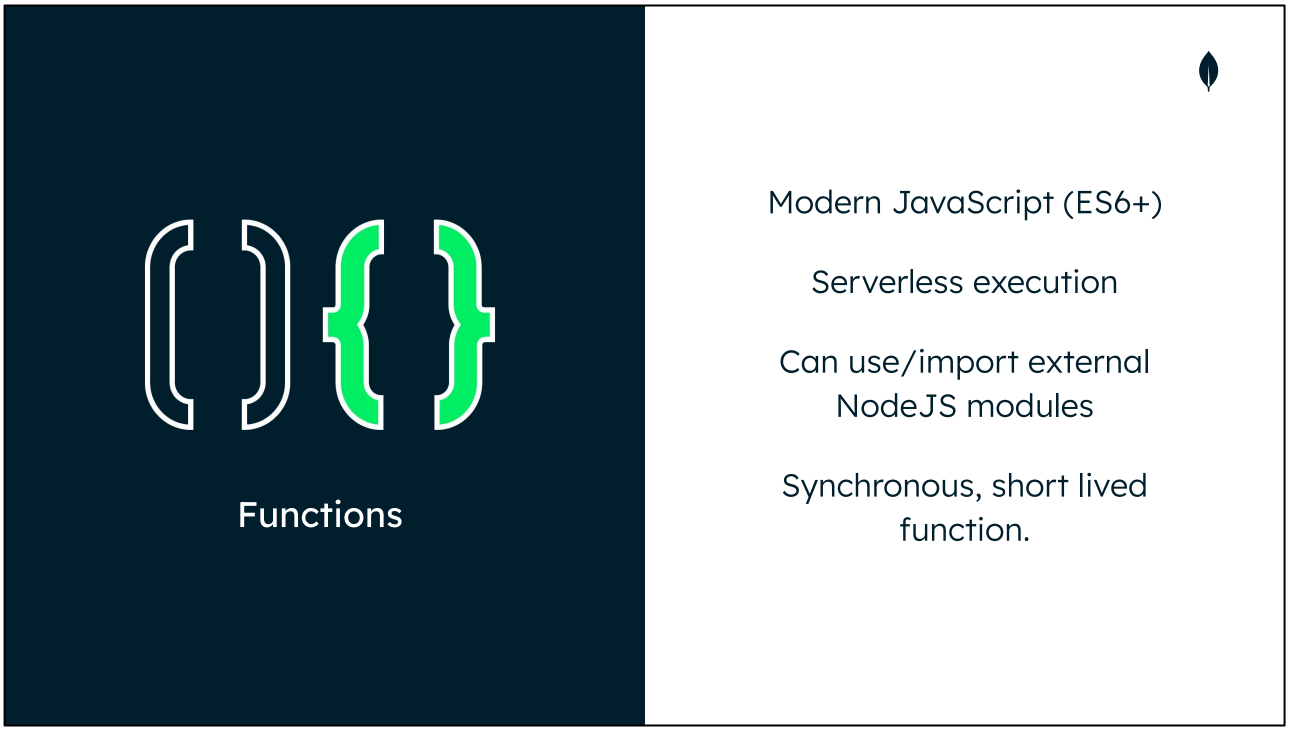
Functions

Atlas Triggers

MongoDB Functions in Atlas allow for the definition and ability to execute server-side logic for your application within Atlas. You can call functions from your client applications as well as from other functions and in JSON expressions.

Atlas Triggers allow you to define server-side logic in Atlas that will execute in response to a database event or on a specified schedule.

Let's look a little more at each of these to better understand what they can provide in terms of functionality.



Functions

Modern JavaScript (ES6+)

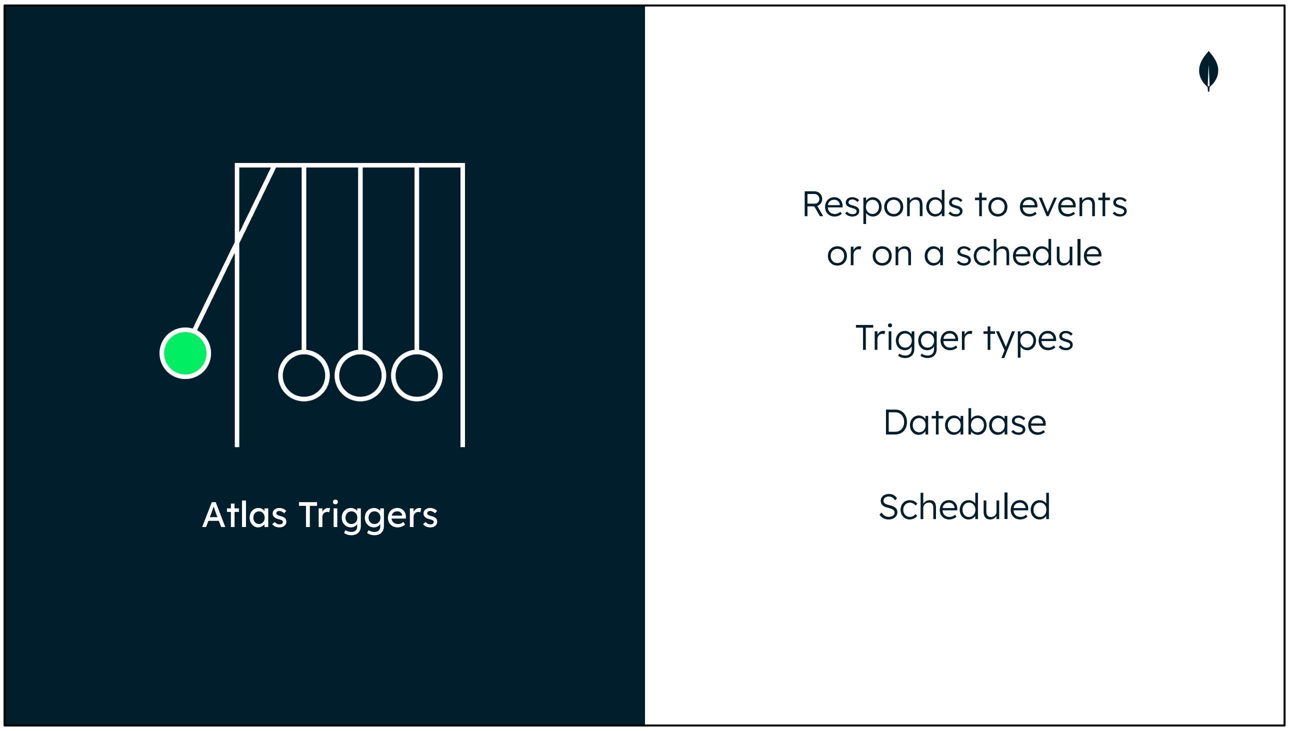
Serverless execution

Can use/import external NodeJS modules

Synchronous, short lived function.

Functions provide a means to add server side logic implemented in ES6+ Javascript which is executed in a serverless fashion (on-demand). These functions can further use the wide variety of NodeJS modules to add specific functionality to the server side logic.

Functions can call other functions and include a built-in client for working with data in MongoDB Atlas clusters. They also include helpful global utilities, support common Node.js built-in modules, and can import and use external packages from the npm registry.



Atlas Triggers

Responds to events
or on a schedule

Trigger types

Database

Scheduled

We’ve previously covered Atlas Triggers in our lesson describing Atlas, however it is worth recapping them here.
Triggers provide server-side code that can respond to database events or according to a schedule.

Atlas has two kinds of triggers, database and scheduled.

A database trigger will execute server-side whenever a document is added, updated or removed in a cluster that is linked.
Database triggers use MongoDB change streams to listen for changes in watched collections and map these to database events. We’ll cover MongoDB change streams in more depth in a later lesson.

Scheduled triggers are very similar to traditional Unix/Linux CRON jobs, these are defined using the CRON temporal expressions to indicate when they should run. A CRON expression consists of five space-delimited fields minute, hour, dayOfMonth, month, weekDay. These can be set to a specific value or to an expression that evaluates to a set of values. Overall, these are quite similar to CRON jobs in Unix/Linux and this was a deliberate design choice to simplify the use of this feature but using the well-established CRON format and syntax.

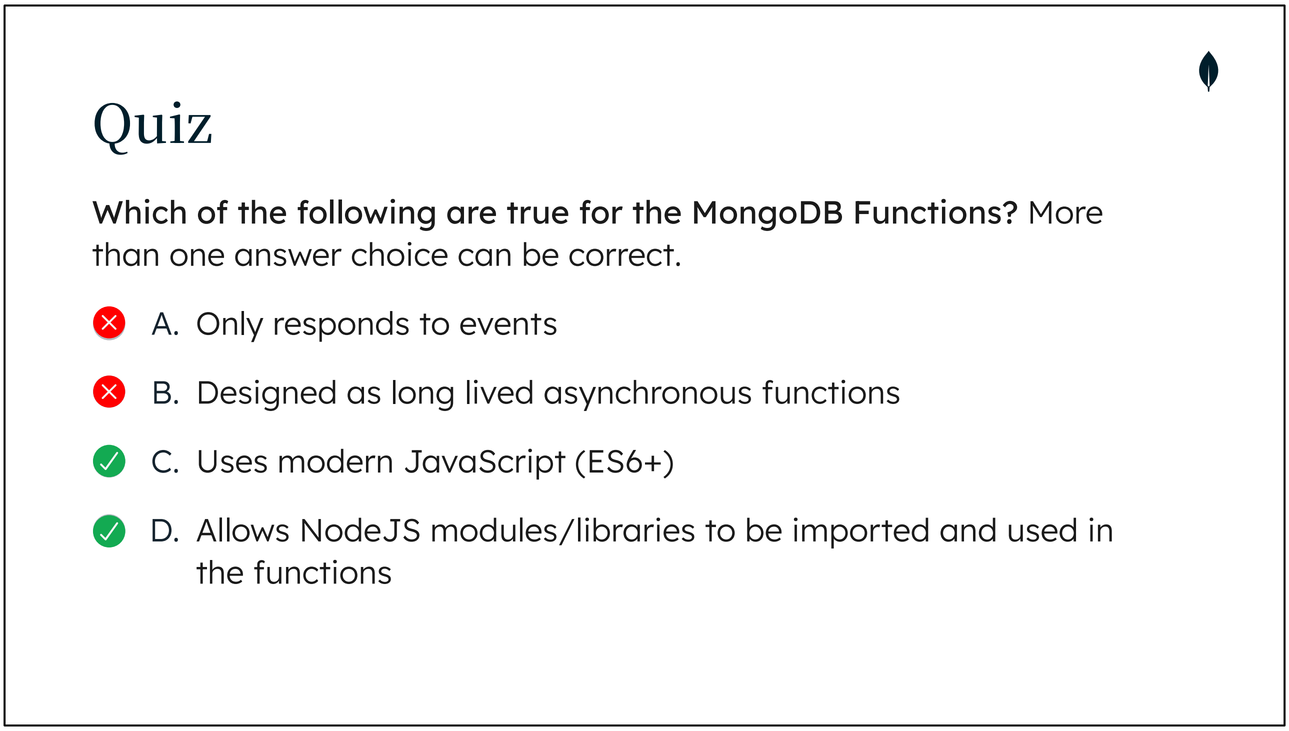
The schedule trigger will then execute the server-side logic on that CRON schedule. This type of trigger functionality is suited to periodic work, such as nightly reporting, updating documents every minute/hour, or indeed generating weekly newsletters or offer emails.

Quiz

Quiz

Which of the following are true for the MongoDB Functions? More than one answer choice can be correct.

- A. Only responds to events
- B. Designed as long lived asynchronous functions
- C. Uses modern JavaScript (ES6+)
- D. Allows NodeJS modules/libraries to be imported and used in the functions

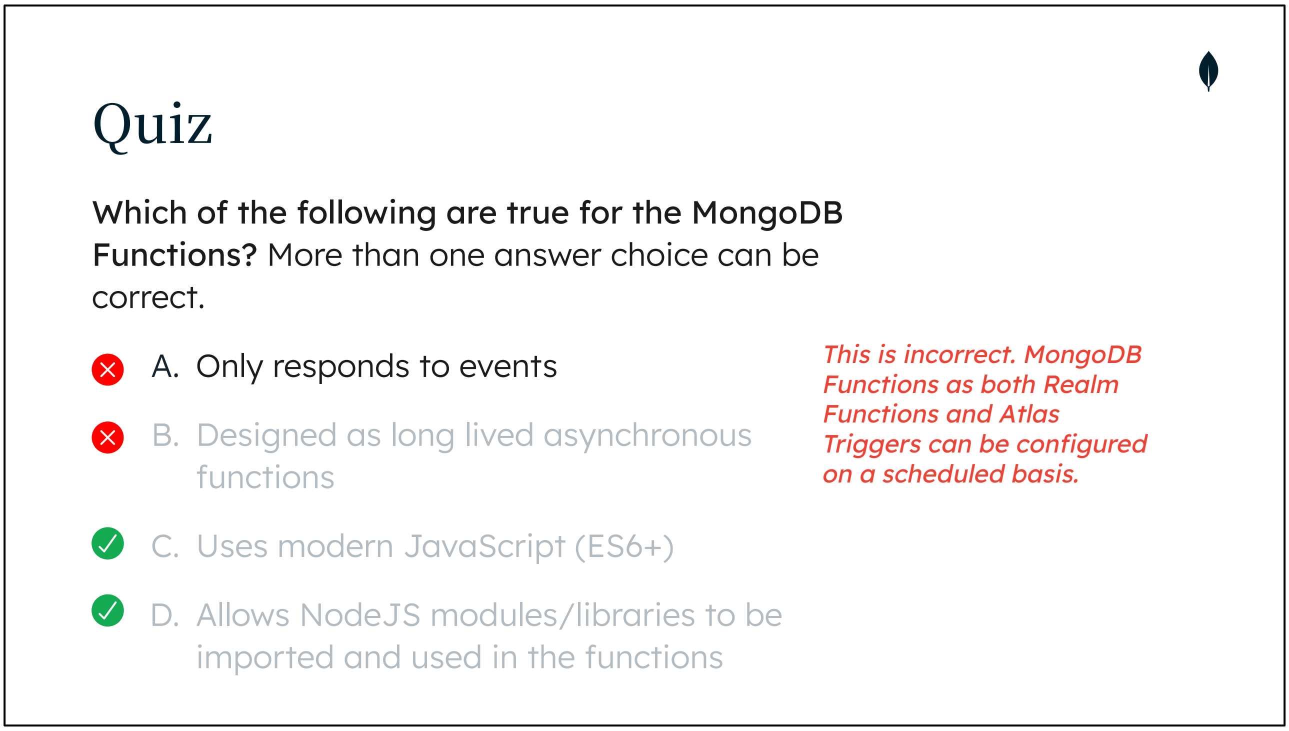


Quiz

Which of the following are true for the MongoDB Functions? More than one answer choice can be correct.

- ✗ A. Only responds to events
- ✗ B. Designed as long lived asynchronous functions
- ✓ C. Uses modern JavaScript (ES6+)
- ✓ D. Allows NodeJS modules/libraries to be imported and used in the functions

INCORRECT: Only responds to events. This is incorrect as MongoDB Functions as both Realm Functions and Atlas Triggers can be configured on a scheduled basis.
INCORRECT: Designed as long lived asynchronous functions. This is incorrect as MongoDB Functions are designed as short lived synchronous functions.
CORRECT: Uses modern JavaScript (ES6+). This is correct. MongoDB Functions, specifically Realm Functions uses modern Javascript.
CORRECT: Allows NodeJS modules/libraries to be imported and used in the functions. This is correct. MongoDB Functions, specifically Realm Functions allows developers to include existing third-party NodeJS modules/libraries to be imported and used.



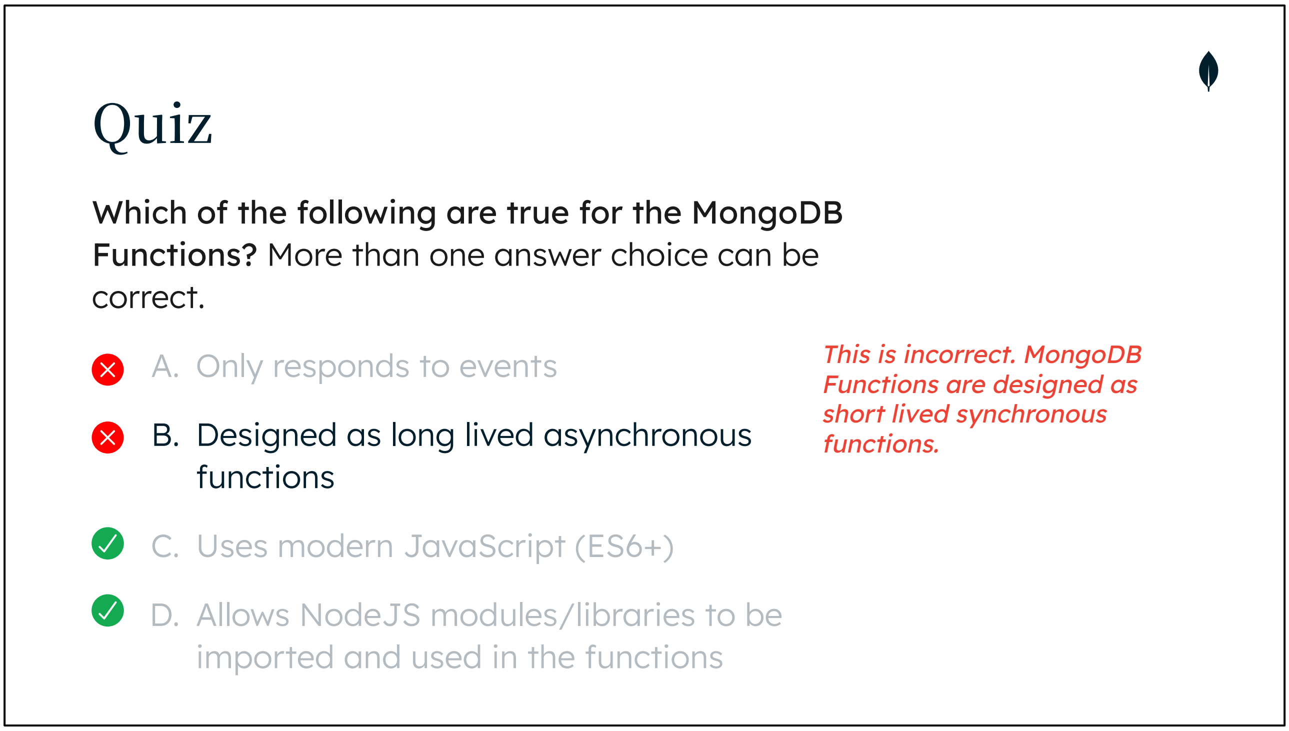
Quiz

Which of the following are true for the MongoDB Functions? More than one answer choice can be correct.

- ✗ A. Only responds to events
- ✗ B. Designed as long lived asynchronous functions
- ✓ C. Uses modern JavaScript (ES6+)
- ✓ D. Allows NodeJS modules/libraries to be imported and used in the functions

This is incorrect. MongoDB Functions as both Realm Functions and Atlas Triggers can be configured on a scheduled basis.

INCORRECT: Only responds to events. This is incorrect as MongoDB Functions as both Realm Functions and Atlas Triggers can be configured on a scheduled basis.



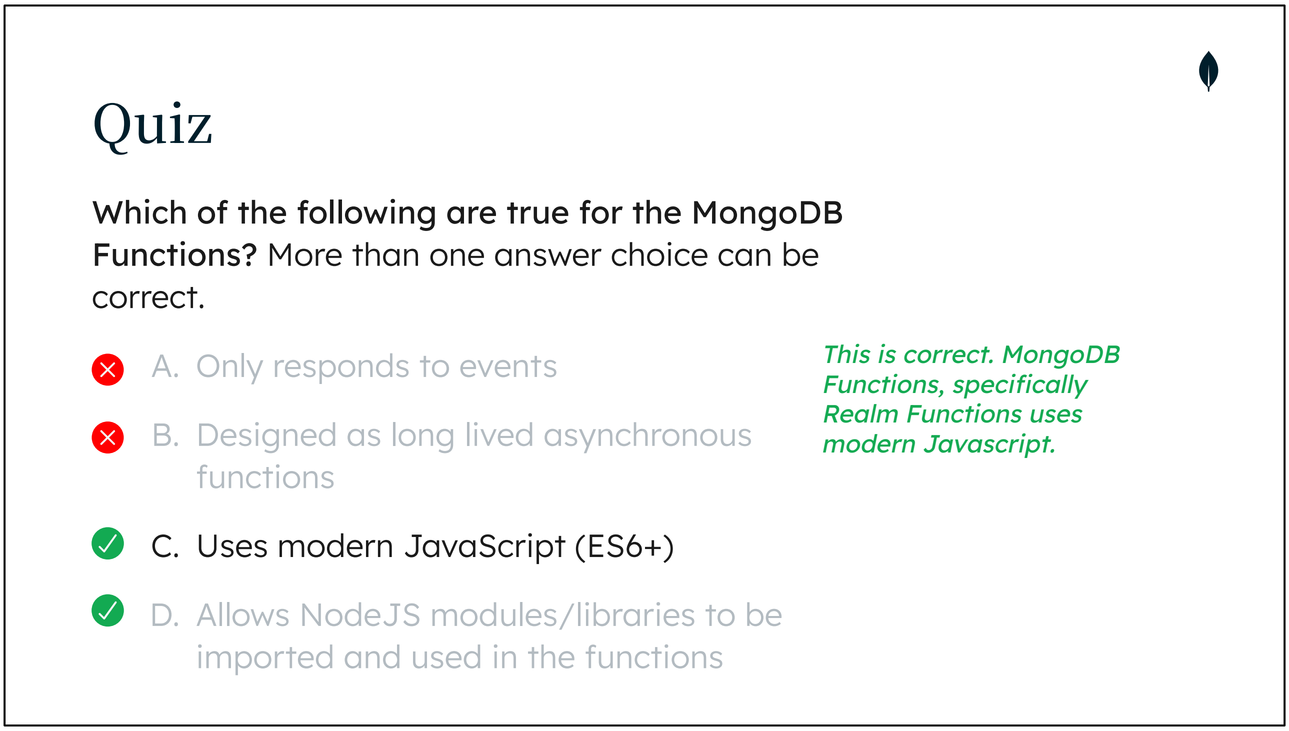
Quiz

Which of the following are true for the MongoDB Functions? More than one answer choice can be correct.

- ✗ A. Only responds to events
- ✗ B. Designed as long lived asynchronous functions
- ✓ C. Uses modern JavaScript (ES6+)
- ✓ D. Allows NodeJS modules/libraries to be imported and used in the functions

This is incorrect. MongoDB Functions are designed as short lived synchronous functions.

INCORRECT: Designed as long lived asynchronous functions. This is incorrect as MongoDB Functions are designed as short lived synchronous functions.



Quiz

Which of the following are true for the MongoDB Functions? More than one answer choice can be correct.

- ✗ A. Only responds to events
- ✗ B. Designed as long lived asynchronous functions
- ✓ C. Uses modern JavaScript (ES6+)
- ✓ D. Allows NodeJS modules/libraries to be imported and used in the functions

This is correct. MongoDB Functions, specifically Realm Functions uses modern Javascript.

CORRECT: Uses modern JavaScript (ES6+). This is correct. MongoDB Functions, specifically Realm Functions uses modern Javascript.

Quiz

Which of the following are true for the MongoDB Functions? More than one answer choice can be correct.

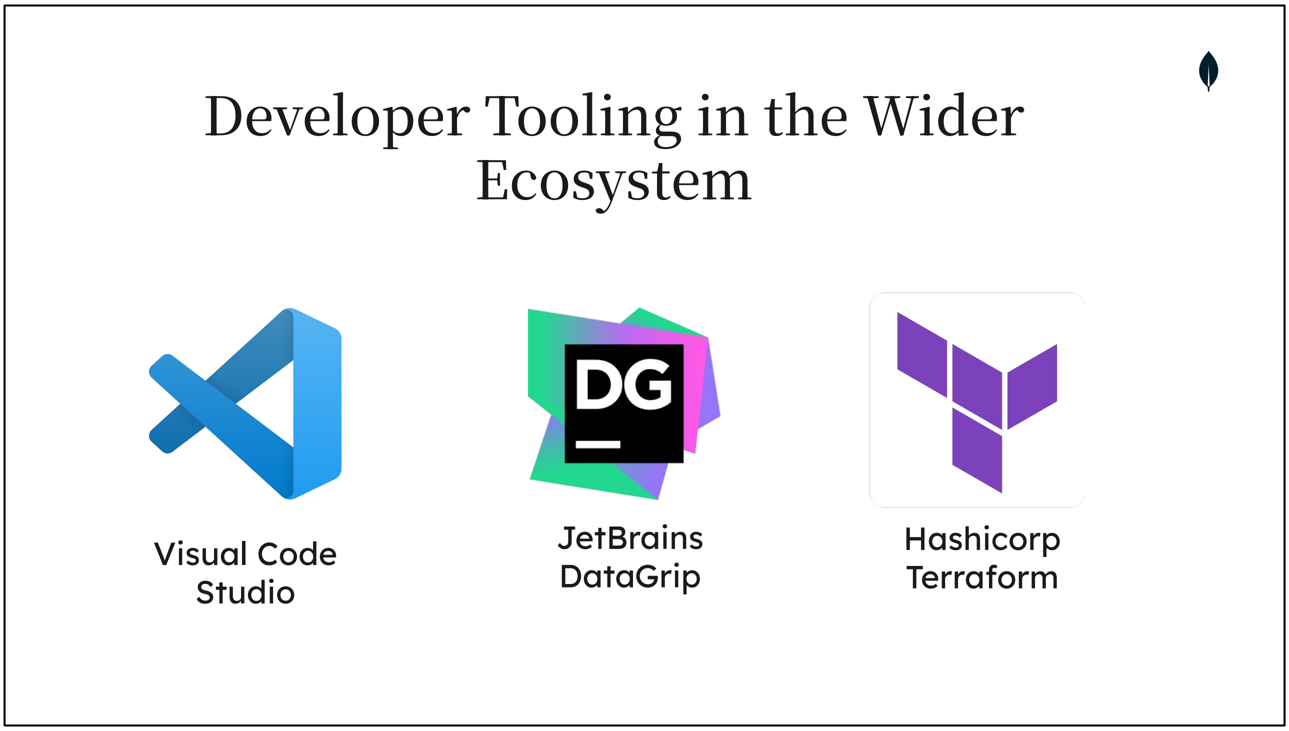
- ✗ A. Only responds to events
- ✗ B. Designed as long lived asynchronous functions
- ✓ C. Uses modern JavaScript (ES6+)
- ✓ D. Allows NodeJS modules/libraries to be imported and used in the functions

This is correct. MongoDB Functions allows developers to include existing third-party NodeJS modules/libraries to be imported and used.

CORRECT: Allows NodeJS modules/libraries to be imported and used in the functions. This is correct. MongoDB Functions allow developers to include existing third-party NodeJS modules/libraries to be imported and used.

The Wider Ecosystem

We’ll finish this lesson with a few examples of developer tooling from the wider ecosystem.



Developer Tooling in the Wider Ecosystem

Visual Code Studio

JetBrains DataGrip

Hashicorp Terraform

In the wider tooling ecosystem, MongoDB has worked with a number of companies to provide or co-develop tooling related to MongoDB.

Visual Code Studio is a good example of where MongoDB built an extension that allows you to use your IDE to build MongoDB Application and easily connect to MongoDB Atlas. It also provides a feature, MongoDB Playgrounds which allow for CRUD operations and MongoDB commands to be executed within the IDE. This allows you to prototype queries and aggregations within the IDE, it further allows you to easily share these. The plugin also links to Hashicorp Terraform plugin that we'll cover in a few moments.

Another useful tool is JetBrains DataGrip, which provides a IDE focused on working with databases. It allows you to explore your collection and documents via the IDE.

The last tool we'll cover in this lesson is Hashicorp's Terraform. We've covered this briefly in the MongoDB Atlas lesson, to briefly recap Terraform is a provisioning tool rather than a configuration management tool..

The full description and details on the MongoDB Atlas Terraform Provider can be found at the website on the slide https://www.mongodb.com/atlas/hashicorp-terraform

A provider in Terraform is used to create, manage, and update infrastructure resources. It is able to understand both Terraform's API and the provider's APIs (in this case MongoDB Atlas's API) and expose the provider's resources. Terraform allows you to describe your infrastructure using the declarative configuration, it can be versioned and treated in a similar fashion to how you manage software source code. It also allows these configurations to be easily shared and re-used.

Quiz

Quiz

Which of the following applications did MongoDB collaborate to develop functionality for? More than one answer choice can be correct.

- A. Microsoft's Visual Code Studio
- B. JetBrains PyCharm
- C. Studio 3T MongoDB interface
- D. Hashicorp Terraform MongoDB plugin

Quiz

Which of the following applications did MongoDB collaborate to develop functionality for? More than one answer choice can be correct.

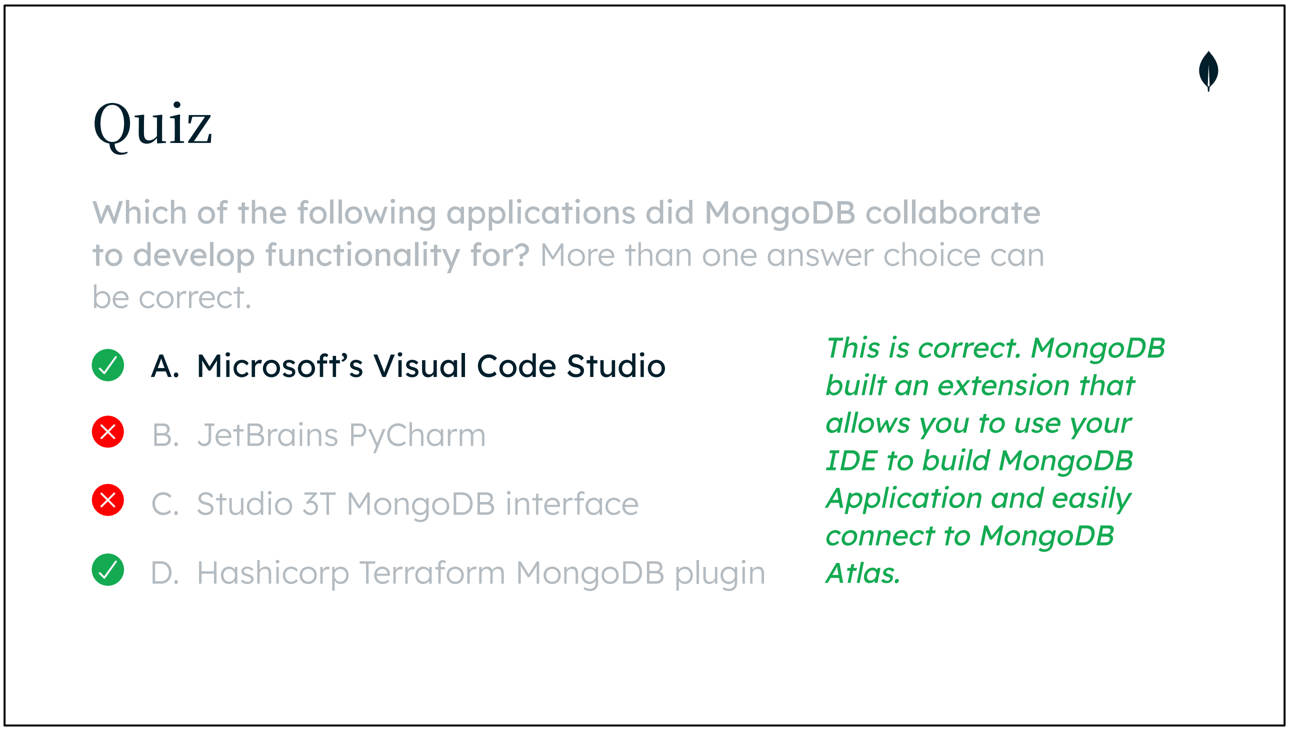
✓ A. Microsoft's Visual Code Studio

✗ B. JetBrains PyCharm

✗ C. Studio 3T MongoDB interface

✓ D. Hashicorp Terraform MongoDB plugin

CORRECT: Microsoft Visual Code Studio. This is correct. MongoDB built an extension that allows you to use your IDE to build MongoDB Application and easily connect to MongoDB Atlas. It also built MongoDB Playgrounds which allow for CRUD operations and MongoDB commands to be executed within the IDE.
INCORRECT: JetBrains PyCharm. This incorrect. JetBrains did built MongoDB functionality into this IDE but they did so without assistance or collaboration from MongoDB. This work did lead to a later collaboration for JetBrains DataGrip product where collaboration did happen between both companies.
INCORRECT: Studio 3T MongoDB interface. This is incorrect. MongoDB did not collaborate with the develop of the 3T interface, it offers a competing product called MongoDB Compass.
CORRECT: Hashicorp Terraform MongoDB Atlas plugin. This is correct. MongoDB worked with Hashicorp to create a MongoDB plugin to allow Terraform/infrastructure as code to be used to deploy MongoDB Atlas clusters.



Quiz

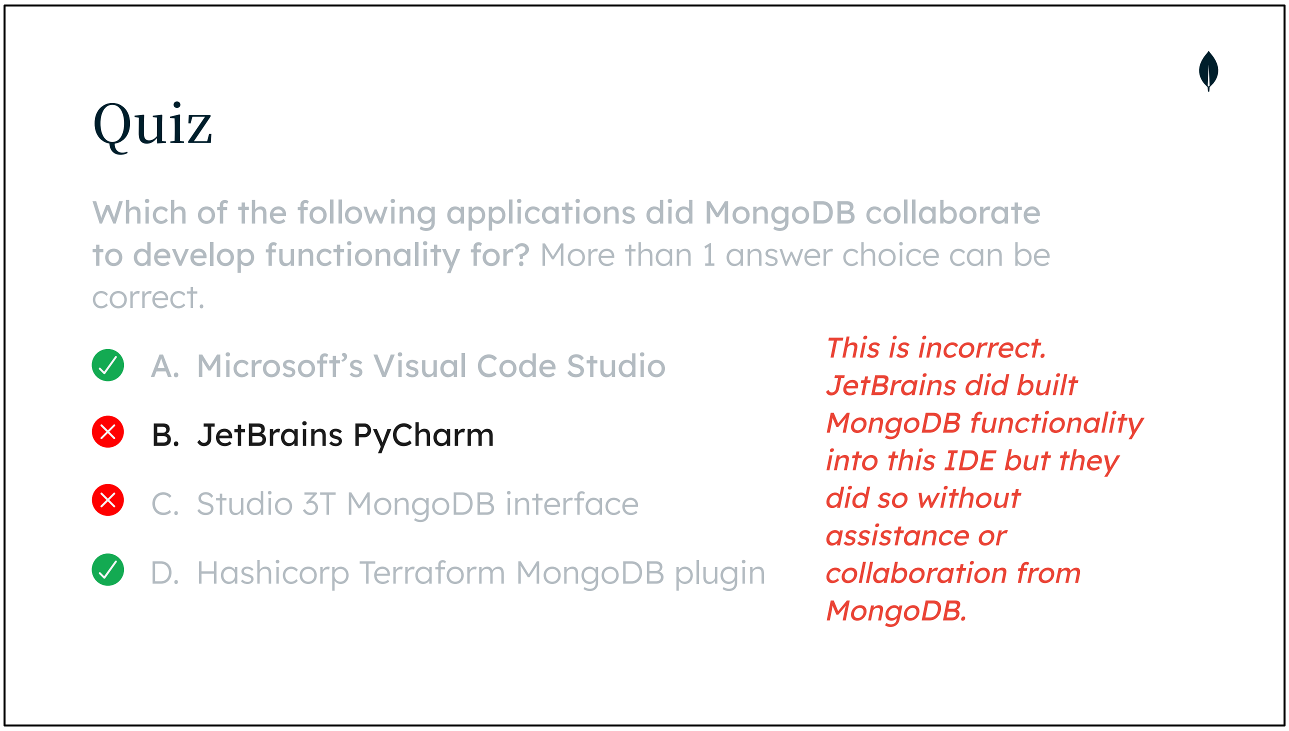
Which of the following applications did MongoDB collaborate to develop functionality for? More than one answer choice can be correct.

- ✓ A. **Microsoft's Visual Code Studio**
- ✗ B. JetBrains PyCharm
- ✗ C. Studio 3T MongoDB interface
- ✓ D. Hashicorp Terraform MongoDB plugin

This is correct. MongoDB built an extension that allows you to use your IDE to build MongoDB Application and easily connect to MongoDB Atlas.

CORRECT: Microsoft Visual Code Studio. This is correct. MongoDB built an extension that allows you to use your IDE to build MongoDB Application and easily connect to MongoDB Atlas. It also built MongoDB Playgrounds which allow for CRUD operations and MongoDB commands to be executed within the IDE.

Note: It also built MongoDB Playgrounds which allow for CRUD operations and MongoDB commands to be executed within the IDE.



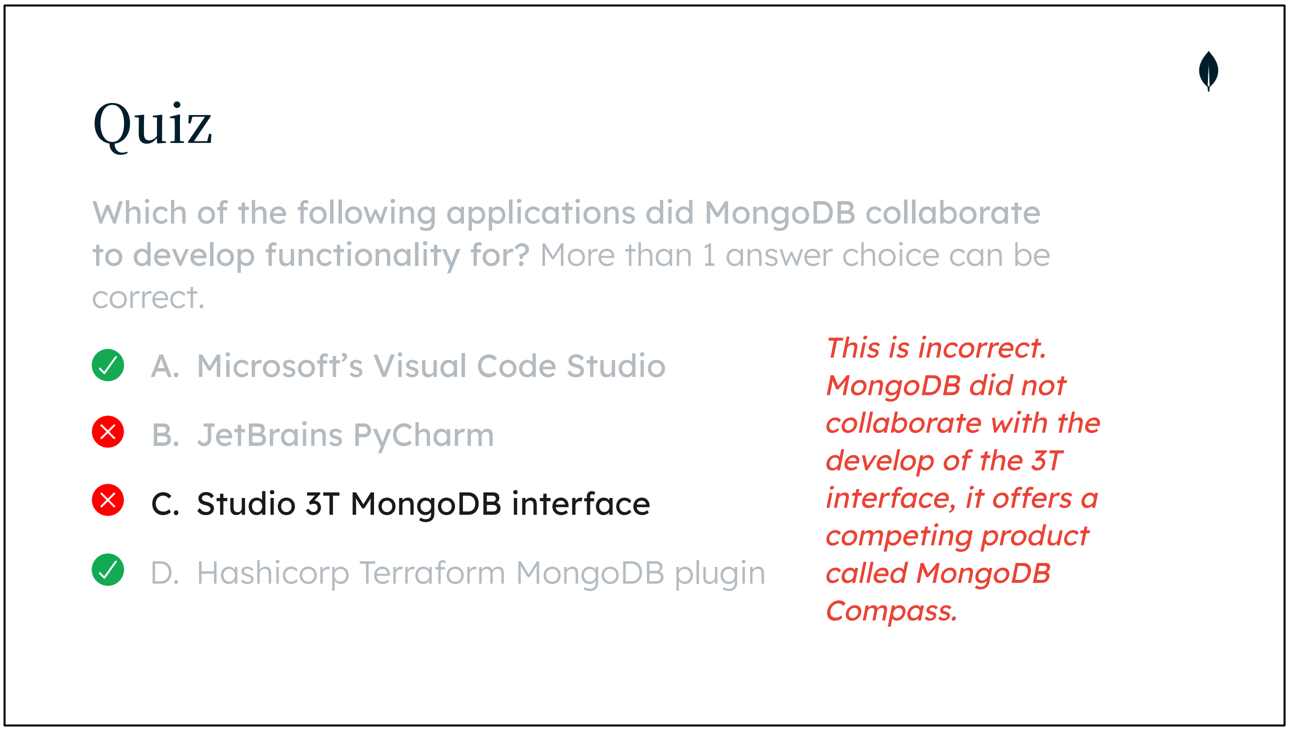
Quiz

Which of the following applications did MongoDB collaborate to develop functionality for? More than 1 answer choice can be correct.

- ✓ A. Microsoft's Visual Code Studio
- ✗ B. JetBrains PyCharm
- ✗ C. Studio 3T MongoDB interface
- ✓ D. Hashicorp Terraform MongoDB plugin

This is incorrect. JetBrains did built MongoDB functionality into this IDE but they did so without assistance or collaboration from MongoDB.

INCORRECT: JetBrains PyCharm. This incorrect. JetBrains did built MongoDB functionality into this IDE but they did so without assistance or collaboration from MongoDB. This work did lead to a later collaboration for JetBrains DataGrip product where collaboration did happen between both companies.



Quiz

Which of the following applications did MongoDB collaborate to develop functionality for? More than 1 answer choice can be correct.

- ✓ A. Microsoft's Visual Code Studio
- ✗ B. JetBrains PyCharm
- ✗ C. **Studio 3T MongoDB interface**
- ✓ D. Hashicorp Terraform MongoDB plugin

This is incorrect. MongoDB did not collaborate with the develop of the 3T interface, it offers a competing product called MongoDB Compass.

INCORRECT: Studio 3T MongoDB interface. This is incorrect. MongoDB did not collaborate with the develop of the 3T interface, it offers a competing product called MongoDB Compass.

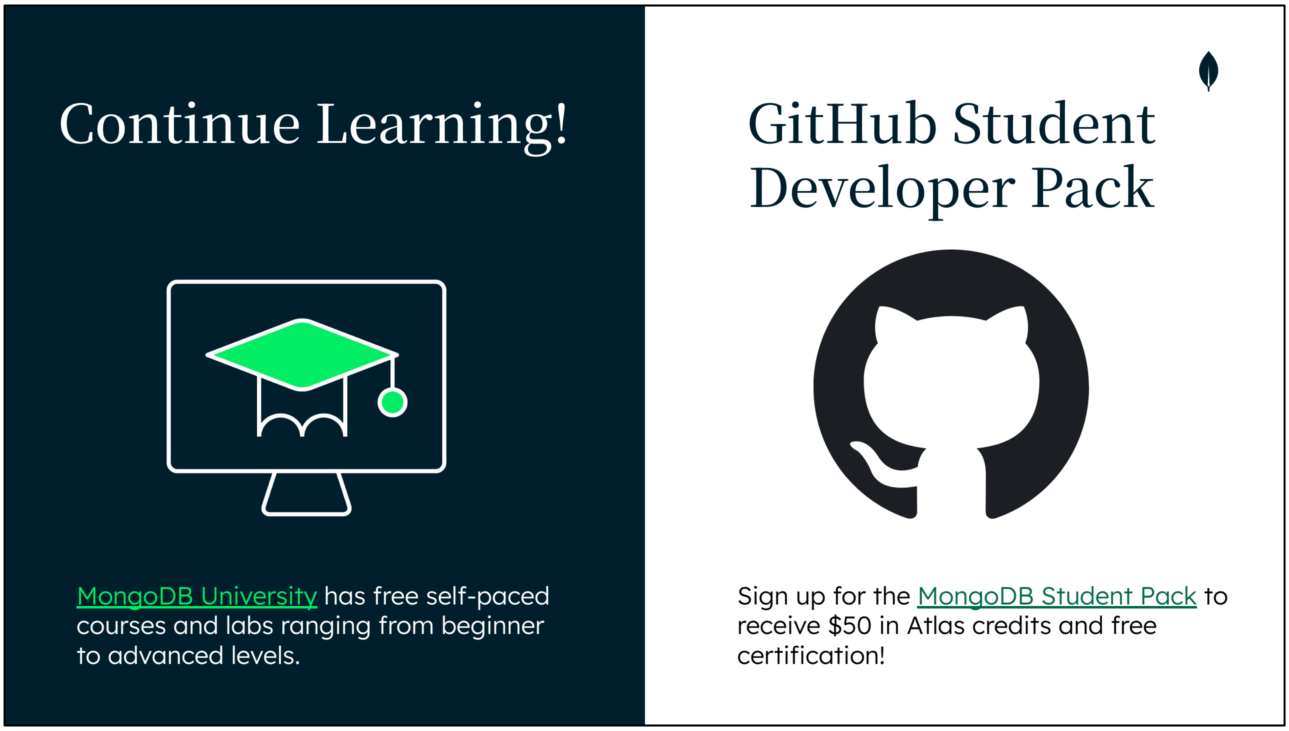
Quiz

Which of the following applications did MongoDB collaborate to develop functionality for? More than 1 answer choice can be correct.

- ✓ A. Microsoft’s Visual Code Studio
- ✗ B. JetBrains PyCharm
- ✗ C. Studio 3T MongoDB interface
- ✓ D. Hashicorp Terraform MongoDB plugin

This is correct. MongoDB worked with Hashicorp to create a MongoDB plugin to allow Terraform/infrastructure as code to be used to deploy MongoDB Atlas clusters.

CORRECT: Hashicorp Terraform MongoDB Atlas plugin. This is correct. MongoDB worked with Hashicorp to create a MongoDB plugin to allow Terraform/infrastructure as code to be used to deploy MongoDB Atlas clusters.



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.